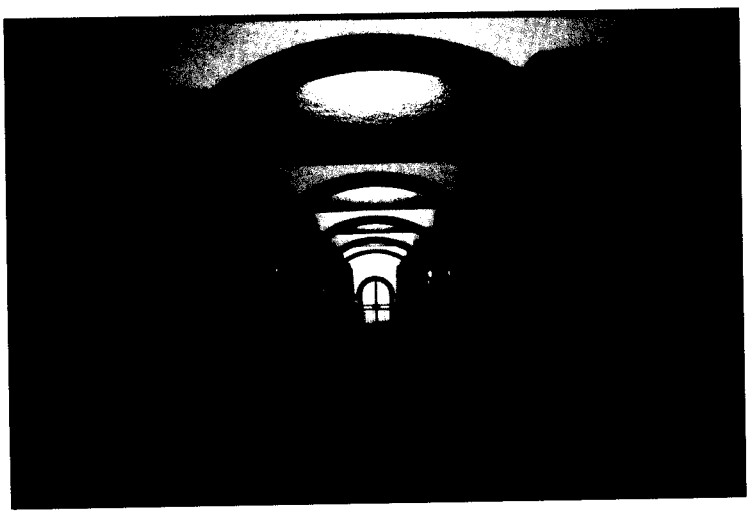
Системы хранения данных в Windows

Серверные технологии хранения данных в среде Windows® 2000 и Windows® Server 2003



Дайлин Наик

Inside Windows Storage

Server Storage Technologies for Windows® 2000, Windows® Server 2003, and Beyond

Dilip C. Naik

ADDISON-WESLEY

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Системы хранения данных в Windows

Системы хранения данных в Windows

Серверные технологии хранения данных в среде Windows® 2000 и Windows® Server 2003

Дайлип Наик

Москва • Санкт-Петербург • Киев
2005

ББК 32.973.26-018.2.75
Н20
УДК 681.3.07

Издательский дом "Вильямс"

Главный редактор *С.Н. Тригуб*

Зав. редакцией *В.Р. Гинзбург*

Перевод с английского *О.А. Лещинского*

Под редакцией *А.Н. Кушнира*

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Наик, Дайлип.

Н20 Системы хранения данных в Windows. : Пер. с англ. — М. : Издательский дом "Вильямс", 2005. — 432 с. : ил. — Парал. тит. англ.

ISBN 5-8459-0746-2 (рус.)

Книга предназначена для читателей, хорошо знакомых с компьютерными системами и индустрией информационных технологий и желающих расширить познания в области систем хранения данных и архитектуры Windows NT, непосредственно связанной с подобными системами. В книге описываются корпоративные системы хранения данных, в то время как системам потребительского уровня уделяется меньше внимания. В этом издании сделана попытка поддержать интересы специалистов по программному обеспечению, мало знакомых с технологиями хранения данных, и профессионалов в области систем хранения данных, которые стремятся получить дополнительные знания по архитектуре обработки и хранения данных в Windows NT. В то же время книга будет интересна всем читателям, намеревающимся получить исчерпывающие сведения по описанной теме.

ББК 32.973.26-018.2.75

ISBN 5-8459-0746-2 (рус.)
ISBN 0-321-12698-X (англ.)

Оглавление

Содержание

Введение

Гордон Мур (Gordon Moore), один из основателей компании Intel, однажды заметил, что плотность транзисторов на квадратный дюйм удваивается каждый год. Впоследствии скорость немного снизилась и удвоение стало происходить за полтора года. Если верить аналитикам, развитие индустрии систем хранения данных для предприятий все еще соответствует закону Мура.

Существует предположение, что в течение нескольких следующих лет будет сгенерировано больше данных, чем было создано за всю историю человечества! Не отвлекаясь на детали, можно согласиться, что серверы под управлением операционной системы Windows играют важную роль в индустрии систем хранения данных для предприятий. Таким образом, знание аспектов систем хранения данных на базе Windows оказывается крайне важным, и эта книга представляет собой скромную попытку предоставить необходимые сведения.

Понятия "Windows NT" и "семейство Windows Server" в данной книге равнозначны. Оба термина упоминаются при рассмотрении возможностей, которые доступны одновременно в операционных системах Windows NT 4.0, Windows 2000 и Windows Server 2003. В случае необходимости указывается определенная версия операционной системы, например Windows 2000 или Windows Server 2003.

Книга предназначена для читателей, хорошо знакомых с компьютерными системами и индустрией информационных технологий и желающих расширить познания в области систем хранения данных и архитектуры Windows NT, непосредственно связанной с подобными системами. Другими словами, в книге описываются корпоративные системы хранения данных, в то время как системам потребительского уровня уделяется гораздо меньше внимания. В книге делается попытка поддержать интересы специалистов по программному обеспечению, мало знакомых с технологиями хранения данных, и специалистов по системам хранения данных, которые стремятся получить дополнительные знания по архитектуре обработки и хранения данных в Windows NT.

Еще одна цель книги — донести до читателя мысль, что каждая новая версия операционной системы Windows NT привносит на рынок массу новых возможностей для корпоративных систем хранения данных.

Следует отметить ряд особенностей, касающихся содержания книги.

- Удобное изложение информации.
- Предоставление подробных сведений с соблюдением прав на интеллектуальную собственность. В книге рассматривается несколько инструментов для разработки программного обеспечения (SDK), которые доступны только после подписания соглашения о неразглашении (non-disclosure agreement — NDA). Таким образом, описание этих инструментов ограничено до информации, которая уже представлена в открытых источниках. Здесь выбран осторожный подход к этому вопросу и приводится только открытая информация, которая, впрочем, тщательно проработана, а к данным, трудным для понимания, добавлены необходимые объяснения.
- Предоставление информации о следующих версиях операционных систем Windows NT, а не только устаревших сведений об уже выпущенных системах. Компания Microsoft намерена уделить серьезное внимание реализации поддержки корпоративных систем хранения данных в следующих версиях Windows NT. Безусловно, это несколько рискованный подход, так как планы могут измениться. Поэтому в книге всегда явно указывается, что та или иная описываемая технология относится к будущим версиям Windows NT.

Небольшое отступление. В книге описывается ряд возможностей, которые будут представлены в следующих версиях Windows NT. Компания Microsoft без устали твердит, что гарантированный метод получения информации о возможностях операционной системы — это знакомство с ней после выпуска финальной версии. Хотя такого рода информация широко обсуждается на выставках и семинарах, нет никакой гарантии, что эти возможности вообще появятся в финальной версии. Подобные предположения не должны становиться основой для конкретных планов.

Каждый читатель, который не понял всей серьезности этого предостережения, должен изучить (а не только прочитать) его еще раз.

В начале книги приводится обзор архитектуры Windows NT, включая подсистему ввода-вывода и архитектуру драйверов подсистемы хранения данных. В главе 1 делается попытка кратко изложить огромный объем информации, которая рассматривалась в серии книг *Inside Windows NT* (издательство Microsoft Press). Глава 1 предназначена для читателей, не имеющих достаточно свободного времени для чтения подобных книг.

В главе 2 описывается технология хранилищ данных, непосредственно подключенных к серверу (direct-attached storage), которая исторически была первой реализацией систем хранения данных.

В главе 3 рассматривается технология NAS (Network-Attached Storage — сетевое устройство хранения данных), которая стала следующим шагом на пути эволюции корпоративных систем хранения данных. Особое внимание уделяется стеку сетевых протоколов Windows NT.

В главе 4 описываются системы SAN (Storage Area Network — сети хранения данных) на основе технологии Fibre Channel. Эта технология продолжает развиваться и по-прежнему составляет конкуренцию таким новшествам, как iSCSI и InfiniBand.

В главе 5 освещаются базовые концепции резервного копирования и восстановления данных. Кроме того, затрагиваются вопросы, связанные с новой службой теневого копирования дисковых томов, которая впервые появилась в операционной системе Windows Server 2003.

В главе 6 рассматриваются файловые системы и виртуализация дисков в контексте Windows NT. Кроме того, описываются кластерные файловые системы.

Глава 7 посвящена управлению системами хранения данных как в рамках общих аспектов, так и в контексте Windows NT.

В главе 8 рассматриваются новые технологии хранения данных (особое внимание уделяется IP-хранилищам, которые являются попыткой связать воедино системы хранения данных и сети на базе протокола IP), а также технология InfiniBand.

В главе 9 описываются методы реализации отказоустойчивых служб (включая защиту и восстановление целостности данных, а также балансировку нагрузки) средствами Windows Server 2003 и Windows 2000 на основе многопортовых парных адаптеров локальной шины (HBA), установленных на серверах Windows NT. Кроме того, приводятся более простые методы обеспечения отказоустойчивости и повышения быстродействия систем, например массивы RAID.

Хотя глава 10 посвящена различным технологиям, она организована в соответствии с разными версиями Windows NT. Независимо от технологий хранения данных, которые рассматриваются в каждой конкретной главе, глава 10 основана на хронологическом порядке появления технологий в операционных системах Windows NT 4.0, Windows 2000, Windows Server 2003. Особое внимание уделяется ожидаемым функциям в следующих версиях Windows.

Надеюсь, книга вам понравится.

Отзывы и предложения присылайте по адресу: `dilipn@niriva.com`.

Дайлип С. Наик
Редмонд, Вашингтон
`dilipn@niriva.com`

Благодарности

Предприятие такого масштаба никогда не могло бы быть завершено в одиночку. Я искренне благодарен всем, кто помог мне в этой работе.

- Моим редакторам Карен Гетман (Karen Gettman) и Эмили Фрей (Emily Frey). Они постоянно поддерживали мою веру в собственные силы, когда я стремился придерживаться графика и старался передать свои идеи в корректной и доступной форме.
- Тому Кларку (Tom Clark), который очень помог при реализации идеи этой книги, а также оказал содействие в других вопросах.
- Техническим рецензентам Джеймсу Андерсону (James Anderson), Элен Бек Гарднер (Ellen Beck Gardner), Роберту Грисволду (Robert Griswold), Варине Хаммонд (Varina Hammond), Милану Мерхару (Milan Merhar), Бобу Сниду (Bob Snead) и Ричарду Вилеру (Richard Wheeler), которые опознали бриллиант в куске необработанной породы и помогали шлифовать его до тех пор, пока он не приобрел вид, вполне отвечающий своему назначению.
- Редактору Лори МакГайр (Laurie McGuire).
- Редактору тиражирования Стефани Гиберт (Stephanie Hiebert).
- Джеффу Голднеру (Jeff Goldner) и Каран Мехра (Karan Mehra) из компании Microsoft, которые предоставили бесценную информацию.
- И наконец, но не в последнюю очередь, моей семье: жене Варше (Varsha), которая несколько месяцев мирилась с моей работой за портативным компьютером IBM Thinkpad, сыну Нихару (Nihar) и дочери Рити (Riti), которые терпели странного папу, приносившего портативный компьютер на игры по футболу и бейсболу, а также на практические занятия.

Спасибо всем вам!

Ждем ваших отзывов!

Вы, уважаемый читатель, и есть главный критик и комментатор этой книги. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию следующих книг. Наши координаты:

E-mail:	`info@williamspublishing.com`
WWW:	`http://www.williamspublishing.com`

Адреса для писем:

из России:	115419, Москва, а/я 783
из Украины:	03150, Киев, а/я 152

Глава 1

Знакомство с Windows NT и драйверами устройств хранения данных

В этой главе рассматриваются драйверы устройств Windows NT, драйверы фильтрации и стек драйверов устройств хранения данных для семейства Windows Server. Приведенных сведений достаточно для того, чтобы познакомить неискушенного читателя с особенностями подсистемы ввода-вывода операционной системы Windows NT, а также структуры драйверов устройств хранения данных. Основное внимание уделяется ключевым понятиям, которые широко рассматриваются в книге, например групповому вводу-выводу (multipath I/O), функции SIS (Single Instance Storage) в службах удаленной установки (Remote Installation Services — RIS), точкам переопределения Windows NT (reparse points) и службам удаленного хранения Windows (Remote Storage Services — RSS).

Если быть более точным, то эта глава *не* предназначена для предоставления всей справочной информации, которая подготовит читателя к самостоятельному написанию драйверов устройств Windows NT. Точно так же глава не содержит полного описания самой операционной системы Windows NT. Читателю рекомендуется использовать справочные материалы, которые указаны в конце книги, чтобы получить достаточный объем знаний о структуре Windows NT для написания различных драйверов, включая драйвера системных фильтров.

Основное внимание в этой главе уделяется Windows NT и архитектуре драйверов в контексте устройств хранения данных. Возможности Windows NT и архитектура драйверов, которые не касаются устройств хранения, если и рассматриваются, то очень кратко.

В разделах 1.1 и 1.2 предоставлена терминология, которая будет часто использоваться на протяжении всей книги. В этих разделах приводятся такие термины, как *режим ядра* (kernel mode), *пользовательский режим* (user mode) и *контекст процесса* (process context). После вступительных разделов в главе рассматривается стек подсистемы хранения данных Windows, вклю-

чая уровни файловых систем, управления томами, классов и портов. Кроме того, кратко рассматриваются драйверы фильтрации. В завершение приводится описание типичного запроса ввода-вывода, а также рассматривается обработка этого запроса на каждом из уровней стека ввода-вывода подсистемы хранения данных.

1.1 Режимы ядра и пользователя Windows

В этой книге регулярно используются термины *режим ядра* (kernel mode) и *пользовательский режим* (user mode). Перед определением этих терминов рассмотрим историю их происхождения.

Система Windows NT проектировалась, как переносимая операционная система, в которой весь код, зависимый от процессора и аппаратного обеспечения, изолирован в модуле, называемом *уровнем аппаратных абстракций* (hardware abstraction layer — HAL). Этот модуль рассматривается в разделе 1.3.1 далее в главе. Хотя Windows NT действительно раньше поддерживала несколько архитектур центральных процессоров, включая PowerPC и Alpha, современные версии Windows NT поддерживают только процессоры компании Intel и совместимые с ними модели (например, компании AMD). Некоторые базовые особенности архитектуры процессоров Intel рассматриваются далее в этом разделе, причем вместо описания всех возможностей архитектуры x86 затрагиваются лишь наиболее важные аспекты.

Архитектура Intel x86 поддерживает четыре режима работы: реальный[1], виртуальный x86, управления системой и защищенный.

В *реальном режиме* (real mode) каждый системный процесс имеет неограниченный доступ к первому мегабайту оперативной памяти. При загрузке процессор всегда запускается в реальном режиме. Процессор можно переключить в защищенный режим, установив соответствующий флаг в управляющем регистре; для обратного переключения флаг нужно сбросить. Реальный режим используется для инициализации Windows NT, однако переключение в защищенный режим происходит задолго до запуска приложений. В процессе развития семейства продуктов Windows NT роль реального режима становится все менее значимой. Как только процессор переключается в защищенный режим работы, Windows NT больше не переходит в реальный режим.

Виртуальный режим x86 (virtual x86 mode) предоставляет возможность выполнять несколько приложений реального режима, когда процессор работает в защищенном режиме. Операционная система Windows NT 4.0 поддерживает этот режим с помощью подсистемы NT Virtual DOS Machine

[1] Данный режим в Windows NT не используется.

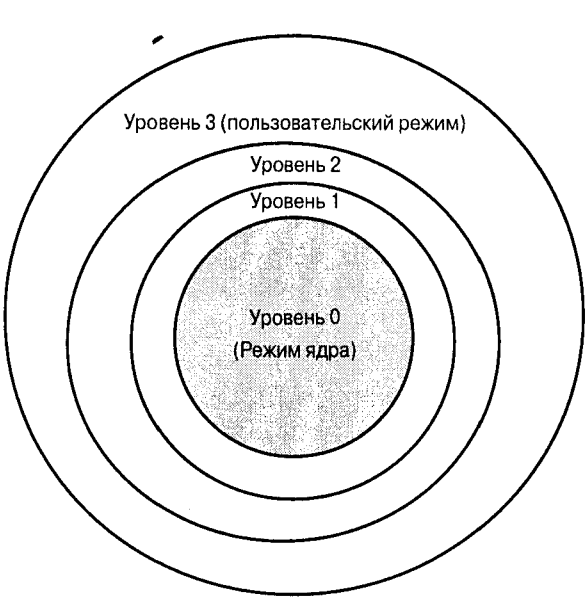


Рис. 1.1. Уровни привилегий архитектуры Intel x86

(NTVDM). Необходимость запуска приложений DOS под управлением Windows NT возникает все реже и реже, поэтому роль подсистемы NTVDM со временем становится все менее важной.

Защищенный режим (protected mode) представляет собой основной режим Windows NT. Он обладает четырьмя рабочими уровнями (рис. 1.1). На уровне 0 (или кольце 0), который чаще всего называется *режим ядра* (kernel mode), доступны инструкции процессора и функции для обеспечения защиты памяти и работы с виртуальной памятью. Кроме того, на уровне 0 доступны привилегированные инструкции, например для управления регистрами процессора. Операционная система Windows NT не использует уровни (кольца) 1 и 2. Самый нижний уровень привилегий — уровень 3, или *пользовательский режим* (user mode), — обеспечивает наилучшую защиту, предотвращая доступ системных процессов к коду и памяти другого процесса.

Далее представлены некоторые функциональные возможности Windows NT для архитектуры x86.

- Всей памятью можно управлять (выделять, считывать и записывать) посредством логических единиц, которые называются *страницами*. Дополнительная информация приводится в разделе 1.3.3.6.
- Каждая страница памяти имеет связанный с ней тег, который определяет возможность чтения или записи этой страницы, а также уровень привилегий, необходимых для чтения и записи. Эта возможность пред-

назначена для защиты пользовательских процессов друг от друга и для защиты системных кода и данных от пользовательских процессов. Обратите внимание, что системный код, выполняемый в режиме ядра, не защищен от другого кода, который выполняется в режиме ядра.

- Страницы памяти, которые содержат код (в отличие от данных), могут быть отмечены как предназначенные только для чтения пользовательскими процессами и кодом на уровне ядра.

Приложения, которые выполняются в пользовательском режиме, получают доступ к службам ядра Windows NT, вызывая специальные инструкции, допускающие управляемый переход в режим ядра и обратно в пользовательский режим, как только запрос в режиме ядра будет выполнен.

1.2 Процесс, контекст процесса и потоки

Процесс — это образ выполняемой программы в памяти. Процессу назначается область памяти до окончания его работы. Процесс может совместно использовать код (динамически подключаемые библиотеки) или данные (области совместно используемой памяти) с другим процессом. Процесс описывается объектом процесса, который поддерживается диспетчером объектов. В объекте процесса содержится информация о виртуальном адресном пространстве процесса, приоритете процесса, а также дескрипторы файлов и информация о выделении памяти. В объекте процесса хранятся и другие параметры, которые здесь не рассматриваются.

В Windows NT несколько процессов могут существовать одновременно; но только один процесс выполняется центральным процессором в определенный момент времени. Обратите внимание: драйверы вообще и драйверы систем хранения данных в частности не создают собственных процессов. Операционная система создает несколько процессов для своих нужд, а также определенные процессы в ответ на пользовательские команды, например когда пользователь запускает приложение, такое, как Microsoft Word или Microsoft Excel. Если драйвер вызывается во время работы процесса, считается, что он работает в контексте вызывающего процесса.

Контекст процесса можно обозначить как всю служебную информацию, необходимую для отслеживания работы процесса. К этой информации относятся виртуальная память процесса, значения регистров центрального процессора, различные дескрипторы файлов и объектов, а также различные маркеры безопасности, связанные с процессом. Контекст процесса исключительно важен, так как множество структур данных и ресурсов, таких, как дескрипторы файлов и указатели памяти, действительны только для данного

процесса. Например, дескриптор файла, созданный в одном процессе, недействителен в другом процессе.

Поток — это структурная единица процесса; процесс может содержать один или несколько потоков. Поток совместно использует глобальные структуры данных и адресное пространство процесса, но при этом имеет собственные данные. Переключение между процессами — задача весьма трудоемкая, которая включает в себя сохранение состояния процессора в специальной структуре данных, зависящей от процесса, и изменение регистров управления памятью и процессором. Переключение между потоками осуществляется намного быстрее, поскольку требует сохранения гораздо меньшего объема данных.

Каждый поток связан с объектом потока, поддерживаемым диспетчером объектов. Информация, которая содержится в объекте, включает в себя данные о первичном центральном процессоре в многопроцессорных системах, состояние потока (т.е. выполняется ли поток, или готов к выполнению, или заблокирован в процессе ввода-вывода), а также другие параметры потока.

Поток может работать в пользовательском режиме и в режиме ядра. Потоки в режиме ядра могут создаваться только структурами, работающими в этом режиме, например драйверами устройств, и они всегда выполняются в режиме ядра. Потоки пользовательского режима обычно выполняются именно в этом режиме, кроме тех случаев, когда это происходит в режиме ядра после запроса к службе Windows NT, как уже отмечалось в разделе 1.1.

1.3 Архитектура Windows NT

Операционная система Windows NT проектировалась как модульная, многоуровневая архитектура, поддерживающая расширения за счет добавлениè новых функций. Архитектура позволяет добавлять поддержку новых устройств и новых возможностей, например шифрующей файловой системы (EFS). Архитектура системы позволяет добавлять поддержку приложений, которые основаны на других операционных системах, например OS/2 или POSIX. Конечно, обе эти системы более важны с исторической точки зрения, но они являются хорошим примером модульной расширяемой архитектуры.

На рис. 1.2 показана высокоуровневая архитектура Windows NT. Как уже отмечалось во вступительном разделе, термин *Windows NT* используется для описания всех версий операционной системы Windows, основанных на технологии NT. В это семейство входят Windows NT 3.x, Windows NT 4.0, Windows 2000, Windows XP и Windows Server 2003.

Далее в этом разделе рассматриваются различные компоненты, показанные на рис. 1.2. Обратите внимание на линию, которая разделяет режим ядра

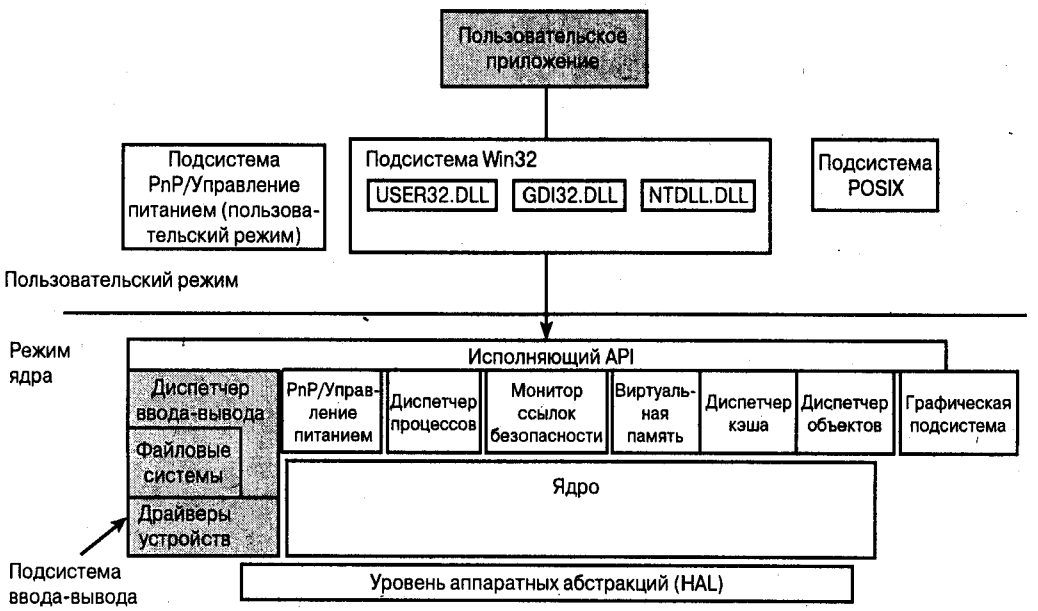


Рис. 1.2. Архитектура Windows NT

и пользовательский режим. Важнейшие различия между этими режимами уже рассматривалась ранее в главе.

Режим ядра содержит все привилегированные процессы, которые выполняются на уровне 0 архитектуры Intel x86. Режим ядра Windows NT состоит из трех основных подсистем.

1. Уровень аппаратных абстракций.
2. Ядро Windows NT.
3. Выполняемый модуль Windows NT.

В следующих трех разделах эти компоненты рассматриваются более подробно.

1.3.1 Уровень аппаратных абстракций

Уровень аппаратных абстракций (Hardware Abstraction Layer — HAL) обеспечивает защиту данных за счет управления доступом к аппаратным ресурсам. Это единственный модуль операционной системы Windows NT, который содержит код, зависящий от аппаратного обеспечения (или от архитектуры процессора). Кроме того, для написания уровня аппаратных абстракций иногда применяется язык ассемблера. В целом уровень аппаратных абстракций предоставляет дополнительный уровень абстракции компонентам более

высокого уровня. Это позволяет создавать высокоуровневые компоненты, не зависящие от аппаратной архитектуры. Ниже описаны функциональные возможности, которые предоставляет уровень аппаратных абстракций.

- Интерфейс службы таймера, благодаря которому выполняемый модуль Windows NT абстрагируется от функций аппаратного обеспечения таймера нижнего уровня.
- Поддержка ввода-вывода в контексте системной шины и прямого доступа к памяти (direct memory access — DMA). Уровень аппаратных абстракций выполняет трансляцию данных между внешней шиной и информацией об адресации Windows NT. Кроме того, предоставляется поддержка для информации о конфигурации шины.
- Поддержка прерываний путем связывания (отображения) внешних прерываний с запросами прерываний (IRQ) Windows NT. Кроме того, предоставляется маскировка/демаскировка служб для прерываний.

1.3.2 Ядро Windows NT

Ядро Windows NT представляет собой следующий уровень после уровня аппаратных абстракций, который обеспечивает работу выполняемого модуля Windows NT (рассматривается в разделе 1.3.3) и других подсистем. Ядро системы выполняет следующие основные функции:

- помощь в синхронизации данных;
- планирование выполнения потоков и процессов;
- управление прерываниями и исключениями;
- восстановление системы после аварийных ситуаций, например после отказа питания.

Данные ядра всегда находятся в оперативной памяти и никогда не выгружаются на диск, как это происходит с пользовательскими приложениями. Данные ядра не могут быть вытеснены другими данными. Это значит, что выполнение кода ядра не может быть прервано ради другого кода, если только ядро не выполняет это самостоятельно. Ядро представляет собой объектно-ориентированную систему, в которой используется два класса объектов.

1. *Объекты-диспетчеры*, которые позволяют управлять потоками и процессами и применяются для синхронизации различных потоков/процессов. В число объектов-диспетчеров входят мьютекс-флаги (mutex — это сокращение от "mutual exclusion", т.е. взаимное исключение), семафоры (semaphore) и таймеры (timer). Мьютекс-флаги являются объектами синхронизации и используются для синхронизации данных между двумя компонентами.

2. *Объекты управления*, например асинхронные вызовы процедур (asynchronous procedure calls — APC) и процедуры обслуживания прерываний (interrupt service routines — ISR), которые рассматриваются более подробно в разделах 1.5.1 и 1.5.3.

1.3.3 Выполняемый модуль Windows NT

Выполняемый модуль (Windows NT Executive) обеспечивает работу ключевых функций, включая программные интерфейсы приложений (API), которые позволяют потокам из пользовательского режима в Windows NT взаимодействовать с ядром Windows NT для запроса на предоставление услуг. Как и ядро Windows NT, выполняемый модуль не может быть выгружен из памяти. Модуль управляет несколькими операциями, включая ввод-вывод данных, поддержку работы системы безопасности, межпроцессное взаимодействие, управление памятью и процессами, поддержку интерфейса Plug and Play, управление питанием, файловыми системами, объектами и графическими устройствами. Весь выполняемый модуль Windows NT размещен в одном файле — `ntoskrnl.exe`. Для выполнения задач модуля создается лишь несколько потоков. Обычно системный процесс из пользовательского режима запрашивает запуск службы, и модуль будет выполняться в контексте запросившего процесса. Примером потока, который создается выполняемым модулем, может служить поток сброса страниц на диск.

Выполняемый модуль Windows NT, в свою очередь, содержит следующие компоненты:

- диспетчер объектов;
- монитор ссылок безопасности;
- диспетчер процессов;
- подсистема Plug and Play;
- диспетчер энергопитания;
- диспетчер виртуальной памяти;
- диспетчер кэша.

1.3.3.1 Диспетчер объектов

Диспетчер объектов (Object Manager) Windows NT предоставляет свои услуги другим компонентам Windows NT, включая непосредственно выполняемый модуль (элементом которого диспетчер и является). Диспетчер объектов предоставляет службы для именования, создания, удаления, манипулирования и совместного использования объектов. Он активно сотрудничает

с монитором ссылок безопасности, чтобы обеспечить соответствующий доступ к определенным объектам только пользователям и процессам с достаточными разрешениями. *Соответствующий* означает, что предоставляется доступ определенного типа, например доступ только для чтения. Каждый объект, созданный диспетчером объектов, имеет связанный с ним список управления доступом (access control list — ACL). На самом деле этот список представляет собой группу объектов, которые указывают разрешения, явно или неявно предоставленные пользователю или группе; кроме того, в список управления доступом могут входить объекты, ограничивающие права доступа для данного пользователя или группы.

Диспетчер объектов назначает дескриптор каждому созданному объекту. При этом обеспечивается уникальность дескрипторов и предоставляется возможность преобразования дескриптора в ссылку на уникальный объект. Но клиенты диспетчера объектов используют дескриптор в виде "непрозрачного" маркера без внутренней структуры. К объектам, с которыми работает диспетчер, относятся файлы, каталоги, порты, процессы, потоки, семафоры и объекты событий.

1.3.3.2 Монитор ссылок безопасности

Монитор ссылок безопасности (Security Reference Monitor) заведует проверкой доступа и протоколированием ресурсов. Проверка доступа выполняется на самом низком уровне, включая не только предоставление доступа, но и определение его типа, например доступ только для чтения или доступ для чтения и записи. Функциональность подсистемы безопасности обеспечивается объектно-ориентированной структурой Windows NT. При предоставлении доступа к объекту монитор ссылок безопасности сравнивает список управления доступом, который связан с объектом, с маркером (token) безопасности процесса перед тем, как предоставить или запретить доступ. Списки управления доступом бывают двух типов: явно или неявно разрешающие или запрещающие доступ. Монитор ссылок безопасности активно используется другими подсистемами выполняемого модуля Windows NT, например диспетчером объектов.

Кроме того, монитор ссылок безопасности предоставляет приложениям пользовательского режима услуги, аналогичные описанным. Монитор обеспечивает возможность генерации маркеров (на уровне процесса), которые могут использоваться для проверки безопасности и разрешений доступа, а также для генерации журналов аудита.

1.3.3.3 Диспетчер процессов

Диспетчер процессов (Process Manager) обеспечивает создание и удаление процессов и потоков, а также управление ими. Диспетчер не поддерживает иерархию компонентов; например, отношения между процессами вида "родитель–потомок" не отслеживаются. Эта работа ложится на компонент, который создал процесс. По аналогии представьте диспетчер файлов, который предоставляет возможность создания файла, однако внедрением этого файла в структуру каталогов должен заниматься пользователь диспетчера файлов. Диспетчер процессов пользуется услугами как диспетчера объектов, так и подсистемы безопасности. Для каждого запущенного процесса передается, как минимум, два вызова диспетчеру процессов: первый вызов для создания процесса, второй — для создания потока в пределах процесса, так как каждый процесс должен содержать хотя бы один поток.

1.3.3.4 Подсистема Plug and Play

На рис. 1.2 управление питанием и подсистема Plug and Play схематически размещены в едином прямоугольнике, что сделано для упрощения структуры диаграммы. На самом же деле это различные подсистемы, хотя и тесно взаимодействующие друг с другом.

Термин *Plug and Play* используется для описания программно и аппаратно реализованных функциональных возможностей, которые позволяют Windows динамически распознавать аппаратное обеспечение и реализовать программную поддержку для корректной работы устройства. В частности, это программное обеспечение отвечает за выполнение следующих функций:

- корректное определение аппаратного обеспечения;
- корректное определение динамического подключения или отключения аппаратного обеспечения;
- выделение и настройка ресурсов для работы аппаратного обеспечения;
- поиск и загрузка драйверов устройств;
- поддержка механизма уведомления, с помощью которого определяется появление и удаление аппаратного обеспечения; этот механизм может использоваться программным обеспечением как в режиме ядра, так и в пользовательском режиме.

Подсистема Plug and Play включает в себя компоненты пользовательского режима и режима ядра. Компонент пользовательского режима предоставляет приложениям метод управления аппаратными устройствами, включая механизмы регистрации, через которые приложения уведомляются о появлении и удалении устройств.

Подсистема Plug and Play играет очень важную роль в обнаружении устройств, присвоении устройствам идентификаторов, инициализации и добавлении/удалении устройств. В частности, Plug and Play отвечает за генерацию пакета запроса ввода-вывода (I/O request packet — IRP), который называется `IRP_MN_QUERY_DEVICE_RELATIONSHIPS` и передается драйверам шины. Этот пакет запроса ввода-вывода в презентациях Microsoft называется QDR. Пакет QDR применяется для перечисления устройств и создания стека устройств. Драйверы фильтрации иногда создаются для отслеживания функций QDR и исправления создаваемого списка устройств. В главе 6 рассматривается диспетчер разделов (Partition Manager), который обеспечивает подобные возможности в качестве драйвера фильтрации.

1.3.3.5 Диспетчер энергопитания

Диспетчер энергопитания (Power Manager) играет важную роль в предоставлении энергосберегающих функций, таких, как снижение оборотов вращения жестких дисков, накопителей для компакт-дисков и DVD, а также отключение питания мониторов и видеоадаптеров. Очевидно, что управление питанием гораздо важнее для портативных компьютеров, чем для серверов, но даже для серверов управление питанием применяется при обслуживании устройств, поддерживающих "горячую" замену, и при управлении устройствами резервного питания. Диспетчер энергопитания предоставляет интерфейс API для приложений более высокого уровня.

1.3.3.6 Диспетчер виртуальной памяти

Диспетчер виртуальной памяти (Virtual Memory Manager — VMM) предоставляет функции управления памятью, благодаря которым процессы могут использовать объем памяти, превышающий размер физической памяти, установленной на компьютере. Запросы приложений на выделение памяти регистрируются диспетчером виртуальной памяти. Если осталось недостаточно памяти, диспетчер виртуальной памяти переместит страницы памяти на жесткий диск, чтобы предоставить место для нового приложения. Если приложение стремится получить доступ к странице, которая отсутствует в физической памяти, диспетчер виртуальной памяти освобождает пространство в памяти перед перемещением страниц с диска в физическую оперативную память. Этот метод получил название *подкачки страниц* (paging).

Область диска, которая используется для хранения страниц, не размещенных в физической памяти, называется *файлом подкачки* (swap file). Операционная система автоматически создает этот файл и обеспечивает его защиту. Администратор может изменять размер файла подкачки, который иногда

называется *страничным файлом*, так как память перемещается в файл и из него страничными блоками.

В Windows NT 4.0 поддерживалось адресное пространство объемом 4 Гбайт, которое поровну распределялось между пользовательским режимом и режимом ядра. Верхние 2 Гбайт выделялись режиму ядра Windows NT, а нижние 2 Гбайт — пользовательскому режиму. В Windows 2000 Advanced Server параметры загрузки позволяют перераспределить адресное пространство, выделив 1 Гбайт режиму ядра и 3 Гбайт пользовательскому режиму. Приложения пользовательского режима следует переписать для использования дополнительного объема адресного пространства. Конечно, для 64-разрядной версии Windows NT подобного ограничения просто не существует.

Диспетчер виртуальной памяти предоставляет программные интерфейсы приложений для выделения и высвобождения памяти, а также для блокировки и разблокировки памяти. Функция *блокировки памяти* позволяет запретить выгрузку области памяти в файл подкачки. Эта функция зачастую ошибочно воспринимается в качестве запрета на изменение физического расположения участка памяти. Хоть это и справедливо для нынешних версий Windows NT, ситуация может измениться в будущих версиях Windows. Посредством описываемого API драйверы по мере необходимости запрашивают выгружаемую или невыгружаемую память.

1.3.3.7 Диспетчер кэша

Это неотъемлемый элемент подсистемы ввода-вывода, который работает в тесной связке с драйверами файловых систем и диспетчером виртуальной памяти. Диспетчер кэша Windows NT взаимодействует с файловой системой и ее драйверами. Подобный метод отличается от стратегии кэширования, свойственной Windows 95, которая предназначалась для взаимодействия непосредственно с дисковыми секторами. Диспетчер обслуживает все файловые системы, локальные и удаленные, с помощью единого кэша. Диспетчер кэша позволяет кэшировать несколько потоков данных из одного файла. Потоки данных относятся к возможностям файловой системы NT (NTFS) и рассматриваются в главе 6.

Весь процесс файлового ввода-вывода данных можно рассматривать как страничный ввод-вывод, что связано с особенностями взаимодействия диспетчера кэша и диспетчера виртуальной памяти. При запросе операции файлового ввода-вывода файловая система сначала обращается к диспетчеру кэша для получения необходимых данных. Если диспетчер кэша обнаруживает, что необходимые данные недоступны, запрос отправляется драйверу файловой системы для считывания данных. В результате получается замкнутый круг, так как файловая система запрашивает страницы у диспетчера кэша,

а диспетчер кэша, в свою очередь, запрашивает страницы у файловой системы. Но при этом диспетчер кэша различными способами регистрирует ввод-вывод данных, в результате чего файловая система больше не пытается повторно получить кэшированные данные.

1.3.4 Подсистема ввода-вывода

Подсистема ввода-вывода отвечает за обработку запросов ввода-вывода и проектировалась для выполнения перечисленных далее задач.

- Обеспечение работы сверхпроизводительных операций ответного ввода-вывода для одно- и многопроцессорных компьютеров.
- Предоставление асинхронного ввода-вывода. Синхронный ввод-вывод осуществляется, по сути, в виде асинхронного запроса ввода-вывода, после которого следует блокирующее ожидание завершения операции ввода-вывода.
- Поддержка нескольких файловых систем, в частности CDFS, NTFS и UDFS.
- Предоставление модульной архитектуры, поддерживающей добавление новых файловых систем и устройств.
- Предоставление устройствам (и их драйверам) возможности подключения и отключения "на лету", без перезагрузки (эта функция реализована в Windows 2000 и более новых версиях Windows NT).
- Предоставление расширенных возможностей, например кэширования и записи содержимого файлов в память (запись содержимого файла в указанной области памяти в адресном пространстве процесса). Для получения доступа или модификации содержимого файла приложение выполняет чтение и запись определенной области адресного пространства.
- Защита ресурсов, которые совместно используются несколькими процессами.

Подсистема ввода-вывода имеет модульную структуру (как и все остальные компоненты Windows NT) и состоит из следующих компонентов:

- программный интерфейс приложений ввода-вывода (I/O API);
- диспетчер ввода-вывода;
- драйверы файловых систем;
- другие драйверы (например, драйверы клавиатуры и драйверы дисков).

Далее эти модули рассматриваются более подробно.

1.3.4.1 Программный интерфейс приложений ввода-вывода (I/O API)

По сути, этот компонент включает функции диспетчера ввода-вывода, предназначенные для более высоких уровней Windows NT, а также компоненты режима ядра, выполняющие операции, связанные с диспетчером печати. Все имена функций программного интерфейса приложений ввода-вывода имеют вид `IoXXXX`, где `XXXX` — строка, после которой указывается список параметров. (Подробная информация приводится в программном инструментарии для разработки драйверов.) В качестве примера функций API можно привести:

- интерфейс `IoCreateDevice`, предназначенный для создания новых объектов устройств (объекты устройств рассматриваются в разделе 1.4.2);
- интерфейс `IoCallDriver`, предназначенный для отправки драйверу пакета запроса ввода-вывода (пакеты запроса ввода-вывода рассматриваются в разделе 1.4.3).

1.3.4.2 Диспетчер ввода-вывода (I/O Manager)

Это элемент выполняемого модуля Windows NT; свойственные ему функции перечислены ниже.

- Создание пакетов запроса ввода-вывода (IRP) и направление их соответствующему драйверу, а также перенаправление пакетов запроса ввода-вывода между драйверами.
- Удаление и освобождение пакетов запроса ввода-вывода после завершения операции ввода-вывода.
- Взаимодействие с диспетчером кэша и другими компонентами NT Executive.
- Взаимодействие с диспетчером виртуальной памяти для предоставления файловым системам функций ввода-вывода с записью данных в память.
- Мониторинг загруженных файловых систем и их вызов по требованию.
- Предоставление поддержки синхронного и асинхронного ввода-вывода. Асинхронный ввод-вывод особенно важен для приложений хранения данных. Например, приложение резервного копирования может использовать асинхронный ввод-вывод для размещения в очереди нескольких запросов, что позволяет полностью загрузить устройство записи на ленту.
- Управление буферами для операций ввода-вывода.

1.3.4.3 Драйверы файловых систем

Операционная система предоставляет функции файловых систем с помощью драйверов режима ядра. Система Windows NT поставляется вместе с такими файловыми системами:

- NTFS (файловая система NT);
- UDFS (универсальная дисковая файловая система);
- CDFS (файловая система компакт-дисков);
- FAT (таблица размещения файлов).

Драйверы сетевых файловых систем рассматриваются в главе 3. Драйверы файловых систем реализуются средствами инструментария разработки драйверов Windows NT (Windows NT DDK) и дополнительного программного продукта, который предлагается компанией Microsoft — Windows NT Installable File System Kit. Этот инструментарий содержит документацию для различных программных интерфейсов приложений, которые понадобятся при создании драйверов файловой системы, а также пример кода, предназначенного для реализации файловых систем FAT и UDFS.

Драйверы файловой системы аналогичны другим драйверам, поскольку взаимодействуют с диспетчером ввода-вывода и IRP. Драйверы файловой системы являются логическими, так как не взаимодействуют непосредственно с аппаратным обеспечением; например, файловая система не делает различия между дисками с интерфейсом SCSI и с интерфейсом ATA (иногда называемым IDE). Тем не менее драйверы файловой системы отличаются от других драйверов. Некоторые из этих отличий приведены ниже.

- Драйверы файловой системы всегда вызываются в контексте потока, запрашивающего операцию ввода-вывода.
- Драйверы файловой системы активно взаимодействуют с диспетчером кэша и диспетчером виртуальной памяти, используя эти два компонента для буферизации данных. Например, файловая система использует услуги диспетчера кэша для кэширования метаданных файловой системы (это может быть расположение файлов и каталогов на диске), чтобы избежать повторных запросов одних и тех же метаданных.
- Драйверы файловой системы являются единственными драйверами, которые обеспечивают работу методов ввода-вывода на основе IRP. Подобный метод называется быстрым вводом-выводом (Fast I/O) и представляет собой несколько входных точек драйвера. Диспетчер ввода-вывода вызывает эти точки для выполнения операций ввода-вывода, поскольку данные могут быть кэшированы и поэтому быстро обработаны. Драйвер

файловой системы может завершить вызов неудачно, если это необходимо, а диспетчер ввода-вывода просто повторит тот же запрос ввода-вывода с помощью обычного пакета IRP.

Понятие драйверов фильтров файловых систем тесно связано с понятием драйверов файловых систем. Драйверы фильтрации файловых систем используются для реализации широкого диапазона различных технологий, например шифрованной файловой системы (EFS) и поддержки служб удаленного хранения (RSS).

1.3.5 Графическая подсистема

Поскольку эта книга посвящена корпоративным системам хранения данных, на рис. 1.2 графическая подсистема демонстрируется в режиме ядра, хотя некоторая область подсистемы также представлена и в пользовательском режиме. В Windows 2000 для повышения производительности значительный объем кода графической подсистемы был перемещен из пользовательского режима в режим ядра. В данном случае к понятию "графическая подсистема" относится весь программный код, предназначенный для обработки оконного интерфейса и поддерживающий видеоустройства, сканеры, принтеры и т.д.

1.3.6 Подсистема Win32

Это наиболее важный компонент Windows NT, особенно для программистов. На основе программного интерфейса Win32 создаются другие подсистемы, такие, как POSIX.

Программный интерфейс приложений Win32 можно разделить на три категории.

1. Обработка оконного интерфейса и API для сообщений оконного интерфейса реализованы в виде динамически подключаемой библиотеки `user32.dll`. Эта библиотека подключается приложениями, использующими интерфейсы, которые предоставляются этим файлом. При этом несколько приложений во время работы задействуют только одну копию библиотеки.
2. Графический API реализован в виде динамически подключаемой библиотеки `gdi32.dll`. В версиях Windows NT до Windows 2000 библиотека `gdi32.dll` являлась клиентом, подключаемым к серверному процессу Win32 (рассматривается далее), так как соответствующие функции были реализованы в серверном процессе. А сервер Win32, в свою очередь, вызывал компонент режима ядра для графической подсистемы. В Windows 2000 библиотека `gdi32.dll` вызывает этот компонент без посредников.

3. Базовые API, например функции открытия файла (`CreateFile`), чтения файла (`ReadFile`) и записи файла (`WriteFile`), реализованы в динамически подключаемой библиотеке, которая называется `ntdll.dll`. При необходимости эта библиотека делает вызовы к выполняемому модулю Windows в режиме ядра. Для этого библиотека использует одно из 256 прерываний, поддерживаемых архитектурой Intel x86. В частности, используется прерывание 46 (десятичный номер 46, шестнадцатеричный — 0x2E). Обработчик прерывания[2] идентифицирует как запрошенный API (выполнив поиск по таблице), так и передаваемые ему параметры. Если все параметры прошли проверку, обработчик вызывает соответствующую подсистему выполняемого модуля Windows для выполнения запрошенной операции.

Приложения, написанные на основе API Win32 и других механизмов поддержки, рассматриваются в документации SDK. В некотором смысле даже подсистема POSIX представляет собой инструмент, разработанный для поддержки приложений UNIX. Хотя подсистема POSIX в настоящий момент существенной роли уже не играет, она все еще служит хорошим примером модульной и расширяемой архитектуры Windows NT.

1.4 Структуры данных, связанные с драйверами устройств Windows

Перед подробным рассмотрением драйверов устройств Windows NT стоит разобраться в некоторых важных структурах данных, которые используются этими драйверами. Каждый драйвер Windows, включая драйверы устройств хранения данных, должен взаимодействовать с тремя основными типами объектов: объектами драйверов, объектами устройств и пакетами запроса ввода-вывода (IRP). Эти объекты и рассматриваются в данном разделе.

1.4.1 Объекты драйверов

Объект драйвера создается выполняемым модулем Windows NT при загрузке драйвера. Объект драйвера выделяется из невыгружаемой памяти. Он содержит важную информацию, например *таблицу вызовов драйвера*, которая, в свою очередь, содержит адреса для различных процедур драйвера. Каждый драйвер, даже если он управляет несколькими устройствами, представлен только одним объектом. Кроме того, когда драйвер обрабатывает-

[2]Строго говоря, это не просто обработчик прерываний, однако в данный момент не будем углубляться в детали, поэтому далее обработчик прерываний будет именоваться просто прерыванием.

ся несколькими ЦПУ в многопроцессорной системе Windows NT, в памяти присутствует только один объект драйвера. Хотя объект драйвера создается выполняемым модулем Windows NT, обязанность по внесению определенной информации, например адресов процедур в таблицу вызовов драйвера, возлагается на создателя драйвера. Это требование относится только к драйверам, которые экспортируют объект драйвера; таким образом, мини-драйверы, которые зависят от классов или портов объекта драйвера, не обязаны предоставлять описываемую информацию об объекте.

1.4.2 Объекты устройств

Объект устройства представляет собой физическое устройство ввода-вывода (т.е. шинный адаптер, диск или привод на магнитных лентах) или логическое устройство (например, драйвер антивирусного фильтра). Каждое устройство может быть представлено только одним объектом устройства. Объект устройства содержит указатель на объект драйвера, который управляет обработкой устройства. Кроме того, объект устройства описывает физические характеристики устройства, например наибольший объем ввода-вывода данных, выполняемый за одну операцию, или компонент, посредством которого выравнивается буфер, предоставляемый устройству.

Три типа объектов устройства имеют одинаковую структуру, однако различаются расширениями и методами использования. Эти объекты описаны ниже.

1. *Объект физического устройства* (physical device object — PDO) представляет собой устройство, подключенное к шине и обычно создается драйвером шины (драйверы шины рассматриваются в разделе 1.7.1). Объект физического устройства должен поддерживать связь с устройством. Такой объект должен хранить статус энергопитания устройства и идентификатор устройства, например идентификатор шины SCSI, целевой идентификатор SCSI и *номер логической единицы* (LUN) SCSI. Эти термины более подробно рассматриваются в главе 2. На данный момент достаточно сказать, что для уникальной идентификации устройства SCSI необходимо указать три значения: идентификатор шины, целевой идентификатор и идентификатор LUN.
2. *Объект функционального устройства* (functional device object — FDO) обычно создается драйвером класса или драйвером порта (драйверы классов и портов рассматриваются в разделах 1.7.2 и 1.7.3). Использование устройства требует наличия объекта функционального устройства. В качестве примера данных, которые содержатся в объекте функционального устройства, можно указать элементы архитектуры диска,

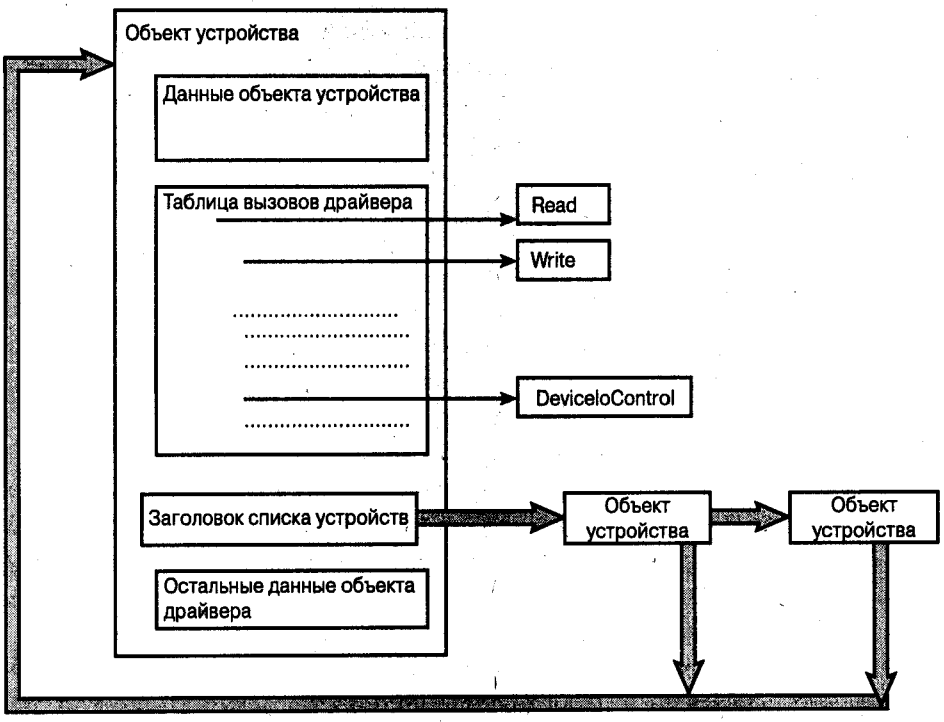


Рис. 1.3. Архитектура объекта устройства Windows NT

например таблицу разделов диска или в контексте привода DVD информацию о регионе DVD.

3. *Объект фильтра устройства* (filter device object — DO) представляет собой устройство для драйвера фильтра.

На рис. 1.3 демонстрируются основные структурные элементы объекта драйвера.

К важным элементам объекта драйвера относится таблица вызовов. В ней определены различные стандартные функции, которые реализуются драйвером устройства. В зависимости от конкретной структуры драйвера, некоторые из этих функций должны быть реализованы в обязательном, а некоторые — в произвольном порядке. На рис. 1.3 показаны только функции `Read`, `Write` и `DeviceIoControl`, однако существует и множество других. Для получения дополнительной информации можно обратиться к инструментарию разработки драйверов Windows.

Устройства, функции которых реализуются с помощью драйвера, описываются посредством объектов устройств (PDO, FDO и DO). Все устройства представлены в связном списке. Заголовок связного списка хранится в объек-

те драйвера, что демонстрируется в нижней левой области рис. 1.3. Обратите внимание, что указатель на объект драйвера в объекте устройства позволяет просмотреть структуру данных, найти объект драйвера и вызвать соответствующую функцию по таблице вызовов.

1.4.3 Пакеты запросов ввода-вывода

Для взаимодействия с драйверами режима ядра многоуровневая операционная система Windows NT использует интерфейсы на основе пакетов данных. Пакеты, которые используются для связи с драйверами, называются *пакетами запроса ввода-вывода* (I/O request packets — IRP). К драйверу может подключаться другой драйвер или подсистема ввода-вывода.

Пакеты IRP выделяются в невыгружаемой памяти — важнейшем системном ресурсе. Пакеты запроса ввода-вывода выделяются и поддерживаются в очереди, связанной с определенным потоком. Операционная система Windows NT поддерживает готовность к работе некоторого количества пакетов IRP, размещенных в выделенном[3] списке, что позволяет после запроса быстро назначать пакеты драйверу или диспетчеру ввода-вывода.

На рис. 1.4 показано, что пакет запроса ввода-вывода имеет заголовок фиксированного размера и непостоянное количество элементов стека ввода-вывода. Элементы стека ввода-вывода представляют структуры данных для отдельных драйверов, которые будут обрабатывать IRP. Таким образом, каждый драйвер, который обрабатывает пакет запроса ввода-вывода, получает закрытую область данных в стеке пакета. При выделении и отправке пакета драйверу требуется формирование стека достаточного объема для каждого драйвера, который будет обрабатывать пакет. Если драйвер попытается получить доступ к несуществующему элементу стека IRP, это может привести к появлению ошибок в работе системы. Таким образом, пакет запроса ввода-вывода, который может использоваться для одной цепи стека драйвера, не всегда подходит для другой цепи этого стека.

На рис. 1.4 демонстрируется, что заголовок пакета запроса ввода-вывода содержит различные поля данных, перечисленные ниже.

- Тип запроса (синхронный или асинхронный).
- Тип операции запроса (операция страничного ввода-вывода или операция, выполняемая без промежуточного кэширования).
- Указатель на буфер для операции ввода-вывода.

[3]Термин "выделенный список" относится к ситуации, когда некий важный системный элемент (например, пакеты IRP) содержится в отдельном списке и может быть успешно перемещен из одного списка в другой (например, список пакетов IRP, присоединенных к определенному потоку).

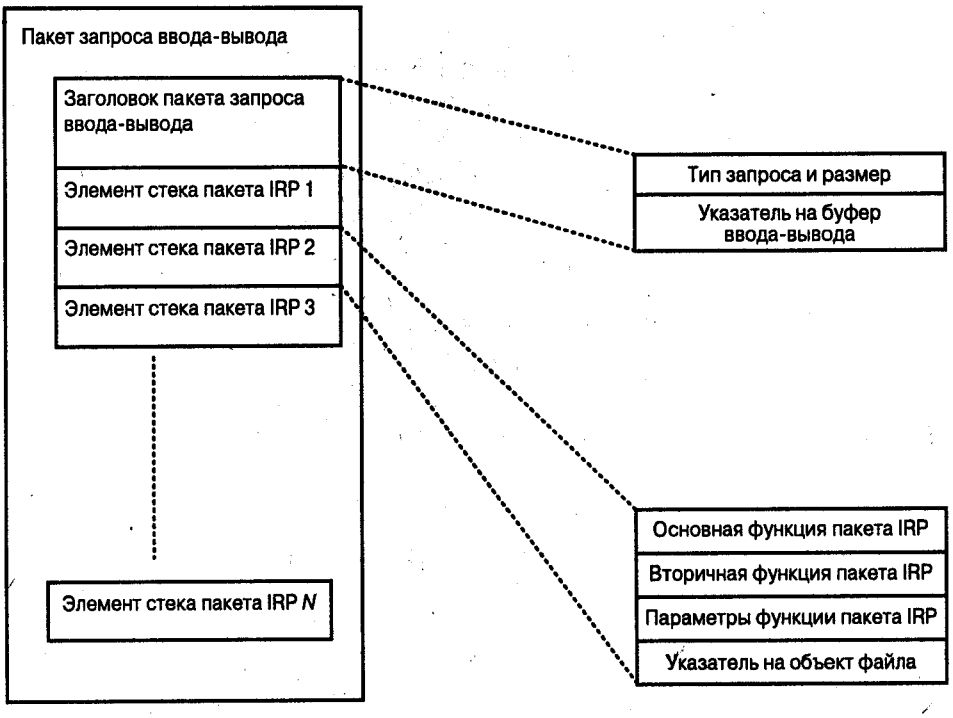


Рис. 1.4. Структура пакета запроса ввода-вывода

- Статус блока ввода-вывода, который представляет состояние пакета запроса ввода-вывода. Этот статус меняется при обработке пакета различными драйверами.
- Информация, необходимая для обработки отмены IRP в случае отмены операции ввода-вывода, указанной в пакете (например, после достижения тайм-аута или по команде пользователя, который решил, что операция выполняется слишком долго).
- Указание области возникновения запроса ввода-вывода (в пользовательском режиме или режиме ядра).

Кроме того, на рис. 1.4 показано, что каждый элемент стека содержит информацию, относящуюся к конкретному драйверу. Ниже приведено несколько примеров информации, которая содержится в элементе стека.

- Код основной функции (чтение, запись и операция по управлению устройством).
- Код вторичной функции, который уточняет необходимое действие и применяется только вместе с соответствующим кодом основной функции.
- Параметры функции, например параметры управления устройством.

- Указатель на объект файла.

Структура пакета запроса ввода-вывода содержит элемент, указывающий на текущий элемент стека пакета. Драйвер отвечает за изменение значения этого элемента перед передачей пакета следующему драйверу в цепочке драйверов. Каждый активный пакет запроса ввода-вывода размещается в очереди, связанной с потоком, который содержит пакеты запроса ввода-вывода, связанные с вводом-выводом, запрошенным этим потоком.

В ответ на полученные пакеты драйвер может создавать вторичные пакеты IRP, которые могут обрабатываться одновременно, имитируя параллельную обработку. Это позволяет ускорить обработку ввода-вывода данных. Конечно, в этом случае существуют определенные ограничения: например, операция записи на ленту не может быть разделена на несколько параллельных операций.

1.5 Структура драйвера устройства Windows

Все драйверы устройств Windows имеют одинаковую структуру. Каждый драйвер имеет объект драйвера, который создается диспетчером ввода-вывода при загрузке драйвера. В разделе 1.4 представлены структуры данных, которые относятся к драйверам устройств, в том числе и объекты драйверов. В этом разделе описываются процедуры, реализуемые драйвером, а также другие характеристики драйвера устройства хранения данных.

Драйвер устройства Windows реализует множество стандартных процедур, причем некоторые из них обязательны для выполнения, а некоторые — нет, что зависит от свойств драйвера. Ниже перечислены основные стандартные процедуры.

- Обязательная *процедура инициализации*, которая используется драйвером для подготовки рабочего окружения и собственной инициализации, а также для настройки объектов устройств (в том числе их подключения в соответствующие цепочки стека драйверов). Эта процедура вызывается диспетчером ввода-вывода при загрузке драйвера.
- Обязательный набор *процедур диспетчеризации* для обеспечения работы определенных функций, например чтения, записи, создания и закрытия файлов. Эти процедуры вызываются диспетчером ввода-вывода и получают в качестве параметра пакет запроса ввода-вывода.
- Необязательная *процедура запуска* (startup routine — StartIO), которая инициирует ввод-вывод данных на физическое устройство. Очевидно, что только драйверы, работающие непосредственно с физическими устройствами (это касается не всех драйверов такого типа), требуют наличия такой процедуры.

- Необязательная *процедура обслуживания прерывания* (interrupt service routine — ISR). Может использоваться драйверами, взаимодействующими с физическими устройствами. Процедуры обслуживания прерываний рассматриваются в разделе 1.5.1.
- Необязательный *отложенный вызов процедуры* (deferred procedure call — DPC), который может использоваться драйвером для дополнительной обработки процедуры обслуживания прерывания. Отложенный вызов процедуры рассматривается в разделе 1.5.2.
- Необязательная *процедура завершения*, которая вызывается диспетчером ввода-вывода (в качестве механизма уведомления), когда драйвер более низкого уровня завершает обработку пакета запроса ввода-вывода. Поскольку вся операция ввода-вывода обрабатывается в качестве асинхронной, процедура завершения используется довольно часто, особенно в высокоуровневых драйверах, которые всегда обеспечивают обработку пакетов IRP более низкого уровня.
- Обязательная *процедура выгрузки*, которая вызывается диспетчером ввода-вывода для выгрузки драйвера.
- Необязательная *процедура отмены* (cancellation routine — CancelIO), которая вызывается диспетчером ввода-вывода для отмены выполнения длительной операции.
- Обязательная *процедура уведомления об отключении системы*, которая вызывается диспетчером ввода-вывода для уведомления драйвера о необходимости быстрого завершения работы, когда пользователь обращается с запросом о завершении работы системы.
- Необязательная *процедура протоколирования ошибок*.

Обработка пакета запроса ввода-вывода совершается драйвером различными способами, в зависимости от структуры драйвера и запроса ввода-вывода в пакете. Ниже приведены некоторые примеры работы драйвера.

- Выполнение запрошенной операции и завершение обработки IRP.
- Выполнение элемента операции и передача IRP драйверу более низкого уровня.
- Обычная передача IRP драйверу более низкого уровня.
- Генерация нескольких пакетов IRP для драйвера более низкого уровня в ответ на получение одного пакета IRP. Например, в ответ на запрос об открытии файла, поступивший от драйвера NTFS, драйверу может потребоваться считать метаданные файла для поиска каталога и подкаталогов, в которых расположен необходимый файл.

Обычно драйверы получают доступ к своей области стека в пакете IRP, а также к области стека следующего драйвера. Самый нижний драйвер в цепочке стека получает доступ только к своему фрагменту стека в пакете IRP. Драйвер отвечает за изменение указателя в пакете IRP, который указывает на область стека, используемую следующим драйвером.

Обратите внимание, что один и тот же код драйвера может одновременно выполняться на разных центральных процессорах в одной системе Windows NT. Код драйвера должен обладать возможностью синхронизировать доступ к критическим данным кода, выполняемого на разных процессорах. Иногда повторное выполнение одного запроса может стать просто катастрофой, например при записи на магнитную ленту одних и тех же данных повторно.

1.5.1 Процедура обслуживания прерывания

Процедура обслуживания прерывания (interrupt service routine — ISR) обычно выполняется в ответ на получение прерывания от аппаратного устройства и может вытеснять любой код с более низким приоритетом. Процедура обслуживания прерывания должна использовать минимальное количество операций, чтобы центральный процессор имел свободные ресурсы для обслуживания других прерываний. Эта процедура собирает минимум необходимой информации и размещает в очереди *вызов отложенной обработки* (deffered processing call — DPC) для завершения обслуживания прерывания. Запуск вызова отложенной обработки не планируется на определенное время, т.е. вызов может быть запущен как немедленно, так и немного позднее, в зависимости от необходимости в другой обработке.

Для того чтобы обеспечить постоянную доступность процедуры обслуживания прерывания, процедуры никогда не выгружаются на жесткий диск. Процедура обслуживания прерывания может быть прервана процедурой обслуживания прерывания с более высоким приоритетом, но ее никогда нельзя вытеснить другим кодом, например вызовом отложенной обработки.

Процедура обслуживания прерывания обычно необходима драйверам, которые работают с аппаратным обеспечением, например с накопителем на магнитной ленте или жестким диском. Чаще всего, драйверы, которые обеспечивают работу некоторых программных функций, например драйверы файловой системы или фильтрации, не используют процедуру обслуживания прерывания.

1.5.2 Вызов отложенной обработки

При запуске от процедуры обслуживания прерывания требуется быстрое и эффективное выполнение поставленной задачи. Таким образом, процедура обслуживания прерывания проводит минимум операций и размещает в очереди запрос на *вызов отложенной обработки*, который используется для завершения оставшихся операций с низким уровнем приоритета (эти уровни обычно называются IRQ или IRQL). Вызов отложенной обработки может быть размещен в очереди не только из процедуры обработки прерывания. Запрос к очереди создает новый объект вызова отложенной обработки (средствами диспетчера объектов). После размещения в очереди создается аппаратный запрос на прерывание (IRQ level 2) для вызова отложенной обработки.

Ниже описаны некоторые важные свойства вызова отложенной обработки.

- Вызов отложенной обработки может быть прерван другой процедурой обработки прерывания, однако никогда не может быть вытеснен кодом пользовательского режима.
- Вызов отложенной обработки не должен приводить к ошибке обращения к странице, поэтому вся память, используемая вызовом отложенной обработки, должна быть заблокирована для выгрузки.
- Вызов отложенной обработки не должен выполнять блокирующие действия, например блокирующий ввод-вывод.
- Вызов отложенной обработки напоминает процедуру обработки прерывания, поскольку также должен выполняться быстро и эффективно. Для минимизации нагрузки на систему при планировании вызовов отложенной обработки Windows NT перед передачей управления DPC сохраняет минимальную информацию о состоянии. После завершения DPC восстановление состояния также занимает мало времени, так как при передаче управления сохранялся минимум информации. В результате DPC может выполняться в контексте произвольного процесса. Например, если программа Excel выполняется в виде процесса и запускает процедуру ввода-вывода, вызов отложенной обработки (если он потребуется) может запускаться в контексте процессов Word или PowerPoint (а не обязательно в контексте процесса Excel).
- Каждый процесс имеет собственную очередь вызовов отложенной обработки. Таким образом, многопроцессорный компьютер с четырьмя центральными процессорами будет иметь четыре отдельных очереди DPC. Вызов отложенной обработки может иметь высокий, средний и низкий приоритет; по умолчанию присваивается средний приоритет. Драйвер может изменить значение приоритета. Вызов отложенной обработки

с высоким приоритетом размещается в начало очереди, а DPC с низким и средним приоритетом — в конец очереди.

- Обычно вызов отложенной обработки выполняется на том же процессоре, что и процедура обслуживания прерывания, что можно изменить с помощью драйвера.
- Если драйвер уже поместил DPC в очередь, следующий запрос на размещение DPC в очереди просто игнорируется. При выполнении DPC выясняется, существует ли несколько рабочих элементов, например, при многократной обработке прерываний, когда каждое прерывание требует наличия отдельного рабочего элемента.
- Вызов отложенной обработки может быть размещен в очереди другого процессора, если очередь DPC текущего процессора превышает определенное значение. Ядро Windows NT периодически пытается выполнить вызов отложенной обработки, генерируя программные прерывания.
- Вызов отложенной обработки не может быть выгружен на диск.

1.5.3 Асинхронный вызов процедуры

Асинхронный вызов процедуры (asynchronous procedure call — APC) немного похож на вызов отложенной обработки, но существуют и заметные различия. Как и вызов отложенной обработки, APC выполняется на уровне привилегий, превышающем уровень привилегий обычного кода. В отличие от вызова отложенной обработки, выполняемого в контексте произвольного процесса, асинхронный вызов процедуры всегда выполняется в контексте определенного процесса. Таким образом, асинхронный вызов процедуры требует больших затрат, чем вызов отложенной обработки, так как приходится сохранять и восстанавливать большее количество параметров. Читателю, знакомому с операционными системами UNIX, асинхронные вызовы процедур напомнят процедуры обработки сигналов UNIX.

Существует два типа APC: вызов в режиме ядра и вызов в пользовательском режиме. Асинхронный вызов процедуры в режиме ядра связан с драйвером или другим кодом режима ядра и обычно используется для передачи данных, например для копирования данных из буфера ядра в пользовательский буфер. Помните, что пользовательский буфер должен быть доступен в контексте процесса, который владеет буфером.

Код пользовательского режима тоже может использовать асинхронный вызов процедур. Для этого необходим прикладной интерфейс программирования QueueUserAPC, который рассматривается в документации к набору Platform SDK. Асинхронные вызовы процедур в пользовательском режиме

предоставляются только тогда, когда поток получает предупреждение, например при блокировании в результате вызова функций `WaitForSingleObject` или `WaitForMultipleObject`. Подробная информация об этих функциях доступна в документации Platform SDK. Достаточно сказать, что эти функции позволяют организовать синхронизацию потоков.

Асинхронный вызов процедуры может быть блокирующим, например для выполнения специального ввода-вывода. Вызовы помещаются в очередь, соответствующую потоку, т.е. существует несколько очередей асинхронных вызовов процедур.

1.6 Драйверы и буферы ввода-вывода

В этом разделе рассматриваются буферы ввода-вывода, которые уже упоминались ранее в главе. Драйверы используют буферы для осуществления ввода-вывода и управления им (IOCTL). Для этого драйверы посредством соответствующего объекта указывают предпочтительный способ ввода-вывода. Существует три поддерживаемых драйверами Windows NT типа ввода-вывода: буферизированный, прямой и небуферизированный. Эти методы рассматриваются в данном разделе.

1.6.1 Буферизированный ввод-вывод

Буферизированный ввод-вывод обычно используется для передачи меньших объемов данных, так как в процессе ввода-вывода участвуют операции копирования данных. После того как приложение отправило запрос на ввод-вывод, диспетчер ввода-вывода проверяет соответствие прав доступа приложения к предоставленному буферу ввода-вывода (*соответствие* прав доступа предполагает, что приложение имеет права на чтение и запись, а также имеет доступ к буферу необходимого размера, указанного в запросе на ввод-вывод). Диспетчер ввода-вывода выделяет буфер в невыгружаемой памяти и в случае запроса на запись копирует данные из буфера приложения в выделенный буфер. Этот буфер передается драйверу.

Драйвер не обязан заботиться о контексте потока, так как буфер находится в невыгружаемой памяти и действителен в контексте любого процесса или потока. Драйвер выполняет необходимую операцию ввода-вывода. Для операции чтения драйвер копирует данные в полученный буфер. Кроме того, драйвер может "предположить", что в контексте виртуального адресного пространства буфер будет непрерывным, но, как и в случае других буферов Windows NT, непрерывность буфера в виртуальном адресном пространстве не гарантирует непрерывности в физическом адресном пространстве.

На этом этапе драйвер завершает обработку IRP, и ответственность за копирование данных из невыгружаемого буфера в буфер приложения возлагается на диспетчер ввода-вывода. Эта операция копирования должна выполняться в контексте процесса, который отправил запрос на ввод-вывод. Кроме того, диспетчер ввода-вывода должен освобождать буфер, выделенный в невыгружаемой области памяти.

1.6.2 Прямой ввод-вывод

Прямой ввод-вывод (Direct I/O) несколько более запутан, чем буферизированный, однако намного эффективнее при выполнении операций ввода-вывода для больших массивов данных. Диспетчер ввода-вывода выполняет базовые проверки, например проверяет разрешения приложения на доступ к буферу посредством области ввода-вывода желательного объема. Буфер в памяти, независимый от процесса, описывается для драйвера средствами структуры данных, которая называется *список дескрипторов памяти* (memory descriptor list — MDL). Адрес буфера используется в качестве общесистемного виртуального адресного пространства.

Операционная система Windows NT предоставляет процедуры драйверов, которые позволяют получать доступ к различным полям списка дескрипторов памяти. Создателям драйверов рекомендуется использовать список дескрипторов памяти в качестве целостного элемента. Дополнительная информация по использованию процедур работы со списками дескрипторов памяти приводится в документации к инструментарию создания драйверов (DDK) Windows NT. Процедуры, описанные в DDK, предоставляют следующие возможности:

- блокирование и разблокирование буфера приложения;
- связывание заблокированного буфера с виртуальным адресом, который доступен из контекста любого потока;
- сбор информации, необходимой для выполнения операции ввода-вывода с прямым доступом к памяти в буфер или из него, который в действительности представляет собой последовательность потенциально несоседних физических страниц памяти.

Прямой ввод-вывод чаще всего используется в драйверах ввода-вывода, например драйверах управления дисками и приводами на магнитной ленте.

1.6.3 Небуферизированный ввод-вывод

Этот тип ввода-вывода позволяет избавиться от генерирования дополнительных данных, что свойственно для буферизированного ввода-вывода

(операций копирования данных и выделение/освобождение буфера) и прямого ввода-вывода (создания и уничтожения списка дескрипторов памяти), но за это приходится расплачиваться ограниченностью применения данного типа ввода-вывода. При небуферизированном вводе-выводе драйверу непосредственно передается адрес буфера запросившего приложения. Внимательный читатель быстро догадается, что, поскольку виртуальный адрес имеет смысл только в контексте определенного процесса или потока, драйвер должен вызываться в контексте запросившего приложения. Более того, драйвер должен выполнить операцию в этом же контексте (т.е. драйвер не может поместить запрос в очереди для последующего выполнения в произвольном контексте).

Это ограничение определяет ситуации использования данного метода ввода-вывода, который не применяется большинством драйверов, исключение составляют, например, драйверы файловой системы. Дело в том, что последние всегда вызываются в контексте процесса, который запрашивает операцию ввода-вывода. Кроме того, небуферизированный ввод-вывод поддерживает копирование между кэшем и буферами данных, так как управлять буферами (связывание адресов) не требуется.

Может сложиться неверное впечатление, что драйвер должен выбрать один из описанных способов ввода-вывода и использовать только его. Драйвер, который выполняет операции управления вводом-выводом (IOCTL), может использовать один метод ввода-вывода для обработки обычных пакетов IRP и совершенно другой метод для операций управления вводом-выводом, которые определяются частным образом между драйвером и соответствующим приложением. Конечно, даже драйвер на нижних уровнях стека драйверов, который не имеет информации о контексте выполнения, не обязательно использует небуферизированный ввод-вывод в частных операциях управления вводом-выводом.

Поскольку здесь затронута тема частного управления вводом-выводом, стоит упомянуть, что компания Microsoft настойчиво советует не применять такой способ управления, особенно при наличии более приемлемой альтернативы. Основная проблема частного управления вводом-выводом заключается в сложности проверки "жизнеспособности" кода драйвера методом намеренной передачи некорректных буферов процедуре управления вводом-выводом, что делается для проверки работоспособности драйвера. Для передачи некорректного буфера необходимо иметь информацию о правильном размере буфера, выравнивании и граничных условиях, которые предполагаются в коде управления вводом-выводом, а для частных операций управления вводом-выводом эти параметры каждый раз имеют другие значения.

1.7 Иерархия драйверов систем хранения и типы драйверов

Как описывалось в предыдущем разделе, Windows NT основана на архитектуре, в которой драйверы формируют многоуровневую иерархию. Преимущество такой архитектуры состоит в расширяемости архитектуры и возможности добавления новых драйверов на любой уровень иерархической структуры. Таким образом, благодаря поуровневому размещению драйверов можно реализовать различные функциональные возможности. В контексте выполняемого модуля Windows NT все драйверы имеют аналогичную структуру, поэтому функции драйвера используются схожим образом вне зависимости от его типа.

В этом разделе представлен обзор стека драйверов устройств хранения Windows NT. Обратите внимание, что речь идет только о базовых, а не обо всех драйверах, связанных с подсистемой хранения данных. Например, драйверы, связанные со службами удаленного хранения (RSS), рассматриваются в главе 7.

На рис. 1.5 демонстрируется стек драйверов подсистемы хранения данных Windows NT. Обратите внимание: здесь представлена многоуровневая архитектура драйверов, однако в зависимости от ситуации те или иные уровни приобретают более важное значение. Ниже приведены примеры подобных ситуаций.

- При вводе-выводе данных на физический диск, подключенный через интерфейс IDE или SCSI, необходимы уровни класса и порта, а также уровни файловой системы и управления томами. Все эти уровни рассматриваются далее в главе.
- При вводе-выводе данных посредством накопителя на магнитной ленте уровни управления томами и файловой системы не требуются.

В следующих подразделах рассматриваются драйверы шины, порта, класса, управления томами, файловой системы и фильтрации, представленные на рис. 1.5.

1.7.1 Драйверы шины

Драйвер шины Windows NT предоставляет функции шины другим драйверам. Термин *шина* используется в универсальном смысле, обозначая любое виртуальное или физическое устройство, к которому подключаются другие устройства. Драйверы шины необходимы для поддержки процедур перебора, которые вызываются диспетчером Plug and Play для перечисления устройств, подключенных к шине. Кроме того, от драйверов шины требуется

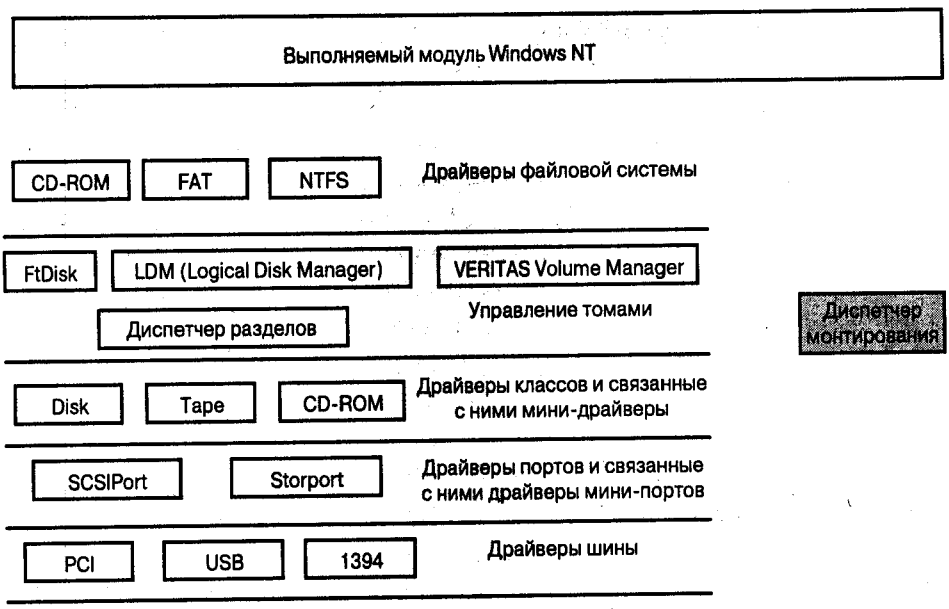


Рис. 1.5. Стек драйверов хранения Windows NT

предоставление кода обработки PnP, а также пакетов IRP для управления энергопитанием. Компания Microsoft предоставляет драйверы ввода-вывода для всех физических шин персональных компьютеров (например, SCSI, PCI, 1394, USB), хотя независимые поставщики оборудования также могут по мере необходимости предоставлять собственные драйверы шин. Драйвер шины создает объект физического устройства (physical device object — PDO) для каждого устройства, указанного процедурой перечисления устройств.

1.7.2 Драйверы портов

Драйвер порта реализует специфичные для устройства функциональные возможности и изолирует драйвер класса от влияния особенностей аппаратного обеспечения. Драйвер порта *должен* реализовать набор указанных функций для драйвера класса и *может* реализовать дополнительные возможности. Драйвер порта получает пакеты IRP и передает блоки запросов SCSI с встроенными блоками дескрипторов команд драйверу мини-порта, который динамически подключается к драйверу порта. Драйверы мини-портов не создают объектов устройств, а используют созданные драйверами порта. Как отмечалось в разделе 1.4.2, драйверы порта создают объект физического устройства, необходимый для взаимодействия с устройством.

В Windows NT предоставляются некоторые драйверы порта, включая SCSIPort и IEEE 1394. В свою очередь, Windows Server 2003 поставляется

с дополнительным драйвером порта, который называется Storport. На данный момент достаточно сказать, что драйвер SCSIPort используется для работы с устройствами SCSI-2 и более старыми устройствами, а драйвер Storport — с устройствами SCSI-3 и Fibre Channel. Дополнительная информация о драйвере Storport приводится в главе 2.

Драйверы порта, в свою очередь, содержат драйверы мини-портов, которые предоставляются независимыми поставщиками оборудования. Мини-порт обеспечивает функции уровня устройства, которые зависят от конкретного поставщика и не обеспечиваются драйвером порта. Драйверы мини-порта создаются с помощью инструментария разработки драйверов Windows NT.

1.7.3 Драйверы классов

Драйвер класса обеспечивает универсальную, не зависящую от устройства поддержку целого диапазона устройств. Драйвер класса зависит от драйверов мини-класса или мини-порта, которые предоставляют конкретные функции устройства. Драйверы класса хранения данных используются для работы с устройствами SCSI и устройствами, подключаемыми через другие интерфейсы. Ниже перечислены другие функции драйверов классов.

- Создание объекта функционального устройства (FDO). Такой объект необходим для непосредственного использования устройства. В FDO могут содержаться такие данные, как структура организации диска (таблица разделов) или номер региона DVD.
- Проверка действительности параметров IRP.
- Повторная отправка запросов, обработка которых завершилась неудачно.

В частности, драйверы класса хранения данных выполняют описанные ниже операции.

- Разбивка больших запросов чтения/записи (полученных с помощью IRP `IRP_MJ_READ` и `IRP_MJ_WRITE`) на меньшие множественные запросы, которые соответствуют возможностям физического адаптера шины.
- Получение пакетов IRP и их трансляция в подходящие блоки запросов SCSI (SCSI request blocks — SRB), которые содержат встроенные блоки дескрипторов команд. После этого блоки запросов SCSI отправляются следующему драйверу в цепочке стека, который может оказаться драйвером фильтрации или драйвером порта. Блоки запросов SCSI при этом будут незавершенными, так как драйвер класса не заполняет информацию об адресации в блоке запроса SCSI и в этом зависит от драйвера более низкого уровня.

- Участие в управлении питанием, добавление и удаление устройств, а также установка тайм-аута для ввода-вывода данных на устройства. Другими словами, драйверы класса устройств хранения активно принимают участие в реализации PnP и управлении энергопитанием.

Драйверы класса взаимодействуют с драйверами порта на следующем нижнем уровне стека хранения данных. Интерфейс между драйверами классов и драйверами порта формируется посредством частного интерфейса управления вводом-выводом, а также обмена блоками запросов SCSI. Некоторые элементы блоков запроса SCSI используются только для интерфейса класса или интерфейса порта, а некоторые предназначены для использования драйвером порта.

К драйверам класса устройств хранения в Windows NT относятся, например, драйверы дисков, приводов компакт-дисков и накопителей на магнитной ленте; они могут работать с различными устройствами, в том числе подключенными к шинам SCSI, IDE, USB и 1394.

В контексте диспетчера ввода-вывода драйвер класса устройства хранения — это такой же драйвер, как и все остальные, поэтому он должен соответствовать определенным требованиям, например предоставлять процедуры инициализации ввода-вывода, выгрузки и завершения.

Драйверы класса устройства хранения обычно работают аналогично драйверам шины, перечисляя дочерние устройства. Хорошим примером может служить драйвер класса диска (`disk.sys`), который считывает таблицу разделов диска и создает объекты устройств для каждого найденного дискового раздела.

Некоторые драйвера класса определяют интерфейс драйвера мини-класса. Драйвер мини-класса, который обычно создается независимым поставщиком оборудования, представляет собой динамически подключаемую библиотеку режима ядра, которая взаимодействует с драйвером класса, предоставляемым Microsoft. Драйвер мини-класса регистрирует адаптеры аппаратного обеспечения с помощью драйвера класса, а последний, в свою очередь, создает объект устройства для каждого зарегистрированного адаптера. Драйвер мини-класса не содержит объектов устройств и использует объект устройства, который создается драйвером класса. Обычно драйвер мини-класса помогает вносить дополнительную информацию в блоки запросов SCSI, которые создаются драйвером класса. В качестве примера можно привести драйвер мини-класса для накопителя на магнитной ленте.

Интересно отметить, что Microsoft добавила в Windows 2000 новую библиотеку, которая называется `ClassPnP`. Эта библиотека реализует функциональные возможности PnP, общие для всех драйверов класса. При этом некоторые функции полностью реализуются драйвером класса. Для других функ-

ций драйвер, который использует библиотеку класса, должен предоставить процедуры обратного вызова, используемые драйвером класса при необходимости. Все драйверы класса, предоставляемые Microsoft (драйверы дисков, накопителей на магнитной ленте и драйверы приводов компакт-дисков), пользуются услугами библиотеки ClassPnP (она реализована в виде файла `classpnp.sys`). Та же ситуация сохранилась и в Windows XP/Windows Server 2003.

1.7.4 Дерево устройств Windows NT для устройств хранения данных

Описание различных уровней, показанных на рис. 1.5, все еще остается неполным. Однако немного отвлечемся перед обсуждением уровней управления томами и файловыми системами и рассмотрим дерево устройств, которое создается операционной системой. Знакомство с объектной моделью дерева устройств упрощает понимание тем, связанных с многопоточным вводом-выводом данных в отказоустойчивых устройствах хранения (см. главу 9) и с точками повторной обработки (см. главу 6). Поэтому имеет смысл рассмотреть данную тему сейчас, пока еще не забылась информация об объектах устройств, драйверах классов и портов, приведенная в предыдущих разделах.

Как уже отмечалось, технология PnP играет важную роль в идентификации устройств. Интерфейс PnP загружает драйверы шины и инициирует обнаружение устройств, подключенных к этой шине. При обнаружении устройств PnP используется для загрузки соответствующих драйверов. В частности, перечисление устройств начинается с драйвера виртуальной (корневой) шины. Драйвер корневой шины отвечает за перечисление устаревших драйверов DOS и обычно используется для идентификации шин PCI. Драйверы, загруженные драйвером корневой шины, часто называют прошедшими *корневое перечисление*. К ним относится, например, драйвер шины MPIO (рассматривается в главе 9).

При загрузке драйверов и перечислении ими соответствующих устройств списки устройств передаются обратно интерфейсу PnP. При этом формируется дерево устройств, демонстрирующее схему их логического и физического взаимодействия. Обратите внимание, что PnP может только создать дерево; граф устройств не поддерживается. Таким образом, при использовании PnP дочерний узел может содержать только один родительский узел.

На рис. 1.6 в верхнем левом углу иллюстрируется несложная системная конфигурация — сервер Windows NT с одним адаптером шины и одним диском, подключенным к этому адаптеру. Другие периферийные устройства, которые обычно подключены к компьютерам под управлением Windows NT, на рисунке не показаны.

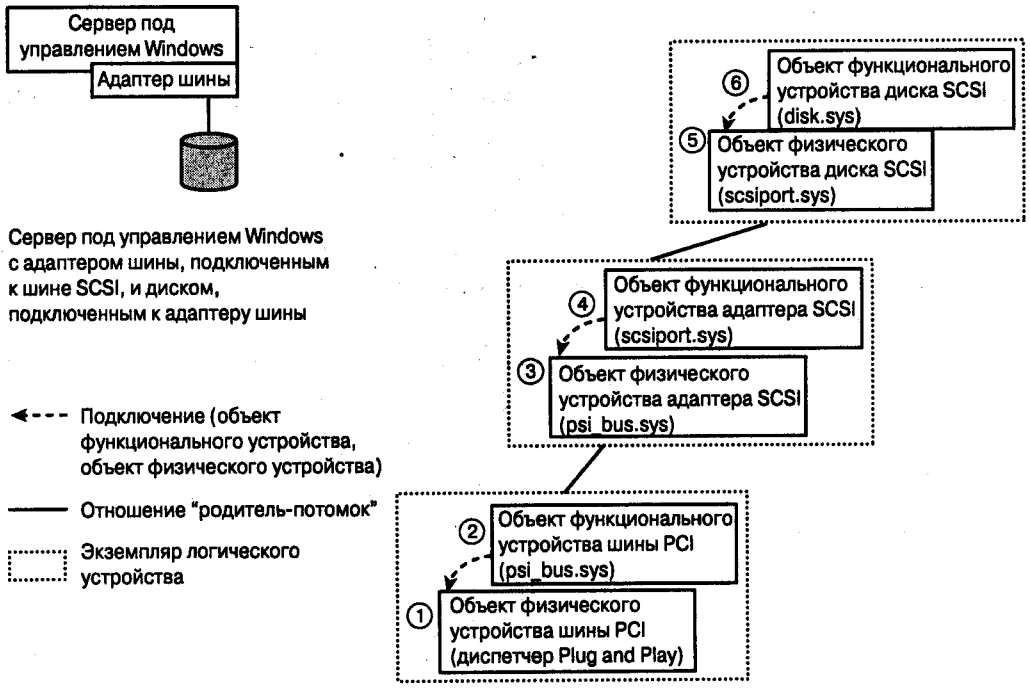


Рис. 1.6. Дерево драйверов устройств

На рис. 1.6 представлен подробный обзор лишь некоторых компонентов с рис. 1.5. В частности, на рис. 1.6 не показаны уровень управления томами и уровень файловых систем, в отличие от уровней драйверов классов и портов на рис. 1.5. Начиная с нижней правой области рис. 1.6, последовательность формирования дерева устройств выглядит так, как описано ниже.

1. Выполняемый модуль Windows NT (в частности, диспетчер PnP) создает объект физического устройства для драйвера шины PCI.
2. Драйвер шины PCI, в свою очередь, создает объект функционального устройства для шины PCI и подключает его к объекту физического устройства.
3. Кроме того, драйвер шины PCI идентифицирует (перечисляет) адаптеры и обнаруживает адаптер шины SCSI, после чего создает для него объект физического устройства. Чтобы не усложнять диаграмму на рис. 1.6, другие адаптеры, подключенные к шине PCI, в данный момент не рассматриваются.
4. Диспетчер PnP загружает драйвер SCSIPort и после инициализации последнего вызывает его через входную точку входа `AddDevice`. При этом

драйверу в качестве параметра передается объект физического устройства, созданный драйвером шины PCI. Драйвер SCSIPort создает объект функционального устройства для устройства шины. Драйвер порта подключает созданный объект функционального устройства к объекту физического устройства, созданному драйвером шины PCI.

5. Как уже отмечалось, драйвер может функционировать по-разному, и в данном случае драйвер SCSIPort действует в качестве драйвера шины, перечисляя устройства, которые подключены к шине SCSI. Драйвер обнаруживает диск и сообщает о нем диспетчеру PnP, который загружает драйвер класса диска (`disk.sys`). После инициализации драйвер класса диска вызывается через входную точку `AddDevice` и в качестве параметра получает объект физического устройства, который создается драйвером SCSI.
6. Драйвер класса диска создает объект функционального устройства для диска, подключенного к адаптеру шины, и подключает его к объекту физического устройства, созданному драйвером SCSIPort. На самом деле адаптер шины может быть подключен к промышленному коммутатору Fibre Channel, за счет чего адаптер главной шины перечислит намного больше устройств. В данном случае описываемая ситуация намеренно упрощена. Кроме того, стек устройств на этом не заканчивается. Дальнейшее обсуждение стека устройств, которое затрагивает уровни управления томами и файловых систем, приводится в следующем разделе.

Обратите внимание на взаимодействие двух драйверов для создания экземпляров таких логических устройств, как драйвер шины PCI и драйвер SCSIPort. Это вполне логично, так как устройство, с одной стороны, подключается к шине PCI, а с другой — обеспечивает работу интерфейса шины SCSI. Таким образом, одно устройство имеет характеристики как устройства PCI, так и устройства SCSI, поэтому оно должно обрабатываться одновременно драйвером шины PCI и драйвером SCSIPort.

1.7.5 Уровень управления томами

На этом этапе можно вернуться к рис. 1.5 и рассмотреть уровень управления томами. Тома — это логические элементы, которые создаются, чтобы упростить управление устройствами хранения данных. Физические диски, которые подключаются через интерфейсы IDE или SCSI, могут быть логически разбиты на *разделы* (partitions). Раздел — это физически непрерывный набор секторов диска. Из разделов определенным образом формируется том. Такая комбинация разделов может предоставлять дополнительные возмож-

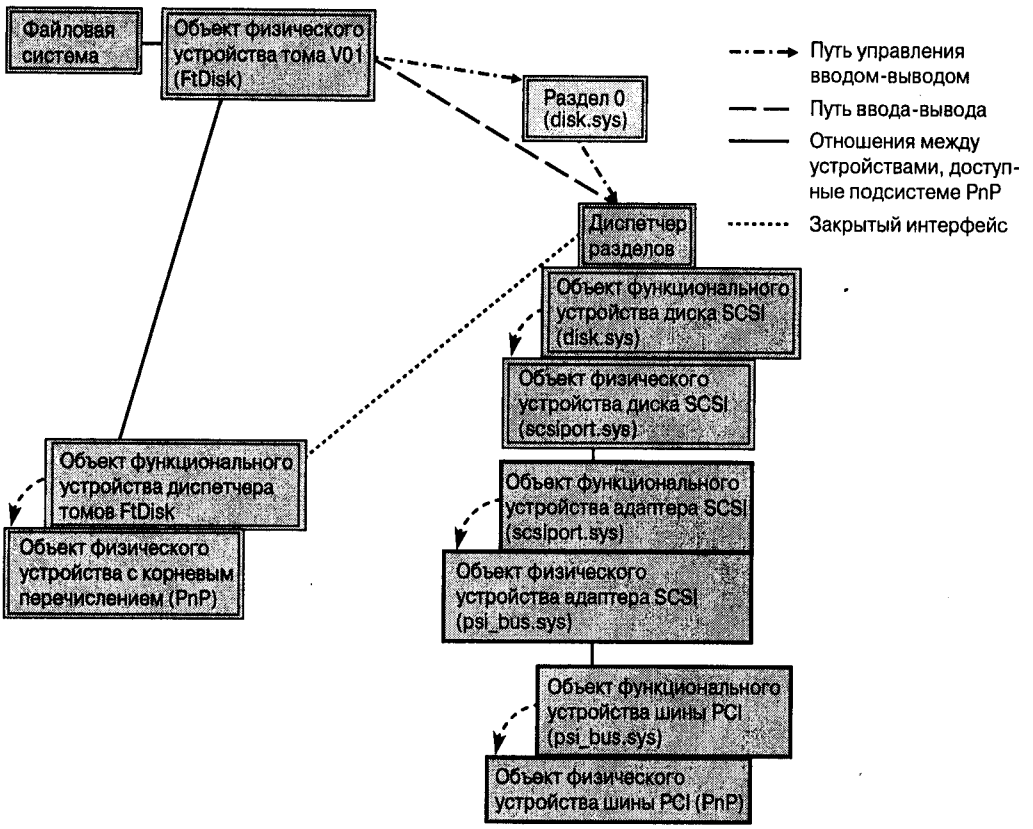


Рис. 1.7. Дерево объектов устройств для стека томов

ности: например, несколько разделов могут быть объединены для создания тома, размер которого превышает размер каждого из физических дисков. Еще одним примером может быть создание зеркального тома на основе двух разделов, размер которых совпадает. Дисковые тома рассматриваются более подробно в главе 6. На данном этапе эта тема затрагивается для описания схемы управления томами в Windows NT с помощью драйвера программного устройства.

В Windows 2000, Windows XP и Windows Server 2003 поддерживается три различных диспетчера томов: FtDisk, Microsoft Logical Disk Manager и VERITAS Volume Manager. Все они подробно рассматриваются в главе 6. В данном случае в качестве примера будет использоваться базовый диспетчер томов Microsoft FtDisk Manager. Дерево устройств с другими диспетчерами томов также описано в главе 6. Дерево устройств на рис. 1.7 иллюстрирует системную конфигурацию с одним подключенным диском SCSI, содержащим два раздела, которые, в свою очередь, формируют один том.

Для понимания принципов работы диспетчеров томов рассмотрим рис. 1.7, начиная с нижнего правого угла. Подсистема PnP и драйверы шины PCI взаимодействуют для создания объектов физического и функционального устройств для шины PCI. После идентификации устройств, подключенных к шине PCI, драйвер шины PCI создает объект физического устройства для адаптера шины SCSI. Драйвер SCSIPort создает объект функционального устройства для адаптера шины SCSI. Затем драйвер SCSIPort и драйверы класса диска создают объект физического устройства и объект функционального устройства для одного диска, который подключен к шине SCSI. До этого момента на рис. 1.7 вкратце дублировалось содержимое рис. 1.6.

Диспетчер разделов представляет собой драйвер фильтрации более высокого уровня (драйверы фильтрации рассматриваются в разделе 1.7.7), который регистрируется в подсистеме Windows NT PnP для получения уведомлений о создании драйвером класса диска новых объектов устройств. Диспетчер разделов появился впервые в Windows 2000 и используется в Windows XP и Windows Server 2003. Диспетчер разделов взаимодействует с диспетчерами томов (на рис. 1.7 это диспетчер FtDisk) с помощью частных интерфейсов и передает уведомление о создании устройства диспетчеру разделов. Обнаружив все дисковые разделы, которые формируют том, диспетчер тома создает объект устройства, представляющего данный том. Диспетчер разделов обеспечивает уведомление подсистемы PnP относительно удаления объекта устройства или раздела (например, при удалении раздела). Диспетчер разделов связывается с драйвером FtDisk для предоставления последнему информации о динамически добавляемых и удаляемых разделах.

На рис. 1.7, в отличие от предыдущих рисунков, впервые показан диспетчер разделов, который следит за пакетами IRP в процессе ввода-вывода и обеспечивает завершение их обработки. Обнаружив завершение обработки QDR (`IRP_MN_QUERY_DEVICE_RELATIONSHIPS`), диспетчер разделов незаметно удаляет дополнительную информацию об обнаруженном устройстве — в данном случае это объект устройства для раздела 0, созданный драйвером класса диска `disk.sys`. Таким образом объект устройства раздела 0 никогда не обнаруживается подсистемой PnP. Именно поэтому объект устройства для раздела 0 закрашен не так, как все остальные объекты на рис. 1.7.

Диспетчер разделов передает информацию об обнаруженных объектах устройств зарегистрированным диспетчерам томов. На данный момент обсуждение будет ограничено одним диспетчером томов. В главе 6 рассматривается аналогичная ситуация, но уже при участии нескольких диспетчеров томов. Диспетчер томов проверяет устройства, представленные объектами устройств, которые "украдены" диспетчером разделов, и принимает или отвергает владение этими объектами устройств. В этом примере драйвер FtDisk подтверждает свое владение объектами устройств. Затем диспетчер FtDisk

проверяет конфигурацию тома и устанавливает, что том сформирован посредством одного раздела, а также определяет владельца соответствующего раздела. На этом этапе драйвер FtDisk создаст объект устройства для тома (на рис. 1.7 он называется "том V01"), после чего можно будет монтировать файловую систему этого тома. Подробности монтирования файловой системы рассматриваются в главе 6.

Следует отметить, что здесь на самом деле рассматриваются два отдельных стека устройств. Один стек представляет логический компонент — том, а второй включает в себя физические устройства системы, например шину PCI, адаптер SCSI и жесткий диск. Диспетчер томов действует как мост между двумя стеками.

На рис. 1.7 драйвер FtDisk отправляет все обработанные пакеты IRP непосредственно драйверу класса диска. Конечно, драйвер FtDisk преобразует смещения относительно тома в смещения относительно диска перед отправкой пакетов IRP. Такие операции ввода-вывода показаны толстыми штриховыми линиями. Тонкие пунктирные линии отображают частный интерфейс между диспетчерами томов и разделов. Кроме того, драйвер FtDisk отправляет необработанные сообщения управления вводом-выводом непосредственно объекту устройства раздела. На рис. 1.7 это показано штрих-пунктирной линией.

1.7.6 Драйверы файловой системы

Драйверы файловой системы представляют собой драйверы устройств Windows NT, которые реализуют возможности файловой системы. Хотя файловая система содержится на физическом носителе, например на компакт-диске или жестком диске, драйверы файловой системы рассматриваются в качестве логических, поскольку не используются для непосредственного управления аппаратным обеспечением. Драйверы файловых систем полагаются на драйверы портов и классов для обеспечения ввода-вывода данных на диск и с него. Драйвер файловой системы обычно получает пакет IRP для выполнения запроса и осуществляет одну из двух операций.

1. Заполняет следующий фрагмент стека пакета IRP необходимой информацией для завершения ввода-вывода. После этого пакет IRP отправляется драйверу класса.
2. Создает последовательность пакетов IRP для выполнения запрошенного ввода-вывода.

Драйвер файловой системы содержит метаданные на самом носителе. В метаданные входит информация о разрешениях доступа к файловой системе и таблица размещения файлов (расположение файлов на диске). Драйвер

файловой системы получает пакет IRP, который указывает на определенную операцию по отношению к файлу, считывает необходимые метаданные и отправляет запрос IRP, который уже относится к блоку диска, а не к файлу. Операционная система поставляется с драйверами файловых систем NTFS, UDFS и FAT.

Создание драйвера файловой системы или драйверов фильтрации для файловых систем в Windows NT 3.x поначалу считали "черной магией". Впоследствии Microsoft предоставила инсталляционный инструментарий файловых систем (Installable File System Kit), который содержит необходимые заголовочные файлы, документацию и примеры создания файловых систем и драйверов фильтрации файловых систем.

Драйверы как файловых систем, так и фильтрации файловых систем должны обеспечивать поддержку пакетов IRP PnP, включая управление питанием, удаление носителя и самого устройства хранения (например, внешнего дисковода на гибких дисках, подключаемого к шине USB).

1.7.7 Драйверы фильтрации

Драйверы фильтрации размещены выше других объектов устройств в иерархической структуре драйверов и выполняют предварительную и/или последующую обработку запросов ввода-вывода для модификации работы системы. Драйверы фильтрации обычно используются для выполнения перечисленных ниже задач.

- Поддержка модульной структуры, например посредством драйвера фильтрации музыкальных компакт-дисков.
- Добавление функциональных возможностей, например записи компакт-дисков на соответствующих устройствах.
- Добавление функциональных возможностей файловой системе, например драйверов фильтрации для шифрования, точек повторной обработки и SIS — Single Instance Storage (рассматривается в главе 6).
- Добавление функций шины. Например, в виде поддержки шины AGP или расширений ACPI BIOS.
- Изменение процесса ввода-вывода для того, чтобы он соответствовал особенностям функционирования аппаратного обеспечения, например разбивка пакетов ввода-вывода на меньшие фрагменты.

Драйверы фильтрации всегда создают объект устройства, который подключен к объекту функционального устройства или к объекту физического устройства (эти объекты рассматриваются в разделе 1.4.2). Объект устройства необходим драйверу фильтрации для получения пакетов IRP и выполнения предварительной и/или последующей обработки запросов ввода-вывода.

Некоторые драйверы фильтрации создают вторичный объект устройства, который часто называется объектом управляющего устройства (control device object — CDO), так как он используется для отправки управляющей информации драйверу фильтрации с помощью соответствующего модуля управления. Драйверы фильтрации, которые подключаются к объекту функционального устройства, созданному драйвером класса, называются *драйверами фильтрации верхнего уровня*. В свою очередь, драйверы фильтрации, которые подключаются к объекту физического устройства, размещенному ниже в стеке и созданному драйвером порта, называются *драйверами фильтрации нижнего уровня*.

Драйверы фильтрации нижнего уровня отличаются более сложной структурой, чем драйверы верхнего уровня. К одной из технических проблем относится выбор сообщаемых ошибок, а также операций ввода-вывода, ошибки которых сообщаются. Еще одна проблема — отсутствие готовых примеров драйверов нижнего уровня, поскольку все доступные примеры описывают драйверы верхнего уровня. Иногда (достаточно редко) драйверы нижнего уровня предоставляют функции протокольного преобразователя или функции, позволяющие обойти некоторые ограничения в работе аппаратного устройства.

Драйверы фильтрации существовали в Windows NT с момента появления ее первой коммерческой версии. Пакет Windows NT Installable File System Kit (`http://www.microsoft.com/ddk/IFSKit`) представляет собой неплохой справочник для разработчиков, в котором подробно документируется архитектура драйверов фильтрации.

Начиная с Windows 2000, в процесс загрузки драйверов фильтрации были внесены значительные изменения. Ранее дополнительная работа по проверке правильности загрузки драйвера ложилась на плечи создателя драйвера. Если драйвер загружался слишком рано, объект устройства, к которому должен подключиться драйвер, мог еще не существовать. Если драйвер загружался слишком поздно, устройство могло оказаться занятым другим драйвером; это приводило к тому, что драйвер фильтрации в стеке драйверов помещался выше, чем было необходимо. В Windows 2000 создатель драйвера указывает размещение драйвера выше или ниже определенного объекта физического устройства или объекта функционального устройства, а диспетчер Plug and Play загружает драйвер в подходящее время.

Похоже, что компания Microsoft чересчур упростила процесс создания драйвера фильтрации. Цепь стека драйверов стала чрезмерно "переполненной", что подразумевает проблемы с производительностью и загрузкой памяти (каждый пакет IRP должен содержать больше фрагментов стека, а пакеты IRP размещаются в невыгружаемой памяти). Достаточно посмотреть на список драйверов фильтрации, которые загружаются в операционной системе:

- драйвер фильтрации шифрованной файловой системы;
- драйвер фильтрации, используемый для управления иерархическим хранением и для служб удаленного хранения;
- драйвер фильтрации SIS для служб удаленной установки (RIS);
- драйверы фильтрации для точек повторной обработки от сторонних поставщиков;
- драйверы фильтрации для антивирусного программного обеспечения.

С другой стороны, создание драйвера фильтрации бывает достаточно сложным. Представьте себе драйвер, который должен выполнять шифрование данных при записи на диск и дешифрацию при считывании. Все, что на самом деле необходимо драйверу, это доступ к буферам до их записи и после считывания. Но создатель драйвера не имеет возможности просто зарегистрировать функцию обратного вызова для быстрого выполнения необходимой функции и вынужден сталкиваться с различными трудностями, в частности с обработкой отмененных пакетов IRP.

1.8 Ввод-вывод типичного приложения хранения данных

Теперь соберем вместе все описываемые концепции и тщательным образом рассмотрим типичное приложение хранения данных. Это позволит разобраться с различными компонентами Windows NT, которые описывались ранее в главе, а также понять принципы их использования.

Рассматриваемое нами приложение будет просто считывать данные из файла. Файл размещен на томе, который управляется диспетчером томов FtDisk. Поскольку эта конфигурация идентична конфигурации, показанной на рис. 1.7, дерево объектов устройств не изменится. На рис. 1.8 представлена упрощенная версия дерева объектов устройств с рис. 1.7. Чтобы уменьшить объем предлагаемой информации, взаимодействие файловой системы с диспетчером кэша во внимание не принимается, т.е. предполагается, что файл не кэшируется.

Ниже описаны операции, которые приводятся на рис. 1.8.

1. Приложение, например резервного копирования, выдает запрос на чтение. После ряда стандартных проверок подсистема ввода-вывода перенаправляет запрос с помощью пакета IRP соответствующей файловой системе.
2. Файловая система определяет, что ей необходимо получить данные тома. Драйвер файловой системы создает пакет IRP с необходимыми па-

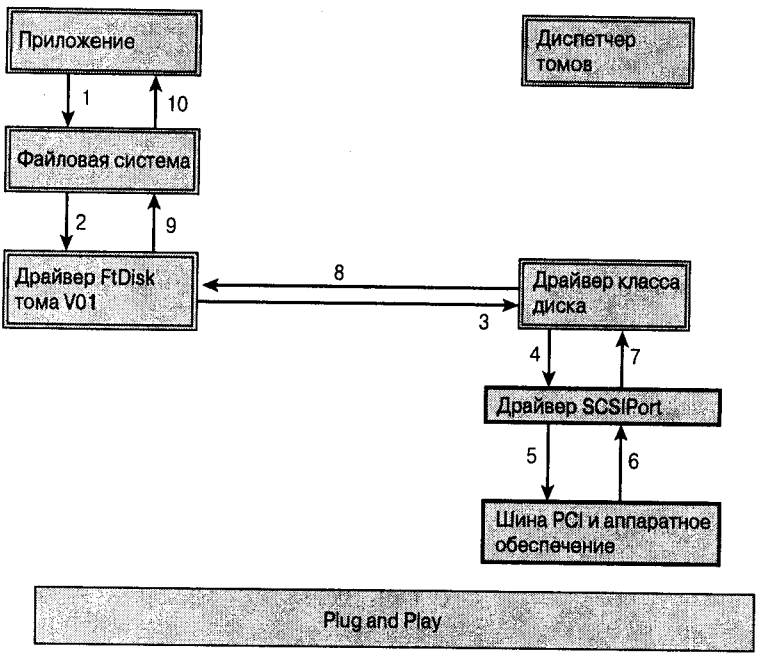


Рис. 1.8. Операция чтения на томе

раметрами, включая смещение в пределах тома, где должна быть выполнена операция чтения, и объем операции чтения. Кроме того, драйвер файловой системы указывает процедуру завершения ввода-вывода. После этого драйвер файловой системы отправляет пакет IRP диспетчеру томов.

3. Диспетчер томов преобразует значение смещения тома в значение смещения на диске и заполняет соответствующий пакет IRP. Затем средствами диспетчера ввода-вывода пакет IRP отправляется драйверу класса диска.

4. После получения пакета IRP драйвер класса диска заполняет процедуру завершения в собственном элементе стека в пакете IRP. Обычно драйвер класса диска создает еще один пакет IRP (он называется *ассоциированным*) для выполнения ввода-вывода. Драйвер класса диска заполняет необходимую информацию в ассоциированном пакете IRP, указав блоки запросов SCSI для запроса чтения. Обратите внимание, что блок запроса SCSI представляет собой просто фрагмент данных пакета IRP, поэтому структура данных все еще остается пакетом IRP. После этого средствами диспетчера ввода-вывода драйвер класса диска отправляет пакет IRP драйверу порта.

5. Драйвер SCSIPort размещает пакет IRP в очереди, запрашивая некоторые операции у драйвера мини-порта, который управляет адаптером SCSI. На этом этапе пакет IRP отмечается как ожидающий выполнения и отправляется назад. Обычно диспетчер ввода-вывода обрабатывает пакет IRP в порядке, обратном только что описанному, т.е. после драйвера порта следует драйвер класса, затем диспетчер томов, а за ним файловая система. На каждом этапе пакет IRP отмечается как ожидающий выполнения. На определенном этапе ввод-вывод будет отправлен на физическое устройство средствами шины PCI.
6. В какой-то момент ввод-вывод будет завершен. Чтобы упростить описание, предположим, что операция ввода-вывода завершилась без ошибок. По завершении операции ввода-вывода генерируется программное прерывание.
7. Запускается процедура обслуживания прерывания, которая помещает в очереди вызов отложенной обработки для завершения обработки ввода-вывода. После запуска вызова отложенной обработки драйвер SCSIPort отмечает пакет IRP как завершенный и вызывает диспетчер ввода-вывода для дальнейшей обработки пакета. Диспетчер ввода-вывода обрабатывает пакет запроса ввода-вывода и вызывает процедуру завершения следующего драйвера в цепочке стека, т.е. драйвера класса диска.
8. Диспетчер ввода-вывода вызывает процедуру завершения драйвера класса диска. Процедура завершения выполняет необходимые операции обслуживания и вызывает диспетчер ввода-вывода для дальнейшей обработки пакета IRP. Диспетчер ввода-вывода переносит обработку на один уровень выше в цепочке стека драйверов. В данном случае к драйверу диспетчера томов FtDisk.
9. Вызывается процедура завершения драйвера FtDisk, которая выполняет собственную обработку. После завершения обработки вызывается диспетчер ввода-вывода, которому сообщается о завершении обработки пакета IRP. Диспетчер ввода-вывода переходит вверх по стеку и вызывает процедуру завершения следующего драйвера — драйвера файловой системы.
10. Вызывается процедура завершения драйвера файловой системы. После необходимого обслуживания вызывается диспетчер ввода-вывода, который планирует запуск асинхронного вызова процедуры. Асинхронный вызов запускается и копирует необходимые данные и код состояния в буфер пользовательского режима приложения резервного копирования. На этом этапе операцию ввода-вывода можно считать завершенной.

1.9 Сложности практической реализации

Стек ввода-вывода подсистемы хранения в семействе Windows Server описан в этой главе довольно подробно. Но помните, что стек подсистемы хранения не обслуживает устройства, поддерживающие несколько протоколов.

Чтобы повысить надежность и безопасность операционной системы Windows, компания Microsoft расширяет и улучшает методику сертификации и подписи драйверов. Поставщикам рекомендуется сертифицировать все обновления драйверов. Хорошим источником информации по этому вопросу может служить Web-узел компании Microsoft, в частности Web-страница по адресу: `http://www.microsoft.com/hwdev/driver/drvsign.asp`.

1.10 Резюме

Операционная система Windows NT проектировалась в качестве многоуровневой и расширяемой, особенно в контексте подсистем хранения и ввода-вывода данных. Создание драйвера соответствующего типа (например, драйвера мини-порта SCSIPort или мини-драйвера Storport) позволяет без проблем добавить поддержку нового устройства.

Драйвер фильтрации позволяет добавить новые функции для Windows NT. Компания Microsoft использовала подобный драйвер при создании приложения Hierarchical Storage Management.

В целом создание драйверов Windows NT требует серьезных знаний в этой области и доступа к подходящему программному инструментарию.

Глава 2

Серверные хранилища данных

Изначально системы хранения данных разрабатывались для мэйнфреймов и большинство из них были основаны на закрытых, частных технологиях. Для мини-компьютеров были созданы такие стандарты, как SCSI. С появлением персональных компьютеров серверные стандарты (SCSI и IDE) получили широкое распространение.

В этой главе рассматриваются системы хранения данных, подключенные к серверу (Direct Attached Storage — DAS), т.е. устройства, подключенные непосредственно к компьютеру под управлением Windows NT. В главе 1 описан стек ввода-вывода Windows NT в контексте подсистемы хранения данных. В этой главе внимание уделяется разработке стека ввода-вывода подсистемы хранения данных Windows NT для улучшения поддержки новых устройств Fibre Channel и SCSI. Подобные устройства воспринимаются Windows NT как подключенные непосредственно к компьютеру.

2.1 Интерфейс SCSI

Аббревиатура SCSI расшифровывается как Small Computer System Interface (*интерфейс для малых компьютеров*), что на данный момент кажется нелогичным, поскольку устройства SCSI обычно подключены к высокопроизводительным серверам и являются наилучшим выбором для использования в качестве подсистемы хранения для многопроцессорных центров хранения данных. Стандарты SCSI утверждаются техническим комитетом T10 (`http://www.t10.org`), подразделением Международного комитета INCITS (InterNational Committee for Information Technology Standards), который, в свою очередь, работает под эгидой Национального института стандартизации США (American National Standards Institute — ANSI; `http://www.ansi.org`).

2.1.1 Стандарты

Шина SCSI изначально разрабатывалась в качестве параллельной архитектуры отправки данных по 8- или 16-разрядной шине. Существовали и последовательные версии архитектуры SCSI, например SSA (Serial Storage Architecture) и 1394. До определенного времени эти архитектуры (особенно SSA) разрабатывались отдельно и лишь позднее были включены в проект стандарта SCSI-3. Оба стандарта — как SSA, так и 1394 — практически не используются на рынке корпоративных подсистем хранения данных, однако достаточно широко распространены на рынке потребительских устройств.

За годы своего существования стандарты SCSI были значительно усовершенствованы, появилось множество версий, которые различаются следующими параметрами:

- ширина шины данных;
- быстродействие шины данных;
- количество устройств, которые могут быть подключены к одной шине;
- электрические и механические характеристики шины;
- максимальная длина шины;
- тип архитектуры шины — последовательная или параллельная. Хотя SCSI исторически ассоциируется с параллельными шинами, компьютерная индустрия активно движется в сторону последовательной архитектуры, и стандарты SCSI не являются исключением.

Обратите внимание: SCSI представляет собой не единый стандарт, а определенный набор стандартов. В одних стандартах заложены механические и электрические характеристики, в других — наборы команд, которые должны быть реализованы устройствами. Эти стандарты реализуются и в других устройствах, например Fibre Channel.

В табл. 2.1 приведены различные стандарты SCSI и их характеристики.

Таблица 2.1. Стандарты и характеристики SCSI

Тип SCSI	Ширина шины (разрядность)	Скорость шины (Мбайт/с)	Максимальное количество идентификаторов SCSI	Максимальная длина шины (метры)
SCSI-1	8	5	8	6 или 25
Fast SCSI (SCSI-2)	8	10	8	6 или 25
Wide SCSI (SCSI-2)	16	10	16	6 или 25

Окончание табл. 2.1

Тип SCSI	Ширина шины (разрядность)	Скорость шины (Мбайт/с)	Максимальное количество идентификаторов SCSI	Максимальная длина шины (метры)
Fast Wide SCSI (SCSI-2)	16	20	16	6 или 25
Ultra SCSI	8	20	8	1.5 или 25
(SCSI-3 SPI-1)			4	3
Wide Ultra SCSI	16	40	16	25
(SCSI-3 SPI-1)			8	1.5
Ultra2 SCSI	8	40	8	12
(SCSI-3 SPI-2)			2	25
Wide Ultra2 SCSI	16	80	16	12
(SCSI-3 SPI-2)			2	25
Ultra3 SCSI или Ultra 160 (SCSI-3 SPI-3)	16	80	16	12
Ultra 320 (SCSI-3 SPI-4)	16	320	16	12

Более старые стандарты, например SCSI-1, сегодня считаются морально устаревшими. Последний стандарт, SCSI-3, на самом деле представляет собой не единую спецификацию, а целое семейство спецификаций, развитие которых продолжается. Таким образом, SCSI следует рассматривать в качестве стандарта, определяющего набор команд и других спецификаций, относящихся к физической реализации этого интерфейса. Очевидно, что определенный стандарт можно взять в качестве основы для реализации нового носителя. Примером может служить тенденция к реализации набора команд SCSI для Fibre Channel и SSA. На данный момент к самым популярным устройствам относятся Ultra SCSI и SCSI-3.

2.1.2 Функциональные возможности и характеристики

Важным параметром считается баланс между расстоянием от устройства до адаптера шины и количеством устройств SCSI, которые можно подключить к шине. Чем больше количество устройств, подключенных к шине, тем

короче должна быть шина. Меньшее количество устройств позволяет удлинить шину. Кроме того, максимальная длина шины зависит от ее электрических характеристик. Обратите внимание, что эти рассуждения касаются исключительно интерфейса SCSI. Новые реализации, например iSCSI (см. главу 8), в значительной степени отменяют старые ограничения на расстояние, поскольку могут передавать команды и результаты их выполнения на большие расстояния, используя транспортный протокол IP.

Устройства SCSI последовательно подключаются к контроллеру. Каждое устройство, подключенное к шине, получает уникальный идентификатор. Идентификаторы SCSI не назначаются произвольно, так как используются для определения относительного приоритета устройства при конкуренции за пропускную способность шины данных между устройствами.

Устройства SCSI обратно совместимы друг с другом, а некоторые даже могут работать в нескольких режимах SCSI. Но в работе более новых устройств под управлением более старого контроллера существуют свои проблемы, которые выходят за пределы тем, рассматриваемых в этой книге. Те, кого интересует описание этих проблем, могут обратиться к списку литературы, приведенному в конце книги.

Высокая производительность устройства хранения данных не гарантирует аналогичной производительности и достаточной пропускной способности сервера. При оценке сервера важно анализировать скорость и пропускную способность не только подсистемы хранения данных, но внутренних шин сервера. Большинство высокопроизводительных серверов используют одновременно несколько адаптеров SCSI, которые подключены к разным шинам PCI. При такой конфигурации шина PCI не станет "узким местом" в общей производительности системы.

2.1.3 Терминология и команды

Для проведения необходимых операций ввода-вывода одно устройство, подключенное к шине SCSI, будет работать как *инициатор*, а другое — как *целевое устройство*. Например, на сервере под управлением Windows NT контроллер SCSI будет инициатором, а жесткий диск или накопитель на магнитной ленте — целевым устройством.

Хотя подобное использование устройства кажется необычным, большинство адаптеров шины SCSI поддерживают обе роли (в зависимости от того, реализована ли подобная функций в прошивке адаптера шины). Контроллер iSCSI относится к устройствам, одновременно представляющим собой как целевое устройство, так и инициатор. Компьютер может быть целевым устройством при экспорте локальных дисков и инициатором при использовании дисков удаленного контроллера iSCSI.

Инициатор отправляет команду целевому устройству, которое выполняет запрошенные операции, отправляя по завершении работы соответствующий ответ. Целевое устройство SCSI остается пассивным до получения команды от инициатора. После получения команды целевое устройство принимает на себя управление шиной перед ответом на команду, например перед отправкой запрошенных данных. Инициатор и все остальные устройства на общей шине совместно определяют хозяина шины.

При загрузке сервера Windows NT контроллер SCSI, который часто называют контроллером шины (об этом речь идет далее), выдает соответствующую команду каждому подключенному к шине и обнаруженному устройству SCSI. Это команда `Report LUNs`, благодаря которой целевое устройство возвращает список номеров логических элементов, управляемых устройством. (Логические элементы описываются в разделе 2.5.)

Не менее важными являются и команды `Reserve/Release`. При наличии нескольких инициаторов одному из них может потребоваться исключительный доступ к целевому устройству. Хорошим примером может служить получение эксклюзивного доступа к накопителю на магнитной ленте. Если два инициатора будут поочередно записывать данные на магнитную ленту, последняя будет содержать бесполезную информацию. Команды `Reserve` и `Release` позволяют обеспечить управление доступом.

Существует два типа команд интерфейса SCSI `Reserve` и `Release`. Один называется *непостоянным резервированием*, поскольку сброс параметров целевого устройства приводит к аннулированию резервирования. Другой тип — *постоянным резервированием*, поскольку резервирование сохраняется и после сброса параметров целевого устройства. Сброс параметров может потребоваться, если устройство-инициатор столкнется с проблемами после операции резервирования. Еще одна команда (*стороннее резервирование*) позволяет инициатору зарезервировать целевое устройство для другого устройства. Отмена резервирований должна осуществляться устройством, выполнившим их ранее, или же устройством, на правах которого выполнялось стороннее резервирование.

Команда `Extended Copy` позволяет инициатору отправить целевому устройству SCSI команду на копирование данных между двумя наборами устройств SCSI. Устройства, между которыми копируются данные, могут (не обязательно) отличаться от устройства, которое получило и обрабатывает команду `Extended Copy`. Дочерняя команда `Receive Copy Results` собирает сведения о завершении выполнения команды `Extended Copy`. Полученный результат может использоваться для определения характера выявленных ошибок команды `Extended Copy`.

В стандарте SCSI-3 определено и множество дополнительных команд, которые включают в себя блочно-ориентированные и графические команды, а также команды модификатора.

Операционная система Windows NT требует от приложений использования *сквозного интерфейса SCSI*, с помощью которого команды передаются устройствам SCSI. На самом деле интерфейс задействуется и для отправки команд устройствам Fibre Channel, которые поддерживают такой же набор команд SCSI. Приложения используют программный интерфейс `DeviceIoControl` с параметром `IoControlCode`, равным `IOCTL_SCSI_PASS_THROUGH` или `IOCTL_SCSI_PASS_THROUGH_DIRECT`. В первую очередь приложению требуется получить дескриптор файла для устройства SCSI посредством функции `CreateFile`. Начиная с Windows 2000, компания Microsoft усилила схему безопасности, требуя от приложений указывать тип доступа (чтение/запись) в параметрах функции `CreateFile` и позволяя только ограниченному количеству учетных записей осуществлять запись. Таким образом, функция `CreateFile` вернет сообщение об ошибке для всех пользователей, которым системный администратор не разрешил запись данных.

Хотя этот интерфейс вполне работоспособен, при детальном рассмотрении архитектура кажется не вполне целостной. С одной стороны, в операционной системе реализован принцип "владею всем, что вижу" по отношению к таким ресурсам, как жесткие диски. С другой стороны, Windows даже не воспринимает резервирования и освобождения устройства хранения данных прикладным приложением. В рамках более эффективной архитектуры приложение будет запрашивать у операционной системы разрешение на проведение необходимой операции, после чего система будет или отказывать, если запрос не может быть выполнен, или выполнять запрошенную операцию, возвращая приложению соответствующие служебные данные. Будет ли развитие Windows двигаться в этом направлении, можно только догадываться.

2.2 Интерфейсы IDE, EIDE и ATA

Устройства с интерфейсом IDE являются самыми распространенными устройствами хранения данных в мире персональных компьютеров, особенно в потребительском сегменте рынка. Аббревиатура IDE расшифровывается как Integrated Drive Electronics (встроенный интерфейс накопителей). В свою очередь, ATA означает AT Attached, где под AT подразумевается классическая модель компьютера IBM PC AT. Обе аббревиатуры указывают на один и тот же стандарт подключения жестких дисков. Основная идея этого интерфейса заключается в интеграции контроллера в жесткий диск, благодаря

чему IDE и называется *встроенным* интерфейсом. В стандарте IDE/ATA описаны характеристики 16-разрядной шины.

Стандарт IDE/ATA, как и SCSI, пережил несколько модификаций. В оригинальном стандарте описывалось использование режима программируемого ввода-вывода (programmed input/output — PIO), в котором центральный процессор играет важную роль при каждой операции ввода-вывода данных. В более поздних стандартах перешли к использованию прямого доступа к памяти (direct memory access — DMA), при котором ввод-вывод данных выполняется без участия центрального процессора.

Кабель IDE/ATA поддерживает подключение двух накопителей. Один из них является *ведущим* (master), а второй — *ведомым* (slave). В любой момент времени только один из дисков может быть активен. Более новый стандарт EIDE (Extended IDE) поддерживает четыре накопителя, так как один контроллер EIDE выступает в роли двух контроллеров IDE. Многозадачная операционная система, например Windows NT, может использовать возможности контроллера EIDE для одновременной передачи двух команд ввода-вывода на два "канала" IDE.

Ниже описаны особенности каждого из стандартов ATA.

- ATA-1 требует использования программируемого режима ввода-вывода.
- ATA-2 был представлен институтом ANSI в 1996 году. В нем описано использование быстрых режимов PIO и допускается применение прямого доступа к памяти (DMA). Кроме того, стандарт ATA-2 позволяет использовать технологию Plug and Play с помощью команды идентификации накопителя, возвращающей информацию о структурных особенностях диска.
- ATA-3 был представлен в 1997 году и может рассматриваться в качестве минимальной модернизации стандарта ATA-2 для повышения надежности в быстрых режимах передачи данных. Главной особенностью стандарта ATA-3 стала поддержка технологии самоконтроля, анализа и отчетности SMART, отвечающей за контроль состояния жестких дисков. Технология SMART поддерживается накопителями SCSI и IDE.
- В ATA-4/ATAPI была введена поддержка таких новых устройств, как дисководы для компакт-дисков и накопители Jazz. Аббревитура ATAPI расшифровывается как AT Attachment Packet Interface. В этом стандарте также описана поддержка Ultra DMA, позволяющая передавать данные с удвоенной скоростью по сравнению с обычным режимом DMA.

Устройства ATA по-прежнему развиваются и постепенно сокращают отрыв в производительности и надежности от устройств SCSI, пытаясь удерживать свое преимущество в цене. При нынешнем увеличении надежности аппа-

ратного обеспечения и появлении специализированного программного обеспечения (например, программных массивов RAID, которые рассматриваются в главе 9) интерфейс ATA со временем может сыграть более важную роль в промышленных системах хранения данных. Однако обсуждение подобных перспектив выходит за рамки данной книги.

2.3 Модель мини-драйвера IDE

В архитектуре Windows Server 2003 поддерживается новая модель мини-драйвера IDE, которая должна заменить существующую модель драйвера IDE. Новый драйвер порта, предоставляемый Microsoft, работает быстрее, обслуживает несколько каналов и позволяет отказаться от разделения канальных интерфейсов и интерфейсов управления. Новая модель драйвера предоставляет более широкие возможности поставщикам жестких дисков; например, драйвер мини-порта от поставщика может изменить значение таймаута запросов и выбрать режим (DMA или PIO) для каждого конкретного запроса. Компания Microsoft считает, что в перспективе устройства ATA будут играть более важную роль, поэтому продолжает развивать модель драйвера ATA. Будем надеяться, что в новых изданиях этой книги появятся дополнительные подробности, которые станут известными в процессе развития новых моделей драйверов.

2.4 Развитие адаптеров шин (HBA)

Изначально такие устройства хранения данных, как жесткие диски и накопители на магнитной ленте, подключались непосредственно к серверу. Единицы хранения размещались непосредственно в сервере или внешних отсеках, подключаемых к серверу с помощью контроллера ввода-вывода. Последний представлял собой плату расширения или модуль системной платы. Со временем для описания контроллеров начал использоваться термин *адаптер шины* (host bus adapter — HBA, т.е. адаптер, который подключен к шине ввода-вывода).

На рис. 2.1 демонстрируется сервер с платой контроллера ввода-вывода и несколькими устройствами, подключенными к контроллеру по шине SCSI. Хранилище данных, подключенное к серверу, — это неплохой вариант для небольших изолированных локальных сетей (например, для локальных сетей определенного отдела компании), однако подобные рещения крайне плохо масштабируются. К одной из очевидных проблем относится количество единиц хранения, которые могут быть использованы при таком подключении.

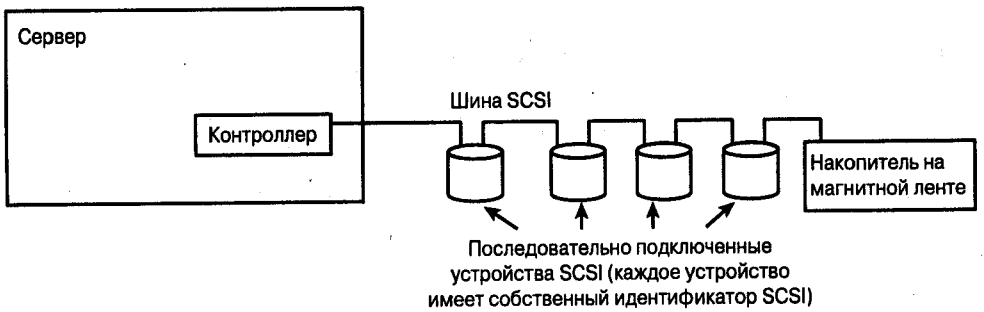


Рис. 2.1. Хранилище SCSI, подключенное к серверу

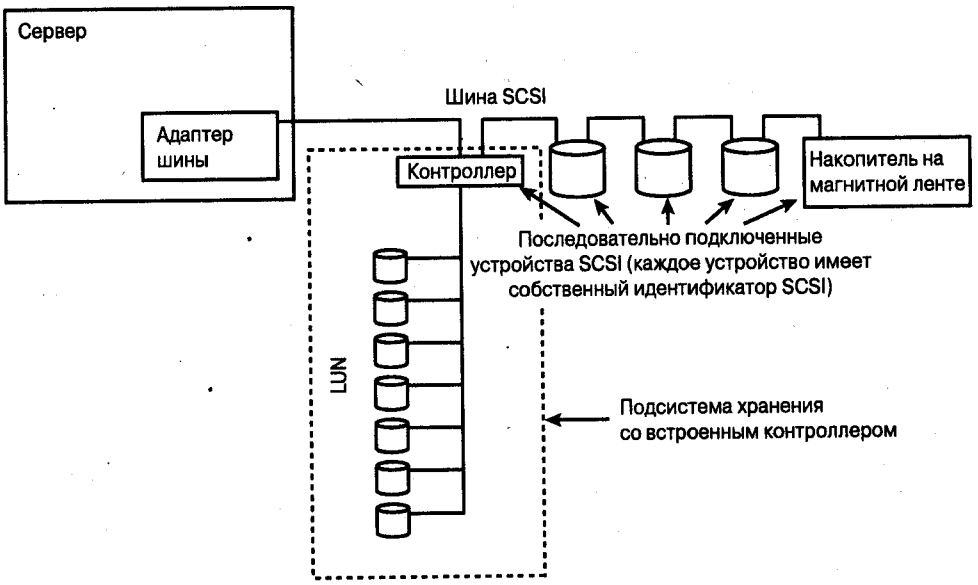


Рис. 2.2. Интеллектуальная подсистема хранения данных

Еще одна проблема — наличие лишь одной шины SCSI для выполнения всех операций ввода-вывода.

Следующим этапом в развитии индустрии хранения данных была разработка подсистем хранения с собственными контроллерами ввода-вывода (рис. 2.2).

Представленная на рис. 2.2 подсистема хранения данных обладает дополнительными функциональными возможностями, что привело к переосмыслению некоторых терминов. Система на рис. 2.1 имеет только один контроллер ввода-вывода, который подключен к серверу. Система на рис. 2.2 содержит два контроллера, что потенциально может привести к путанице при указании

контроллера. Если существуют также дополнительные подсистемы хранения данных с собственными контроллерами, все может стать еще более запутанным. В связи с этим и была принята терминология, в которой контроллер, подключенный к серверу, называется *адаптером шины*. Адаптер шины может обеспечивать подключение к шине SCSI, IDE или интерфейсу Fibre Channel.

2.5 Логические единицы хранения (LUN)

Единицы хранения, которые расположены в схеме, приведенной рис. 2.2, за контроллером подсистемы хранения, должны поддерживать определенный метод адресации. Эти единицы называются LUN (logical unit number). В контексте приложений хранения данных или Windows NT три отдельных идентификатора взаимодействуют друг с другом для уникального определения каждой LUN. Обратите внимание, что в литературе, посвященной стандарту SCSI, используется выражение "определение LUN для целевого устройства SCSI", однако я предпочитаю выражение "для идентификатора SCSI", поскольку устройство SCSI может быть как целевым, так и инициатором. Это различие станет еще более очевидным при распространении интерфейса iSCSI, где устройство SCSI может быть инициатором в одном случае и целевым устройством — в другом.

Команда SCSI отправляется устройству, которое уникально идентифицируется тремя параметрами.

1. *Шина SCSI* (адаптер шины может поддерживать несколько шин или на сервере Windows NT может быть установлено несколько адаптеров шин). Операционная система Windows NT поддерживает до восьми шин для одного адаптера шины.
2. *Идентификатор устройства SCSI на шине.* Операционная система Windows NT поддерживает до 128 идентификаторов SCSI для одной шины.
3. *Идентификатор SCSI LUN.* Операционная система Windows NT SP4 и более поздние версии Windows NT поддерживают до 254 идентификаторов LUN на один идентификатор SCSI. Иногда возможность поддержки такого большого количества идентификаторов LUN обозначается термином *Large LUN support* (расширенная поддержка LUN). Предыдущие версии Windows NT поддерживали до восьми идентификаторов LUN на один идентификатор SCSI. Стандарт SCSI-2 поддерживает до восьми LUN на идентификатор SCSI. В свою очередь, в стандарте SCSI-3 описана поддержка 64-разрядного значения, указывающего количество LUN на идентификатор SCSI, но конкретное количество зависит от типа устройства.

Поддержка такого большого количества единиц LUN (более восьми) зависит от драйвера адаптера шины, а также от устройств SCSI, которые должны обработать команду `Report LUNs` (SCSI). Более новые драйверы поддерживают по умолчанию более восьми LUN. Более старые драйверы могут вообще не поддерживать такой идентификатор, как LUN, или же потребуют изменений в системном реестре Windows NT. Операционная система Windows NT обнаруживает наличие таких устройств, отправляя команду `Report LUNs` на каждое устройство с идентификатором LUN, равным 0. Для активизации в Windows NT поддержки большого количества единиц LUN устройство должно правильно обрабатывать команду `Report LUNs`.

2.6 Драйвер Storport

В главе 1 рассматривается стек ввода-вывода Windows NT, в том числе один из важных его компонентов — драйвер SCSIPort (разработанный в Microsoft). Этот драйвер взаимодействует с драйвером мини-порта, создаваемым поставщиком конкретного устройства и предназначенным для обслуживания последнего. До выхода Windows Server 2003 устройства хранения данных различного типа, например более старые жесткие диски SCSI, новые устройства SCSI-3 и Fibre Channel, использовали специфичный для устройства драйвер мини-порта, который подключался к драйверу SCSIPort. Этот сценарий имел несколько недостатков.

- В описываемой модели подразумевается, что устройства Fibre Channel имеют возможности, сопоставимые с аналогичными функциями устройств SCSI, что не имеет под собой реальной основы.
- Драйвер SCSIPort использовал однопоточную модель, не поддерживая полнодуплексный режим. Это означает, что одни запросы не могли выполняться, пока в очереди находились другие запросы, и наоборот. При обслуживании прерывания драйвер мини-порта не мог принимать новые запросы и сообщать о завершении уже выполненного запроса. Использование Windows на многопроцессорных компьютерах делало эту проблему еще более актуальной.
- Драйвер порта содержал информацию, которая не отправлялась драйверу мини-порта, что требовало от последнего самостоятельного сбора необходимых сведений с помощью нескольких запросов. В частности, это было характерно для работы со списками разборки/сборки[1].

[1] Список разборки/сборки — это термин, обозначающий одновременную инициализацию ввода-вывода в несколько буферов приема.

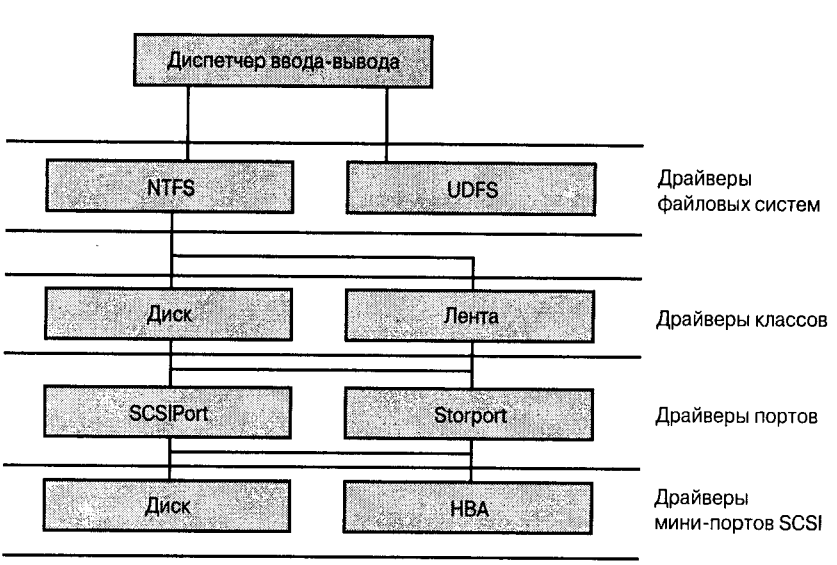


Рис. 2.3. Драйвер Storport в иерархии драйверов Windows Server 2003

- Не удовлетворенные таким положением вещей, поставщики устройств Fibre Channel решили проигнорировать драйвер SCSIPort (созданный компанией Microsoft) и реализовали комплексные драйверы (обеспечивающие возможности как драйвера порта, так и драйвера мини-порта) или собственные драйверы портов и мини-портов. Поскольку этот процесс был основан на инженерном анализе функций драйвера порта, полученные драйверы в лучшем случае работали с минимальным количеством проблем. Зачастую возникали ситуации, при которых два устройства от разных поставщиков не могли одновременно работать на компьютере под управлением Windows NT.

В Windows Server 2003 компания Microsoft представила новую модель драйвера с поддержкой драйвера порта Storport (рис. 2.3). Драйвер Storport предназначен для использования поставщиками более новых устройств SCSI-3 и Fibre Channel. Таким образом, Microsoft намерена постепенно прекратить поддержку драйвера SCSIPort, который, хотя и не удален из Windows, предназначен для использования только вместе с более старыми устройствами хранения данных.

На данном этапе поставщикам адаптеров шины предлагается создавать драйверы мини-портов, которые должны подключаться к драйверу Storport, а не к SCSIPort. Чтобы упростить процесс перехода, Microsoft сделала драйвер Storport обратно совместимым с моделью SCSIPort. Это значит, что поставщики, не желающие затрачивать время и силы, могут воспользоваться

некоторыми (но не всеми) преимуществами новой модели драйверов. Для максимального использования возможностей новой модели драйверов поставщикам требуется выполнить больше работы, чем обычная перекомпиляция и сборка с другой библиотекой.

Новая архитектура обеспечивает значительный прирост производительности, который достигается несколькими методами.

- Новая модель драйверов поддерживает полнодуплексный режим, при котором новый запрос может быть помещен в очередь драйвера одновременно с выполнением операций ввода-вывода. Новая модель поддерживает использование прерываний для выполнения большего объема работы драйвером мини-порта, созданным поставщиком. Обратите внимание, что эта эффективность достигается за счет усложнения структуры драйвера мини-порта, так как от него требуется поддержка синхронизации. Драйверы мини-портов SCSIPort поддерживали только полудуплексный режим и работали не так эффективно, однако поддержка синхронизации не обеспечивалась (поскольку эта обязанность была возложена на драйвер порта).
- Новая модель минимизирует количество вызовов между драйверами порта и мини-порта. Например, в старой модели драйвер мини-порта осуществлял несколько вызовов драйвера SCSIPort для получения списков разборки/сборки.

Кроме повышения быстродействия, модель Storport обеспечивает ряд других преимуществ.

- Модель Storport улучшает взаимодействие между драйверами порта и мини-порта. В модели SCSIPort драйверу порта не разрешалось сообщать о занятости устройства. В свою очередь, модель Storport позволяет драйверу мини-порта передавать драйверу Storport команды на приостановку и возобновление передачи данных, а также сообщать о занятости адаптера.
- В новой архитектуре реализован сложный механизм управления ошибками. Драйвер SCSIPort просто пытался повторно инициировать шину, что требует немало ресурсов и нарушает стабильность работы системы. При соответствующей поддержке драйвера мини-порта от независимых поставщиков аппаратного обеспечения модель Storport позволяет аннулировать параметры конкретной логической единицы, устройства и только в крайнем случае — самой шины. Еще одним улучшением в механизме обработки ошибок является расширение списка ошибок, который поддерживается драйвером порта. Модель Storport позволяет обрабатывать сообщения об ошибках, которые выдают более новые

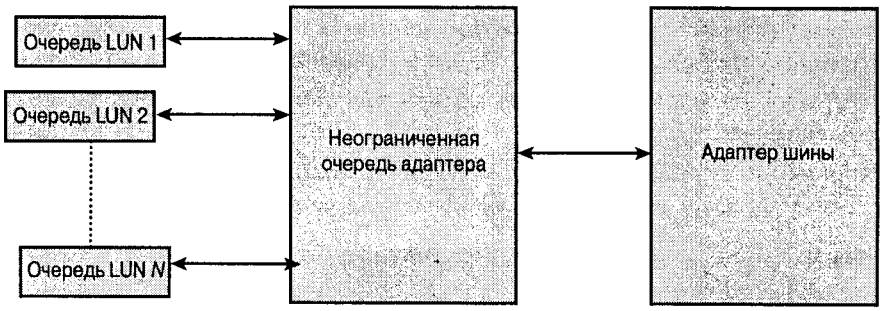


Рис. 2.4. Работа с очередью запросов в модели Storport

устройства SCSI-3, в то время как драйвер SCSIPort маскировал такие ошибки в более старый тип ошибок, поддерживаемых драйвером.

- Все новые возможности имеют минимальный шанс нарушить нормальную работу системы, так как модель Storport обратно совместима с SCSIPort. Поставщики могут просто перекомпилировать и скомпоновать старый код с библиотекой драйвера Storport (вместо библиотеки драйвера SCSIPort), не затрачивая усилий на перенос кода на другую платформу. Это позволит использовать новую модель, хотя и не предоставит всех преимуществ, доступных при полноценном переносе.
- Новая архитектура предоставляет интерфейс, отменяющий необходимость создания "виртуального устройства" (ghost device). Модель SCSI не позволяет приложению запрашивать функции без монтирования модуля мини-порта. В свою очередь, модель Storport не требует создания виртуального устройства, поддерживая запрос функций даже без монтирования этого модуля.

Кроме всего прочего, в новой архитектуре улучшен интерфейс с целью обеспечить соответствие требованиям поставщиков систем хранения данных высокого уровня. Особенно это касается поставщиков систем RAID и Fibre Channel. Например, старая модель SCSIPort поддерживала ограниченный набор возможностей по управлению очередью запросов. Новые устройства, в свою очередь, требуют расширенного управления очередью. Модель Storport поддерживает использование 254 запросов на каждую логическую единицу каждого адаптера. Максимальное количество запросов на адаптер ограничено только количеством логических единиц, поддерживаемых адаптером (рис. 2.4).

Еще одно преимущество иерархической структуры в модели Storport заключается в наличии интерфейса управления для конфигурации и управления высокоуровневыми устройствами хранения данных. Этот интерфейс

основан на инструментарии управления Windows (Windows Management Instrumentation — WMI). Интерфейс используется другими элементами Windows, например интерфейсом управления для командной строки. Технология WMI рассматривается в главе 7. Интерфейс управления WMI поддерживает четыре различных класса WMI.

1. *Класс адаптера шины.* На компьютере может быть установлено несколько таких адаптеров и для каждого из них создается экземпляр такого класса.
2. *Класс канала.* Каждый адаптер может иметь один или несколько каналов, которые соединяют адаптер с его физическими дисками.
3. *Класс массива дисков.* Каждый канал может иметь один или несколько массивов дисков или не иметь таковых вообще.
4. *Класс физического диска.* Каждый адаптер может иметь один или несколько подключенных физических дисков или не иметь таковых вообще.

2.7 Сложности практической реализации

В новой модели драйверов Storport оптимизированы функции ввода-вывода и общая производительность системы управления. Однако системный администратор и менеджеры по закупкам в информационных отделах компаний должны знать, что новая модель драйверов Storport доступна только в Windows Server 2003. Тем же, кто сделал ставку на платформу Windows, рекомендуется ознакомиться с планами поставщика оборудования по переходу к модели Storport и наравне с этим проанализировать поддержку устройств конкретного поставщика в Windows 2000, включая подробности реализации драйвера.

Особое внимание необходимо обратить на параметры пропускной способности системы при использовании модели драйверов SCSIPort (которая скоро станет устаревшей), если эта модель поддерживается поставщиком. Кроме того, стоит обратить внимание на наличие продукции поставщика, в которой вообще не используется модель драйверов SCSIPort. Следует проверить наличие сертификации и поддержки частного драйвера всеми заинтересованными сторонами. Наконец, нужно узнать о возможности "безболезненного" перехода с текущей технологии для Windows 2000 к модели драйверов Storport в Windows Server 2003.

2.8 Резюме

Стандарт SCSI развивается уже много лет и на данный момент превратился в группу стандартов. Стандарты SCSI значительно изменились в контексте

увеличения скорости передачи данных. Пользователям требуется компромисс между количеством устройств, которые подключены к шине, и длиной шины. Кроме того, в процессе развития стандартов SCSI возникли новые команды, например `Extended Copy`.

Интерфейсы ATA/IDE также не отстают и продолжают развиваться, благодаря чему возрастает их надежность и производительность.

Маскировка LUN в Windows NT может выполняться драйвером адаптера шины, который предоставляется производителем.

В Windows Server 2003 внедрена новая модель драйверов Storport, предназначенная для замены старой модели SCSIPort. Более новый драйвер Storport ориентирован на поддержку высокопроизводительных интеллектуальных устройств SCSI-3 и Fibre Channel. По сравнению с драйвером SCSIPort драйвер Storport обеспечивает повышенное быстродействие, расширенные функции по управлению и обработке ошибок. Поставщики устройств могут максимально использовать возможности Storport, переписав драйверы для использования новой модели. И даже обычная перекомпоновка драйвера с библиотекой новой модели Storport дает возможность задействовать некоторые из функций Storport.

Индустрия хранилищ данных развивается от серверов с непосредственно подключенными устройствами в сторону архитектуры клиент/сервер, в которой применяется несколько серверов в локальной сети. При таком подходе одни серверы предназначены для приложений, управляющих базами данных, а другие выделяются исключительно для хранения данных. Выделенные серверы для хранения данных и называются хранилищами, подключенными к сети (NAS), чему посвящена следующая глава.

Глава 3

Сетевые хранилища данных

В предыдущей главе описаны хранилища данных, подключаемые к серверу. Вне зависимости от размещения устройств (внутри сервера или во внешних стойках), только один сервер имеет к ним доступ. В этой главе рассматривается следующее поколение систем хранения данных: хранилища, подключенные к сети (Network Attached Storage — NAS).

Сначала обратимся к истории NAS, после чего рассмотрим стек сетевых протоколов семейства операционных систем Windows Server и архитектуру файловых систем CIFS (Common Internet File System) и NFS (Network File System). Далее вкратце обсуждаются технические аспекты, связанные с необходимостью обслуживания клиентских систем, работающих под управлением различных операционных систем с отличающимися файловыми системами. И в завершение описывается роль Windows в реализации устройств NAS.

3.1 Появление NAS

После появления серверных хранилищ данных по разным причинам возникла необходимость установки нескольких серверов с непосредственно подключенными к ним устройствами хранения данных. Доступ к хранилищам осуществлялся с помощью сетевой файловой системы (обычно в середине 1980-х годов для этого использовалась NFS). Возникновение и развитие архитектуры клиент/сервер, в частности мэйнфреймов и клиентских терминалов, привело к всеобщей реализации данной архитектуры в коммерческих приложениях уровня предприятия.

Устройства хранения данных оставались достаточно дорогими, а потребность в объемах хранимых данных постоянно возрастала. Возможность установки большего количества устройств ограничивалась количеством жестких дисков, подключаемых к одному серверу (обычно поддерживалось семь или восемь адресов, причем адаптеру шины также требовался один из адресов; развитие стандарта SCSI привело к отмене этого ограничения). В результате приходилось устанавливать множество однотипных серверов. Наконец, операции ввода-вывода были ограничены пропускной способностью шины ввода-вывода. Только установка большего количества серверов позволяло снизить нагрузку на шину каждого сервера.

Производители воспользовались сложившейся ситуацией и приступили к развитию концепции сетевых хранилищ данных. Как отмечает Том Кларк (Tom Clark) в книге *Designing Storage Area Networks*, NAS представляет собой скорее маркетинговый, чем технический термин. Устройства NAS разрабатывались как простые в использовании, управлении и развертывании. Кроме того, устройства NAS представлялись в качестве специализированных устройств хранения данных, обеспечивающих оптимальную пропускную способность операций ввода-вывода. Хотя в некоторых случаях это соответствовало истине, т.е. операционные системы NAS были максимально упрощены, чаще всего NAS оказывались обычными, минимально "замаскированными" серверами общего назначения. Даже термин *устройство NAS* указывает на специализированное устройство, а не на обычный сервер, к которому подключено множество устройств хранения данных[1].

Устройство NAS включает в себя серверное программное обеспечение, которое работает под управлением определенной операционной системы. Более того, стек программного обеспечения NAS основан на стандартной серверной файловой системе с добавлением стека ввода-вывода, основанного на коде сетевого сервера, стеках сетевого протокола и сетевой файловой системы (рис. 3.1). Последняя предоставляет такие возможности распределенной файловой системы, как кэширование и синхронизация. Три дополнительных элемента на рис. 3.1 обозначены серым цветом.

В целом сетевые особенности устройства NAS показаны на рис. 3.1 слева, а стек локальной файловой системы и подсистемы хранения — справа. Чтобы упростить изложение, рассмотрим только стек протоколов TCP/IP, хотя иногда применяются другие сетевые протоколы, такие, как UDP/IP или устаревший Netware IPX/SPX.

На рис. 3.1 программное обеспечение сервера NAS отправляет запрос ожидания TCP/IP и ожидает входящего запроса от клиента. После того как клиент отправит запрос, ожидание сервера завершается и начинается сеанс TCP/IP. Как только сеанс TCP/IP будет установлен, клиент может пройти аутентификацию и отправить запрос на открытие, чтение или запись файлов по протоколам CIFS/SMB или NFS (эти протоколы рассматриваются далее в главе). Как только программное обеспечение NAS получает запрос на ввод-вывод данных файла, сервер NAS использует локальную файловую систему для выполнения операции ввода-вывода. Результат операции (считанные данные или статус после записи) отправляется клиенту с помощью сетевой файловой системы и стека протоколов сетевого устройства.

[1]Обратите внимание, что локальное хранилище данных, подключенное к устройству NAS, обычно представляет собой устройство типа DAS (см. главу 2).

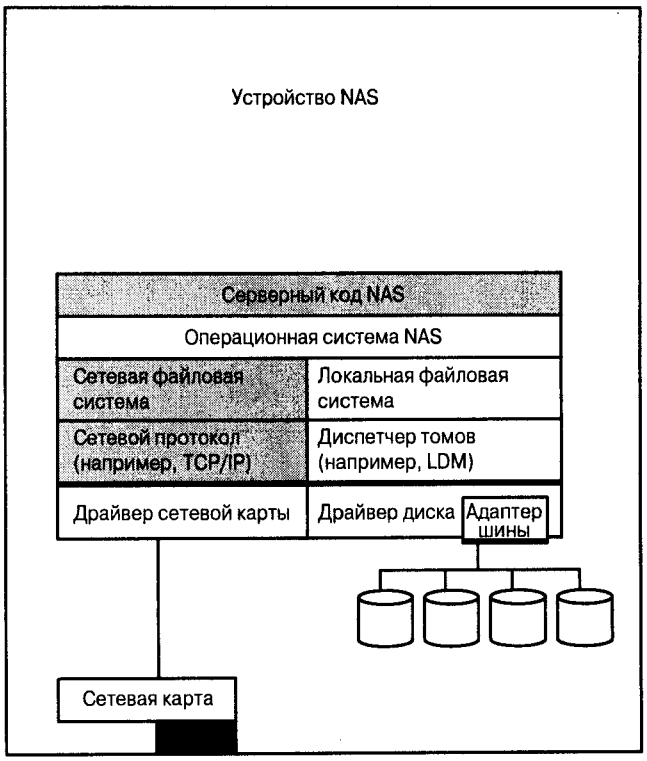


Рис. 3.1. Стек ввода-вывода устройства NAS

Производители устройств NAS придерживаются разных стратегий при разработке операционных и файловых систем, применяемых в устройствах NAS:

- использование стандартной операционной системы, например Windows NT или UNIX;
- разработка собственных операционной и файловой систем, например Network Appliance;
- приобретение операционной и файловой систем у другого производителя.

3.2 Сетевой стек Windows NT

Разобраться в особенностях стека сетевого ввода-вывода Windows NT важно по нескольким причинам. Клиент Windows NT использует стек сетевого ввода-вывода для получения доступа к ресурсам, которые находятся под управлением сервера, а также для передачи данных. Кроме того, при использовании сетевого хранилища часто возникает ситуация, когда один сервер

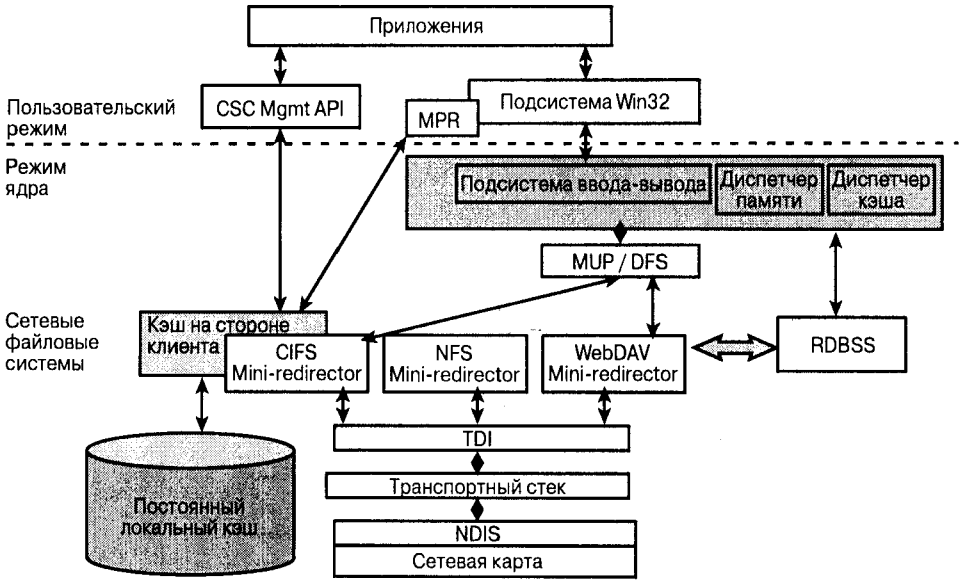


Рис. 3.2. Сетевой стек Windows NT

получает доступ к ресурсам, которыми управляет другой сервер. Хорошим примером будет Web-сервер под управлением Windows NT, который для ответа на запрос клиента запрашивает базу данных, размещенную на отдельном сервере SQL под управлением Windows NT. Web-сервер получает доступ к серверу SQL средствами стека сетевого ввода-вывода Windows NT.

На рис. 3.2 показан стек сетевого ввода-вывода Windows NT. В разделах 3.2.1–3.2.6 рассматриваются различные компоненты, представленные на рис. 3.2 снизу вверх (а также стек ввода-вывода).

Сетевой платой обычно управляет драйвер, совместимый со спецификацией NDIS (Network Driver Interface Specification). Она представляет собой интерфейс, обеспечивающий взаимодействие драйверов сетевой платы со стеком сетевых протоколов более высокого уровня. Следующий уровень — это стек TCP/IP. Термин TCP/IP используется для описания всех компонентов, которые задействованы в передаче данных по сети, например IP, DHCP или TCP. К следующему уровню относится интерфейс транспортного драйвера.

3.2.1 Интерфейс транспортного драйвера

Уровнем выше над стеком сетевых протоколов TCP/IP находится *интерфейс транспортного драйвера* (Transport Driver Interface — TDI). Этот высокопроизводительный интерфейс режима ядра предназначен для сетевых приложений, которым требуется запрос и получение услуг сетевых транспортных

протоколов. Интерфейс TDI реализован в виде библиотеки, к которой подключаются драйверы режима ядра; он позволяет осуществлять связь по сети и пытается сократить количество операций копирования данных, например между буфером приложения и буфером сетевого транспортного протокола.

В частности, TDI позволяет приложениям регистрировать функции обратного вызова. Когда приложение получает пакет с данными, вызывается указанная функция обратного вызова. Процедура обратного вызова может просто просматривать пакет с данными, особенно если он небольшой, и сообщать, что обработка пакета завершена (после чего он будет отмечен как успешно доставленный). Если необходимо, клиент может копировать пакет в буфер для последующей обработки.

3.2.2 Подсистема буферизации перенаправленных дисков

На следующем уровне находится подсистема буферизации перенаправленных дисков (Redirected Drive Buffering Subsystem — RDBSS), которая отвечает за предоставление кода буферизации всем перенаправителям (redirectors). Подсистема обеспечивает полноценное взаимодействие с диспетчером кэша Windows NT для всех сетевых файловых систем. Подсистема RDBSS впервые появилась в Windows 2000; до этого момента всем сетевым файловым системам требовался полноценный драйвер, который обеспечивал взаимодействие с операционной системой.

3.2.3 Мини-перенаправители

Эти устройства обеспечивают работу функций, относящихся к конкретной сетевой файловой системе или протоколу и пользуются услугами RDBSS для обработки рутинных операций взаимодействия с Windows NT. В Windows 2000 и Windows XP предоставляется несколько мини-перенаправителей.

- Мини-перенаправителем CIFS оснащены Windows 2000, Windows XP и Windows Server 2003. В более ранних версий Windows NT был реализван монолитный перенаправитель. Код RDBSS, общий для всех мини-перенаправителей, реализовался независимо каждым поставщиком. Технология CIFS более подробно рассматривается в разделе 3.3.
- Мини-перенаправитель WebDAV (Web Distributed Authoring and Versioning) поддерживает протокол HTTP 1.1 и расширения WebDAV для чтения и записи документов Web. Он предоставляет возможности, позволяющие назначать буквы дисков серверу независимо от протокола доступа к последнему. Назначение поддерживается как для серверов HTTP, так и для серверов CIFS. Корпоративные брандмауэры обычно разрешают прохождение пакетов HTTP и зачастую блокируют за-

просы/ответы протокола CIFS. Таким образом, мини-перенаправитель WebDAV обеспечивает доступ к серверам HTTP через брандмауэры, блокирующие пакеты CIFS. Поскольку стандарты XML и HTTP играют все большую роль, важность мини-перенаправителя WebDAV со временем будет увеличиваться. Обратите внимание, что WebDAV представляет собой исключительно клиент-серверный протокол, не поддерживающий взаимодействие между серверами.

- Система Windows NT Services for UNIX (SFU) поставляется с мини-перенаправителем, поддерживающим протокол NFS. Файловая система NFS рассматривается в разделе 3.4.

3.2.4 Поставщик множественных имен UNC

В Windows поддерживается универсальное соглашение об именовании (Universal Naming Convention — UNC), необходимое для получения доступа к удаленным файлам. Поддержка UNC своими корнями уходит во времена MS DOS 3.3, которая существовала задолго до создания Windows NT. Имена UNC позволяют приложениям указывать файлы согласно их расположению на сервере и общем ресурсе (помните, что на одном сервере может существовать несколько общих ресурсов). Формат имен UNC выглядит следующим образом:

```
\\ИмяСервера\Ресурс\Подкаталог1\...\ПодкаталогN\ИмяФайла
```

Поддержка UNC в разных версиях Windows различается следующими параметрами:

- максимальная длина пути UNC;
- максимальная длина каждого элемента пути UNC, например длина имени подкаталога;
- максимальная длина имени сервера.

Для предоставления приложениям и утилитам возможности использования сетевых ресурсов с помощью путей UNC компания Microsoft создала поставщика множественных имен UNC (Multiple UNC Provider).

Поскольку Windows NT поддерживает несколько сетевых файловых систем, MUP работает как маршрутизатор, перенаправляя запросы нужной сетевой файловой системе, или же как мини-перенаправитель, который поддерживает конкретную сетевую файловую систему. Маршрутизация выполняется одним из двух способов.

1. Проверяется кэш MUP, чтобы узнать, осуществлялось ли ранее подключение к серверу. В этом случае используется мини-перенаправитель, который применялся при последнем подключении.

2. По порядку опрашивается каждый мини-перенаправитель для подключений UNC, которые отсутствуют в кэше.

3.2.5 Маршрутизатор множественных поставщиков

Маршрутизатор множественных поставщиков (Multi-Provider Router — MPR) предоставляет функции, аналогичные MUP, перенаправляя запросы приложений соответствующему мини-перенаправителю, однако существует два отличия.

1. Код MPR работает в пользовательском режиме, а не в режиме ядра.
2. Маршрутизатор MPR предназначен для приложений, не использующих пути UNC, например для приложений, применяющих WinINet API (термин WinINet расшифровывается, как Windows Internet). Это программный интерфейс приложений, который предоставляет дополнительный уровень абстракции для программ, использующих стандартные протоколы Internet — HTTP, FTP и Gopher.

Маршрутизатор MPR представляет собой динамически подключаемую библиотеку с удобным интерфейсом от компании Microsoft. Интерфейс этой библиотеки используется поставщиками устройств для создания мини-перенаправителей, причем библиотека устанавливается при установке перенаправителя.

3.2.6 Клиентское кэширование

Мини-перенаправитель CIFS в Windows 2000 поддерживает функцию *клиентского кэширования*, которая позволяет кэшировать файлы локально на компьютере клиента. Кэшироваться могут как файлы документов (например, файлы Microsoft Word или Excel), так и выполняемые файлы (например, файлы приложений из пакета Microsoft Office). Кэширование инициируется одним из двух способов.

1. Пользователь явно запрашивает кэширование.
2. Мини-перенаправитель инициирует кэширование при открытии файла.

Кэширование на стороне клиента обеспечивает достаточную производительность и масштабируемость системы. Клиент может продолжать работу даже после отключения от сервера (например, при отказе в работе сервера или при нарушении работы сети). Нагрузка на сервер снижается, и один сервер может обслуживать большее количество клиентов. Клиентские приложения работают быстрее, так как получение файла с локального диска осуществляется быстрее, чем по сети.

Кэширование возможно не для всех файлов — только для тех файлов общего ресурса, которые явно отмечены (администратором сервера) как разрешенные для кэширования. Эта информация передается мини-перенаправителю CIFS в ответном пакете SMB (Server Message Block). При этом протокол SMB модифицирован для доставки клиенту информации о возможности кэширования файлов, а также о типе кэширования, выполняемого клиентом. Допускается три типа кэширования.

1. Общие ресурсы, в которых разрешено кэширование только документов.
2. Общие ресурсы, в которых разрешено кэширование только программ.
3. Общие ресурсы, в которых кэширование запрещено.

На данный момент только клиенты CIFS от компании Microsoft поддерживают клиентское кэширование. Клиентские приложения других производителей также кэшируют файлы, но только при наличии активного подключения к серверу и только файлы, открытые клиентом на сервере. Клиентское кэширование, реализованное в Windows, позволяет кэшировать файлы даже при отключении от сервера. Кроме того, поддерживается кэширование ранее открытых файлов, которые закрыты в текущий момент. Кэшированные файлы остаются доступными для клиента, даже если он отключен от сервера.

Технология клиентского кэширования в Windows 2000 проектировалась для предоставления соответствующих функций приложениям Windows, для которых кэширование происходит совершенно незаметно. Хотя кэширование должно быть незаметным и для пользователей, им все же придется вмешиваться в процесс синхронизации данных и обработки ошибок при клиентском кэшировании. Другими словами, если несколько клиентов вносят конфликтующие между собой изменения в один документ в собственных кэшах, пользователю выдается сообщение о программном конфликте, причем решить проблему пользователь должен самостоятельно.

Можно выделить ряд компонентов, которые совместно предоставляют клиентам необходимые функции.

- Мини-перенаправитель CIFS.
- Программный интерфейс приложений, который предоставляется мини-перенаправителем CIFS и связанными с ним компонентами пользовательского режима. Этот интерфейс позволяет приложениям использовать предоставляемые возможности и управлять ими.
- Компоненты пользовательского режима, которые получают данные с сервера и записывают их в локальный кэшированный файл.
- Локальная база данных, которая отслеживает данные, кэшированные локально, и их текущее состояние. В этой базе данных создаются записи для каждого общего ресурса, к которому подключается клиент

(ресурс должен быть отмечен, как допускающий кэширование) и для каждого поддерживающего кэширование файла, который открывается клиентом. Кроме того, записи создаются для всех каталогов, которые размещены между файлом и корнем общего ресурса. Для каждой записи, созданной в базе данных, параметры безопасности пользователя и гостевого пользователя клиентской системы копируются с сервера CIFS. База данных работает в качестве файловой системы для файлов, кэшированных на локальном компьютере.

При первом кэшировании файл программы отмечается как разреженный и локальная копия постепенно заполняется данными параллельно с тем, как приложение запрашивает их с сервера. При кэшировании файл документа также отмечается в виде разреженного. Все команды записи передаются сначала документу удаленной системы, после чего записываются в локальной кэшированной копии. Программа-агент, работающая в пользовательском режиме, постепенно копирует данные с удаленного общего ресурса в локальную кэшированную версию документа. После полного копирования файла флаг разреженности снимается и файл может использоваться вместо копии, хранящейся на сервере.

Если несколько клиентов кэшируют документы и редактируют их в автономном режиме, возникает необходимость в поддержке последовательности (когерентности) данных между различными клиентами. Одно из простых решений заключается в запрете на редактирование кэшированных копий документа. Однако компания Microsoft не пошла на такое ограничение и сделала возможным редактирование автономных документов.

Изменения файла, внесенные в автономном режиме, сохраняются и синхронизируются. В случае конфликта в процессе синхронизации пользователь получает уведомление и запрос на решение конфликта.

3.3 Технологии CIFS и SMB

Общий протокол доступа к файлам Internet (Common Internet File System — CIFS) своим происхождением обязан технологии блока серверных сообщений (Server Message Block — SMB), которая впервые появилась в MS DOS 3.3. В стандарте SMB описан протокол отправки команд файловой системы (открыть файл, считать, записать, блокировать и закрыть) от клиента к файловому серверу.

Перед обсуждением технических подробностей технологий CIFS и SMB необходимо выяснить основные различия между ними. Изначально существовала только технология SMB, которая использовалась в качестве клиент-серверного файлового протокола в мире персональных компьютеров. В середине

1980-х годов компания Microsoft дала своей реализации протокола SMB название CIFS и начала позиционировать CIFS в качестве прямого конкурента стандартов WebNFS и NFS. Компания Microsoft предоставила ознакомительный документ RFC на рассмотрение группе IETF (Internet Engineering Task Force)[2], и впоследствии срок действия документа истек без попыток превратить RFC в одну из спецификаций IETF.

Независимые от компании Microsoft поставщики устройств NAS приступили к разработке спецификации CIFS и организовали несколько мероприятий для популяризации CIFS. Ассоциация SNIA (Storage Networking Industry Association) взяла на себя задачу публикации CIFS. Компания Microsoft также выпустила спецификацию CIFS (она называлась Common Internet Filesystem Access Protocol), распространявшуюся бесплатно (ссылка на спецификацию находится в списке основных источников информации, приведенном в конце книги).

В похожих друг на друга спецификациях SNIA CIFS и CIFS от компании Microsoft описывается протокол, используемый клиентами Windows NT 4.0 для получения доступа к ресурсам серверов Windows NT. В обеих спецификациях не рассматривается протокол SMB, который применяется в новых версиях Windows (например, не затрагивается клиентское кэширование, которое поддерживается в Windows 2000 и описано в разделе 3.2.6). Кроме того, в спецификациях не описаны все протоколы взаимодействия между серверами. Новый стандарт SMB, не относящийся к бесплатным спецификациям, описан в соответствующей спецификации, которая за определенную плату распространяется компанией Microsoft, что стало возможным благодаря судебным решениям Европейского Союза и правительства США. Дополнительная информация доступна в статье "Microsoft Settlement Program: Communications Protocol Program" на Web-узле компании Microsoft по адресу: `http://www.microsoft.com/legal/protocols`.

Таким образом, компания Microsoft вновь стала называть свою реализацию описываемой технологии блоком SMB. По сути, SMB от Microsoft — это закрытый протокол, представляющий собой расширенную версию индустриального стандарта CIFS.

Кроме того, следует обратить внимание на историческую связь между SMB/CIFS и NetBIOS. Программный интерфейс NetBIOS (уровень сеанса в модели OSI) на данный момент безнадежно устарел. Интерфейс реализует уровень абстракции, который позволяет приложениям работать с различными транспортными протоколами, например TCP/IP, NetWare или уже забытым протоколом XNS (Xerox Network System). Необходимость в программном интерфейсе приложений, который предоставляет возможность создания

[2] Автор этой книги также принимал участие в создание описываемого документа RFC.

приложений, не зависящих от сетевого протокола, существует и поныне. Однако в настоящий момент для этого обычно используется интерфейс сокетов, в частности в мире Windows — интерфейс Winsock.

Компания Microsoft использовала NetBIOS для преобразования имен (преобразования имени сервера в сетевой адрес), но сейчас для этого предназначена стандартная служба DNS.

Изначально Microsoft не использовала TCP/IP в качестве транспортного протокола, что кардинально изменилось со временем, однако поддержка NetBIOS продолжала присутствовать. Тем не менее роль NetBIOS постоянно уменьшалась. После назначения порта TCP/IP для файловых серверов SMB зависимость от NetBIOS была полностью "излечена", по крайней мере в контексте базового протокола. Но ситуация оставалась запутанной, так как некоторым вторичным службам клиентов и серверов Windows все равно требовался протокол NetBIOS. Хорошим примером будет объявление серверами о своем присутствии в сети и предоставление списка доступных служб, а также передача этих объявлений клиентам другими серверами. Со временем службы были переделаны и NetBIOS был полностью снят со счетов с выходом Windows 2000.

Наконец, наследие SMB можно заметить в каждом запросе CIFS, поскольку каждый запрос и ответ должны начинаться со значение "0xFF", после чего следуют такие символы ASCII, как "SMB".

3.3.1 Разновидности стандарта CIFS

К сожалению, точного определения стандарта CIFS не существует. Различные типы протоколов SMB называются *диалектами*. Вот несколько возможных вариантов:

- применяемый клиентами DOS и Windows 3.x;
- используемый при подключении к серверам, не работающим под управлением Windows;
- применяемый клиентами под управлением Windows NT.

Чаще всего клиент отправляет серверу запрос на установку сеанса и передает список всех поддерживаемых вариантов протокола. Сервер выбирает наиболее функциональный вариант и отправляет клиенту соответствующий запрос. В зависимости от протокола, о котором "договорились" клиент и сервер, некоторые запросы и соответствующие им ответы могут быть недопустимыми. Согласованный вариант протокола не определяет однозначно фактическую реализацию функций протокола, что вносит еще бóльшую путаницу; для указания поддержки определенных функций некоторые флаги могут

быть установлены или сброшены. Другими словами, даже при выборе протокола существуют различные варианты предоставляемых функций: например, один из флагов может указывать на наличие поддержки длинных имен файлов.

Как описано в документе RFC компании Microsoft (по правилам IETF на данный момент он уже устарел), протокол CIFS обеспечивает взаимодействие клиента и сервера для доступа к файлам и управления ими. Такие функции, как объявление о наличии в сети доступных принтеров и серверов, выходят за рамки возможностей протокола CIFS.

Организация SNIA продолжает работу над спецификацией CIFS. Кроме того, SNIA проводит ежегодную конференцию, посвященную CIFS, и организует другие мероприятия, на которых обсуждаются вопросы взаимодействия систем по протоколу CIFS.

Спецификация SMB стала стандартом с 1992 года (X/Open CAE Specification C209) и описывает SMB как протокол для обеспечения взаимодействия между компьютерами под управлением DOS, Windows, OS/2 и UNIX.

3.3.2 Описание протокола CIFS

Запросы и ответы CIFS имеют четкую, понятную структуру. Поля пакетов SMB также стандартизированы, и отличия зависят от выбранной разновидности CIFS и функций, поддерживаемых как клиентом, так и сервером.

В табл. 3.1 приведена общая структура пакета SMB. Обратите внимание, что показаны только общие элементы для всех вариантов SMB. Подробности строения каждого варианта пакета SMB выходят за пределы тематики этой книги.

Некоторые поля в табл. 3.1 требуют более полного описания. Поле команды имеет размер в один байт и описывает тип запроса. Сервер копирует это значение в ответ, что позволяет клиенту анализировать последний. Спецификация CIFS содержит значения и определения для этого поля. Описанные команды позволяют выполнять такие операции, как открытие файла, чтение, запись и блокирование определенного диапазона файла. Все эти операции выполняются в качестве ответа на запрос приложения.

Кроме того, запросы клиента CIFS (и связанные с ними ответы сервера) инициируются кодом перенаправителя без явного вмешательства приложения. Примерами будут кэширование и оппортунистическая блокировка (opportunistic locking), которые рассматриваются в разделе 3.3.5. В спецификациях CIFS RFC и SNIA, а также Open Group определены значения и семантика кода команды CIFS размером в 1 байт.

Таблица 3.1. Структура заголовка SMB

Поле	Размер	Описание
0xFFSMB	4 байт	Всегда имеет значение `0xFFSMB`
Команда	1 байт	Указание типа запроса
Статус	4 байт	32-разрядный код ошибки (генерируется серверами Windows NT и возвращается в виде 32-разрядного кода ошибки клиентам, поддерживающим коды ошибок Windows NT) ИЛИ Для более старых клиентов, которые не поддерживают 32-разрядные коды ошибок, сообщение об ошибке преобразуется в старый структурный тип. Старый тип включает в себя: ■ 8-разрядный класс ошибки, который указывает на ее разновидность, т.е. сообщается ли эта ошибка операционной системой сервера или самим сервером; кроме того, это может быть ошибка аппаратного обеспечения или протокола SMB; ■ 8 разрядов игнорируются; ■ 16-разрядный код ошибки, который имеет значение только в рамках определенного класса ошибки
`Flags`	1 байт	Семантика представлена в табл. 3.2
`Flags2`	2 байт	Семантика представлена в табл. 3.3
Заполнение/ подпись	12 байт	Заполнение/подпись. Рассматривается в тексте раздела
Tid-значение	2 байт	Используется для идентификации ресурса сервера, который запрашивается клиентом. Указывается с помощью запроса SMB TreeConnect

Окончание табл. 3.1

Поле	Размер	Описание
Идентификатор процесса (Pid)	2 байт, но при необходимости может быть расширен до 4 байт	Устанавливается клиентом. Указание на клиентский процесс, который осуществляет запрос. Используется сервером для отслеживания режима открытия файлов и для блокировок. Отправляется сервером обратно клиенту вместе с идентификатором мультиплексора (mid) для уникальной идентификации запросов, на которые отвечает сервер
Идентификатор мультиплексора (Mid)	2 байт	Устанавливается и используется клиентом. Сервер возвращает полученный Mid в ответе на запрос. Клиент использует значения Mid и Pid, чтобы идентифицировать запрос, для которого пришел ответ
Uid-значение	2 байт	Назначается сервером после аутентификации клиента. Клиент должен использовать Uid во всех запросах
Параметры	Переменная	Состоит из 16-разрядного счетчика слов, который указывает на количество следующих за счетчиком 16-разрядных слов. Для каждой команды SMB этот счетчик обычно представляет собой фиксированное значение с одним счетчиком слов для команды и вторым — для ответа. Обычно этот счетчик имеет значение 5 или меньше
Данные	Переменная	Состоит из 16-разрядного счетчика слов, который указывает на количество следующих за счетчиком байт (8-разрядных слов) данных. По сравнению с полем параметров поле данных может иметь намного больший размер, например 1 Кбайт и более. В запросах SMB на чтение и запись это поле содержит фактические данные, которые считываются или записываются

Помните, что новые значения и семантика полей могут появиться без предупреждения с выходом новых версий Windows, так как протокол CIFS продолжает развиваться.

Существует несколько команд для выполнения одинаковых базовых операций; например, для открытия, чтения и записи существует несколько раз-

личных команд. Некоторые команды уже не применяются, а в ряде случаев могут использоваться альтернативные команды, в зависимости от выбранного диалекта протокола.

Далее представлены примеры функций, описанных в поле команды.

- Выбор типа SMB.
- Установка сеанса связи.
- Переход по каталогам и перечисление файлов и каталогов.
- Открытие, создание, закрытие или удаление файлов.
- Блокировка и разблокировка определенных фрагментов файла.
- Операции печати.
- Уведомления об изменении файлов и каталогов.
- Транзакции, при которых указываются данные, параметры и операции. Сервер CIFS выполняет запрошенную операцию и возвращает результат — данные и параметры. Примерами транзакций могут служить ссылки в распределенной файловой системе и управление расширенными атрибутами.

В табл. 3.2 представлены функции поля `Flags` (флаги) из 3.1.

Таблица 3.2. Семантика поля `Flags`

Значение	Описание
`0x01`	Зарезервировано. Используется устаревшими запросами
`0x02`	Зарезервировано. Должно быть равным нулю
`0x04`	Указывает на необходимость учета регистра в именах файлов и каталогов
`0x08`	Зарезервировано
`0x10`	Зарезервировано. Используется устаревшими запросами
`0x20`	Зарезервировано. Используется устаревшими запросами
`0x40`	Зарезервировано. Используется устаревшими запросами
`0x80`	Указание, что это ответ SMB

В поле `Flag2` из табл. 3.1 описано еще больше необязательных функций. Значения этого поля приведены в табл. 3.3.

Поле *Заполнение/подпись* изначально представляло собой последовательность "холостых" байт. Со временем значение этого поля изменилось. Поле заполнения теперь может включать в себя следующие элементы:

Таблица 3.3. Семантика поля `Flags2`

Значение	Описание
0x0001	Клиент поддерживает длинные имена файлов. Сервер может возвращать длинные имена файлов
0x0002	Клиент поддерживает расширенные атрибуты OS/2
0x0004	Включено подписывание пакетов SMB
0x0008	Зарезервировано
0x0010	Зарезервировано
0x0020	Зарезервировано
0x0040	Каждое имя пути в запросе представляет собой длинное имя
0x0080	Зарезервировано
0x0100	Зарезервировано
0x0200	Зарезервировано
0x0400	Зарезервировано
0x0800	Указание на использование расширенного механизма безопасности, который рассматривается в разделе 3.3.3
0x1000	Пути в запросе должны быть преобразованы средствами распределенной файловой системы
0x2000	Страничный ввод-вывод, указывающий на то, что чтение должно быть разрешено, когда у клиента есть соответствующее разрешение
0x4000	Указание на возврат 32-разрядного кода ошибки. Если флаг не установлен, используется код ошибки в стиле DOS
0x8000	Если флаг установлен, пути в пакете SMB указаны в формате Unicode. В противном случае применяется кодировка ASCII

- 2 байта идентификатора процесса, что позволяет указывать 32-разрядные идентификаторы процесса;
- 8 байт для хранения подписи пакета SMB, если эта функция активизирована (см. описание поля `Flags2` в табл. 3.3 и в разделе 3.3.3);
- 2 неиспользуемых байта.

3.3.3 Безопасность CIFS

Протокол CIFS обеспечивает безопасность средствами сервера. Администратор может отключить систему встроенной безопасности CIFS, в чем ед-

ва ли появится необходимость, поэтому система безопасности включена по умолчанию.

В старых вариантах CIFS допускается отправка незашифрованного текстового пароля от клиента к серверу, что категорически не рекомендуется. Протокол CIFS допускает защиту ресурсов сервера с помощью паролей отдельных пользователей (это называется *безопасностью на уровне пользователя*). Для обеспечения обратной совместимости серверы CIFS поддерживают защиту общего ресурса на базе пароля, одинакового для всех пользователей. Поскольку ресурс будет предоставлен в общее пользование, этот метод называется *безопасностью на уровне ресурса*. Использовать механизм безопасности на уровне ресурса не рекомендуется, и в Windows 2000 Server эта система отсутствует. Первый пакет SMB, который отправляется серверу клиентом, называется `SMB_NEGOTIATE_PROTOCOL`. Пакет применяется для выбора типа CIFS. В ответ на запрос `SMB_NEGOTIATE_PROTOCOL` сервер сообщает об используемом механизме безопасности (уровень пользователя или ресурса).

Начиная с Windows NT4 SP3 и Windows 2000, компания Microsoft предоставила возможность размещения в пакетах SMB цифровой подписи. Сервер может быть настроен на обязательное требование цифровой подписи от клиента; в противном случае клиенту будет запрещен доступ к ресурсам. Использование цифровой подписи отражается на производительности как сервера, так и клиента, но это цена, которую приходится платить за безопасность. Обратите внимание, что подписывание и проверка имеют двунаправленную природу, т.е. клиент подписывает отправляемые запросы, сервер проверяет подпись клиента и подписывает отправляемые ответы, после чего клиент проверяет подпись сервера. Подпись пакета SMB хранится в поле *Заполнение/подпись* (см. табл. 3.1).

Ответ на запрос `SMB_NEGOTIATE_PROTOCOL` используется для предоставления клиенту информации о поддержке сервером подписывания пакетов SMB и о необходимости обязательного подписывания пакетов SMB.

3.3.4 Аутентификация CIFS

Протокол CIFS позволяет определять уровень безопасности при взаимодействии серверов и клиентов. Сервер может быть настроен на отказ в обслуживании клиентов, которые предлагают слишком низкий уровень безопасности.

Протокол CIFS предоставляет механизмы аутентификации, необходимые серверу для аутентификации клиента. Кроме того, предоставляются методы аутентификации сервера клиентом. В базовом уровне аутентификации клиент сообщает имя пользователя и незашифрованный пароль. По очевидным причинам такой подход нежелателен. Более того, сервер можно настроить на

отказ в обслуживании клиентов, которые отправляют пароли в незашифрованном виде.

Аутентификация может выполняться с помощью технологии, которая называется *протокол запрос/ответ* (challenge/response protocol). При отправке клиентом пакета `SMB_NEGOTIATE_PROTOCOL` для выбора типа CIFS флаг в ответе сервера указывает на возможность использования протокола запрос/ответ. Если сервер поддерживает этот протокол, в ответе сервера предоставляется 8-байтовый запрос. Запрос — это случайное значение с очень низкой вероятностью повторной генерации. И клиент и сервер формируют ключ из пароля пользователя. После этого запрос шифруется с помощью ключа и алгоритма DES (Data Encryption Standart). Клиент отправляет запрос серверу, а сервер сравнивает ответ с собственным подсчитанным значением. Если два значения совпадают, клиент доказывает знание пароля и подтверждает свою аутентичность.

Кроме того, протокол CIFS поддерживает систему *расширенной безопасности* (можете и не надеяться, что читатель, который догадается об указании на поддержку расширенной безопасности в ответе сервера на запрос `SMB_NEGOTIATE_PROTOCOL`, получит награду). Механизм расширенной безопасности предоставляет возможность поддержки произвольного протокола аутентификации в рамках протокола CIFS. При выборе расширенной безопасности первый двоичный объект безопасности предоставляется в ответе на запрос `SMB_NEGOTIATE_PROTOCOL`. Двоичные объекты безопасности не обрабатываются протоколом CIFS. В этом он полагается на механизмы клиента и сервера, предназначенные для генерации и обработки двоичных объектов. Последующие двоичные объекты безопасности могут передаваться с данными SMB.

Использование механизма расширенной безопасности позволило Microsoft обеспечить поддержку протокола Kerberos в Windows 2000 и более поздних версиях. Реализация Kerberos в Windows 2000 является примером использования закрытых элементов. Например, некоторые поля мандатов Kerberos применяются для передачи информации о группах, в которые входит клиент. Реализация Kerberos от Microsoft допускает взаимную аутентификацию, когда не только сервер проводит аутентификацию клиента, но и наоборот, клиент аутентифицирует сервер.

Компания Microsoft предлагает еще один способ установки сеанса связи между клиентом и сервером, который называется *Netlogon*. При этом используются данные о компьютере (а не о пользователе). Протокол Netlogon необходим для установки безопасного сеанса RPC и имеет намного больше возможностей, так как поддерживает маркеры доступа пользователей, которые не поддерживаются при регистрации средствами протокола CIFS. Обычно Netlogon используется для связи между серверами (один сервер выступает

в роли клиента другого сервера). В ныне устаревшем документе RFC от Microsoft протокол Netlogon не описан.

Наконец, сервер CIFS не обязательно должен обеспечивать работу механизма аутентификации. Протокол CIFS поддерживает сквозную аутентификацию, когда сервер получает запрос у другого сервера, передает этот запрос клиенту и возвращает ответ клиента серверу аутентификации. При этом, если сервер аутентификации отвечает положительно, клиенту предоставляется доступ к запрошенным ресурсам. Этот механизм известен как *сквозная аутентификация.*

3.3.5 Возможности по оптимизации CIFS

Протокол CIFS обладает различными возможностями по оптимизации взаимодействия между клиентом и сервером. Эти возможности рассматриваются в разделах 3.3.5.1 и 3.3.5.2.

3.3.5.1 Функция CIFS AndX

Протокол CIFS позволяет формировать последовательность взаимно зависящих друг от друга запросов, поэтому оптимизация этих операций позволяет завершить выполнение запроса за одно обращение к серверу. Эта функция называется *AndX*; Файловая система NFS версии 4 обеспечивает подобную функцию в виде процедуры `COMPOUND`. Примером может быть отправка запросов `OpenAndRead` или `WriteAndClose` серверу CIFS. При этом вместо отправки отдельных двух запросов, например `Open`, а затем `Read`, и получения двух ответов отправляется один запрос `OpenAndRead` и получается один ответ. Это имеет особое значение в том случае, когда время обращения запрос/ответ слишком велико.

3.3.5.2 Оппортунистическая блокировка

Протокол CIFS поддерживает такую технологию оптимизации производительности, как *оппортунистическая блокировка* (opportunistic locking, или oplock). Существует две основные причины для использования оппортунистической блокировки.

Первая заключается в блокировке файла и инициализации его локального кэширования. Когда условие блокировки больше не может поддерживаться, протокол допускает определенную задержку, в течение которой клиент должен очистить кэш. Блокировка и разблокировка выполняются незаметно для приложения с помощью механизмов CIFS клиентской и серверной систем. При этом для повышения производительности модификации приложения не требуется.

Представьте себе приложение, которое открывает файл на сетевом сервере для чтения и записи и записывает в файл 128-байтовые записи. Без оппортунистической блокировки каждая запись размером 128 байт потребует передачи данных по сети. Использование oplock позволяет локально кэшировать файл на клиентской системе и объединять несколько операций записи в одну, которая приводит к передаче данных по сети. Например, предположим, что клиент использует буферы размером 4096 и последовательно записывает в файл по 128 байт. Первый буфер будет содержать данные 32 операций записи ($4096/128 = 32$), и все данные 32 записей будут переданы по сети одним запросом на запись в файл. Если операция записи не может быть кэширована, по сети будет передаваться 32 операции записи (а не одна, как при кэшировании). Сокращение количества операций записи с 32 до одной приводит к значительному снижению нагрузки на сеть и существенному повышению производительности.

Вторым назначением оппортунистической блокировки является расширение условий, при которых подобная блокировка возможна. При использовании oplock можно увеличить эффективность кэширования. Расширение условий, при которых возможна оппортунистическая блокировка, предоставляет несколько дополнительных преимуществ. Предположим, что экземпляр приложения открывает файл (на сетевом сервере) для чтения и записи. При этом запрашивается и предоставляется оппортунистическая блокировка. Программный код клиента может кэшировать операции записи в файле. Предположим, что другой экземпляр того же приложения запущен на другом клиенте. Одним из выходов из подобной ситуации будет снятие оппортунистической блокировки и использование сетевого ввода-вывода для передачи запросов на запись в файл от обоих приложений. Еще одним способом будет снятие оппортунистической блокировки в тот момент, когда второй экземпляр приложения попытается выполнить операцию записи. Очень часто приложения вообще не выполняют операции записи.

При изменении условий сервер отправляет уведомление клиенту о снятии оппортунистической блокировки. В качестве примера ситуации, когда сервер отправляет уведомление о снятии оппортунистической блокировки, можно указать запрос на доступ к файлу или запись данных в файл другим клиентом. Сервер обеспечивает очистку данных состояния сервера (включая закрытие сеанса клиента), если клиент не отвечает на запрос о снятии оппортунистической блокировки. Клиент запрашивает oplock только в случае необходимости; например, если приложение запрашивает открытие файла для эксклюзивного доступа, запрос оппортунистической блокировки просто не имеет смысла.

Оппортунистические блокировки реализуются в трех вариантах:

Рис. 3.3. Последовательность операций при эксклюзивной оппортунистической блокировке

- эксклюзивная оппортунистическая блокировка;
- пакетная оппортунистическая блокировка;
- оппортунистическая блокировка второго уровня.

Далее эти сценарии описываются более подробно.

Эксклюзивная оппортунистическая блокировка

Этот вариант блокировки запрашивается мини-перенаправителем CIFS, когда приложение открывает файл для чтения или записи. Сервер предоставляет оппортунистическую блокировку, если файл еще не открыт другим клиентом. Последовательность операций показана на рис. 3.3.

Для начала первый клиент отправляет запрос на открытие файла, запрашивая эксклюзивную оппортунистическую блокировку. Сервер выполняет необходимую проверку и предоставляет ее. Первый клиент начинает кэширование файла, выполняя операции упреждающего чтения и отложенной записи. Через некоторое время другой клиент, например клиент 2 (на рис. 3.3 не показан), отправляет серверу запрос на открытие того же файла. Сервер

отмечает, что клиент 1 владеет эксклюзивной оппортунистической блокировкой для запрошенного файла, и отправляет уведомление о снятии блокировки клиенту 1. Клиент 1 очищает буферы данных, отправляя необходимые запросы на запись и на блокировку файла. Как только данные состояния файла будут записаны, клиент 1 сообщает серверу, что обработка уведомления о снятии оппортунистической блокировки завершена. На этом этапе сервер отправляет ответ клиенту 2, позволяя ему открыть файл. Клиент 1 продолжает работу с файлом, не проводя при этом локального кэширования. В данном случае предполагается, что клиент 1 открыл файл в режиме, допускающем открытие файла другими клиентами.

Оппортунистическая блокировка второго уровня

Очень часто клиенты открывают файл в режиме чтения/записи и ничего не записывают в файл до его закрытия. Оппортунистическая блокировка второго уровня проектировалась для обеспечения совместного использования и кэширования файлов в такой ситуации. Эксклюзивная оппортунистическая блокировка и пакетная оппортунистическая блокировка (она рассматривается в следующем разделе) всегда предоставляются по запросу клиента. Но оппортунистическая блокировка второго уровня никогда не запрашивается клиентом. Клиент начинает с запроса эксклюзивной оппортунистической блокировки. Если такая блокировка предоставляется, сервер при выполнении определенных условий (они описаны далее) может понизить эксклюзивную оппортунистическую блокировку до блокировки второго уровня.

Обратимся к рис. 3.4. Клиент 1 начинает работу с запроса эксклюзивной оппортунистической блокировки и приступает к локальному кэшированию файла. В частности, клиент 1 проводит упреждающее чтение и локально кэширует блокируемые данные. Помните, что в данном случае клиент не собирается записывать данные в файл. На определенном этапе клиент 2 (он не показан на рис. 3.4) запрашивает доступ к этому же файлу. Сервер отправляет клиенту 1 уведомление с требованием понизить эксклюзивную оппортунистическую блокировку до блокировки второго уровня. Клиент аннулирует блокировки и сообщает, что обработка уведомления завершена. Далее сервер отправляет клиенту 2 сообщение об успешном открытии файла и предоставляет клиенту оппортунистическую блокировку второго уровня. В данном случае предполагается, что клиент 1, открывая файл, сообщил серверу, что другие клиенты также могут получить доступ к файлу.

При использовании оппортунистической блокировки второго уровня, клиентам запрещено буферизировать блокируемые данные. Преимущество этой схемы заключается в упрощенной когерентности данных на стороне сервера, в то время как клиент будет кэшировать считываемые данные, что позво-

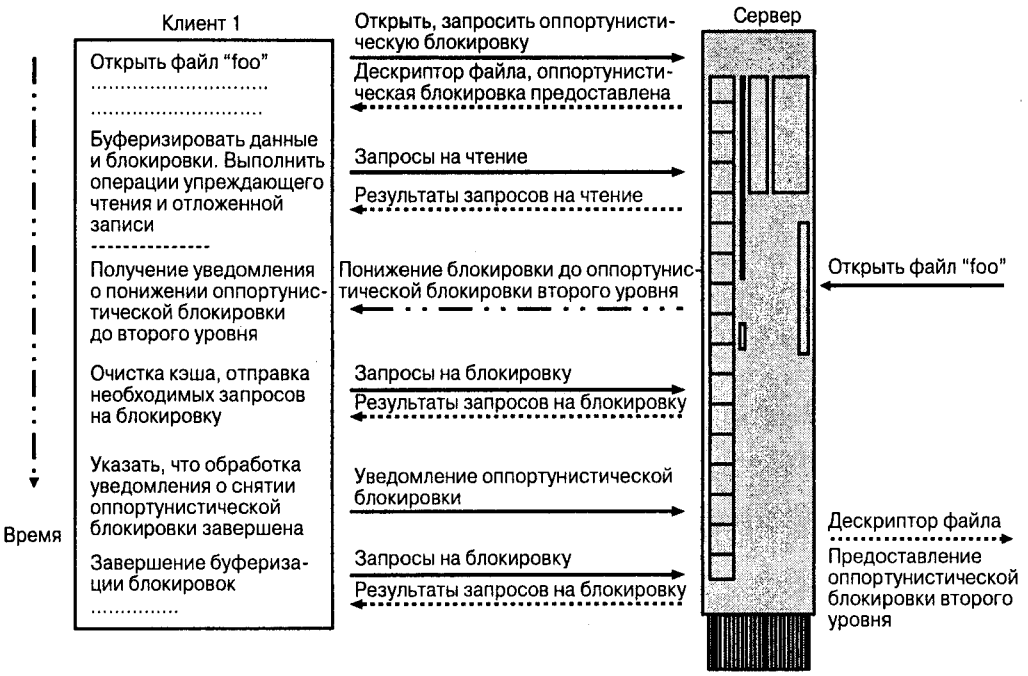


Рис. 3.4. Оппортунистическая блокировка второго уровня

ляет сократить объем информации, передаваемой по сети. Как только один из клиентов выдает запрос на запись, сервер снимает оппортунистическую блокировку второго уровня, после чего блокировок вообще не остается. Поскольку ни один из клиентов не буферизирует блокируемые данные при наличии блокировок второго уровня, успешная операция записи означает, что запись выполнялась в область файла, не заблокированную другими клиентами. После снятия оппортунистической блокировки второго уровня клиентам запрещается буферизировать считанные данные.

Пакетная оппортунистическая блокировка

Этот вариант блокировки используется для оптимизации быстродействия при обработке командных файлов. Командный процессор обычно открывает файл, ищет необходимую строку, считывает ее, закрывает файл и обрабатывает прочитанную строку средствами интерпретатора командной строки. После этого файл открывается снова, в нем находится следующая строка, файл закрывается, и следующая строка обрабатывается интерпретатором командной строки. Этот цикл выполняется, пока все строки в командном файле не закончатся.

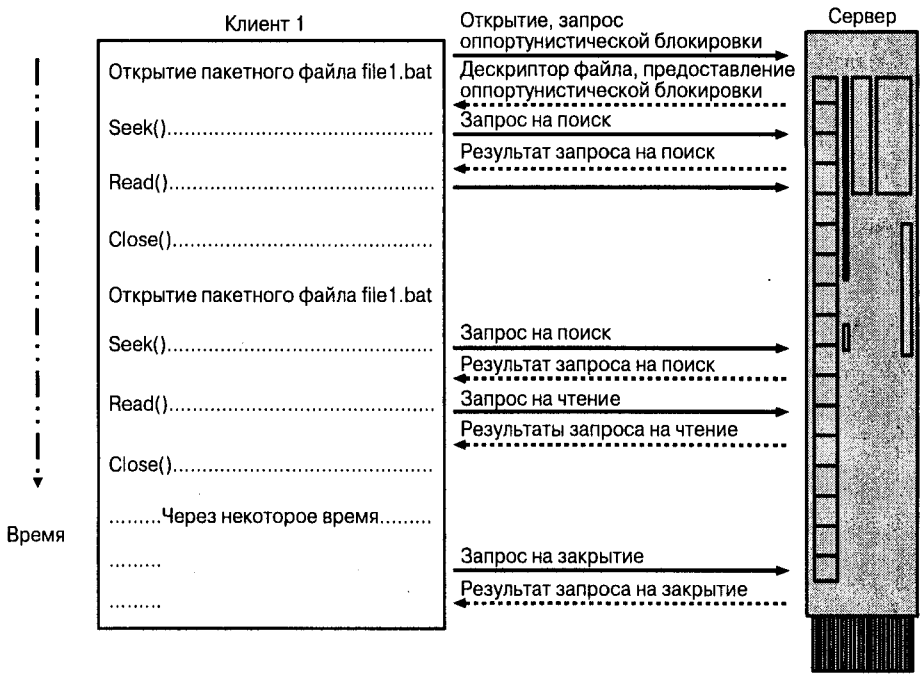


Рис. 3.5. Пакетная оппортунистическая блокировка

На рис. 3.5 приведена последовательность операций. Клиент 1 открывает командный файл и запрашивает пакетную оппортунистическую блокировку. Предположим, что сервер предоставляет пакетную блокировку, так как больше никто не выполняет запись данных в файл. Клиент 1 ищет в файле определенную строку и осуществляет операцию чтения. Интерпретатор выполняет прочитанную строку. Затем файл закрывается. Мини-перенаправитель CIFS не выполняет никаких действий при получении запроса на закрытие файла (т.е. выполняется операция отложенного закрытия). Интерпретатор командной строки открывает файл, но мини-перенаправитель CIFS не выполняет операцию открытия, а просто отменяет размещенную в очереди операцию отложенного закрытия файла. Когда интерпретатор командной строки выполняет операции поиска и чтения строки, мини-перенаправитель CIFS отправляет запросы на поиск и чтение.

В результате сокращается объем данных, передаваемых по сети (выполняется меньше запросов на открытие и закрытие файла). Кроме того, оптимизируется использование ресурсов сервера, так как от сервера не требуется немедленной обработки запроса на закрытие файла с последующим его повторным открытием.

3.4 Сетевая файловая система

Файловая система CIFS доминирует на рынке сетевых файловых систем для платформы Windows. На платформе UNIX основной является сетевая файловая система (Network File System — NFS). Кроме того, NFS считается первой широко распространенной файловой системой, что произошло еще в середине 1980-х годов. Однако, несмотря на некоторые общие функциональные возможности CIFS и NFS (это сетевые файловые системы, позволяющие клиентам получать доступ к ресурсам серверов), эти системы имеют совершенно различные архитектурные особенности. С выходом NFS версии 4 некоторые различия были пересмотрены.

Протокол CIFS сохраняет сервисные данные, относящиеся к каждому клиенту. До версии 3 файловая система NFS не сохраняла статус клиента, что изменилось в версии 4.

Клиент NFS не "договаривается" с сервером NFS об установлении сеанса. Меры безопасности предпринимаются для всего сеанса или каждой операции обмена данными между клиентом и сервером. Реализация последнего варианта чрезмерно дорогостоящая, поэтому NFS возлагает задачу обеспечения безопасности на клиента. Сервер "предполагает", что идентификаторы пользователя на клиентских и серверной системах совпадают (а клиент проверил личность пользователя перед тем, как дать ему зарегистрироваться под указанным идентификатором). Кроме того, NFS обеспечивает определенный уровень безопасности, контролируя список файловых систем, которые может монтировать клиент. Каждый раз, когда клиент CIFS открывает файл, получает дескриптор файла (т.е. сервисные данные, которые должен сохранять сервер) и использует его для проведения операций чтения или записи на стороне клиента, сервер NFS запрашивает сервер, который возвращает дескриптор файла. Этот дескриптор файла обрабатывается клиентами, поддерживающими стандарты NFS 3 и NFS 2. Клиент кэширует полученный дескриптор файла и ожидает, что дескриптор всегда будет указывать на один и тот же файл.

Для тех, кто знаком с UNIX, можно отметить, что дескриптор файла обычно состоит из *номера inode* (inode number), *счетчика поколения inode* (inode generation count) и идентификатора файла, который связан с разделом диска. Достаточно сказать, что inode представляет собой исключительно важную структуру данных, которая используется в файловых системах UNIX. Для удаления дескрипторов, кэшированных клиентами, хранится достаточный объем информации, необходимой, если соответствующий дескриптору файл изменился и дескриптор должен указывать на другой файл. Например, если файл удален и на его место скопирован файл с таким же именем, счетчик поколения inode будет изменен и кэшированный клиентом де-

скриптор файла окажется недействительным. Файловая система NFS 4 имеет отличия в реализации, которые рассматриваются в разделе 3.4.2.

Некоторые клиенты NFS проводят кэширование на стороне клиента, храня данные на дисках, что напоминает кэширование в CIFS. Также некоторые клиенты NFS меняют значение тайм-аутов в зависимости от времени отклика сервера. Чем медленнее отзывается сервер, тем больше значение тайм-аута, и наоборот.

Файловая система NFS проектировалась, как независящая от транспорта и изначально использовала транспортный протокол UDP. Различные типы NFS могут использовать протокол TCP и другие протоколы.

В разделах 3.4.1 и 3.4.2 рассматриваются особенности файловых систем NFS 3 и NFS 4. Информация, приведенная в этой книге, не заменит ряд отличных изданий, которые представлены в списке источников информации в конце книги.

3.4.1 Сетевая файловая система, версия 3

Файловая система NFS 3 позволяет увеличить быстродействие, особенно для больших файлов, разрешая клиенту и серверу динамически выбирать максимальный объем данных, которые передаются в одном логическом элементе пакета при записи или чтении. В файловой системе NFS 2 на размер пакета накладывалось ограничение в 8 Кбайт. Другими словами, клиент мог отправить максимум 8 Кбайт в запросе на запись, а сервер — максимум 8 Кбайт в ответе на запрос чтения. Кроме того, в NFS 3 переопределены смещения в файлах и размеры данных. Теперь это 64-разрядные значения, вместо 32-разрядных в NFS 2.

Далее представлены некоторые особенности NFS 3.

- В дескрипторах файлов в NFS 3 указан переменный размер; их максимальных размер составляет 64 бит.
- Файловая система NFS 3 позволяет клиентам и серверам выбирать максимальный размер имен файлов и каталогов.
- В NFS 3 определяется список ошибок, которые сервер может возвращать клиентам. Сервер должен вернуть одну из определенных ошибок или не возвращать ошибку вообще.
- В NFS 3 серверу разрешено кэшировать данные, которые клиент отправил вместе с запросом на запись. Сервер может кэшировать данные и отправлять клиенту ответ на запрос еще до того, как данные будут записаны на диск. Также добавлена команда `COMMIT`, которая позволяет клиенту убедиться, что все отправленные данные были записаны на

диск. Это дает возможность соблюсти баланс между повышением производительности и сохранением целостности данных.

- В NFS 3 сокращено количество операций запрос/ответ между клиентом и сервером. Для этого данные об атрибутах файла отправляются вместе с первоначальным запросом. В NFS 2 от клиента требовалось получение имен файлов и дескриптора для каждого файла, только после этого передавались атрибуты файла.

3.4.2 Сетевая файловая система, версия 4

В NFS 4 полностью пересмотрены основополагающие принципы и реализовано много функций, характерных для CIFS, что весьма расстроило некоторых апологетов NFS. Если посмотреть на историю сетевых файловых систем, то можно увидеть, что NFS получила широкое распространение. Файловая система SMB разрабатывалась с учетом сильных и слабых сторон NFS и теперь, по крайней мере в среде клиентов, CIFS/SMB распространены больше, а NFS развивается, учитывая все недостатки и преимущества CIFS/SMB. Ниже рассматриваются возможности, которые были добавлены в NFS 4 для повышения быстродействия и безопасности, а также для улучшения взаимодействия с CIFS.

- В NFS 4 появился запрос `COMPOUND`, который позволяет запаковывать несколько запросов в один запрос и несколько ответов в один ответ. Это нововведение предназначено для повышения производительности за счет снижения нагрузки на сеть и сокращения задержек при передаче запросов и ответов по сети. Если это несколько напоминает функцию CIFS AndX SMB (см. раздел 3.3.5.1), то, возможно, дело не в обычном совпадении.
- Сетевая файловая система версии 4 заимствовала некоторые возможности у WebNFS, созданной компанией Sun. В частности, в NFS 4 некоторые вторичные протоколы поддерживаются в базовой спецификации, что делает NFS более подходящей для применения вместе с брандмауэрами. В NFS 3 и более ранних версиях использовался специальный протокол для монтирования общего ресурса сервера в дерево локальной файловой системы. Поскольку служба протокола монтирования не имела назначенного порта TCP или UDP, клиент сначала отправлял запрос службе отображения портов (portmapper daemon), предоставляющей номер порта, посредством которого ожидает запросов служба монтирования. Таким образом, кроме NFS, в процессе принимали участие протоколы монтирования и отображения портов. Более того, так как служба монтирования могла использовать произвольный порт, настрой-

ка брандмауэра весьма усложнялась. В NFS 4 протоколы монтирования и отображения портов были исключены. Кроме того, блокирование было включено в базовую спецификацию протокола NFS, а протокол NLM (Network Lock Manager), который применялся в более ранних версиях NFS, окончательно устарел.

- Файловая система NFS 4 требует использования транспортного протокола, который предоставляет возможность обнаружения "заторов" в сети. Это значит, что клиенты и серверы NFS постепенно будут переходить к протоколу TCP вместо UDP, который обычно используется вместе с NFS 3.
- В NFS 2 и NFS 3 допускалось использование набора символов U.S. ASCII или ISO Latin 1. Это приводило к возникновению проблем, когда клиент, использующий один набор символов, создавал файл и к этому файлу получал доступ клиент с другим набором символов. В NFS 4 используется набор символов UTF-8, который поддерживает компактное сжатие 16- и 32-разрядных символов для их передачи по сети. Кроме того, набор символов UTF-8 содержит достаточный объем информации, чтобы избежать проблем при создании файла посредством одного набора символов и получении доступа к файлу с другим набором.
- Файловая система NFS 4 требует от клиента отдельной обработки дескрипторов файлов. В NFS 3 клиент мог кэшировать дескриптор в качестве объекта, в то время как сервер заботился о том, чтобы дескриптор всегда указывал на файл. В NFS 4 определены два типа файловых дескрипторов. Один называется *постоянные дескрипторы файлов* и обладает возможностями дескрипторов файлов из NFS 3. Второй — *временные дескрипторы файлов* — предполагает истечение срока действия дескриптора после определенного промежутка времени или события. Это функция для серверов, файловые системы которых (например, NTFS) не могут обеспечить постоянного соответствия между отображаемыми файлами и дескрипторами.
- В NFS 4 добавлена поддержка операций `OPEN` и `CLOSE`, семантика которых допускает взаимодействие с клиентами CIFS. Команда `OPEN` создает данные состояния на сервере.
- Поддержка запроса `OPEN` в NFS 4 позволяет клиенту осуществлять запрос на открытие файла, структура которого будет аналогична запросам на открытие приложений Windows. Также поддерживается выбор совместного использования файла с другими клиентами или эксклюзивный доступ к файлу.

3.4.2.1 Безопасность NFS 4

Файловая система NFS 4 позволяет усилить безопасность хранимых данных. В частности, в NFS 4 добавлена поддержка большего количества атрибутов файла. К одному из этих атрибутов относится список управления доступом (ACL) в стиле Windows NT. Это позволяет улучшить взаимодействие между файловыми системами и укрепить структуру безопасности.

В то время как в NFS 2 и NFS 3 использование возможностей системы безопасности только рекомендовалось, в NFS 4 это стало обязательным. Файловая система NFS 4 требует реализации механизма безопасности с помощью интерфейса RPCSEC_GSS (Generic Security Services) в общем и протоколов Kerberos 5/LIPKEY в частности. Обратите внимание, что RPCSEC_GSS просто выполняет роль интерфейса API и транспортного механизма для меток и данных, связанных с безопасностью. Файловая система NFS 4 позволяет использовать несколько схем аутентификации и обеспечения безопасности, а также дает возможность выбрать подходящую схему для клиентов и серверов.

Уделим некоторое внимание изучению технологии LIPKEY, использующей комбинацию симметричного и асимметричного шифрования. Клиент шифрует данные о пользователе и пароль, применяя случайно сгенерированный ключ размером 128 бит. Шифрование выполняется с помощью симметричного алгоритма, т.е. для дешифрации должен использоваться тот же ключ. Поскольку серверу необходим этот ключ для дешифрации сообщения, случайно сгенерированный ключ должен быть отправлен серверу. Клиент шифрует ключ (который генерируется случайно) с помощью открытого ключа сервера. Сервер дешифрует данные своим закрытым ключом, извлекает симметричный ключ и дешифрует данные о пользователе и пароль.

Клиенты могут аутентифицировать серверы по серверному сертификату, а для проверки сертификата используются службы сертификационного центра. Одним из популярных методов взлома является перехват "чужих" пакетов данных с их последующей отправкой через некоторый временной промежуток. При использовании Kerberos файловая система NFS добавляет в каждый пакет временную метку. Сервер записывает недавно полученные временные метки и сравнивает их с временными метками новых пакетов RPC. Если временные метки пакетов старше, чем полученные сервером ранее, сервер игнорирует полученные пакеты.

3.5 Проблемы доступа при использовании нескольких протоколов

Несколько компаний стали предлагать системы, в которых одновременно реализована поддержка CIFS, NFS и других клиентов сетевых файловых систем. Поставщики проделали немалую работу, пытаясь преодолеть технические проблемы, которые возникают из-за потенциального использования клиентами различных операционных и файловых систем. Обратите внимание, что проблемы возникают не с самими данными, а с метаданными файлов. Простым тестом на наличие подобных проблем будет копирование файла с сервера на клиент и обратно на сервер (или наоборот). После размещения файла в первоначальном ресурсе метаданные должны содержать базовые значения, т.е. права доступа к файлу и временные метки не должны измениться. Если это не соответствует истине, то проблема обнаружена.

Далее представлены примеры некоторых возможных технических проблем.

- В различных операционных системах используются разные методы для отслеживания разрешений доступа пользователей и групп.
- В различных операционных и файловых системах существует разная семантика открытия и блокировки файлов.
- Соглашения по именованию файлов обрабатываются разными способами. Различные файловые системы по-разному представляют максимальный размер имени файла, значение регистра в имени файла и набор символов, допустимый в именах.
- Данные и их структура различаются в различных файловых системах; например, одни файловые системы отслеживают две временные метки, в то время как другие — три метки (время последнего доступа к файлу, последней модификации и создания файла). Даже если обе файловые системы отслеживают две временные метки, единицы измерения могут отличаться. Еще одним примером служат единицы измерения смещений в файлах. В некоторых файловых системах поддерживаются 32-разрядные смещения, а в некоторых — 16- или 64-разрядные.
- Проблемы с адресацией отображаемых блокировок. Сервер CIFS принудительно поддерживает блокировку: если один клиент заблокировал область файла, то любая операция записи в эту область файла со стороны другого клиента приведет к возникновению ошибки. Однако принудительная блокировка не поддерживается серверами NFS. Поэтому необходимо выбрать, будет ли блокировка поддерживаться принудительно, что приведет к отправке сообщения об ошибке клиенту NFS.

3.6 Windows и NAS

Компания Microsoft предлагает приобрести коммерческий пакет SAK (Server Appliance Kit). Он предназначен для производителей аппаратного обеспечения, которым требуется создать приложения для хранилищ данных, подключаемых к сети. Пакет SAK — это просто версия Windows, имеющая модульную структуру, а также набор средств программной разработки. Модульная структура версии Windows позволяет производителю выбрать (в пределах разумного) компоненты Windows, которые должны быть включены в готовое приложение. Таким образом, предоставляется возможность убрать из Windows все компоненты, которые не требуются для поддержки стека сетевых протоколов, файлового сервера и локального стека подсистемы хранения данных. Программные средства разработки позволяют производителям оборудования создавать собственные инструменты администрирования и распространять их вместе с аппаратными комплексами.

Кроме того, в состав SAK входят компоненты, не поставляемые с обычными серверными версиями Windows. Хорошим примером будет инфраструктура обеспечения надежности, которая следит за драйверами на предмет утечки памяти и незаметно перезапускает критические компоненты, если в этом возникает необходимость. Несколько поставщиков предлагают устройства NAS, работающие под управлением Windows NT. Одно из преимуществ использования Windows NT для организации NAS заключается в сокращении стоимости и временных затрат на создание готового решения. Компания Microsoft предоставляет стеки протоколов CIFS и NFS. Возможно, одно из главных преимуществ использования Windows в подобных системах — это доступность всех новейших функций SMB и отсутствие необходимости в обратном проектировании и дополнительной разработке.

Следующим преимуществом является возможность модификации программных решений посредством SAK. В состав SAK входят компоненты управления (в частности, удаленного управления) и компоненты защиты данных (например, функции для создания "моментальных снимков" образа диска). (Описание технологии моментальных снимков и сферы их применения приводится в главе 5.)

Некоторые производители аппаратного обеспечения разработали свои решения NAS, не используя пакет SAK. Доступно несколько альтернативных стеков протоколов CIFS и NFS. Феномен открытого исходного кода набирает популярность, и, если есть возможность потратить дополнительные ресурсы, использование исходного кода, допускающего внесение изменений, имеет ряд преимуществ. Например, доступность открытого исходного кода позволяет максимально сократить накладные расходы на уровне операционной систе-

мы. Маркетинговое исследование этих подходов не является предметом нашего обсуждения.

3.7 Система Microsoft Exchange 2000 и NAS

Популярная корпоративная система обмена мгновенными сообщениями Microsoft Exchange 2000 представляет собой один из главных продуктов в семействе BackOffice от компании Microsoft. Архитектура Exchange 2000 значительно отличается от Exchange 5.5. Хотя в Microsoft Exchange 2000 было включено множество новых возможностей, не обошлось и без недостатков; один из них состоит в том, что эту систему нельзя использовать совместно с устройствами хранения NAS. В данной книге рассматриваются те компоненты Exchange 2000, которые имеют практическое значение.

На рис. 3.6 показан элемент архитектуры Microsoft Exchange 2000. Обратите внимание, что в данном случае рассматриваются только отдельные компоненты системы Exchange 2000, полное описание которой выходит за рамки книги. Протокольное ядро, показанное на рис. 3.6, предоставляет серверные функции для таких почтовых протоколов, как IMAP, POP3 и SMTP. Ядро Exchange Store (ESE) основано на базе данных Jet и используется для хранения и извлечения различных объектов, которые и формируют сообщение электронной почты. Файловая система ExIFS (Exchange Installable File System) предоставляет важные функции, описанные ниже.

- Исключительно эффективный механизм межпроцессного взаимодействия для ESE и протокольного ядра Exchange.

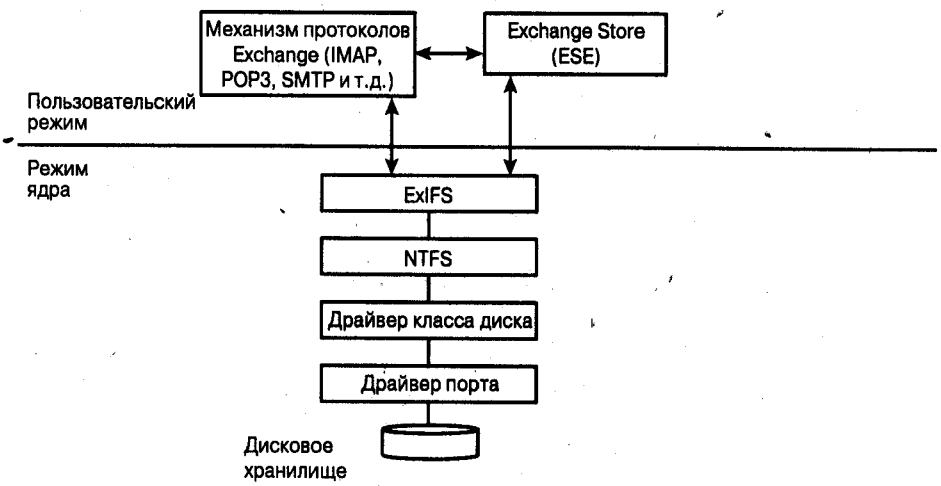


Рис. 3.6. Архитектура подсистемы хранения Microsoft Exchange 2000

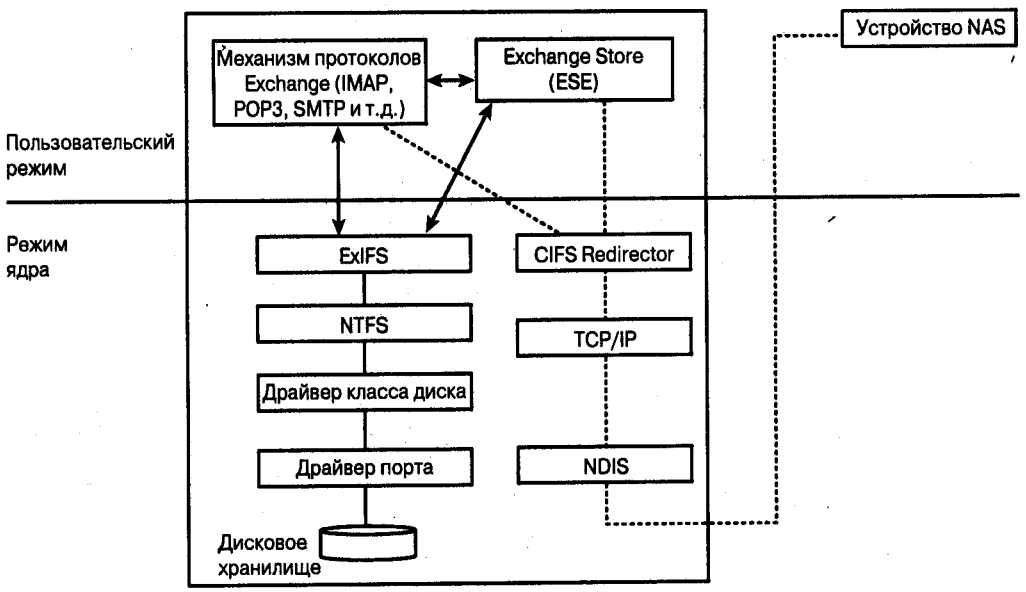


Рис. 3.7. Система Microsoft Exchange 2000 при использовании локального хранилища и хранилища NAS

- Межпротокольный кэш файловых дескрипторов Windows NT. Дескриптор, созданный одним протоколом (например, POP), может использоваться другим протоколом (например, DAV).
- Функции для обеспечения минимального преобразования данных, например хранение сообщений в базовом формате MIME.

Хотя ExIFS и предоставляет необходимые возможности, с точки зрения поставщиков решений NAS, эта файловая система является причиной основных проблем. В частности, при использовании NAS для хранилища Exchange драйвер ExIFS исключается из маршрута ввода-вывода данных (рис. 3.7).

Обратите внимание, что запросы Exchange Store или протокольного ядра Exchange к локальным файлам проходят через стек протоколов, изображенный в левой части диаграммы. Например, запрос на чтение проходит от Exchange Store к ExIFS, оттуда к NTFS, драйверу класса диска и драйверу порта. Но если запрос требует обращения к файлу, который хранится на устройстве NAS, то драйвер ExIFS находится вне маршрута ввода-вывода, поэтому запросы проходят через сетевой стек, изображенный в правой части диаграммы. Таким образом, запрос от Exchange Store будет направлен к перенаправителю CIFS, оттуда в стек протоколов TCP/IP и к сетевой карте. Драйвер ExIFS в этом маршруте отсутствует.

Обратите внимание: архитектурные особенности, представленные на рис. 3.6 и рис. 3.7, относятся только к Microsoft Exchange 2000 и не имеют никакого отношения к Microsoft Exchange 5.5.

3.8 Сложности практической реализации

Начиная с Windows 2000, компания Microsoft предоставляет новую модель мини-драйвера, которая позволяет независимым поставщикам создавать сетевые файловые системы. От поставщиков требуется создать мини-драйвер, который поддерживает особенности протокола выбранной сетевой файловой системы. Общие для всех сетевых файловых систем возможности (взаимодействие с Windows и диспетчером кэша) обеспечиваются библиотекой драйвера. Компания Microsoft предоставляет мини-драйверы для файловых систем CIFS, NFS и WebDAV.

На данный момент CIFS де-факто является индустриальным стандартом для обеспечения взаимодействия между клиентами и серверами Windows. Файловая система CIFS реализована на различных платформах, поддерживающих сетевые файловые системы. Компания Microsoft, как и ассоциация SNIA (Storage Networking Industry Association), предоставляет бесплатную спецификацию CIFS. В обоих спецификациях CIFS рассматривается более старый протокол, реализованный в Windows NT 4.0. Кроме того, Microsoft за определенную плату предоставляет спецификации протокола SMB, который используется в более новых версиях Windows NT.

Компания Microsoft вступила в конкурентную борьбу с поставщиками решений NAS, построенных на платформах, отличных от Windows. Со своей стороны Microsoft предоставляет поставщикам устройств модифицированную версию операционной системы Windows. Время покажет, насколько успешной окажется эта стратегия, но на данный момент компании Microsoft удалось занять заметную нишу рынка операционных систем для решений NAS.

Программный продукт Microsoft Exchange 2000 не работает с устройствами NAS. Ожидается, что со временем компания Microsoft предложит версию этого продукта, которая поддерживает хранение данных на базе программы SQL Server. Учитывая, что новая версия SQL все еще не появилась в продаже, гарантировать сроки появления новых версий этих продуктов невозможно. Предпримет ли Microsoft шаги, необходимые для решения проблемы взаимодействия Microsoft Exchange 2000 и NAS? Помните о реальных сроках выхода продуктов на рынок и регулярных их переносах. Несмотря на то что Microsoft Exchange 2000 уже давно доступен на рынке, множество клиентов продолжают использовать Microsoft Exchange 5.5.

3.9 Резюме

Сетевое хранилище данных (NAS) — это промышленный термин, которым описывается определенный класс устройств, включающих в себя жесткие диски, адаптер сетевого интерфейса и центральный процессор, обрабатывающий данные протокола файлового сервера. Устройства NAS обычно настраиваются на максимальную производительность (с помощью специально настроенных операционных систем) и предоставляют удобные возможности для администрирования. При этом проблема администрирования устройств хранения данных превращается в более понятную проблему — администрирование сервера.

Обычно к устройствам NAS обращаются клиентские системы. Обратите внимание: в данном случае *клиент* — термин относительный. Клиентом устройства NAS может быть сервер баз данных. Клиенты используют протоколы CIFS/SMB или NFS. Первый, как правило, применяется клиентами, работающими под управлением Windows. Протокол NFS, в свою очередь, предназначен для клиентов на базе UNIX. Спецификации протокола CIFS разрабатываются компанией Microsoft и организацией SNIA. В этих спецификациях рассматриваются реализации протокола, которые характерны для Windows NT 4.0. Лицензия на протокол SMB для Windows 2000 и Windows Server 2003 предоставляется Microsoft за определенную плату.

Устройства NAS, которые одновременно обслуживают клиентов NFS и CIFS/SMB, характеризуются наличием серьезных проблем при обеспечении взаимодействия клиентов и сохранении метаданных файлов.

Глава 4

Сети хранения данных на базе интерфейса Fibre Channel

Эту главу можно рассматривать как введение в сети хранения данных (storage area network — SAN) в общем и в сети хранения данных на базе интерфейса Fibre[1] Channel в частности. Хотя сети хранения данных могут создаваться и на основе технологий, отличных от Fibre Channel, большинство из них будут использовать Fibre Channel еще достаточно долгое время. Именно поэтому интерфейсу Fibre Channel уделяется в этой главе основное внимание. Сети хранения данных, основанные на других технологиях, например iSCSI, рассматриваются в главе 8.

Интерфейс Fibre Channel — это технология межсистемного взаимодействия, которая объединяет в себе возможности высокоскоростного ввода-вывода и сетевого обмена данными. Когда эта книга готовилась к печати, сети хранения данных на основе Fibre Channel обеспечивали быстродействие 1 Гбит/с; кроме того, растет количество сетей, поддерживающих скорость 2 Гбит/с.

В терминологии Fibre Channel устройства называются *узлами* (nodes). Это весьма напоминает узлы в терминологии сетей IP. Узел Fibre Channel может иметь несколько портов, как и узел IP, который зачастую получает несколько адресов IP. Разница между ними в том, что порт Fibre Channel представляет собой физический элемент, а порт IP — логический. Каждый узел Fibre Channel имеет уникальное 64-разрядное имя WWN (World Wide Name), которое назначается производителем. Это напоминает уникальные адреса MAC, которые назначаются производителями сетевым адаптерам Ethernet. Каждый порт сети хранения данных на базе кольца с разделением доступа Fibre Channel имеет 8-битовый адрес, а порт в коммутируемой связной архитектуре — 24-битовый. При подключении *кольца с разделяемым доступом* (arbitrated loop) к *коммутатору связной архитектуры* (fabric switch),

[1]Написание Fiber было заменено на Fibre, чтобы указать, что технология Fibre Channel может использовать медные и оптические носители.

коммутатор представляет 8-битовый адрес в виде 24-битового. Оба идентификатора порта назначаются динамически. Концепция портов и различные их типы обсуждаются в разделе 4.5.

Сфера применения Fibre Channel рассматривается в следующем разделе, после чего технология SAN сравнивается с NAS (см. главу 3). В этой главе сначала описываются принципы технологии Fibre Channel, затем внимание акцентируется на протоколах нижнего уровня и наконец рассматриваются различные элементы (включая устройства), из которых и создаются системы SAN на базе интерфейса Fibre Channel.

4.1 Сферы применения технологии Fibre Channel

В технологии Fibre Channel предпринята попытка объединить лучшее из двух миров — каналов передачи данных и сетей. Термин *канал* впервые стал использоваться в мире мэйнфреймов и описывал структурированный механизм передачи данных. В большинстве случаев передача данных выполняется между компьютерной системой и периферийным устройством, например жестким диском или накопителем на магнитной ленте. К таковым каналам относятся интерфейсы SCSI (Small Computer System Interface) и HIPPI (High-Performance Parallel Interface). Работа каналов обычно реализуются средствами аппаратного обеспечения.

По сравнению с каналом *сеть* представляет собой более универсальный механизм для передачи данных, который, однако, менее структурирован. Кроме того, сеть может работать на значительно большем расстоянии и подключаться к большему количеству устройств, чем канал. В отличие от каналов, сети в основном реализуются средствами программного, а не аппаратного уровня.

Один из подходов в объединении систем хранения данных и сетей заключается в том, что сеть становится ключевым элементом, к которому добавляются новые возможности с одновременной компенсацией недостатков подобного подхода. Речь идет о технологии хранения данных на базе протокола IP (см. главу 8).

Другой подход состоит в использовании центрального хранилища данных (канальная система) и расширения существующих технологических функций. На базе этого метода создавалась технология Fibre Channel. Одно из основных преимуществ Fibre Channel по сравнению с IP Storage заключается в разработке продуктов Fibre Channel уже в течение 10 лет, в то время как решения для IP Storage появились сравнительно недавно.

Учитывая, что Fibre Channel основана на структуре каналов, стоит рассмотреть недостатки другой известной технологии — SCSI.

- Максимальная скорость передачи данных — 80 Мбайт/с (впоследствии скорость возросла до 320 Мбайт/с, но уже после появления технологии Fibre Channel), чего явно недостаточно для хранилищ данных большого объема.
- Адаптер поддерживает только 16 устройств.
- Преимущество, которое одновременно является и недостатком, — обратная совместимость. Стандарт SCSI развивался много лет, и производители устройств смогли обеспечить обратную совместимость для нескольких поколений устройств. Но администратор должен следить, чтобы к шине не подключались устройства предыдущих поколений, поскольку шина автоматически перейдет в режим, поддерживающий работу самого старого устройства.
- Поддерживается длина кабелей, составляющая несколько десятков метров. Этого недостаточно для создания обычных, а тем более географически распределенных кластеров.
- Существуют и другие альтернативы SCSI, например SSA (Serial Storage Architecture), которые еще находятся в рамках архитектуры Intel или вообще представляют собой открытый стандарт.

4.2 Сравнение SAN и NAS

В главе 3 рассматривается технология NAS. Прежде чем знакомиться с архитектурой сетей хранения данных на базе Fibre Channel, следует провести сравнение принципов создания хранилищ. В табл. 4.1 описываются различия и общие черты этих технологий.

Таблица 4.1. Сравнение технологий NAS и SAN

NAS	SAN
NAS предоставляет доступ к файлам по сети. Поскольку пользователи знают о существовании файлов, например файлов электронных таблиц Microsoft Excel или файлов документов Word, устройства NAS более интуитивны для восприятия	SAN предоставляют доступ к дисковым блокам по сети. Пользователи обычно знают о существовании файлов, а файловая система предоставляет уровень абстракции для сокрытия информации о том, в каком дисковом блоке находится определенный файл
NAS использует сетевые протоколы доступа к файлам, например CIFS и NFS. Эти протоколы обычно пользуются услугами протокола TCP/IP, который, в свою очередь, зависит от технологии Ethernet	SAN предоставляет доступ к дисковым блокам по протоколу Fibre Channel. Верхний уровень этого протокола обычно представляет собой протокол SCSI

Окончание табл. 4.1

NAS	SAN
Устройства NAS — это, по сути, серверы, к файлам которых получают доступ клиенты	Устройства SAN — это устройства хранения данных, к которым могут получать доступ программы-инициаторы
Устройства NAS несложно внедрять в локальную сеть. NAS не требует усложнять структуру сети, однако все данные передаются средствами локальной сети	SAN требует создания новой сетевой архитектуры, которая обычно основана на интерфейсе Fibre Channel. Новая сеть снижает вероятность "заторов" в локальной сети и позволяет выполнять такие операции, как резервное копирование вне локальной сети
Подключение устройства NAS к локальной сети не требует больших затрат	Подключение SAN к локальной сети обходится довольно дорого
Обычно устройства NAS подключаются с помощью интерфейса Ethernet со скоростью передачи 100 Мбит/с или 1 Гбит/с, если используется технология Gigabit Ethernet	Сети хранения данных на основе Fibre Channel обычно работают на скоростях 1 или 2 Гбит/с

4.3 Преимущества Fibre Channel

Рассмотрим преимущества использования SAN на базе Fibre Channel. Хотя возможности, описанные в разделах 4.3.1–4.3.7, все еще широко применяются, не забывайте, что такая ситуация может измениться в любой момент, так как технологический прогресс не стоит на месте.

4.3.1 Масштабируемость

Сети хранения данных обладают способностью к масштабированию в аспекте объема и скорости передачи данных. Можно добавить коммутатор и расширить кольцо отдела до магистрали, которая каскадом соединяет иерархии коммутаторов. Конечно, сети IP также поддерживают масштабирование, что является основной причиной такого быстрого развития технологии IP Storage (см. главу 8).

4.3.2 Сегрегация хранилищ

При корректной настройке и управлении SAN предоставляет налучшие возможности обоих вариантов построения хранилищ данных: хранилище отделено от сервера приложений и в то же время обеспечивает защиту и целостность данных. Например, можно разделить данные отделов, которые физически будут находиться в одном пуле устройств хранения данных.

4.3.3 Централизация и управление хранилищем

Сети хранения данных позволяют консолидировать устройства хранения, оптимизировать их использование и управление. Речь идет о том, что дублировать данные, которые размещены вне инфраструктуры SAN, нет необходимости. Еще одним преимуществом является возможность распределения хранилищ. Это позволяет избежать ситуации, когда у одного сервера объем хранилища избыточен, а второму серверу объема не хватает. Эффективное распределение объема хранилища позволяет сократить накладные расходы, связанные с добавлением устройств хранения к определенному серверу. Например, если управление дисками проводится централизованно, хранилище одного сервера может быть передано в использование другому серверу, которому не хватает свободного дискового пространства. Еще одним примером может служить хранилище, необходимое для создания "моментального снимка" в процессе резервного копирования. В классической схеме каждый сервер имеет для этих целей собственный жесткий диск. В технологии SAN все серверы совместно используют несколько дисков для создания временных моментальных снимков.

4.3.4 Поддержка устаревших устройств

Сети хранения данных позволяют защитить инвестиции в устаревшие устройства, например в устройства хранения SCSI, или, с ультрасовременной точки зрения, в кольцевые концентраторы. Устройства-мосты, которые обеспечивают взаимодействие между старыми устройствами и интерфейсом Fibre Channel, часто являются основой для поддержки устаревших устройств. Конечно, такая схема позволяет использовать далеко не все устаревшие устройства.

4.3.5 Поддержка большего количества устройств

Сети хранения данных предоставляют доступ к большему количеству устройств. Кольцо с разделяемым доступом поддерживает теоретический максимум, равный 127 портам (обычно подключается до 15 устройств, а значительные задержки в работе возникают после подключения 50 устройств).

Сеть хранения данных на основе связной архитектуры теоретически поддерживает около 15 млн. (2^{24}) портов.

4.3.6 Расстояние

Сети хранения данных позволяют размещать устройства хранения гораздо дальше от узлов (серверов и рабочих станций), чем другие интерфейсы. Шина SCSI поддерживает расстояния порядка десятков метров, а Fibre Channel — десятков километров.

4.3.7 Новые возможности

Интерфейс Fibre Channel позволяет использовать совершенно новые функции. Один из примеров — резервное копирование данных без использования локальной сети, когда резервная копия передается по сети хранения данных, а локальная сеть остается свободной для запросов клиентов и серверов к устройствам хранения. Еще одним примером является перенос данных между двумя подсистемами хранения без участия сервера. Кроме того, файловые системы SAN (они рассматриваются в главе 6) позволяют нескольким серверам получить одновременный доступ к одним и тем же томам.

4.4 Топологии Fibre Channel

В разделах 4.4.1–4.4.3 рассматриваются различные топологии подключения устройств, которые формируют сеть хранения данных на базе Fibre Channel. Топологии "точка–точка" (point to point), кольцо с разделяемым доступом (arbitrated loop) и коммутируемая связная архитектура (switched fabric) перечислены в порядке возрастания их сложности.

4.4.1 Топология "точка–точка"

Технология Fibre Channel поддерживает подключение по топологии "точка–точка". В этом случае сервер обычно подключается к выделенной подсистеме хранения, причем данные не используются совместно. На рис. 4.1 показана сеть, построенная на основе этой топологии.

Для реализации топологии "точка–точка", как минимум, необходим сервер, адаптер Fibre Channel (*адаптер шины*) и устройство хранения (например, жесткий диск или накопитель на магнитной ленте), оснащенное интерфейсом Fibre Channel.

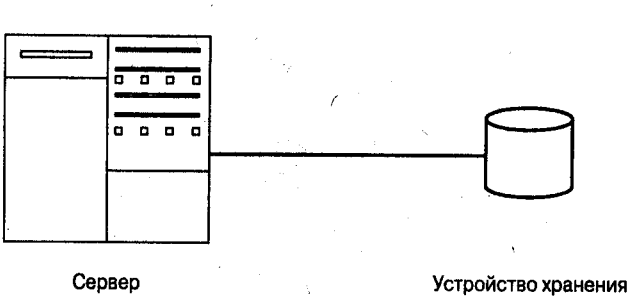


Рис. 4.1. Топология "точка–точка"

4.4.2 Кольцо с разделяемым доступом

Кольцо — это схема логического подключения устройств, при котором данные передаются по логически замкнутому контуру. В *кольце с разделением доступа* (arbitrated loop) протокол описывает порядок, в котором узел получает разрешение на передачу данных. Кольцо Fibre Channel с разделением доступа (Fibre Channel arbitrated loop — FC-AL) может быть реализовано на базе различных устройств хранения (жестких дисков, накопителей на магнитной ленте), серверов, адаптеров шины и устройств для их подключения. В качестве устройств подключения могут выступать концентраторы или коммутаторы Fibre Channel (см. раздел 4.7.4). На данный момент достаточно указать, что устройства подключения играют важную роль в организации инфраструктуры кольца, в его работе и управлении им.

На рис. 4.2 показан пример кольца Fibre Channel с разделением доступа. Конфигурация аналогична физической звезде и логическому кольцу, используемым в локальных сетях на базе технологии Token Ring. Кроме того, как и в сетях Token Ring, данные перемещаются по кольцу в одном направлении. Но, в отличие от Token Ring, устройство может запросить право на передачу данных, а не ждать получения пустого маркера от коммутатора.

Команды Fibre Channel поддерживают согласование и доступ к кольцу для передачи данных. Кроме того, предоставляются команды для назначения адресов портов кольца с разделением доступа (arbitrated-loop port addresses — AL-PA) различным узлам кольца. Каждый узел в управляемом кольце Fibre Channel имеет контур для собственного отключения от кольца и сохранения непрерывности кольца в случае ошибки.

Кольца Fibre Channel с разделением доступа могут адресовать до 127 портов, в частности типа NL (дополнительная информация приводится в разделе 4.5), что обусловлено методом указания адреса AL-PA. Один из этих портов зарезервирован для подключения к коммутатору связной архитектуры (см. раздел 4.7.4.3). Остальные 126 портов доступны для предоставления узлам. Практика показывает, что типичные кольца с разделением доступа

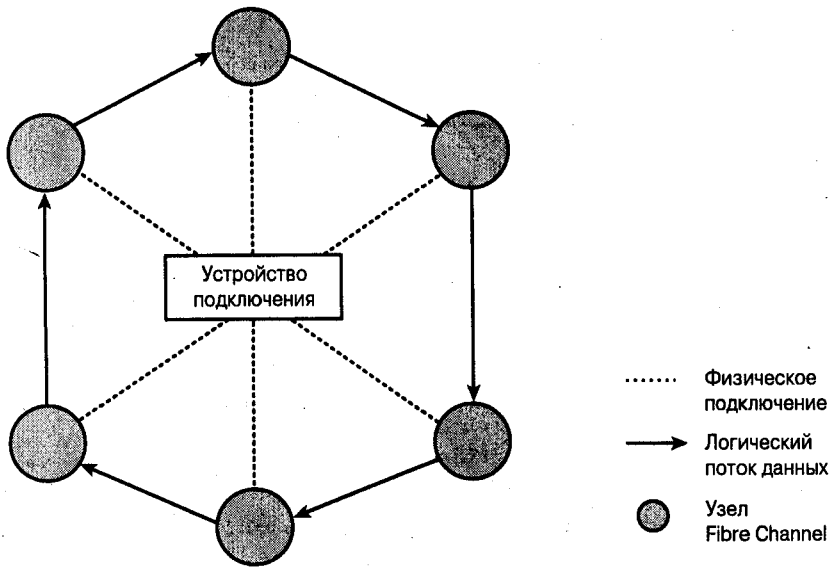


Рис. 4.2. Топология кольца Fibre Channel с разделением доступа

содержат до 12 узлов, а после подключения 50 узлов производительность падает до такой степени, что имеет смысл перейти на коммутатор связной архитектуры. Кольца с разделяемым доступом позволяют использовать все преимущества Fibre Channel при значительно меньших денежных затратах. Однако стоимость коммутаторов Fibre Channel стремительно снижается, поэтому применение коммутируемых связных архитектур становится все более предпочтительным.

В разделах 4.4.2.1–4.4.2.3 описываются важные концепции, связанные с кольцом Fibre Channel с разделяемым доступом: инициализация кольца, управление кольцом и различные типы колец (закрытые и открытые кольца).

4.4.2.1 Инициализация кольца Fibre Channel

Протокол инициализации кольца Fibre Channel обладает четко определенной структурой. Инициализация выполняется в таких случаях:

- при первой установке и запуске кольца;
- при подключении нового устройства;
- при отключении или перезапуске существующего устройства.

Кольца инициализируются с помощью специальных управляющих кадров. В процессе инициализации выполняются описанные ниже действия.

- На время до следующей инициализации выбирается хозяин кольца (loop master). Одной из наиболее важных задач хозяина кольца является назначение адресов различным портам в кольце.
- Выполняется назначение адресов в кольце. Значения адресов, присвоенных устройствам, используются для решения конфликтов, когда несколько устройств пытаются одновременно передавать данные. Коммутаторы связных архитектуры имеют наибольший приоритет; они могут выбрать любой произвольный адрес. Портам разрешается получать адрес, который использовался ранее.

4.4.2.2 Разделение доступа в кольце Fibre Channel

Если через один порт организуется передача данных на другой порт, сначала следует выяснить владельца кольца. Для этого порт отправляет специальный кадр, который называется `ARB primitive`. В этом кадре содержится адрес порта, который намерен получить управление над кольцом. При получении портом кадр `ARB primitive` передается дальше, если порт не собирается передавать данные. Если порт-получатель намерен сам передавать данные, он сравнивает адрес из `ARB primitive` с собственным адресом. Если у получателя адрес имеет меньшее значение (больший приоритет), то кадр отправляется дальше, но содержит адрес данного порта. В противном случае кадр `ARB primitive` отправляется дальше с неизменным адресом. Через некоторое время порт получает кадр `ARB primitive` со своим адресом обратно. Именно на этом этапе порт получает управление кольцом.

Порт, который выиграл управление кольцом, отправляет кадр `OPN primitive` тому порту, которому нужно передать данные. Этот кадр передается дальше промежуточными портами, пока не достигнет целевого порта. Порт отвечает другим кадром `ARB primitive`, после получения которого портом-инициатором может начинаться сеанс передачи данных. После завершения отправки данных порт-инициатор отправляет кадр `CLS primitive`. Однако целевой порт может продолжать отправку кадров для завершения сеанса передачи данных, а порт-инициатор должен быть готов принять эти кадры даже после отправки `CLS primitive`. После завершения отправки кадров целевой порт может ответить на кадр `CLS primitive` собственным кадром `CLS primitive`. На этом этапе кольцо готово к началу следующего сеанса передачи данных.

В стандарте Fibre Channel описан необязательный алгоритм предотвращения ситуации, когда устройства с низким приоритетом никогда не получают доступ к кольцу из-за устройств с высоким приоритетом. Этот необязательный алгоритм запрещает устройствам с высоким приоритетом получать

кольцо в управление сразу после завершения сеанса связи, пока возможность передачи данных не появится у устройств с низким приоритетом.

4.4.2.3 Открытые и закрытые кольца

Закрытое кольцо представляет собой изолированное кольцо Fibre Channel с разделением доступа (кольцо не подключено к связной архитектуре, которая рассматривается в разделе 4.4.3). *Открытое кольцо* — это кольцо с разделением доступа, подключенное к связной архитектуре через коммутатор. В данном случае обеспечение совместимости зависит непосредственно от коммутатора, как описано ниже.

- Коммутатор связной архитектуры должен обеспечивать уникальность адресов в пределах кольца и связной архитектуры, а также заменять трехбайтовые адреса в связной архитектуре на однобайтовые адреса в пределах кольца. Этот процесс обычно называется *подменой адресов*.
- От коммутатора связной архитектуры требуется, чтобы подключенные к ней устройства в пределах кольца с разделением доступа выглядели как подключенные к кольцу.
- Обнаружение устройств необходимо для поддержки передачи данных в рамках общей и частной транспортной инфраструктуры. Устройства Fibre Channel, поддерживающие связную архитектуру, самостоятельно регистрируются на сервере SNS — Simple Name Server (см. раздел 4.4.3.1), однако устройства Fibre Channel для кольца с разделением доступа не поддерживают такую возможность. На коммутатор возлагается задача по обнаружению каждого устройства в кольце Fibre Channel и добавлению характеристик устройства на SNS. Это необходимо для устройств Fibre Channel, поддерживающих работу в рамках связной архитектуры.

4.4.3 Коммутируемая связная архитектура

В топологии коммутируемой связной архитектуры Fibre Channel (Fibre Channel switched-fabric) каждое устройство имеет логическое подключение к любому другому устройству. Обратите внимание на слово *логическое*. Обеспечение физического подключения устройств по топологии "каждый с каждым" потребовало бы огромных затрат, так как для N устройств необходимо N^2 портов и физических подключений. В реальности каждое устройство подключается к коммутатору, а коммутатор поддерживает логические подключения между всеми своими портами.

На рис. 4.3 показан простейший вариант топологии коммутируемой связной архитектуры Fibre Channel. Несколько узлов (устройств хранения и компьютерных систем) подключены к коммутатору Fibre Channel. Коммутатор —

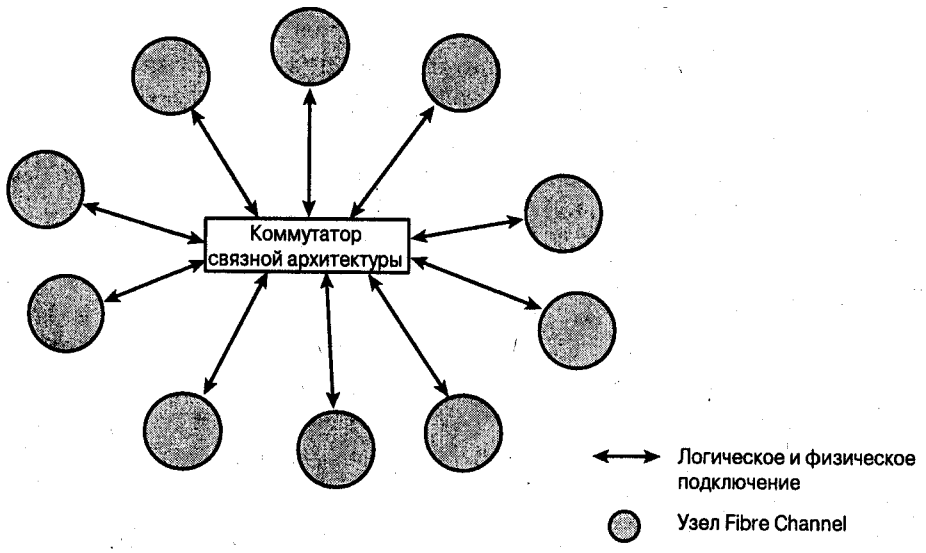


Рис. 4.3. Топология коммутируемой связной архитектуры Fibre Channel

это высокоскоростное устройство, которое обеспечивает подключение по схеме "каждый с каждым" и обрабатывает несколько одновременных подключений. Кроме того, коммутатор поддерживает такие службы, как Fabric Login (см. раздел 4.4.3.6). Коммутаторы могут быть подключены каскадно (в виде иерархии) или в виде сети, что позволяет формировать более сложные конфигурации. Очень часто строится трехуровневая иерархия, на самом нижнем уровне которой размещены кольца с разделением доступа Fibre Channel (FC-AL), подключенные к малопроизводительным коммутаторам. Эти коммутаторы, в свою очередь, подключаются к высокоскоростным коммутаторам, обеспечивающим максимально возможную пропускную способность.

Несколько коммутаторов могут быть соединены друг с другом. В топологии связной архитектуры используется 3-байтовый идентификатор (24 бит) для уникальной идентификации каждого устройства, что *теоретически* допускает подключение по схеме "точка–точка" для 2^{24} устройств (примерно 15 млн.). Конечно, на практике сети хранения данных включают в себя намного меньше устройств.

Поскольку количество потенциальных устройств может быть достаточно большим, от сетей хранения данных на базе Fibre Channel требуется предоставление служб по управлению адресами, а также дополнительных служб; в противном случае ручное управление сетями хранения данных было бы невозможным. Учитывая, что обеспечение взаимодействия Fibre Channel и протоколов верхнего уровня (например, SCSI) проводится самими узлами, а ад-

ресное пространство весьма велико, динамическое управление адресами на инфраструктурном уровне будет обязательным (не требуя ручной настройки).

Топология связной архитектуры реализуется на основе коммутатора связной архитектуры (см. раздел 4.4.3.5). Кроме предоставления подключений "точка–точка", коммутатор предоставляет такие службы, как SNS (Simple Name Server), RSCN (Registered State Change Notification), FAN (Fabric Address Notification), Broadcast Server IP-FC, Principal Switch и Fabric Login. Эти службы кратко рассматриваются в разделах 4.4.3.1–4.4.3.6. Обратите внимание, что службы основаны на модели клиент/сервер. Серверный компонент реализован в виде коммутатора связной архитектуры, а клиентский компонент представляет собой такие устройства, как адаптеры шины и контроллеры RAID.

4.4.3.1 Сервер SNS

Этот сервер представляет собой базу данных коммутатора связной архитектуры. База данных отслеживает два типа информации.

1. Имя устройства и информация об адресе.
2. Поддерживаемые протоколы высокого уровня.

Сервер SNS позволяет инициатору сеанса связи, например адаптеру шины сервера, делать запрос к службе имен и получать список доступных устройств хранения данных. Это обеспечивает автоматическое управление адресами, что имеет особое значение, поскольку адресное пространство сетей хранения данных поддерживает до 15 млн. адресов.

При инициализации устройство связной архитектуры отправляет базовый пакет регистрации (в терминологии Fibre Channel он называется *FLOGI*) на известный адрес 0xFFFFFE. Запрос на регистрацию вместо адреса заполняется нулевым значением, что обозначает запрос адреса.

Как только устройство подключается к коммутатору, инициируется другой базовый запрос (*PLOGI*), отправляемый серверу SNS, который имеет фиксированный адрес 0xFFFFFC. Устройство отправляет соответствующую информацию, например адрес в формате WWN (World Wide Name), тип порта, адрес коммутатора, протоколы верхнего уровня (SCSI), которые поддерживаются устройством, и другие служебные параметры.

Некоторые устройства не отправляют SNS всю информацию о себе. Отдельные модели коммутаторов отправляют устройствам запросы целевого определения и обновляют значения соответствующих полей в базе данных SNS.

4.4.3.2 Служба RSCN

Служба RSCN (Registered State Change Notification) позволяет устройству регистрироваться для получения уведомлений о подключении к коммутатору или отключении от него других устройств. Например, сервер может зарегистрироваться для получения уведомлений при подключении к коммутатору или отключении от него связной архитектуры устройств хранения. Таким образом, RSCN дополняет сервер SNS: сервер позволяет компоненту Fibre Channel (например, узлу) обнаруживать другие компоненты (например, единицы хранения). В то же время служба RSCN позволяет узлу (компоненту Fibre Channel) регистрироваться для получения уведомлений об изменениях в составе компонентов Fibre Channel.

Пакеты RSCN могут прерывать передачу данных. Кадры RSCN используются получателем для обновления собственных таблиц маршрутизации. Некоторые поставщики предоставляют возможности по остановке широковещательного распространения пакетов RSCN на уровне коммутатора. Хотя это полезная функция, существует опасность, что из-за неправильной настройки кадр RSCN никогда не достигнет устройства, которое должно его получить.

4.4.3.3 Служба FAN

Служба FAN (Fabric Address Notification) обеспечивает ликвидность транзакций, выполняемых устройствами. Обычно это необходимо в том случае, когда инициализация кольца прерывает активную передачу данных между двумя компонентами Fibre Channel.

4.4.3.4 Служба Broadcast Server

Интерфейс Fibre Channel поддерживает протоколы более высокого уровня, требования для реализации которых должны быть выполнены. К одному из таких протоколов относится IP. Так как в рамках сетей IP поддерживается вторичный протокол ARP (Address Resolution Protocol), посредством которого устройства отправляют широковещательные запросы, этот же протокол должен поддерживаться и устройствами Fibre Channel.

4.4.3.5 Служба Principal Switch

При использовании в сети хранения данных на основе Fibre Channel нескольких коммутаторов связной архитектуры необходим протокол выбора одного из коммутаторов в качестве главного коммутатора (principal switch). Выбранный главный коммутатор будет работать как коммутатор с адресом

0xFFFFFE и обеспечивать уникальность адресов устройств в сети хранения данных на основе Fibre Channel.

4.4.3.6 Служба Fabric Login

Устройства связной архитектуры всегда стремятся воспользоваться службой Fabric Login. Для этого на фиксированный адрес 0xFFFFFE отправляется специальный кадр *FLOGI*. Кадр FLOGI содержит адрес отправляющего устройства, который устанавливается в нулевое значение, если пакет используется для запроса нового адреса.

Служба Fabric Login позволяет выполнять следующие задачи:

- получение информации о параметрах связной архитектуры, например списков поддерживаемых служб;
- назначение адресов;
- инициализация буфера для контроля передачи данных.

4.5 Типы портов Fibre Channel

В стандарте Fibre Channel определено несколько типов портов, зависящих от топологии сети хранения данных и от устройства, к которому относится порт. Различные типы портов представлены в табл. 4.2.

Таблица 4.2. Типы портов Fibre Channel

Тип порта	Устройство, поддерживающее порт	Описание	Типы портов, которые можно подключать к этому порту
F	Коммутатор связной архитектуры	Может работать только в рамках сети на базе связной архитектуры. Никогда не является источником или точкой назначения данных. Работает как посредник для обеспечения связи между устройствами	N
FL	Коммутатор связной архитектуры	Может работать только в кольце. Обеспечивает подключение кольца Fibre Channel к сети на базе связной архитектуры. Никогда не является источником или точкой назначения данных	NL

Окончание табл. 4.2

Тип порта	Устройство, поддерживающее порт	Описание	Типы портов, которые можно подключать к этому порту
E	Коммутатор связной архитектуры	Порт расширения, который используется для подключения коммутаторов связной архитектуры друг к другу. Никогда не является источником или точкой назначения данных. Выполняет роль посредника	E
N	Адаптер шины или устройство хранения	Не поддерживает работу в кольце. *Всегда* представляет собой источник или точку назначения данных. Реализован в сетях "точка–точка" или в сетях на базе связной архитектуры	F или N
NL	Адаптер шины или устройство хранения	Аналогичен типу N, однако поддерживает работу в кольце. *Всегда* представляет собой источник или точку назначения данных	NL или FL
U		Универсальный неиспользуемый порт. После активизации может выступать в роли портов E, F или FL	Нет
L		Универсальный термин для описания портов NL или FL	
G		Может выступать в роли портов E или F	E или N

Чтобы облегчить жизнь администраторам сетей хранения данных, поставщики устройств обычно предоставляют порты с автоматическим конфигурированием. Такое конфигурирование проводится следующим образом.

1. Порт пытается инициироваться как порт кольца (FL). Если инициализация проходит успешно, порт настраивается как порт FL.

2. Если инициализация кольца завершается неудачно, делается попытка инициализации в виде порта E; таким образом, проверяется возможность подключения к сети на базе связной архитектуры.
3. Если инициализация в виде порта E завершается неудачно, порт пытается активизировать службу Fabric Login. Если регистрация выполняется успешно, порт инициируется как порт F.

Не стоит рассчитывать, что все поставщики устройств устанавливают на них динамически конфигурируемые порты, которые могут работать вместо любого порта. Такая возможность встроена лишь в некоторые устройства. Это позволяет переконфигурировать порты без перезагрузки или отключения питания. Остальные устройства для получения определенного типа порта требуют замены прошивки или установки платы расширения, что может потребовать выключения питания.

4.6 Протокол Fibre Channel

В общем контексте *Fibre Channel* — это набор стандартов, разработанных в Национальном институте стандартизации США. Интерфейс Fibre Channel предоставляет высокопроизводительное последовательное подключение между хостом и единицами хранения, а также между самими единицами хранения. Стандарт позволяет обеспечить высокоскоростную передачу данных в сетях с топологией "точка–точка" и кольцо. Более того, Fibre Channel предоставляет все эти возможности вместе с проверкой ошибок.

В стандарте Fibre Channel определено пять функциональных уровней: от FC-0 до FC-4. Уровни рассматриваются в разделах 4.6.1–4.6.5. Обратите внимание, что по практическим соображениям уровни FC-0, FC-1 и FC-2 реализуются аппаратно.

4.6.1 Уровень FC-0

Определяет физические характеристики интерфейса и носителя. В частности, посредством FC-0 определяются спецификации уровней сигналов, носителя и получателей/отправителей. Уровень FC-0 позволяет использовать несколько интерфейсов, что дает возможность выбирать разные скорости передачи данных и различные передающие среды. В качестве примера физической передающей среды можно привести медный провод, одномодовый и многомодовый кабели. Скорость передачи варьируется от 12,5 до 106,25 Мбайт/с.

Те, кто знаком с семиуровневой сетевой моделью ISO OSI[2], могут заметить, что FC-0 соответствует седьмому уровню модели ISO OSI.

4.6.2 Уровень FC-1

Определяет схемы кодирования и декодирования данных, сигналов и специальных символов, а также управление ошибками. Кроме того, уровень FC-1 отвечает за обслуживание линий связи.

Уровень FC-1 использует схему кодирования, которая называется *8B/10B*. Схема проектировалась для того, чтобы обеспечить следующее:

- эффективную синхронизацию данных;
- расширенное обнаружение ошибок;
- эффективное обнаружение управляющих символов;
- упрощенное проектирование аппаратного обеспечения приемников/передатчиков.

Схема кодирования 8B/10B преобразует каждые 8 бит в два возможных значения, объемом 10 бит. Эти 10 бит используются в виде `Ann.m`, где `A` — значение *K* для индикации команды или *D* для индикации данных; `nn` — десятичное значения последних пяти битов байта; `M` — десятичное значение первых трех битов байта.

Два возможных значения появляются потому, что посредством спецификации выбирается одно из значений для кодирования данных при их передаче на базе недавней истории передачи. Это необходимо для того, чтобы обеспечить минимальное количество переходов состояния (между 0 и 1), что повысит эффективность передачи. Недавняя история передачи называется *динамическим рассогласованием*.

Как уже отмечалось, все данные кодируются с помощью 10 бит. Некоторые неиспользованные 10-битовые символы (в контексте данных) применяются для отделения фреймов и сигналов, включая сигналы о готовности порта для принятия данных, а также другие типы сигналов. Основное внимание уделяется обнаружению и исправлению ошибок на этапе передачи. Данные Fibre Channel всегда передаются группами по 4 байта, которые называются *словами передачи* (transmission words).

4.6.3 Уровень FC-2

Определяет передачу данных от одного узла к другому, т.е. непосредственно транспортный механизм. Уровень FC-2 формирует кадры, опреде-

[2]ISO — International Organization for Standardization (Международная организация по стандартизации);
OSI — Open System Interconnection (взаимодействие открытых систем).

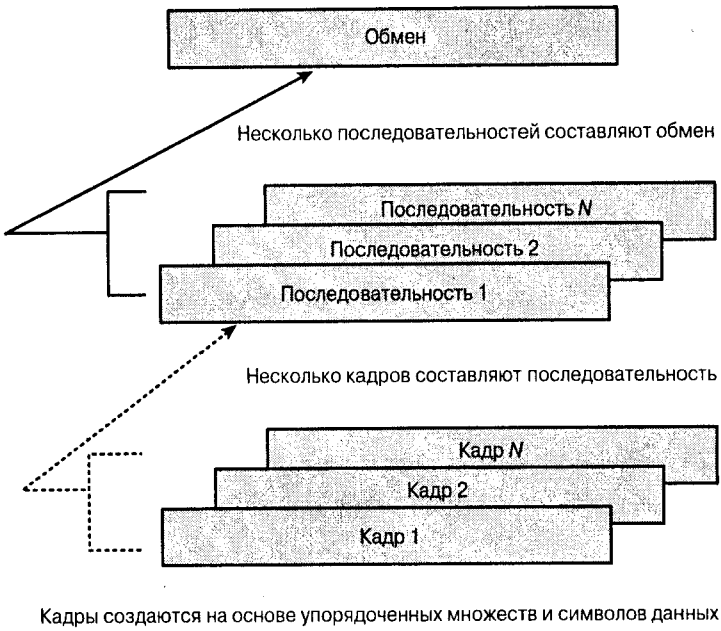


Рис. 4.4. Иерархия передачи данных Fibre Channel

ляет классы обслуживания и службы регистрации связной архитектуры или портов. Этот уровень можно представить в качестве аналога уровню MAC (Media Access Control) в модели ISO OSI.

Уровень FC-2 определяет:

- иерархию передачи данных Fibre Channel, которая включает в себя упорядоченные множества, кадры, последовательности и обмены;
- управление потоком Fibre Channel;
- протоколы FC-2;
- классы обслуживания FC-2.

Эти компоненты рассматриваются в разделах 4.6.3.1–4.6.3.7, но перед этим следует выяснить, как соединяются различные структурные элементы иерархии.

В Fibre Channel данные передаются с помощью *кадров* (frame). Кадр представляет собой эквивалент пакета TCP/IP. Кадры создаются из *упорядоченных множеств* и символов данных. Несколько кадров группируются вместе и формируют *последовательность*, а несколько последовательностей формируют *обмен* (exchange). Это демонстрируется на рис. 4.4.

В разделах 4.6.3.1–4.6.3.4 все эти компоненты рассматриваются более подробно.

4.6.3.1 Упорядоченные множества Fibre Channel

Упорядоченные множества — это структуры последовательной передачи данных объемом 4 байта, которые представляют собой специальные символы или сигналы линии связи. Далее приведены примеры подобных множеств.

- Разделители кадров SOF (Start Of Frame) и EOF (End Of Frame), которые являются аналогами пакетов SOF и EOF в сетях Ethernet. В отличие от Ethernet, в Fibre Channel определено несколько вариантов SOF и EOF, поскольку уровнем FC-1 используется схема кодирования, формирующая несколько представлений для каждого передаваемого символа.
- Два базовых сигнала для указания состояния порта.
  - *Idle* — указание, что порт готов для передачи или приема данных.
  - *Receiver Ready* — указание, что буфер интерфейса (устройства взаимодействия) готов для приема данных.
- Базовая последовательность. Простое упорядоченное множество, которое регулярно передается для указания особого статуса порта. К особым статусам относятся:
  - *Not Operational* (NOS) — используется только в сетях с топологией "точка-точка" или в связной архитектуре (но не в кольце с разделением доступа) для указания на отказ в работе линии связи или появление определенной ошибки;
  - *Offline* (OLS) — передается во время инициализации порта или при получении базового статуса NOS; таким образом, в ответ на NOS порт отправляет ответ OLS;
  - *Link Reset* (LR) — используется для указания на необходимость повторной инициализации линии связи;
  - *Link Reset Response* (LS) — используется для указания, что данные LR получены и обработаны.

4.6.3.2 Кадр Fibre Channel

Как пакет IP является базовым элементом протокола Internet (IP), так и кадр представляет собой основной структурный элемент интерфейса Fibre Channel. Существует три типа кадров.

1. *Кадры управления линией связи* (link control frames), используемые для отправки команды управления линией связи.

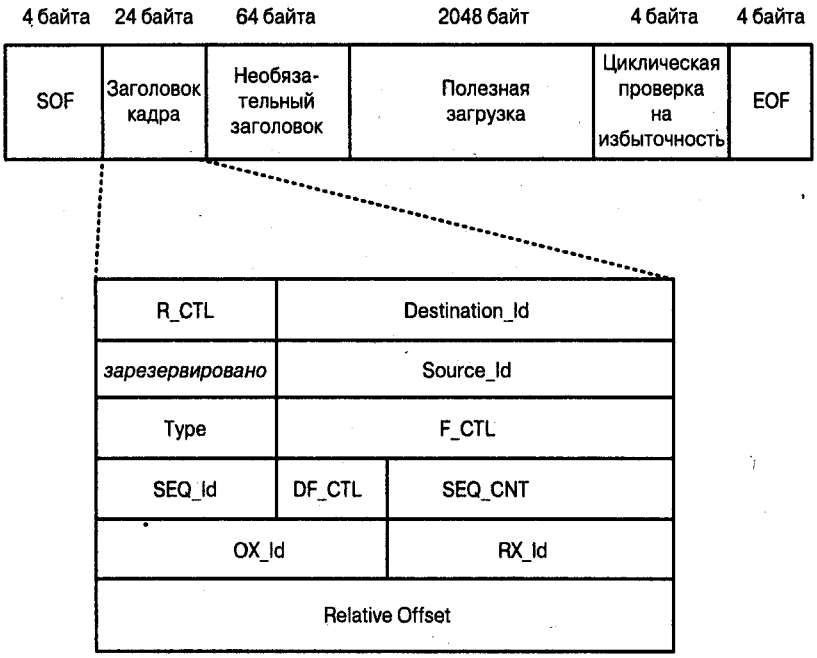


Рис. 4.5. Заголовок кадра Fibre Channel

2. *Кадры данных линии связи* (link data frames), используемые для отправки данных, необходимых для управления линией связи.
3. *Кадры данных устройства* (device data frames), которые содержат данные для протоколов более высокого уровня, например данные, считанные с жесткого диска.

На рис. 4.5 показан заголовок кадра Fibre Channel. Кадр проектировался для передачи 2048 байт данных и необязательного заголовка размером 64 байт. Подобный размер кадра позволяет передавать за один раз большой объем данных с минимальными накладными расходами (около 1,5%). Однако при этом другому узлу придется ждать, пока завершится передача большого кадра, что подразумевает увеличение задержек в передаче. Сравним это с протоколом ATM (Asynchronous Transfer Mode), где кадр имеет размер 53 байт и накладные расходы протокола составляют около 10%. Это позволяет снизить задержки, но время передачи определенного объема данных возрастает.

Каждый кадр начинается и заканчивается специальным ограничителем, как и в других сетевых протоколах. Это SOF и EOF соответственно. Каждый кадр имеет заголовок, выполняющий несколько функций. Одна из них — это предоставление адресов назначения и источника для обеспечения коммута-

ции данных. Еще одна задача состоит в переносе информации для управления линией связи, включая управление самой передачей.

Другие поля заголовка кадра Fibre Channel рассматриваются ниже.

- Поле `Destination_Id` используется для маршрутизации кадра. В топологиях "точка–точка" и кольцо с разделением доступа маршрутизация может проходить обычным образом, что не относится к топологии коммутируемой связной архитектуры. Поле `Source_Id` предназначено для передачи сообщений об ошибках и для предотвращения циклов данных при маршрутизации.
- Поля `R_CTL` и `Type` используются для сортировки различных кадров уровня FC-4 по их прибытию в точку назначения. Таким образом, эти поля указывают, содержит ли прибывший кадр данные интерфейса SCSI, протокола IP или другие данные. Значения поля `Type` описываются в табл. 4.3.
- Поле `R_CTL` используется для указания содержимого кадра. Кадр может содержать данные или информацию для управления линией связи; в последнем случае кадры могут быть запрошенными и незапрошенными.

Таблица 4.3. Значения поля `Type` в кадрах Fibre Channel

Значение	Описание
0x00	Базовое обслуживание линии связи
0x01	Порт Fibre Channel — расширенное обслуживание линии связи
0x02	Порт памяти
0x03	Инициализация памяти
0x04	Интерфейс IEEE 802.2
0x05	Протокол Internet (IP)
0x06	Подчиненное устройство IPI (Intelligent Peripheral Interface)
0x07	Основное устройство IPI
0x08	Инициатор SCSI
0x09	Целевое устройство SCSI
0x0A	Интерфейс HIPPI
0x0B	Код SBCCS (Single-Byte Command Code Sets)
0x0C	Зарезервировано для новых типов шины
0x10	Зарезервировано
0xD0	Уникально для производителя
0xF0	Прямой канал

- Поле F_CTL используется для описания информации кадра, например первой или последней последовательности.
- Поле DF_CTL указывает на присутствие или отсутствие необязательных заголовков.
- Поля SEQ_Id и SEQ_CNT уникально идентифицируют счетчик последовательности обмена (см. раздел 4.6.3.3).
- Поле OX_Id (идентификатор обмена источника) используется для связывания кадра с определенным обменом исходного порта.
- Поле RX_Id (идентификатор обмена ответчика) используется для связывания кадра с определенным обменом отвечающего порта.
- Поле Relative Offset идентифицирует относительное смещение первого байта основного содержания кадра от базового адреса.

4.6.3.3 Последовательность Fibre Channel

Последовательность представляет собой набор кадров, которые передаются из одной точки в другую. Для исправления возможных ошибок каждый кадр содержит уникальный счетчик последовательности. Исправление ошибок осуществляется протоколом более высокого уровня, обычно на уровне FC-4. Обратите внимание, что все кадры в последовательности передаются в одном направлении (а не в обоих одновременно). На рис. 4.4 показано, как кадры, последовательности и обмены взаимодействуют друг с другом.

4.6.3.4 Обмен Fibre Channel

Несколько последовательностей составляют *обмен* (exchange). Обмены представляют собой последовательности двусторонних направлений; т.е. в обмен входят последовательности данных, передающихся в разных направлениях, хотя каждая последовательность передается только в одном направлении. При каждом обмене только одна последовательность может быть активна в текущий момент времени. Но, так как одновременно могут быть активны несколько обменов, различные последовательности из этих обменов также могут быть активны одновременно.

Каждый обмен выполняет одну функцию, например реализует команду SCSI Read.

4.6.3.5 Управление потоком Fibre Channel

Конечные узлы Fibre Channel взаимодействуют непосредственно друг с другом и не создают сеансовых подключений к промежуточным узлам.

Устройства не "подозревают" о коммутаторе связной архитектуры и концентраторе. Конечно, при этом коммутаторы и концентраторы обмениваются пакетами управления потоком с этими устройствами.

Механизм управления потоком требует, чтобы передающий порт не отправлял кадры быстрее, чем принимающий порт может их обработать. Порты Fibre Channel имеют буфера для временного хранения кадров и последующей их обработки. Под обработкой подразумевается отправка кадра на другой порт или передача кадра протоколу более высокого уровня. Схема управления потоком, которая используется в Fibre Channel, очень напоминает протокол плавающего окна в TCP/IP. Размер окна, представляющий собой количество кадров, которые можно отправить без подтверждения их приема, устанавливается сторонами обмена заранее. При этом согласованное значение не может быть изменено. Для каждого отправленного кадра размер окна уменьшается на единицу, а для каждого подтвержденного кадра — увеличивается на единицу. Управление потоком может осуществляться одним из двух способов: "точка–точка" или "буфер–буфер". При этом требуется совместное использование обоих методов.

Управление потоком по схеме "точка–точка" осуществляется между двумя конечными точками: источником данных (например, сервером) и получателем данных (например, жестким диском). Управление потоком "точка–точка" проводится между двумя портами типа N (между ними могут находиться промежуточные узлы). Два порта типа N регистрируют друг друга, в процессе чего каждый порт выделяет для себя определенное количество буферов у другого порта. Это количество называется *резервированием буфера*. Отправитель может отправлять количество кадров, не превышающее это значение. Получатель отправляет кадр ACK (позитивное подтверждение) для каждого успешно полученного и обработанного кадра, а отправитель при получении кадра ACK может увеличить значение резервирования (credit count) на единицу для каждого полученного кадра ACK. Получатель может подтвердить успешное получение нескольких кадров или даже целой последовательности и получатель должен будет, не ожидая подтверждения для каждого кадра в отдельности, увеличить количество кадров, которое можно отправлять.

Управление потоком по схеме "буфер–буфер" выполняется между двумя соседними узлами, которые представляют собой промежуточные узлы или находятся между конечным и промежуточным узлом. Таким образом, управление потоком от буфера к буферу выполняется между портами типа N или между портом F и портом N. Как уже отмечалось, порты обмениваются данными, указывающими на количество буферов, зарезервированных для каждого узла. Эти значения могут отличаться, например один порт может выделить два буфера, а второй — четыре буфера. Получение кадра подтвер-

ждается кадром `Receiver Ready`, а не кадром `ACK` как в управлении потоком "точка–точка".

4.6.3.6 Протоколы FC-2

В стандартах Fibre Channel определены протоколы для управления передачей данных и линией связи. Кроме того, описаны дополнительные стандарты для поддержки протоколов более высокого уровня, применяемых на уровне FC-4. Эти протоколы описаны ниже.

- Протокол *Fabric Login*, который определяет обмен параметрами между портом и коммутатором связной архитектуры. Протокол и служба Fabric Login более подробно описываются в разделе 4.4.3.6.
- Протокол *Port Login*, требующий, чтобы независимо от топологии ("точка–точка", кольцо с разделением доступа или коммутируемая связная архитектура) два порта проводили взаимную регистрацию перед подключением друг к другу. Взаимная регистрация выполняется с помощью специального кадра `PLOGI`. С помощью протокола Port Login обеспечивается использование двух важных функций.
  - Возможность получения информации о порте N, на котором выполняется регистрация. К такой информации относится описание классов обслуживания, поддерживаемых портом N.
  - Инициализация буфера резервирования для управления потоком "точка–точка". Обратите внимание, что в контексте прямого подключения управление потоком "точка–точка" ничем не отличается от управления потоком "буфер–буфер".
- Протокол *Data Transfer*, определяющий, как данные протокола верхнего уровня (уровня FC-4) передаются с помощью схем управления потоком, описанных в разделе 4.6.3.5.
- Протокол *Arbitrated Loop*, который определяет методы инициализации и управления кольцом.

4.6.3.7 Классы обслуживания FC-2

Интерфейс Fibre Channel проектировался для обеспечения различных способов передачи данных. Ряд служб отличается такими характеристиками:

- тип сервисного подключения, т.е. аналогично TCP или без установки подключения, как в UDP;
- поддержка многоабонентской доставки (multicast);
- поддержка уведомления о доставке или невыполненной доставке;

- поддержка гарантированной доставки кадров в том же порядке, в котором они были отправлены;
- тип предоставляемых служб, например резервирование пропускной способности для соединения, если служба ориентирована на соединение;
- тип механизмов управления потоком данных.

Для предоставления широкого диапазона вариантов передачи данных, определено неколько классов обслуживания.

- Тип *Class 1* определяет выделенное подключение, подобное подключению TCP/IP. Как и в TCP, Class 1 гарантирует, что кадры доставляются в той же последовательности, в которой они были отправлены. Тип Class 1 используется при передаче больших объемов данных, когда время, потраченное на установку соединения, на порядок меньше времени, необходимого для передачи данных.
- Тип *Class 2* определяет обслуживание без подключения (по аналогии с дейтаграммами), при котором кадры потенциально могут быть доставлены не в той последовательности, в которой они отправлялись (что подразумевает смену последовательности кадров протоколом более высокого уровня). Как и в случае использования сетевых протоколов, обслуживание Class 2 имеет смысл тогда, когда объем передаваемых данных достаточно мал и накладные расходы на установку соединения сравнимы с расходами на передачу самих данных. Получатель кадра Class 2 должен отправить подтверждение при получении кадра.
- Тип *Class 3* также подразумевает обслуживание без установки подключения. Основное отличие от Class 2 состоит в том, что подтверждать успешное получение кадра нет необходимости. Это сравнимо с дейтаграммами IP, метод использования которых иногда в шутку называется "отправь и молись".
- Тип *Class 4*, который также называется *Intermix*, является необязательным классом обслуживания. Класс гарантирует определенную пропускную способность кадрам Class 1, а оставшаяся пропускная способность используется для кадров Class 2 и Class 3.
- Тип *Class 6* представляет собой однонаправленное, ориентированное на подключение обслуживание с предоставлением возможности многоабонентской доставки (Class 5 зарезервирован).

В табл. 4.4 собрана вся информация о классах обслуживания Fibre Channel.

Обратите внимание, что большинство поставщиков поддерживают классы 1, 2 и 3. В то же время некоторые поставщики поддерживают только классы, не ориентированные на соединение (Class 2 и Class 3).

Таблица 4.4. Классы обслуживания Fibre Channel

	Class 1	Class 2	Class 3	Class 4	Class 6
Тип обслуживания	Ориентирован на подключение	Не поддерживает подключение	Не поддерживает подключение	Ориентирован на подключение	Ориентирован на подключение
Поддержка многоабонентской доставки	Нет	Нет	Да	Нет	Да
Подтверждение доставки	Нет	Нет	Да	Нет	Да
Использование пропускной способности	Полное	Не гарантируется	Не гарантируется	Частичное	Не гарантируется
Управление потоком	Точка–Точка	Точка–Точка	Буфер–Буфер	Буфер–Буфер	Точка–Точка
Доставка кадров	Гарантируется	Порядок может нарушаться	Порядок может нарушаться	Гарантируется	Гарантируется

4.6.4 Уровень FC-3

Определяет общие службы, включая службы по управлению и общему транспортному механизму. Уровень FC-3 — общий для всех портов узла. Уровни FC-1, FC-2 и FC-4 реализованы отдельно для каждого порта. В этой схеме поддерживается использование разными портами различной конфигурации (рис. 4.6). Например, один порт может передавать данные SCSI, а другой в это же время будет передавать данные ATM.

Кроме того, обратите внимание, что уровни FC-0, FC-1 и FC-2 относятся к различным портам, а уровень FC-3 относится к узлу. Верхний уровень FC-4 также относится к порту. На рис. 4.6 показан узел с четырьмя портами и двумя протоколами верхнего уровня — SCSI и IP. Если бы узел поддерживал больше протоколов верхнего уровня, на диаграмме присутствовали бы дополнительные блоки FC-4.

Далее представлены некоторые функции, реализованные на уровне FC-3.

- *Транкинг* (trunking) или *канальное уплотнение* (striping), при котором параллельные линии связи и порты "сворачиваются" для обеспечения большей пропускной способности между узлами.
- *Многоабонентская доставка*, при которой единая передача данных может быть направлена одновременно к нескольким портам. Реализуется посредством службы регистрации на коммутаторе связной архитектуры, с помощью которой узлы регистрируются и отменяют регистрацию для многоабонентской доставки. Доставка в этом случае очень похожа на режим групповой отправки (multicast) в IP. Обратите внимание: многоабонентская доставка может осуществляться на все порты коммутато-

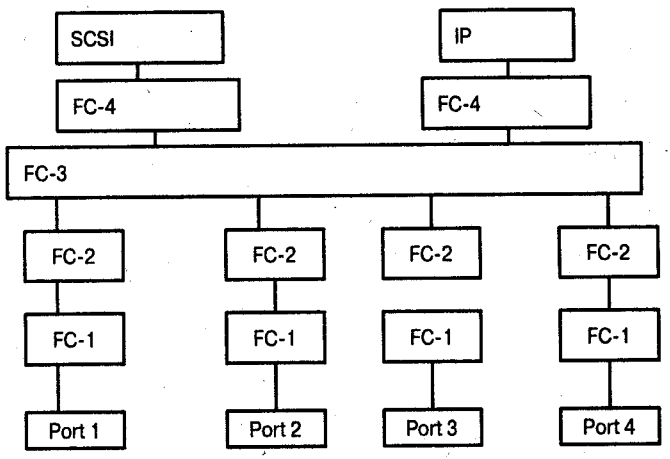


Рис. 4.6. Уровни Fibre Channel

ра связной архитектуры, т.е. может использоваться как широковещание (broadcast).

- *Свободный поиск* (hunting), при котором несколько портов используют одинаковый псевдоним. Это похоже на объединение нескольких телефонных линий компании под одним номером, когда звонок на этот номер перенаправляется на одну из свободных линий. Очевидным преимуществом такой схемы будет повышение вероятности получить свободный порт.

Компания Brocade одна из многих реализует функцию транкинга в своих продуктах, хотя функция получила названия ISL (Inter Switch Link) Trunking. Называние подчеркивает высокую производительность коммутаторов Brocade, которые предназначены для подключения к другим коммутаторам, а не к серверам и устройствам хранения данных. Обратите внимание, что транкинг поддерживается и другими поставщиками.

4.6.5 Уровень FC-4

Определяет связывание протоколов верхнего уровня с Fibre Channel. Вот эти протоколы (со временем будет реализована поддержка и других протоколов):

- SCSI;
- IP;
- IPI (Intelligent Peripheral Interface);
- HIPPI (High-Performance Parallel Interface);
- IEEE 802.2;
- SBCCS (Single-Byte Command Code Sets);
- AAL5 (ATM Adaptation Layer);
- FC-LE (Link Encapsulation).

Обратите внимание: одна линия связи Fibre Channel может одновременно передавать пакета данных нескольких протоколов верхнего уровня.

4.7 Структурные элементы SAN

В разделах 4.4 и 4.6 приведен обзор топологий и протокола Fibre Channel. Теперь рассмотрим различные устройства и компоненты, которые используются для создания сетей хранения данных Fibre Channel. К основным структурным элементам SAN относятся:

- адаптеры шины;
- кабели Fibre Channel;
- разъемы;
- устройства подключения, в число которых входят концентраторы, коммутаторы и коммутаторы связной архитектуры.

Все эти элементы рассматриваются в разделах 4.7.1–4.7.4. Обратите внимание, что все адресуемые компоненты в пределах сети хранения данных на основе Fibre Channel имеют уникальные имена WWN (World Wide Names), которые представляют собой аналоги уникальных адресов MAC. Имя WWN в спецификации Fibre Channel — это 64-разрядное число, записываемое в виде `XX:XX:XX:XX:XX:XX:XX:XX`. Институт IEEE назначает каждому производителю определенный диапазон адресов. Производитель отвечает за уникальное выделение назначенных адресов.

4.7.1 Адаптеры шины

Адаптер шины (host bus adapter — HBA) подключается к компьютеру и обеспечивает взаимодействие с устройствами хранения данных. В мире персональных компьютеров под управлением Windows адаптеры шины обычно подключаются к шине PCI и могут предоставлять подключение для устройств IDE, SCSI и Fibre Channel. Адаптеры шины работают под управлением драйвера устройства, т.е. драйвера мини-порта SCSIPort или Storport.

При инициализации порт адаптера шины регистрируется на коммутаторе связной архитектуры (если таковой доступен) и регистрирует хранящиеся на нем атрибуты. Атрибуты доступны приложениям с помощью API от производителя коммутатора или адаптера шины. Ассоциация SNIA (Storage Networking Industry Association) работает над стандартизированным API, поддерживающим различные API производителей.

Для сети хранения данных, к которой выдвигаются серьезные требования по отказоустойчивости, некоторые производители адаптеров шины предоставляют дополнительные возможности, например автоматическое переключение на другой адаптер шины при отказе в работе основного. Эти функции, а также дополнительная архитектура рассматриваются в главе 9.

В кольце с разделением доступа только два устройства могут одновременно осуществлять прием и передачу данных. Предположим, что одно из них — это адаптер шины, подключенный к узлу и получающий данные от устройства хранения. Но, если этот адаптер подключен к сети хранения данных на основе коммутируемой связной архитектуры, он может одновременно отправлять несколько запросов на чтение нескольким устройствам хранения.

Ответы на эти запросы могут приходить в любом порядке. Обычно коммутатор связной архитектуры предоставляет службу циклического обслуживания для портов, что еще более усложняет задачу адаптера шины; в этом случае порядок поступления пакетов окажется таким, что каждый следующий пакет будет приходить от другого источника.

Адаптеры шины позволяют решить эту проблему одним из двух способов. Первая стратегия, которая называется *сохранить и отсортировать*, подразумевает хранение данных в памяти узла с последующей сортировкой буферов за счет центрального процессора. Очевидно, что это неэффективный подход с точки зрения центрального процессора и общая нагрузка связана с переключением контекста каждые несколько десятков микросекунд. Другая стратегия — *на лету* — подразумевает использование дополнительной системной логики и микросхем на самом адаптере шины, что позволяет осуществлять переключение контекста без использования циклов центрального процессора. Обычно время между переключениями контекста при использовании такой стратегии составляет несколько секунд.

Как отмечается в разделе 4.6.3.5, понятие *резервирование буфера* определено, как часть стандарта Fibre Channel. Одно резервирование позволяет отправить один кадр Fibre Channel. Перед отправкой следующего кадра отправитель должен получить сигнал `Receiver Ready`. Для эффективного использования канала Fibre Channel необходимо одновременно передавать несколько кадров, что потребует несколько резервирований, следовательно, понадобится больший объем памяти для принятия кадров. Некоторые адаптеры шины имеют четыре буфера размером 1 Кбайт и два буфера по 2 Кбайт, хотя на некоторых высокоуровневых адаптерах устанавливается 128 и 256 Кбайт для резервирования буфера. Обратите внимание, что для этой памяти обычно требуется два порта; т.е. когда одна область памяти принимает данные от SAN Fibre Channel, остальные области памяти могут передавать данные шине PCI узла.

Кроме того, адаптеры шины играют роль в обеспечении отказоустойчивости и в архитектуре с аварийным восстановлением данных, в которой предоставляется несколько маршрутов ввода-вывода к одному устройству хранения данных. Эти технологии рассматриваются в главе 9.

4.7.1.1 Операционная система Windows и адаптеры шины

В Windows NT и Windows 2000 адаптеры Fibre Channel рассматриваются как устройства SCSI, а драйверы создаются в виде драйверов мини-портов SCSI. Как отмечается в главе 2, проблема состоит в том, что драйвер SCSIPort устарел и не поддерживает возможности, предоставляемые новыми устройствами SCSI, не говоря уже об устройствах Fibre Channel. Поэтому в Windows

Server 2003 была введена новая модель драйвера Storport, которая должна заменить модель SCSIPort, особенно для устройств SCSI-3 и Fibre Channel. Обратите внимание, что диски Fibre Channel используются Windows в качестве DAS-устройств, что обеспечивается уровнем абстракции, предоставляемым драйверами SCSIPort и Storport.

4.7.1.2 Двойные маршруты

Иногда необходима повышенная производительность и надежность, даже за счет увеличения стоимости готового решения. В таких случаях сервер подключается к двухпортовым дискам через несколько адаптеров шины и несколько независимых сетей хранения данных Fibre Channel. Основная идея — исключить единую точку отказа в работе сети. Кроме того, в те моменты, когда система работает нормально, несколько маршрутов могут использоваться для балансировки нагрузки и повышения производительности. Дополнительная информация, включая методы, с помощью которых поставщики оборудования и компания Microsoft создают многомаршрутные системы, приводится в главе 9.

4.7.2 Типы кабелей Fibre Channel

В основном используется два типа кабелей: оптические и медные. Ниже перечислены основные преимущества и недостатки кабелей.

- Медные кабели дешевле оптических.
- Оптические кабели поддерживают более высокие скорости передачи данных по сравнению с медными кабелями.
- Медный кабель может использоваться на меньшем расстоянии, до 30 метров. При этом оптический кабель может использоваться на расстоянии до 2 километров (многомодовый кабель) или до 10 километров (одномодовый кабель).
- Медный кабель более восприимчив к электромагнитным помехам и взаимному влиянию других кабелей.
- Оптические данные обычно должны быть преобразованы в электрические сигналы для передачи через коммутатор и обратно в оптическую форму для дальнейшей передачи.

Существует только один тип медного кабеля, в отличие от оптического, который представлен двумя видами: многомодовым и одномодовым.

На коротких дистанциях используется многомодовый кабель, который имеет сердцевину диаметром 50 или 62,5 микрона (микрон — микрометр, или одна миллионная часть метра.) Световая волна, которая используется

в многомодовом кабеле, имеет длину 780 нанометров, что не поддерживается в одномодовых кабелях. Для больших расстояний предназначен одномодовый кабель, диаметр сердцевины которого составляет 9 микрон. В одномодовом кабеле используется световой луч с длиной волны в 1300 нанометров. Несмотря на тематику этой главы (интерфейс Fibre Channel), стоит упомянуть, что такие кабели могут использоваться для построения сетей на основе других интерфейсов, например Gigabit Ethernet.

4.7.3 Разъемы

Поскольку интерфейсом Fibre Channel поддерживается несколько типов кабелей (и технологий передачи данных), устройства (например, адаптеры шины, устройства взаимодействия и хранения данных) выпускаются с разъемами, которые поддерживают подключение к передающей среде, что делается для снижения общих затрат. Существует несколько видов разъемов, предназначенных для различных передающих сред и интерфейсов[3].

- *Конверторы интерфейса Gigabit* (Gigabit interface converters — GBIC) поддерживают последовательную и параллельную трансляцию передаваемых данных. Конверторы GBIC предоставляют возможность "горячего" подключения, т.е. включение/выключение GBIC не влияет на работу других портов. Конверторами используется 20-битовый параллельный интерфейс.
- *Модули линий Gigabit* (Gigabit link modules — GLM) предоставляют функции, аналогичные GBIC, но для своей установки требуют отключения устройства. С другой стороны, они несколько дешевле, чем GBIC.
- *Адаптеры интерфейса носителя* (Media Interface Adapters) используются для преобразования сигналов между медным и оптическим носителем и наоборот. Адаптеры интерфейса носителя обычно используются в адаптерах шины, но могут применяться и на коммутаторах и концентраторах.
- *Адаптеры малого формфактора* (Small Form Factor Adapters — SFF) позволяют размещать большее количество разъемов различных интерфейсов на плате определенного размера.

[3] В настоящий момент существует несколько различных физических стандартов, и тот факт, что используется лишь три базовых типа кабелей (медные, одно- и многомодовые), не означает наличия трех типов физических разъемов. Кроме того, эти типы применяются и в других интерфейсах, например Gigabit Ethernet.

4.7.4 Устройства взаимодействия

Устройства взаимодействия соединяют между собой компоненты сетей хранения данных. К ним относятся различные устройства, начиная от дешевых концентраторов Fibre Channel и заканчивая дорогими, высокопроизводительными и управляемыми коммутаторами связной архитектуры. Эти устройства рассматриваются в разделах 4.7.4.1–4.7.4.3.

4.7.4.1 Концентраторы кольца Fibre Channel с разделением доступа

Концентраторы FC-AL представляют собой бюджетный вариант для подключения нескольких узлов Fibre Channel (устройств хранения данных, серверов, компьютерных систем, других концентраторов и коммутаторов) в кольцевую конфигурацию. Обычно в концентраторах предоставляется от 8 до 16 портов. Концентратор может поддерживать различные среды передачи, например медные или оптические.

Концентраторы Fibre Channel — это пассивные устройства, т.е. любое другое устройство в кольце не может обнаружить их присутствия. Концентраторы обеспечивают следующие возможности:

- внутренние соединения, которые позволяют любому порту подключаться к любому другому порту;
- возможность обхода порта, к которому подключено неправильно работающее устройство.

Самая большая проблема в работе портов связана с тем, что в текущий момент времени они могут поддерживать только одно подключение Fibre Channel. На рис. 4.7 показано, что, если порт 1 получил управление для установки сеанса с портом 8, ни один другой порт не сможет передавать данные, пока установленный сеанс не завершится.

Концентраторы могут быть подключены к коммутаторам связной архитектуры Fibre Channel (они рассматриваются в разделе 4.7.4.3) без модификации. Кроме того, можно создавать каскад концентраторов, соединив два концентратора кабелем.

Концентраторы FC-AL занимают лидирующее положение на рынке Fibre Channel, но в процессе снижения стоимости коммутаторы связной архитектуры Fibre Channel становятся все более популярными.

Концентраторы FC-AL создаются такими компаниями, как Gadzoox Networks, Emulex и Brocade.

4.7.4.2 Коммутаторы кольца Fibre Channel с разделением доступа

Самое значительное преимущество коммутаторов FC-AL перед концентраторами состоит в одновременной поддержке нескольких подключений, то-

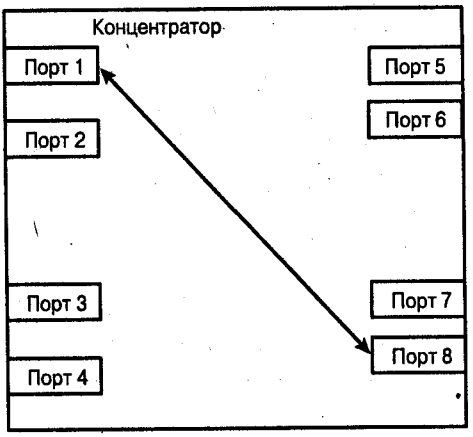


Рис. 4.7. Концентратор Fibre Channel

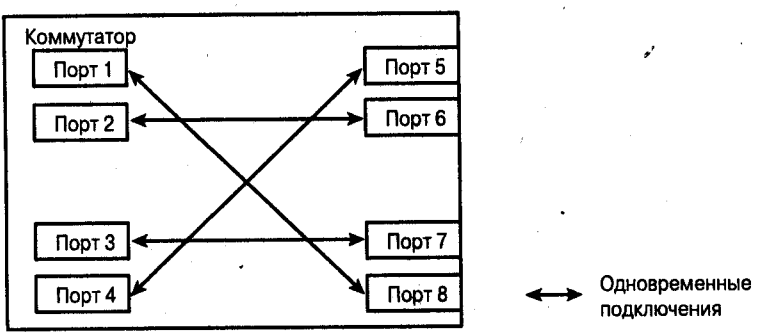


Рис. 4.8. Коммутатор Fibre Channel

гда как концентраторы поддерживают только одно подключение в текущий момент времени (рис. 4.8).

Возможность одновременной поддержки нескольких подключений связана с определенными сложностями. Устройства, подключенные к коммутатору кольца, даже не "подозревают" о своей роли. Коммутаторы кольца участвуют как в передаче данных, так и в адресации кольца. Ниже приводится дополнительная информация по этому вопросу, а также рассматривается роль коммутаторов в сетях хранения данных и методы, с помощью которых поставщики добавляют новые функции к своим продуктам.

Коммутаторы кольца и передача данных

Сервер, который намерен получить доступ к устройству хранения данных, должен отправить арбитражный запрос на управление кольцом. В нормальном кольце FC-AL на базе концентратора каждое устройство получает

арбитражный пакет до его возвращения адаптеру шины сервера, благодаря чему сервер получает контроль над кольцом. Коммутатор кольца отправит ответ об успешном получении управления немедленно, не отправляя запросы другим узлам. На этом этапе адаптер шины отправит базовый пакет `Open`, предназначенный для порта устройства хранения, который будет передан коммутатором кольца. Если порт в это время не выполняет передачи данных, особых проблем не должно возникнуть. В противном случае возможно появление конфликтных ситуаций. Для решения этой проблемы коммутатор кольца должен предоставить буфера для временного хранения кадров, предназначенных для порта 7. Некоторые поставщики коммутаторов предоставляют для этих целей 32 буфера на каждый порт.

Коммутаторы кольца и адресация FC-AL

Концентраторы FC-AL не играют роли в назначении адресов устройствам, а только передают базовые кадры адресов по кольцу. То же можно сказать и о большинстве коммутаторов. Однако некоторые устройства могут настаивать на получении определенного адреса. Некоторые концентраторы имеют возможность управлять порядком инициализации портов, что позволяет определенному порту инициализироваться первому, после чего устройство будет подключено к требующемуся порту.

Коммутаторы и инициализация кольца

Протокол FC-AL требует повторной инициализации кольца при подключении, отключении или повторной инициализации устройства. Такая инициализация кольца может привести к нарушению существующей связи между другими двумя устройствами. Некоторые производители коммутаторов предоставляют возможность выборочно экранировать и передавать пакеты *LIP* (Loop Initialization Primitives). Эта операция предназначена для минимизации проблем, сокращения времени повторной инициализации кольца и по возможности сохранения существующих сеансов передачи данных. В то же время необходимо обеспечить уникальность адресов устройств.

Если все устройства принимают участие в повторной инициализации кольца, дублирования адресов не происходит, так как устройства "защищают" свои адреса. Но, если некоторые устройства не принимают участия в повторной инициализации кольца, необходимо предотвратить назначение уже распределенных адресов устройствам, принимающим участие в повторной инициализации кольца. Уникальность адресов обеспечивается дополнительной логикой коммутатора кольца. При добавлении устройства хранения пакет LIP должен быть отправлен на сервер, однако LIP не требуется передавать

устройствам хранения, которые никогда не устанавливают связь с другими устройствами хранения данных.

Некоторые устройства хранения могут устанавливать связь непосредственно с другими устройствами хранения, что используется для резервного копирования данных. Дополнительная информация об операциях копирования приводится в главе 5.

Коммутаторы кольца и связная архитектура

Если все устройства в кольце "знают" о связной архитектуре, коммутатор кольца передает обычным образом необходимые кадры, например кадры Fabric Login. Если устройства в кольце не поддерживают связную архитектуру, коммутатор кольца должен самостоятельно выполнять достаточно большой объем работы.

Коммутаторы кольца некоторых поставщиков не поддерживают каскадирование. Кроме того, некоторым коммутаторам кольца требуется обновление прошивки перед подключением к коммутаторам связной архитектуры. Ряд коммутаторов следует модернизировать для полной поддержки связной архитектуры перед их подключением к SAN.

Коммутаторы FC-AL производятся такими компаниями, как Brocade, McDATA, Gadzoox Networks, Vixel и QLogic.

4.7.4.3 Коммутаторы связной архитектуры Fibre Channel

Коммутаторы связной архитектуры Fibre Channel (Fibre Channel Fabric Switches — FC-SW) обеспечивают несколько высокоскоростных сеансов связи одновременно со всеми устройствами. На данный момент основные коммутаторы поддерживают быстродействие порядка 1 Гбит/с, в то время как скорость в 2 Гбит/с также перестает быть диковинкой. В основном коммутаторы связной архитектуры в пересчете на один порт стоят дороже, чем концентраторы и коммутаторы FC-AL, но они предоставляют намного больше функциональных возможностей.

Коммутаторы связной архитектуры более эффективны в сравнении с концентраторами и коммутаторами FC-AL. Например, коммутаторы предоставляют специальные службы, описанные выше, обеспечивают управление потоком с помощью базовых пакетов управления, а также, что гораздо важнее, некоторые коммутаторы способны эмулировать функции FC-AL для обеспечения обратной совместимости с более старыми устройствами.

Некоторые коммутаторы связной архитектуры поддерживают *маршрутизацию без буферизации*. Суть ее в том, что при получении заголовка кадра коммутатор быстро находит заголовок точки назначения, пока кадр все еще

принимается. Преимущество такого подхода — снижение задержек при доставке кадра и отсутствие необходимости хранения содержимого кадра в памяти буфера. А недостаток заключается в немедленной передаче всех кадров, включая поврежденные.

Коммутаторы связной архитектуры играют важную роль в безопасности сетей хранения данных на основе Fibre Channel, что описывается более подробно в главе 7.

4.7.4.4 Сравнение трех устройств подключения

В табл. 4.5 приведены функциональные возможности и различия между тремя типами устройств Fibre Channel.

4.7.4.5 Мосты и маршрутизаторы

Как в этой главе, так и во всей книге термины *мосты* (bridges) и *маршрутизаторы* (routers) не относятся к традиционным мостам Ethernet и маршрутизаторам IP. В данном случае под мостами и маршрутизаторами подразумеваются устройства для Fibre Channel, а не для сетевых протоколов 2-го и 3-го уровней.

Мосты — это устройства, обеспечивающие взаимодействие между Fibre Channel и устаревшими протоколами, например SCSI. Мосты Fibre Channel-SCSI позволяют сохранить существующие инвестиции в устройства хранения SCSI. Такие мосты поддерживают интерфейсы SCSI и Fibre Channel и преобразуют данные двух протоколов. Таким образом, новый сервер с установленным адаптером шины Fibre Channel может получить доступ к существующим устройствам хранения SCSI. Мосты предоставляют интерфейс между параллельной шиной SCSI и интерфейсом Fibre Channel. Маршрутизаторы обладают аналогичными возможностями, но для нескольких шин SCSI и интерфейсов Fibre Channel. Маршрутизаторы систем хранения данных, или "интеллектуальные" мосты, предоставляют такие дополнительные возможности, как маскировка и отображение LUN, а также поддерживают команды SCSI Extended Copy. В качестве устройств, передающих данные, маршрутизаторы применяют команды Extended Copy для использования библиотеками хранения, что позволяет копировать данные между указанным целевым устройством и подключенной библиотекой. Эта функция также называется *независимым резервным копированием* (без сервера).

В качестве примера производителей маршрутизаторов и мостов можно привести такие компании, как Crossroads Systems, Chaparral Network Storage, Advanced Digital Information Corporation (ADIC после приобретения Pathlight) и MTI.

Таблица 4.5. Устройства Fibre Channel

	Концентратор	Коммутатор FC-AL	Коммутатор FC-SW
Функциональность	Единая передача на скорости 100 Мбайт/с после переговоров на право передачи	Несколько передач на скорости 100 Мбайт/с после переговоров на право передачи	Несколько передач данных на скорости 1 или 2 Гбит/с без необходимости в переговорах. Возможна эмуляция FC-AL
Производительность	Снижается с увеличением количества узлов	Сохраняется при увеличении количества узлов	Сохраняется при увеличении количества узлов
Доступность данных	Всем узлам доступны все данные, вне зависимости от назначения последних	Данные доступны только получающему и передающему узлу	Данные доступны только получающему и передающему узлу
Восстановление после ошибок	Сложный процесс восстановления после ошибок требует повторной инициализации кольца, что влияет на все узлы, в том числе работающие в обычном режиме	Восстановление после ошибок затрагивает только неправильно работающие узлы	Восстановление после ошибок затрагивает только неправильно работающие узлы
Повторное конфигурирование	При добавлении или удалении узла все узлы принимают участие в повторной инициализации кольца	Только новые узлы и коммутаторы принимают участие в повторной инициализации	Только новые узлы и коммутаторы принимают участие в повторной инициализации
Буфер данных	Концентратор не имеет буфера данных	Коммутатор не имеет буфера данных	Буфера данных у порта и коммутатора. Позволяют выполнять передачу данных без проверки доступности получающего узла

Окончание табл. 4.5

	Концентратор	Коммутатор FC-AL	Коммутатор FC-SW
Адресация	8-разрядная адресация, 127 узлов. Один узел зарезервирован для подключения к коммутатору	16-разрядная адресация, до 16 млн. узлов. Поддерживается вложенная адресация — использование меньшего количества бит, например подобно тому, как последние пять цифр номера используются для внутрикорпоративных звонков	16-разрядная адресация, до 16 млн. узлов
Сложность и стоимость реализации	Низкая	Средняя	Высокая
Управляемость	Обычно не поддерживает управление, но иногда эта функция доступна в виде дополнительного модуля за дополнительную плату	Хорошая управляемость	Отличная управляемость
Дополнительные возможности	Нет	Зонирование	Зонирование и расширенные возможности безопасности. Многомаршрутный режим защиты целостности в сетях с соответствующей топологией. Транкинг линий связи, поддерживаемый некоторыми коммутаторами

4.8 Методы управления Fibre Channel

В предыдущих разделах рассматривались аппаратные элементы, формирующие сети хранения данных. В работе SAN также участвует немало различных программ, в основном предназначенных для управления, обеспечения безопасности, резервного копирования и восстановления данных. В разделах 4.8.1 и 4.8.2 рассматривается ряд концепций, необходимых для управления SAN и обеспечения безопасности данных. По сути, эти концепции предоставляют собой "сердце" SAN.

В ситуации, когда одна сеть содержит несколько компьютеров и единиц хранения данных, желательно ограничить влияние некоторых компьютеров (в терминологии Fibre Channel они называются *узлами*) до определенных подсистем хранения и некоторых единиц в рамках этих подсистем. Это имеет особый смысл в том случае, когда узел работает под управлением Windows NT, которая требует монтирования каждого обнаруженного устройства. С другой стороны, у UNIX есть таблица монтирования, благодаря чему монтируются только устройства, непосредственно указанные в таблице. Даже при использовании узлов под управлением UNIX желательно ограничить доступ из соображений обеспечения безопасности и для снижения вероятности повреждения данных. Доступ может быть ограничен тремя различными типами функций отображения и зонирования.

1. Базовая функция, реализованная в рамках узла; возможно, средствами программного драйвера адаптера шины.
2. Функция коммутатора.
3. Функция на уровне подсистемы хранения данных.

4.8.1 Зонирование

Термин *зонирование* связан с коммутаторами. Зонирование позволяет одним портам коммутатора подключаться только к заранее определенным портам. В некоторых случаях зонирование может ограничивать распространение управляющих кадров Fibre Channel; например, при появлении в кольце нового устройства хранения можно ограничить распространение кадра LIP среди других устройств.

С функциональной точки зрения зонирование дает возможность некоему компьютеру непосредственно подключаться к определенной подсистеме хранения данных. Недостаток такого подхода состоит в предоставлении всех ресурсов SAN для одного компьютера, который обычно не в состоянии полностью их использовать. В частности, зонирование не позволяет совместно загружать канал доступа сети или применять ресурсы подсистемы хранения данных.

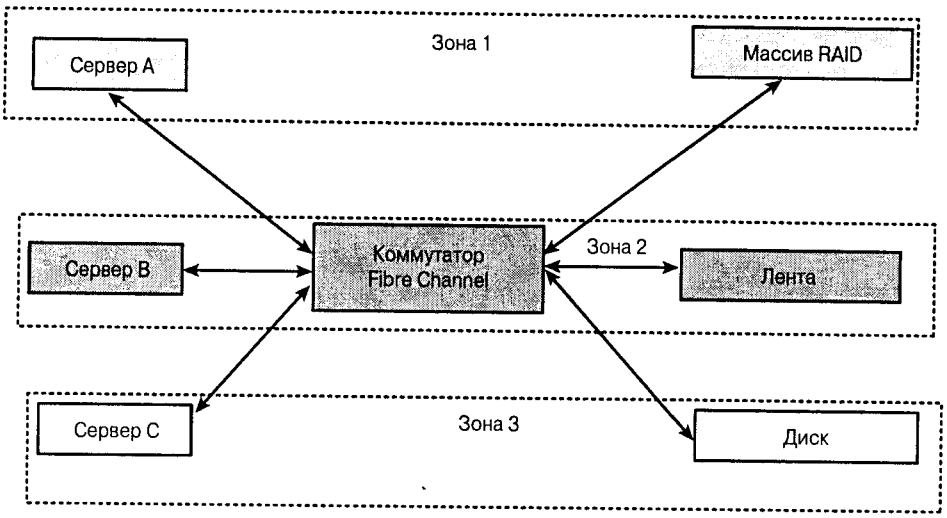


Рис. 4.9. Зонирование SAN

Зонирование можно воспринимать в качестве аналога конфигурирования порта IP для маршрутизатора с поддержкой брандмауэра. Еще одним примером может быть настройка виртуальных локальных сетей (VLAN) в существующей физической локальной сети. В виртуальной локальной сети только некоторые устройства "видят" друг друга, даже если в той же физической локальной сети находятся и другие устройства. Точно так же зонирование ограничивает возможности компонентов SAN (особенно инициаторов), предоставляя ограниченные данные об определенных единицах хранения и возможность доступа к ним, даже если в этой же физической сети хранения данных размещены и другие устройства хранения.

На рис. 4.9 демонстрируется концепция зонирования. Сеть хранения данных имеет три сервера и три единицы хранения. Различными оттенками указываются разные зоны.

Имена LUN могут совместно использоваться программным обеспечением файловой системы SAN. В этом программном обеспечении один или несколько серверов работают как серверы метаданных. Программное обеспечение устанавливается на клиентском компьютере (на компьютере, который желает получить доступ к файлам в сети хранения данных) и на сервере метаданных. Метаданные предоставляют клиентскому компьютеру информацию для отображения логического смещения в файле на физический номер блока указанного устройства. Это позволяет клиентскому компьютеру непосредственно получать доступ к файлу через SAN, без переноса данных через сервер. При достаточно грамотной организации обычные разрешения для файлов на

клиентском компьютере будут относиться и к файлам, хранящимся удаленно, что не требует от администратора дополнительных действий по настройке разрешений на совместный доступ к файлам.

Можно определить несколько зон, причем один узел имеет возможность входить в несколько зон одновременно; таким образом, некоторые зоны будут перекрываться. Зонирование выполняется несколькими способами.

- *Зонирование по номеру порта.* Преимущество такого подхода — эффективность. Если устройство, подключенное к порту, заменено другим устройством, повторная настройка не потребуется.
- *Зонирование по имени WWN.* Осуществляется путем указания имен WWN, которые входят в одну зону. Некоторые WWN могут быть указаны в нескольких зонах. Преимущество состоит в безопасности, которая, однако, достигается за счет эффективности. Изменения в конфигурации могут потребовать перезагрузки сервера.
- *Программное зонирование.* Проводится средствами сервера имен (программного обеспечения), который выполняется на коммутаторе. Для программного зонирования могут использоваться номера портов, WWN или комбинация этих параметров. Сервер имен содержит базу данных, в которой хранятся WWN, номера портов и идентификаторы зон.
- *Аппаратное зонирование.* Осуществляется с помощью таблицы маршрутизации, которая хранится на коммутаторе. Аппаратное зонирование выполняется на основе WWN и не принимает во внимание номера портов.

4.8.2 Маскировка LUN

Ресурсы хранения могут быть "разделены" на несколько вложенных единиц (субъединиц), которые называются *номером логического устройства* (logical unit number — LUN). Стандарт SCSI-2 поддерживает до 64 LUN на одно устройство.

С функциональной точки зрения маскировка LUN позволяет определенному компьютеру получить доступ к конкретной субъединице на некой системе хранения данных. Однако гораздо важнее то, что с помощью этого способа можно запретить доступ к определенным LUN для некоторых компьютеров или серверов. Маскировка LUN дает возможность совместно использовать ресурсы хранилищ данных и (неявно) пропускную способность сети, однако непосредственно LUN совместно использоваться не может. Для совместного использования одного LUN несколькими компьютерами необходима файловая система с дополнительными возможностями, которая описана в главе 6.

Маскировка LUN необходима для гарантирования целостности данных в среде SAN. Обратите внимание: маскировка LUN — это средство обеспечения безопасности на уровне дисков, но не обязательно на уровне файлов. В последнем случае (на уровне файлов) потребуется дополнительное программное обеспечение.

Маскировка LUN предоставляет дополнительные возможности, в частности номера LUN могут быть переназначены другим компьютерам. Существует несколько способов обеспечения маскировки LUN. Каждый способ обладает своими достоинствами и недостатками. Обычно, маскировка выполняется средствами:

- аппаратного обеспечения адаптера шины;
- аппаратного обеспечения коммутатора Fibre Channel;
- аппаратного обеспечения устройства хранения Fibre Channel;
- программного обеспечения узла.

Эти варианты рассматриваются в разделах 4.8.2.1–4.8.2.4.

4.8.2.1 Маскировка LUN средствами BIOS адаптера шины

В BIOS адаптера шины осуществляется маскировка всех LUN, которые не отображены в таблице BIOS адаптера шины. Таким образом, узел (с установленным адаптером шины) попросту не "замечает" существования LUN, которые он и не должен "видеть".

Недостаток такого метода состоит в необходимости проведения корректной настройки; кроме того, метод не обязателен к применению. Все системы, адаптеры шины которых настроены неправильно или не поддерживают описываемую функцию, могут получить доступ к тем LUN, к которым доступ на самом деле нежелателен. Еще одна проблема заключается в сложности динамического управления и перенастройки подобных систем.

4.8.2.2 Маскировка LUN коммутаторами Fibre Channel

Коммутаторами Fibre Channel зонирование проводится достаточно просто. Входящий пакет передается или не передается дальше, что зависит от адресов исходного порта и порта назначения. Маскировка LUN возлагает дополнительную нагрузку на коммутаторы Fibre Channel, поскольку коммутатору приходится проверять первые 64 байта каждого пакета данных. Это приводит к снижению производительности большинства коммутаторов Fibre Channel, поэтому описываемая функция обычно не реализуется.

4.8.2.3 Маскировка LUN контроллерами подсистем хранения данных Fibre Channel и маршрутизаторами

Этот метод маскировки LUN является принудительным для подключенных узлов или требует от узла минимального участия. Маскировка LUN реализуется контроллером подсистемы хранения данных или маршрутизатором (с помощью соответствующей прошивки). Эти устройства настроены на поддержку таблицы имен WWN адаптера шины, отображённых на номера LUN, к которым им (контроллеру или маршрутизатору) разрешен доступ. Значительное преимущество такого подхода заключается в формировании конфигурации, независимой от промежуточных коммутаторов или концентраторов.

Недостаток метода заключается в закрытой реализации этой технологии каждым поставщиком и сложности создания единой консоли управления для перенастройки или даже получения информации о текущих параметрах, хотя каждый поставщик предоставляет интерфейсы для управления связками WWN–LUN.

К поставщикам систем, поддерживающим эту технологию, относятся Crossroads Systems, EMC, Dot Hill и HP (в продуктах StorageWorks). Поставщики присваивают реализации технологии собственные названия; например, компания Crossroads называет это *Access Controls*, а компания HP в продуктах StorageWorks выбрала название *Selective Storage Presentation*.

4.8.2.4 Маскировка LUN программным обеспечением узла

Маскировка LUN выполняется программным обеспечением узла, в частности кодом драйвера устройства. Код должен работать в режиме ядра, так как основная идея заключается в том, чтобы предотвратить доступ операционной системы к LUN, а операционная система сделает это еще до запуска первого приложения пользовательского режима.

Такая маскировка может выполняться в виде функции операционной системы или вне системы. За неимением конкретного решения от Microsoft некоторые поставщики добавили необходимый код в драйвер адаптеров шины. Обычно драйвер выдает команду `Report LUNs` каждому устройству, подключенному к шине, и перед предоставлением списка LUN системе Windows NT драйвер "вырезает" LUN из списка на основе дополнительно запрошенных данных (например, информации системного реестра Windows NT), таким образом "скрывая" некоторые LUN от Windows.

Основная проблема такого метода — необязательная настройка, а следовательно, необходимость частичного участия узла в процессе маскировки LUN. Это означает, что компьютеры, не имеющие модифицированного драйвера адаптера шины, не принимают участия в маскировке LUN. Кроме того, присутствуют и проблемы масштабирования, так как в особенно больших сетях

хранения данных сложно настроить каждый сервер и каждый адаптер шины сервера. Что касается преимуществ, то LUN может эффективно использоваться несколькими серверами.

Описываемая функция реализуется в продуктах компаний Emulex, Dell и JNI.

4.8.2.5 Маскировка LUN и будущее Windows NT

На данный момент существует информация, что Microsoft работает над реализацией возможностей маскировки LUN в драйвере порта. Тем не менее такая возможность отсутствует в Windows Server 2003. Преимущество использования драйвера порта состоит в постоянном присутствии драйвера порта в памяти, поэтому время, в течение которого компьютер не будет принимать участие в маскировке LUN, существенно снижается. Вероятность загрузки неправильного драйвера порта намного ниже, чем вероятность загрузки неправильного драйвера порта и мини-порта. Судя по предварительным прогнозам, если описываемая функция будет реализована в Windows, администратор получит возможность самостоятельно определять и изменять список LUN, видимых для сервера; при этом список может быть изменен временно. В последнем случае изменения не будут сохраняться после перезагрузки сервера.

4.9 Обеспечение взаимодействия устройств Fibre Channel

Призыв "Покупатель, берегись!" хорошо описывает состояние взаимодействия устройств в мире Fibre Channel.

Можно сказать, что большинство проблем во взаимодействии конфигураций FC-AL связаны с устройствами хранения, адаптерами шины, коммутаторами FC-AL и поставщиками маршрутизаторов. Поставщики устройств проводят серьезное тестирование своих продуктов, но, хотя теоретически взаимодействие с другими устройствам и должно быть гарантировано, на практике для получения результата требуется немало дополнительного тестирования и настроек различных параметров. Рекомендуется использовать конфигурации, которые были протестированы поставщиком или продавцом готовых решений SAN.

Наибольшая проблема состоит в отсутствии гарантированного соответствия промышленным стандартам. Более того, даже соответствие стандартам также не обеспечивает 100%-ного взаимодействия.

Поставщики готовых решений, такие, как IBM, HP и EMC, создают лаборатории для тестирования взаимодействия различных устройств и проводят

собственную сертификацию. До определенной степени другие поставщики поступают аналогично. Рекомендуется использовать именно такие сертифицированные решения, что позволяет избежать проблем, часто возникающих при добавлении новых, не сертифицированных поставщиком устройств.

Хотя немало сетей хранения данных на основе Fibre Channel обеспечивают быстродействие 1 Гбит/с, в последнее время в продаже появились устройства, поддерживающие скорость 2 Гбит/с. Новые устройства — новые проблемы. В стандартах, которым следуют производители, поддерживается скорость 2 Гбит/с, однако устройства автоматически переходят на скорость 1 Гбит/с, если на этой скорости работают другие устройства в сети. Дело в том, что сети хранения данных на базе Fibre Channel должны работать на скорости самого медленного устройства в сети. Таким образом, даже единственное устройство, работающее на скорости 1 Гбит/с, заставит всю сеть хранения данных работать на этом уровне быстродействия.

4.10 Сложности практической реализации

Сети хранения данных на основе Fibre Channel эмулируют прямое подключение устройства хранения данных к серверу, даже если устройство на самом деле подключено через коммутатор. Таким образом, в контексте Windows доступ к устройствам Fibre Channel осуществляется с помощью драйверов SCSIPort или Storport, описанных в главе 2. Таким образом, особенности работы с хранилищем, подключенным непосредственно к серверу (DAS), имеют отношение и к SAN.

Новая модель драйверов Storport предоставляет массу функциональных возможностей, включая оптимизацию ввода-вывода и управление пропускной способностью сети, однако системные администраторы и ответственные лица в информационных отделах компаний должны обратить внимание на тот факт, что модель драйверов Storport поддерживается исключительно в Windows Server 2003. Принявшим решение об использовании платформы Windows стоит изучить планы поставщика устройств хранения данных относительно перехода на модель Storport. В то же время необходимо обратить внимание на реализацию поддержки этих устройств на базе платформы Windows 2000, включая подробности реализации драйвера устройства. Это особенно важно для определения адекватности пропускной способности устаревающей модели драйверов SCSIPort, если поставщик будет продолжать ее применение. Кроме того, необходимо узнать, предоставляет ли поставщик собственную архитектуру SAN, без модели драйверов SCSIPort, а также сертифицировано ли это решение и поддерживается ли оно всеми заинтересо-

ванными сторонами. Наконец, обратите внимание на планы поставщика по переходу на модель драйверов Storport для Windows Server 2003.

Маскировка LUN на данный момент не поддерживается в продаваемых версиях Windows, причем выпуск Windows Server 2003 не изменил ситуации. Прежде чем приобретать новое программное и аппаратное обеспечение, выясните, какую технологию использует поставщик для реализации маскировки LUN и насколько она подходит для работы в среде Windows.

4.11 Резюме

Сети хранения данных Fibre Channel составляют существенную часть корпоративных подсистем хранения данных. Технология Fibre Channel может внедряться в виде недорогих конфигураций на основе кольца или на базе набирающей популярность топологии коммутируемой связной архитектуры.

Операционная система Windows Server 2003 поддерживает устройства Fibre Channel с помощью драйвера Storport, предоставляемого поставщиком аппаратного обеспечения. Поставщик вместо этого может предоставить мини-драйвер порта SCSI, но в таком случае преимущества драйвера Storport (например, повышенная производительность и обработка ошибок) окажутся недоступными для пользователей. Операционная система Windows 2000 и предыдущие ее версии поддерживают устройства Fibre Channel посредством мини-драйвера SCSIPort, предоставляемого поставщиками аппаратного обеспечения.

Несмотря на то что Windows NT поддерживает технологию маскировки LUN и зонирования, базовая поддержка маскировки LUN в Windows NT отсутствует. Маскировка LUN в Windows NT может быть реализована в драйвере от поставщика аппаратного обеспечения.

Глава 5

Технологии резервного копирования и восстановления данных

Резервное копирование — это процесс создания когерентной (непротиворечивой) копии данных. Резервное копирование становится все более важным на фоне значительного увеличения объема данных в компьютерной индустрии. Некоторые исследования показывают, что в ближайшие несколько лет будет создано больше данных, чем за всю историю человечества! Очень интересно сравнить увеличение емкости подсистем хранения данных с более известным ростом плотности транзисторов в электронных компонентах. Закон Мура гласит, что количество транзисторов на единицу площади электронных микросхем удваивается каждые 18 месяцев. Аналитики предполагают, что рост объемов подсистем хранения данных намного обгоняет закон Мура и объемы хранилищ удваиваются значительно быстрее, чем за 18 месяцев.

Исторически сложилось, что для резервного копирования данных используются накопители на магнитной ленте. Изначально лента считалась более дешевым носителем, чем жесткие диски. Впоследствии возникло мнение, что самыми дешевыми являются оптические носители, но по ряду причин это мнение не нашло практической реализации. Хотя для резервного копирования в основном применяется магнитная лента, обычные жесткие диски также стали популярным средством первичного резервного копирования и зеркального отражения систем. Эта тенденция связана со снижением цен на жесткие диски, что сокращает преимущество накопителей на магнитной ленте в стоимости. Еще одна причина использования жестких дисков — более высокое быстродействие, что приводит к снижению времени обслуживания серверных приложений.

Обратите внимание: и жесткие диски, и ленточные приводы в качестве носителей для резервного копирования обладают определенными преимуществами и недостатками. Несмотря на недостатки, и тот и другой носитель будут использоваться в дальнейшем. Накопители на магнитной ленте обладают высокой емкостью, кассеты можно легко переносить в отдельно распо-

ложенный архив или использовать для восстановления после сбоев в работе. После создания первоначальной копии данных на жестком диске вторичное резервное копирование зачастую выполняется с помощью магнитной ленты.

В этой главе рассматриваются технические трудности, которые необходимо преодолеть для обеспечения своевременного резервного копирования и восстановления данных. Здесь также приводится классификация методов восстановления и резервного копирования. Кроме того, описываются возможности Windows Server 2003 по созданию "моментальных снимков" (служба теневого копирования томов) и по работе с сетевым протоколом управления данными (Network Data Management Protocol — NDMP), а также видение компании Microsoft относительно управления подсистемами хранения данных, что будет реализовано в следующих версиях Windows.

5.1 Причины резервного копирования и восстановления данных

Резервное копирование проводится по ряду причин, которые обычно оправдывают инвестиции, вложенные в соответствующее оборудование. Основная цель создания резервных копий — это обеспечение гарантированной доступности данных. Чем важнее постоянный доступ к данным, тем больше инвестиций потребуется. Например, одна из популярных методик резервного копирования заключается в зеркальном копировании дисков, при котором каждая операция записи дублируется для второго диска, что гарантирует доступность данных при отказе в работе первого диска.

Кроме того, архивирование данных проводится для выполнения различных корпоративных требований, при которых данные не должны быть обязательно доступными мгновенно, но могут быть затребованы позднее. В таких случаях доступность данных должна быть обеспечена в течение разумного периода времени, который измеряется в часах, днях или неделях.

Резервные копии иногда используются для перемещения данных, например при создании удаленного центра хранения данных в другом географическом регионе. Такой же причиной будет перенос данных на новое аппаратное обеспечение или, что случается реже, на другую серверную платформу.

5.2 Проблемы при резервном копировании

Перед подробным обсуждением различных способов резервного копирования и восстановления желательно разобраться в проблемах, которые необходимо решить для получения требующегося результата. Далее перечислены основные проблемы.

- Сокращение промежутка времени, который называется *окном резервного копирования*, в течение которого должна быть завершена операция резервного копирования.
- Постоянно увеличивающееся количество программных интерфейсов приложений (API), которые должны поддерживаться приложениями резервного копирования.
- Невозможность резервного копирования файлов, которые открыты и активно используются приложениями.

Более подробно эти проблемы рассматриваются в разделах 5.2.1–5.2.3.

5.2.1 Время, затрачиваемое на резервное копирование

Исторически сложилось так, что серверные приложения запускались только в рабочее время. Операции резервного копирования соответственно выполнялись в нерабочее время, т.е. ночью, когда работу приложений можно остановить, не оказывая влияния на пользователей. Как только работа приложения прекращена, сервер можно отключить от сети и провести резервное копирование данных. С этим подходом связаны две проблемы.

1. Значительное увеличение объема данных усложняет завершение резервного копирования в выделенный период времени. Как ни странно, но запись на магнитную ленту в контексте как затрачиваемых машиночасов, так и времени обслуживающего персонала весьма неэффективна. Необходимо найти ленту, вставить ее в накопитель и перемотать на нужную позицию. Как только позиция будет найдена, запись данных на ленту будет проводиться намного медленнее, чем на жесткий диск. Интерфейсы жестких дисков поддерживают запись со скоростью на порядок больше 80 Мбайт/с, а самые быстрые накопители на магнитной ленте поддерживают максимальную скорость передачи 30 Мбайт/с. Для управления несколькими накопителями могут использоваться роботизированные библиотеки, которые весьма недешевы и помогают сократить затраты времени только на поиск и загрузку ленты. Такие библиотеки не в состоянии увеличить скорость чтения данных или их записи на ленту.
2. Вторая проблема заключает в том, что все больше приложений, а также создаваемые, управляемые и модифицированные ими данные рассматриваются как важные, если не критические, для выживания компании в конкурентной среде. Это означает, что время, в течение которого можно отключить сервер от сети для резервного копирования, сокращается.

5.2.2 Увеличение количества программных интерфейсов приложений

Потребители используют все больше корпоративных приложений, которые очень редко, если это вообще возможно, разрешено останавливать для резервного копирования. По этой причине каждый поставщик приложений предоставляет API для резервного копирования и восстановления файлов с данными приложения. Хотя создание таких API выглядит весьма оптимистично, на самом деле ситуация только ухудшилась.

На рис. 5.1 представлена проблема поддержки все увеличивающегося количества API для резервного копирования и восстановления данных. Как видите, потребители обычно используют несколько приложений, и очень часто применяется несколько версий одного и того же приложения. Каждый поставщик систем резервного копирования должен создавать программный код, использующий API, предоставленный для каждого корпоративного приложения. Поскольку многие поставщики приложений отдельно лицензируют агенты резервного копирования для различных приложений, сам процесс отслеживания лицензий на программное обеспечение и их стоимости может вызвать смятение у менеджера отдела информационных технологий. Более того, следует учесть развертывание инфраструктуры, обучение персонала и четкое выполнение инструкций, необходимых для эффективного резервного копирования.

5.2.3 Проблема открытых файлов

Еще одна проблема при выполнении резервного копирования связана с тем, что процесс занимает значительное время. Если устройство записи на магнитную ленту поддерживает запись со скоростью 10 Гбайт/мин, резервное копирование диска объемом в 100 Гбайт займет 10 мин. В течение этих 10 мин приложения будут получать доступ к диску и вносить изменения в данные, записанные на диске. Существует три подхода к обеспечению целостности резервной копии.

1. Запрет приложениям доступа к диску в процессе резервного копирования. Блокирование одновременного доступа пользователей к диску во время резервного копирования было достаточно распространенным на раннем этапе использования персональных компьютеров, когда работа в режиме 24x7 не практиковалась. Резервное копирование выполнялось в периоды пониженной нагрузки, например в ночные часы. Теперь этот подход не всегда возможен, и тому есть ряд причин.

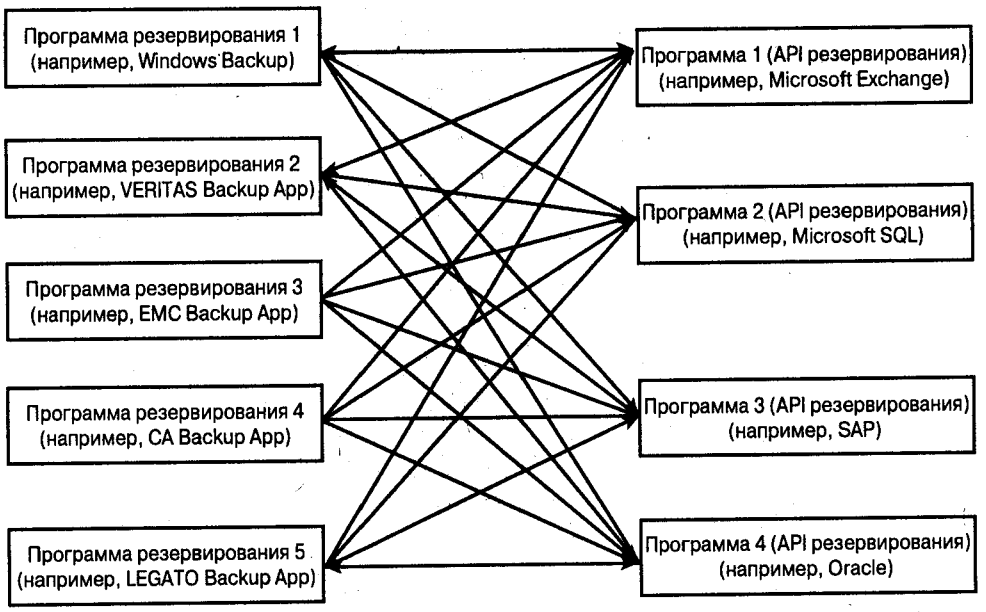


Рис. 5.1. Экспоненциальное увеличение количества API для резервного копирования

- В требованиях к работоспособности системы часто указан режим работы 24x7, поэтому более подходящего времени для резервного копирования попросту не существует.
- Объем данных, которые необходимо разместить в резервной копии, возрастает, как и время активного использования этих данных, поэтому окна резервного копирования не всегда хватает для завершения операции копирования.

2. Резервное копирование данных, в то время когда приложения получают доступ к диску, пропуская открытые файлы. Проблема заключается в том, что в процессе резервного копирования работают только действительно важные приложения, поэтому при таком подходе крайне необходимые данные могут не попасть в резервную копию!
3. Разделение ввода-вывода, инициированного приложением резервного копирования, и ввода-вывода, инициированного другими приложениями. Поставщики программ резервного копирования частично смоделировали ряд функций операционной системы. В частности, их программы зависят от возможности различать источники ввода-вывода. Однако такой метод вполне может оказаться бесполезным. Программы резервного копирования обычно в той или иной мере используют недо-

кументированные возможности операционной системы, которые могут измениться с выходом новой версии. Кроме того, требуется достаточно большой объем свободного дискового пространства. Еще один вариант заключается в обработке каждого файла в отдельности или всех файлов одновременно.

Для резервного копирования открытых файлов с одновременным сохранением целостности резервной копии данных также используется три подхода.

Первый подход — перенос записи приложений во вторичную область хранения, что позволяет приложению резервного копирования делать резервную копию всех файлов. Такой подход должен работать выборочно; например, запись в файл подкачки будет разрешена, а запись в файлы данных приложений должна откладываться или размещаться в предварительно определенном вторичном кэше (он часто называется *вторичным хранилищем*), что позволяет обеспечить целостное резервное копирование данных. Ввод-вывод данных во вторичную область хранения также должен осуществляться особым образом, в зависимости от того, выполняется ли он приложением для резервного копирования или другой программой. Как только приложение для резервного копирования завершит работу, данные из вторичного хранилища должны быть скопированы поверх обычных файлов.

Второй подход — копирование данных при их записи приложением резервного копирования. Как только приложение резервного копирования открывает файл, другим приложениям будет по-прежнему разрешена в него запись. Для того чтобы старые и новые данные не смешались, перезаписываемые данные копируются во вторичное хранилище. Если обычные приложения запрашивают эти данные, операция чтения обрабатывается базовыми драйверами файловой системы Windows. По запросу приложения резервного копирования данные извлекаются из хранилища. Компания St. Bernard Software реализовала такой подход в своих системах для резервного копирования открытых файлов.

Обратите внимание на уровни драйверов, показанные на рис. 5.2 (подробное описание драйверов Windows, объектов устройств и т.д. приводится в главе 1). Драйверы фильтрации файловой системы размещены над драйвером файловой системы NT (NTFS), который, в свою очередь, расположен над драйвером фильтрации диска. Последний находится над драйвером класса диска, ниже которого находятся и другие драйверы (см. главу 1), однако в данном случае они нас не интересуют. Как только приложение открывает файл, NTFS (в ответ на запрос приложения) отправляет последовательность команд для чтения метаданных (расположение файла на диске) и отправляет запросы на чтение и запись логических блоков, где хранится этот файл.

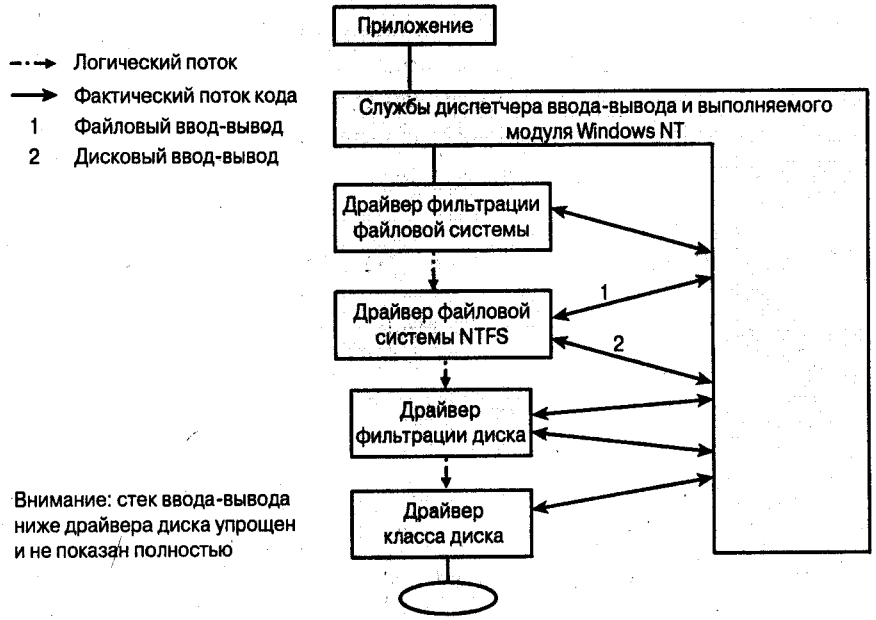


Рис. 5.2. Драйверы фильтрации Windows NT

Драйвер фильтрации верхнего уровня (он расположен над файловой системой) показан на рис. 5.2. Расположение этого драйвера идеально подходит для перехвата выполняемых над файлами операций и перенаправления вызовов, если это необходимо для решения проблемы резервного копирования открытых файлов. Компания Microsoft предлагает набор Windows Installable File System (IFS), в котором представлена информация, необходимая для написания подобного драйвера фильтрации. Разработчики программ резервного копирования могут решить проблему на более низком уровне; например, уровень образа обычно требует создания драйвера фильтрации нижнего уровня (он находится над драйвером класса диска), что показано на рис. 5.2.

Операции ввода-вывода (см. рис. 5.2) выполняются на уровне файловой системы, что показано стрелкой, обозначенной цифрой 1. Драйвер NTFS управляет отображением данных файла на дисковые блоки; операции ввода-вывода выполняются на уровне дисковых блоков, что показано стрелкой, обозначенной цифрой 2. Компания Microsoft предоставляет драйвер фильтрации `diskperf.sys`, который входит в набор разработки Windows Driver Development Kit (DDK). Несколько поставщиков систем резервного копирования использовали набор DDK для создания программ, с помощью которых выполняется моментальный снимок данных.

Третий подход — создание моментального снимка данных и резервное копирование этого снимка в то время, когда приложения будут продолжать использовать оригинальный том. Моментальный снимок может быть создан с помощью различных программных и аппаратных решений, которые Microsoft предлагает в качестве базовой стратегии в Windows Server 2003.

5.3 Классификация типов резервного копирования

Существуют различные схемы резервного копирования, которые применяются, например, в центре хранения данных. Стоит отметить, что различные категории резервного копирования могут использоваться совместно. Резервное копирование классифицируется следующим образом:

- на базе архитектуры;
- на основе функциональных возможностей;
- на базе сетевой инфраструктуры.

Рассмотрим каждый тип классификации подробнее.

5.3.1 Классификация резервного копирования на базе архитектуры

Один из типов классификации резервного копирования основан на архитектуре. Резервное копирование зависит от объектов, к которым оно применяется, и от того, насколько приложение резервного копирования поддерживает подобные объекты. Доступные архитектурные типы резервного копирования описаны в разделах 5.3.1.1–5.3.1.3.

5.3.1.1 Резервное копирование на уровне дисковых образов и логических блоков

В этом случае приложение резервного копирования работает с блоками данных. Обычно подобная схема резервного копирования требует прекращения доступа к копируемым данным со стороны всех приложений на сервере. Приложение получает доступ к жесткому диску независимо от его внутренней структуры, после чего выполняет операции чтения/записи на уровне логических блоков.

Преимущество такого типа резервного копирования состоит в быстродействии операций резервного копирования и восстановления данных, что особенно важно для восстановления данных после критических сбоев в работе систем. Недостаток заключается в том, что существует запрет на доступ к диску со стороны приложений и даже операционной системы. Еще один

недостаток — это копирование излишнего количества неиспользуемых логических блоков с резервной копии при резервировании диска с разрешенными файлами. Некоторые приложения резервного копирования предоставляют соответствующую программную логику, необходимую для обнаружения и пропуска неиспользованных логических блоков. Такие резервные копии называются *разреженными копиями дискового образа.*

Наконец, довольно сложно получить только определенный файл или несколько файлов, в отличие от восстановления всех данных на диске. Для этого программное обеспечение резервного копирования должно обработать метаданные файловой системы, сохраненные на магнитной ленте, и вычислить расположение на ленте необходимого файла. Некоторые программы позволяют восстанавливать определенные файлы из резервной копии на уровне образа, однако лишь для некоторых операционных систем. Другие приложения пытаются оптимизировать восстановление файла из резервной копии уровня образа, записывая на ленту метаданные файла, например таблицу расположения файлов для файловой системы FAT16.

Версия NTFS, которая поставляется вместе с Windows 2000, уже содержит все метаданные в файлах, например битовую карту, которая соответствует расположению логических блоков. Программа восстановления данных находит необходимые метаданные, из которых рассчитывает расположение на магнитной ленте каждого необходимого логического блока требующегося файла. После этого лента прокручивается в одном направлении и все необходимые участки считываются в процессе перемотки, что позволяет получить все данные для восстановления файла. Лента не перематывается в обоих направлениях, поэтому сокращается не только время восстановления, но и срок жизни ленты. К описываемым приложениям резервного копирования относится, например, программа Legato Celestra.

Обратите внимание, что иногда выбор метода резервного копирования ограничен. Если база данных использует чистый дисковый том без файловой системы, то выбирать приходится только между резервной копией на уровне образа и резервной копией на уровне приложения (такой тип резервного копирования рассматривается в разделе 5.3.1.3).

5.3.1.2 Резервное копирование на уровне файлов

В этом типе резервного копирования программа резервирования пользуется услугами операционной и файловой систем. Одно из преимуществ заключается в эффективности восстановления конкретного файла или набора файлов. Еще одно преимущество состоит в возможности одновременного доступа к файлам со стороны операционной системы и приложений, когда проводится резервное копирование.

Не обошлось здесь, впрочем, и без недостатков. Резервное копирование выполняется дольше, особенно по сравнению с резервным копированием на уровне образа. Если проводится копирование большого количества небольших файлов, нагрузка на операционную и файловую систему при доступе к метаданным каталогов может оказаться значительной. Кроме того, существует проблема открытых файлов, которая была описана ранее.

Еще один недостаток связан с безопасностью. Эта проблема возникает вне зависимости от метода создания резервной копии (на уровне образа или файла) и заключается в том, что резервное копирование выполняется на правах учетной записи администратора или оператора резервного копирования, а не пользователя. Это единственный способ восстановить файлы различных пользователей в ходе одной операции восстановления. Необходимым условием является корректная настройка метаданных файлов, например списков управления доступом и данных о владельцах файлов. Решение проблемы требует поддержки со стороны API файловой и операционной систем, что необходимо для настройки метаданных при восстановлении данных из резервной копии. Кроме того, приложение резервного копирования и восстановления должно корректно использовать предоставленные возможности.

5.3.1.3 Резервное копирование на уровне приложения

В этом случае резервное копирование и восстановление данных выполняется на уровне приложения, например Microsoft SQL Server или Microsoft Exchange. Резервное копирование проводится с помощью API, предоставленного приложением. В данном случае резервная копия состоит из набора файлов и объектов, которые формируют состояние системы на определенный момент времени. Основная проблема заключается в том, что операции резервного копирования и восстановления тесно связаны с приложением. Если с выходом нового приложения изменится API или функции уже существующего API, администратору придется переходить к новой версии программы резервирования.

Приложения используют чистый диск без файловой системы или записывают на него огромный файл, в котором размещены собственные метаданные приложения. В качестве примера подобного приложения можно указать Microsoft Exchange. В Windows XP и Windows Server 2003 поддерживаются важные функции NTFS, благодаря которым возможно восстановление таких файлов. Файл восстанавливается логическими блоками и в конце маркируется новой функцией Win32 API, которая называется `SetFileValidData`.

5.3.2 Классификация резервного копирования на базе функциональных возможностей

Еще один метод классификации приложений резервного копирования заключается в классификация на базе функций, предоставляемых в процессе резервного копирования. Обратите внимание, что обычно в центрах хранения данных используется, как минимум, два, а чаще всего все типы резервирования, описанные ниже, а именно: полное, дифференциальное и инкрементное.

5.3.2.1 Полное резервное копирование

При *полном резервном копировании* (full backup) полный набор файлов или объектов, а также связанные с ними метаданные копируются на носитель резервной копии. Преимущество состоит в том, что используется только один набор носителей для восстановления в случае отказа в работе системы. Недостаток заключается во времени копирования, так как копируются все данные. Полное резервное копирование часто выполняется на уровне дискового образа или на уровне блоков.

5.3.2.2 Дифференциальное резервное копирование

При *дифференциальном резервном копировании* (differential backup) архивируются *все изменения, которые произошли с момента последнего полного резервного копирования*. Так как дифференциальные резервные копии могут создаваться на уровне образа или на уровне файлов, этот набор изменений будет представлять собой набор изменившихся дисковых блоков (для резервной копии на уровне образа) или набор изменившихся файлов (для резервной копии на уровне файлов). Основное преимущество дифференциального резервного копирования состоит в значительном уменьшении времени копирования по сравнению с полным резервным копированием. С другой стороны, восстановление после сбоя занимает больше времени. Восстановление после сбоя потребует проведения двух операций по восстановлению данных. В ходе первой будут восстанавливаться данные из полной резервной копии, а во время второй — данные из дифференциальной резервной копии.

При использовании недорогих подсистем хранения данных дифференциальное резервное копирование на уровне файлов применяется в тех случаях, когда приложения создают множество небольших файлов и после создания полной резервной копии меняют некоторые файлы. В то же время такое резервное копирование не применяется, если жесткий диск используется приложениями управления базами данных, которые постоянно вносят небольшие изменения в огромные файлы баз данных. Таким образом, при резервировании на уровне файла будет создана копия целого файла. Примером такой

программы служит Microsoft Exchange, которая постоянно стремится вносить небольшие изменения в огромные файлы баз данных.

При использовании старших моделей подсистем хранения данных дифференциальное резервное копирование на уровне образа можно использовать в любой ситуации, включая резервное копирование файлов приложений баз данных. Причина такой эффективности состоит в хранении большого объема метаданных, которые позволяют быстро определить изменившиеся с момента резервного копирования дисковые блоки. Таким образом, будет проведено резервное копирование только изменившихся дисковых блоков, а большое количество не изменившихся дисковых блоков не будут скопированы. Даже несмотря на более высокую эффективность резервного копирования при использовании старших моделей подсистем хранения данных, остается необходимость в использовании API, который позволит начать резервирование в определенный момент времени и продолжить ввод-вывод данных после завершения резервного копирования. Метод работы старшей модели подсистемы хранения заключается в сокращении операций ввода-вывода данных, которые должны быть остановлены при резервном копировании.

5.3.2.3 Инкрементное резервное копирование

При *инкрементном резервном копировании* (incremental backup) архивируются *только изменения с момента последнего полного или дифференциального резервного копирования.* Очевидно, что этот вид резервного копирования требует меньше времени, так как на резервный носитель не копируются файлы, которые не изменились с момента создания последней полной или добавочной резервной копии. Недостатком этого метода является длительность операции восстановления после сбоя, так как оно выполняется с помощью набора из нескольких носителей, соответствующих последней полной резервной копии и нескольким добавочным резервным копиям.

В случае отсутствия старших моделей подсистемы хранения добавочное резервное копирование выполняется при изменении или добавлении различных наборов файлов. При использовании старших моделей подсистемы хранения может применяться добавочное резервное копирование на основе блоков, так как в этом случае доступен достаточный объем метаданных для идентификации изменившихся блоков.

5.3.3 Классификация резервного копирования на основе сетевой инфраструктуры

Один из способов классификации резервного копирования основан на сетевой топологии и ее влиянии на выбор наилучшего метода резервирования

подключенных узлов. Типы резервного копирования, зависящие от сетевой инфраструктуры (резервирование DAS, NAS, SAN, не зависящее от локальной сети и от сервера) рассматриваются в разделах 5.3.3.1–5.3.3.4.

5.3.3.1 Резервирование DAS

Эта старейшая разновидность резервного копирования возникла во времена, когда устройства хранения подключались непосредственно к серверу. Несмотря на развитие сетевых устройств хранения, резервирование DAS остается достаточно популярным для копирования данных, размещенных на серверах Windows. Схема резервирования DAS представлена на рис. 5.3.

Преимуществом резервирования DAS является простота его использования. Приложение на сервере считывает данные с соответствующего дискового тома и записывает их на магнитную ленту. Однако резервирование DAS имеет ряд недостатков.

- Использование нескольких накопителей на магнитной ленте (по одному на каждый сервер, нуждающийся в резервном копировании), что требует существенных финансовых затрат. Другими словами, совместное использование одного накопителя несколькими серверами практически невозможно.
- Высокая общая стоимость владения (TCO), так как для резервного копирования с помощью нескольких накопителей на магнитной ленте требуется иметь в штате несколько администраторов.
- Хранение нескольких лент может привести к путанице.
- Поскольку данные на нескольких серверах часто дублируются, но не синхронизированы, одинаковые данные переносятся и на ленту, поэтому хранение похожих данных на нескольких лентах может привести к путанице.

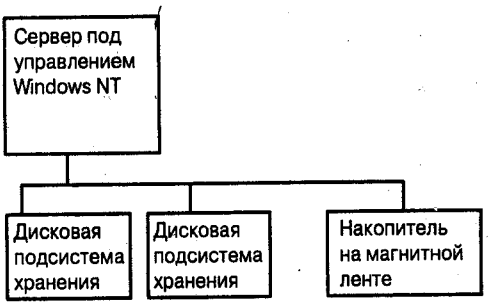


Рис. 5.3. Резервирование DAS

- Наконец, но не в последнюю очередь, сервер должен обрабатывать запросы чтения/записи данных между диском и накопителем на магнитной ленте.

5.3.3.2 Резервирование NAS

Как отмечалось в главе 3, эра хранилищ DAS закончилась с появлением систем типа клиент/сервер, когда клиенты и серверы стали совместно использовать ресурсы локальной сети. Это позволило сформировать архитектуру, в которой к накопителю на магнитной ленте, подключенному к серверу, получают доступ несколько сетевых серверов.

На рис. 5.4 показан типичный сценарий резервирования NAS. В левой области диаграммы указано несколько серверов. Это могут быть серверы приложений или файловые серверы и серверы печати. В правой области находится сервер резервного копирования и подключенный к нему накопитель на магнитной ленте. Этот накопитель может использоваться для резервного копирования информации с нескольких серверов приложений, файловых серверов и серверов печати. Таким образом, резервирование NAS позволяет совместно использовать накопитель на магнитной ленте для резервного копирования данных нескольких серверов, что приводит к снижению общих затрат.

Резервированию NAS свойственны некоторые недостатки.

- Операция резервного копирования отражается на пропускной способности локальной сети, что зачастую требует сегментации LAN для перенаправления потоков резервного копирования в отдельный сетевой сегмент.
- Время работы узлов увеличивается. Другими словами, возрастает время, в течение которого серверы должны быть доступны для обслужива-

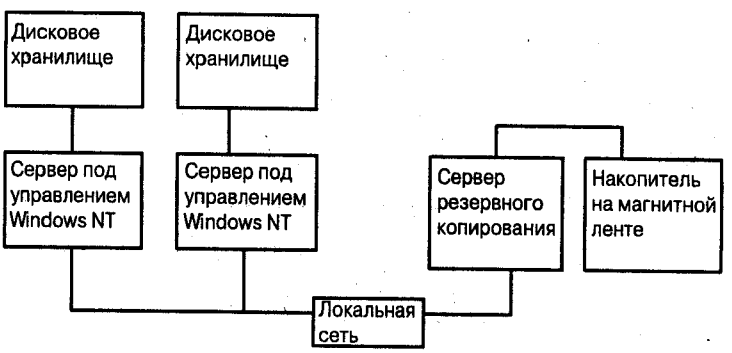


Рис. 5.4. Схема резервирования NAS

ния пользовательских запросов и транзакций. Кроме того, увеличивается объем данных, хранящихся на сервере, что требует большего времени на резервирование этих данных.

Учитывая актуальность описанных проблем, обеспечение эффективности резервного копирования становится единственным критерием при проектировании сетей и определении точного количества необходимых устройств резервирования.

5.3.3.3 Резервирование SAN

Развитие сетей хранения данных привело к появлению новых концепций резервного копирования. Новые возможности основаны та том, что сеть хранения данных может обеспечить достаточную пропускную способность между любыми двумя устройствами и, в зависимости от топологии, способна предоставить одновременную связь с малыми задержками между несколькими парами устройств. С другой стороны, использование топологии кольца Fibre Channel с количеством устройств больше 30 не дает возможности создавать несколько соединений с высокой пропускной способностью и малыми задержками, так как общая пропускная способность кольца будет совместно разделена между всеми подключенными устройствами.

На рис. 5.5 представлена архитектура типичного приложения SAN для резервного копирования. Обратите внимание на мост Fibre Channel. Большинство накопителей на магнитной ленте не поддерживают интерфейс Fibre Channel (они используют параллельный интерфейс SCSI), поэтому для подключения таких устройств понадобится мост. На рис. 5.5 серверы Windows NT подключены одновременно к локальной сети и к сети хранения данных.

Топология резервного копирования (см. рис. 5.5) имеет ряд преимуществ.

- Накопитель на магнитной ленте может находиться довольно далеко от сервера, данные которого резервируются. Такие накопители обычно оснащены интерфейсом SCSI, хотя в последнее время все чаще появляются накопители с интерфейсом Fibre Channel. Это означает, что их можно подключать только к одной шине SCSI, в результате чего усложняется совместное использование накопителя несколькими серверами. Сети хранения данных на основе Fibre Channel благодаря поддержке различных устройств позволяют успешно решать проблемы совместного использования. Обратите внимание: при этом все равно требуется метод, обеспечивающий корректный доступ к накопителю на магнитной ленте с использованием соответствующих разрешений. Примеры подобных методов представлены ниже.

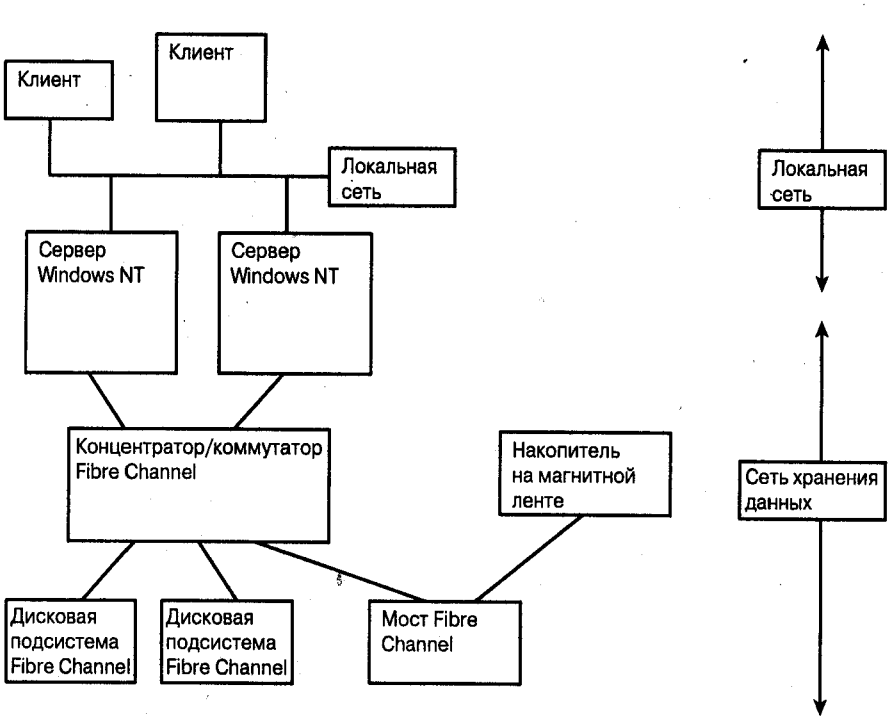


Рис. 5.5. Резервное копирование средствами сети хранения данных

- Метод зонирования позволяет в определенный момент времени получить доступ к накопителю на магнитной ленте одному серверу. Проблема заключается в обеспечении соответствия серверов требованиям зонирования. Кроме того, необходимо обеспечить корректное использование сменщика лент или накопителя с поддержкой нескольких кассет.
- Следующий метод — использование таких команд интерфейса SCSI, как `Reserve` и `Release`.
- Метод подключения накопителя на магнитной ленте к серверу позволяет получить совместный доступ к устройству посредством специального программного обеспечения сервера. Совместное использование накопителя на магнитной ленте является весьма привлекательным решением, поскольку накопители — довольно дорогие устройства. К описанным накопителям относится, например, устройство Tivoli от компании IBM.

■ Технология резервного копирования без локальной сети получила свое название потому, что передача данных выполняется за пределами локальной сети средствами SAN. Это снижает нагрузку на локальную

сеть, благодаря чему приложения не страдают от снижения пропускной способности сети при резервировании данных.

- Резервное копирование без локальной сети позволяет более эффективно использовать ресурсы с помощью совместного использования накопителей на магнитной ленте.
- Резервное копирование и восстановление данных без локальной сети более устойчиво к ошибкам, поскольку резервирование может проводиться несколькими устройствами одновременно, если одно устройство отказало в работе. Аналогичным образом несколько устройств могут использоваться при восстановлении данных, что позволяет эффективнее планировать использование ресурсов.
- Наконец, операции резервного копирования и восстановления завершаются значительно быстрее, так как сети хранения данных обеспечивают более высокую скорость передачи данных.

5.3.3.4 Резервирование, не зависящее от сервера

Такое резервное копирование иногда называют *резервным копированием без сервера* или даже *сторонним копированием*. Обратите внимание, что резервное копирование, не зависящее от сервера, обычно представляет собой резервирование, не зависящее от локальной сети, что избавляет от необходимости перемещать данные с определенного узла. Идея такого способа резервного копирования состоит в применении команды SCSI `Extended Copy`.

В основе резервного копирования, не зависящего от сервера, лежит инициатива ассоциации SNIA, которая была реализована в командах SCSI `Extended Copy`, утвержденных комитетом INCITS, а точнее, техническим подкомитетом T10 (документ ANSI INCITS.351:2001, SCSI Primary Commands-2). Обратите внимание: в стандарте SCSI уже описывалась поддержка команд копирования, однако ранее для использования команд требовалось подключение всех устройств SCSI к одной шине (с тех пор команда `Copy` считается устаревшей; более подробная информация представлена на Web-узле `http://www.t10.org`). Команда `Extended Copy` добавляет такие дополнительные возможности, как использование источника и пункта назначения данных через различные шины SCSI. При этом в полной мере сохраняется адресация, поддерживаемая синтаксисом команды.

В резервном копировании, не зависящем от сервера, сервер резервирования может обрабатывать другие запросы, пока данные копируются с помощью агента перемещения данных. Данные переносятся непосредственно от источника данных в точку назначения, а именно в резервный носитель (вместо копирования из источника на сервер резервного копирования с последующим переносом на резервный носитель).

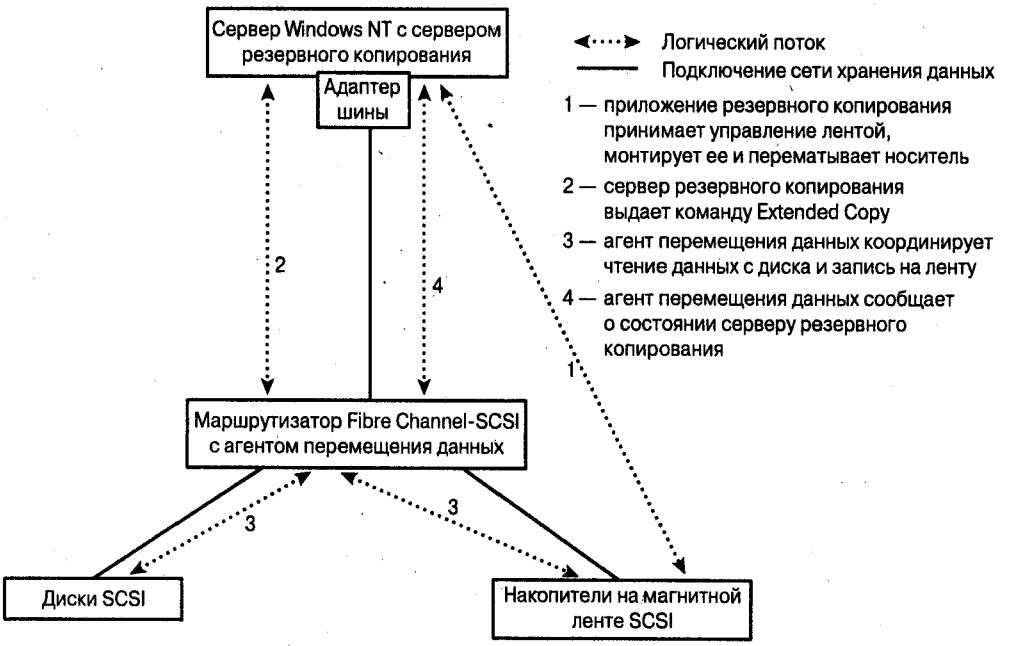


Рис. 5.6. Резервное копирование, не зависящее от сервера

Осознавая преимущества резервного копирования, не зависящего от сервера, не следует забывать, что восстановление данных представляет собой совершенно другую проблему. Операции восстановления, не зависящие от сервера, остаются крайне редким явлением. Резервные копии, созданные с помощью этой технологии, очень часто восстанавливаются традиционными методами, в которых подразумевается использование сервера с неким программным обеспечением для резервного копирования и восстановления данных.

Принцип резервного копирования, не зависящего от сервера, демонстрируется на рис. 5.6. Для упрощения схемы на рисунке показано минимальное количество компонентов, необходимых для иллюстрации резервного копирования. На практике сети хранения данных имеют более сложную структуру. На рис. 5.6 показан сервер под управлением Windows, подключенный к коммутатору Fibre Channel с помощью адаптера шины Fibre Channel. Кроме того, используется маршрутизатор Fibre Channel-к-SCSI, к которому подключается накопитель на магнитной ленте с интерфейсом SCSI и дисковые устройства. Дисковые и ленточные устройства не обязательно должны подключаться к одному маршрутизатору.

Приложение сервера резервного копирования на сервере Windows находит агента перемещения данных на маршрутизаторе с помощью технологии Plug and Play. Приложение резервного копирования определяет дополнитель-

ную информацию о резервировании (идентификатор дискового устройства, начальный логический блок, объем копируемых данных и т.д.). Программное обеспечение сервера резервирования изначально передает последовательность команд накопителю на магнитной ленте для резервирования устройства и монтирования необходимого носителя. Далее программное обеспечение сервера резервного копирования передает команду Extended Copy агенту перемещения данных, который выполняется на маршрутизаторе. Агент координирует перенос необходимых данных. По завершении копирования агент возвращает сервисную информацию программе резервирования, выполняемой на сервере Windows.

В процессе резервного копирования, не зависящего от сервера, важную роль играют несколько компонентов, включая источник и точку назначения данных, агент перемещения и сервер резервного копирования.

Источник данных — это устройство, содержащее данные, для которых необходимо создать резервную копию. Обычно выполняется резервное копирование целого тома или дискового раздела. К источнику данных должен получать доступ непосредственно агент перемещения данных (о нем идет речь несколько ниже). Это означает, что устройства хранения, подключенные к серверу, не могут быть источниками данных для резервного копирования, не зависящего от сервера, так как прямая адресация вне сервера невозможна.

Точка назначения данных обычно представляет собой накопитель на магнитной ленте, на который записываются данные. В качестве устройства может выступать диск, если резервное копирование выполняется на диск, а не на ленту. Ленточные устройства обычно подключены к порту связной архитектуры, чтобы избежать повреждения данных, передаваемых на ленту, в случае отказа других частей сети хранения данных. Например, если накопитель на магнитной ленте подключен к кольцу Fibre Channel с разделением доступа, ошибка в работе другого устройства или подключение/отключение устройства от кольца может привести к остановке записи данных и повторной инициализации кольца, что нарушит целостность данных, записываемых на ленту.

Агент перемещения данных обычно встраивается в маршрутизатор с помощью прошивки, так как он должен обрабатывать команду SCSI Extended Copy, которая отправляется маршрутизатору в виде пакета Fibre Channel. Коммутаторы и концентраторы, обрабатывающие только заголовок кадра Fibre Channel, не совсем подходят для поддержки работы агента перемещения данных, однако в будущем это может измениться.

Агент перемещения данных активизируется после получения инструкций от сервера резервного копирования. Большинство накопителей на магнитной ленте, подключенных к SAN, представляют собой устройства SCSI. Поэтому требуется наличие маршрутизатора, который поддерживает преобразова-

ние пакетов между интерфейсами Fibre Channel и SCSI. На данный момент все чаще появляются накопители на магнитной ленте с интерфейсом Fibre Channel, а некоторые компании, например Exabyte, предоставляют прошивки для подобных накопителей, добавляющие функции агента перемещения данных. Кроме того, базовые библиотеки накопителей на магнитной ленте с интерфейсом Fibre Channel обычно имеют встроенные маршрутизаторы Fibre Channel-SCSI, что позволяет библиотеке использовать собственный агент перемещения данных. Обратите внимание, что агент может быть реализован в программном обеспечении младшей рабочей станции или даже сервера. Компании Crossroads, Pathlight (теперь ADIC) и Chaparral предоставляют маршрутизаторы со встроенными в прошивку агентами перемещения данных. Сеть хранения данных может иметь несколько агентов от нескольких производителей, что не мешает агентам сосуществовать в одной сети.

Конечно, для того чтобы агент перемещения данных можно было использовать, его нужно найти (с помощью команды SCSI `Report LUNs`) и обеспечить должную адресацию (посредством имени WWN) с сервера резервного копирования. Кроме того, агент может проводить два резервных копирования одновременно. Например, один сеанс копирования может проводиться на географически удаленный зеркальный ресурс, однако для этого сервер резервирования должен передать две команды.

Сервер резервного копирования отвечает за все команды и управление операциями. Перечислим еще раз все основные обязанности сервера резервирования.

- Программное обеспечение сервера обеспечивает доступность накопителя на магнитной ленте, применяя соответствующие команды SCSI `Reserve` и `Release`.
- Монтирование носителя для резервного копирования.
- Определение точного адреса источника данных и размещения данных в логических блоках, а также объема данных для резервирования.

Получив всю необходимую информацию, сервер отправляет команду `Extended Copy` агенту перемещения данных. Затем агент отправляет последовательность команд `Read` источнику данных и записывает информацию в точке назначения.

Компании Computer Associates, CommVault, LEGATO и VERITAS предоставляют программы для резервирования, не зависящего от сервера. Поставщики маршрутизаторов с функциями резервного копирования, не зависящего от сервера, постоянно сотрудничают с компаниями — разработчиками программного обеспечения, чтобы сделать возможной совместимость своих продуктов. Дело в том, что для поддержки базовых команд SCSI `Extended Copy` производителями применяются различные команды.

Обратите внимание: несмотря на достаточно зрелый возраст технологии резервирования, не зависящей от сервера, поддержка восстановления, не зависящего от сервера, со стороны производителей крайне ограниченна.

5.3.3.5 Семейство операционных систем Windows Server и резервное копирование, не зависящее от сервера

В многочисленных рекламных материалах и маркетинговой литературе утверждается, что конкретный метод внедрения технологии резервного копирования, не зависящего от сервера, совместим с Windows 2000. Рассмотрим эту концепцию более подробно. Далее описывается каждый из четырех компонентов, формирующих резервирование, не зависящее от сервера: источник данных, точка назначения данных, программное обеспечение сервера резервирования и агент перемещения данных.

В большинстве случаев агент перемещения данных, работающий вне сервера Windows NT, не может адресовать данные, хранящиеся на сервере Windows NT. Адаптеры шины, подключенные к серверу Windows NT, обычно работают, как инициаторы и не отвечают на команды `Report LUNs`. Если сервер Windows NT использует устройство хранения за пределами сервера, например массив RAID, подключенный к коммутатору Fibre Channel, то это устройство будет доступно агенту перемещения. Поэтому вместо утверждений о том, что устройство хранения, используемое Windows NT, не может быть источником данных для резервирования, не зависящего от сервера, следует уточнить, что источником данных не может быть устройство хранения, которое является внутренним для сервера Windows NT.

Использование внутреннего хранилища Windows NT в качестве точки назначения данных также невозможно, так как точка назначения тоже должна быть доступна агенту перемещения данных для адресации.

Выполнение программы резервирования на компьютере под управлением Windows представляет собой неплохой вариант. Адаптер шины, подключенный к серверу Windows, может выдать последовательность команд `Report LUNs` каждому устройству (LUN 0), которое будет обнаружено. Затем программа резервирования просматривает все видимые устройства и логические единицы, после чего выясняет, какие из них могут выступать в роли агента стороннего копирования. Некоторые программы сообщают о дополнительных LUN, которые необходимы при выдаче команд `Extended Copy`. Множество программ резервирования, которые используют дополнительные LUN, проходят через процесс обнаружения устройств для проверки функций агента перемещения данных.

Промежуточный интерфейс SCSI (IOCTL) в Windows NT может использоваться для передачи команды `Extended Copy` агенту перемещения данных

(команда передается с сервера резервного копирования под управлением Windows NT). Операционная система Windows NT не имеет встроенной поддержки агентов перемещения; технология Plug and Play позволяет обнаружить агент, но для регистрации последнего в системном реестре необходимы дополнительные драйверы.

Остается последний вопрос: можно ли запустить программное обеспечение агента перемещения данных на сервере или рабочей станции под управлением Windows NT? Одним из преимуществ такого решения является то, что агент перемещения сможет адресовать устройства хранения, "видимые" для сервера Windows, а также получать к ним доступ. Но сервер резервного копирования, размещенный вне Windows NT, не сможет обнаружить устройства хранения, подключенные к компьютеру с агентом перемещения данных. Агент должен иметь возможность работать в качестве инициатора и целевого устройства для команд SCSI. Поскольку адаптер шины, подключенный к компьютеру под управлением Windows NT, редко выполняет роль целевого устройства, команда `Extended Copy` может не дойти до агента перемещения данных.

Обратите внимание: в Windows NT для выдачи команд SCSI приложения используют промежуточный интерфейс (`DeviceIoControl` с параметром `IoControlCode`, равным `IOCTOL_SCSI_PASS__THROUGH` или `IOCTL_SCSI_PASS_THROUGH_DIRECT`).

5.4 Утилита резервного копирования Windows 2000

В составе Windows 2000 поставляется программа резервного копирования, которая на самом деле представляет собой облегченную версию программы VERITAS Backup Exec. Встроенная утилита резервного копирования интегрирована с другими компонентами операционной системы, например с шифрованной файловой системой и интерфейсом управления иерархическим хранилищем. Утилита резервного копирования поддерживает резервное копирование и восстановление EFS в Windows 2000. Более подробная информация представлена в главе 6. Кроме того, утилита поддерживает работу с менеджером RSM — Removable Storage Manager (см. главу 7). Менеджер предоставляет поддержку операций, необходимых для резервного копирования:

- инвентаризацию носителей, загруженных в библиотеку ленточных библиотек;
- загрузку и извлечение носителей из ленточных библиотек;
- предоставление безопасного доступа и обеспечение целостности данных на смонтированном носителе;

- выполнение обслуживающих функций для управления носителями и ленточной библиотекой, например очистка ленточного привода или библиотеки носителей.

Полнофункциональные утилиты резервного копирования предоставляют возможности, которые не поддерживаются описываемой утилитой Windows 2000. Вот некоторые из этих возможностей:

- агенты резервного копирования для приложений уровня предприятия, например серверов SQL и IIS;
- поддержка резервного копирования открытых файлов;
- повышенное быстродействие;
- централизованное управление, включая централизованную базу данных с каталогом и управляющим программным обеспечением для всех устройств и каталогов резервного копирования;
- поддержка команды `Extended Copy` и агентов перемещения данных других поставщиков.

Обратите внимание, что утилита резервного копирования в Windows Server 2003 поддерживает копирование открытых файлов, так как резервирование выполняется на основе моментальных снимков.

5.5 Технологии создания моментальных снимков тома

Моментальный снимок (snapshot) — это целостная копия состояния тома в определенный момент времени. Под целостностью в данном случае подразумевается возможность для приложений обработать данные, хранящиеся в снимке; например, сервер Microsoft Exchange воспринимает данные как действительное хранилище Exchange, а сервер Microsoft SQL — как реальную базу данных SQL. Рассматриваемый в этом случае набор данных в дальнейшем будет именоваться логическим томом.

Моментальные снимки набирают популярность и используются по ряду различных причин.

- Резервное копирование клонированного тома, созданного с помощью технологии моментального снимка, в то время пока основной том используется приложениями. Именно для этого компания Microsoft создала службу теневого копирования томов в Windows XP. Служба (в определенный момент времени) используется для создания образа оригинального тома (при наличии свободного дискового пространства). Этот образ применяется затем для операций резервного копирования.

- Создание образа активных данных, в котором поддерживается поиск.
- Создание образа активных данных для тестирования новой версии приложения в "реальном" окружении.
- Создание образа активных данных для последующего восстановления системы после сбоев в работе.

Как же создается моментальный снимок тома? Для этого существует несколько способов, каждый из которых представляет собой вариант дублирования операций записи. Единственное отличие между ними состоит в том, где именно проводится дублирование. Существует четыре возможных уровня.

1. *Уровень аппаратного обеспечения.* Первый очевидный способ создания моментального снимка тома заключается в зеркальном отражении тома на уровне аппаратного обеспечения с последующим разделением зеркала. При использовании простого аппаратного обеспечения и программных томов, каждая операция записи преобразуется в две операции: для оригинального тома и для зеркала. Этот подход требует большого объема ресурсов, так как обе операции записи должны быть завершены до отправки результата записи. Кроме того, необходимо обеспечить обработку ошибок записи. Преимуществом такого подхода, несмотря на требовательность к ресурсам, является значительная скорость разбития зеркала для создания моментального снимка тома. Высокая скорость и надежность зеркального отражения с помощью аппаратного обеспечения связана с определенными накладными расходами: основную часть составляет стоимость жестких дисков для дублирования записи. При использовании дорогих подсистем хранения для каждой дорожки хранится большой объем метаданных, так как вместо разделения самих записей, после разбивки зеркала записи отслеживаются через метаданные. Более того, дорогие подсистемы хранения не задерживают зеркальные записи, так как записи считаются завершенными, как только необходимые данные попадают в кэш подсистемы хранения.
2. *Уровень над файловой системой.* Второй способ создания моментального снимка тома в операционных системах семейства Windows Server заключается в создании драйвера фильтрации файловой системы, который размещен над драйвером файловой системы, например NTFS или FAT. Драйвер фильтрации файловой системы дублирует каждый пакет IRP (пакет запроса ввода-вывода), который передается драйверу файловой системы. Этот процесс довольно сложен и не может использовать преимущества технологии моментальных снимков, которая предоставляется аппаратным обеспечением. В качестве примера программ, внедряющих фильтры над NTFS, можно указать Open File Manager компании St. Bernard, Vinca (теперь LEGATO) Open File Manager и Open File

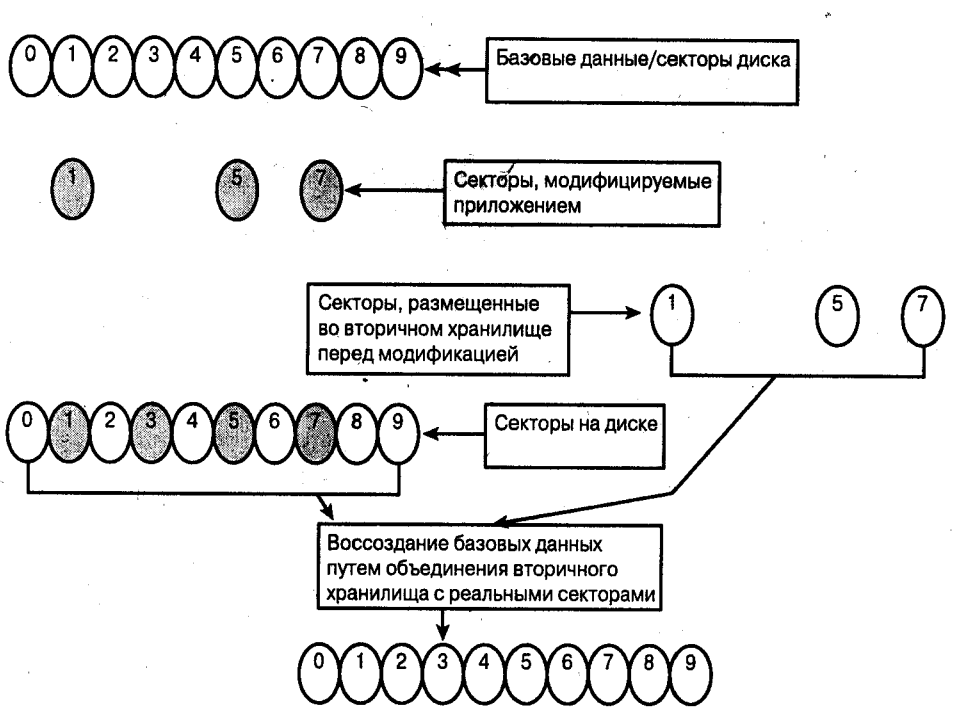


Рис. 5.7. Моментальные снимки по методу копирования при записи

Agent компании Cheyenne. Обратите внимание, что не каждая программа поддерживает технологию моментальных снимков.

3. *Уровень непосредственно файловой системы.* Переместившись ниже по стеку драйверов, можно реализовать третий способ создания моментальных снимков на уровне файловой системы, такой, как WAFL (Write Anywhere File Layout) и Linux SnapFS. Очевидно, что в данном случае речь не идет о Windows NT. Создание файловой системы — достаточно сложный процесс, а создание технологии моментальных снимков на уровне файловой системы еще больше его усложняет. В результате файловые системы, реализованные в Windows NT, не поддерживают моментальных снимков.

4. *Уровень ниже файловой системы.* Четвертый способ — использование драйвера фильтрации ниже файловой системы для реализации технологии копирования при записи. Основная идея состоит в том, чтобы логические блоки, в которые приложение вносит изменения, сначала записывались во вторичное хранилище и только потом — на жесткие диски. Этот принцип демонстрируется на рис. 5.7. Описываемая технология также называется *дифференциальным моментальным снимком*,

поскольку сохраняются только измененные файлы (полное зеркальное отражение тома не проводится).

5.6 Служба теневого копирования томов в Windows XP и Windows Server 2003

В Windows XP и Windows Server 2003 компания Microsoft реализовала службу теневого копирования. Таким образом, предоставляется инфраструктура, позволяющая создавать целостные копии дисковых томов в заранее определенный момент времени. По юридическим причинам Microsoft решила назвать эту технологию *теневым копированием томов* (volume shadow copy), что на самом деле не отличается от более популярного термина — *моментальные снимки*. Служба теневого копирования томов реализована в драйвере, который называется `volsnap.sys` и находится ниже уровня файловой системы.

Компания Microsoft предоставляет инструментарий разработки программного обеспечения для теневого копирования томов, реализуемый на базе договора о неразглашении. Набор SDK в основном предназначен для представителей трех обширных аудиторий.

1. Независимые поставщики программного обеспечения, предназначенного для теневого копирования томов, включая Microsoft Exchange, SQL Server, Oracle, SAP, Sybase и др.
2. Независимые поставщики программного обеспечения, разрабатывающие приложения резервного копирования и управления подсистемами хранения. Такие поставщики могут создавать программное обеспечение для отправки запросов службе теневого копирования томов.
3. Независимые поставщики программного и аппаратного обеспечения, предназначенного для резервного копирования, восстановления после ошибок и обеспечения целостности данных. В качестве примера можно привести компании VERITAS, EMC и их конкурентов в этой отрасли. Такие поставщики разрабатывают программное обеспечение для предоставления моментальных снимков.

Если приложение не имеет кода, поддерживающего службу моментальных снимков, данные приложения все равно будут скопированы с сохранением целостности, как в том случае, если файловая система не может штатным образом завершить свою работу. Если приложение содержит код для поддержки службы моментальных снимков, от приложения ожидается поддержка службы и в операциях восстановления данных. Приложение выдает некоторые данные (например, какие файлы необходимо копировать, а также

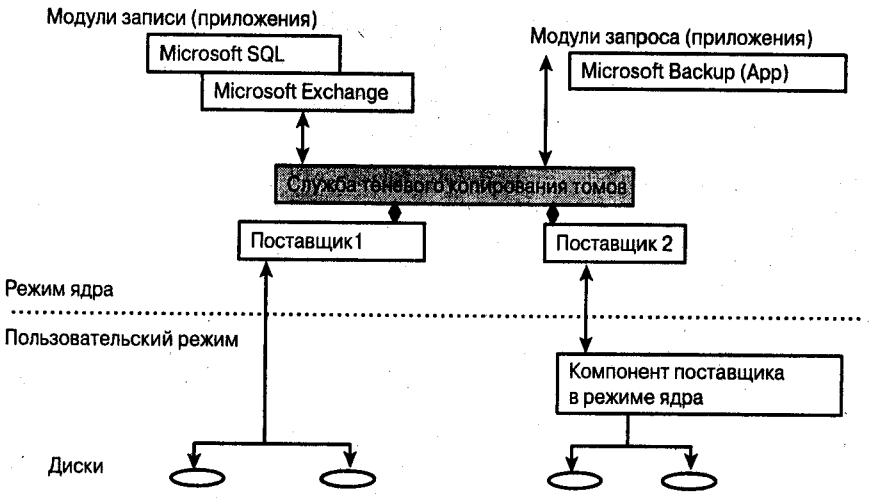


Рис. 5.8. Архитектура теневого копирования томов

методологию резервного копирования и восстановления) при запросе резервного копирования. От приложения также ожидается обработка этих данных, когда потребуется операция восстановления.

Резервное копирование и восстановление данных, реализованное в операционных системах Windows, не всегда работает со стопроцентной надежностью. Новая служба моментальных снимков позволяет достичь максимальной надежности, поддерживая более сложные сценарии, которые до этого момента были недоступны в Windows.

Архитектура теневого копирования томов в Windows XP и Windows Server 2003 включает в себя четыре типа модулей (рис. 5.8).

1. Модули записи.
2. Модули запроса.
3. Служба теневого копирования томов.
4. Поставщики.

Эти модули подробно рассматриваются в разделах 5.6.1–5.6.4.

В контексте различных модулей, формирующих службу моментальных снимков, второй поставщик моментальных снимков снабжен компонентом для режима ядра, который отсутствует у первого поставщика. Служба поставщика моментальных снимков на рис. 5.8 выделена серым цветом, что подчеркивает предоставление компанией Microsoft функции, которую остальные поставщики программного обеспечения не пожелали предоставлять в виде службы. Кроме того, Microsoft предлагает и другие компоненты, но их функциональность ограничена определенным приложением или набором опреде-

ленных функций; таким образом, от поставщиков ожидается создание компонентов средствами SDK.

Модули записи и поставщики должны внедрить отдельного внепроцессного поставщика COM, как описано в SDK[1], для теневого копирования томов. Поставщики обычно реализуются в виде "конечного автомата". Автомат переходит из одного состояния в другое при получении события, сгенерированного службой теневого копирования. Поставщик получит событие (сгенерированное службой теневого копирования), однако тип события зависит от текущего статуса поставщика и от наличия ошибок в работе. Другими словами, поставщик ожидает получения предпочтительного события, которое позволит ему перейти в следующее нормальное состояние. В случае ошибки поставщик получит событие, которое отличается от ожидаемого. Тем не менее такое событие также может быть обработано поставщиком.

Инфраструктура теневого копирования в Windows XP и Windows Server 2003 предоставляет базовые функции, необходимые для управления подсистемой хранения данных, включая следующие:

- определение момента времени, в который должен создаваться моментальный снимок;
- предоставление служб синхронизации для приложений, баз данных, операционных систем и файловых систем для чистки кэшированных данных, приостановки записи, создания моментальных снимков и предоставления уведомлений приложениям, базам данных и элементам операционных систем о возможности продолжения нормального режима работы;
- предоставление единого API, который может использоваться во время операций резервного копирования и восстановления;
- предоставление общей платформы для управления моментальными снимками.

Инфраструктура теневого копирования Microsoft поддерживает обработку набора томов, для которых моментальный снимок должен быть сделан

[1] Для тех читателей, кто незнаком с термином COM, стоит упомянуть, что это чрезвычайно важная архитектура Windows, позволяющая создавать программы в виде взаимодействующих друг с другом объектов. Объекты COM можно написать с помощью различных языков программирования, в том числе C, C++ и Visual Basic. Два объекта COM могут быть расположены на одной локальной системе или одновременно выполняться на двух удаленных системах. Два объекта могут выполняться в контексте одного процесса (в локальной системе) или же в двух отдельных процессах. В последнем случае предоставляются специальные механизмы для обеспечения взаимодействия этих объектов. Таким образом, внепроцессный поставщик — это поставщик, реализованный в виде объекта COM и выполняемый в отдельном контексте процесса.

как для одного тома. Если одна операция завершится неудачно, неудача постигнет и все другие операции. Кроме того, инфраструктура теневого копирования Microsoft выдает запрос (поставщику моментального снимка) на удаление моментального снимка, когда запрашивающее приложение завершило его обработку. Если необходимо, чтобы моментальный снимок оставался доступным для последующего использования, поставщик моментальных снимков или запрашивающее приложение должны предоставить необходимые функции. Независимые поставщики программного обеспечения разрабатывают приложения, которые на основе архитектуры теневого копирования создают и каталогизируют несколько моментальных снимков, а также управляют ими; такие программы не поставляются в комплекте с Windows Server 2003.

5.6.1 Модули записи

Модули записи моментальных снимков представляют собой приложения, которые записывают данные. К модулям записи относятся программы Microsoft Exchange, Microsoft SQL Server 2000, SAP и Oracle. Компания Microsoft и независимые поставщики программного обеспечения разрабатывают системы, поддерживающие запись моментальных снимков. При этом модули записи моментальных снимков должны быть реализованы с помощью набора SDK. В частности, модули записи получают от службы моментальных снимков два события, в результате чего приложения прекращают запись данных на диск, а также отдельное событие, позволяющее продолжить запись (это событие указывает на успешное создание моментального снимка). Существуют и другие события, которые может генерировать служба создания моментальных снимков. Дополнительная информация об этих событиях доступна в наборе SDK. Так как приложения могут определять целостность получаемых данных, операции сохранения должны проводиться достаточно быстро.

Необходимо также отметить одно важное преимущество архитектуры службы теневого копирования томов по сравнению с традиционными механизмами создания моментальных снимков, основанными на аппаратных решениях. При использовании в Windows 2000 и более ранних версиях традиционных механизмов аппаратное обеспечение не имело возможности "узнать" состояние приложений, программного обеспечения операционной системы и содержимого кэша. Это означает, что целостность заметного процента моментальных снимков нарушена. Более того, единственным способом определения целостности моментального снимка был запуск проверки целостности в конкретном приложении, на что уходило несколько часов.

В то же время при использовании архитектуры теневого копирования томов можно не только аннулировать содержимое кэша и приостановить опе-

рации записи, но и проверить целостность снимка за несколько минут. Как только приложениям выдается событие продолжения записи, служба теневого копирования томов запрашивает у приложений сведения об успешной задержке записи между событиями остановки и продолжения записи. Если один из модулей записи не смог остановить запись данных на диск после прихода соответствующего события, создание моментального снимка можно считать неудачным.

От модулей записи моментальных снимков ожидаются данные, необходимые для резервного копирования и восстановления. К ним относятся, например, список файлов, которые требуется скопировать или исключить из процесса копирования, а также группа объектов, которые должны обрабатываться в виде единого набора. Такие данные сохраняются службой создания моментальных снимков в документе метаданных модуля записи, который имеет формат XML. Модули записи могут использовать документ для хранения других необходимых данных. При восстановлении данных приложение предоставляет набор данных записавшему приложению, которое и проводит операцию восстановления.

Компания Microsoft объявила, что будет предоставлять модули записи для программ SQL Server 2000 и Exchange, а также других компонентов Windows Server. Компания сотрудничает с независимыми поставщиками программного обеспечения для разработки модулей записи к другим приложениям, включая службу каталогов Active Directory.

5.6.2 Модули запроса

Обычно *модуль запроса* — это приложение резервного копирования, которое запрашивает создание моментального снимка с помощью соответствующего вызова API, направленного к службе теневого копирования томов от Microsoft. Такая модель значительно упрощает решение некоторых проблем для создателя программы резервного копирования. Программе более не придется решать сложную задачу поиска копируемых данных или определять файлы журналов приложений, которым требуется специальная обработка.

Соответствующий модуль записи (например, Microsoft SQL Server) отвечает за создание набора файлов, которые будут включены в резервную копию. Операция восстановления также намного упрощается, поскольку приложение восстановления не должно искать данные и определять, какие файлы передавать API приложения (например, Exchange или SQL). Приложение передает набор данных модулю записи (приложению) и предоставляет ему возможность выполнить операцию восстановления.

5.6.3 Служба теневого копирования томов

Созданная компанией Microsoft *служба теневого копирования томов* Windows NT координирует деятельность всех компонентов создания моментальных снимков. В частности, служба предоставляет описанные ниже возможности.

- Единый интерфейс для приложений резервного копирования или модулей запроса создания моментальных снимков. Ранее приложения резервного копирования должны были работать с несколькими API от нескольких приложений.
- Единый интерфейс для создания, управления и удаления целостных моментальных снимков томов или теневых томов.
- Единый интерфейс, который позволяет различным приложениям и поставщикам моментальных снимков регистрироваться и удалять регистрацию модулей записи и поставщиков моментальных снимков.
- Синхронизация и координация различных компонентов для создания, удаления и перемещения моментальных снимков, а также для резервного копирования и восстановления. Служба расставляет приоритеты поставщиков моментальных снимков таким образом: аппаратные поставщики имеют наивысший приоритет, затем идут программные поставщики, а самым низким приоритетом обладают поставщики, которые по умолчанию предоставляются Microsoft.

Независимые поставщики программного обеспечения могут не беспокоиться о необходимости создания службы теневого копирования томов. Службу, предоставленную компанией Microsoft, можно воспринимать, как планировщик печати (spooler). На компьютере должен быть один планировщик печати. Некоторые производители (например, создатели поставщиков) должны написать эквивалент драйвера печати, в то время как остальным достаточно создать приложение печати.

5.6.4 Поставщики

Поставщики моментальных снимков должны разрабатываться независимыми компаниями для создания, удаления и управления моментальными снимками. Как отмечалось ранее в главе, поставщики моментальных снимков должны быть созданы в виде внепроцессных поставщиков COM средствами соответствующего набора SDK.

Поставщик может иметь компонент, работающий в режиме ядра, например драйвер фильтрации, который расположен между файловой системой

и диспетчером логических дисков (logical disk manager — LDM). При необходимости функции режима ядра могут быть реализованы в аппаратном обеспечении. Обратите внимание, что даже аппаратный поставщик будет использовать остальные возможности инфраструктуры, например определение временно́й точки, синхронизацию ввода-вывода и платформу для создания приложений управления подсистемой хранения, включая резервное копирование/восстановление данных и приложения управления моментальными снимками.

Одним из примеров поставщика моментальных снимков служит драйвер `volsnap.sys`, который предоставляется в Windows XP и ожидается в Windows Server 2003. Этот поставщик использует технологию копирования при записи для создания минимального необходимого набора данных во вторичном хранилище, чтобы воссоздать том с определенной временно́й точки. При этом должен быть доступен достаточный объем свободного дискового пространства. Поставщик может обрабатывать тома NTFS, FAT32 и тома без файловой системы в Windows Server 2003. Однако поставщик может создавать моментальные снимки, предназначенные только для чтения, и для каждого тома создается только один моментальный снимок. Это ограничение самого поставщика, а не инфраструктуры, на базе которой он создан. Независимые производители программного и аппаратного обеспечения при желании могут предоставлять более широкие возможности.

Полное описание всех событий, которые получает поставщик моментальных снимков, приводится в документации SDK. Далее рассматривается два наиболее важных события.

- `PreCommitSnapshot`. При получении этого события поставщик должен начать выполнение всех операций ввода-вывода, которые занимают довольно много времени, например синхронизацию зеркала.
- `CommitSnapshot`. С получением этого события поставщик уведомляется о том, что служба моментальных снимков завершит работу через 10 секунд. Таким образом, быстродействие поставщика должно быть достаточно высоким. Более того, пока создание моментального снимка не будет завершено, Windows NT не будет выполнять операции записи на том, для которого создается моментальный снимок. Это значит, что поставщик не должен выполнять операции ввода-вывода с этим томом, а если такая операция выполнена, то она не должна завершаться до тех пор, пока создание моментального снимка не будет завершено или прервано.

Поставщики моментальных снимков должны предоставлять определенные функции, а также возможности, выходящие за рамки обязательных. Ниже описаны обязательные функции.

- Поставщики отвечают за поиск хранилища, необходимого для создания моментального снимка. Инфраструктура Microsoft не обеспечивает подобной возможности.
- Поставщик должен монтировать моментальный снимок в другом пространстве имен й не в виде отдельного тома. Внимательно рассмотрев архитектуру Windows XP, можно отметить, что поставщик моментальных снимков от компании Microsoft монтирует их с использованием адреса `\Device\HarddiskSnapshotX`.

5.6.5 Модификации подсистемы ввода-вывода Windows NT

Хотя модификации подсистемы ввода-вывода Windows NT не обязательно относятся к окружению моментальных снимков, следует обратить внимание, что для создания целостных и надежных моментальных снимков в определенный момент времени требуются значительные доработки файловой системы, стека ввода-вывода и драйверов фильтрации файловой системы. В частности, компания Microsoft добавила два вызова управления вводом-выводом (IOCTL), которые должны быть реализованы во всех файловых системах и драйверах фильтрации файловых систем.

1. Вызов `IOCTL_VOLSNAP_FLUSH_AND_HOLD_WRITES`, который должен входить в последовательность и обрабатываться. Включение в последовательность позволяет обрабатывать вызов драйверам более низкого уровня. Обработка заключается в очистке и удержании всех метаданных файловой системы. Как только все данные будут сброшены, до завершения работы пакетов IRP, предназначенных для аннулирования данных и метаданных, другие запросы на запись выдаваться не должны.
2. Вызов `IOCTL_VOLSNAP_RELEASE_WRITES` также должен обрабатываться и включаться в последовательность. Этот вызов указывает на успешное завершение создания моментального снимка или на прерывание операции создания моментального снимка.

Некоторые компоненты Windows NT также были модифицированы для вызова указанных IOCTL в соответствующие моменты времени. Хотя компания Microsoft уже модифицировала файловую систему и драйвер фильтрации файловой системы для предоставления соответствующей функции, от драйверов фильтрации независимых производителей программного обеспечения ожидается то же самое.

В разделе 5.8 рассматривается промышленный стандарт NDMP (Network Data Management Protocol — сетевой протокол управления данными). Но перед обсуждением этой темы следует рассмотреть взаимосвязь между службой теневого копирования томов Windows XP/Windows Server 2003 и прото-

колом NDMP. Служба теневого копирования используется для клонирования данных, которые необходимо скопировать на резервный носитель, в то время как NDMP применяется для переноса данных с клонированного образа на магнитную ленту или другой резервный носитель.

5.7 Устройства NAS под управлением Windows и моментальные снимки

Компания Microsoft предоставляет версию Windows NT, которую иногда называют Embedded NT (встроенная NT), однако обычно она известна как Server Appliance Kit, или SAK. Эта система основана на ядре Windows 2000, которая не содержит службы теневого копирования томов. Для производителей аппаратного обеспечения, использующих SAK при разработке устройств NAS, компания Microsoft лицензировала технологию создания моментальных снимков и добавила ее в SAK. В последнем случае для создания моментальных снимков используется технология PSM (Persistent Storage Management) от компании Columbia Data Products. В этом разделе приводится краткое описание архитектуры и функций PSM.

Архитектура PSM представлена на рис. 5.9. Модуль PSM включает в себя компонент пользовательского режима, который обеспечивает управление моментальными снимками, включая запуск и планирование снимка. Моментальные снимки создаются с помощью драйверов фильтрации PSM, которые расположены над драйвером класса диска.

Архитектура PSM предоставляет возможность создания нескольких моментальных снимков, а также средства для их управления. Моментальные снимки можно создавать по расписанию, а более старые снимки могут быть сохранены или перезаписаны. Кроме того, старые снимки можно монтировать для использования при резервном копировании или для других целей. С каждым моментальным снимком связана информация о дате и временной отметке.

5.8 Протокол NDMP

В основе протокола NDMP лежит инициатива компаний Network Appliance и Intelliguard (теперь подразделение компании LEGATO). Он предназначался для предоставления дополнительных возможностей приложениям резервного копирования и восстановления данных, а именно:

- возможность сократить или, как минимум, изолировать зависящие от операционной системы элементы программного обеспечения для резерв-

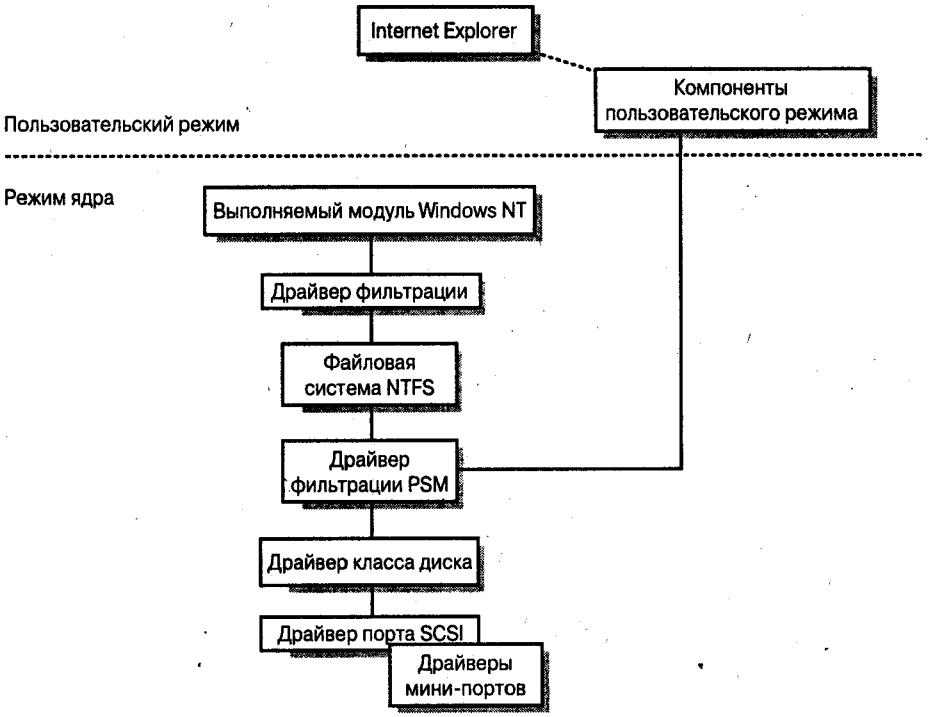


Рис. 5.9. Архитектура Persistent Storage Manager

ного копирования и восстановления; функции приложений предлагалось разделить на модульной основе;

- стандартизированные способы связи между модулями;
- возможность разделить перенос данных и команд на отдельные каналы или даже сети;
- интеграция программного обеспечения от разных производителей.

На данный момент NDMP позиционируется как открытый стандарт для выполнения операций резервного копирования и восстановления в среде NAS. Протокол постоянно развивается и в будущем может обеспечить преобразование копируемых данных стороннего ресурса в данные протокола NDMP.

Протокол NDMP может использоваться в различных сетевых средах, например Ethernet, Gigabit Ethernet, Fibre Channel. Главным условием является использование IP (Internet Protocol) в качестве транспортного протокола. Сервер NDMP и агенты используют протокол IP, а сервер NDMP выдает необходимые команды SCSI блочного уровня.

Протокол NDMP ориентирован на устройства NAS и определяет метод резервного копирования и восстановления данных с устройства, например

системы NAS, на которой сложно установить агент резервного копирования. При отсутствии NDMP данные копируются с общего диска по протоколам CIFS и NFS.

Протокол NDMP имеет ряд преимуществ.

- Интерфейсы предоставляются производителями, которые имеют доступ к ядру системы.
- Интерфейсы стандартизированы и позволяют использовать Plug and Play для модулей разных производителей.
- Протокол NDMP может сократить использование пропускной способности сети, обеспечивая передачу данных непосредственно между первичным и вторичным носителями, не требуя отправки данных сначала на сервер резервного копирования, а затем на вторичный носитель.
- Протокол NDMP совмещает в себе лучшие качества обоих "миров", так как поддерживает удаленное управление через сеанс управления NDMP, а данные могут передаваться локально через сеансы данных NDMP.

5.8.1 Архитектура NDMP

Посредством протокола NDMP определяется стандартизированный способ разбивки операций резервного копирования и восстановления на несколько томов, с учетом того, что каждый производитель предоставляет отдельные модули. Протокол NDMP включает в себя следующие компоненты:

- агент перемещения данных;
- службы NDMP;
- сеансы NDMP.

Рассмотрим эти компоненты подробнее.

5.8.1.1 Агент перемещения данных

Агент перемещения данных (data mover agent — DMA) представляет собой основное приложение резервного копирования. Он устанавливает сеансы NDMP с поставщиком услуг NDMP (рассматривается далее) и управляет последовательностью действий, необходимых для настройки операции резервного копирования или восстановления. Кроме того, иногда агент перемещения данных называется *клиентом NDMP*.

5.8.1.2 Службы NDMP

Протокол NDMP поддерживает службы, которые могут выступать в роли абонентов или генераторов и позволяют одному устройству быть и абонентом

и генератором потоков данных. Протоколом NDMP версии 5 поддерживается три типа служб.

1. *Служба данных* взаимодействует с первичным устройством хранения (например, с устройством NAS). Эта служба работает с томом или файловой системой, которая копируется на резервный носитель или восстанавливается.
2. *Служба магнитных накопителей* взаимодействует со вторичным устройством хранения, как правило, накопителем на магнитной ленте.
3. *Служба трансляции* выполняет преобразование данных, включая мультиплексирование нескольких потоков данных в один поток и наоборот.

5.8.1.3 Сеансы NDMP

Службы NDMP взаимодействуют друг с другом с помощью интерфейсов NDMP. В результате устанавливается сеанс NDMP, который называется *управляющим сеансом*, если используется для получения управления над операцией резервного копирования или восстановления, либо *сеансом данных*, если используется для передачи файловой системы или данных тома (включая метаданные). Между каждой службой NDMP и агентом перемещения данных существует только один сеанс. Сеансы управления всегда создаются на основе протокола TCP/IP; потоки данных могут передаваться по протоколу TCP/IP или по сетям хранения данных. Даже несмотря на возможность передачи данных по сетям хранения данных, использование протокола TCP/IP для сеансов управления требует наличия локальной сети. Потоки данных могут проходить между агентом перемещения данных и службой NDMP или непосредственно между двумя службами NDMP.

Агент перемещения данных и службы могут быть распределены на два или более компьютеров. На рис. 5.10 показан NDMP и работа двух компьютеров. Агент перемещения данных NDMP связывается с сервером NDMP и управляет перемещением данных с первичного на вторичный носитель. Сеансы данных устанавливаются между сервером NDMP, источником данных и точкой назначения данных.

На рис. 5.11 представлена концептуальная реализация протокола NDMP в Windows NT. Агент перемещения данных NDMP устанавливает два сеанса управления NDMP, по одному с каждым сервером NDMP. Данные передаются непосредственно между двумя серверами NDMP, а не "перетаскиваются" с одного сервера NDMP к агенту перемещения данных NDMP и оттуда к другому серверу NDMP.

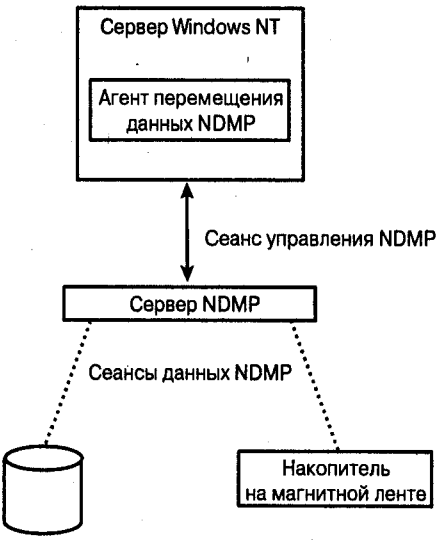


Рис. 5.10. Архитектура NDMP

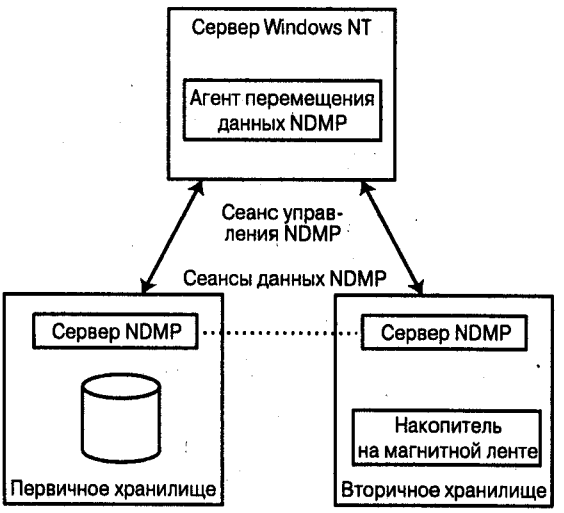


Рис. 5.11. Концептуальная реализация NDMP в Windows NT

5.9 Сложности практической реализации

Дорогостоящие подсистемы хранения, которые могут отслеживать метаданные для каждого сектора, позволяют намного эффективнее выполнять операции резервного копирования и восстановления, так как изменения от-

слеживаются очень точно и операции резервного копирования затрагивают изменившиеся данные.

Для обеспечения целостности моментального снимка в контексте приложения важно, чтобы операционная система (включая файловые системы) и приложение принимали участие в чистке кэша и временной приостановке операций записи на время создания моментального снимка. Служба теневого копирования томов, которая поставляется в составе Windows Server 2003, обеспечивает поддержку со стороны операционной системы и файловых систем, а также делает доступной архитектуру поддержки со стороны приложений. Эта архитектура используется такими важными приложениями, как Microsoft Exchange и Microsoft SQL Server.

Остается подождать и посмотреть, как остальные поставщики программного обеспечения будут поддерживать архитектуру теневого копирования томов.

5.10 Резюме

Операции резервного копирования эволюционировали с точки зрения требований пользователей и технологий, применяемых для резервирования данных. В последнее время требуется все более частое резервирование информации, причем без нарушения доступа приложений к диску. Операции резервного копирования прошли путь от изолированного резервного копирования до резервного копирования, выполняемого по локальной сети и в сетях хранения данных. Одной из проблем, решение которой требовалось от приложений резервного копирования, было копирование открытых файлов, доступ к которым осуществляется работающими приложениями.

Кроме того, приложения для резервного копирования должны обрабатывать множество API, специфических для различных версий приложений и операционных систем. Еще одна тенденция состоит в резервировании "с диска на диск" с помощью операции создания моментального снимка. Резервное копирование на магнитную ленту превращается во вторичную операцию, которая подразумевает копирование данных с моментального снимка тома.

Служба теневого копирования томов Windows предоставляет эффективный способ создания моментальных снимков. Архитектура содержит все необходимые компоненты, включая основные приложения, например базы данных и серверы передачи сообщений, которые принимают участие в создании моментальных снимков. Компания Microsoft только предоставляет инфраструктуру для создания моментальных снимков. Производители программного обеспечения могут использовать эту инфраструктуру для генерации

приложений, поддерживающих создание нескольких моментальных снимков и управление ими.

Созданный моментальный снимок можно использовать для резервного копирования данных. Чтобы выполнить резервирование, можно воспользоваться стандартным протоколом, например NDMP.

Глава 6

Файловые системы

Файловая система обеспечивает работу важнейших функций; основные из них перечислены ниже.

- Поддержка целостности данных и предоставление пользователю необходимых возможностей для создания, удаления, чтения и записи файлов.
- Предоставление высокой пропускной способности и производительности с сохранением устойчивости к ошибкам файловой системы и аппаратного обеспечения.
- Поддержка различных устройств, например жестких дисков и сменных дисков, у каждого из которых разные объемы и производительность.
- Предоставление уровня абстракции, позволяющего изолировать приложения от физических характеристик устройства хранения и конкретного местоположения необходимых данных на физическом носителе. Приложения обрабатывают файлы в качестве линейной последовательности байтов и существуют в рамках этой последовательности, произвольно проводя чтение и запись данных.
- Предоставление уровня абстракции, который позволяет приложениям организовывать файлы в иерархии, например в каталоги, содержащие файлы и другие каталоги. На практике иерархия — сугубо логическая структура, поддерживаемая файловой системой, так как все данные размещены в случайных кластерах диска.
- Предоставление уровня безопасности для данных; например, файловая система NTFS принудительно требует проверки прав доступа к определенным ресурсам, например файлам, каталогам и томам.
- Поддержка параллельного доступа, т.е. работы нескольких пользователей и обработки одновременных запросов на ввод-вывод от каждого пользователя. Кроме того, предоставляется служба индексирования и блокировок, которая позволяет операции ввода-вывода от какого-либо пользователя получить в исключительное владение определенный файл или область файла. Кроме того, файловая система должна восстанавливаться после отказов в работе приложений, которые приняли во владение такие ресурсы файловой системы, как блокированный фрагмент файла.

Файловые системы существуют не сами по себе, а размещены на каких-либо носителях. Хотя оптические носители (компакт-диски и DVD) также имеют файловые системы, в этой книге рассматриваются корпоративные подсистемы хранения данных на базе Windows, которые обычно основаны на жестких дисках. Таким образом, основное внимание в этой главе уделяется организации работы жестких дисков в Windows NT. В первой части главы объясняются такие понятия, как базовые и динамические диски Windows NT, их взаимодействие с разделами и томами.

После этого обсуждаются особенности работы NTFS, рассматриваются важные внутренние структуры данных, например главная таблица файлов (MFT), а также такие функции, как сжатие данных, журнал изменений, отслеживание связей, точки повторной обработки и шифрованная файловая система (EFS).

В последней части главы описываются файловые системы, применяемые в сетях хранения данных, их преимущества и технические проблемы, возникающие при создании файловой системы для сети хранения данных. Кроме того, представлены технологии некоторых конкретных производителей.

6.1 Диски, разделы и тома

Термин *диск* используется при описании такого физического носителя, как накопитель IDE или SCSI, а также сменных носителей, например накопителей USB, компакт-дисков и DVD. Логически диск состоит из кластеров, которые характеризуются размером, например 512, 1024, 4096 байт. Термин *диск* всегда используется при ссылке на физический объект, который можно увидеть и взять в руки. С другой стороны, термины *раздел* и *том*, которые описываются в следующих абзацах, представляют собой логические концепции.

В контексте администрирования некоторые диски (но не все)[1] могут быть разделены на несколько логических областей, которые называются *разделами* (partitions). Каждый раздел может содержать указанный объем данных, и это значение представляет собой интегральное количество кластерных единиц. Например, диск объемом 80 Гбайт может быть разбит на два раздела, один из которых предназначен для установки файловой системы и системных утилит, а второй — для хранения пользовательских данных. Еще одна причина выделения небольшого раздела на корпоративных серверах связана с установкой диагностических утилит.

[1] Примером неразделямых дисков служат компакт-диски и DVD, которые имеют только один раздел.

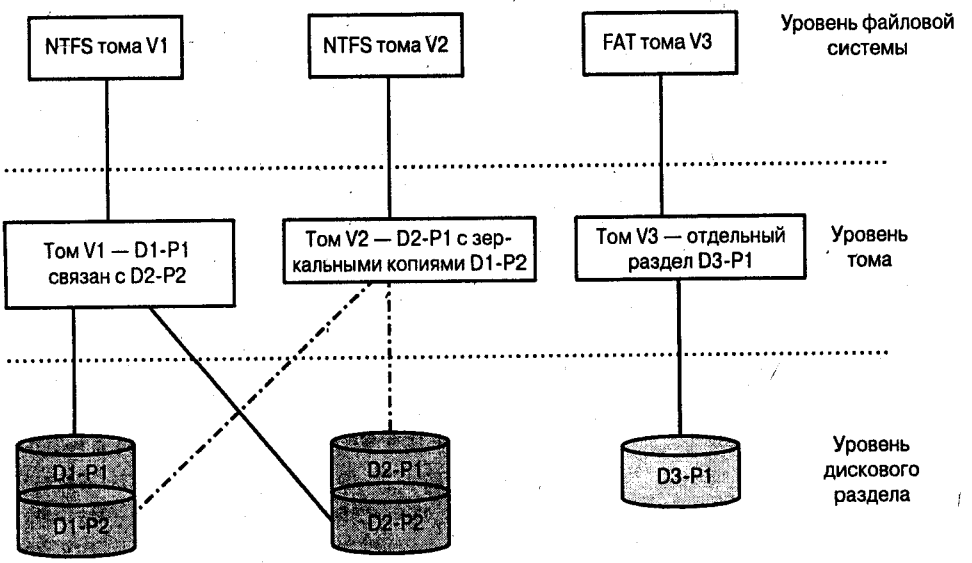


Рис. 6.1. Диски, разделы и тома

Том (volume) — это набор, состоящий из одного или нескольких разделов. Комбинация разделов может быть подобрана для обеспечения быстродействия, объема, целостности данных или сочетания этих параметров. Например, два раздела могут быть объединены для создания тома большего объема или два раздела одинакового размера могут составлять единый зеркальный том, в котором данные дублируются на каждый из разделов. Такие методы комбинации разделов рассматриваются в главе 9, посвященной массивам RAID. Обратите внимание: комбинирование разделов имеет ряд особенностей, которые зависят от типа диска и версии Windows; более подробно они рассматриваются далее.

Если быть точным, то файловые системы размещены на томах. Тома, как уже отмечалось, включают в себя один или несколько дисковых разделов. Дисковые разделы — это логические элементы физического диска. На рис. 6.1 приведена многоуровневая структура. Сверху представлен уровень файловой системы, ниже уровень томов, еще ниже — уровень дисковых разделов.

На рис. 6.1 показана схема NTFS тома V1, который, в свою очередь, состоит из разделов D1-P1 и D2-P2. Кроме того, NTFS установлена на томе V2, который создан из зеркальных разделов D1-P2 и D2-P1 (при этом разделы должны быть одинакового размера). Для обозначения другого метода формирования тома V2 связь между томом и разделами показана штрихпунктирной линией. Файловая система FAT установлена на томе V3, который состоит из одного раздела D3-P1. Диск D1 разбит на два раздела: D1-P1

и D1-P2. Диск D2 также разбит на два раздела: D2-P1 и D2-P2. Диск D3 имеет только один раздел — D3-P1. Обратите внимание, что рис. 6.1 приведен в качестве примера и не описывает всех возможных способов создания томов.

Учитывая базовые идеи, лежащие в основе файловых систем, можно обратиться к деталям. В частности, описанию раздела как набора дисковых кластеров не хватает ряда важных подробностей, например: каким образом операционная система собирает данные на нескольких разделах, существующих на определенном диске, и в каком месте начинается и заканчивается определенный раздел. Ответы на эти вопросы требуют более подробного описания, которое приводиться в разделах 6.1.1 и 6.1.2. Кроме того, разделы на базовых дисках иногда в шутку называют "жесткими" разделами, а разделы на динамических дисках — "мягкими" (об этом речь идет далее в главе). Эти названия отражают неизменность размера жестких разделов (их невозможно увеличить или уменьшить) и возможность изменения размера мягких разделов, даже если они активно используются в текущий момент времени.

Следует упомянуть еще одну особенность. В этой книге рассматривается иерархия, в которой диски делятся на разделы, а разделы составляют тома. Но в некоторых книгах термины *том* и *раздел* взаимозаменяемы, особенно если предполагается, что том состоит из одного раздела. Более того, в некоторых книгах взаимозаменяемыми являются термины *раздел* и *диск*. Читая эту книгу, помните о трехуровневой иерархии: физические диски делятся на логические элементы, которые называются *разделами*, а разделы, в свою очередь, определенным образом формируют другие логические элементы, которые называются *томами*. Файловая система всегда взаимодействует непосредственно с томами.

6.1.1 Базовые диски

Идея разделения рабочего пространства дисков существует уже довольно давно. Термин *базовый диск* (basic disk) впервые появился в Windows 2000 и описывал диски, в которых для предоставления информации о разбивке диска на логические разделы применялись устаревшие технологии DOS.

Первый физический сектор на каждом диске — как базовый, так и динамический — имеет важнейшее значение. Этот сектор содержит структуру данных, которая называется *главной загрузочной записью* (Master Boot Record — MBR) и предоставляет информацию об организации диска, а также участвует в загрузке компьютера. Независимо от операционной системы, запись MBR одинакова для всех персональных компьютеров. Она может иметь размер до 512 байт и включает в себя четыре элемента.

1. *Программа загрузки*, которая имеет размер до 442 байт. Этот код отвечает за сканирование таблицы разделов (описывается ниже), поиск на-

чального сектора загрузочного раздела, загрузку содержимого раздела в оперативную память и передачу управления этому коду, размещенному в памяти.

2. Уникальный номер размером 4 байт, который называется *сигнатурой диска*. Сигнатура всегда расположена со смещением 0x01B8 (с начала сектора) и используется в качестве индекса в системном реестре для сохранения и получения информации о диске.

3. *Таблица разделов*, которая может содержать до четырех записей. Первая запись всегда расположена со смещением 0x01BE. Каждая запись имеет размер ровно 16 байт. Один из описанных в таблице разделов отмечен как активный. С этого раздела загружается операционная система. Первый сектор каждого раздела называется *загрузочным сектором тома* (volume boot sector — VBS) и по своей структуре напоминает MBR. Разница между ними заключается в том, что запись MBR одна на весь диск, а загрузочный сектор существует у каждого тома. Таким образом, один физический диск может иметь несколько загрузочных секторов тома. Каждый загрузочный сектор диска содержит информацию о томе, например размер, количество секторов и метку. Кроме того, загрузочный сектор тома содержит код загрузки операционной системы, который загружается кодом раздела MBR.

4. *Маркер завершения MBR*, который всегда равен 0x55AA.

Обратите внимание: раздел MBR может содержать записи только о четырех разделах, т.е. диск можно разбить максимум на четыре раздела. Для предоставления большего количества разделов один из четырех разделов можно преобразовать (получив тем самым *расширенный раздел*). Такой раздел может содержать собственную таблицу разделов. Описываемое разделение таблиц разделов теоретически может продолжаться до бесконечности.

Базовые диски в полной мере поддерживаются в Windows 2000, Windows XP и Windows Server 2003. Хотя эта поддержка необходима, она не лишена определенных недостатков.

Одна из проблем состоит в том, что упомянутые системы не поддерживают создания сложных томов на основе разделов базовых дисков. Под термином *сложный том* подразумевается том, состоящий из нескольких разделов различных базовых дисков. Примером служит том, состоящий из двух разделов различных базовых дисков для получения единого тома большего объема. В Windows NT 4.0 такие тома называются *составными* (spanned). Операционные системы Windows 2000 и Windows Server 2003 поддерживают уже существующие составные тома на основе базовых дисков, однако создание новых составных томов на основе базовых дисков не поддерживается.

Устаревшие составные тома, которые создавались в более ранних версиях Windows NT (до Windows 2000), могут быть импортированы в Windows 2000, Windows XP и Windows Server 2003.

Базовые диски обладают определенными недостатками. Например, они весьма чувствительны к повреждению первого сектора, который содержит запись MBR. Только аппаратные (а не программные) массивы RAID позволяют обойти это ограничение. Массивы RAID рассматриваются в главе 9. Данные записи MBR не дублируются в программных системах RAID, и для этого есть свои причины: если система загружается и получает доступ к записи MBR, драйвер программного решения RAID еще не загружен, поэтому не сможет помочь в решении проблем с повреждением записи MBR.

Еще один недостаток базовых дисков — необходимость перезагружать компьютер при изменении конфигурации разделов.

6.1.2 Динамические диски

Операционная система Windows 2000 ввела в обиход концепцию *динамических дисков* (dynamic disks). Динамические диски предназначены для преодоления недостатков базовых дисков.

Динамические диски можно перенастраивать "на лету", даже когда приложения (и файловые системы) получают доступ к диску. Сервер не требует перезагрузки для того, чтобы новая конфигурация динамического диска была активизирована для системы. Тома динамических дисков могут быть перенастроены без нарушения обращений к тому или перезагрузки операционной системы. Кроме того, динамические диски более устойчивы к отказам в работе, чем базовые диски, так как важная конфигурационная информация дублируется и копируется на другие физические диски, входящие в дисковую группу.

Как показано на рис. 6.2, *а*, каждый динамический диск поддерживает базу данных объемом 1 Мбайт; речь идет о базе данных диспетчера логических дисков (logical disk manager — LDM). Чтобы предотвратить повреждение данных (например, вследствие работы устаревшей утилиты или в результате вмешательства другой операционной системы), все динамические диски содержат запись MBR, как и базовые диски. Для динамических дисков, которые не содержат файлов операционной системы, запись MBR создается так, чтобы охватить один раздел, занимающий весь физический диск.

Вспомним понятия загрузочного (boot) и системного (system) разделов в Windows NT[2]. Загрузочный раздел содержит файлы операционной систе-

[2]Эта терминология грешит неточностями. Во-первых, на томах размещены файловые системы; таким образом, более корректным будет обозначение загрузочный и незагрузочный раздел. Во-вторых, загрузочный код размещен в "системном" разделе, а файлы операционной системы — в загрузочном разделе.

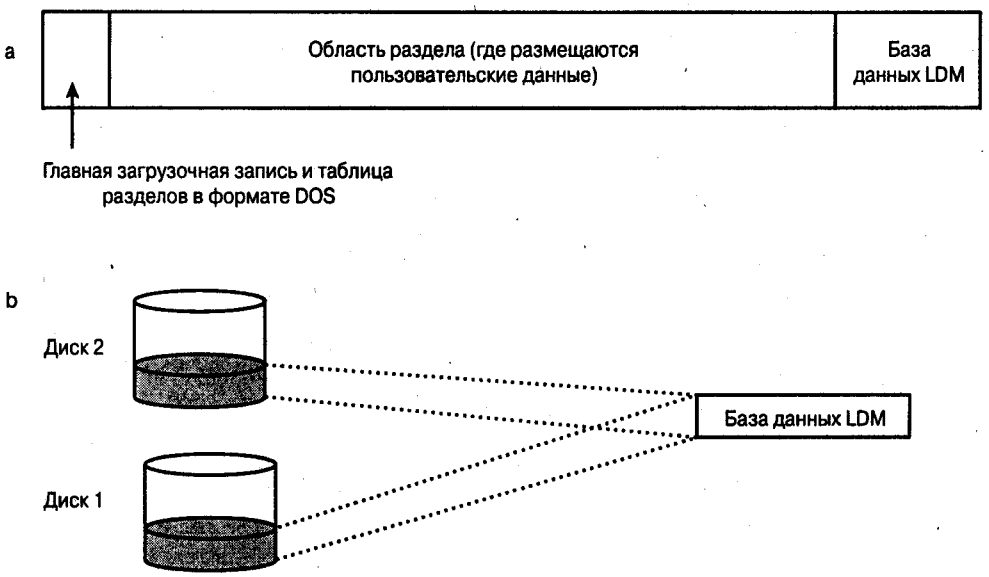


Рис. 6.2. Разбивка динамических дисков (а); база данных диспетчера логических дисков, зеркально отраженная в пределах дисковой группы (б)

мы, например каталог WinNT. Системный раздел содержит загрузочный код. Поскольку загрузочный код Windows NT не поддерживает динамических томов, запись MBR иногда (когда диск содержит системный раздел) создается таким образом, чтобы был обнаружен один раздел и загрузочный код был бы доступен для обнаружения и запуска.

Динамические диски организованы в группы, но диспетчер логических дисков поддерживает только одну дисковую группу. Имя дисковой группы, принятой по умолчанию (и единственной поддерживаемой), создается с помощью добавления значения `dg0` в конец имени компьютера. Как показано на рис. 6.2, *б*, эта база данных содержит информацию обо всех динамических дисках в пределах группы, что предоставляет возможность защиты данных. Здесь показано два диска, входящих в дисковую группу, хотя их может быть и больше. База данных содержит не только данные об организации дисков, но и подробную информацию о монтировании томов, т.е. о букве диска или о каталоге, в котором монтируются тома на данном диске.

Другими словами, вся служебная информация динамического диска размещена на самом диске, поэтому их иногда называют *дисками с самоописанием*. В то же время информация о монтировании (букве диска) для базовых дисков хранится в реестре системы.

Хотя динамические диски обладают массой преимуществ для семейства Windows NT, они не лишены некоторых недостатков. Основные из них описаны ниже.

- Динамические диски не поддерживаются другими операционными системами (отличающимися от Windows 2000 и более поздних версий). Необходимость в считывании таких дисков может возникать в том случае, когда диски подключаются к компьютерам под управлением другой операционной системы или к коммутатору связной архитектуры, взаимодействующему с сервером. Другие операционные системы могут получить доступ к диску средствами Windows 2000, подключаясь к общему ресурсу сервера Windows 2000.
- Динамические диски не поддерживаются некоторыми портативными компьютерами и сменными носителями, например дисками Iomega Jazz или сменными жесткими дисками с интерфейсом 1394/USB.
- Система Microsoft Cluster Server не поддерживает динамические диски, подключенные к совместно используемой кластером шине SCSI, если не установлен менеджер VERITAS Volume Manager. Можно сделать предположение, что для этого существуют причины как технического, так и юридического характера. Достаточно вспомнить, что только диспетчер Volume Manager компании VERITAS поддерживает несколько дисковых групп. Кластеры часто используются в режиме "активный–активный". Если рассмотреть наиболее простую схему кластера, включающего в себя два узла, типичным решением будет разделение дисковых ресурсов на две группы и назначение каждой группы отдельному узлу. Если работа одного узла будет нарушена, дисковая группа этого узла перейдет под управление другого узла. Таким образом, требуется наличие двух дисковых групп, а значит, поставляемый в комплекте с системой диспетчер логических дисков (LDM), поддерживающий только одну дисковую группу, не сможет справиться с описанной ситуацией.
- Единственный способ преобразования динамического диска в базовый состоит в резервном копировании данных, переформатировании диска и обратном восстановлении информации.
- Технология динамических дисков все еще развивается. В частности, поскольку такие важные системные компоненты, как загрузочный код, не поддерживают динамических дисков, некоторые функции старых разделов по-прежнему реализованы в Windows.

Операционные системы Windows 2000 и Windows Server 2003 поддерживают как базовые диски, так и преобразование базовых дисков в динамические. Тем не менее преобразование имеет некоторые особенности, поэтому перед

ним желательно провести резервное копирование, удалить все данные и разделы с диска, преобразовать "пустой" диск в динамический, реорганизовать диск и восстановить данные из резервной копии. Если выполнить обычное преобразование базового диска в динамический, не все преимущества динамических дисков могут оказаться доступными; например, если динамический диск создан на основе базового, динамическое изменение размеров разделов может оказаться недоступным.

6.2 Тома и диспетчеры томов

Как уже отмечалось, том — это логический компонент, включающий в себя дисковые разделы. Эти разделы могут быть реализованы на динамических или базовых дисках. Тома в семействе Windows Server внедряются с помощью драйвера устройства, который называется *диспетчер томов*. Диспетчеры томов и их место в стеке подсистемы хранения данных рассматриваются в главе 1. В этом разделе основное внимание уделяется возможностям томов в операционных системах, появившихся после Windows 2000; в частности, рассматриваются три диспетчера томов.

- Диспетчер *FtDisk*, предоставляемый в Windows 2000 и Windows Server 2003. В Windows NT 4.0 драйвер FtDisk загружался только по требованию, поскольку работал исключительно с расширенными функциями томов, например обеспечением устойчивости к ошибкам. В Windows 2000 FtDisk драйвер загружается по умолчанию, поскольку обрабатывает все тома базовых дисков.
- Диспетчер *LDM* (*Logical Disk Manager*), который предоставляется в Windows 2000 и Windows Server 2003.
- Диспетчер *LVM (VERITAS Logical Volume Manager)*, предлагаемый компанией VERITAS в качестве платной системы; LVM расширяет базовые возможности LDM.

Эти диспетчеры томов обеспечивают перечисленные ниже возможности.

- Виртуализация подсистемы хранения данных; при этом файловая система абстрагируется от физического диска, на котором она размещена. Кроме того, диспетчер томов может объединять несколько разделов нескольких дисков в один большой том.
- Защита данных путем обеспечения их избыточности (с помощью одной из технологий массива RAID, рассматриваемых в главе 9) или посредством их сохранения вместе с контрольной суммой.
- Повышение производительности с помощью хранения данных в формате, поддерживающем эффективное извлечение информации.

В табл. 6.1 приведены возможности каждого диспетчера томов в Windows 2000 и Windows Server 2003.

Таблица 6.1. Возможности диспетчеров томов

Функция	Диспетчер FtDisk	Диспетчер LDM	Диспетчер VERITAS LVM
Простые тома (все разделы на одном физическом диске)	Да	Да	Да
Составные тома (комбинация разделов, расположенных на различных физических дисках)	Нет	Да	Да
Массив RAID 0 (чередование)	Да	Да	Да
Массив RAID 1 (зеркальное отражение)	Да	Да	Да
Массив RAID 10	Нет	Да	Да
Максимальное количество разделов, которые могут составлять один том	32	32	256
Интерактивное создание тома	Нет	Да	Да
Интерактивное расширение составных и простых томов	Нет	Да	Да
Интерактивное расширение томов RAID	Нет	Нет	Да
Поддержка зеркального отражения	Нет	Нет	Да, до 32 зеркал
Поддержка кластеризации	Нет	Нет	Да
Поддержка нескольких групп дисков	Нет	Нет	Да

Диспетчеры LDM и LVM рассматриваются в контексте схожих параметров, однако каждый из них реализован на базе четырех разных драйверов, обладающих различными возможностями. Таким образом достигается уменьшение размера программного кода и упрощается обработка стандартных процедур. Эти драйверы описаны далее.

1. Драйвер *DMConfig* может читать и обновлять базу данных LDM. При изменении конфигурации он отражает изменения в копии базы данных, которая находится в оперативной памяти. Соответствующий драйвер LVM называется *VxConfig*.

2. Драйвер *DMIO* (`dmio.sys`) является аналогом диспетчера FtDisk и реализует возможности диспетчера томов для стандартных операций чтения и записи данных, размещенных на дисковых разделах. Кроме того, DMIO создает объекты устройств томов. Этот драйвер имеет меньший размер, поскольку не содержит кода для чтения и записи базы данных LDM или обработки LDM в дисковом формате. Соответствующий драйвер LVM называется *VxIO*.
3. Драйвер *DMBoot* (`dmboot.sys`) поддерживает только считывание базы данных LDM. Он загружается только после того, как драйвер *DMLoad* (`dmload.sys`) обнаружит более одного динамического диска. Оба драйвера используются только в процессе загрузки. Соответствующие драйверы LVM называются *VxBoot* и *VxLoad*.

При преобразовании базового диска в динамический все старые компоненты преобразуются в новые (табл. 6.2).

Таблица 6.2. Терминология томов в Windows NT 4.0 и Windows 2000

Старый компонент Windows NT 4.0	**Аналогичный новый компонент Windows 2000**
Системный раздел	Простой том
Загрузочный раздел	Простой том
Набор томов	Составной том (том, состоящий максимум из 32 разделов, расположенных на одном или нескольких дисках)
Чередующийся набор	Чередующийся том (том RAID 0, созданный на основе максимум 32 разделов)*
Зеркальный набор	Зеркальный том (том RAID 1, созданный на основе максимум 32 разделов)*
Чередующийся набор с четностью	Том RAID 5*

* Массивам RAID посвящена глава 9.

В Windows 2000 методы управления томами были существенно изменены. Например, Windows 2000 не только аннулирует ограничение в 26 томов, но и позволяет динамически добавлять/удалять тома, не перезагружая систему. Для реализации таких изменений в управлении томами были введены два новых компонента — диспетчер разделов и диспетчер монтирования. Новые возможности по управлению томами, а также эти два диспетчера описаны в разделах 6.2.1–6.2.4.

6.2.1 Диспетчер разделов

Диспетчер разделов — это драйвер в Windows 2000 и Windows Server 2003. Он представляет собой драйвер фильтрации верхнего уровня (драйверы фильтрации рассматриваются в главе 1), который регистрируется в подсистеме Windows NT Plug and Play и запрашивает уведомление о создании новых объектов устройств в драйвере класса диска.

Диспетчер разделов взаимодействует с диспетчерами томов посредством закрытого интерфейса и передает диспетчерам томов уведомления о создании устройств. Как только диспетчер томов обнаруживает, что доступны все разделы, формирующие том, он создает объект устройства, представляющий том. Кроме того, диспетчер разделов уведомляет подсистему PnP об удалении объекта устройства или раздела (например, когда выполняется удаление раздела). Диспетчер разделов сообщает диспетчерам томов о динамическом создании и удалении разделов. Более подробно он рассматривается в разделах 6.2.3 и 6.2.4.

6.2.2 Диспетчер монтирования

Диспетчер монтирования — это драйвер, который появился в Windows 2000 и доступен в Windows Server 2003 и Windows XP. Диспетчер монтирования Windows NT (`mountggr.sys`) предоставляет возможности для управления хранилищем данных на базе томов. К этим возможностям относятся:

- монтирование томов;
- размонтирование томов;
- отслеживание точек монтирования томов и букв дисков в файле базы данных, который называется `:$MountMgrRemoteDatabase` и находится в корневом каталоге каждого тома NTFS;
- создание и удаление пространств имен (и их ассоциаций), которые делают тома доступными для приложений пользовательского режима.

Неудивительно, что диспетчер монтирования зависит от подсистемы Plug and Play, так как требует уведомлений о событиях, которые соответствуют активизации и отключению тома. При монтировании тома диспетчер монтирования "консультируется" с соответствующим диспетчером тома через закрытый интерфейс. Если том расположен на базовом диске, применяется диспетчер томов FtDisk, который обращается к системному реестру для предоставления соответствующей буквы диска. Если том находится на динамическом диске, используется диспетчер томов LDM, который обращается к данным точки монтирования, содержащимся в базе данных LDM динамического диска.

Диспетчер монтирования поддерживает базу данных (уникальных) идентификаторов томов, которые встречались диспетчеру ранее, а также путь или букву диска, назначаемую смонтированному ранее тому. Как только диспетчер монтирования получает уведомление о появлении тома и диспетчер томов не в состоянии предложить точку монтирования, диспетчер монтирования обращается к базе данных и пытается воспользоваться той же буквой диска или путем монтирования, которые использовались при последнем монтировании тома. Если для этого тома в базе данных не содержится информации или точка монтирования уже занята, диспетчер монтирования назначает диску новую точку монтирования. При отключении устройства диспетчер монтирования также отвечает за размонтирование тома. Но запись о точке монтирования из базы данных не удаляется.

Диспетчер монтирования отвечает за назначение букв дисков томам на базовых дисках (но только тех, которые появились после запуска системы). Диспетчер монтирования назначает буквы диска начиная с буквы `C:` в соответствии с приоритетами. Устойчивые к ошибкам (устаревшие) наборы томов[3] Windows NT 4.0 имеют наивысший приоритет, после чего следует основной раздел на фиксированном диске и сменные диски (Jazz, USB). После сменных дисков буквы назначаются накопителям на компакт-дисках. Как только все эти устройства получают буквы дисков начиная с `A:`, буквы назначаются томам на гибких дисках. Накопители на компакт-дисках получают буквы начиная с `D:`.

Диспетчер монтирования сохраняет назначенные буквы дисков в системном реестре, обеспечивая их неизменность, т.е. при изменении конфигурации системы каждому тому будет назначена буква, которая назначалась ранее. Диспетчер монтирования не поддерживает принудительную защиту буквы диска, если соответствующий раздел отключается. Таким образом, если разделу назначается буква диска, и он отключается, эта же буква диска может быть назначена другому разделу, который активизируется первым[4].

Операциями диспетчера монтирования можно управлять из командной строки с помощью утилиты `mountvol.exe`. В Windows Server 2003 возможности диспетчера монтирования были обновлены, поэтому приложения могут отказаться от монтирования томов, которые ранее никогда не монтировались в этой системе. Это слабая попытка обеспечить функции таблицы монтирования UNIX. Смысл нововведения — избежать возможного повреждения тома, если он становится доступен Windows по ошибке.

[3]Буква диска, присвоенная набору томов, устойчивому к ошибкам, хранится в самом томе. При переходе к составному типу тома (начиная с Windows 2000) буква диска может быть изменена.

[4]Исчерпывающая информация по этой теме представлена в статье 234048 базы знаний Microsoft (Knowledge Base).

6.2.3 Дерево устройств для томов базовых дисков

Разобравшись с функциями томов, рассмотрим, как тома обрабатываются в стеке ввода-вывода устройств хранения данных. В этом разделе представлена информация о стеке ввода-вывода для томов базовых дисков, а в разделе 6.2.4 — для томов динамических дисков.

В главе 1 описано дерево устройств для простого тома базового диска, где том состоит из одного раздела. Обратите внимание, что диспетчер томов FtDisk поддерживает устаревшие тома, созданные из нескольких разделов. Программный код для поддержки томов с несколькими разделами так и остался в диспетчере томов FtDisk, но, начиная с Windows 2000, возможность создания тома с несколькими разделами на основе базовых дисков более недоступна.

Обратите внимание на рис. 6.3. Начиная с нижнего правого угла схемы, отображается взаимодействие подсистемы PnP и драйверов шины PCI для создания объектов физического и функционального устройств для шины PCI. После этого драйвер шины PCI перечисляет устройства, подключенные к шине PCI, и создает объект физического устройства для адаптера шины SCSI. Драйвер SCSIPort создает объект функционального устройства для адаптера SCSI. Затем драйвер SCSIPort создает объект физического устройства для одного диска, подключенного к системе, а драйвер класса диска создает объект функционального устройства для этого диска.

Диспетчер разделов следит за проходящими пакетами IRP и обеспечивает правильный путь ввода-вывода для завершения обработки пакетов. Как только диспетчер разделов отмечает завершение обработки запроса `IRP_MN_QUERY_DEVICE_RELATIONSHIPS`, он незаметно удаляет все данные об обнаруженных устройствах (дисках). В данном случае речь идет об объектах устройств для двух разделов — 0 и 1, которые создаются драйвером класса диска `disk.sys`. Таким образом, объекты устройств для разделов 0 и 1 никогда не обнаруживаются подсистемой PnP. Именно поэтому объекты для разделов 0 и 1 обозначены цветом, отличным от цвета остальных объектов устройств (см. рис. 6.3).

Диспетчер разделов передает подробную информацию об обнаруженных объектах устройств зарегистрированным диспетчерам томов. Как только диспетчер тома завершает процедуру инициализации, он регистрируется в подсистеме ввода-вывода. Диспетчер разделов вызывает каждый диспетчер томов по очереди, передавая им информацию об объектах устройств дисков, перехваченных в запросе `IRP_MN_QUERY_DEVICE_RELATIONSHIPS`. Диспетчеры томов проверяют объекты дисковых устройств, перехваченные диспетчером разделов и принимают или отвергают свои права владения этими объекта-

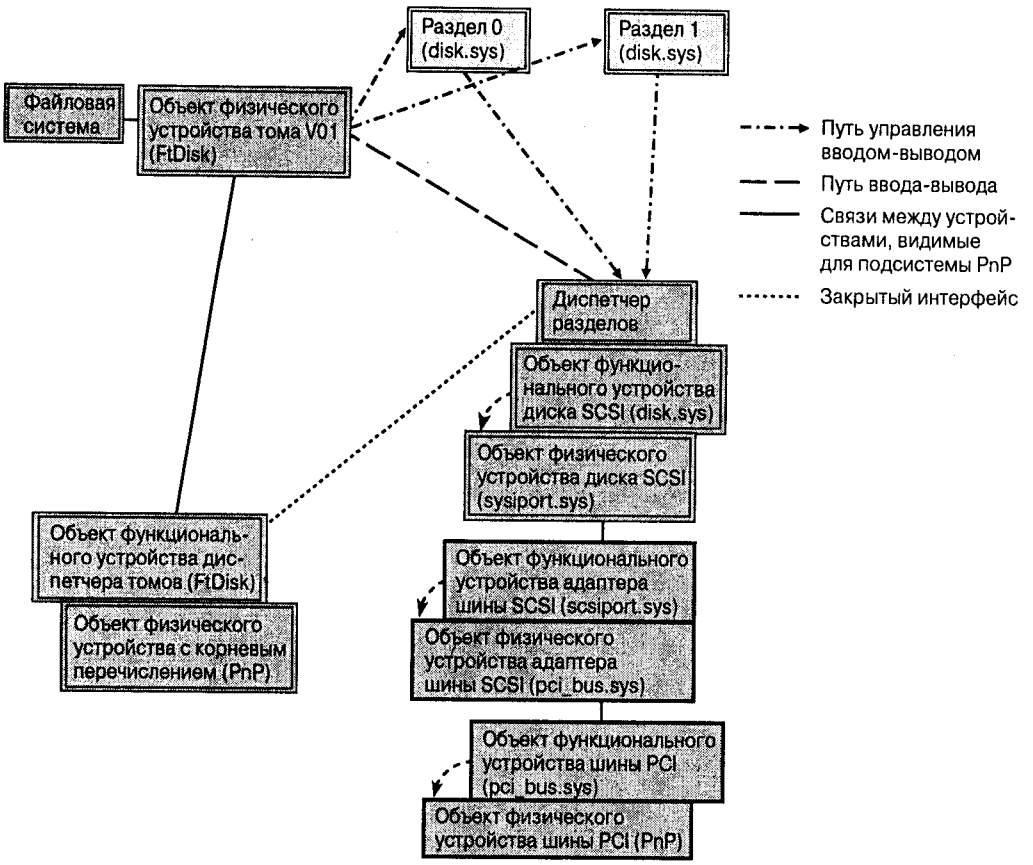


Рис. 6.3. Дерево объектов устройств для устаревшего составного тома на базовом диске

ми. Диспетчер FtDisk принимает права владения всеми томами, от которых отказываются остальные диспетчеры томов.

В примере, показанном на рис. 6.3, драйвер FtDisk становится владельцем устройств объектов. После этого диспетчер FtDisk проверяет конфигурацию тома и определяет, что том состоит из двух разделов, которые диспетчер принимает во владение. На этом этапе диспетчер FtDisk создает объект устройства для представления тома (который на рис. 6.3 называется *Том V01*). Затем для тома может быть смонтирована файловая система.

Стоит упомянуть, что в данном случае используется два отдельных стека устройств. Один стек представляет собой логический элемент — том; второй стек содержит физические устройства системы, например шину PCI, адаптер шины SCSI и дисковый привод. Диспетчер томов выступает в роли моста между этими двумя стеками.

На рис. 6.3 драйвер FtDisk отправляет все распознанные запросы IRP непосредственно драйверу класса диска. Конечно, драйвер FtDisk перед отправкой пакета проводит преобразование смещений относительно тома в смещения относительно диска. Кроме того, запросы управления вводом-выводом, которые не воспринимаются драйвером FtDisk, отправляются разделу 0 или 1, в зависимости от назначения запроса.

6.2.4 Дерево устройств для томов динамических дисков

Рассмотрим подробнее дерево устройств для томов динамических дисков, которое отличается рядом параметров от дерева устройств для томов на базовых дисках. На рис. 6.4 показано дерево устройств для томов на динамических дисках.

Чтобы не усложнять пример, представим, что рассматриваемый том — исключительно программный и находится на динамическом диске. Динами-

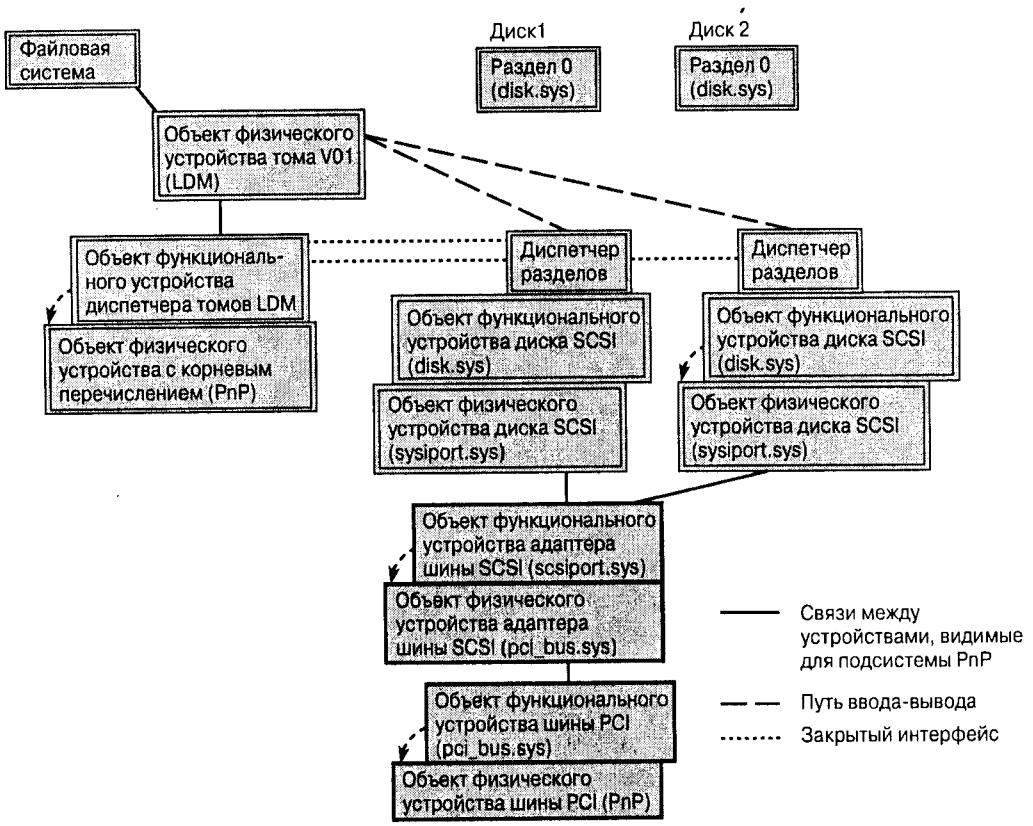


Рис. 6.4. Дерево устройств для тома на динамическом диске

ческий диск не содержит системного или загрузочного раздела операционной системы, т.е. в записи MBR динамического диска указано, что диск имеет один раздел, занимающий весь свободный объем. Конечно, последний мегабайт жесткого диска содержит базу данных диспетчера томов LDM.

На рис. 6.4 представлены два отдельных дерева объектов, которые читателю уже знакомы: дерево физических объектов (справа), дерево логических объектов (слева) и диспетчер логических дисков (LDM), соединяющий оба дерева. Начиная с нижнего правого угла, на рис. 6.4 представлена пары объект физического устройства–объект функционального устройства для шины PCI. Выше можно заметить пару "объект физического устройства–объект функционального устройства" для шины SCSI. В этом примере к системе подключены два динамических диска. На следующем уровне показана пара "объект физического устройства–объект функционального устройства" для двух физических дисков, установленных в системе. Диспетчер разделов расположен над объектами функциональных устройств, созданными драйвером класса диска.

Диспетчер разделов регистрирует процедуру завершения для всех обрабатываемых им пакетов IRP. В процедуре завершения особое внимание уделяется пакетам `IRP_MN_QUERY_DEVICE_RELATIONSHIPS` и ожидаются сообщения о создании объектов устройств дисков. Драйвер класса диска, работая как драйвер шины, создает объекты устройств для описания одного раздела, который представляет динамический диск. Диспетчер разделов передает эту информацию об объектах устройств дисков зарегистрированным диспетчерам томов. В этом примере показан единственный диспетчер томов — LDM, который загружается как логический драйвер с корневым перечислением. В данном контексте диспетчер LDM примет на себя ответственность за объекты устройств дисков. Предположим, что нужный том сформирован из двух дисков. Диспетчер томов LDM проверяет конфигурацию тома и, как только владение всеми объектами разделов будет подтверждено, создает объект устройства, представляющий том, который в этом примере называется *Том V01*.

Диспетчер томов LDM перенаправляет ввод-вывод от объекта тома к одному из дисков, который содержит необходимые данные. Обратите внимание, что два объекта разделов — раздел 0 для диска 1 и раздел 0 для диска 2 — никогда не передаются подсистеме PnP, поэтому они не связаны с другими объектами.

6.3 Пространство имен устройств

Данное пространство имен развилось за несколько лет существования Windows NT. Для обеспечения обратной совместимости новое пространство имен объединяется со старым пространством с помощью символических ссылок. Кроме того, драйверы режима ядра создают пространство имен в режиме ядра, используя возможности диспетчера объектов. Приложения пользовательского режима воспринимают другие пространства имен, взаимодействующие друг с другом посредством символических ссылок. Иногда разобраться с хитростями пространств имен не так-то просто. В этом разделе делается скромная попытка перечислить наиболее важные для подсистемы хранения имена устройств.

Драйвер класса диска создает объекты устройств для представления каждого физического диска. Эти объекты имеют название `\device\harddiskX`, где X — число, начинающееся с нуля и увеличивающееся для каждого найденного жесткого диска.

Кроме того, драйвер класса диска создает объект устройства для каждого найденного основного раздела. Драйвер класса диска использует функцию `IoReadPartitionTable` диспетчера ввода-вывода для поиска всех основных разделов на диске. Такие основные разделы называются `\device\harddiskX \partitionY`, где X — номер диска, а Y — номер основного раздела, расположенного на этом физическом диске. Диспетчер ввода-вывода создает символьную ссылку в формате `\??\PhysicalDriveX`, где X — число больше нуля, отображаемое на ссылку `\device\harddiskX\partitionY`.

Диспетчер томов LDM создает объект для каждого поддерживаемого тома. Эти объекты устройств имеют имена в формате `\Device\HarddiskVolumes \PhysicalDmVolumes\BlockVolumeX`, где X — идентификатор, который назначается тому диспетчером томов. Это устройство режима ядра соотносится с устройством Win32, которое создается диспетчером монтирования и имеет вид `\??\Volume[GUID]`, где *GUID* — глобально уникальный идентификатор. Диспетчер томов также создает символьную ссылку в формате `\Device\HarddiskDmVolumes\ComputerNameDg0\VolumeY` для каждого тома и соотносит ссылки с определенными устройствами в каталоге `PhysicalVolumes`. При этом значение *ComputerName* заменяется фактическим именем компьютера, а Y — идентификатором тома.

Для предоставления прямого доступа к тому диспетчер томов LDM создает объект для каждого поддерживаемого тома. Этот объект устройства имеет имя в формате `\Device\HarddiskDmVolumes\PhysicalDmVolumes\RawVolumeX`.

Обратите внимание, что в данном разделе не описываются все возможные имена устройств и их форматы. Любознательным читателям рекомендуется

запустить утилиты управления дисками и томами с графическим интерфейсом и обратить внимание на предоставляемые ими сведения.

6.4 Другие файловые системы

Операционная система Windows NT обеспечивает поддержку различных файловых систем. Основной, обладающей полным спектром возможностей файловой системой является NTFS. Компания Microsoft утверждает, что в ныне устаревшую файловую систему FAT новые функции вноситься не будут. Кроме NTFS и FAT (которые рассматриваются далее в главе), операционная система поддерживает и другие файловые системы.

- Файловая система CDFS, совместимая со стандартом ISO 9660 и предназначенная для ввода-вывода данных на дисководы для компакт-дисков.
- Файловая система UDF (Universal Disk Format), разработанная ассоциацией OSTA. Поддержка UDF была впервые реализована в Windows 2000. Эта файловая система — наследник CDFS, поддерживающий DVD. В Windows 2000 предоставляется поддержка только чтения, а в Windows XP — как чтения, так и записи данных.

Обратите внимание, что один экземпляр операционной системы может одновременно использовать несколько файловых систем. Более того, компьютер под управлением Windows может иметь несколько томов и вовсе не обязательно, чтобы все тома имели одинаковую файловую систему.

6.4.1 Файловая система FAT

Операционная система Windows 2000 поддерживает обновленную версию FAT (File Allocation Table). Следует отметить ряд особенностей FAT.

- Существует две версии файловой системы FAT — FAT16 с 16-разрядными указателями и FAT32 с 32-разрядными указателями.
- Файловая система FAT16 поддерживает тома размером до 4 Гбайт при условии применения кластеров размером 64 Кбайт. Файловая система FAT32 поддерживает разделы размером до 32 Гбайт при условии применения кластеров размером 16 Кбайт.
- Корневой каталог может содержать не более 512 записей.
- Сжатие и механизмы безопасности не поддерживаются.

6.5 Файловая система NTFS

Эта файловая система проектировалась специально для Windows NT. С момента первого появления NTFS в нее вносилось несколько модификаций, но основная архитектура оставалась неизменной. Файловых систем FAT и HPFS (High-Performance File System), поддерживаемых Microsoft в момент появления NTFS, было явно недостаточно для удовлетворения потребностей Windows NT.

- Файловая система FAT не предоставляет необходимого уровня безопасности файлов и объектов.
- Файловая система FAT не поддерживает возможностей по обработке вместительных жестких дисков, доступных в настоящий момент. (Вспомните, что изначально FAT проектировалась для использования на дисках объемом 1 Мбайт.)
- Как FAT, так и HPFS не поддерживают транзакций, которые необходимы для обеспечения надежности данных и их восстановления после отказов в работе системы.

Файловая система NTFS предоставляет различные возможности, которые перечислены ниже и подробно описываются далее в главе.

- Поддержка транзакций в NTFS обеспечивает регистрацию всех изменений метаданных файловой системы в специальном журнале, который поддерживает восстановление в случае неисправности.
- Все данные, включая метаданные системы, хранятся в файлах.
- NTFS и программные интерфейсы приложений Win32 поддерживают использование 64-разрядных указателей для структур данных файлов.
- NTFS поддерживает имена файлов длиной до 255 символов и кодировку Unicode.
- Структуры данных поддерживают быстрое перемещение и поиск в каталогах.
- Файловая система поддерживает сжатие и разреженные файлы.
- Начиная с Windows 2000 поддерживается шифрованная файловая система (EFS).
- Файловая система обеспечивает устойчивость к отказам, например к появлению поврежденных кластеров на диске или перераспределению секторов.
- NTFS не имеет ограничения на длину имен 8.3, которое было характерно для MS DOS. Кроме того, обеспечивается совместимость с именами

файлов POSIX, включая точки и пробелы в начале имени. Тем не менее иногда при использовании имен, не соответствующих стандарту 8.3, возникают проблемы. Основной их причиной могут быть утилиты, не поддерживающие длинных имен, которые не соответствуют стандарту 8.3. Отдельные имена файлов NTFS могут иметь размер до 255 символов, а полный путь к файлу не должен превышать 32 767 символов.

- В NTFS используются 64-разрядные указатели файлов и теоретически может поддерживаться размер файла 2^{64} байт.

В NTFS поддерживается несколько потоков данных для одного файла. Поток можно открыть с помощью функции Win32 API `CreateFile`, а имя потока в виде `:ИмяПотока` может быть добавлено к имени файла, например `File1:Stream25`. Потоки поддерживают запись, чтение и независимую от других открытых потоков блокировку. Операционная система Windows NT для серверов Macintosh использует эту функцию при поддержке клиентов Mac, на которых файл имеет две "ветви": ветвь данных и ветвь ресурсов.

Обратите внимание, что, хотя NTFS и поддерживает несколько потоков, множеству утилит и программ об этом ничего не известно. Таким образом, о файле, содержащем 1024 байт в обычном неименованном потоке и 1 Мбайт данных в именованном потоке, команда `dir` сообщит, как о файле размером 1024 байт (команда `dir` не поддерживает многопоточность). При копировании файлов с несколькими потоками с раздела NTFS в FAT копируется только неименованный поток, принятый по умолчанию. Данные из остальных потоков считаются потерянными.

В табл. 6.3 сравниваются FAT и NTFS.

Таблица 6.3. Сравнение файловых систем, поддерживаемых Windows NT

Комментарий	FAT16	FAT32	NTFS
Максимальная длина имени файла	8.3	255	255
Максимальный размер файла	2 Гбайт	4 Гбайт	Теоретический максимум 16 Эбайт
Максимальный размер тома	2 Гбайт	2 Тбайт	2 Тбайт
Совместимость с гибкими дисками	Да	Да	Нет
Несколько дисков в одном томе	Нет	Нет	Да
Безопасность на уровне файлов и каталогов	Нет	Нет	Да
Проверка доступа на уровне файлов и каталогов	Нет	Нет	Да

Окончание табл. 6.3

Комментарий	FAT16	FAT32	NTFS
Возможности устойчивости к отказам (несколько копий критических данных, журнал метаданных)	Нет	Нет	Да
Шифрование и сжатие файлов	Нет	Нет	Да

6.5.1 Системные файлы NTFS

Файловая система организует все содержимое диска в виде файлов, включая не только пользовательские файлы, но и файлы, содержащие метаданные, которые относятся к файловой системе. В этом разделе рассматриваются файлы, которые NTFS использует для внутренней организации функционирования.

Главная файловая таблица (Master File Table — MFT; `$Mft`) всегда представляет собой первый файл тома NTFS. Она содержит множество записей и, как минимум, одну запись для каждого файла и каталога тома, включая запись для самой MFT. Каждая запись в MFT может иметь размер 1024–4096 Кбайт, в зависимости от размера тома, на котором размещена файловая система. Для файлов с несколькими атрибутами или чрезмерной фрагментацией может потребоваться несколько записей в MFT. Таблица MFT хранится в начале тома.

Производительность системы существенно повышается, если записи MFT хранятся в соседних кластерах диска, т.е. когда MFT не фрагментирована и занимает непрерывную область диска. Для выполнения этого условия NTFS резервирует область, которая называется *зоной MFT*, в начале тома или раздела и стремится не использовать эту область ни для чего, кроме хранения записей MFT. Первые файлы и каталоги записываются сразу после MFT. Около 12% тома зарезервировано для зоны MFT. Начиная с Windows NT 4.0 SP4, в реестр добавлена специальная запись, которая позволяет управлять размером зоны MFT. Эта запись может иметь значения в диапазоне от 1 до 4, указывая размер зоны MFT от минимального (1) до максимального (4). Программа дефрагментации указывает текущий размер зоны MFT.

Первые 24 записи в MFT зарезервированы. Некоторые записи меняют свое назначение с выходом новой версии операционной системы, как это произошло с выходом Windows 2000. В табл. 6.4 описаны различные системные файлы NTFS, которые также известны как *файлы метаданных*.

Таблица 6.4. Системные файлы NTFS

Файл*	Номер записи	Описание
`$Mft`	0	Главная файловая таблица
`$MftMirr`	1	Зеркало MFT, содержащее копию первых 16 файлов MFT
`$LogFile`	2	Файл журнала (для восстановления после сбоев и поддержания целостности файловой системы)
`$Volume`	3	Описание тома, включая серийный номер тома, дату и время создания, а также флаг тома
`$AttrDef`	4	Определение атрибута
. (точка)	5	Корневой каталог
`$Bitmap`	6	Битовая карта размещения кластеров
`$Boot`	7	Загрузочная запись диска
`$BadClus`	8	Список поврежденных кластеров
`$Quota`	9	В Windows NT 4 определен как файл пользовательских квот, однако никогда не использовался
`$Secure`	9	В Windows 2000 переопределен как дескриптор безопасности
`$UpCase`	10	Таблица верхнего регистра
`$Extend`	11	Каталог, который содержит файлы `$ObjId`, `$Quota` и `$UsrJrnl`. Используется в Windows 2000 и более поздних версиях
—	12–23	Зарезервированы для будущего использования

* Знак доллара ($) перед именем файла означает, что это файл метаданных.

Файл `$MftMirr` содержит зеркальную копию первых 16 записей MFT и представляет собой метод обеспечения целостности тома, даже если секторы MFT окажутся поврежденными. Файл `$MftMirr` хранится в середине тома. Более объемные тома могут иметь более одного зеркала MFT.

Файл журнала `$LogFile` используется для восстановления после отказов в работе системы или возникновения неожиданных ситуаций. NTFS поддерживает транзакции и протоколирование: сначала в журнал заносятся все изменения системных метаданных и только затем проводятся изменения. Файл журнала содержит информацию для отмены и повтора операций, которая используется для восстановления после отказов в работе системы и поддержки целостности файловой системы.

Обратите внимание: метаданных из файла журнала транзакций достаточно лишь для поддержки целостности файловой системы, например для обеспечения правильной маркировки кластеров: или как свободных, или принадлежащих определенному файлу. Содержимое этих кластеров — пользовательские данные — в журнале не отслеживается. После фиксирования транзакции и изменения метаданных запись о завершении операции заносится в файл журнала.

Все операции с файлом журнала выполняются средствами службы файлового журнала NTFS, которая предоставляет собой набор процедур драйвера NTFS. Для хранения файла журнала в NTFS используется циклический буфер. В начале буфера содержится указатель на область в самом буфере, в которой должен начаться процесс восстановления. Этот указатель в файле журнала сохраняется дважды (избыточно), чтобы обеспечить возможность восстановления в том случае, если одна из областей файла журнала окажется поврежденной.

Файл `$Volume` содержит имя тома, дату и временную метку (указывающую на время создания тома), информацию о версии NTFS и дополнительный бит, используемый для проверки правильности завершения работы системы. Кроме того, этот бит проверяется при загрузке системы для определения необходимости запуска утилиты `CHKDSK`.

Файл `$AttrDef` содержит все атрибуты, которые поддерживаются на данном томе. Для каждого атрибута указывается различная информация, например имя атрибута, тип атрибута, минимальная и максимальная длина.

Корневой каталог также обладает большим значением. Все операции поиска файлов и каталогов должны начинаться с корневого каталога, если они не кэшированы во время предыдущих операций поиска.

Файл `$Bitmap` представляет каждый кластер диска, на котором находится том. Каждый бит указывает занятость одного кластера диска.

Файл `$Boot` предназначен для хранения и защиты кода загрузчика, который всегда размещен в фиксированной области, удаленной от начала тома. При форматировании тома с использованием NTFS утилита форматирования проверяет, владеет ли файл `$Boot` кластерами, в которые помещен код загрузчика. Таким образом, код загрузчика защищен от перезаписи другими данными.

Файл `$BadClus` содержит записи для каждого поврежденного кластера на диске. Этот файл обновляется динамически; т.е. для каждого динамически обнаруженного поврежденного кластера в файл добавляется новая запись.

Файл `$Secure` впервые появился в Windows 2000. Файловая система NTFS поддерживает механизмы безопасности для каждого файла и каталога. До выхода Windows 2000 данные о безопасности хранились в каждой записи для файла и каталога в MFT. Поскольку множество файлов и каталогов

имели одинаковую информацию о правах доступа, такая информация часто дублировалась. Например, если у пользователя есть право доступа на чтение и запуск 100 файлов, формирующих приложение (например, Microsoft Office), все эти файлы будут иметь одинаковую информацию безопасности. Начиная с Windows 2000 информация о безопасности сохраняется один раз в файле `$Secure`, и все файлы просто ссылаются на эти данные.

Файл `$UpCase` содержит таблицу, используемую для преобразования имен файлов и путей из нижнего регистра в верхний, а также для преобразования имен файлов в верхний регистр для приложений, которые не различают регистр в именах файлов.

Каталог `$Extend` впервые появился в Windows 2000 и содержит файлы, позволяющие реализовать необязательные возможности NTFS. Далее приведен перечень файлов, которые содержатся в каталоге `$Extend`.

- Файл `$ObjId` содержит идентификаторы объектов для файлов и каталогов. Идентификаторы объектов используются для отслеживания файлов и каталогов при их перемещении. Дополнительная информация по этой теме приводится в разделе 6.5.15.
- Файл `$Quota` используется для хранения информации о квотах для тех томов, где квотирование включено. Отслеживание квот рассматривается в разделе 6.5.9.
- Файл `$UsnJrnl` содержит информацию, которая относится к изменениям в файлах и каталогах. Дополнительная информация по этой теме приводится в разделе 6.5.13.
- Файл `$Reparse` содержит информацию обо всех файлах и каталогах, с которыми связан тег точки повторной обработки. Точки повторной обработки (reparse points) представляют собой механизм реализации символьных ссылок. Эта тема рассматривается в разделе 6.5.22.

6.5.2 Логические и виртуальные номера кластеров NTFS

Файловая система NTFS работает с целым числом дисковых секторов как с минимальным единичным блоком данных. Такой блок называется *кластером*. Размер кластера определяется при форматировании тома. Разные тома могут иметь различные размеры кластеров. Для читателей, знакомых с UNIX, можно отметить, что термин *кластер* в Windows аналогичен термину *размер блока файловой системы* в UNIX. Файловая система вычисляет размер кластера, принимая во внимание размер диска и тип используемой файловой системы. Кластер может иметь размер в диапазоне 1–64 Кбайт. Размер кластера задается при форматировании тома в виде параметра команды `format` или параметра утилиты управления дисками с графическим

интерфейсом. Очевидно, что большой размер кластера приводит к потерям дискового пространства; например, для хранения файла размером 1 Кбайт файловой системе придется выделить кластер размером 64 Кбайт.

К кластерам относится несколько важных параметров NTFS. Первый параметр называется *логическим номером кластера* (logical cluster number — LCN). Файловая система NTFS делит весь диск на кластеры и назначает каждому кластеру номер, начиная с нуля. Таким образом, первый кластер будет иметь номер 0, второй кластер — номер 1 и т.д. Этот номер и называется LCN. Вторым важным параметром является *виртуальный номер кластера* (virtual cluster number — VCN), который указывает номер кластера внутри определенного файла. Таким образом, логический номер кластера, равный 25, указывает на 26-й (помните, что нумерация начинается с нуля) кластер тома, а виртуальный номер кластера, равный 25, указывает на 26-й кластер определенного файла.

Виртуальный номер кластера позволяет вычислить местоположение атрибута, например смещения данных внутри файла, а логический номер кластера дает возможность вычислить смещение относительно тома или раздела для определенного блока данных.

6.5.3 Структура записи MFT в файловой системе NTFS

Как уже отмечалось, каждый файл и каталог в NTFS имеет собственную запись в главной таблице файлов. Эта запись иногда упоминается как запись MFT. Каждая запись MFT имеет фиксированный размер, который определяется в момент форматирования диска и находится в диапазоне 1024–4096 байт. В Windows NT 3.51 запись MFT имела размер 4 Кбайт. В Windows NT 4.0 компания Microsoft изменила минимальный размер записи, чтобы он составлял 1 Кбайт или был равен размеру кластера, в зависимости от того, что больше. Это было сделано после проведения анализа, показавшего, что записи MFT чрезмерно занимают дисковое пространство.

Запись MFT содержит стандартный заголовок, после которого идет последовательность атрибутов, сохраняемых в такой форме:

- заголовок атрибута;
- название атрибута;
- данные атрибута.

В качестве примера атрибутов можно указать имя файла, список управления доступом файла и данные файла. В табл. 6.5 приведена информация о разных атрибутах, которые может иметь файл или каталог NTFS.

Таблица 6.5. Атрибуты NTFS

Атрибуты	Значение типа атрибута	Описание
$STANDARD_INFORMATION	0x10	Информация о стандарте файла
$ATTRIBUTE_LIST	0x20	Используется для указания нерезидентных атрибутов
$FILENAME	0x30	Имя файла сохраняется как атрибут с несколькими потенциальными значениями, так как у файла может быть несколько имен (имя NTFS, имя DOS и закрепленные ссылки)
$VOLUME_VERSION	0x40	Атрибут определен, однако не используется в Windows NT 4.0. Удален из Windows 2000
$OBJECT_ID	0x40	Значение размером 64 байт, которое используется в Windows 2000 для отслеживания ссылок. Не применяется в более ранних версиях Windows NT. Дополнительная информация приводится в разделе 6.5.15
$SECURITY_DESCRIPTOR	0x50	Дескриптор безопасности (список управления доступом файла). Дополнительная информация приводится в разделе 6.5.6
$VOLUME_NAME	0x60	Имя тома. Содержится только в файле **$Volume**
$VOLUME_INFORMATION	0x70	Информация о томе. Содержится только в файле **$Volume**
$DATA	0x80	Пользовательские данные файла, которые сохраняются в атрибуте с несколькими потенциальными значениями, так как файлы NTFS могут иметь несколько потоков
$INDEX_ROOT	0x90	Используется в больших каталогах
$INDEX_ALLOCATION	0xA0	Используется в больших каталогах
$BITMAP	0xB0	Используется только в каталогах
$SYMBOLIC_LINK	0xC0	Атрибут определен, однако не используется в Windows NT 4.0

Окончание табл. 6.5

Атрибуты	Значение типа атрибута	Описание
$REPARSE_POINT	0xC0	Наличие атрибута указывает на то, что файл метаданных содержит точки повторной обработки. Дополнительная информация предоставлена в разделе 6.5.22
$EA	0xE0	Расширенные атрибуты OS/2
$EA_INFORMATION	0xD0	Информация о расширенных атрибутах OS/2
$PROPERTY_SET	0xF0	Набор свойств. Атрибут определен, однако не используется в Windows NT
$LOGGED_UTILITY_STREAM	0x100	Используется EFS. Дополнительная информация предоставлена в разделе 6.5.20

Если данные атрибута имеют небольшой размер, они будут сохранены непосредственно в записи MFT. Такие атрибуты называются *резидентными*. С другой стороны, если данные слишком велики для хранения в MFT, сохраняется информация о расположении этих данных (номера кластеров, в которых размещены необходимые данные). Такие атрибуты называются *нерезидентными*. Ничего особенного в этих атрибутах нет, поскольку любой атрибут может быть резидентным или нерезидентным.

На рис. 6.5 представлены данные файлов, сохраненные в виде нерезидентных атрибутов. В этом случае структура данных включает три элемента.

1. *Виртуальный номер кластера* (VCN), который указывает расположение кластера относительно начала файла. Например, виртуальный номер кластера, равный 0, указывает, что необходимый кластер является первым кластером атрибута файла.
2. *Логический номер кластера* (LCN), который указывает расположение кластера относительно тома или раздела. Например, логический номер кластера, равный 25, указывает, что необходимый кластер является 26-м кластером от начала тома или раздела.
3. *Количество кластеров* в определенной "цепочке", т.е. количество кластеров в непрерывной последовательности, выделенных для хранения атрибутов файла.

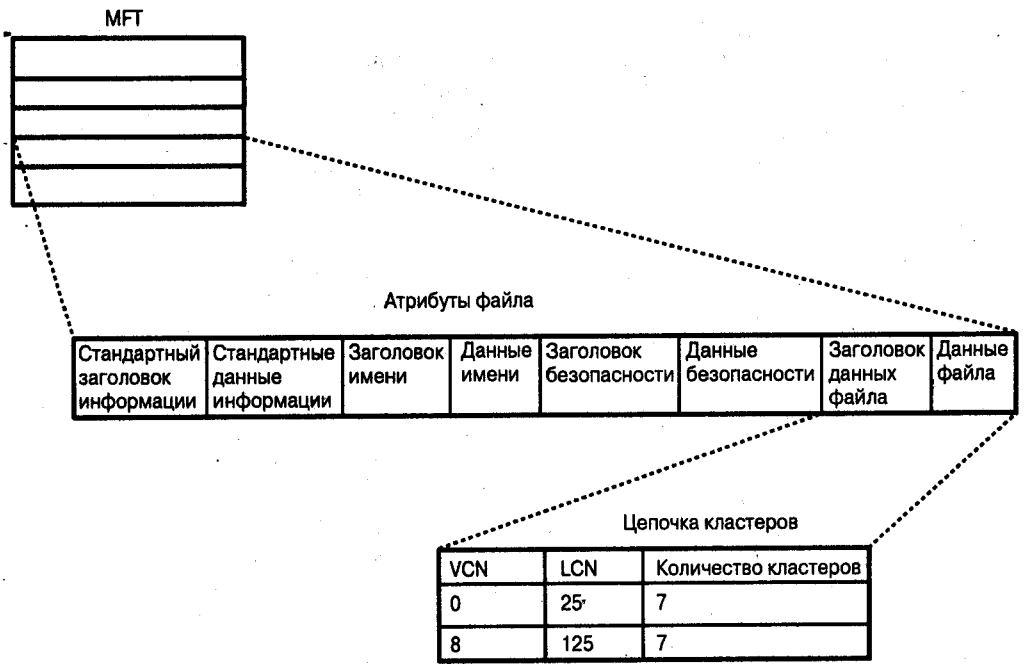


Рис. 6.5. Структура записи MFT

Если цепочка кластеров файла не помещается в одну запись MFT, он сохраняется в дополнительных записях MFT.

Кроме того, NTFS поддерживает несколько потоков данных. Поток, принятый по умолчанию, открывается при использовании функции `CreateFile` с именем файла в виде относительного или абсолютного пути. Указав имя файла и имя потока через двоеточие, можно открыть другой поток данных, например `\directory1\File1:DataStream2`. Файловая система NTFS хранит эту информацию как еще один атрибут в MFT, а данные второго потока хранит в виде другого атрибута.

6.5.4 Каталоги NTFS

Это обычные файлы, которые содержат информацию о каталоге. Файловая система NTFS хранит каталоги способом, ускоряющим просмотр их содержимого. Если данные каталога не помещаются в MFT, то NTFS выделяет кластер и последовательную структуру, аналогичную области хранения данных и другим атрибутам. Все записи каталогов хранятся в деревьях B+.

Дерево B+ — это структура данных, которая поддерживает упорядоченное хранение информации и эффективна в контексте операций поиска, удале-

ния, вставки и просмотра данных. Деревья B+ состоят из "узловых" записей, содержащих ключи, а также указателей, которые соединяют узлы дерева. Преимущество использования такой структуры данных состоит в тенденции к расширению дерева, а не к увеличению в глубину, что позволяет сохранить приемлемую производительность даже для каталогов, которые содержат большое количество записей.

Записи в каталоге хранятся в отсортированном виде. Каждая запись содержит имя файла, указатель на запись файла в MFT, а также дату/временную метку (эта информация уже хранится в записи файла MFT). Это позволяет добиться высокой производительности при просмотре содержимого каталога. Деревья B+ эффективны в аспекте количества сравнений, необходимых для поиска заданной записи каталога.

6.5.5 Журнал восстановления NTFS

Файловая система NTFS проектировалась для обеспечения высокой производительности и надежности, с поддержкой восстановления после отказов в работе системы. Для достижения этой цели NTFS заносит в журнал `$LogFile` все изменения метаданных файлов перед каждой фактической попыткой их изменения. В файлах журнала содержится информация, необходимая для восстановления и повтора изменений в случае отказа в работе системы.

В Windows NT 4.0 файл журнала очищался при каждой успешной перезагрузке. В Windows 2000 записи файла журнала сохраняются в течение нескольких перезагрузок.

6.5.6 Безопасность NTFS

Безопасность NTFS унаследована от схемы безопасности Windows NT и ее объектной модели. Каждый файл или каталог имеет дескриптор безопасности, который состоит из следующих элементов:

- токен безопасности, указывающий на владельца файла;
- последовательность списков управления доступом (ACL), которые явно или неявно предоставляют доступ к файлу пользователям, указанным в списке;
- необязательная последовательность списков управления доступом, которые явно или неявно запрещают доступ к файлу определенным пользователям; если пользователь находится в списке, разрешающем доступ, и в списке, запрещающем доступ, доступ не предоставляется.

В Windows NT 4.0 дескриптор безопасности хранился в записи MFT конкретного файла. Поскольку множество файлов и каталогов имели похожие списки управления доступом, информация о безопасности многократно дублировалась. Например, если у пользователя есть права на чтение и выполнение 100 файлов, составляющих приложение (примером может служить Microsoft Office), все файлы получат одинаковую информацию о безопасности. Начиная с Windows 2000 информация о безопасности хранится в файле `$Secure`, и все остальные файлы просто ссылаются на этот файл.

6.5.7 Разреженные файлы NTFS

Файловая система поддерживает *разреженные файлы*, которые позволяют хранить только ненулевые данные. Если файл используется для представления такой структуры данных, как разреженная матрица, эта возможность оказывается исключительно полезной. Поддержка разреженных файлов может быть включена или отключена администратором для целого тома, каталога (для файлов и каталогов, расположенных в этом каталоге) или для отдельного файла. Этот параметр может быть определен приложением при создании файла или каталога. Если текущий том или каталог помечен как *разреженный*, никаких действий по отношению к уже существующим файлам не выполняется и параметр относится только к новым файлам и каталогам.

Разреженные и сжатые файлы представляют собой две совершенно разные независимые структуры, предназначенные для сокращения объема используемого дискового пространства. Файл может быть сжатым, но неразреженным и наоборот. Сжатие рассматривается в разделе 6.5.8.

Термин *разреженный* относится к файлам, содержащим данные, после которых идет большая область без данных, за ней небольшой фрагмент данных и т.д. Файловая система NTFS не выделяет дискового пространства для хранения пустых областей файлов. Вспомните, что виртуальный номер кластера определяет положение кластера относительно начала файла, а логический номер кластера определяет положение кластера относительно начала тома. Для разреженных файлов в NTFS выделяется виртуальный номер кластера, однако кластеры тома не выделяются. Таким образом, логический номер кластера, относящийся к тому, для некоторых виртуальных номеров кластеров не выделяется. Если приложение пытается считать файл, NTFS заполняет участки буфера, соответствующие пустым пространствам файла, нулями. Если приложение попытается записать данные в пустые области файла, файловая система выделит необходимый объем дискового пространства.

Обратите внимание на рис. 6.6: виртуальный номер кластера определяет положение относительно начала файла, а логический номер — положение относительно начала тома.

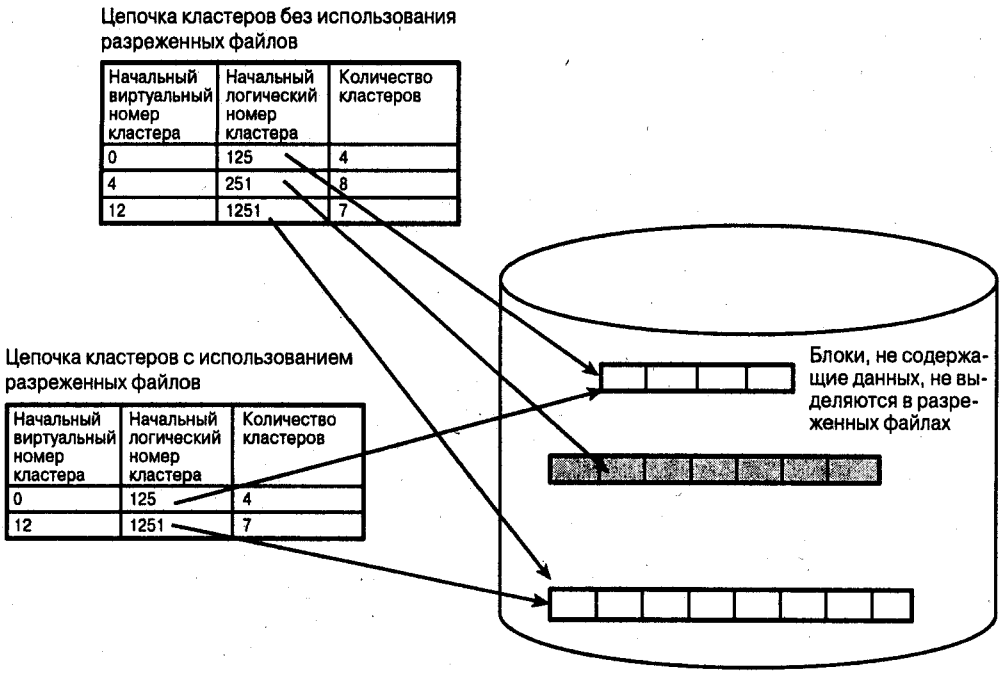


Рис. 6.6. Выделение кластеров для разреженных файлов в NTFS

На рис. 6.6 показаны две цепочки кластеров: для неразреженного файла и для того же файла, но разреженного. Цепочка кластеров для неразреженного файла содержит три записи. Первая начинается с нулевого значения виртуального номера кластера (указывая на начало файла), расположенного в логическом кластере 125. Вместе с этой информацией указано четыре кластера, т.е. четыре кластера расположены в непрерывном участке диска. Вторая запись указывает, что следующий фрагмент файла с виртуальным номером кластера 4 (пятый кластер файла) начинается по смещению логического номера кластера, равного 251, и имеет размер в восемь кластеров. Этот кластер на рис. 6.6 показан не так, как другие кластеры, поскольку в соответствующей области файла нет данных. Последняя запись показывает, что следующий кластер файла расположен по смещению логического номера кластера, равного 1251.

Первая запись второй цепочки кластеров практически идентична. Файл по-прежнему состоит из четырех кластеров, начиная с логического номера кластера 125. Следующая запись в цепочке указывает на последние семь кластеров файла. Виртуальный номер кластера, равный 12 (11-й кластер файла), начинается с логического номера кластера 1251. Для промежуточной области файла, не содержащей данных, запись в цепочке кластеров отсутствует.

После запроса данных из файла, NTFS обращается к записи MFT, находит соответствующий виртуальный номер кластера в файле, затем логический номера кластер и преобразует последний в смещение относительно начала тома. Если это необходимо, считываются данные из требующейся области тома. Для этого используются функции драйвера класса диска и диспетчера томов. Если логический номер кластера не выделен, в буфер данных возвращается последовательность нулей. Когда приложение записывает данные в фрагмент файла, для которого не выделен логический номер кластера, NTFS просто выделяет кластеры для этой области файла и добавляет их в цепочку кластеров. Данные затем копируются из буферов и записываются в выделенные кластеры.

Учитывая снижение стоимости жестких дисков, экономия дискового пространства с помощью разреженных файлов уже не столь актуальна. Тем не менее доступ к разреженным файлам осуществляется более эффективно за счет сокращения ввода-вывода (отпадает необходимость в получении данных, представляющих собой набор нулей).

Приложения могут указать атрибут разреженности для файла с помощью параметра `FSCTL_SET_SPARSE` функции `DeviceIoControl`. Для получения информации о разреженности файла требуется функция `GetFileAttributes`.

6.5.8 Сжатые файлы NTFS

Файловая система NTFS поддерживает *сжатие* файлов, если файлы размещены на томе с размером кластера менее 4 Кбайт. Данные сжимаются и разархивируются "на лету" в тот момент, когда приложение вызывает функции API для чтения и записи. Сжатие может быть отключено или включено для всего тома, каталога (файлов и каталогов, расположенных в этом каталоге) или отдельного файла. И в этом случае приложения могут переопределить параметры сжатия при создании файла или каталога. Если существующий том или каталог помечается как сжатый, над уже существующими файлами никаких действий не выполняется. Параметр относится только к новым файлам и каталогам, которые будут создаваться в этом каталоге.

Сжатые файлы хранятся в цепочках длиной по 16 кластеров каждая. Файловая система NTFS берет первые 16 кластеров и пытается их сжать. Если в результате сжатия получается 15 кластеров или меньше, файл становится сжатым; в противном случае дальнейшие попытки сжатия файла не предпринимаются.

При чтении файла NTFS должна определить, сжат ли он. Один из способов сделать это — проверить конечный логический номер кластера файла. Нулевое значение этого параметра указывает на сжатость файла. Вспомним, что присвоение нуля логическому номеру кластера указывает на загрузочный

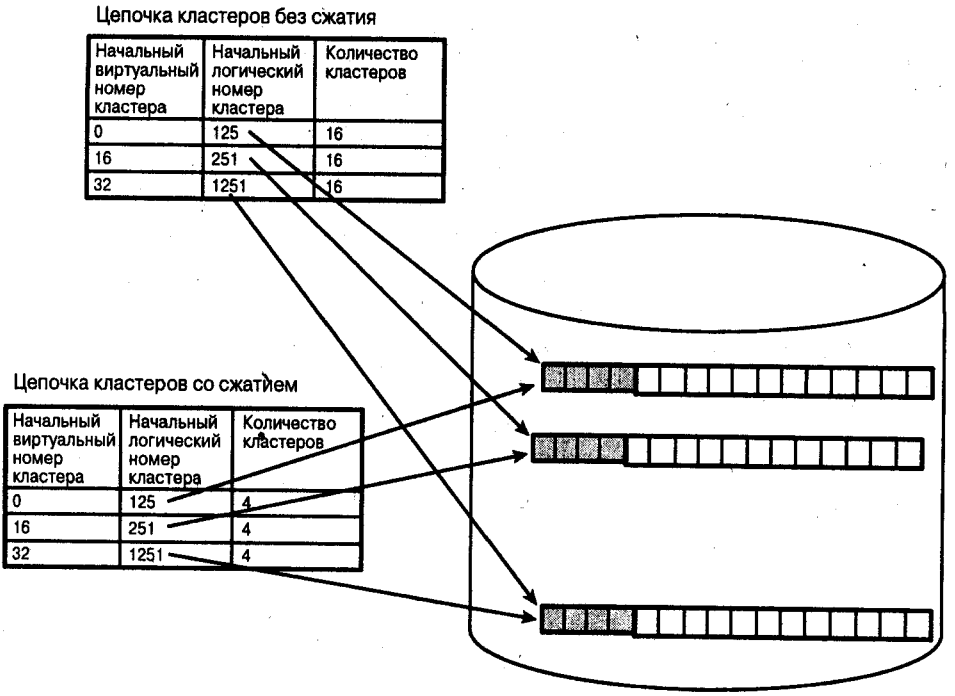


Рис. 6.7. Выделение кластеров для сжатых файлов в NTFS

сектор; т.е. такой номер никогда не может быть частью обычного файла (его цепочки кластеров). Если приложение проводит поиск в случайных участках сжатого файла, от NTFS может потребоваться распаковка целой последовательности кластеров.

На рис. 6.7 показаны две цепочки кластеров для одного и того же файла. В первой цепочке кластеров (в левой верхней части диаграммы) сжатие не применяется. Файл хранится в трех последовательностях длиной по 16 кластеров каждая. Первые 16 кластеров начинаются с логического номера кластера, равного 125, следующие 16 кластеров — с логического номера, равного 251, последние 16 кластеров — с логического номера, равного 1251. Файл занимает 48 кластеров тома. Во второй цепочке кластеров на рис. 6.7 использовано сжатие. Теперь файл содержит всего 12 кластеров в трех последовательностях. Первые четыре кластера начинаются с логического номера кластера, равного 125, следующие четыре кластера — с логического номера, равного 251, последние четыре кластера — с логического номера, равного 1251.

Получив следующий виртуальный номер кластера и количество кластеров, NTFS может определить, сжат ли файл. Сжатые данные распаковыва-

ются во временный буфер и сохраняются в кэше. При необходимости данные копируются в буфер приложения.

Сжатие требует затрат ресурсов центрального процессора и замедляет ввод-вывод данных, что приводит к снижению производительности. На фоне стремительного снижения цен на жесткие диски сжатие не всегда приводит к очевидным преимуществам; таким образом, в Windows 2000 сжатие по умолчанию отключено. В некоторых статьях базы знаний Microsoft не рекомендуется использовать сжатие, особенно для приложений, проводящих активный ввод-вывод.

Приложения могут выбрать сжатие файлов, указав параметр `FSTCL_SET_COMPRESSION` при вызове функции `DeviceIoControl`. Для получения информации о сжатости файла применяется функция `GetFileAttributes`.

6.5.9 Пользовательские квоты дискового пространства

В Windows 2000 компания Microsoft впервые предоставила технологию, с помощью которой NTFS может отслеживать и ограничивать использование дисковых ресурсов пользователями, а также выдавать им соответствующие уведомления. Это возможности NTFS, а не Windows 2000, которые недоступны в FAT и UDF. Далее представлены особенности реализации квот.

- Квоты внедряются и отслеживаются для тома и для пользователя. Все данные, связанные с квотами, хранятся системой NTFS в файле `$Quota`, который расположен в каталоге `$Extend`. Различные пользователи могут иметь разные ограничения квот. Системные администраторы не подпадают под действие квот. Более того, ограничения квот, принятых по умолчанию, относятся к новому пользователю при первом применении дисковых ресурсов.
- Системным администраторам предоставляются инструменты для управления квотами. Эти инструменты позволяют установить квоты тома в одно из трех состояний.
  - Квота отключена. NTFS сохраняет информацию, но не предпринимает никаких действий. Если включить квоты, сохраненная информация станет немедленно доступной для применения.
  - Квота включена только для отслеживания.
  - Квота включена для отслеживания и ограничения использования дисковых ресурсов, т.е. используется для ограничения дискового пространства.
- Как только пользователь превышает лимит предупреждения, установленный дисковой квотой, NTFS создает запись в журнале событий Windows NT. В течение часа для пользователя может быть создана только

одна такая запись, хотя превышений лимита предупреждения может быть несколько.

- В данных квот сообщается о размере сжатых файлов, как будто сжатые кластеры были несжатыми, а невыделенные разреженные кластеры были на самом деле выделены. При подсчете размера принимается во внимание размер дисковых кластеров, выделенных файлу. Таким образом, файл длиной 5000 байт может быть показан как имеющий размер 8192 байт, если он использует два кластера размером 4096 байт. Это сделано специально, чтобы не возникло ошибки превышения квоты при копировании файлов со сжатого тома на несжатый. При такой обработке размера файлов, квоты на различных томах можно сравнивать вне зависимости от сжатия файлов.
- Компания Microsoft предоставляет утилиты с графическим интерфейсом, которые позволяют редактировать параметры квот. Перенос параметров квот между томами позволяет установить одинаковые параметры на всех интересующих томах. Более того, утилиты поддерживают экспорт параметров квот в различных форматах, включая CSV (comma-separated values) и текст Unicode, что предоставляет дополнительные возможности для администрирования. Затем параметры можно импортировать в другую программу, например в Excel, для последующей обработки.

6.5.10 Базовые наборы атрибутов NTFS

В Windows 2000 появилась поддержка *базовых наборов атрибутов*, которые состоят из определенных пользователем метаданных, связываемых с файлом. Эти метаданные могут индексироваться средствами сервера индексации, который предоставляется в составе Windows 2000. Например, метаданные могут использоваться для обозначения автора и целевой аудитории документа. После этого можно выполнять поиск документа по определенным пользователем тегам или метаданным. NTFS рассматривает файлы как набор пар "атрибут–значение". Определенные пользователем свойства сохраняются как дополнительные атрибуты файла.

6.5.11 Владение файлом

В Windows 2000 каждому пользователю, группе и компьютеру с учетной записью в домене присваивается идентификатор безопасности (security identifier — SID). Все внутренние проверки системы безопасности выполняются с помощью идентификатора безопасности. Файловая система NTFS в Windows 2000 может сканировать MFT и идентифицировать все файлы, принад-

лежащие определенному пользователю. Для этого существует идентификатор безопасности определенного пользователя. Одним из применений SID является удаление всех файлов после удаления идентификатора пользователя.

6.5.12 Расширенная проверка списков управления доступом

Файловая система NTFS в Windows NT 4.0 отслеживала списки управления доступом для файлов и каталогов. Если пользователь владел 50 файлами, одинаковые списки управления доступом создавались 50 раз, по одному для каждого файла. NTFS в Windows 2000 хранит списки управления доступом в каталоге и выполняет их индексацию. Таким образом, в описанном случае список управления доступом будет создан один раз и каждый файл будет владеть "указателем" на этот список управления доступом. В результате требования к объему свободного дискового пространства сокращаются. Кроме того, система проверки ACL реализована более эффективно.

Внесенные изменения обеспечивают массовые проверки списков управления доступом, которые проводит служба индексации. Если пользователь выполняет поиск, служба индексации подготавливает список файлов и возвращает его пользователю. При этом служба индексации проверяет ACL и удаляет все файлы, к которым пользователь не имеет доступа. Таким образом, пользователь может просматривать только те файлы, к которым ему разрешен доступ.

Новый механизм поддерживает и другие варианты использования, например определение действий, которые пользователь может выполнять над заданным набором файлов.

6.5.13 Журнал изменений, журнал USN и файл журнала изменений

Начиная с Windows 2000 файловая система NTFS предоставляет разработчикам приложений возможность надежно отслеживать изменения файлов и каталогов. Для этого используются уникальные порядковые номера (Update Sequence Number — USN). Эта необязательная служба NTFS проектировалась специально для разработчиков программ по управлению подсистемами хранения данных. Номера USN позволяют приложениям индексировать содержимое подсистемы хранения, реплицировать файлы и работать с системой управления иерархическим хранилищем данных. Для всех изменений, внесенных в файл или каталог, в файл журнала вносится соответствующая запись. Каждая запись имеет уникальный номер — USN. Такая запись создается при возникновении следующих событий:

- создание или удаление файла;

- создание или удаление каталога;
- изменение, удаление или добавление данных в поток данных файла (любой из именованных или неименованных потоков);
- изменение (включая добавление и удаление) атрибутов файла или каталога.

Эти записи изменений хранятся в файлах журнала `$Extend` или `$UsrJrnl`. Разреженный файл может храниться в течение нескольких перезагрузок системы. Для ограничения размера файла более старые записи удаляются блоками по 4 Кбайт. Таким образом, приложение может потенциально не получить информации обо всех происшедших изменениях. Однако предоставляемый API позволяет приложениям определять недоступность некоторых записей журнала. В подобном случае приложение может выполнить соответствующие действия, к которым относится полное сканирование всего тома. Для повышения производительности значение USN на самом деле представляет смещение в файле, указанное в соответствующей записи журнала. Если информация не была потеряна, то приложение последовательно запрашивает все записи журнала и определяет файлы и каталоги, в которые вносились изменения.

Описанная технология не только позволяет получить список изменившихся файлов и каталогов, но и указывает причину внесения изменений. Существует три типа изменений.

1. Изменения, внесенные приложениями.
2. Изменения, внесенные приложениями управления подсистемой хранения данных (например, системой управления иерархическим хранилищем) и приложениями репликации.
3. Изменения, внесенные приложениями, которые создают вспомогательные данные на основе первичных данных, содержащихся в файле. Примером может служить программа обработки изображений, создающая уменьшенную копию изображения для предварительного просмотра.

Метод определения причины изменений базируется на том, что приложения могут выбирать и игнорировать изменения определенного рода.

Приложения могут запускать службу журнала изменений, передав параметр `FSCTL_CREATE_USN_JOURNAL` функции `DeviceIoControl`. Для чтения записей USN необходимо воспользоваться параметром `FSCTL_QUERY_USN_JOURNAL`.

6.5.14 Переименование потоков NTFS

Файловая система NTFS в Windows NT обладает возможностью применять несколько потоков данных одного файла. В качестве примера програм-

мы, использующей несколько потоков данных, можно указать сервер Windows NT Macintosh. Потоки создаются с помощью функции `CreateFile` и удаляются функцией `DeleteFile`. Обратите внимание: при копировании файлов с дополнительными потоками на том с файловой системой FAT, которая не поддерживает использования нескольких потоков, именованные потоки данных теряются.

До появления Windows 2000 не существовало способа переименования потока данных после его создания. Можно было создать новый файл с новым именованным потоком данных и скопировать содержимое старого файла в новый, включая содержимое старого именованного потока данных, но этот подход весьма неэффективен. Файловая система NTFS в Windows 2000 поддерживает функции, позволяющие приложениям переименовывать потоки данных.

6.5.15 Идентификаторы объектов и отслеживание ссылок

Операционная система Windows 2000 поддерживает отслеживание ссылок. Ссылки могут быть ярлыками для файлов или объектами OLE (например, для документов Excel или PowerPoint), встроенными в файл. Приложение может отслеживать ссылки, даже если источник ссылки переместится в другое место. Примеры такого переноса представлены далее.

- Перенос документа, являющегося источником ссылки, в пределах одного тома Windows NT.
- Перенос документа, являющегося источником ссылки, между томами в пределах одного сервера под управлением Windows NT.
- Перенос документа, являющегося источником ссылки, с одного сервера под управлением Windows NT на другой сервер Windows NT в пределах одного домена.
- Перенос целого тома, содержащего источник ссылки, с одного сервера под управлением Windows NT на другой сервер в пределах одного домена.
- Переименование сервера под управлением Windows NT, на котором хранится источник ссылки.
- Переименование документа, который является источником ссылки.
- Любая комбинация перечисленных выше действий.

Все эти возможности основаны на требовании к размещению источника и точки назначения ссылки на томах с NTFS под управлением Windows 2000.

Каждый файл в Windows 2000 (и более новых версиях Windows NT) имеет необязательный уникальный идентификатор объекта (16-разрядная структура $OBJECT_ID, описанная в разделе 6.5.3). Для отслеживания файла приложение ссылается на файл по уникальному идентификатору объекта. Если ссылка больше не указывает на необходимый файл (когда файл был перемещен), операционной системой вызывается служба пользовательского режима для отслеживания ссылок. Служба методом проб и ошибок пытается найти необходимый файл, воспользовавшись идентификатором объекта.

Для программного использование идентификаторов объектов и отслеживания ссылок доступны перечисленные ниже параметры функций API.

- Чтобы создать идентификатор объекта для файла или каталога, приложение должно воспользоваться параметром FSCTL_CREATE_OR_GET_OBJECT_ID функции управления файловой системой.
- Чтобы удалить идентификатор объекта для файла или каталога, приложение должно воспользоваться параметром FSCTL_DELETE_OR_GET_OBJECT_ID функции управления файловой системой.
- Чтобы запросить идентификатор объекта для файла или каталога, приложение должно воспользоваться параметром FSCTL_CREATE_OR_GET_OBJECT_ID функции управления файловой системой.

6.5.16 Улучшения утилиты CHKDSK

В Windows 2000 файловой системе NTFS требуется меньшее количество запусков утилиты CHKDSK, а также сокращено время работы этой утилиты. Безусловно, эффективность улучшений связана с размером тома и конкретным типом повреждения. Для томов с миллионами файлов ускорение на порядок работы утилиты CHKDSK более чем вероятно.

6.5.17 Индексация содержимого файловой системы

Операционная система Windows 2000 Server содержит интегрированную службу индексирования, которая взаимодействует со служебными утилитами. Особенности службы индексирования описаны ниже.

- Доступ к функциям индексации возможен с помощью диалогового окна Найти файлы и папки (Find files or folders) в программе Проводник (Windows Explorer).
- Служба индексирования поддерживает индексирование содержимого файлов и атрибутов файлов, включая определенные пользователем атрибуты.

- Служба индексирования индексирует автономные файлы, которыми управляют службы удаленных хранилищ данных (RSS).
- На томах с NTFS служба индексирования использует журнал изменений для определения, какие файлы изменились с момента последней индексации.
- Служба индексации применяет механизм массовой проверки списков управления доступом и сообщает только о тех файлах, к которым пользователь может получать доступ. Файлы, к которым пользователю доступ запрещен, в результатах поиска не отображаются.
- Служба индексирования может работать на томах с файловой системой FAT. Однако по сравнению с томами NTFS служба работает не так эффективно, поскольку недоступны расширенные возможности NTFS, например журнал изменений.

6.5.18 Тома NTFS, предназначенные только для чтения

Начиная с Windows XP, файловая система NTFS может работать с томами, поддерживающими только чтение. При этом сам том отмечается как предназначенный только для чтения. NTFS по-прежнему проверяет файл журнала изменений, и, если в журнале указано, что некоторые изменения необходимо провести повторно, запрос на монтирование тома завершается неудачей. Тома, предназначенные для чтения, имеют важную сферу применения, включая монтирование нескольких версий одного тома, которые созданы с помощью технологии моментальных снимков.

6.5.19 Фрагментация и дефрагментация NTFS

При фрагментации файл хранится в несоседних наборах кластеров. Файл размером 64 Кбайт будет занимать 16 кластеров, каждый из которых имеет размер 4 Кбайт. Если все кластеры будут находиться рядом, то для преобразования номера LCN файла в VCN потребуется только одна запись MFT. Когда же диск фрагментирован до такой степени, что файл хранится в 16 несоседних кластерах, в MFT будет храниться 16 записей, каждая из которых будет связывать соответствующий номер LCN с VCN. Увеличение фрагментации приводит к снижению производительности системы. Однократное позиционирование головки диска и чтение 16 кластеров выполняется намного быстрее, чем 16 позиционирований с чтением одного кластера за раз.

Фрагментация может возникать по ряду причин. Сразу после создания NTFS компактно расположена на томе и содержит:

- таблицу MFT в начале тома;

- свободное пространство для расширения MFT;
- системные и пользовательские файлы;
- дополнительное свободное пространство.

Фрагментация возникает в результате работы файловой системы и приложений. Некоторые причины описаны ниже.

- Инсталляция пакета обновлений Windows NT, в процессе которой создаются новые файлы и удаляются старые. На идеально дефрагментированном диске[5] выделение новых файлов начинается с начала свободного дискового пространства. При удалении старых файлов остаются небольшие свободные области, окруженные пространством, занятым файлами.
- Активная работа других приложений, например сохранение временного файла в Word или Excel, удаление старого файла и переименование временного файла с помощью имени только что удаленного файла.
- Работа приложений, которая может привести к выделению дискового пространства, на самом деле не используемого приложением. Хорошим примером служит документ OLE, содержащий набор файлов Microsoft Office, например документ Word со слайдом PowerPoint и электронной таблицей Excel. При изменении одного из этих компонентов, новая версия сохраняется в конце файла, а старая отмечается как удаленная, но все еще остается внутри файла документа. Компания Microsoft предоставляет драйвер фильтрации точек повторной обработки, который поддерживает функцию NSS (Native Structured Storage — естественное структурированное хранилище) для решения этой проблемы. При этом документы Microsoft Office хранятся в различных файлах, но для приложения Office "выглядят" как один документ. Хотя эта возможность была в одной из тестовых версий Windows 2000, ее исключили из финальной версии Windows 2000[6].
- Фрагментация каталогов является еще одной проблемой. Некоторые каталоги, например с файлами приложений, редко увеличиваются или уменьшаются в размерах; другие каталоги, например `Мои документы` или `Temp`, содержат постоянно меняющийся набор файлов. Если область

[5] В технической литературе, как и в этой книге, зачастую используется понятие "дефрагментация диска", хотя на самом деле речь идет о дефрагментации тома, а не диска.

[6] Это живой пример, иллюстрирующий, что некоторые возможности, описанные в книге, относятся к будущим версиям Windows. Компания Microsoft неоднократно отмечала, что для изучения новых функций Windows следует использовать не прогнозы и предварительные описания, а реально выпущенную версию этой операционной системы.

MFT для таких каталогов выделяется на ранних этапах работы системы, каталог будет иметь множество записей MFT, потенциально расположенных в различных областях таблицы.

В Windows NT 4.0 появилась поддержка нескольких программных интерфейсов приложений для дефрагментации. Эти интерфейсы позволяли приложениям запрашивать данные о расположении файлов и управлять ими. Интерфейсы работали как с файловой системой FAT, так и с NTFS и позволяли создавать приложения дефрагментации без тщательного изучения структуры диска. Кроме того, существуют программные интерфейсы приложений для считывания таблицы MFT в том виде, в каком она хранится на диске. Документация на такие API предоставляется на Web-узлах сторонних компаний, однако не предлагается самой Microsoft. Возможно, Microsoft планирует вносить изменения в формат и структуру интерфейсов в будущем.

В Windows 2000, Windows XP и Windows Server 2003 компания Microsoft успешно расширила поддержку дефрагментации. Ниже описаны некоторые особенности Windows XP и Windows Server 2003.

- Поддержка дефрагментации MFT. Первые 16 записей MFT переместить невозможно, впрочем, они и без того обычно не фрагментируются. Единственным исключением является корневой каталог.
- Поддержка дефрагментации дисков, размер кластеров которых превышает 4 Кбайт.
- Использование зоны MFT для временных файлов дефрагментации. Если диск заполнен целиком и пользователь или администратор пытаются провести дефрагментацию, временно понадобится дополнительное дисковое пространство. В более старых версиях Windows операция дефрагментации завершалась неудачно при отсутствии свободного дискового пространства за пределами зоны MFT, даже если в зоне MFT дискового пространства было достаточно.
- Дефрагментация потока данных, принятого по умолчанию, а также данных точек повторной обработки и списков атрибутов. Точки повторной обработки и списки атрибутов могут быть открыты, как и обычные именованные потоки данных.
- Дефрагментация области между логическим и физическим окончаниями файла, что обозначено выделенными кластерами. Такие приложения, как приложения резервного копирования и восстановления, предварительно выделяют кластеры для файла и устанавливают действительный размер файла с помощью функции `SetFileValidData`. В Windows 2000 можно дефрагментировать только действительные данные

файла, а дефрагментация области между логическим окончанием файла и конечными кластерами, выделенными файлу, не поддерживается.

- Предотвращение дефрагментации открытых файлов, которое возможно, если приложения открывают файлы, указывая новый параметр FSCTL (`FSCTL_MARK_HANDLE` с параметром `FSCTL_MARK_HANDLE_PROTECT_CLUSTERS`).
- Дефрагментация шифрованной файловой системы без чтения содержимого файлов (т.е. дешифрации). Это позволяет закрыть брешь в системе безопасности, которая позволяет дешифровать файлы в системный кэш, откуда их может прочитать злоумышленник.

6.5.20 Шифрованная файловая система

В Windows 2000 поставляется шифрованная файловая система (Encrypted File System — EFS), позволяющая закрыть основную брешь безопасности Windows. Файловая система NTFS обеспечивает безопасность, если приложения получают доступ к файлам средствами самой NTFS. Однако злоумышленник, получивший физический доступ к серверу, может перезагрузить компьютер с помощью другой операционной системы и получить доступ к диску в "непосредственном режиме" с помощью другой файловой системы. Для исключения подобной возможности в Windows 2000 появилась EFS, которая обеспечивает шифрование данных перед записью на жесткий диск. Шифрованная файловая система может быть включена для отдельных каталогов или отдельных файлов. С другой стороны, версии Windows $9x$ позволяли использовать шифрование для отдельных разделов.

В шифрованной файловой системе используется симметричная и асимметричная криптография. Архитектура файловой системы поддерживает и другие алгоритмы шифрования. Данные могут быть дешифрованы с помощью того же ключа, так как DES (Data Encryption Standard) — это симметричный шифратор.

На рис. 6.8 показано, что данные шифруются с помощью случайно сгенерированного 12-разрядного ключа. Для шифрации используется один из вариантов алгоритма DES. На первом этапе показано шифрование данных файла с помощью случайно сгенерированного ключа, на втором — шифрование случайно сгенерированного ключа с помощью открытого ключа пользователя. После этого ключ сохраняется в поле `Data Decryption` в атрибутах файла. Это поле используется для дешифрации, как отмечается далее в этом разделе. Наконец, случайно сгенерированный ключ шифруется еще раз с помощью открытого ключа *агента восстановления*. Агентом может быть системный администратор или другой пользователь, обладающий соответствующими полномочиями. Генерация поля `Data Recovery` показана на третьем

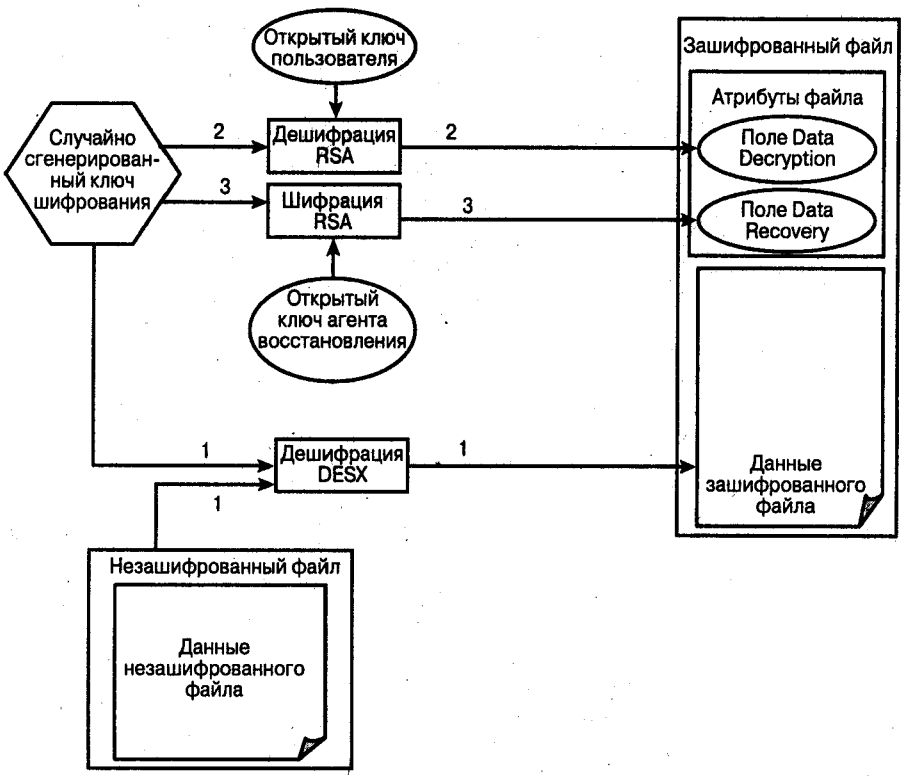


Рис. 6.8. Схема шифрации файлов в EFS[7]

этапе (см. рис. 6.8). Поле `Data Recovery` предоставляет вторичный способ получения данных, если пользователь окажется недоступным или вдруг решит сделать данные невосстановимыми.

При чтении файла считывается содержимое поля `Data Decryption` и дешифруется закрытым ключом пользователя (первый этап на рис. 6.9) для получения 128-разрядного ключа, необходимого для дешифрации данных файла. После этого данные дешифруются с помощью 128-разрядного ключа (второй этап). На рис. 6.9 приведен необязательный третий этап, на котором восстанавливается 128-разрядный ключ шифрации/дешифрации. Для этого проводится дешифрация поля `Data Recovery` (а не поля `Data Decryption`).

На рис. 6.10 показана архитектура EFS. Драйвер EFS — это драйвер фильтрации файловой системы, который расположен уровнем выше NTFS. (Шиф-

[7]Фактически стандартом асимметричного шифрования ключа является технология RSA, а стандартом симметричного шифрования — DESX. Более подробная информация представлена на Web-узле по адресу: `http://www.rsasecurity.com/rsalabs/faq/3-1-1.html`.

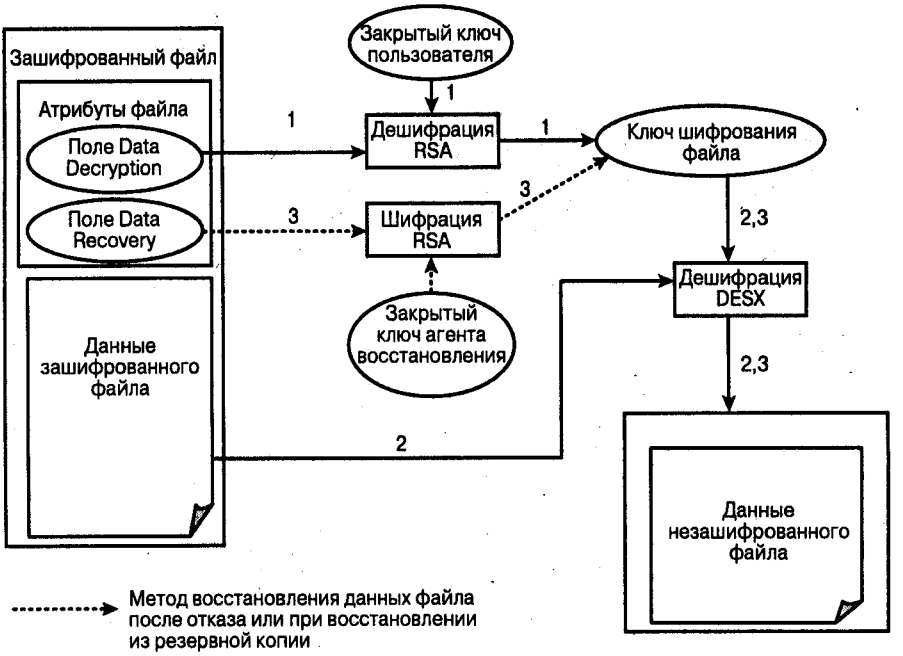


Рис. 6.9. Схема дешифрации файлов в EFS

рованная файловая система не поддерживается другими файловыми системами, включая FAT.) Драйвер обрабатывает рабочие вызовы, которые называются FSRTL (File System Run-Time Library) и используются для чтения, записи и открытия/закрытия зашифрованных файлов. Эти вызовы применяют средства NTFS для чтения или записи метаданных, относящихся к шифрованию, например полей `Data Decryption` и `Data Encryption`.

Служба EFS обеспечивает работоспособность функций шифрации/дешифрации и генерацию шифрованных ключей с помощью инфраструктуры API шифрования, входящей в Windows NT. Служба EFS взаимодействует с драйвером EFS с помощью локальных вызовов процедур (LPC), предоставляемых операционной системой.

В отличие от Windows Server 2003, операционная система Windows 2000 не поддерживает использование шифрованного файла несколькими пользователями. Тем не менее симметричный ключ можно зашифровать несколько раз открытыми ключами нескольких пользователей.

Программный доступ к шифрованным файлам обеспечивается функциями API `EncryptFile` и `Decrypt File`.

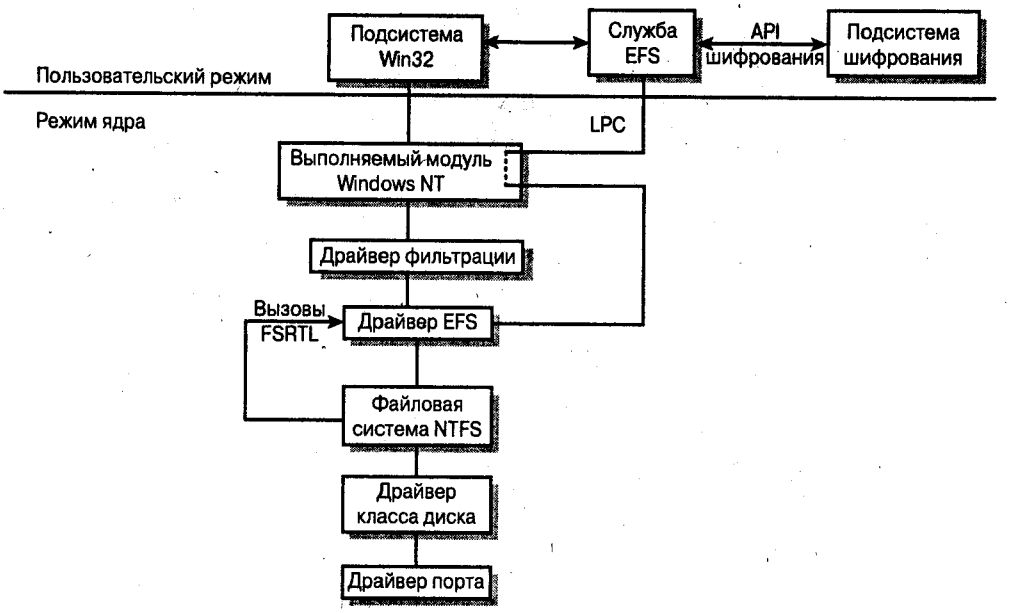


Рис. 6.10. Архитектура EFS

6.5.21 Закрепленные ссылки NTFS

Файловая система NTFS поддерживает два типа ссылок: модифицируемые (soft) и закрепленные (hard). Обратите внимание, что в Windows Server 2003 закрепленные ссылки поддерживаются только для файлов, а модифицируемые — только для каталогов. Модифицируемые ссылки реализуются с помощью *точек повторной обработки* (reparse points). Эта технология рассматривается в разделе 6.5.22.

Закрепленные ссылки позволяют одному файлу иметь несколько имен путей. Использование таких ссылок возможно только для файлов и не поддерживается для каталогов. Закрепленные ссылки могут применяться, например, в нескольких компилируемых проектах, где изменения в ряде файлов заголовков должны отражаться во всех проектах одновременно. Альтернативой закрепленным ссылкам может служить использование нескольких копий файла. Закрепленные ссылки реализуются с помощью одной записи MFT, в которой размещено несколько атрибутов с именем файла. Вызов `CreateHardLink` из Win32 API позволяет создать закрепленные ссылки и в качестве параметров принимает путь к существующему файлу и имя еще не существующего файла.

Закрепленные ссылки поддерживаются в NTFS еще со времен Windows NT 3.*x*, так как требовались для подсистемы POSIX. Последним изменени-

ем стало предоставление открытого доступа к API для создания и удаления закрепленных ссылок. Файлы удаляются после удаления последнего имени, связанного с этим файлом. Другими словами, если файл имеет две закрепленные ссылки, а именно `link1.doc` и `link2.doc`, удаление `link1.doc` не приведет к удалению `link2.doc`.

6.5.22 Точки повторной обработки

Это относительно новая функция NTFS и подсистемы ввода-вывода Windows NT. Точки повторной обработки представляют способ реализации следующих функций:

- точки монтирования томов;
- точки соединения каталогов;
- хранилище SIS (Single Instance Storage);
- удаленное хранилище (HSM).

В этом разделе подробно рассматривается архитектура точек повторной обработки. В разделах 6.5.22.1–6.5.22.4 описываются методы применения точек повторной обработки, показанных ранее.

Обратите внимание, что в этом разделе точки повторной обработки рассматриваются как неотъемлемая часть файловой системы NTFS. Хотя FAT не поддерживает точек повторной обработки, независимый поставщик программного обеспечения или компания Microsoft могут создать файловую систему, которая отличается от NTFS, но поддерживает точки повторной обработки. Такая задача далеко не тривиальна, однако использовать при этом три перечисленных ниже компонента обязательно.

1. Файловая система, например NTFS.
2. Подсистема ввода-вывода и Win32 API.
3. Утилиты и инструменты.

Компания Microsoft провела необходимую работу во всех трех областях; таким образом, новая файловая система вполне может содержать точки повторной обработки.

Точки повторной обработки — это объект каталога или файла NTFS. Приложение может создавать их, управлять ими и удалять их с помощью функций Win32 API в целом и функций `CreateFile`, `ReadFile` и `WriteFile` в частности. Набор функций Win32 API предоставляет приложению возможность создания определенных пользователем атрибутов для файлов и каталогов. Точки повторной обработки можно воспринимать как определенные пользователем атрибуты, которые обрабатываются особым образом. Это подразумевает обеспечение уникальности некоторых фрагментов объекта атрибута

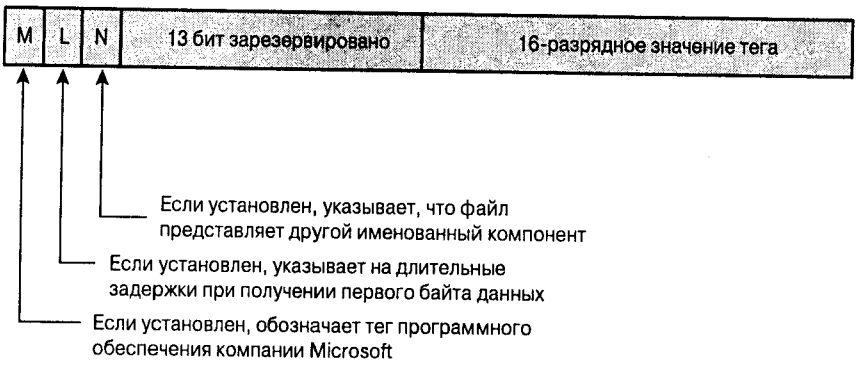


Рис. 6.11. Тег точки повторной обработки

и обработку в подсистеме ввода-вывода. Независимый поставщик программного обеспечения должен создать:

- утилиты пользовательского режима, позволяющие создавать и удалять точки повторной обработки, а также управлять ими;
- драйвер фильтрации файловой системы, который реализует функции, относящиеся к точкам повторной обработки.

Точки повторной обработки состоят из компонентов, описанных ниже.

- Уникальный тег размером 32 бит, назначаемый Microsoft. Независимые поставщики программного обеспечения могут запросить назначение уникального тега. Тег точки повторной обработки обладает хорошо определенной структурой (рис. 6.11).
  - Флаг M указывает, предназначен ли тег для драйвера устройства Microsoft.
  - Флаг L указывает, должен ли драйвер использовать большие временные задержки при получении первого байта данных. Примером служит система HSM, в которой получение данных из удаленного источника связано с длительными задержками.
  - Флаг N указывает, является ли файл или каталог псевдонимом другого файла или каталога.
  - Зарезервированные флаги.
  - Фактическое значение 16-разрядного тега.
- Двоичный объект данных размером 16 Кбайт. Файловая система NTFS делает этот объект доступным драйверу устройства стороннего разработчика в качестве элемента операции ввода-вывода, проводимой с точкой повторной обработки.

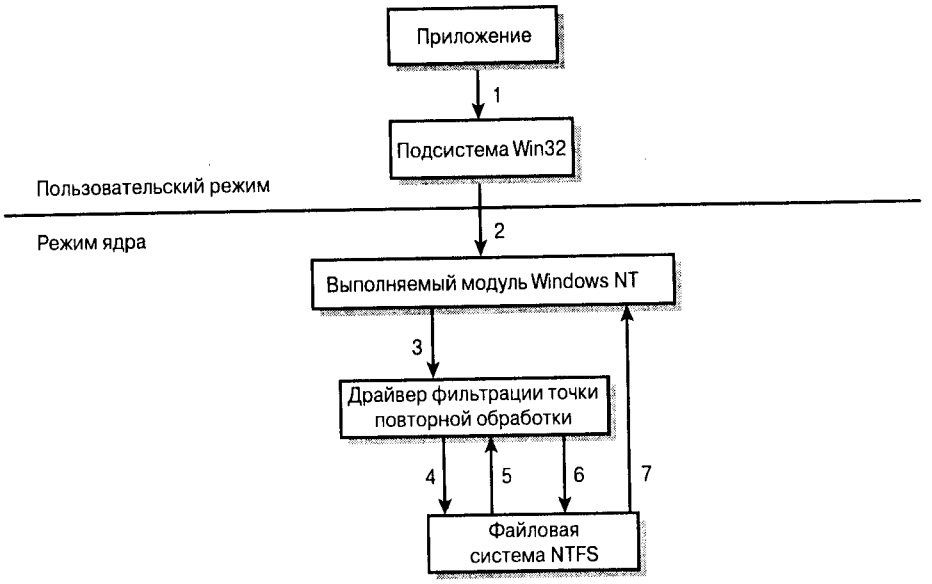


Рис. 6.12. Архитектура точек повторной обработки

На рис. 6.12 показана последовательность операций и реализация точек повторной обработки. Предположим, у пользователя есть необходимые права для выполнения запрошенной операции. Чтобы не усложнять рис. 6.12, на нем представлен только один драйвер фильтрации файловой системы.

Далее приведена последовательность действий для реализации функций точек повторной обработки, которые показаны на рис. 6.12.

1. С помощью подсистемы Win32 приложение запрашивает открытие файла.
2. После определенных проверок подсистема Win32 направляет запрос выполняемому модулю NT.
3. Диспетчер ввода-вывода Windows NT создает пакет запроса ввода-вывода (IRP) с запросом на открытие (`IRP_MJ_OPEN`). Обычно этот запрос переходит драйверу NTFS. Так как в процессе участвует драйвер фильтрации, например драйвер фильтрации точки повторной обработки, диспетчер ввода-вывода отправляет запрос драйверу фильтрации, предоставляя ему возможность предварительной обработки пакета IRP до того, как пакет попадет на обработку драйверу NTFS.
4. Драйвер фильтрации точки повторной обработки указывает процедуру завершения в своем фрагменте пакета IRP и отправляет пакет драйверу NTFS.

5. Пакет IRP достигает файловой системы. Она принимает запрос IRP_MJ_OPEN, находит файл или каталог и отмечает тег точки повторной обработки, который связан с этим каталогом или файлом. Файловая система NTFS размещает тег точки повторной обработки и данные в пакете IRP и неудачно завершает обработку пакета IRP с возвратом специального кода ошибки.
6. Подсистема ввода-вывода вызывает каждый драйвер фильтрации (по одному разу), который зарегистрировал процедуру завершения для этого пакета IRP. Каждая процедура завершения отмечает код ошибки; если это специальный код ошибки точки повторной обработки, то процедура завершения драйвера проверяет тег точки повторной обработки, сохраненный в пакете IRP. Если драйвер не распознает тег как собственный, он вызывает диспетчер ввода-вывода для вызова следующей процедуры завершения. Предположим, что один из драйверов распознал тег повторной обработки в качестве собственного. Далее драйвер может использовать данные из пакета IRP для повторной передачи пакета IRP в обработку, внеся изменения на основе данных точки повторной обработки; например, перед повторной отправкой меняется имя файла.
7. Файловая система NTFS завершает повторную операцию IRP. Типичным примером может быть изменение пути и успешное завершение обработки запроса. Диспетчер ввода-вывода завершает обработку запроса открытия; после этого каждый драйвер фильтрации файловой системы еще раз вызывается через процедуру завершения. Драйвер отмечает, что запрос на открытие завершился успешно, и выполняет необходимую последовательность действий. Наконец обработка пакета IRP завершается, и приложение получает управление для обработки файла.

Если ни один из драйверов фильтрации не распознал тег повторной обработки, запрос на открытие файла или каталога завершается неудачно.

Некоторым приложениям требуются точки повторной обработки; в то время как другие приложения совершенно в них не нуждаются. Приложению Microsoft Office, открывающему документ Word, PowerPoint или Excel, могут не понадобиться функции точки повторной обработки, перенаправляющей запрос на другой том. Однако некоторые приложения, рекурсивно обрабатывающие дерево каталогов, должны "знать" о возможности создания цикличных путей.

Приложения могут подавлять функции точек повторной обработки. Для этого запросам `CreateFile`, `DeleteFile` и `RemoveDir` необходимо передать параметр `FILE_OPEN_REPARSE_POINT`. Вызов функции `GetVolumeInformation` возвращает флаг `FILE_SUPPORTS_REPARSE_POINTS`. Вызовы `GetFileAttributes`, `FindFirstFile` и `FindNextFile` возвращают флаг `FILE_ATTRIBUTE_REPARSE_`

POINT для индикации наличия точки повторной обработки. Точки повторной обработки создаются с помощью параметра FSCTL_SET_REPARSE_POINT функции DeviceIoControl.

Операционная система Windows 2000 позволяет приложениям перечислять все точки повторной обработки и/или точки монтирования, расположенные в пределах тома. Для этого NTFS хранит информацию о точках повторной обработки (включая точки монтирования) в файле $Extend\$Reparse. Все точки повторной обработки на томе NTFS индексируются в файле $Index, который находится в каталоге \$Extend. Таким образом, приложение может быстро индексировать все эти точки.

6.5.22.1 Точки монтирования томов

Операционная система Windows NT 4.0 для монтирования тома или раздела требовала использования буквы диска. Это ограничение не позволяло операционной системе иметь больше 26 томов. Операционная система Windows 2000 позволяет монтировать том без буквы диска. Однако существуют некоторые ограничения:

- том может быть смонтирован только на локальный каталог; другими словами, том нельзя смонтировать на каталог сетевого ресурса;
- том может быть смонтирован только на пустой каталог;
- пустой каталог должен располагаться в разделе NTFS (поскольку только NTFS поддерживает точки повторной обработки).

Приложения, получающие доступ в каталог, который содержит точку монтирования, не воспринимают особенностей каталога, если приложение специально не запросит необходимую информацию.

В состав пакета Windows SDK включены программные интерфейсы приложений, которые поддерживают добавление и изменение точек монтирования томов. Некоторые функции перечислены ниже.

- GetVolumeInformation может использоваться для получения информации о томе, включая индикацию поддержки точек монтирования.
- FindFirstVolumeMountPoint и FindNextVolumeMountPoint используются для поиска точек монтирования томов.
- FindVolumeMountPointClose освобождает ресурсы, полученные функциями FindFirstVolumeMountPoint и FindNextVolumeMountPoint.
- GetVolumeNameForMountPoint возвращает соответствующее имя тома, в которое преобразуется имя точки монтирования.

6.5.22.2 Точки соединения каталогов

Точки соединения каталогов связаны с точками монтирования томов. Разница между ними заключается в том, что точка монтирования "превращает" каталог в новый том, а точка соединения каталогов превращает каталог в другой каталог, который расположен на том же локальном томе, что и точка соединения каталогов. Точки соединения каталогов могут создаваться с помощью утилиты `linkd.exe` или утилиты `junction.exe`, которая поставляется в наборе Resource Kit для Windows 2000 или в комплекте самой операционной системы.

6.5.22.3 Хранилище SIS

В Windows 2000 поддерживается технология SIS (Single Instance Storage) для служб удаленной установки (RIS). Службы позволяют создавать загрузочные образы и образы приложений, которые хранятся на сетевых ресурсах и доступны для клиентов. Зачастую клиенты создают несколько образов, например один для отдела проектирования, другой для бухгалтерии, третий для отдела кадров и т.д. Многие файлы дублируются в нескольких образах.

Применение символических ссылок для одновременного использования файла в нескольких установочных образах связано с опасностью переноса изменений с одного образа на все остальные. Представьте файл `.ini`, который совместно используется тремя отделами. Если проектный отдел внесет изменения в этот файл, установочные образы остальных отделов также будут содержать эти изменения. Технология SIS предоставляет способ использования единственной копии там, где это возможно. В то же время, если понадобится, копию можно автоматически разделить на несколько разных версий.

Пользователи Windows обычно применяют несколько образов для разных клиентов. Например, рабочие станции отдела проектирования настраиваются с другими параметрами, чем персональные компьютеры в бухгалтерии, и т.д. В результате создается много дублирующихся файлов, расположенных в установочных образах. Служба SIS позволяет хранить такие файлы в единственном экземпляре.

Далее перечислены основные возможности SIS.

- Обнаружение идентичных файлов, которые хранятся в нескольких экземплярах.
- Копирование файлов, которые хранятся в нескольких экземплярах, в специальное хранилище. Эта функция напоминает службу HSM, но файл всегда хранится в специальном хранилище SIS на диске.
- Реализация ссылок SIS для этих файлов. Ссылки SIS — это файлы-"заглушки" на месте оригинального файла. Файл-заглушка имеет точку

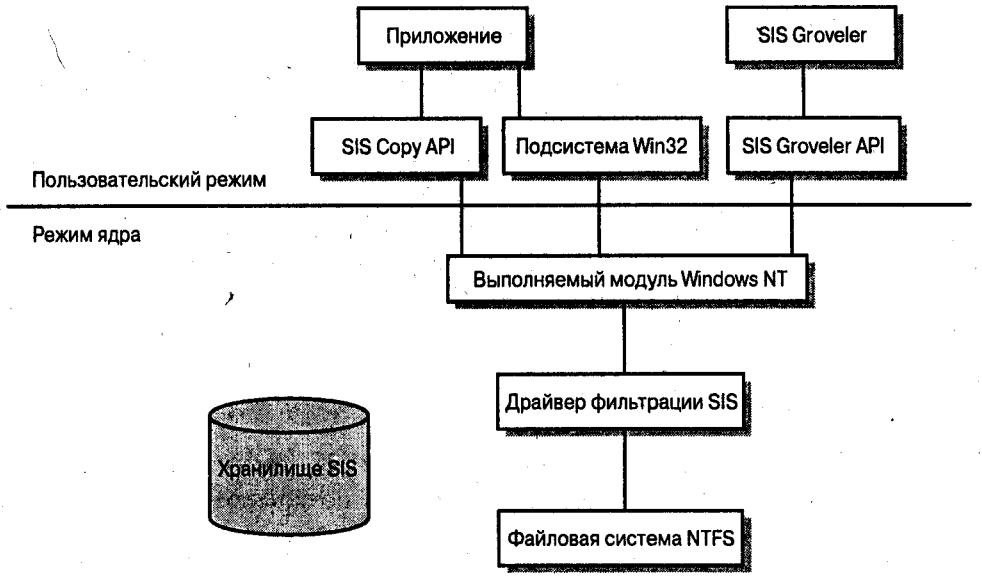


Рис. 6.13. Архитектура Single Instance Storage

повторной обработки, которая указывает на копию файла в хранилище SIS.

На рис. 6.13 демонстрируется архитектура SIS со следующими компонентами:

- хранилище SIS;
- драйвер фильтрации SIS;
- программные интерфейсы приложений SIS;
- анализатор SIS.

Служба SIS обеспечивает работу защищенного хранилища данных, которое содержит все файлы, обозначенные в качестве кандидатов SIS. Преимущество такой схемы состоит в исключении проблем, связанных с удалением, перемещением и переименованием общих файлов. Недостаток такого подхода связан с накладными расходами операции копирования. Файлы в общем хранилище содержат обратный указатель на файлы, которые они представляют.

Драйвер фильтрации файловой системы обеспечивает работу точки повторной обработки, которая предоставляет связь между файлом и копией в общем хранилище SIS. Драйвер SIS реализует две важные функции управления вводом-выводом (IOCTL).

Первая функция — `SIS_COPYFILE` — копирует файл в общее хранилище SIS и превращает исходный файл в ссылку; этот файл-ссылка на самом деле

создается как разреженный файл без данных, имеющий только запись в MFT. Файл содержит тег точки повторной обработки SIS, который указывает на реальный файл в общем хранилище SIS, содержащий данные оригинального файла. Эти функции IOCTL могут вызываться приложениями, если у них есть право чтения источника и право записи в точку назначения. Копирование выполняется в том случае, если файл еще не содержится в общем хранилище SIS. Когда исходный файл уже размещен в хранилище SIS, к нему добавляется обратный указатель на файл в общем хранилище. Одна из причин копирования файлов — возможность открыть файл приложением по идентификатору файла (ID). Если файл перемещается или переименовывается, его идентификатор сохраняется. Таким образом, при переименовании файла приложение будет открывать в общем хранилище SIS файл, а не ссылку на него. Недостатком этой функции является снижение производительности при копировании больших файлов между различными областями диска.

Второй важной функцией IOCTL, реализованной драйвером SIS, является `SIS_MERGE_FILES`, которая используется для слияния двух файлов. Это защищенная функция вызывается только анализатором SIS пользовательского режима.

Кроме определенных функций IOCTL драйвер SIS отвечает за реализацию ссылок SIS (они напоминают символьные ссылки), позволяющих приложению ссылаться на файл. При этом драйвер обеспечивает работу функций, предоставляющих данные файла из файла в общем хранилище SIS.

Анализатор SIS (SIS Groveler) отвечает за сканирование всех файлов на томе и определение дублированных файлов. Анализатор использует возможности драйвера SIS для перемещения обнаруженных дублированных файлов в общее хранилище SIS. Для обнаружения изменений в файлах применяется журнал изменений NTFS. Как только полное сканирование диска завершено, журнал изменений повышает эффективность работы анализатора и ограничивает сканирование только измененными файлами.

Служба SIS не управляет всеми томами. При запуске SIS все тома NTFS сканируются на предмет наличия папки общего хранилища SIS. Служба подключается к томам, на которых расположена эта папка. Как уже отмечалось, папка представляет собой общее хранилище SIS, создаваемое при инсталляции этой службы.

Когда приложение открывает файл, на самом деле может открываться лишь файл-ссылка SIS, в то время как реальное содержимое файла будет получено из общего файла в хранилище SIS. Рассмотрим пример, в котором файл `.ini` совместно используется тремя отделами: проектным, отделом кадров и бухгалтерией. В общем хранилище будет сохранена единственная копия файла `.ini`, и в образах будут сохранены три ссылки на этот файл. Предположим, что в проектном отделе принято решение изменить значение

параметра, сохраненного в файле .ini. Служба SIS обеспечивает изолированность других отделов от этого изменения.

Служба SIS изолирует отделы посредством копирования при закрытии, когда приложение записывает изменения в объект, который воспринимается как файл .ini для проектного отдела, и закрывает файл после внесения необходимых изменений. Причина использования операции копирования при закрытии, а не при записи, состоит в статистике, согласно которой большой процент операций записи приводит к перезаписи всего файла, а не определенного его фрагмента. Копирование при записи в такой ситуации приведет к ненужному копированию данных из существующего файла с последующей перезаписью только что скопированных данных. Если целый файл не записывается сразу, не изменившиеся фрагменты файла извлекаются из существующего общего хранилища SIS и добавляются к только что записанным фрагментам файла.

Служба SIS предоставляет API для приложений резервного копирования. Это сделано для того, чтобы все ссылки SIS не стали полноценными файлами при копировании на резервный носитель. Таким образом обеспечивается резервное копирование только одной копии данных из общего хранилища SIS с возможностью восстановления всех файлов ссылок и данных.

6.5.22.4 Технология HSM

Технология управления иерархическим хранилищем (Hierarchical Storage Management — HSM) более подробно рассматривается в главе 7. Здесь же достаточно отметить, что такие приложения с поддержкой HSM могут быть созданы поверх механизма точек повторной обработки, описанного в разделе 6.5.22.3. По сути, реализация HSM от Microsoft — пример такого использования точек повторной обработки. Служба HSM переносит файлы с дисков на другие носители, оставляя вместо них файлы-заглушки с точками повторной обработки. Как только приложение открывает этот файл, механизм точек повторной обработки вызывается для незаметного извлечения данных с другого носителя.

6.6 Файловые системы для сетей хранения данных

Сети хранения данных позволяют администраторам сформировать пул ресурсов хранения сосуществующих с группой серверов, когда отдельные ресурсы подсистемы хранения можно назначать разным серверам. Для SAN необходимо, чтобы в любой момент времени только определенный сервер мог получить доступ к определенным ресурсам подсистемы хранения. Сети хранения данных предоставляют эффективный способ переназначения ресурсов

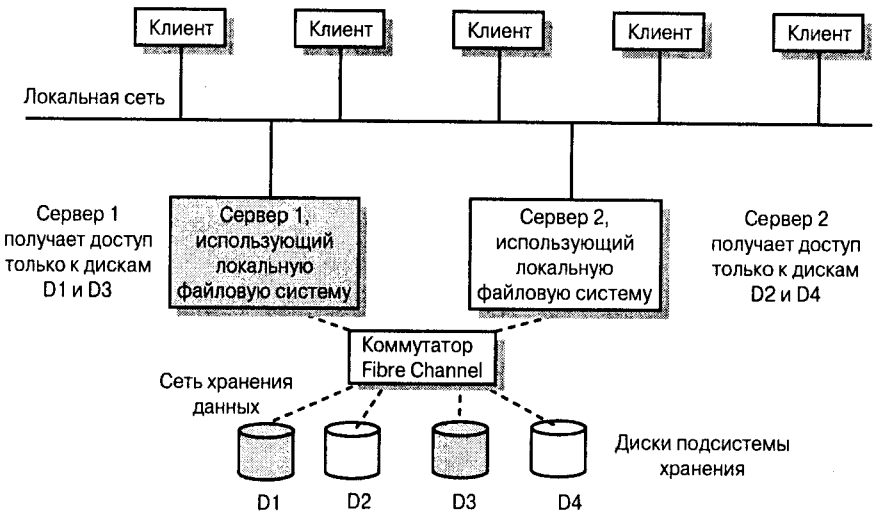


Рис. 6.14. Использование сети хранения данных с локальной файловой системой

подсистемы хранения данных между серверами. Чтобы лучше понять этот механизм, обратитесь к рис. 6.14.

На рис. 6.14 демонстрируется типичная трехуровневая архитектура сети хранения данных. На верхнем уровне находятся клиенты, получающие доступ к серверам по локальной сети. Серверы подключены к коммутатору Fibre Channel. Кроме того, к коммутатору подключено несколько дисков. Диски подсистемы хранения можно рассматривать как пул дисков, состоящий из дисков D1–D4. На рис. 6.14 показан сервер 1 и диски D1 и D3, закрашенные другим цветом, так как сервер 1 получает эксклюзивный доступ к дискам D1 и D3. Сервер 2 получает эксклюзивный доступ к дискам D2 и D4.

Сеть хранения данных обеспечивает относительно простое переключение дисков между серверами. Сети хранения не поддерживают одновременного совместного использования устройств подсистемы хранения. В контексте верхних программных уровней (в частности, файловых систем и выше) некоторые ресурсы подсистемы хранения выглядят как подключенные непосредственно к серверу. Это утверждение справедливо для сетей хранения данных на основе как технологии Fibre Channel, так и протокола IP[8].

Для одновременного совместного доступа к ресурсу подсистемы хранения (например, к тому) несколькими серверами необходима поддержка рас-

[8]Хранилища данных IP описаны в главе 8.

ширенной файловой системы, которую часто называют файловой системой сети хранения данных (SAN file system). Файловые системы SAN позволяют нескольким серверам одновременно получить доступ к одному и тому же хранилищу, предоставляя возможность эксклюзивного доступа к некоторым файлам или компонентам файлов только определенным серверным процессам на протяжении указанного промежутка времени. Внимательный читатель может заметить, что даже хранилище, подключенное к сети, позволяет совместно использовать файлы, и это верно. Но хранилище, подключенное к сети, имеет единый сервер (сервер NAS), который выполняет роль шлюза, и все команды для операций с файлами (например, открытие, закрытие, запись и блокировка) издаются этим сервером.

Сервер NAS может легко превратиться в проблемный элемент системы хранения данных. Такие сетевые файловые системы, как CIFS и NFS (см. главу 3), предоставляют совместный доступ на уровне файлов. При этом поддерживаются клиенты, которые обращаются к серверу по сетевому протоколу, например TCP/IP. Файловые системы сетей хранения данных предоставляют совместный доступ к устройствам хранения на уровне блоков. При этом поддерживаются клиенты, использующие протокол уровня блоков, например SCSI. При использовании файловых систем SAN каждый сервер использует то, что воспринимается им в качестве файловой системы на локальном диске. Но в реальности в рамках подобной "иллюзии" работает несколько серверов. При этом файловая система сети хранения данных, выполняющаяся на каждом из серверов, корректно поддерживает состояние файловой системы на томе, на котором одновременно работает несколько серверов.

Чтобы понять такую архитектуру, следует обратиться к диаграмме. На рис. 6.15 демонстрируется два сценария работы. В левой части диаграммы представлен диск NAS, к которому несколько серверов получают доступ с помощью сетевой файловой системы. В правой части диаграммы показано несколько серверов, получающих доступ к единственному диску с помощью файловой системы сети хранения данных. В первом случае каждый сервер использует сетевую файловую систему (например, SMB или NFS) для отправки запросов на сервер устройства NAS. Таким образом, устройство NAS потенциально является единственной точкой отказа, а также проблемным элементом в контексте производительности всей системы. При использовании сетевой файловой системы единственной точки отказа или проблемного элемента производительности просто не существует. Диск подсистемы хранения данных может предоставляться клиентам через схему балансировки нагрузки. Если один из серверов откажет в работе, данные диска можно будет получить, обратившись к другому диску. Конечно, при этом схема усложняется и становится дороже за счет стоимости файловой системы SAN.

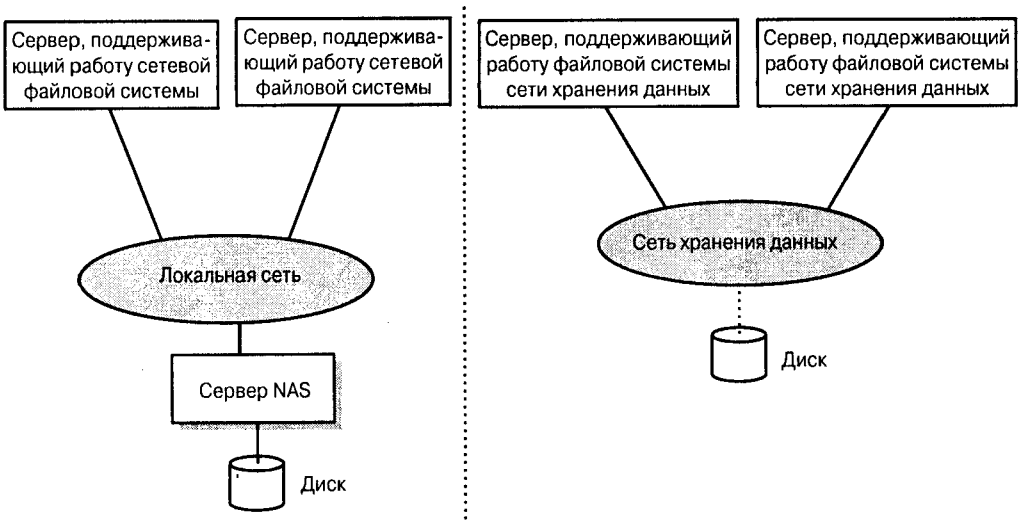


Рис. 6.15. Сценарии использования файловых систем NAS и SAN

6.6.1 Преимущества файловых систем SAN

Ниже перечислены преимущества файловых систем SAN.

- Реализация отказоустойчивых систем. Несколько серверов могут получить доступ к одному тому, поэтому серверы не являются единственным проблемным компонентом. Кроме того, относительно недорого можно обеспечить устойчивость к ошибкам на уровне тома (для этого можно воспользоваться соответствующим массивом RAID). Создание отказоустойчивого кластера обойдется намного дороже, не говоря уже о стоимости его эксплуатации.
- Предоставление высокой пропускной способности, так как обычно узким местом в производительности ввода-вывода является шина ввода-вывода на сервере. Если несколько серверов получают доступ к одному тому, ограничения на пропускную способность шины исчезают. Операции копирования данных сокращаются до минимума по сравнению с NAS. Например, для чтения данных, запрошенных клиентом, данные обычно копируются из буфера сервера в буфер TCP/IP, а на стороне клиента — из буфера TCP/IP в буфер приложения.
- Консолидация подсистемы хранения, благодаря которой пользователи могут избежать излишнего дублирования и синхронизации данных, когда сервер не в состоянии обеспечить требующуюся производительность из-за высокой нагрузки. Кроме того, серверу необходима дополнительная единица хранения с идентичными томами, так как без использо-

вания файловой системы сети хранения данных два сервера не могут одновременно получать доступ к одному и тому же жесткому диску.

- Снижение требований к накладным расходам на управление данными, что приводит к снижению общей стоимости владения (TCO). От администратора требуется управление только одним экземпляром файловой системы вместо управления несколькими экземплярами.
- Файловые системы SAN представляют собой исключительно масштабируемые системы, в которые можно добавлять новые серверы, подсистемы хранения или дополнительные устройства SAN (например, коммутаторы), если требования к архитектуре подсистемы хранения изменились.
- Приложения могут выбирать тип хранилища, наиболее подходящий их требованиям, например RAID 0, RAID 1 или RAID 5.
- Масштабируемость SAN можно сравнить с компьютерным эквивалентом конструктора LEGO. Для получения дополнительной мощности достаточно добавить сервер и настроить его на доступ к существующему диску.

6.6.2 Проблемы использования файловых систем SAN

Настоящий "инженерный подвиг" при реализации файловых систем сетей хранения данных состоит в том, чтобы достичь баланса между параллельным доступом и сериализацией. Параллельный доступ к файлам и дискам необходим для создания масштабируемой системы, которая дает возможность нескольким процессам получить доступ к одному и тому же набору файлов. При этом для обеспечения целостности пользовательских данных и метаданных файловой системы необходима синхронизация, даже если несколько пользователей и процессов получают доступ к файлам.

Обратите внимание: сложность достижения баланса между параллельностью и сериализацией присуща и другим файловым системам, не связанным с сетями хранения данных, например к NTFS. Различие заключается в том, что механизмы обеспечения корректной сериализации имеют более простую структуру и предоставляются операционными системами. Например, механизмы синхронизации, предоставленные Windows, в частности spin-блокировка и семафоры, отлично подходят для использования в файловых системах, не связанных с SAN, например в NTFS.

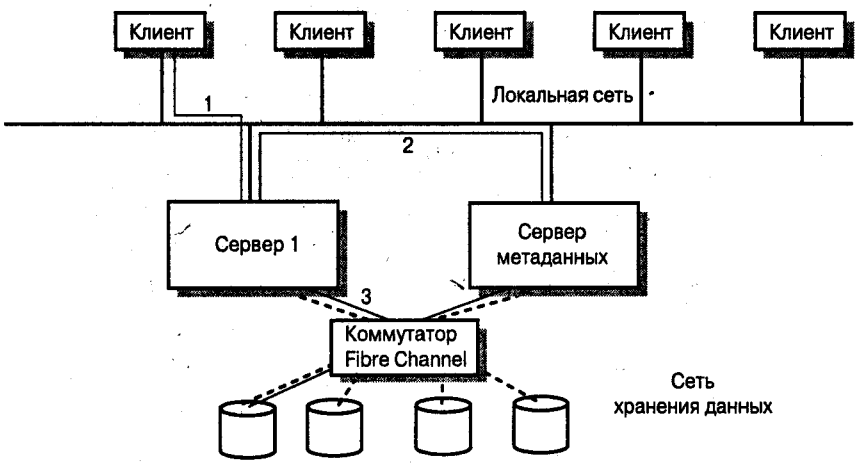
Полное описание технологии, связанной с созданием файловых систем сетей хранения данных, выходит за рамки данной книги. Достаточно перечислить ряд проблем, возникающих при создании таких файловых систем.

- Механизм синхронизации, который часто называют *диспетчером распределенных блокировок*, должен работать на нескольких компьютерах и быть устойчивым к отказам в работе и задержкам в передаче данных по сети.
- Возможны проблемы, когда отказывают в работе компьютеры, "удерживающие" ресурсы, например при блокировке файлов.
- Возможны проблемы при произвольном или непроизвольном изменении конфигурации. Например, в сети (TCP/IP или сети хранения данных) меняется топология, что приводит к прекращению доступа к некоторым узлам.
- Необходима технология обнаружения взаимных блокировок. Взаимные блокировки возникают, когда клиент удерживает некоторые ресурсы, освобождения которых ожидает второй клиент. Одновременно второй клиент удерживает ресурсы, освобождения которых ожидает первый клиент.
- Программные массивы RAID вообще не могут быть использованы или требуют сложной схемы реализации. Применению программных массивов RAID соответствует два уровня связывания. На первом уровне файловая система соотносит ввод-вывод с уровнем тома или раздела. На следующем уровне компонент программного массива RAID (диспетчер логических дисков в Windows 2000) соотносит ввод-вывод на уровне тома с вводом-выводом на уровне блоков физических дисков. Более того, для предотвращения повреждения данных необходимы методы, задействующие два уровня блокировки сетей хранения данных. Первая блокировка выполняется на уровне файловой системы, что обеспечивает сериализацию между различными системами под управлением Windows, пытающимися записывать совмещающиеся данные в один и тот же файл. Кроме того, поскольку компоненты программного массива RAID попытаются обновить данные о четности файла, диспетчер логических дисков или его аналоги, работающие на разных компьютерах, должны обеспечить механизм взаимоисключающей блокировки сети хранения данных.
- Разные клиенты могут использовать различные операционные и файловые системы. Это сложная проблема сама по себе, которую различные поставщики, включая поставщиков NAS, решают с разной степенью успешности. Кроме того, эта проблема включает ряд дополнительных сложностей.

- Обеспечение взаимодействия между разными способами отслеживания разрешений для учетных записей пользователей и групп, которые применяются в различных операционных системах.
- Обработка семантических различий между открытием и блокировкой файла в операционных и файловых системах.
- Обработка различий в соглашениях по именованию. Разные файловые системы поддерживают имена неодинаковой максимальной длины, по-разному обрабатывают регистр в именах файлов и поддерживают различные доступные символы в именах файлов.
- Файловые системы поддерживают разные временные метки. Операционная система Windows NT поддерживает три временные метки для одного файла. Файловые системы UNIX обычно поддерживают только две временные метки. Даже если количество временных меток совпадает, единицы измерения времени могут различаться.
- Файловые системы в гетерогенных системах могут иметь различные размеры; например, существуют 32- и 64-разрядные файловые системы. Все структуры требуют обеспечения корректного взаимодействия. В практической реализации структуры данных приходится преобразовывать в обоих направлениях, не забывая о необходимости дополнения до размерности в 4, 8, 16, 32 и 64 бит.

На исключительно высоком уровне файловые системы сетей хранения данных могут проектироваться двумя способами.

- Симметричный подход, при котором каждый узел в сети хранения данных равен другим узлам и механизм синхронизации распределен по всем узлам. На данный момент симметричные файловые системы для платформы Windows недоступны в продаже.
- Асимметричный подход, при котором определенный узел выступает в роли сервера метаданных и центральной точки синхронизации. Сервер метаданных отвечает за управление всеми метаданными файловой системы (например, данные о выделении дисковых кластеров). Другие серверы с файловой системой SAN обращаются к этому серверу для получения метаданных, например информации о выделении дисковых кластеров, а также идентификатора диска, идентификатора LUN и т.д. После получения метаданных серверы могут выполнять ввод-вывод непосредственно по сети хранения данных. Некоторые поставщики, например ADIC и EMC, предоставляют коммерческие продукты для платформы Windows NT, основанные на асимметричной технологии.



1 — клиент по локальной сети отправляет запрос серверу 1

2 — сервер 1 запрашивает метаданные файла у сервера метаданных. Запрос завершается успешно, и сервер 1 получает информацию о диске и блоке файла по локальной сети

3 — сервер 1 получает данные файла непосредственно от единицы хранения по сети хранения данных

Рис. 6.16. Файловая система SAN с сервером метаданных

Асимметричный подход к проектированию файловой системы SAN продемонстрирован на рис. 6.16.

1. Клиент подключается к серверу и запрашивает данные с помощью одного из протоколов, например CIFS (см. главу 3).
2. Сервер обращается к серверу метаданных и получает информацию об устройстве хранения, на котором находится необходимый файл, включая информацию о конкретном блоке диска, в котором расположен файл.
3. Затем сервер может выполнять непосредственный ввод-вывод, используя информацию, полученную от сервера метаданных.

6.6.3 Коммерчески доступные файловые системы SAN

Некоторые поставщики реализовали файловые системы SAN для платформы Windows NT на базе асимметричной технологии. В качестве примеров можно привести линию продуктов Celerra HighRoad от компании EMC,

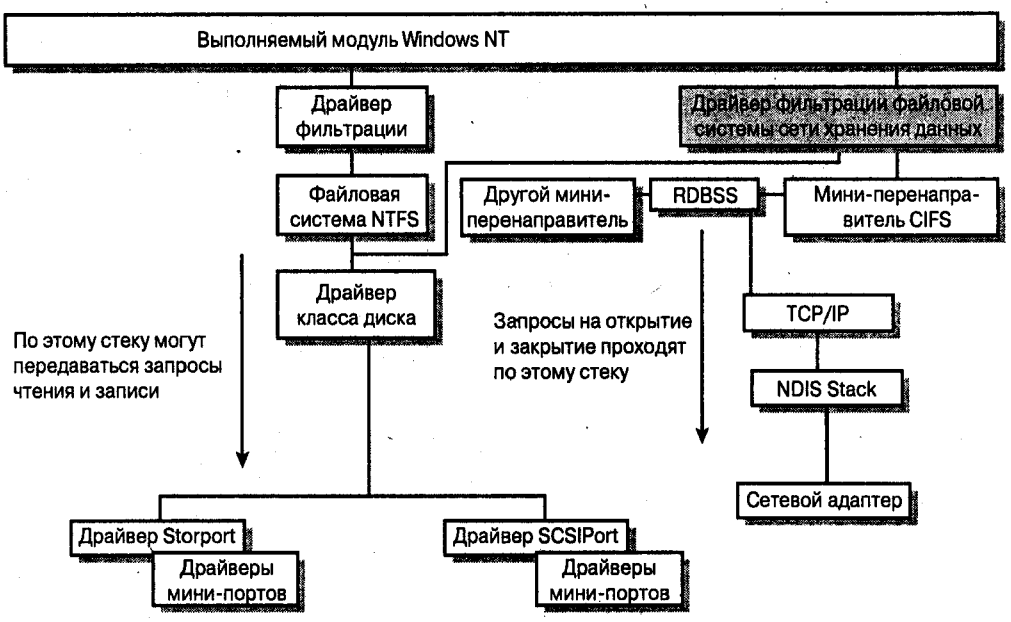


Рис. 6.17. Стек ввода-вывода файловой системы SAN в Windows NT

продукт SANergy от компании Tivoli и продукт StorNext (ранее известный как CentraVision) от компании ADIC. Все эти системы используют сервер Windows для реализации сервера метаданных и поддержки доступа к серверу метаданных со стороны вторичных серверов Windows. Некоторые системы поддерживают изолированный сервер метаданных, а некоторые — нет. Кроме того, ряд систем поддерживают другие серверы (например, Novell, UNIX или Solaris), которые могут получать доступ к серверу метаданных.

Рассмотрим более подробно реализацию описанных функций в контексте стека ввода-вывода Windows NT.

На рис. 6.17 показан стек сетевого ввода-вывода Windows NT и локального хранилища (на основе моделей драйверов Storport и SCSIPort). Драйвер фильтрации файловой системы SAN (закрашенный на рисунке другим цветом) находится уровнем выше сетевой файловой системы в целом и мини-перенаправителя CIFS в частности. Драйвер фильтрации перехватывает запросы на открытие, закрытие, создание и удаление файлов и позволяет им перемещаться по пути стандартного стека сетевой файловой системы. Перехват запросов используется для регистрации процедуры завершения. Для всех успешно открытых файлов драйвер фильтрации получает информацию о точной дорожке, секторе и блоке диска, в которых расположен файл.

Такая процедура выполняется для всех больших файлов. Некоторые системы отказываются от выполнения этой процедуры для небольших фай-

лов, поскольку служебные издержки операции чтения/записи на нескольких секторах и операция получения информации о дорожке/секторе диска для небольших файлов совпадают. Таким образом, все операции чтения и записи, которые не требуют манипуляций с метаданными файловой системы, обрабатываются непосредственно на уровне блоков между сервером и диском подсистемы хранения.

Недостаток использования централизованного сервера метаданных состоит в потенциальном превращении сервера в проблемный компонент и единственную точку отказа. Некоторые поставщики предоставляют возможность создания резервного сервера метаданных, который начинает работу в случае отказа основного сервера. С другой стороны, сервер метаданных — это единственный сервер, кэширующий метаданные, что позволяет избежать их ввода-вывода в пределах всего кластера.

6.7 Сложности практической реализации

В Windows 2000 впервые появилась поддержка динамических дисков. Следует обратить внимание на процесс улучшения поддержки динамических дисков компанией Microsoft, что понадобится при установке системы и предустановке дисков независимыми производителями компьютеров.

Архитектура точек повторной обработки должна заинтересовать независимых производителей программного обеспечения. Персонал отделов информационных технологий должен обратить пристальное внимание на поддержку этой архитектуры поставщиками решений HSM.

Отметим, что на данный момент ни один поставщик не разрабатывает и не продает файловую систему для платформы Windows NT (кроме, естественно, самой Microsoft). Но в будущем ситуация может измениться.

6.8 Резюме

Вместе с выходами новых версий Windows NT демонстрирует расширение поддержки новых технологий хранения. В Windows 2000 предоставляется поддержка динамических дисков, позволяющих интерактивно управлять томами файловых систем, не прерывая при этом доступа приложений к ресурсам тома.

Файловая система NTFS обладает богатыми возможностями, которые были расширены с выходом Windows 2000. Теперь тома NTFS можно шифровать, предотвращая несанкционированный доступ даже средствами других операционных систем. Еще одной особенностью является журнал изменений, который позволяет приложениям хранения, например агентам репликации

или приложениям HSM, быстро и эффективно идентифицировать изменившиеся файлы и каталоги. Файловая система NTFS в Windows 2000 предоставляет поддержку точек повторной обработки, которые являются основой реализации HSM, символьных ссылок, точек монтирования томов и службы SIS.

Файловые системы SAN предоставляют определенные преимущества при работе в сетях хранения данных. Некоторые поставщики реализовали файловые системы сетей хранения данных в среде Windows NT, применив подход с централизованным сервером метаданных.

Глава 7

Управление хранилищем данных

Эффективное управление хранилищами данных в сфере современных информационных технологий играет все более важную роль. Основные проблемы перечислены далее.

- Увеличение объемов хранения требует расширения административного штата, даже несмотря на то, что на одного администратора возлагается обслуживание все большего количества единиц хранения данных. Значительное сокращение стоимости дисковых хранилищ (особенно дисков начального уровня) также приводит к увеличению подсистем хранения, используемых в компаниях.
- Требования к времени простоя современных подсистем хранения постоянно ужесточаются.
- Рабочее окружение развивается от поставщика, предлагающего систему высокого уровня для больших корпораций, к поставщикам набора операционных систем и подсистем хранения, внедряемых в более мелких корпоративных средах. Проблему составляют требования к минимизации стоимости и многочисленные закрытые системы управления, предлагаемые разными поставщиками.

Как отмечается во введении, эта книга — не пошаговое руководство. Поэтому вместо описания различных утилит с графическим интерфейсом, которые могут использоваться для управления хранилищами (например, для администрирования дисков), в этой главе рассматривается архитектура и технологии для управления хранилищами данных.

Глава начинается введением в общую информационную модель управления (Common Information Model — CIM), определенную рабочей группой DMTF и принятую ассоциацией SNIA. Далее следует описание технологии WMI (Windows Management Instrumentation), которая представляет собой реализацию модели CIM от компании Microsoft. Затем рассматривается схема виртуализация хранилищ, а также технология виртуализации хранилищ от Microsoft, включая службы виртуализации дисков и связных архитектур. Далее обсуждаются методы использования API для адаптеров шины, принятые

в ассоциации SNIA и компании Microsoft, после чего рассматривается архитектура HSM, которая впервые появилась в Windows 2000. И в завершение описывается стандарт управления хранилищем от ассоциации SNIA, который изначально имел кодовое имя Bluefin. В настоящее время стандарт официально называется SMI (Storage Management Initiative).

7.1 Общая информационная модель и стандарт WBEM

Протокол SNMP (Simple Network Management Protocol) представляет собой стабильную технологию управления, имеющую свои преимущества и недостатки. Он обеспечивает эффективный мониторинг систем, однако не подходит для моделирования взаимоотношений между ними и предоставления активных методов управления хранилищами.

Учитывая необходимость в разработке новой парадигмы управления хранилищами, в 1996 году появился промышленный стандарт WBEM (Web-Based Enterprise Management). Разработка стандарта проводилась под наблюдением группы DMTF (Distributed Management Task Force, ранее известной как Desktop Management Task Force). В стандарте WBEM определена общая информационная модель (CIM), которая включает в себя такие уже существующие стандарты, как SNMP и DMI (Desktop Management Inferface).

Общая информационная модель относится не только к подсистеме хранения в целом, но к сетям и персональным компьютерам в частности, а также к моделированию внешних и внутренних взаимоотношений между структурными элементами, например между устройствами хранения и компьютерами. В модели CIM определена последовательность схем для реализации управленческой информации и взаимоотношений между компонентами. В следующем списке представлен набор схем CIM, определенных рабочей группой DMTF:

- базовая CIM;
- CIM приложений;
- CIM сети;
- CIM QoS сети;
- CIM пользователей;
- CIM систем хранения данных.

Стоит отметить, что рабочая группа DMTF определила CIM в качестве модели, а не конкретной реализации. Для того чтобы сделать CIM более привлекательной и обеспечить межплатформенную совместимость в гетероген-

ном окружении, рабочая группа DMTF разработала метод, с помощью которого гетерогенные системы могут обмениваться управляющей информацией по протоколу HTTP, применяя для хранения этих данных документы XML.

Модель CIM все еще развивается и, без сомнений, приобретет большую популярность в ближайшем будущем, особенно после недавних демонстраций некоторыми поставщиками на конференции Storage Networking World в марте 2002 года. Основная проблема реализации CIM состоит в том, что поставщикам приложений управления и аппаратных систем приходится действовать методом проб и ошибок. Поставщики аппаратного обеспечения не имеют ясного представления о том, какой инструментарий должен быть разработан, а поставщики приложений управления не знают, каких классов и инструментария стоит ожидать, чтобы все возможности были доступны постоянно, а не только в отдельных случаях (в зависимости от производителя). Тем не менее через некоторое время эта проблем будет решена.

7.2 Интерфейс WMI

Как уже отмечалось, модель CIM определена в рабочей группе DMTF и принята к использованию ассоциацией SNIA. Интерфейс WMI представляет собой реализацию модели CIM от Microsoft. Другими словами, WMI — это "CIM для Windows".

Интерфейс WMI был разработан для режима ядра и пользовательского режима. Операционная система Windows NT 4.0 SP6 поставлялась с интерфейсом WMI только для пользовательского режима. В Windows 2000, Windows XP и Windows Server 2003 интерфейс WMI реализован как в режиме ядра, так и в пользовательском режиме.

Интерфейс WMI имеет трехуровневую архитектуру (рис. 7.1).

1. Уровень приложений WMI.
2. Уровень служб WMI.
3. Уровень поставщиков WMI.

На самом верху находится уровень приложений WMI. Приложения WMI обычно используются для управления системой. Независимые поставщики программного обеспечения создают приложения управления с помощью интерфейса, предоставляемого службой WMI, или с помощью интерфейса сценариев, который размещен уровнем выше службы WMI. Несколько приложений WMI могут сосуществовать на одном и том же компьютере.

Далее следует уровень службы WMI, который включает в себя единственный компонент — собственно службу WMI. Эта служба поставляется в составе Windows NT и представляет собой реализацию схемы, основанной на модели CIM. Иногда службу WMI называют объектным диспетчером CIM, или

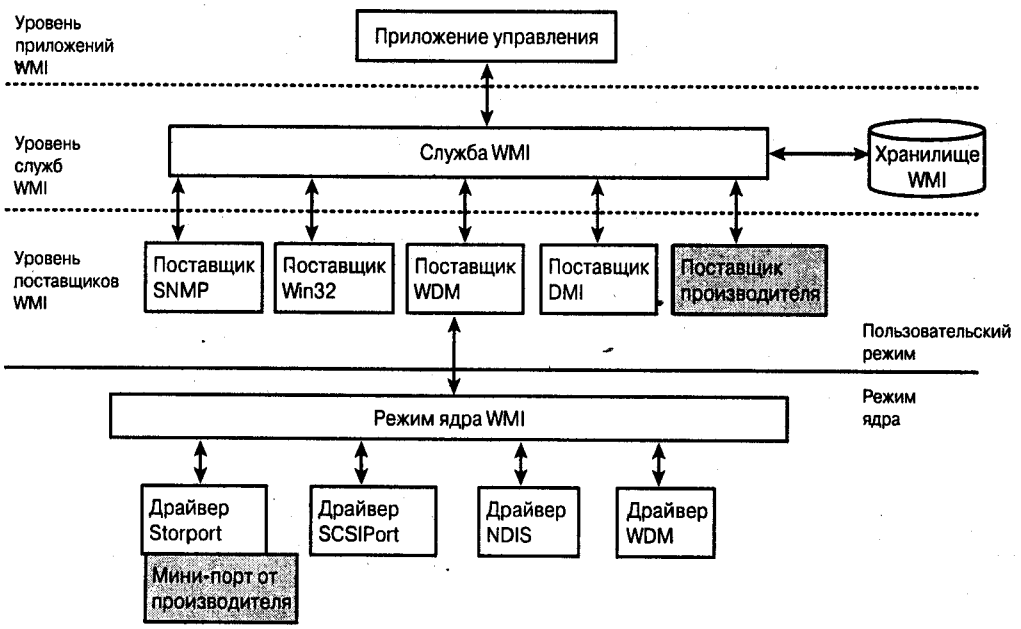


Рис. 7.1. Архитектура WMI

CIMOM (CIM Object Manager). Служба WMI предоставляет единый интерфейс для приложений управления. Источник информации и тип приложения управления не имеют значения. Другими словами, служба WMI осуществляет операции мультиплексирования и демультиплексирования запросов между приложениями и различными поставщиками. Кроме того, служба предоставляет интерфейс сценариев WMI, который позволяет с помощью любого языка сценариев, поддерживаемого интерфейсом Windows Scripting Host (Visual Basic, JScript, VBScript), получать доступ к объектам службы WMI.

Последним в иерархии представлен уровень поставщиков WMI. На этом уровне находятся различные поставщики WMI: одни из них созданы компанией Microsoft, другие предоставлены сторонними производителями. Поставщики WMI создаются с помощью набора WMI SDK, который входит в пакет Platform SDK и включает в себя обычные динамически подключаемые библиотеки пользовательского режима. Компания Microsoft создала большое количество поставщиков WMI. Производители компьютеров, программного и аппаратного обеспечения также создают различных поставщиков. Например, производитель компьютеров может создать поставщика WMI, который позволяет наблюдать за различными параметрами внутри корпуса сервера, например за температурой и скоростью вращения вентиляторов. Такой по-

ставщик WMI может даже подавать сигнал тревоги, когда открывается корпус сервера.

Многие поставщики WMI от Microsoft поддерживают управление приложениями пользовательского уровня и службами. Особое значение имеет поставщик WDM (Windows Driver Model), который позволяет предоставлять события и данные драйверов режима ядра пользовательским приложениям. Программный код режима ядра и драйвера находится под управлением кода WMI, однако от разработчиков драйверов создание поставщика WMI не требуется.

Расширение WMI для WDM предоставляет методы публикации информации и передачи событий от драйверов режима ядра. WMI может быть реализован в виде драйвера, который получает пакет запроса ввода-вывода (IRP). Кроме того, интерфейс WMI может быть реализован в виде драйвера мини-портов SCSI, Storport или NDIS. В каждом из этих случаев драйвер порта, предоставленный компанией Microsoft, выполнит необходимую трансляцию данных между пакетом IRP и интерфейсом мини-порта. Компания Microsoft реализовала интерфейс WMI в различных драйверах классов и портов. Производители также могут добавлять возможности WMI в драйверы собственных устройств. WMI реализуется в драйверах с помощью набора Windows Driver Development Kit. Интерфейс WMI внедряет новую функцию с помощью пакета `IRP_MJ_SYSTEMCONTROL` и позволяет управлять параметрами устройства.

Интерфейс WMI предоставляет массу возможностей для приложений управления. Например, приложение управления может зарегистрироваться в WMI для получения уведомлений в описанных ниже ситуациях.

- Изменение определенного управляемого объекта описанным способом, например при достижении температурой определенного значения.
- Создание нового экземпляра управляемого объекта.
- Удаление экземпляра управляемого объекта.
- Удовлетворение сложного условия (например, условий 1 и 2, но не условия 3). Обратите внимание, что контроль за условиями могут обеспечивать различные поставщики. Более того, запросы проверки сложных условий встроены в язык, который называется WMI Query Language (WQL); его синтаксис напоминает SQL. Язык WQL позволяет опрашивать управляемые объекты, однако не модифицировать их.
- Возможность расширения управляемого объекта путем создания поставщика.

Еще один важный аспект состоит в предоставлении стандартного способа дифференциации продуктов производителей программного и аппаратного

обеспечения. Производитель может создать драйвер, поддерживающий WMI, или написать поставщика WMI, который работает с необходимым компонентом конкретного устройства. Приложения управления могут динамически обнаруживать и обрабатывать новые объекты управления, которые предоставляет производитель. Это делается с помощью кода WMI созданного производителем драйвера или поставщика.

Компания Microsoft широко использует интерфейс WMI для управления подсистемами хранения и приложениями. Программы Microsoft Exchange 2000 и SQL Server 2000 находятся под управлением поставщиков WMI, специально созданных для этой цели. Управление дисками и интерфейсом iSCSI также являются примером использования интерфейса WMI.

7.3 Виртуализация хранилищ данных

Виртуализация хранилищ — это новая технология, зачастую предлагаемая в качестве панацеи от всех бед. В действительности термин *виртуализация хранилища* описывает несколько технологий, одни из которых появились недавно, а другие известны уже длительное время.

Виртуализация хранилищ предоставляет возможность определения функциональных характеристик и характеристик приложений для интересующего вас хранилища, независимо от параметров аппаратного обеспечения (например, его типа, поставщика и размещения). Представьте, например, что администратор указывает на необходимость использовать том размером 80 Гбайт для хранения резервной копии данных, но, к сожалению, максимальный объем дискового пространства составляет только 40 Гбайт. Для решения этой проблемы диспетчер томов может преобразовать два отдельных физических диска в один большой том. Учитывая рост внимания к общей стоимости владения (TCO) и увеличение запросов к объему подсистемы хранения, виртуализация хранилища предоставляет возможность эффективного управления ресурсами хранилищ. Кроме того, сокращается общая стоимость владения хранилищами.

Виртуализация хранилищ может быть реализована несколькими способами, каждый из которых предоставляет разные функции и преимущества. Один из них состоит в указании физического расположения, другой предполагает указание функционального уровня, на котором размещена виртуализация. В следующих разделах рассматриваются различные способы виртуализации хранилища.

7.3.1 Серверная (узловая) виртуализация

Виртуализация на уровне узла существует уже довольно длительное время, хотя обычно она имеет другие названия, в частности LVM (Logical Volume Manager — диспетчер логического тома) и HSM (Hierarchical Storage Management — управление иерархическим хранилищем). На уровне диспетчера тома (например, LDM в Windows 2000) виртуализация осуществляется следующим образом:

- преобразование недорогих ненадежных дисков в надежный диск с помощью программного обеспечения RAID;
- предоставление больших томов с помощью виртуализации дисков меньшего объема.

Диспетчер LDM подробно рассматривается в главе 6.

Технология HSM предоставляет метод доступа к файлам вне зависимости от их расположения — на диске или на сменном носителе, например магнитной ленте. Служба HSM, предоставляемая Microsoft, рассматривается в разделе 7.8.

Виртуализация на уровне узла представляет собой зрелую и стабильную технологию. Но существует точка зрения, что этот тип виртуализации способствует созданию "островов" хранения, на которых определенное хранилище связано с определенным сервером. Эта точка зрения оказывается еще более справедливой, если учесть незначительную популярность и недоступность кластерных файловых систем и диспетчеров томов с поддержкой кластеров, которые позволяют нескольким серверам в полной мере совместно использовать одни и те же устройства хранения, файловые системы или тома.

7.3.2 Виртуализация на уровне аппаратного обеспечения хранилища

Этот способ виртуализации также известен довольно давно. Хорошим примером служит аппаратная реализация массива RAID. Преимущество такого типа виртуализации состоит в эффективности, быстродействии и безопасности. В свою очередь, представьте устройство хранения данных, управляемое через виртуализацию на уровне узла (сервера). Злоумышленник может обойти программную виртуализацию на уровне узла или произвольно загрузить узел без программного обеспечения виртуализации, после чего получить доступ к хранилищу данных.

Катастрофический эффект такого сценария может быть предотвращен единицей хранения данных с собственным механизмом виртуализации. Еще одним преимуществом является возможность использования описываемой

технологии на нескольких серверах, вне зависимости от операционной системы. Недостаток состоит в том, что система управления виртуализацией, которая всецело зависит от производителя, является закрытой, что может привести к возникновению проблем несовместимости различных систем.

7.3.3 Виртуализация в сетях хранения

Вместо того чтобы применять виртуализацию на двух узловых системах (серверах или аппаратных подсистем хранения), ее можно реализовать в сети хранения, на базе коммутаторов, маршрутизаторов или с помощью специального аппаратного обеспечения, которое предназначено исключительно для виртуализации и управления хранилищем.

7.3.4 Внутриполосная виртуализация

При таком подходе виртуализация проводится на активном пути данных. Программное обеспечение узла осуществляет ввод-вывод данных, и устройство, которое находится между узлом и аппаратным обеспечением хранилища, обрабатывает ввод-вывод.

Преимущество *внутриполосной виртуализации* (in-band virtualization) состоит в обеспечении работы при различных методах ввода-вывода, включая доступ к хранилищу на уровнях файлов и блоков. При этом предоставляется единственная точка управления.

Недостаток заключается в превращении устройства виртуализации в единственную точку отказа и в проблемный компонент производительности системы. Эти ограничения можно обойти, развернув несколько устройств виртуализации, добавив устройству дополнительную память, используя высокопроизводительные компоненты или добавив кэширование. Тем не менее подобные методы увеличивают стоимость системы и приводят к реализации более сложной архитектуры, требующей кластеризации устройств управления.

7.3.5 Внеполосная виртуализация

При *внеполосной виртуализации* (out-of-band virtualization) пути передачи данных и управляющей информации разделены. Внеполосную виртуализацию можно сравнить с асимметричной кластерной файловой системой, в которой сервер метаданных управляет доступом к метаданным файловой системы. Сервер виртуализации обычно взаимодействует с клиентами виртуализации, которые представляют собой программное обеспечение, работающее на узле или на устройствах в сети хранения, например коммутаторах связной архитектуры или адаптерах шины.

Преимущество состоит в ускорении доступа к данным, так как данные не проходят через дополнительное устройство. Кроме того, в этой схеме присутствует единственная точка управления.

К недостаткам можно отнести проблемы, связанные с управлением несколькими устройствами внешней виртуализации. Более одного устройства внешней виртуализации требуется по следующим причинам:

- обеспечение избыточности;
- предоставление необходимой производительности;
- обеспечение соответствия требованиям топологии связной архитектуры (например, несколько "островов" SAN, разделенные физически на большое расстояние, требуют отдельных устройств виртуализации).

Виртуализация диска — программное обеспечение и прошивка, которые поставляются вместе с дисками уже несколько лет. Эта технология используется для преобразования поврежденных секторов в неповрежденные незаметно для операционной системы. Кроме того, от операционной системы скрываются такие подробности, как количество головок и цилиндров.

Виртуализация блока также доступна уже длительное время. Примером такой технологии на платформе Windows NT служит драйвер LDM (Logical Disk Manager) или FtDisk, которые предоставляют функции массивов RAID и возможность создания единых томов с объемом, превышающим объем каждого жесткого диска в отдельности.

Виртуализация файлов заключается в добавлении уровня абстракции для параметров и расположения файлов и каталогов. В качестве примера можно указать службу HSM (Hierarchical Storage Management).

Технология *виртуализации файловой системы* добавляет уровень абстракции, размещенный над несколькими файловыми системами. Пример — распределенная файловая система (DFS) в Windows NT, которая рассматривается в главе 3.

Виртуализация магнитной ленты добавляет уровень абстракции над накопителем магнитной ленты. При отсутствии виртуализации узел обычно получает выделенный накопитель, собственную библиотеку лент и носители. Виртуализация ленты обычно подразумевает эмуляцию накопителя на магнитной ленте, когда ввод-вывод кэшируется на эмулируемый накопитель, а не на жесткий диск. Кэшированные потоки ввода-вывода, как только появляется возможность, записываются на реальную магнитную ленту. Преимущество такого подхода состоит в быстром завершении операций ввода-вывода и сокращении количества необходимых физических накопителей.

7.4 Технология виртуализации хранилища от компании Microsoft

Возможности Windows Server 2003 и сведения о будущих версиях Windows свидетельствуют о тенденции к увеличению количества функций, связанных с хранилищами данных на платформе Windows NT. Виртуализация на уровне тома (FtDisk и LDM) и на уровне файловой системы (HSM и DFS) доступна в Windows NT уже довольно давно. Компания Microsoft продолжает создавать более развитую инфраструктуру управления хранилищами в рамках среды Windows NT. Кроме всего прочего, эта инфраструктура позволит администратору без проблем автоматизировать выполнение рутинных операций.

В качестве примера можно рассмотреть последовательность действий, выполняемых администратором для резервного копирования данных.

1. Создание тома (может потребоваться управление дисками или массивами RAID).
2. Обеспечение видимости тома для механизма создания моментальных снимков (может потребоваться перенастройка зонирования).
3. Создание моментального снимка.
4. Обеспечение видимости тома с моментальным снимком для сервера резервного копирования.
5. Резервное копирование.
6. Освобождение тома и его перемещение обратно в пул свободных ресурсов хранилища.

Основная цель заключается в предоставлении эффективных средств управления на каждом из этапов, перечисленных выше. Для управления может использоваться как интерфейс командной строки (автоматическое выполнение операций), так и графический интерфейс приложения управления. На каждом этапе упор делается на функциональные, а не физические аспекты. Поэтому этап создания тома должен описываться следующим образом: "создать том, размером 50 Гбайт, RAID 5 и т.д.", а не "создать E: 50 Гбайт". Для достижения этого уровня компания Microsoft предоставляет службу виртуализации диска в составе Windows Server 2003. Служба виртуализации для связной архитектуры будет представлена в следующих версиях Windows.

7.4.1 Служба виртуализации дисков

Эта служба обеспечивает управление хранилищем блоков, которое, в свою очередь, также виртуализировано. При этом конкретный тип виртуализации

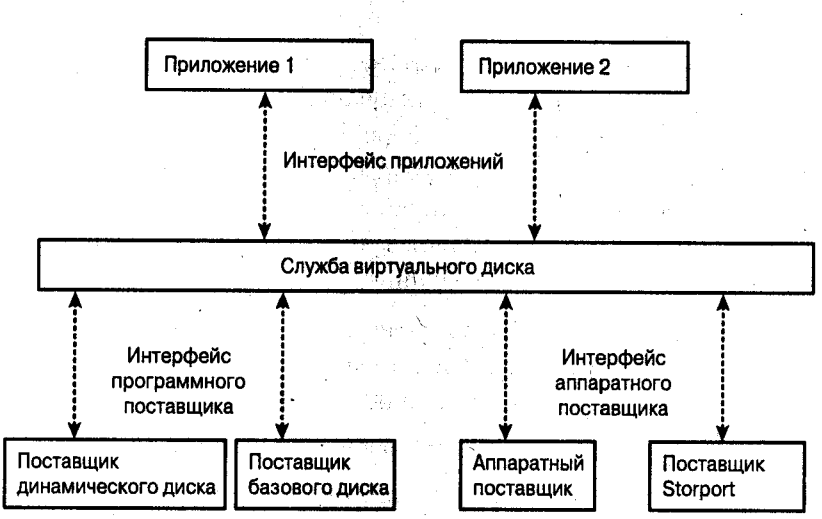


Рис. 7.2. Служба виртуального диска

(узловой, аппаратное обеспечение, массив RAID и т.д.) значения не имеет. В частности, служба виртуализации дисков предоставляет следующие возможности:

- создание номеров LUN по характеристикам;
- создание файловых систем;
- управление маршрутами.

Как показано на рис. 7.2, служба виртуального диска имеет трехуровневую модель, состоящую из приложений, службы виртуализации дисков и нескольких поставщиков, одни из которых разработаны компанией Microsoft, в то время как другие — независимыми компаниями.

Программные поставщики отвечают за управление томами и реализованы в виде серверов COM. Эти поставщики обеспечивают работу функций, указанных в наборе SDK для службы виртуальных дисков, который доступен в Microsoft при условии подписания договора о неразглашении. Функции отвечают за создание объектов COM, предоставляющих том, диск и самого поставщика, а также за создание других объектов. Программный поставщик экспортирует информацию о состоянии и работоспособности объектов, которыми он управляет, а также отправляет уведомления. Служба виртуального диска получает все уведомления; затем проводится фильтрация и отправка уведомлений приложениям, которые зарегистрировались для получения соответствующих предупреждений. Служба виртуального диска по мере необходимости (например, при расширении или сокращении размера тома) координирует свои действия с файловой системой.

Аппаратные поставщики отвечают за управление номерами LUN. Как и программные поставщики, они реализованы в виде серверов COM, поддерживающих функции, указанные в наборе SDK службы виртуального диска. Аппаратные поставщики обеспечивают работу функций, относящихся к LUN, дискам, маскировке LUN и т.д., а также отправляют уведомления.

Компания Microsoft предоставляет три поставщика, поэтому от производителей RAID-систем требуется создание собственных поставщиков.

1. Поставщик Storport, который относится к аппаратному обеспечению RAID, подключаемому непосредственно к узлу (это локальное аппаратное обеспечение, а не аппаратное обеспечение SAN).
2. Поставщик, который управляет базовыми дисками (см. главу 6).
3. Поставщик, который управляет динамическими дисками (см. главу 6).

Служба виртуального диска имеет немаловажное значение для различных приложений управления хранилищем, включая утилиты управления дисками, резервного копирования, зеркального отражения и создания моментальных снимков. Кроме того, Microsoft будет создавать новые приложения управления и утилиты, использующие функции службы виртуального диска. Обратите внимание: Microsoft размещает важные утилиты и инструменты в комплекте Resource Kit, а не в составе самой операционной системы. Примером служит, например, утилита `linkd.exe` для создания ссылок. Таким образом, читателю стоит тщательно просмотреть утилиты, входящие в набор Resource Kit для Windows Server 2003.

Служба виртуального диска должна запускаться на нескольких серверах Windows NT. Программные поставщики запускаются только на тех серверах, на которых смонтированы соответствующие тома. Приложение управления, которое использует службу виртуального диска, может работать удаленно или на том же компьютере. Служба виртуального диска будет работать в установочной среде OEM (она носит название WinPE или Windows NT Lite). В WinPE компьютер загружается с минимальной конфигурацией Windows NT и устанавливает Windows NT в другой конфигурации на этот же компьютер или выполняет последовательность тестов. Минимальная конфигурация может включать в себя службу виртуального диска и поставщика базовых дисков.

7.4.2 Служба виртуализации для связной архитектуры

Эта служба играет важную роль в автоматизации управления хранилищем. В частности, служба виртуализации связной архитектуры предоставляет программный или интерактивный доступ (с помощью командной строки

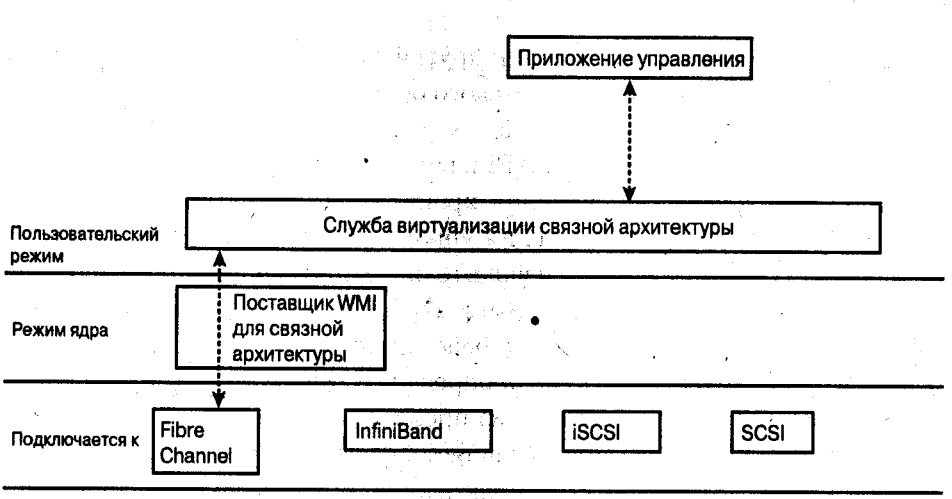


Рис. 7.3. Служба виртуализации связной архитектуры

или с помощью графического интерфейса) к управлению соединениями между хранилищами.

Обратите внимание: служба виртуализации связной архитектуры не внедрена в Windows Server 2003 по умолчанию. По сути, эта служба может вообще не предоставляться в составе операционной системы. Слишком мало дополнительной информации, разрешенной к разглашению, предоставлено по этому вопросу. Кроме того, интересно заметить, что многие производители, включая одного известного производителя коммутаторов, объявили о принятии модели CIM, разработанной рабочей группой DTMF и ассоциацией SNIA. Можно сделать логический вывод, что API управления коммутаторами, предоставленные производителем, будут не так важны, как методы и объекты, предоставленные с помощью CIM. Что из этого получится и как будет развиваться служба виртуализации связной архитектуры, может сказать только время (и, возможно, одна из следующих редакций этой книги).

На рис. 7.3 показана приблизительная попытка определения службы виртуализации связной архитектуры. Служба виртуализации предоставляет API для приложений управления и получает информацию о связной архитектуре от поставщика WMI.

7.5 Программные интерфейсы приложений для адаптеров шины

В сетях хранения данных адаптеры шины обеспечивают физическое подключение между сервером и другими элементами сети хранения, включая

устройства хранения, коммутаторы и другие узлы. С ростом сложности сетей хранения необходимость автоматического обнаружения и управления устройствами становится все более важной.

Ассоциация SNIA определила API в виде библиотеки на языке C, который позволяет приложениям управления хранилищами контролировать адаптеры шины Fibre Channel. Приложения управления могут реализовать единый интерфейс, не зависящий от производителя адаптера шины или типа последнего. Достаточно, чтобы производитель адаптера шины поддерживал утвержденный SNIA программный интерфейс приложений для адаптеров шины. В этот интерфейс входит поддержка опроса и установки параметров управления адаптером шины, а также анализ производительности адаптера. В частности, API адаптера шины предоставляет перечисленные ниже элементы.

- Атрибуты адаптера шины, например модель, название производителя и имя WWN (уникальное 64-разрядное имя, назначаемое производителем и зарегистрированное в IEEE, который назначает диапазон имен определенному производителю).
- Атрибуты порта, например идентификатор порта (24-разрядное число, которое уникально для каждого узла), тип порта, текущий статус порта и максимальный размер кадра, который поддерживается портом.
- Атрибуты протокола порта Fibre Channel, например номер шины SCSI или целевой идентификатор SCSI.
- Статистика порта, например количество принятых и отправленных кадров, время с момента последнего сброса данных статистики и возникшие ошибки (причина и их количество).
- Управляющая информация FC-3 (третий функциональный уровень Fibre Channel), например имя WWN или идентификатор порта; следует отметить, что API позволяет приложению управления опрашивать такие службы Fibre Channel, как сервер имен. Кроме того, API поддерживает назначение информации об узле, например имени WWN и идентификатора.

Хотя компания Microsoft и члены ассоциации SNIA поддерживают программный интерфейс приложений адаптера шины, эта поддержка реализуется по-разному. Архитектура, принятая ассоциацией SNIA, включает в себя три компонента (рис. 7.4).

1. Универсальная динамически подключаемая библиотека API адаптера шины, которая поддерживается ассоциацией SNIA. Эта библиотека предоставляет стандартный интерфейс для приложений управления; на нижнем уровне библиотека связывается с динамически подключаемыми библиотеками, предоставленными производителями.

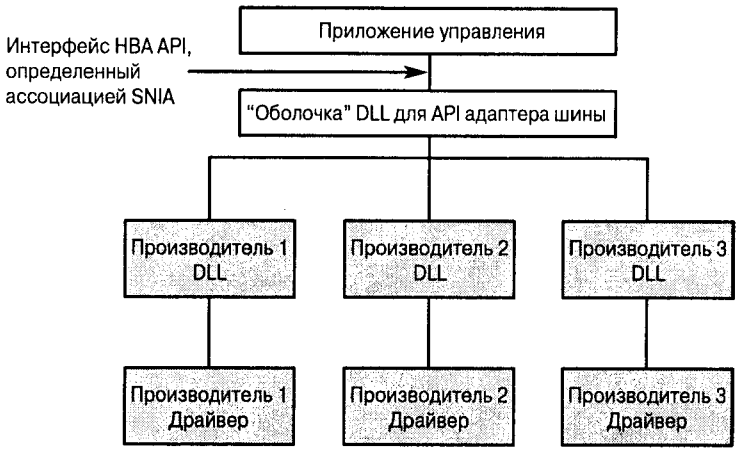


Рис. 7.4. Программный интерфейс приложений для адаптера шины от ассоциации SNIA

2. Динамически подключаемая библиотека, созданная производителем адаптера шины, которая подключается к универсальной библиотеке. Библиотека производителя предоставляет управляющую информацию и взаимодействует с драйвером от того же производителя с помощью закрытых вызовов управления вводом-выводом.
3. Драйвер адаптера шины, предоставленный производителем.

Хотя продвижение в сторону стандартизации имеет свои преимущества, Microsoft указывает на возможные проблемы при использовании такого подхода.

- Отсутствует эффективный способ управления распространением и учетом версий динамически подключаемых библиотек. Это еще один пример ситуации, когда несколько приложений устанавливают собственные библиотеки, потенциально перезаписывая библиотеки, установленные другими приложениями.
- Производитель адаптера шины должен создать не только драйвер устройства и закрытый интерфейс управления вводом-выводом для этого драйвера, но и собственную динамически подключаемую библиотеку, а также внести определенные изменения в библиотеку-оболочку, чтобы предоставить необходимые интерфейсы собственной динамически подключаемой библиотеке.
- Тестирование и сертификация драйверов от производителя, использующих закрытые вызовы управления вводом-выводом, является исключительно сложным процессом; например, как проверить, что некоррект-

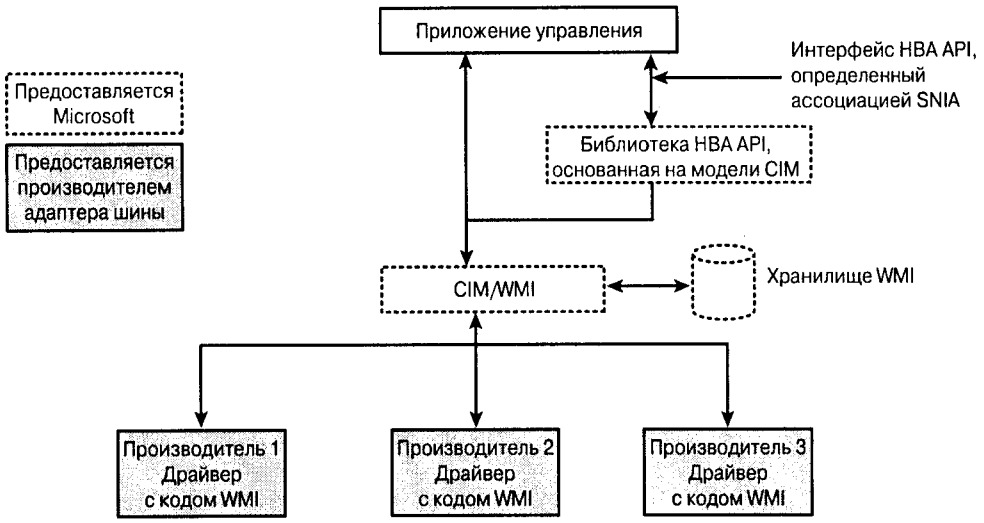


Рис. 7.5. Программный интерфейс приложений для адаптера шины от Microsoft

ные параметры вызова управления вводом-выводом не приведут к переполнению буфера?

- Архитектура на первый взгляд выглядит расширяемой, но при более детальном рассмотрении можно понять, что производители адаптеров шины всегда будут "догонять" разработчиков приложений управления, добавляя код, который работает с расширениями, характерными для определенного производителя.
- Не поддерживаются связи и управление в режиме ядра. Для архитектуры SNIA требуется, чтобы приложение управления ожидало полноценной работоспособности подсистемы пользовательского режима Windows NT. Но для некоторых аспектов управления, например для маскировки LUN, необходимо наличие драйвера в режиме ядра, который будет выполнять определенные действия в процессе загрузки операционной системы еще до того, как будет запущена подсистема пользовательского режима.

Компания Microsoft предлагает несколько иной метод (рис. 7.5) и подразумевает применение описанных ниже компонентов.

- Универсальная динамически подключаемая библиотека API для адаптера шины, которая поддерживается компанией Microsoft. Эта библиотека предоставляет интерфейс, определенный ассоциацией SNIA, для приложений управления более высокого уровня. На более низком уровне библиотека подключается к интерфейсу WMI, представляющему собой

реализацию модели CIM от Microsoft. (Как уже отмечалось, CIM — это объектно-ориентированная модель управления, принятая ассоциацией SNIA и рабочей группой DMTF.) Библиотека также будет проводить преобразование данных между API адаптера шины, отвечающего спецификации SNIA и интерфейсом WMI.

- Драйвер устройства для адаптера шины, созданный производителем; драйвер использует WMI и делает интерфейс управления/настройки доступным в репозитории WMI. Поскольку WMI представляет собой двунаправленный интерфейс, драйвер реализует функции пакетов IRP для интерфейса WMI, которые позволяют приложению управления устанавливать конфигурационные параметры драйвера. Обратите внимание, что от производителя все равно требуется создание драйвера, в который будет добавлен код поддержки WMI.

Метод компании Microsoft имеет ряд преимуществ.

- Все представленные интерфейсы стандартизированы, тогда как в архитектуре ассоциации SNIA интерфейс между универсальной библиотекой API адаптера шины и библиотекой производителя закрыт и определяется производителем. Подход компании Microsoft соответствует принятой модели CIM.
- Поддержка модификаций, так как производитель может расширить класс WMI или определить новый класс и внедрить управляющую информацию в пределах этого класса. В данном случае также просматриваются общие преимущества модели CIM.
- Самым существенным преимуществом является возможность использования модели CIM или API адаптера шины от SNIA. Программа, использующая API адаптера шины от SNIA, может работать без изменений, поскольку код WMI в драйвере и динамически подключаемая библиотека от Microsoft выполняют преобразование данных WMI в данные API адаптера шины от SNIA.
- Такая архитектура позволяет компоненту режима ядра опрашивать драйвер производителя и выполнять соответствующие действия.
- Приложения управления не интересуются (и не должны интересоваться), каким образом код нижнего уровня получает необходимую информацию. Пока приложение управления использует API адаптера шины от SNIA (как рекомендуется Microsoft), код на нижнем уровне может получать необходимую информацию как от динамически подключаемой библиотеки SNIA, так и от библиотеки WMI. В любом случае приложение управления осуществляет запись данных на базе единого интерфейса.

- Безопасные изменения конфигурации SAN, проводимые с помощью эффективных технологий безопасности; например, все изменения конфигурации требуют цифрового сертификата, идентификации пользователя и пароля.
- Ограничение методов разрешения изменений; например, только определенные коммутаторы могут вносить изменения в связную архитектуру, что приводит к отказу от принятия изменений со всех остальных коммутаторов.
- Зонирование (см. главу 4) — это еще один способ обеспечения безопасности в сетях хранения данных.

7.8 Управление иерархическим хранилищем

Службу HSM (Hierarchical Storage Management) иногда путают с продуктами для основного и вспомогательного резервного копирования, которые помогают отслеживать размещение файла на архивной магнитной ленте или даже состояние файла в архиве. Хотя HSM и резервное копирование переносят файлы и данные между основным носителем (обычно это жесткие диски) и вторичным носителем (магнитная лента или оптический носитель), эти технологии существенно различаются. При использовании HSM файл выглядит неизменным. В некоторых случаях файл может иметь специальные атрибуты, которые указывают на увеличение времени доступа. В свою очередь, программы резервного копирования не воспринимают ранее скопированный и удаленный файл.

Службу HSM также уместно сравнить с виртуальной памятью. *Виртуальная память* — это физически не существующая область памяти, т.е. память, воспринимаемая компьютером, однако не установленная на самом деле. Операционные системы создают виртуальную память, выгружая содержимое неиспользуемых областей памяти (дорогостоящего и быстродействующего интерактивного ресурса) на жесткий диск (относительно недорогой и медленный носитель). После этого освободившаяся память выделяется другому процессу. При попытке другого процесса получить доступ к памяти, выгруженной на диск, процесс временно приостанавливает работу, ожидая, пока содержимое памяти не будет восстановлено с диска. После восстановления области памяти процесс может продолжать нормальную работу. Служба HSM выполняет аналогичную функцию, однако в этом случае дорогим и быстрым ресурсом считается дисковое пространство, а недорогим — ленточный или другой носитель, например оптические диски.

Некоторые продукты HSM интегрируют в себе службы резервного копирования. Чтобы достичь баланса в функциональности HSM, а также избе-

жать нарушений в нормальной работе центров хранения данных, продукты HSM устанавливают определенные граничные параметры. Как только размер свободного дискового пространства достигает определенного граничного параметра, продукт HSM начинает работу даже в рабочее время сервера, перемещая некоторые файлы на неинтерактивный носитель и тем самым освобождая дисковое пространство.

7.8.1 Модуль RSS

Операционная система Windows 2000 Server содержит модуль RSS (Remote Storage Services), который предоставляет функции управления иерархическим хранилищем (HSM). Служба RSS отслеживает свободное пространство и статистику обращения к файлам на *управляемых томах*. Администратор хранилища устанавливает критерии минимального объема свободного дискового пространства, а также срок, в течение которого к файлу не должны обращаться, чтобы файл попал на неинтерактивный носитель, и т.д.

Служба RSS имеет специальный механизм, который периодически проверяет объем свободного дискового пространства, сканирует файлы для предварительного перемещения и, как только дисковое пространство сокращается ниже определенного минимума, отбрасывает предварительно перемещенные файлы (полное описание процесса предварительного перемещения приводится несколько ниже). Неиспользованные файлы перемещаются на неинтерактивный носитель, например на магнитную ленту, в соответствии с набором критериев, которые рассматриваются далее. Для этого файла устанавливается точка повторной обработки, которая содержит информацию, необходимую для перемещения файла.

Файлы для перемещения выбираются на основе политик, установленных администратором. Эти политики описывают:

- объем свободного дискового пространства, который должен быть доступен;
- последнюю дату и время доступа к файлу;
- размер файла;
- определенные администратором правила включения и исключения файлов и каталогов.

Модуль RSS интегрируется в Windows 2000 следующим образом:

- графический интерфейс программы Проводник Windows отмечает перемещенные файлы специальной пиктограммой;
- в окне командной строки размер перемещенных файлов указывается в скобках;

- приложение резервного копирования Windows NT координирует свои действия с модулем RSS; при резервном копировании файл открывается с параметром `FILE_OPEN_NO_RECALL` с помощью функции `CreateFile`, что позволяет обеспечить резервное копирование файла без перемещения файла обратно на диск;
- задания модуля RSS передаются планировщику Windows NT, и расписанием заданий модуля RSS можно управлять, как расписанием остальных заданий;
- служба индексации Windows 2000 распознает отсутствие изменений в файле, даже если файл был перемещен с диска в удаленное хранилище;
- при доступе к файлу, расположенному в удаленном хранилище, сетевые задержки автоматически увеличиваются, что позволяет выполнять перемещение файла.

Модуль RSS работает с файлами, находящимися в одном из трех состояний.

1. *Обычный файл*, который находится на жестком диске.
2. *Предварительно перемещенный файл*, неименованный поток данных которого скопирован на неинтерактивный носитель, но еще не удален с диска. Предварительно перемещенный файл содержит точку повторной обработки
3. *Перемещенный файл*, неименованный поток данных которого перемещен на неинтерактивный носитель и удален с диска; файл отмечен как разреженный и имеет установленный атрибут `FILE_ATTRIBUTE_OFFLINE`. Точка повторной обработки файла устанавливается со специальным тегом, который называется `IO_REPARSE_TAG_HSM`.

На рис. 7.6 представлена архитектура модуля RSS. Этот модуль состоит из нескольких компонентов пользовательского режима и режима ядра, которые рассматриваются далее.

При получении доступа к перемещенному файлу драйвер фильтрации RSS перехватывает пакеты IRP и размещает их в очереди. Драйвер фильтрации RSS взаимодействует с агентом файловой системы RSS. В свою очередь, агент вызывает службы ядра RSS. Ядро получает данные и передает их драйверу фильтрации RSS, который восстанавливает данные файла фрагмент за фрагментом по мере их получения. Если для получаемых данных в очереди находится операция ввода-вывода, эта операция завершается. Полученные данные используются для завершения всех ожидающих в очереди операций ввода-вывода. Состояние файла (включая данные точки повторной

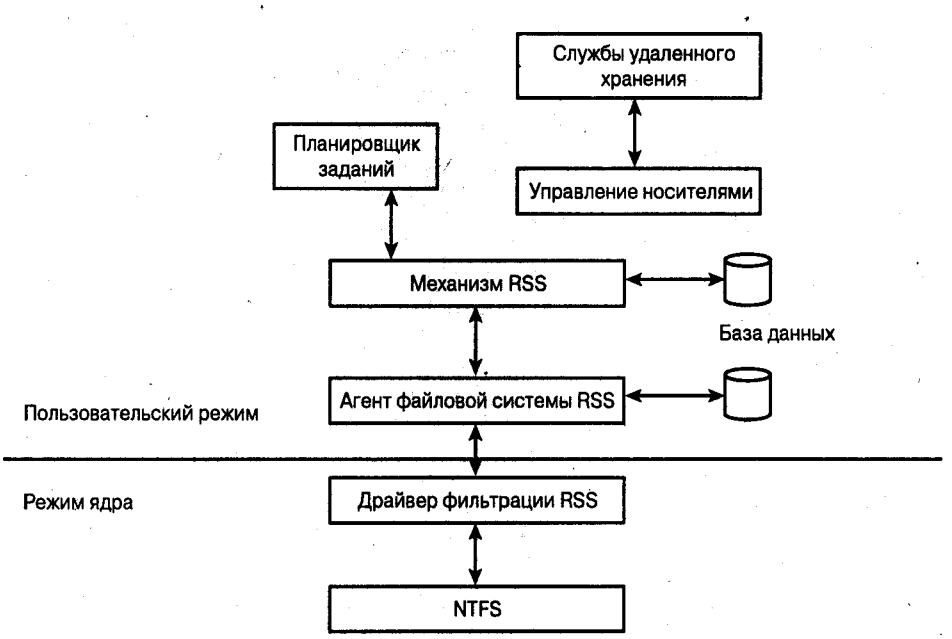


Рис. 7.6. Архитектура RSS

обработки) обновляются, указывая, что файл больше не перемещен, а стал предварительно перемещенным.

При доступе к перемещенному файлу с помощью функции `CreateFile` с параметром `FILE_OPEN_NO_RECALL` пакеты IRP обрабатываются подобным образом, но файл не перемещается с магнитной ленты на диск. Данные восстанавливаются с ленты и передаются приложению без записи на диск. Обратите внимание: Windows 2000 поддерживает одновременное перемещение только одного файла с удаленного хранилища на магнитную ленту[1]. Это справедливо даже при наличии нескольких накопителей удаленного хранилища, когда допуск к обрабатываемым файлам осуществляется по двум путям, а сами файлы расположены на двух различных носителях в двух разных накопителях.

Ядро RSS работает в качестве клиента RSM (Removable Storage Management), использующего RSM для управления носителем. Для хранения подробной информации о носителе ядро RSS использует базу данных.

Агент файловой системы RSS отвечает за периодическое сканирование управляемых томов NTFS и подготовку списка файлов, которые должны

[1] Подобное ограничение существует и в Windows Server 2003. Учитывая, что код технологии HSM лицензирован Microsoft у компании, ныне входящей в корпорацию Sun Microsystems, в следующей после Windows Server 2003 операционной системе ожидается наличие многих интересных функций.

быть перемещены. Этот список составляется на основе критериев, определенных администратором. Агент файловой системы сообщает список файлов ядру RSS. Как только ядро завершает предварительное перемещение файлов, агент файловой системы RSS добавляет файл в свою базу данных, содержащую список предварительно перемещенных файлов.

Как только свободное дисковое пространство становится меньше определенного граничного значения, установленного администратором, запускается автоматическое задание, запланированное агентом файловой системы RSS. Агент удаляет потоки данных файлов, которые были предварительно перемещены (переводит файл из предварительно перемещенного состояния в полностью перемещенное состояние). Перед окончательным перемещением агент файловой системы проверяет, не изменились ли файлы с момента предварительного перемещения. Эта проверка выполняется с помощью механизма журнала USN (см. главу 6).

Служба RSS не устанавливается по умолчанию, а предоставляется вместе с утилитой управления с графическим интерфейсом, которая поддерживает запуск на удаленном компьютере под управлением Windows. Графический интерфейс утилиты имеет несколько компонентов.

- *Компонент программы* Проводник *Windows.* Позволяет пользователям просматривать информацию о дате и времени предварительного перемещения файла или о расположении данных файла в удаленном хранилище; пользователи могут принудительно включить предварительное перемещение выбранных файлов (если обладают соответствующими разрешениями на доступ к этим файлам).
- *Компонент управления дисками.* Отображает информацию о томе, например объем свободного и занятого дискового пространства, объем дискового пространства, занятого под предварительно перемещенные файлы, объем дискового пространства, занятый под указатели на файлы, и т.д.
- *Пользовательский интерфейс.* Позволяет администратору отменять восстановление файла (перемещение с удаленного хранилища на диск).
- *Интерфейс управления.* Позволяет устанавливать граничные значения, критерии выбора файлов для перемещения и т.д.

Модуль RSS имеет несколько ограничений; некоторые из них описаны далее.

- В отличие от Windows Server 2003, модуль RSS, предоставляемый в Windows 2000, не поддерживает кластеризацию.
- Операционная система Windows 2000 в качестве вторичного носителя поддерживает ленту шириной 4 мм, 8 мм и формата DLT.

- Модуль RSS предоставляет возможность перемещения только неименованных потоков данных. Именованные потоки данных не обрабатываются вообще.
- Модуль RSS использует точки повторной обработки, которые появились в файловой системе NTFS Windows 2000. Таким образом, модуль RSS не поддерживает более ранние версии NTFS.
- Модуль RSS может работать только с фиксированными томами и не поддерживает тома на сменных носителях, например дисках DVD или Jazz.
- Модуль RSS должен устанавливаться только после сжатия управляемого тома, если необходима работа со сжатыми томами.
- Модуль RSS должен устанавливаться только после установки службы индексации, если в ней есть необходимость.
- Модуль RSS хранит базу данных на системном томе. Это означает, что тома, управляемые модулем RSS, не являются самодостаточными и не могут быть перемещены с одного сервера на другой.
- Модуль RSS не может использоваться для перемещения скрытых, системных, шифрованных или разреженных файлов, а также любых файлов с расширенными атрибутами.

7.8.2 Подсистема RSM в Windows 2000

Подсистема RSM (Removable Storage Management) в Windows 2000 обеспечивает работу важных функций, включая:

- поддержку накопителей на магнитной ленте и ленточных автоматов;
- управление сменными носителями, например лентами и компакт-дисками;
- возможность совместного использования накопителей на магнитной ленте и ленточных автоматов в различных приложениях, например приложениях резервного копирования и HSM.

Операционная система Windows 2000 предоставляет набор компонентов, которые позволяют управлять хранилищами и разрабатывать приложения, использующие сменные носители. Компоненты включают:

- модули администрирования сменного хранилища;
- диспетчер сменного хранилища (API);
- базы данных сменных хранилищ.

Служба RSS представляет собой инструмент для решения широкого диапазона задач, включая резервное копирование. Таким образом, RSS не заменяет резервное копирование, а выступает в качестве метода управления операциями резервного копирования и восстановления данных.

Чтобы понять принципы работы этих компонентов, необходимо разобраться в архитектуре RSM.

7.8.2.1 Архитектура RSM в Windows 2000

На рис. 7.7 представлена общая архитектура подсистемы RSM в Windows 2000. Служба RSM играет важную роль во всей подсистеме RSM. Служба работает в качестве хранилища для кода реализации API RSM. Она получает запросы от приложений и размещает их в очереди, обрабатывая при получении доступа к соответствующим ресурсам. При запуске службы осуществляется поиск и инициализация различных библиотек, определение изолированных накопителей и ассоциирование накопителей с системами замены носителей.

Производители, разрабатывающие аппаратное обеспечение для RSS, должны создавать соответствующий мини-драйвер. Драйвер класса реализует многие функции, общие для устройств, а также отвечает за создание объ-

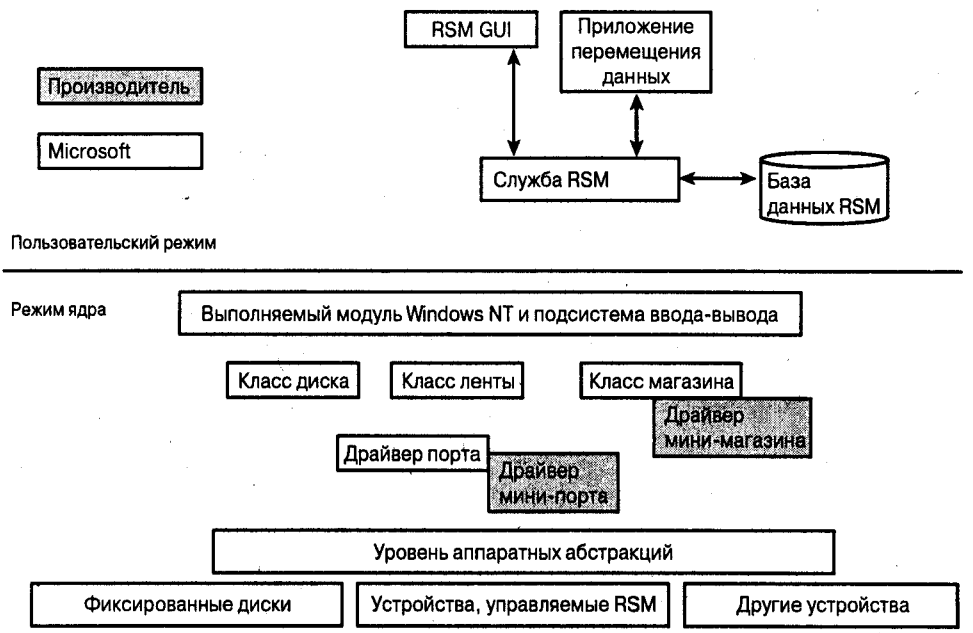


Рис. 7.7. Архитектура RSM

ектов устройства, предоставляющих его для других подсистем. Однако от мини-драйвера ожидается обработка структур данных драйверов Windows NT, включая пакеты IRP. Несмотря на это, возможности, которые должен предоставить мини-драйвер, ограничены по сравнению с возможностями обычных драйверов Windows NT.

Обратите внимание: RSM принимает участие в управлении и настройке устройства, которое содержит сменный носитель, а также в управлении и настройке самого носителя. После установки прав владения устройством, монтирования и позиционирования носителя RSM исключается из пути передачи данных, т.е. дополнительные операции ввода-вывода не проводятся.

7.8.2.2 Программные интерфейсы приложений RSM в Windows 2000

В наборе Windows 2000 Platform SDK описано создание приложения с помощью API RSM и предоставлена дополнительная информация об этом интерфейсе. Преимущество таких API состоит в эффективности создания приложений управления хранилищами.

На рис. 7.8 демонстрируется ситуация, которая существовала до появления интерфейсов RSM. Каждое приложение обладало сложной структурой, так как должно было работать со множеством устройств и поддерживать разные их типы. Более того, каждый раз при появлении нового устройства или типа устройства приложения требовали модификации.

На рис. 7.9 приведена схема реализации, предоставленная после появления RSM API. Очевидно, что, кроме всего прочего, RSM обеспечивает расширяемость. Как только RSM добавляет поддержку для нового устройства, приложение может использовать это устройство практически без изменений. Список устройств, поддерживаемых RSM, постоянно меняется; последняя

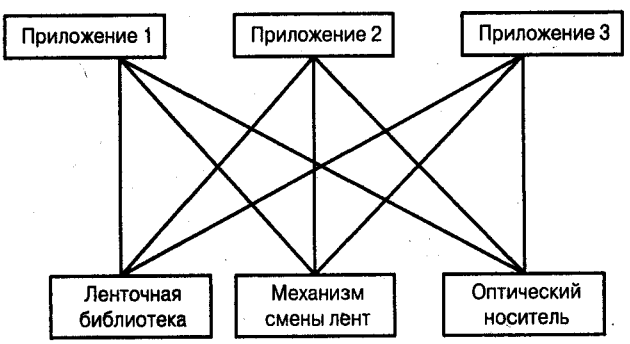


Рис. 7.8. Схема разработки приложений до появления RSM API

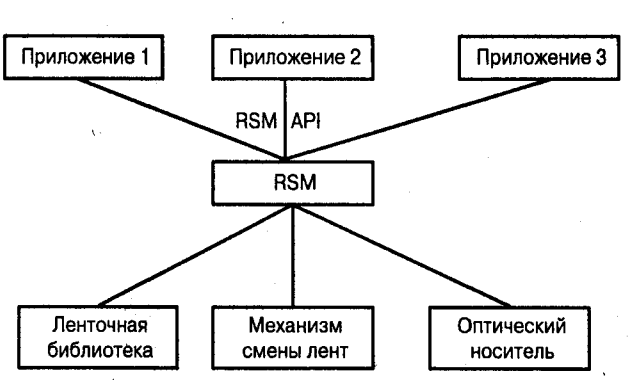


Рис. 7.9. Разработка приложений, упрощенная благодаря использованию RSM API

версия списка доступна на Web-узле списка совместимого аппаратного обеспечения по адресу: `http://www.microsoft.com/hwdq/hcl`.

Программные интерфейсы приложений RSM можно разделить на несколько категорий в зависимости от предоставляемых функций. Вот некоторые из них:

- интерфейсы очистки содержимого накопителя, например резервная очистка, принудительная, запланированная и интерактивная;
- интерфейс для определения изменений состояния изолированных дисков;
- интерфейс базы данных для резервного копирования и восстановления базы данных RSM, а также для регистрации и прекращения регистрации уведомлений базы данных;
- функции управления библиотекой для вставки, перемещения и извлечения носителя (в рамках библиотеки), а также для включения или отключения ресурсов накопителя или автоматической системы смены носителей;
- монтирование, размонтирование и управление пулами носителей;
- другие интерфейсы, необходимые для создания эффективного приложения управления хранилищем, например интерфейсы для опроса состояния, отмены выполняющихся операций или для работы с объектами RSM.

7.8.2.3 База данных RSM

База данных RSM хранит информацию, необходимую для работы подсистемы RSM. Примерами такой информации могут служить:

- список носителей;
- подробная информация о пуле носителей, включая конфигурацию пула и его содержимое;
- конфигурация библиотеки.

База данных *не* содержит информацию каталога, описывающую расположение файлов на носителях. Эти сведения должна предоставлять программа хранения данных.

Пользователи могут создавать резервную копию базы данных, вручную копируя файлы. Обычно эти файлы расположены в папке `%SystemRoot%\System32\ntmsdata`. Очевидно, что служба RSM должна быть остановлена перед копированием файлов.

Кроме того, базу данных можно скопировать средствами интерфейса RSM. В этом случае служба должна быть запущена, но не рекомендуется активно использовать службу другими приложениями.

7.8.2.4 Пулы носителей

Пулы обеспечивают эффективное администрирование носителей. Пул представляет собой обычный набор носителей, например магнитных лент или компакт-дисков, обладающих общим свойством. Существует два типа пулов носителей: пулы приложений и системные пулы.

Системные пулы используются операционной системой для собственных нужд. Три типа системных пулов носителей описаны ниже.

1. *Свободный пул* содержит носители, которые может использовать и возвращать в свободный пул по окончании работы с ними любое приложение.
2. *Пул импорта* содержит распознанные, однако не использовавшиеся ранее носители. Хорошим примером служит носитель, записанный программой резервного копирования на другом компьютере и впервые смонтированный на данном компьютере.
3. *Нераспознанный пул* содержит носители, распознанные как содержащие данные, однако тип данных RSS не может распознать. Примером служит носитель, созданный в другой операционной системе или записанный приложением для резервного копирования с неизвестной сигнатурой. Этот пул предназначен для уведомления администратора, который может предотвратить перезапись носителя.

Пулы приложений создаются с помощью интерфейса RSM приложениями управления хранилищами. В качестве примера можно указать такие приложения, как Windows Backup и Windows Remote Storage (HSM).

7.8.2.5 Пользовательский интерфейс администратора RSM

Этот интерфейс реализован в Windows 2000 и представляет собой оснастку MMC, которая дает возможность администратору системы или хранилища просматривать и настраивать объекты. Для представления пулов носителей, рабочих очередей и физических свойств устройств хранения используются различные объекты.

7.9 Будущее управления хранилищами по версии ассоциации SNIA: стандарты SMI

Недавно ассоциация SNIA разработала спецификацию стандартов, которая называется SMI (Storage Management Initiative) и в момент рассмотрения имела кодовое наименование Bluefin. В этой спецификации описано управление хранилищами логических и физических ресурсов в сетях хранения данных с помощью API, не относящихся к конкретным производителям. К методам управления хранилищем, описанным в SMI, относятся обнаружение устройств и управление операциями, в частности управление конфигурациями, уведомлениями, производительностью и безопасностью. Спецификация SMI впервые продемонстрирована весной 2002 года и получила поддержку нескольких производителей.

Спецификация SMI включает в себя существующие промышленные стандарты.

- Проект SMI основан на модели CIM, предложенной ассоциацией SNIA. В спецификации SMI протокол HTTP указан в качестве транспортного механизма, а содержимое сообщений будет передаваться по этому протоколу на языке XML. Для обеспечения защиты и целостности данных в SMI описано применение технологии TLS (Transport Layer Security), рассмотренной в стандарте RFC 2246. Технология TLS — это протокол обеспечения защиты и целостности данных при передаче между двумя приложениями. В его основе лежит широко известный протокол SSL (Secure Sockets Layer).
- Для обеспечения безопасности спецификация SMI требует использования стандарта аутентификации HTTP DAA (Digest Access Authentication). Этот метод аутентификации определен в стандарте RFC 2617 и представляет собой расширение протокола HTTP, позволяющее клиенту отправлять идентификационную информацию и пароль в зашифрованном виде, а не открытым текстом.
- В спецификации SMI предполагается использование протокола SLP (Service Location Protocol) для обнаружения ресурсов хранилищ. Прото-

кол описан в стандарте RFC 3224 и предоставляет методы обнаружения сетевых служб с поддержкой расширений сторонних производителей.

- В SMI содержится описание управления блокировками, которое позволяет нескольким приложениям, возможно от разных производителей, использовать службы управления блокировками для совместного использования ресурсов хранилища.

7.10 Сложности практической реализации

Интересно наблюдать, как поддержка интерфейса WMI развивается от рекомендуемой до обязательной в наборах программной разработки Microsoft и в требованиях этой компании. Развитие WMI происходит согласно устоявшейся практике Microsoft, которая меняет рекомендуемый статус технологии на обязательный параллельно с развитием самой технологии[2].

Кроме того, стоит проследить, как производители будут разрабатывать поддержку стандартов управления хранилищами и насколько успешно будут взаимодействовать друг с другом продукты, следующие этим стандартам.

Службы теневого копирования томов и виртуального диска предоставляют производителям возможность создания приложений для управления хранилищами. Такие приложения без проблем интегрируются с остальными компонентами платформы Windows NT.

7.11 Резюме

Методы управления хранилищами приобретают все большее значение, поскольку растет объем хранимых данных. Несколько производителей предоставили собственные технологии, однако большинство из них являются закрытыми. Ассоциация SNIA пытается стандартизировать управление хранилищем с помощью общей информационной модели (CIM). Компания Microsoft также уловила тенденцию и предоставляет возможности управления хранилищем через командную строку и графический интерфейс. Интерфейс для командной строки позволяет автоматизировать управление хранилищем с помощью командных сценариев.

Компания Microsoft в рамках платформы Windows NT создает инфраструктуру для управления хранилищем. Служба виртуального диска предоставляет стандартизированный способ обнаружения ресурсов хранилищ и управления ими с помощью диспетчерских программ. Служба виртуального

[2]Во всяком случае, автор этой книги будет следить за развитием WMI с неусыпным вниманием, поскольку в свое время работал на должности менеджера проекта WMI в компании Microsoft.

диска также позволяет производителям аппаратного обеспечения хранилищ предоставлять информацию о своих устройствах подобным диспетчерским приложениям. Служба виртуализации для связной архитектуры (информация об этой службе в данный момент весьма ограниченна) позволяет стандартизированным способом управлять связной архитектурой.

Кроме того, в Windows 2000 предоставляется комплексная инфраструктура HSM (Hierarchical Storage Management), которая полностью интегрирована с остальными элементами операционной системы и включает в себя такие приложения, как графический интерфейс программы Проводник Windows, окно интерпретатора командной строки, приложения резервного копирования и планировщик заданий.

Глава 8

Технологии IP Storage и InfiniBand

В этой главе рассматриваются две развивающиеся технологии: IP Storage и InfiniBand.

Набор технологий IP Storage позволяет предоставлять доступ к корпоративным хранилищам данных по протоколу IP. Под термином *IP Storage* подразумевается набор технологий iSCSI, FCIP и iFCP (и это далеко не исчерпывающий список), каждая из которых описывается в настоящей главе.

Технология InfiniBand — это не сколько протокол для хранилищ данных, как высокоскоростная, обладающая низкими задержками универсальная сетевая связная архитектура. Некоторые рабочие группы рассматривают вопрос использования InfiniBand для подключения хранилищ, но основная сфера применения относится к серверам кластеров. С появлением стандарта 3GIO перспективы использования технологии InfiniBand в качестве замены шины PCI были частично утрачены.

В данной главе приводится общее описание IP Storage и InfiniBand, которое будет полезно читателям, не знакомым с этими технологиями. Цель описания — предоставить информацию, необходимую для понимания особенностей систем Microsoft, основанных на IP Storage и InfiniBand. Читатели, желающие более подробно изучить эти технологии, могут обратиться к источникам, приведенным в конце книги. Поскольку компания Microsoft еще не выпустила упомянутое программное обеспечение, его описание можно рассматривать как попытку предугадать будущие тенденции. На основе материалов этой главы нельзя строить коммерческие планы или планы по развертыванию тех или иных технологий.

Технологии IP Storage и InfiniBand на данный момент находятся в стадии становления, хотя и разрабатываются уже достаточно длительное время.

8.1 Технология IP Storage

Термин IP Storage охватывает набор технологий, обеспечивающих взаимодействие на уровне блоков между устройствами хранения и серверами по

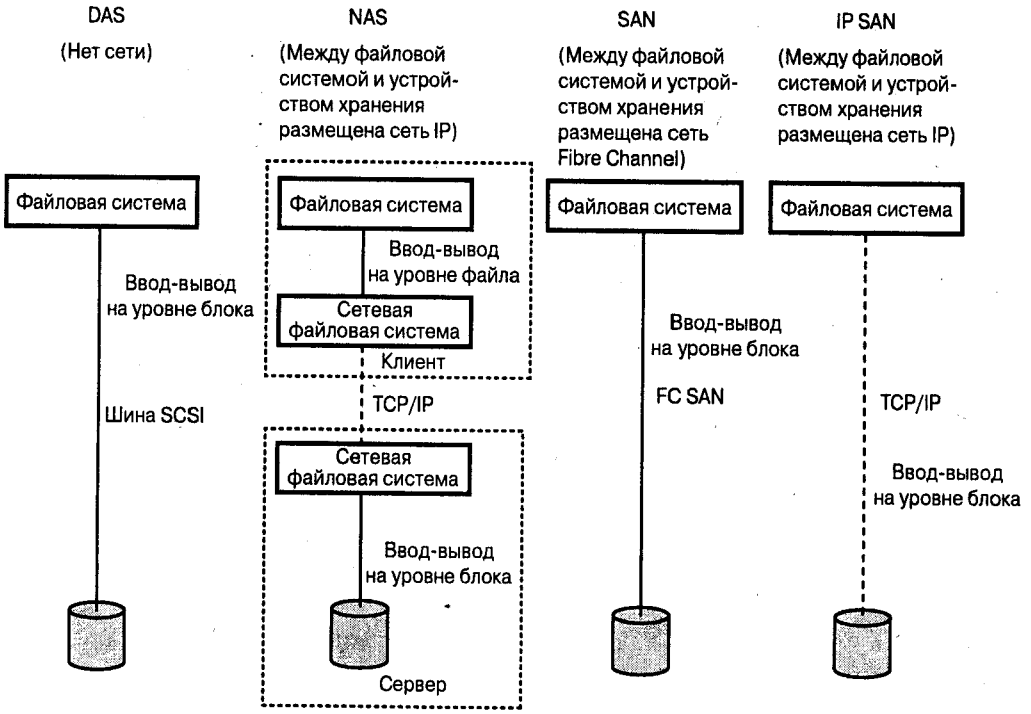


Рис. 8.1. Сравнение технологий DAS, NAS, SAN и IP SAN

протоколу IP. Внимательный читатель может заметить, что доступ к данным по сетям на основе протокола IP реализован уже довольно давно, например приложения могут получить доступ к серверу с помощью сетевой файловой системы CIFS или NFS. Разница в данном случае заключается в том, что приложение ориентировано на уровень файлов и преобразование между вводом-выводом на уровне файлов и на уровне блоков выполняется устройством NAS или сервером после получения запроса по сети. При использовании SAN на основе протокола IP запрос и ответ, передающиеся по сети, содержат данные ввода-вывода на уровне блоков, а не на уровне файлов.

На рис. 8.1 показаны основные отличия хранилища, подключенного к серверу (direct-attached storage — DAS), сетевого хранилища (network-attached storage — NAS), сети хранения данных (storage area networks — SAN) и сети хранения данных на основе протокола IP. Ниже перечислены особенности, на которые следует обратить внимание.

- При использовании хранилища, подключенного к серверу, для передачи запросов и ответов сеть не применяется.

- При использовании сетевого хранилища сеть на основе протокола IP размещена между файловой системой и устройством хранения. Конечно, устройство NAS можно подключать не только к сети на основе протокола IP, однако этот протокол наиболее популярен. По сети передаются данные ввода-вывода на уровне файлов. Из этого правила всегда существуют исключения, и примером может служить устройство EMC Celerra HighRoad, которое подключается одновременно к локальной сети и к сети хранения данных. Такой вариант подключения подробно рассматривается в главе 6 (см. раздел 6.6).
- При использовании SAN сеть размещается между файловой системой и устройством хранения; однако по сети передаются данные ввода-вывода на уровне блоков, а не на уровне файлов. В классических SAN практически всегда используется интерфейс Fibre Channel.
- При использовании IP Storage сеть также размещается между файловой системой и устройством хранения, но в данном случае она основана на протоколе IP.

8.1.1 Почему IP Storage?

Идея технологии IP Storage возникла благодаря осознанию того, что иметь в наличии два типа сетей вовсе не обязательно. Этими двумя сетями являются IP и Ethernet, которые используются для соединения клиентов и серверов (так называемые *интерфейсные сети* (front-end)). В свою очередь, сеть между сервером и хранилищем данных называется *серверной* (back-end).

Ниже описаны причины, лежащие в основе развития IP Storage.

- Протокол IP — это развитая, хорошо себя зарекомендовавшая технология, которая нашла применение в сетях Ethernet, ATM и т.д.
- Маршрутизация протокола IP также весьма развита и предоставляет несколько маршрутов между серверами и устройствами хранения в условиях постоянно модифицируемой сети.
- В некотором смысле проблема управления хранилищем аналогична известной проблеме управления сетями IP.
- Протокол IP обеспечивает взаимодействие географически разделенных серверов и устройств хранения данных.
- Сети IP используются для создания сетей наибольшего масштаба, включая Internet, поэтому большинство проблем, связанных с масштабированием и сетевыми "заторами", в них решены.

Компании, развивающие технологию IP Storage, отмечают, что протокол IP победил и пора переходить от концепции "IP поверх всего" (Ethernet, Token

Ring, ATM, Gigabit Ethernet и т.д.) к "все поверх IP" (включая блоки данных команд SCSI или CDB, SCSI-команды/результат-по-IP и т.д.).

В главе 4 описаны два глобальных системных окружения: каналы ввода-вывода и сети. Каналы, например SCSI, обычно работают на небольших расстояниях и предназначены для решения узкого круга задач, поэтому бóльшая часть работы возложена на аппаратное обеспечение. Сети, как правило, могут работать на больших расстояниях, по своей природе более универсальны и немалая часть функций реализована в виде программного обеспечения. Технология Fibre Channel стала попыткой централизации этих двух окружений на уровне каналов, в то время как IP Storage разрабатывается для обеспечения централизации на сетевом уровне.

Новым термином *глобальная сеть хранения данных* (Storage Wide Area Network — SWAN) описывается развертывание и использование технологий IP Storage поверх глобальных сетей на основе протокола IP.

В следующих разделах рассматриваются различные технологии, формирующие стандарт IP Storage. Также приводятся ссылки на реализацию этих технологий от компании Microsoft.

8.1.2 Протокол iSCSI

Протокол iSCSI (сокращение от Internet SCSI) позволяет реализовать одно или несколько подключений TCP/IP между устройствами, обменивающимися командами SCSI, ответами и информацией о состоянии. Другими словами, iSCSI представляет собой протокол для соединения "точка–точка", инкапсулирующий команды SCSI, ответы и информацию о состоянии.

На рис. 8.2 показано, как компоненты IP, TCP, iSCSI и SCSI работают вместе в процессе инкапсуляции. Пакет iSCSI содержит данные и команду SCSI для стека TCP/IP. Заголовок iSCSI содержит информацию об извлечении и интерпретации команды SCSI, которая размещена в пакете. Заголовок пакета TCP отвечает за гарантированную и последовательную доставку пакетов. Пакет TCP содержит данные и полезную нагрузку (payload) пакета IP. Заголовок IP используется в процессе маршрутизации.

Из трех основных протоколов работы с хранилищами IP — iSCSI, FCIP (Fibre Channel over IP) и iFCP (Internet Fibre Channel Protocol) — протокол iSCSI является единственным, не связанным с Fibre Channel. Как видите, на

Рис. 8.2. Инкапсуляция протокола iSCSI

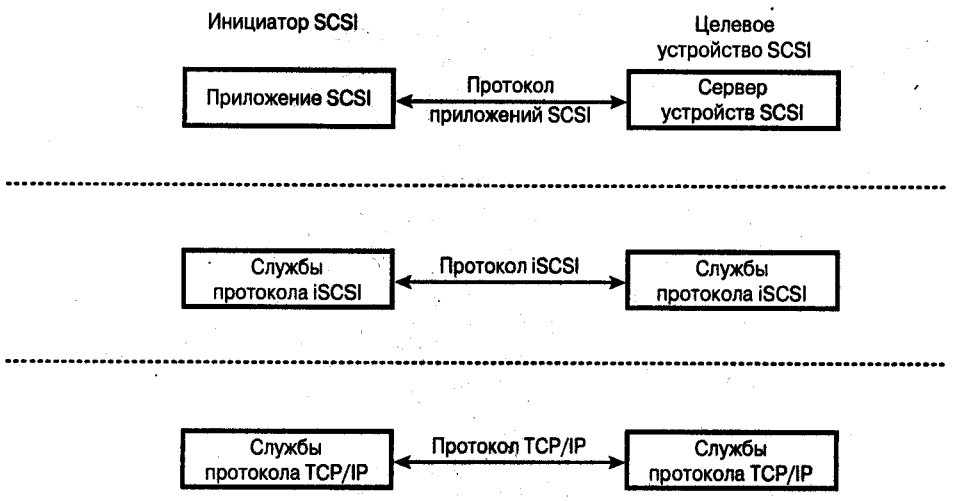


Рис. 8.3. Уровни протокола iSCSI

рис. 8.2 Fibre Channel не упоминается, поскольку протокол iSCSI разрабатывался без поддержки Fibre Channel.

Протокол iSCSI размещен поверх существующих уровней протоколов TCP, IP и низкоуровневых аппаратных протоколов, поддерживающих TCP/IP (например, Ethernet и Gigabit Ethernet).

Как можно заметить (рис. 8.3), SCSI — это протокол уровня приложения. Протокол iSCSI предоставляет услуги протоколу приложений SCSI и применяет TCP/IP для обеспечения гарантированной доставки, маршрутизации и т.д.

Все устройства iSCSI (целевые и устройства-инициаторы) имеют два разных имени.

1. *Адрес iSCSI*, который состоит из адреса IP, порта TCP и имени iSCSI и имеет формат "`<имя домена>:<номер порта>:<имя iSCSI>`".
2. *Имя iSCSI*, которое имеет наглядный формат, например "`Имя_FQN.Поставщик_диска.Модель_диска.Номер`".

Служба именования iSNS (Internet Storage Name Service) является общей для iSCSI, iFCP и FCP (Fibre Channel Protocol). Протоколы iFCP и FCP рассматриваются в разделе 8.1.5. Кроме использования iSNS в качестве службы именования, протокол iSCSI имеет спецификацию, в которой описана информационная база управления (Management Information Base — MIB) для управления устройствами iSCSI на основе протокола SNMP. Кроме того, протокол iSCSI определяет процесс реализации удаленной загрузки.

Протокол iSCSI устанавливает сеансы связи между инициатором и целевым устройством. Один или несколько сеансов протокола TCP могут использоваться одним или несколькими сеансами iSCSI. После установки сеанса две стороны (инициатор и целевое устройство) обмениваются такими параметрами, как безопасность, размер буфера и возможность отправки незапрошенных данных. Сеанс iSCSI может закрываться стандартным образом: посредством завершения регистрации или в связи с возникновением ошибки. Независимо от количества использованных сессий TCP, протокол iSCSI гарантирует, что команда SCSI и ответы на нее будут доставлены в правильном порядке. Обратите внимание: протокол TCP гарантирует последовательную доставку для определенного сеанса TCP, но не обеспечивает синхронизации передаваемых данных между различными сеансами TCP. Таким образом, синхронизации сеансов TCP возлагается на протокол iSCSI. Можно перечислить ряд требований к протоколу iSCSI.

- Разные команды SCSI *могут* передаваться через различные сеансы TCP.
- Все данные и параметры, соответствующие определенной команде, должны передаваться в рамках того же сеанса TCP, по которому передавалась команда SCSI.
- В протоколе iSCSI определена концепция тега-инициатора. Все ответы будут иметь соответствующий тег инициатора, который высылается вместе с первоначальной командой. Инициатор должен обеспечить уникальность тегов и исключить возможность повторного использования тега, пока инициатор не получит все ответы на соответствующую команду. Тег должен быть уникальным в рамках инициатора (операционная система Windows NT является многозадачной, и инициатор может работать для нескольких процессов и приложений).
- Протоколом iSCSI определена концепция нумерации команд, которая обеспечивает последовательную доставку команд через несколько сеансов TCP.
- Кроме того, в протоколе iSCSI определен механизм CRC типа "точка–точка". Проверка CRC на втором уровне (например, на уровне Gigabit Ethernet) или на третьем уровне (контрольные суммы TCP/IP) может оказаться ненадежной, особенно если на пути передачи пакетов размещены другие устройства IP, например трансляторы сетевых адресов, маршрутизаторы и т.д. Поставщики подсистем хранения всегда с большой осторожностью относились к методам проверки целостности данных.

Протокол iSCSI имеет свои недостатки. При его использовании возникают проблемы, связанные с безопасностью, управлением сетевыми "заторами" и качеством обслуживания. Но эти вопросы в основном касаются работы сетей на базе протоколов TCP/IP, проблемы которых уже хорошо изучены.

8.1.3 Реализация протокола iSCSI в Windows NT

Поддержка протокола iSCSI в Windows NT активно внедряется компанией Microsoft. Несмотря на это, Windows Server 2003 не обеспечена поддержкой iSCSI по умолчанию.

На рис. 8.4 показана архитектура реализации протокола iSCSI в Windows NT.

Инициатор iSCSI реализован в виде драйвера мини-порта для модели SCSIPort или Storport.

Динамически подключаемая библиотека обнаружения iSCSI отслеживает все изменения и выступает в роли единственного хранилища для всех номеров LUN, обнаруженных тем или иным способом, включая использование

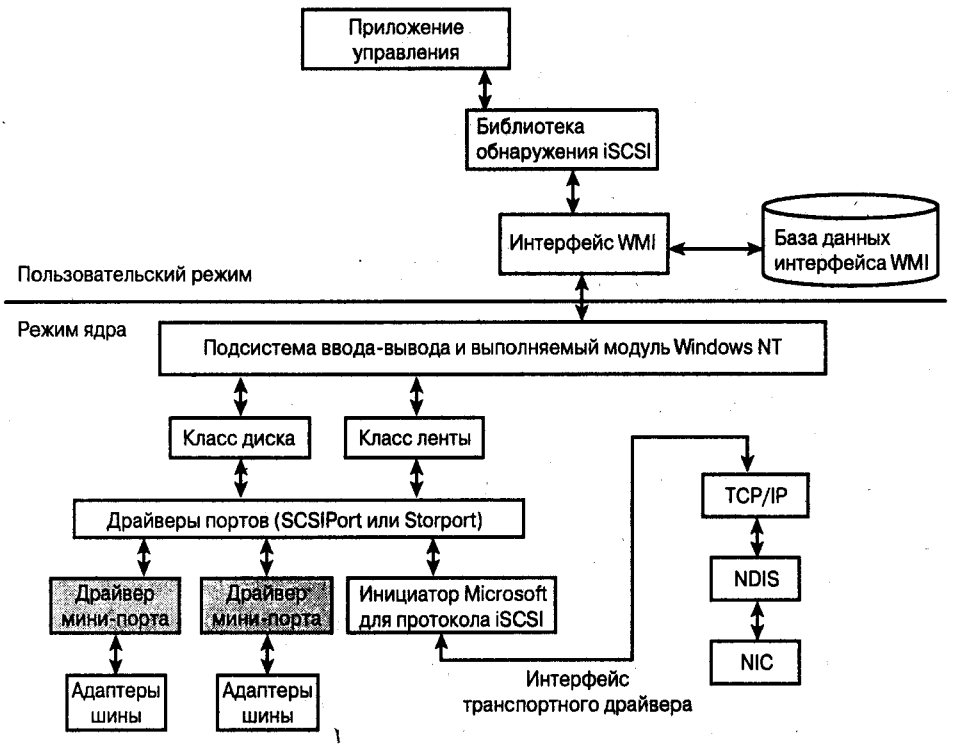


Рис. 8.4. Архитектура iSCSI

клиентов iSNS или уведомление портов. Библиотека обнаружения предоставляет API обнаружения новых LUN для приложений управления и, если это необходимо, методы, с помощью которых приложение управления может дать указание библиотеке обнаружения зарегистрировать новый LUN.

Рассмотрим, каковы планы компании Microsoft относительно протокола iSCSI.

- Основное внимание уделяется реализации протокола iSCSI на платформе Windows Server 2003, однако рассматривается и поддержка протокола iSCSI в Windows 2000. Только объявление о выпуске финальной версии продукта можно считать достоверным источником информации о доступности этого продукта на рынке.
- Основное внимание уделяется реализации протокола iSCSI на платформе Windows Server 2003. Тем не менее код поддержки iSCSI для Windows 2000 и Windows XP также был предложен компанией Microsoft через несколько месяцев после предоставления кода поддержки iSCSI в Windows Server 2003.
- Компания Microsoft предоставила код клиента и сервера службы iSNS.
- В качестве механизма защиты данных и обеспечения безопасности Microsoft предполагает использование протокола IPSec.
- Основное внимание уделяется реализации инициатора iSCSI, и на данный момент нет конкретных планов по реализации функциональности точки назначения iSCSI на платформе Windows NT.
- Взаимодействие между библиотекой обнаружения и инициатором iSCSI будет вестись через интерфейс WMI. Библиотека обнаружения может быть преобразована в службу (точные сведения на этот счет отсутствуют).
- Основная особенность поддержки iSCSI в Windows NT состоит в разделении процессов обнаружения и доступа устройств. Приложение управления инициирует доступ через динамическую библиотеку обнаружения, которая, в свою очередь, связывается с драйвером мини-порта (через интерфейс WMI). Драйвер мини-порта передает событие `BusChangeDetected` (описано в Windows NT SDK), запуская процедуру перечисления устройств. С этого момента перечисление устройств проводится обычным образом; например, обнаруженному устройству будет отправлена команда `Report LUNs`.
- Настоятельно рекомендуется использовать протокол IPSec для обеспечения безопасности.
- Для создания отчетов и управления ими рекомендуется применять интерфейс WMI. Ожидается, что некоторые классы WMI будут обяза-

тельны для реализации, в то время как другие классы — лишь рекомендованы. Точные данные доступны только при подписании соглашения с компанией Microsoft[1].

Несмотря на то что большая часть информации, представленной в этой главе, основана на предположениях, всеобщее распространение протокола iSCSI возможно только при наличии соответствующей поддержки со стороны операционной системы. Это означает, что читателю следует ознакомиться с текущими планами производителя операционной системы.

8.1.4 Протокол FCIP

Протокол FCIP (Fibre Channel over IP) предоставляет возможность сохранения существующих инвестиций в оборудование и позволяет объединить географически удаленные сети хранения данных по протоколу туннелирования на основе стека TCP/IP. В спецификации IETF FCIP описаны следующие особенности этого протокола.

- Инкапсуляция кадров Fibre Channel, передающихся с помощью TCP/IP, включая инкапсуляцию, необходимую для создания виртуальной линии связи Fibre Channel, которая используется для подключения устройств Fibre Channel к элементам связной архитектуры.
- Спецификация среды TCP/IP, включая безопасность, управление сетевыми "заторами" и восстановление после ошибок. Протокол FCIP требует от Fibre Channel и TCP совместного вклада в обработку ошибок и восстановление после сбоев.

На рис. 8.5 приведена схема инкапсуляции FCIP.

Рис. 8.5. Инкапсуляция FCIP

Полезной нагрузкой в данном случае являются данные SCSI. Они инкапсулируются в пакетах FCP (Fibre Channel Protocol), которые, в свою очередь, инкапсулируются в пакеты протокола FCIP. Протокол TCP обрабатывает пакеты FCIP как собственную полезную нагрузку. При использовании такой инкапсуляции протокол IP не "подозревает" о связи данных с Fibre Channel, а протоколу Fibre Channel точно также "неизвестно" о наличии протокола IP.

Протоколы инкапсуляции обычно характеризуются наличием определенных накладных расходов, возникающих при перемещении данных между

[1]Для получения необходимой информации следует связаться с представителями Microsoft по электронной почте (`iscsi@microsoft.com`).

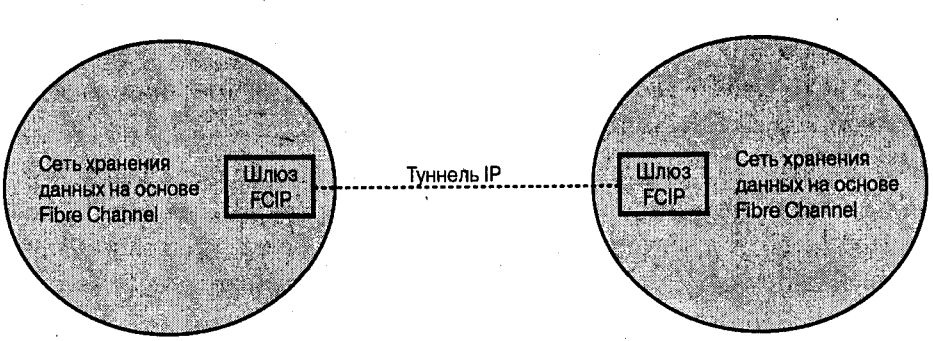


Рис. 8.6. Протокол FCIP, обеспечивающий взаимодействие двух сетей хранения данных

уровнями. Протокол FCIP не стал исключением из правила. Для сети на базе IP (рис. 8.6) шлюзы FCIP обрабатываются в качестве устройств IP; однако для сети Fibre Channel шлюзы FCIP воспринимаются в качестве устройств именно Fibre Channel. Инкапсуляция Fibre Channel без дополнительных преобразований возможна только при передаче данных между двумя шлюзами FCIP.

На рис. 8.6 показано, как для объединения двух изолированных "островов" SAN применяется FCIP. При этом в SAN точно так же используется технология Fibre Channel, и вся адресация, маршрутизация и другие операции SAN выполняются обычным образом, т.е. как в обычной сети на основе технологии Fibre Channel. Протокол FCIP полагается на TCP/IP в вопросах маршрутизации и управления, включая управление сетевыми "заторами". Кроме того, FCIP использует возможности TCP/IP и Fibre Channel для восстановления данных, а также преобразует адреса Fibre Channel в адреса IP. Протокол FCIP используется для установки связи между портами типа E. (Различные типы портов Fibre Channel рассматриваются в главе 4.)

Обычное приложение FCIP поддерживает:

- удаленное резервное копирование и восстановление данных;
- удаленное резервное копирование в качестве элемента системы по восстановлению данных после крупномасштабных катастроф;
- синхронное зеркальное отражение и совместное использование данных между географически распределенными SAN (при достаточной пропускной способности сети между двумя шлюзами FCIP, т.е. выделенной линии между двумя шлюзами), включая совместное использование пула устройств хранения.

Протокол FCIP не требует внесения изменений в сеть на основе технологии Fibre Channel. На рис. 8.7 показаны стеки протоколов FCIP и iFCP (про-

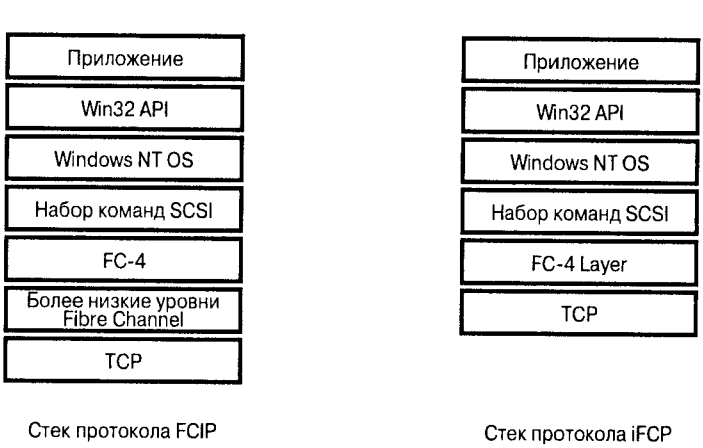


Рис. 8.7. Сравнение стеков протоколов FCIP и iFCP

токол iFCP рассматривается в разделе 8.1.5). Обратите внимание на функциональные уровни Fibre Channel, включая FC-4 и более низкие уровни Fibre Channel, остающиеся неизменными при использовании FCIP. По сравнению с типичной иерархией файловой системы, управления томами, уровнями класса и порта, описанными в главе 1, уровень команд SCSI в этой модели присутствует на уровне драйвера порта. Таким образом, уровень FC-4 и более низкие уровни, показанные на рис. 8.7, должны быть реализованы в аппаратном обеспечении ниже уровня драйвера порта Windows NT. Обычно от аппаратного обеспечения требуется поддержка работы драйвера порта. В данном случае стек TCP/IP также часто реализуется в виде программного обеспечения.

По сравнению со схемой "IP поверх Ethernet", протокол FCIP имеет определенные преимущества. В то время как пакеты Ethernet обычно содержат около 1500 байт данных, кадры FCIP содержат около 2000 байт данных. Однако если рассмотреть реализацию кадров Ethernet в технологии Gigabit Ethernet, поддерживающей кадры размером 8 Кбайт и более, это преимущество сходит на нет.

Проблема протокола FCIP состоит в необходимости поддержки двух сетей. Ожидается, что FCIP будет применяться в качестве средства расширения пропускной способности каналов или удаленного зеркального отражения данных на существующее устройство, а не в роли "нового" протокола, внедряемого на уровне узлов.

8.1.5 Протокол iFCP

Межшлюзовой протокол iFCP (Internet Fibre Channel Protocol) позволяет двум сетям на базе Fibre Channel взаимодействовать друг с другом через промежуточную сеть на основе TCP/IP. Компоненты связной архитектуры Fibre Channel заменяются коммутацией TCP/IP и элементами маршрутизации. В то время как FCIP нацелен на соединение сетей хранения данных, протокол iFCP больше предназначен для подключения отдельных устройств Fibre Channel к сетям на базе протокола IP.

Межшлюзовому протоколу iFCP требуется два устройства-шлюза для предоставления связи остальным устройствам в SAN на основе Fibre Channel. На рис. 8.8 приведена типичная схема применения iFCP.

Два шлюза iFCP устанавливаются как пограничные устройства сети IP. К шлюзам могут быть подключены такие устройства Fibre Channel, как жесткие диски, накопители на магнитной ленте или серверы. Как показано на рис. 8.8, два шлюза устанавливают соединение IP, которое используется для передачи данных сеансов между устройствами. Таким образом, iFCP работает на уровне соединений между устройствами, а FCIP больше похож на мост Ethernet, передающий все пакеты между двумя "островами" SAN.

Протокол iFCP поддерживает протокол FCP (Fibre Channel Protocol) — стандарт для передачи команд и ответов SCSI по последовательной линии

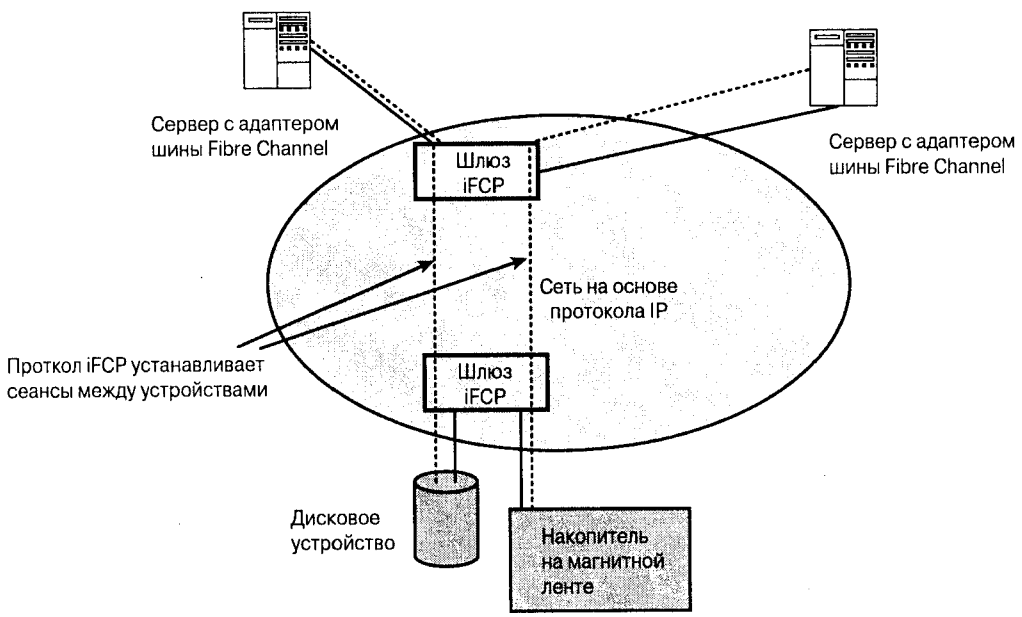


Рис. 8.8. Схема использования iFCP

связи. Как показано на рис. 8.7, стек протокола iFCP заменяет уровень FC-2 (транспортный уровень стека Fibre Channel, который рассматривается в главе 4) транспортным уровнем протокола TCP, однако уровень FC-4 остается неизменным. Данные передаваемых сообщений и маршрутизации iFCP не выходят за границу шлюзов, т.е. несмотря на наличие соединения между устройствами, сети хранения данных на базе Fibre Channel остаются изолированными. Эта ситуация напоминает блокирование широковещательных кадров на маршрутизаторе. Протокол iFCP предоставляет связь только портам типа F (дополнительная информация о типах портов Fibre Channel и их возможностях приводится в главе 4), а также создает несколько сеансов связи TCP/IP между устройствами Fibre Channel.

Сравнивая стеки протоколов FCIP и iFCP (см. рис. 8.7), можно заметить, что FCIP реализует все уровни протокола Fibre Channel, а iFCP — только уровень 4. Таким образом, протокол FCIP ориентирован на технологию Fibre Channel.

Протокол iFCP "полагается" на TCP/IP для обеспечения надежной доставки данных. Это означает, что сеть IP на более низких уровнях может не предоставлять надежной доставки. Спецификация протокола iFCP допускает повышенные задержки в работе сети, что помогает работать в сетях с низкой надежностью и небольшими задержками, которые выглядят, как сети с большими задержками и высокой надежностью. Это реализуется за счет механизма надежной последовательной доставки пакетов протокола TCP. Так как iFCP задействует несколько соединений TCP/IP, он более эффективен и устойчив к заторам в сети по сравнению с использованием одного соединения TCP/IP для передачи всех данных между устройствами.

Устройства шлюзов iFCP предоставляют возможности регистрации устройств хранения на сервере имен iSNS (дополнительная информация приводится в следующем разделе).

8.1.6 Служба iSNS

Служба *iSNS* (Internet Storage Name Service) предоставляет методы регистрации и обнаружения устройств хранения. Поддержка протокола iSNS может быть реализована на серверах и устройствах хранения. Служба iSNS предоставляет единую модель, которая может применяться к устройствам SCSI и Fibre Channel. Устройства Fibre Channel регистрируются службой iSNS с помощью шлюза iFCP, в то время как устройства iSCSI регистрируются самой службой iSNS. Инициаторы обнаруживают серверы iSNS одним из двух способов.

1. С помощью статической информации.
2. С помощью протокола SLP (Service Location Protocol).

Служба iSNS предоставляет функции зонирования благодаря концепции доменов обнаружения, что позволяет администратору указывать группы устройств. Когда член группы запрашивает сервер iSNS, в ответном сообщении указываются устройства только из той же группы. Кроме того, серверы iSNS предоставляет услуги уведомления, например когда к сети подключается новое целевое устройство.

Серверы iSNS играют важную роль в обеспечении безопасности хранилищ. Домены обнаружения помогают усилить политику безопасности. Другими словами, серверы iSNS хранят и навязывают политику управления доступом (в которой описано, какие инициаторы могут получать доступ к определенным устройствам). Кроме того, серверы iSNS играют определенную роль в обеспечении регистрации устройства с помощью сертификата с открытым ключом, после чего сервер может предоставлять полученную информацию о сертификате другим устройствам.

Компания Microsoft активно занимается продвижением протокола iSNS, однако еще не сообщила о планах выпуска сервера iSNS, клиента iSNS или обоих продуктов iSNS.

8.1.7 Методы эффективного внедрения TCP/IP

С развитием технологии IP Storage необходимость эффективной реализации протокола TCP/IP стала еще более актуальной. Анализ показал, что обработка пакетов TCP/IP и даже подсчет их контрольных сумм может существенно загрузить центральный процессор. Кроме того, данные копируются неоднократно, что еще больше увеличивает накладные расходы по их обработке.

Например, протокол TCP должен поддерживать последовательную доставку (которая не предоставляется IP), поэтому пакеты, пришедшие раньше своей очереди, должны быть помещены на временное хранение. Это означает, что данные копируются во временный буфер и позднее копируются из него в пользовательский буфер. Требования к аппаратному обеспечению только в случае поддержки последовательной доставки могут быть весьма существенными. Линия связи WAN со скоростью передачи 1 Гбит/с может потребовать 16 Мбайт оперативной памяти для хранения пакетов, пришедших раньше своей очереди, и для восстановления последовательности пакетов. При быстродействии 10 Гбит/с необходимый объем возрастает до 125 Мбайт. Другими словами, необходимо сократить количество копирований данных в буфер, что можно сделать с помощью более эффективного программного или аппаратного обеспечения.

Технологии оптимизации TCP, разработанные в последнее время, рассчитаны на более активное применение аппаратных компонентов сетевого адап-

тера. С увеличением потребности в быстродействии TCP/IP (особенно учитывая развитие технологии IP Storage) оптимизация TCP стала крайне важной. Далее вкратце описаны некоторые из предлагаемых технологий.

- Перемещение всего стека протокола TCP/IP на уровень аппаратного обеспечения. Хотя в аспекте производительности это наилучший вариант, он наиболее амбициозен, поскольку требует решения некоторых сложных проблем, например координации различных стеков протокола TCP/IP, работающих на двух сетевых адаптерах одного сервера под управлением Windows NT.
- Поддержка аппаратным обеспечением перемещения данных и генерации контрольных сумм, причем управлением соединениями будет заниматься программное обеспечение.
- Перенос на уровень аппаратного обеспечения стандартной обработки; при этом обработка исключений осуществляется программно.
- Перенос на уровень аппаратного обеспечения обработки данных IPSec и даже некоторых данных iSCSI.

В Windows 2000 была представлена версия 5.0 спецификации NDIS (Network Driver Interface Specification), которая содержит поддержку оптимизации TCP/IP. В частности, в Windows 2000 поддерживаются описанные ниже функции.

- Перенос вычисления контрольных сумм TCP/IP на уровень аппаратного обеспечения как для отправки (генерация контрольной суммы), так и для приема (проверка контрольной суммы).
- Перенос сегментации TCP на уровень аппаратного обеспечения. При этом, если объем данных превышает максимальную единицу передачи (MTU), аппаратное обеспечение будет сегментировать данные в несколько пакетов.
- Перенос на уровень аппаратного обеспечения реализации IPSec. Стандарт IPSec (для протоколов IPv4 и IPv6) обеспечивает целостность данных и аутентификацию на уровне пакетов. Протокол IPSec может использоваться в двух режимах: транспортном, который обеспечивает целостность данных и аутентификацию между пользовательскими приложениями, и туннельном, обеспечивающем безопасность обмена данными между двумя маршрутизаторами. На уровень аппаратного обеспечения могут быть перенесены оба режима.
- Быстрая передача пакетов, когда код маршрутизации Windows 2000 передает пакет с одного сетевого порта на другой, минуя копирование пакета в память узла.

8.2 Стандарт InfiniBand

Новый стандарт (и архитектура) InfiniBand предназначен для соединения коммутируемых связных архитектур между узлом и хранилищем или сетевыми периферийными устройствами. Разработкой спецификации InfiniBand занимается ассоциация IBTA (InfiniBand Trade Association). Эта ассоциация была сформирована после слияния двух конкурирующих спецификаций: Future I/O, продвигавшейся усилиями компании Intel, и Next Generation I/O, разрабатываемой компаниями IBM, HP и Compaq.

Архитектура InfiniBand предлагает несколько модификаций, в частности замену шины ввода-вывода (например, PCI) сетью коммутируемой связной архитектуры. Последовательная коммутируемая связная архитектура обладает определенными преимуществами по сравнению с обычной шиной ввода-вывода. Наиболее заметное преимущество состоит в поддержке большего количества устройств на значительно большем расстоянии с использованием на порядок меньшего количества электрических проводников. Кроме того, связная архитектура поддерживает несколько одновременных сеансов передачи данных и обеспечивает устойчивость к ошибкам. Прежде чем подробно обсуждать архитектуру InfiniBand, рассмотрим недостатки шины PCI.

Хотя шина PCI обладает определенными позитивными свойствами, например в свое время она заменила сразу несколько конкурирующих стандартов (ISA, EISA, MCA), в свете быстрого развития центральных процессоров, памяти и технологий периферийных устройств ее ограничения становятся все более явными. Приведем некоторые из них.

- Хотя шина PCI обладала достаточным быстродействием в момент своего появления, ее пропускная способность более не соответствует современным требованиям: шина FSB центрального процессора имеет пропускную способность 1066 Мбит/с, а для устройств Gigabit Ethernet и высокоуровневых устройств хранения (SCSI 3) шина PCI станет "бутылочным горлышком", ограничивающим эффективность работы.
- Шина PCI имеет определенные проблемы в управлении вообще и в обнаружении ошибок в частности. Одна некорректно работающая плата PCI может привести к неисправности всей системы, в то время как обнаружить поврежденную плату крайне сложно.
- Существуют физические ограничения на длину шины и скорость передачи данных, кроме того ограничено количество шин. При использовании наибольшей скорости передачи данных к шине можно подключить только одно периферийное устройство.

Обратите внимание: хотя технология InfiniBand изначально позиционировалась как замена шины PCI, с появлением технологии 3GIO роль InfiniBand в качестве PCI нового поколения заметно снизилась.

8.2.1 Преимущества технологии InfiniBand

Технология InfiniBand предоставляет множество преимуществ.

- Сокращение сложности кабельных подключений, так как InfiniBand позволяет заменить три кабеля — Ethernet, кабели к подсистеме хранения и кабели межпроцессного взаимодействия — одним соединением. В результате намного упрощается аппаратная структура монтажной платы, разводка соединителей на платах и общая структура системы. Более того, монтажные платы подключаются к граничным разъемам и из стойки доступна только пара высокоскоростных соединений.
- Встроенное обнаружение ошибок, упрощающее поиск неисправного компонента.
- Сокращение потребления пропускной способности памяти, так как сокращается количество операций копирования в память и из нее.
- Сокращение контекстных переключений, в том числе переключения между пользовательским режимом и режимом ядра.
- Уменьшение накладных расходов, например при вычислении контрольных сумм TCP/IP.
- Обеспечение отказоустойчивости за счет использования альтернативных маршрутизаторов в среде связной архитектуры и таких избыточных компонентов, как маршрутизаторы и коммутаторы. Избыточные маршруты позволяют выполнять отправку данных по нескольким маршрутам и балансировать общую нагрузку. Более того, компоненты InfiniBand поддерживают "горячую" замену.
- Технология InfiniBand обеспечивает эффективную загрузку монтажной платы из внешнего устройства, что позволяет еще более упростить аппаратную структуру платы.

8.2.2 Архитектура InfiniBand

Архитектурой InfiniBand определена топология логического подключения "точка–точка", которое формируется поверх связной архитектуры. Непосредственно коммутируемая связная архитектура содержит следующие компоненты:

- адаптеры канала узла;

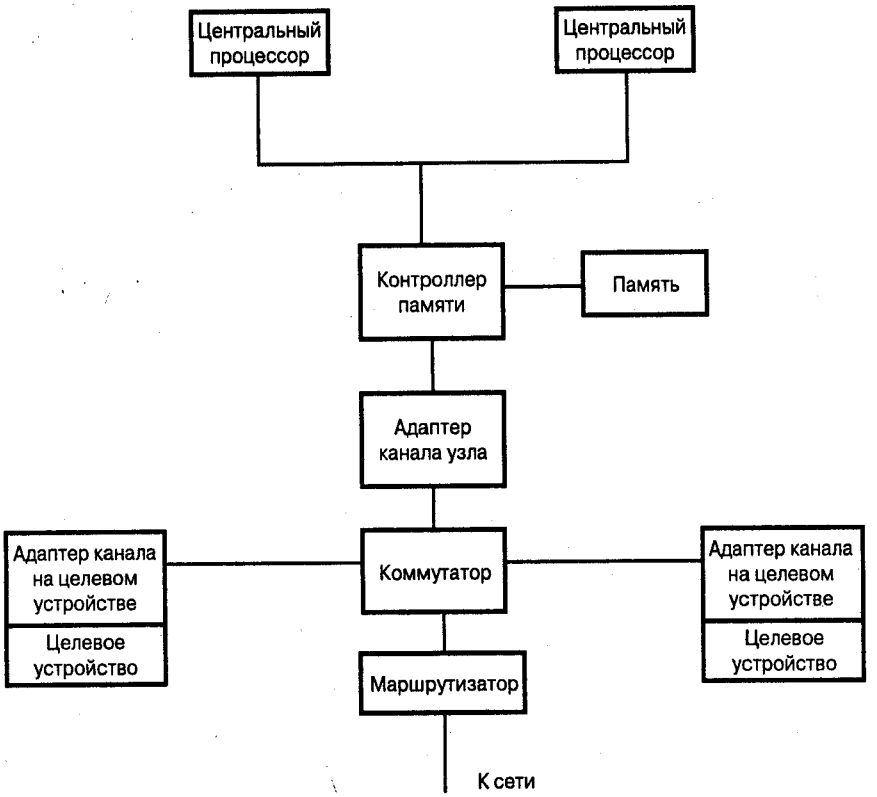


Рис. 8.9. Шина ввода-вывода InfiniBand

- адаптеры канала целевого устройства;
- коммутаторы InfiniBand;
- физический носитель.

На рис. 8.9 демонстрируется структура шины ввода-вывода InfiniBand и взаимоотношение различных компонентов, которые более подробно рассматриваются далее.

Адаптер канала узла (host channel adapter — HCA) представляет собой устройство, которое работает в качестве точки соединения центрального процессора узла и связной архитектуры InfiniBand. Адаптеры каналов узла тесно связаны с серверами и размещаются рядом с ними. Адаптер HCA уменьшает количество прерываний, сдерживает "вмешательство" операционной системы и может обмениваться данными непосредственно с памятью. Каждый адаптер имеет уникальный идентификатор, созданный на основе адреса протокола IPv6.

Адаптер канала целевого устройства (target channel adapter — TCA) является аналогом адаптера HCA, предназначенного для периферийных устройств, а не ЦПУ узлов. Адаптеры TCA обычно взаимодействуют с периферийными устройствами и также размещены недалеко от них.

Коммутатор InfiniBand предоставляет возможность подключения нескольких адаптеров TCA к одному адаптеру HCA. Кроме того, коммутаторы поддерживают маршрутизацию нескольких логических элементов связной архитектуры (они называются *подсетями*). В то время как коммутаторы InfiniBand поддерживают подключения *внутри* подсети, маршрутизаторы InfiniBand реализуют подключение *между* подсетями. Маршрутизаторы обычно настраиваются так, чтобы дополнять возможности коммутаторов новыми функциями.

Технология InfiniBand поддерживает использование обычных медных или оптических кабелей, при этом линии связи могут достигать 17 метров для медных носителей и 100 метров для оптических кабелей. Подключение InfiniBand может включать в себя разное количество линий связи, каждая из которых имеет быстродействие 2,5 Гбит/с. Поддерживаются комбинации от 1X (1 кабель) до 4X (4 кабеля) для быстродействия 10 Гбит/с или 12X (12 кабелей) для быстродействия 30 Гбит/с.

Архитектура InfiniBand включает в себя не только физические, но и логические компоненты, описанные далее.

8.2.2.1 Линии связи InfiniBand и сетевые уровни

На уровне канала передачи данных (уровень 2 модели OSI) каждая линия InfiniBand делится максимум на 16 потоков и минимум на 2 потока. Один из потоков всегда выделяется на управление связной архитектурой. Потоки получают приоритеты QoS, а поток управления получает наивысший приоритет (виртуальный поток 15; потоки 0–14 используются для передачи данных) и соответствующий уровень QoS.

Следует отметить, что, в отличие от потока, канал передачи данных представляет собой физический компонент. Поток позволяет двум узловым системам связываться друг с другом. Установившие сеанс связи системы могут поддерживать различное количество виртуальных потоков, для чего в спецификации InfiniBand описан соответствующий алгоритм. Установившие связь конечные точки называются *парой очереди* (queue pair). Пары очереди имеют связанные с ними буферы приема и отправки. Каждый виртуальный поток (virtual lane — VL) предоставляет собственное управление потоком на кредитной основе. Пакеты управления обеспечивают перечисление устройств, управления подсетями и устойчивость к ошибкам.

Технология InfiniBand поддерживает одновременную передачу данных без коллизий[2]. Для обеспечения надежной передачи данных технология InfiniBand использует сквозное управление потоком и не одну, а две проверки целостности по алгоритму CRC. Управление потоком выполняется с помощью выделения буферов на принимающем узле и передачи количества буферов отправляющему узлу. Этим значением указывается количество буферов данных, которые можно отправить без подтверждения приема. Два значения CRC подсчитываются и передаются вместе с данными. Получающая сторона также подсчитывает эти значения. Значение CRC размером 32 бит создается для связи "точка–точка". В то же время данные могут передаваться через промежуточные узлы. Между промежуточными узлами или между промежуточным и конечным узлами используются значения размером 16 бит.

В архитектуре InfiniBand базовой единицей передачи является *сообщение* (message). Сообщение может быть отправлено или принято с помощью операции RDMA, операции отправки или приема или с помощью операции групповой отправки. Операция удаленного прямого доступа к памяти (Remote Direct Memory Access — RDMA) обеспечивает непосредственный обмен сообщениями между памятью узлов, минуя прерывания операционной системы и не требуя использования служб. Технологией InfiniBand определено шесть режимов передачи данных.

1. Надежное соединение, при котором аппаратное обеспечение отвечает за генерацию и проверку номеров последовательности пакетов, реализуя тем самым надежность соединения; кроме того, аппаратное обеспечение отвечает за обнаружение дублированных и потерянных пакетов, а также восстановление после ошибок.
2. Ненадежное соединение.
3. Надежная дейтаграмма.
4. Ненадежная дейтаграмма.
5. Соединение групповой отправки (реализация необязательна).
6. Пакетная передача (реализация необязательна).

Сообщение InfiniBand может состоять из одного или нескольких пактов. Пакеты могут иметь размер до 4096 байт. На виртуальных потоках (VL) возможно чередование пакетов. Маршрутизация проводится на уровне пакетов,

[2]Хотя InfiniBand не поддерживает задержки случайной передачи и задержки коллизий Ethernet, эта технология предоставляет механизмы управления потоком и дает возможность задерживать передачу данных в случае необходимости. Существенная задержка в получении/отправке данных и медленная передача данных будут обрабатываться приложением одинаково.

причем пакеты, маршрутизируемые между подсетями, содержат глобальный заголовок, благодаря которому и реализуется маршрутизация.

Кроме того, в архитектуре InfiniBand определен сетевой уровень, обеспечивающий маршрутизацию между различными подсетями. Подсети позволяют локализовать передачу данных в пределах определенной подсети; например широковещательные пакеты и пакеты групповой отправки не выходят за рамки подсети. Подсети предоставляют функции, аналогичные возможностям виртуальной LAN (VLAN), и могут использоваться для обеспечения безопасности. Для каждого устройства в подсети используется 16-разрядный идентификатор, уникальный в пределах подсети. Каждый маршрутизируемый пакет содержит адреса IPv6 исходного узла и узла назначения.

8.2.3 Компания Microsoft и технология InfiniBand

Компания Microsoft в свое время указала, что в неопределенном будущем семейство операционных систем Microsoft Windows будет поддерживать технологию InfiniBand. В третьем квартале 2002 года Microsoft и другие лидеры индустрии информационных технологий сообщили о пересмотре своих планов поддержки технологии InfiniBand; в частности, Microsoft сообщила о перераспределении ресурсов, направленных ранее на поддержку технологии InfiniBand, на другие области, включая IP Storage поверх Gigabit Ethernet. Таким образом, данный раздел может показаться неполным, поскольку не существует конкретной реализации InfiniBand от Microsoft, которую можно было бы описать в полной мере.

8.3 Сложности практической реализации

Администраторы и руководители отделов информационных технологий должны обратить внимание на то, что поддержка протокола iSCSI не обеспечена в базовой версии Windows Server 2003. Таким образом, все системы на базе iSCSI, которые будут использоваться в Windows до выхода официальной поддержки, должны полностью проектироваться производителем. Заранее невозможно сказать, насколько успешно такие системы будут взаимодействовать с реализацией протокола iSCSI в Windows.

Производители, стремящиеся реализовать системы iSCSI для платформы Windows, оказываются в запутанной ситуации. Рекомендуется связаться с компанией Microsoft (`iscsi@microsoft.com`) и принять участие в разработке реализации протокола iSCSI. На данный момент существует мнение, что проектируемый стек iSCSI будет поддерживаться как в Windows 2000, так и в Windows Server 2003. Поскольку для работы некоторых систем от производителей потребуется создание драйвера мини-порта, следует обратить

внимание на то, что модель драйверов Storport в Windows Server 2000 не поддерживается (модель драйверов Storport рассматривается в главе 2). Таким образом, производителю придется принимать решение о разработке драйвера на основе модели SCSIPort или о создании двух драйверов для моделей Storport и SCSIPort.

Высока вероятность быстрого развития технологии IP Storage, после того как будет предоставлена широкая поддержка со стороны операционных систем. Некоторые версии Linux уже поддерживают iSCSI, т.е. в данном случае Windows придется уже наверстывать упущенное. Новые возможности, предоставляемые IP Storage, с высокой долей вероятности окажутся весьма привлекательными и потребуют значительных ресурсов.

8.4 Резюме

Набор технологий IP Storage предназначен для доступа к устройствам хранения по сетям на основе протокола IP. Важность этой технологии растет вместе с определением стандартов и появлением готовых решений. Компания Microsoft сообщила, что операционная система Windows NT будет поддерживать, как минимум, протокол iSCSI, который представляет собой один из ключевых компонентов IP Storage.

С ростом систем на базе IP Storage повышается необходимость создания еще более эффективной реализации стека протоколов TCP/IP (и смежных с ним протоколов). Один из способов состоит в обработке данных TCP/IP на уровне аппаратного обеспечения. С выходом стандарта NDIS 5.1 компания Microsoft сообщила о поддержке в Windows NT сетевых адаптеров с частичной аппаратной реализацией протокола TCP.

Технология InfiniBand — это новый стандарт, с помощью которого коммутируемая связная архитектура предназначается для соединения центральных процессоров узлов и периферийных устройств хранения данных. Хотя некоторые компании продолжают разработку InfiniBand, компания Microsoft решила отказаться от ее поддержки в Windows NT. Таким образом, будущее InfiniBand находится под большим вопросом, поэтому имеет смысл обратить более пристальное внимание на реализацию технологии IP Storage поверх Gigabit Ethernet, активно продвигаемой Microsoft.

Глава 9

Построение отказоустойчивых систем

Увеличение объема хранилищ данных требует усиления мер по защите информации и уменьшения периодов ее недоступности. Это означает, что данные необходимо защищать от повреждения, одновременно стремясь избежать значительных потерь производительности. Массивы RAID представляют собой один из методов достижения баланса между надежностью и производительностью и подробно описываются в первой части главы.

Для обеспечения надежности и восстановления после сбоев данные хранятся в различных местах и постоянно обновляются, благодаря чему изменение в одной реплике данных отражается и на других репликах. Именно эта идея лежит в основе схем зеркального отражения и репликации, разработанных, в частности, для платформы Windows Server.

На пути передачи данных от сервера к устройству хранения не должно быть единственной точки отказа, для чего понадобится технология группового ввода-вывода. Методы группового ввода-вывода для семейства Windows Server рассматриваются в разделе 9.3.

9.1 Массивы RAID

Аббревиатура *RAID* расшифровывается как *R*edundant *A*rray of *I*ndependent *D*isks — *избыточный массив независимых дисков*. Массивы RAID были разработаны в Калифорнийском университете и описаны в знаменитом документе *The Berkeley Paper* в 1988 году. Сейчас системы RAID предлагаются многими производителями. В 1993 году организация RAID Advisory Group сформировала программу RAID Conformance, которая позволяла производителям тестировать аппаратное обеспечение для прохождения сертификации RAID Advisory Board. После успешной сертификации производитель получал возможность использовать официальный логотип RAID. Разные версии RAID предоставляют различные уровни производительности и защиты данных, поэтому реализация выбирается в зависимости от конкретных требований.

Практически для всех реализаций массивов RAID общей является идея *чередования* (striping). Чередование состоит из определения базовой единицы ввода-вывода (обычно она имеет размер от 512 байт до 4 или 8 Мбайт)[1] и метода физического размещения этих единиц на различных дисках. Таким образом, формируются логические блоки для кластеров более высокого уровня. Первая единица может находиться на диске 1, вторая — на диске 2 и т.д.

Массив RAID может быть реализован аппаратно или программно:

- в программном обеспечении узла (сервера);
- в адаптере шины, подключенном к узлу (серверу);
- в устройстве хранения.

В некоторых случаях используется сразу несколько методов, например реализация массива RAID как для узла, так и для устройств хранения.

Диспетчер логических дисков (LDM) в Windows NT служит одним из примеров программной реализации массива RAID. Массив *RAID, реализованный на узле* (host-based RAID), иногда называется *программным RAID* и обладает преимуществами, описанными ниже.

- Стоимость аппаратного обеспечения ниже, так как устройствам хранения не требуется высокоэффективная аппаратная логика.
- Существует возможность использования дисков от различных поставщиков.
- При реализации системы, аналогичной диспетчеру томов, предоставленные службы виртуализации используются для преодоления ограничений аппаратного обеспечения; например, несколько небольших дисков могут комбинироваться для формирования большого логического тома.
- Программный массив RAID обладает гибкой конфигурацией; например, может обеспечивать зеркальное отражение с помощью двух независимых систем RAID, объединять несколько номеров LUN в один более крупный номер LUN или разделить один LUN на несколько меньших логических томов.

В то же время программному массиву RAID присущи определенные недостатки.

- Вычисление четности и других данных, необходимых для работы массива, требует значительных затрат времени центрального процессора. Для некоторых приложений Microsoft не рекомендуется использование

[1] Бывают и исключения. Например, суперкомпьютер в компании Thinking Machines использует единицы ввода-вывода объемом 16 байт в подсистеме хранения RAID 3.

программного массива RAID, который чрезмерно нагружает ресурсы процессора.

- Для каждой операции записи по шине ввода-вывода передается две операции записи, одна для данных, вторая для четности.
- При использовании программных массивов RAID два уровня абстракции могут значительно повысить сложность и накладные расходы.

  1. Файловая система добавляет уровень абстракции, осуществляя преобразование данных файлового ввода-вывода в данные блочного ввод-вывода на уровне тома.
  2. Диспетчер тома (внедряющий программный массив RAID) добавляет еще один уровень абстракции, выполняя преобразование блоков тома в дисковые блоки. В контексте распределенной файловой системы (DFS) это означает необходимость получения двухуровневых блокировок — физических или виртуальных (т.е. файл, открытый в эксклюзивном режиме, может обрабатываться как виртуально заблокированный).

Существует несколько типов реализаций массива RAID, которые рассматриваются в разделах 9.1.1–9.1.7.

9.1.1 Массив RAID 0

В массиве *RAID 0* данные расслаиваются на несколько жестких дисков. Таким образом, массив RAID 0 выигрывает в производительности, но не предоставляет избыточности или защиты данных.

Как показано на рис. 9.1, при использовании массива RAID 0 данные по очереди записываются на разные диски. Приложение выдает несколько запросов на запись. В данном случае предполагается, что приложение всегда выдает запрос на запись с фиксированным буфером и размер буфера совпадает с размером единицы записи в массиве RAID. Данные на рис. 9.1 распределены по нескольким дискам поочередно (каждое число в прямоугольнике представляет собой запрос на запись).

Массив RAID 0 особенно эффективен в аспекте производительности операций ввода-вывода, так как чтение и запись на разные диски может выполняться параллельно. Однако самый большой недостаток RAID 0 — это отсутствие устойчивости к ошибкам. Если окажется неисправным один диск в массиве RAID 0, данные будут утрачены и на других дисках, следуя правилу "потерял один — потеряли все".

Более того, чередование может повысить быстродействие, но при чрезмерном использовании или в неподходящий момент ситуация фактически

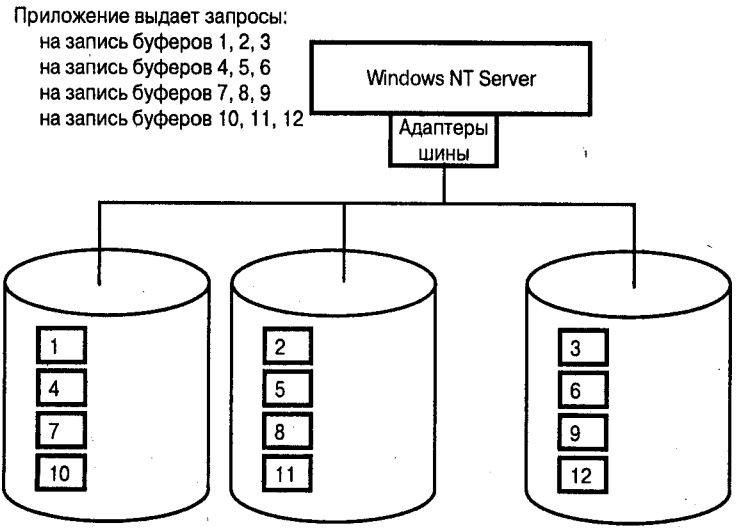


Рис. 9.1. Массив RAID 0

становится прямо противоположной и быстродействие снижается. Приложения, ориентированные на транзакции, обычно выполняют ввод-вывод небольшими блоками размером 1 Кбайт и меньше. Приложения, обрабатывающие потоčные данные, обычно осуществляют ввод-вывод гораздо бóльшими блоками. Несмотря на это, устройства хранения достигают наивысшей пропускной способности, когда ввод-вывод выполняется блоками намного большего размера, например до 512 Кбайт. Если в такой ситуации воспользоваться восьмикратным чередованием, то размер блока данных для каждого диска составит 64 Кбайт, что приведет к снижению общей производительности.

9.1.2 Массив RAID 1

В массиве *RAID 1* операция записи зеркально отражается на первичный и вторичный накопители. На рис. 9.2 показана конфигурация массива RAID 1. Приложение выдает запрос на запись буфера, обозначенного как *буфер 1*. Данные буфера 1 записываются на два отдельных физических диска. Точно так же приложение выдает запрос на запись буфера 2. Эти данные также записываются не на один, а на два жестких диска.

В идеальных условиях основной и вторичный диски могут быть идентичными. Это означает эффективное выполнение операций чтения в том случае, если они распределены между первичным и вторичным диском. Массив RAID 1 обеспечивает наилучшую скорость чтения среди всех массивов RAID. К его недостаткам относится требование к свободному дисковому простран-

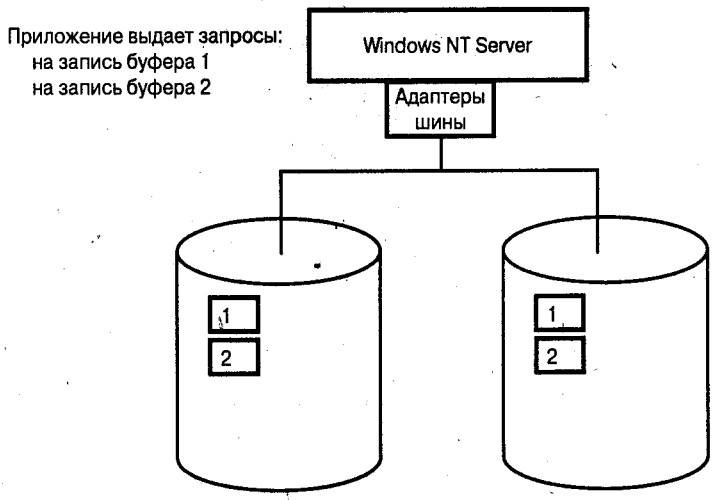


Рис. 9.2. Массив RAID 1

ству: для RAID 1 требуется, как минимум, вдвое больший объем дискового пространства, чем для других массивов. Снижение стоимости жестких дисков несколько уменьшает значимость этого недостатка. Существенное преимущество RAID 1 состоит в мгновенной доступности данных на вторичном диске в случае неисправности первичного диска. Массив RAID 1 — идеальный выбор при использовании в массиве только двух дисков.

9.1.3 Массив RAID 2

Массив *RAID 2* больше не используется. В массиве вычисляется код коррекции ошибок (error correction code — ECC) для чередующихся записей, однако современные жесткие диски и без того содержат подобную информацию.

9.1.4 Массив RAID 3

Массив *RAID 3* обеспечивает *чередование с выделенной четностью*. При этом информация о четности всегда хранится на определенном выделенном диске, а данные распределены по нескольким дискам.

На рис. 9.3 показана конфигурация массива RAID 3. Приложение выдает один запрос записи, который содержит буфера 1, 2 и 3. Эти данные размещаются на разных дисках. Кроме того, массив RAID 3 вычисляет информацию о четности (P1), охватывающую все три буфера. Информация о четности записывается на отдельный диск, отличный от дисков, на которые были записаны данные буферов 1, 2 и 3. Точно так же буфера данных 4, 5 и 6 записываются на различные диски, а информация о четности для этих буферов

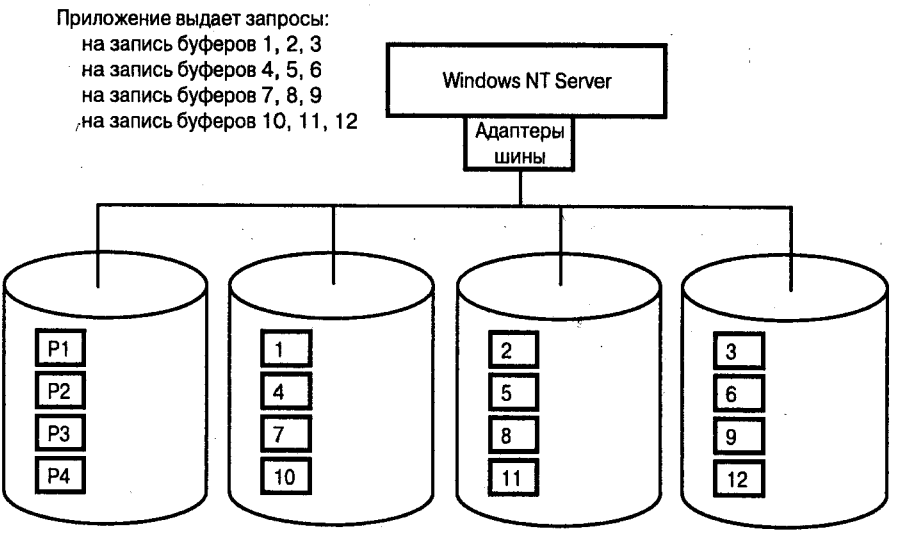


Рис. 9.3. Массив RAID 3

называется P2; буфера 7, 8 и 9 аналогичным образом записываются на разные диски, данные о четности этих буферов называются P3 и т.д. Обратите внимание, что информация о четности записывается на один и тот же диск.

Массив RAID 3 обычно требует участия всех дисков массива в операции чтения или записи. Если диск оказывается неисправным, в массив добавляется новый диск и данные с потерянного диска восстанавливаются на основе данных других дисков и информации о четности. Хотя массив RAID 3 имеет слабую поддержку в мире персональных компьютеров (например, не поддерживается контроллерами Adaptec), подобные ему часто применяются в высокопроизводительных суперкомпьютерах.

9.1.5 Массив RAID 4

Массив *RAID 4* напоминает массив RAID 3, но при чередовании используются бóльшие единицы записи. Операции чтения распределяются между несколькими дисками, а операции записи выполняются последовательно. Массив RAID 4 также практически не поддерживается производителями, например контроллерами Adaptec.

9.1.6 Массив RAID 5

Массив *RAID 5* обеспечивает *чередование с распределенной четностью* и представляет собой наиболее эффективную реализацию концепции чередо-

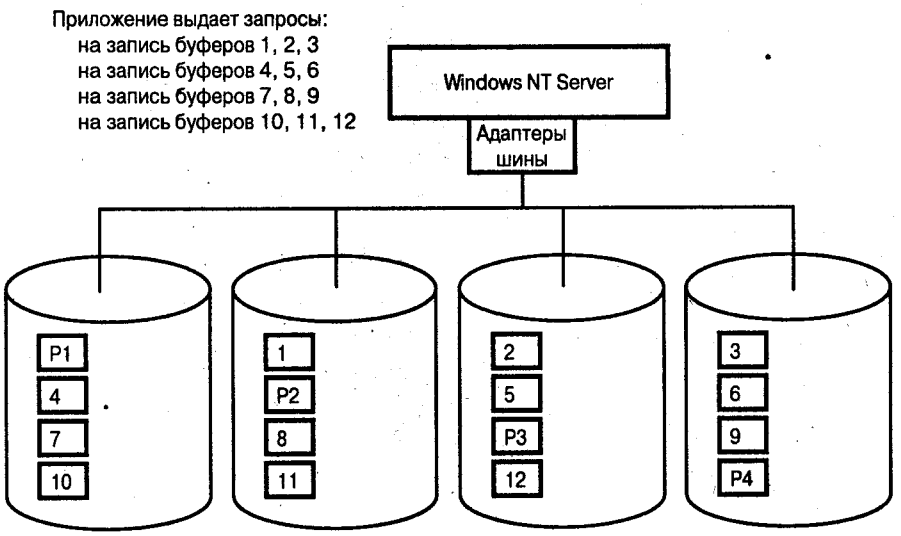


Рис. 9.4. Массив RAID 5

вания данных в массивах RAID. Единственное техническое отличие от массива RAID 3 состоит в хранении информации о четности на всех дисках массива.

На рис. 9.4 показана конфигурация массива RAID 5. Приложение выдает единственный запрос на запись данных из буферов 1, 2 и 3. Эти данные размещаются на разных дисках. Кроме того, массив RAID 5 вычисляет четность для всех трех буферов, и эта информация о четности (P1) записывается на диск, отличный от дисков, на которые были записаны данные из буферов 1, 2 и 3. Точно так же буфера 4. 5 и 6 записываются на разные диски, а информация о четности для этих буферов называется P2; буфера 7, 8 и 9 аналогичным образом записываются на разные диски, и данные о четности для этих буферов называются P3. Обратите внимание: в отличие от массива RAID 3, информация о четности распределена по всем дискам, а не хранится только на одном из них.

Массив RAID 5 обеспечивает баланс между производительностью и защитой данных. В массиве операции чтения могут обрабатываться параллельно несколькими дисками. Операции записи обычно требуют использования, как минимум, двух дисков: одного для данных и одного для четности. RAID 5 — наиболее популярный массив, требующий применения, как минимум, трех дисков. Кроме того, операция восстановления массива при отказе одного из дисков остается весьма трудоемкой.

9.1.7 Двухуровневый массив RAID

Двухуровневый массив RAID (Dual-level RAID) иногда называют *гибридным массивом RAID*. В двухуровневых схемах RAID различным образом комбинируются базовые реализации массива RAID с сохранением их преимуществ и сокращением влияния некоторых недостатков. В этом разделе вкратце рассматриваются массивы RAID 10, RAID 30 и RAID 50.

Как показано на рис. 9.5, массив *RAID 10* комбинирует возможности массивов RAID 0 и RAID 1. Данные записываются на два различных диска, каждый из которых зеркально отражается. Массив RAID 10 обеспечивает *чередование зеркальных массивов* на базе, как минимум, четырех дисков; при этом даже отказ в работе двух дисков не приведет к нарушению доступа к данным. Массив RAID 10 довольно популярен, так как позволяет реализовать избыточность данных и повышенное быстродействие при относительно несложной реализации.

Массив RAID 30 — это комбинация возможностей массивов RAID 0 и RAID 3 (рис. 9.6). При использовании массива RAID 30 данные записываются на несколько дисков (RAID 0) и выделенный диск четности. Этот массив обеспечивает *чередование с выделенной четностью*. Для реализации массива RAID 30 требуется, как минимум, шесть дисков, причем допускается отказ в работе до двух дисков (по одному диску на массив). Массив RAID 30

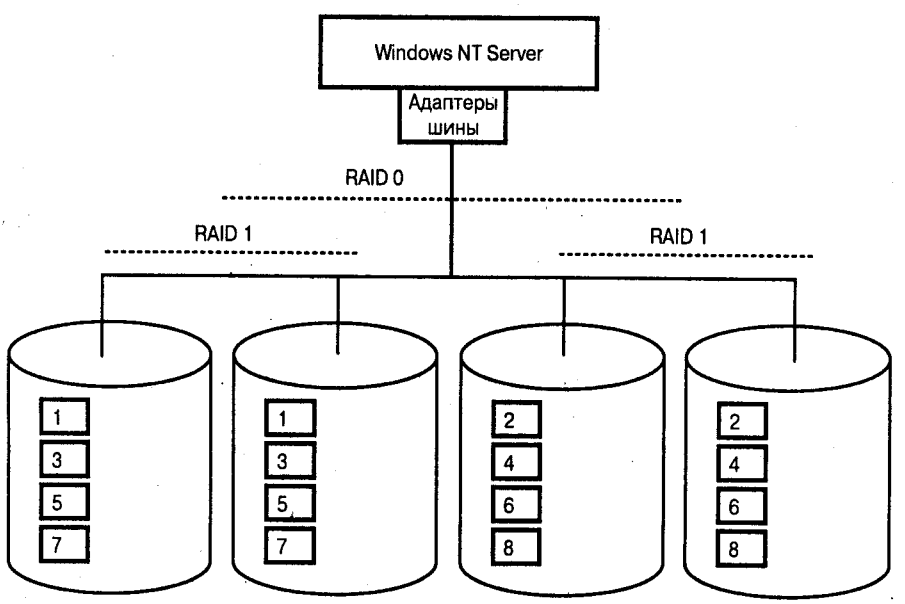


Рис. 9.5. Массив RAID 10

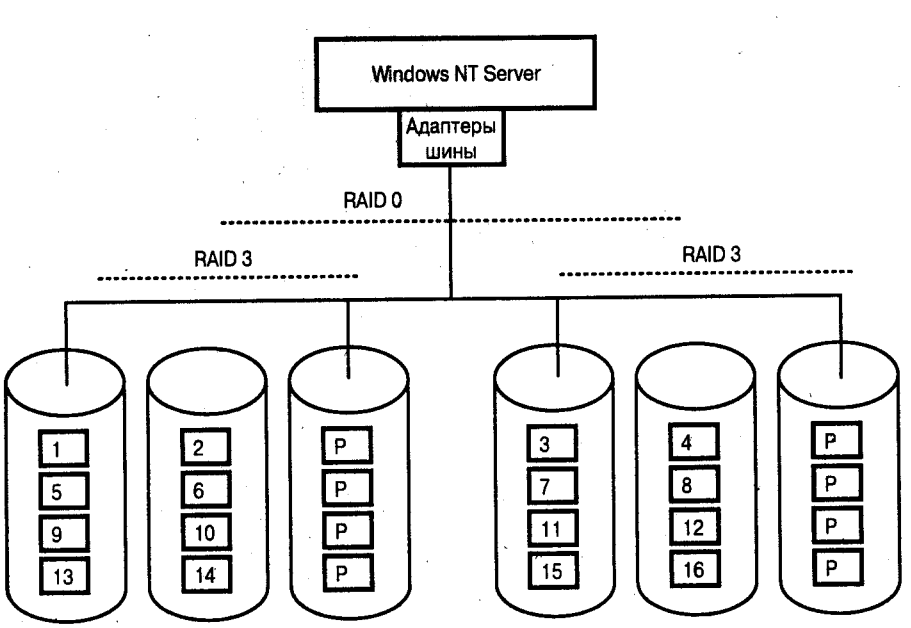


Рис. 9.6. Массив RAID 30

обычно используется в том случае, когда данные включают в себя большие файлы с последовательным доступом, например поточное видео.

Как показано на рис. 9.7, в массиве RAID 50 комбинируются возможности массивов RAID 0 и RAID 5. При использовании массива RAID 50 данные записываются на несколько дисков, как в массиве RAID 0, и на все диски поочередно записывается информация о четности, как в массиве RAID 5. Этот массив основан на *чередовании с невыделенной или распределенной четностью.* Для реализации RAID 50 требуется, как минимум, шесть дисков и возможен отказ в работе до двух дисков (по одному на каждый массив). В отличие от RAID 30, массив RAID 50 эффективен при работе с относительно небольшими файлами.

Некоторые контроллеры RAID поддерживают несколько идентификаторов SCSI. Кроме того, контроллеры RAID допускают использование нескольких LUN на одно устройство SCSI (так называемая *поддержка множественных номеров LUN*).

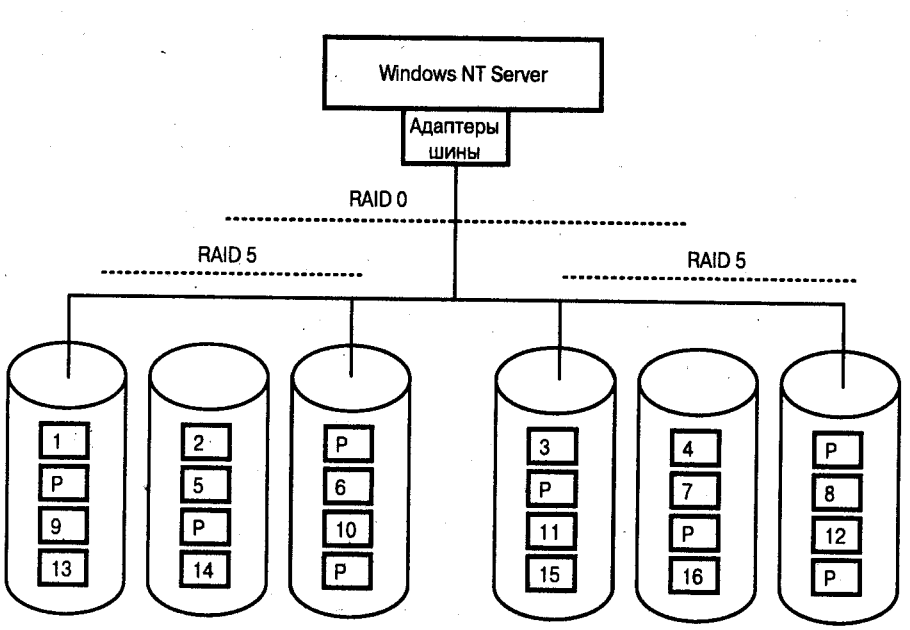


Рис. 9.7. Массив RAID 50

9.2 Реализация массива RAID на платформе Windows NT

Один из очевидных методов реализации массива RAID на платформе Windows NT состоит в поддержке массива на уровне аппаратного обеспечения, например в адаптере шины или контроллере устройства хранения.

В семействе Windows Server массив RAID поддерживают несколько драйверов, включая устойчивый к отказам драйвер FtDisk, драйвер LDM, который поставляется в Windows 2000, и драйвер LVM VERITAS, доступный для платформы Windows 2000. Все эти драйверы рассматривались в главах 1 и 6. Поддержка массивов RAID, обеспечиваемая этими драйверами, рассматривается в табл. 9.1.

Таблица 9.1. Массивы RAID, поддерживаемые различными диспетчерами томов Windows

Уровень RAID	FtDisk	Windows 2000 LDM	VERITAS LVM
RAID 0	Да	Да	Да
RAID 1	Да	Да	Да
RAID 5	Да	Да	Да
RAID 10	Нет	Нет	Да

9.3 Обеспечение избыточной отказоустойчивости

Одним из методов достижения отказоустойчивости является буквальное дублирование каждого компонента. Вместо одного сервера можно установить кластер, что позволит не потерять доступ к данным при отказе в работе одного из серверов.

На рис. 9.8 показан сервер с несколькими адаптерами шины. Каждый адаптер подключен к коммутатору, а каждый коммутатор снабжен двойным подключением к устройствам хранения с двумя портами. Единственной точкой отказа на рис. 9.8 является сервер. Как уже отмечалось, создание кластера позволит избавиться от этой точки отказа. Тем не менее в этой главе рассматривается конфигурация с единственным сервером. Основное внимание стоит обратить на два адаптера шины, установленные на сервере, — подобную архитектуру можно использовать и для кластеров. Простой установки двух адаптеров шины на компьютере под управлением Windows NT недостаточно. Необходимо использовать специальное программное обеспечение, которое рассматривается более подробно в разделах 9.3.1 и 9.3.2.

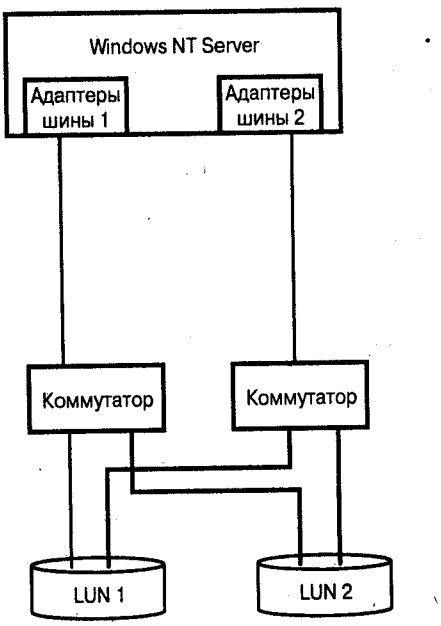


Рис. 9.8. Отказоустойчивая конфигурация

9.3.1 Поддержка группового ввода-вывода в Windows 2000 и Windows Server 2003

Компания Microsoft сообщила о поддержке технологий группового ввода-вывода, защиты целостности данных и балансировки нагрузки в Windows 2000 и Windows Server 2003. При этом предоставляется универсальная система, которую производители компьютеров и независимые поставщики аппаратного обеспечения должны настроить под конкретные особенности различных устройств. Производителю следует получить инструментарий разработки, который доступен при условии подписания договора о неразглашении. Конечный пользователь может получить готовую систему только от производителя, а не от Microsoft.

Еще раз обратите внимание на рис. 9.8, демонстрирующий схему сервера под управлением Windows NT с установленными двухпортовыми адаптерами шины. Не усложняя ситуации, предположим, что каждый диск отформатирован как единый том. Основная идея такой конфигурации — наличие нескольких путей ввода-вывода между жесткими дисками и сервером, что позволяет добиться устойчивости к отказам. Для конфигурации, приведенной на рис. 9.8, можно рассмотреть иерархию объектов устройств, которые создаются в стеке драйверов подсистемы хранения Windows.

На рис. 9.9 представлено дерево объектов для конфигурации рис. 9.8. Обратите внимание на пары PDO–FDO (объект физического устройства–объект функционального устройства), которые позволяют использовать возможности определенного устройства (см. главу 1). Вспомните, что объекты физического устройства, кроме всего прочего, предоставляют информацию, необходимую для использования устройства. Для устройств хранения эта информация может содержать идентификатор шины SCSI, идентификатор целевого устройства и LUN. Объект функционального устройства предоставляет сведения, необходимые для получения доступа к устройству. Для устройств хранения примером такой информации служат данные о системной организации диска.

В нижней части на рис. 9.9 показано, что подсистема PnP находит шину PCI, создает для нее объект физического устройства и загружает драйвер шины PCI, который создает объект функционального устройства для шины и подключает его к объекту физического устройства. Далее перечисляются устройства, подключенные к шине PCI. В результате обнаруживаются два адаптера шины. Драйвер шины PCI создает два объекта физического устройства, по одному для каждого адаптера шины. Подсистема PnP обнаруживает драйвер и загружает SCSIPort или Storport вместе с драйвером мини-порта, предоставленным производителем. Драйвер SCSIPort или Storport создает объект функционального устройства для каждого адаптера и подключает его к соответствующим объектам физического устройства.

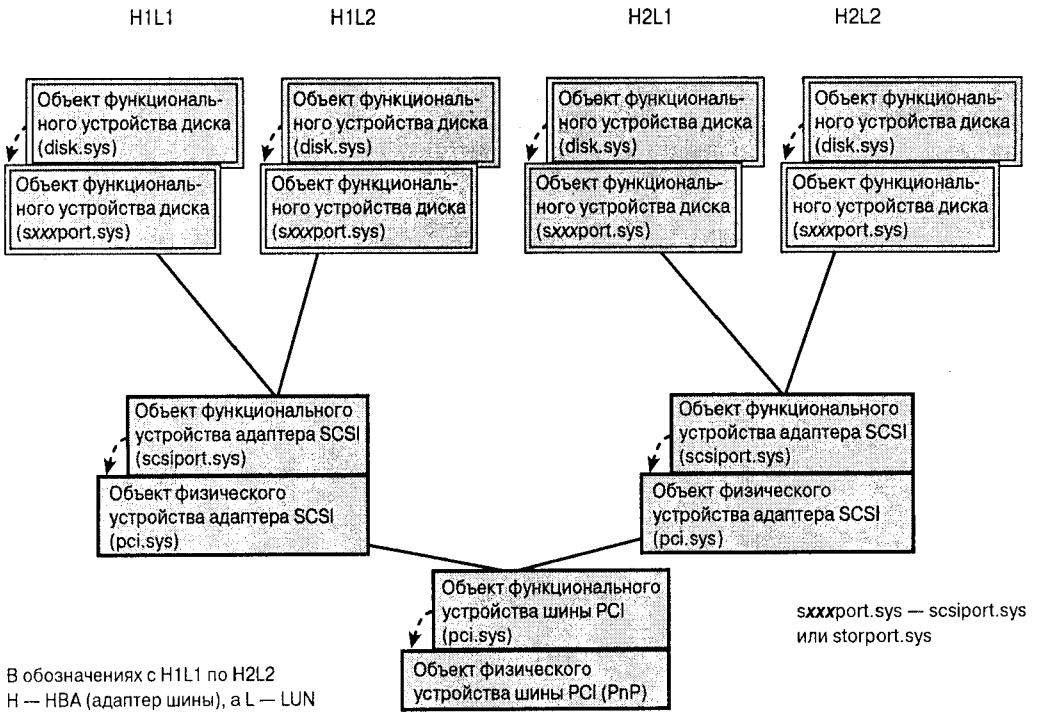


Рис. 9.9. Дерево объектов устройств без группового ввода-вывода

После этого драйвер SCSIPort или Storport начинает работать как драйвер шины и перечисляет устройства, подключенные к шине SCSI. Поскольку к шине подключено два устройства, сообщается о двух (дисковых) устройствах. Кроме того, так как к системе подключено два адаптера шины SCSI и перечисление выполняется для каждого из них, каждый адаптер сообщит о двух устройствах. Таким образом, драйвер SCSIPort или Storport в этом случае "увидит" четыре дисковых устройства. Создаются объекты физического устройства для четырех дисков и загружается драйвер класса диска, который создает четыре объекта функционального устройства и подключает каждый объект к соответствующему объекту физического устройства. Без группового ввода-вывода перечисляются разделы объектов функционального устройства диска и для управления томами, существующими на этих разделах, загружается драйвер FtDisk или LDM. (Чтобы не усложнять обсуждение, предположим, что каждый диск состоит из одного раздела и каждый раздел содержит один том.)

Программные системы более высокого уровня получат сведения о четырех томах устройств хранения, тогда как в действительности существует только два тома. Предположим, что эти два тома отформатированы для NTFS.

Файловая система также получит данные о четырех томах, для которых будет произведена попытка запуска NTFS. Очевидно, что NTFS, работающая на томах H1L1 и H2L1 (см. рис. 9.9), окажется несинхронизированной. При этом обязательно будут перезаписываться данные каждой файловой системы, например данные в журнале изменений, и в результате том окажется поврежденным.

Несколько производителей разработали метод, который не только решает описанную проблему, но и предоставляет богатый набор дополнительных возможностей, например сохранение и восстановление целостности данных, а также балансировка нагрузки. Функция *сохранения целостности* (failover) автоматически перенаправляет ввод-вывод с неисправного пути на другой путь ввода-вывода. В свою очередь, функция *восстановления целостности* (failback) исправляет неисправный путь ввода-вывода и снова вводит его в строй. *Балансировка нагрузки* позволяет распределить ввод-вывод на все доступные пути по определенному алгоритму. В качестве алгоритма может применяться поочередное распределение ввода-вывода, распределение на основе загрузки пути, простое распределение по всем путям или другой способ. Технология компании Microsoft рассматривается далее, после чего приводится описание продуктов от других производителей.

Компания Microsoft, создавая архитектуру группового ввода-вывода, преследовала несколько целей.

- Совместная работа с другими внедренными драйверами и архитектурами, включая PnP и управление питанием. На самом деле суть не просто в совместной работе, а в использовании уже существующей архитектуры, например когда уведомления устройств передаются с помощью базового механизма PnP.
- Динамическое обнаружение устройств и путей без применения статической конфигурации.
- Обеспечение совместного применения различных методов группового ввода-вывода от нескольких производителей. На данный момент это исключительно сложно (практически невозможно) реализовать.
- Предоставление универсальной технологии, которая позволяет производителям компьютеров и независимым производителям программного и аппаратного обеспечения добавлять такие возможности, как балансировка нагрузки или сохранение целостности данных. Тестовый модуль от Microsoft, связанный с определенным устройством (device-specific module — DSM), обеспечивает балансировку нагрузки, которая, впрочем, будет максимально эффективна при статическом использовании; например, для всего ввода-вывода на LUN 1 будет применяться первый путь, а для всего ввода-вывода на LUN 2 — второй путь.

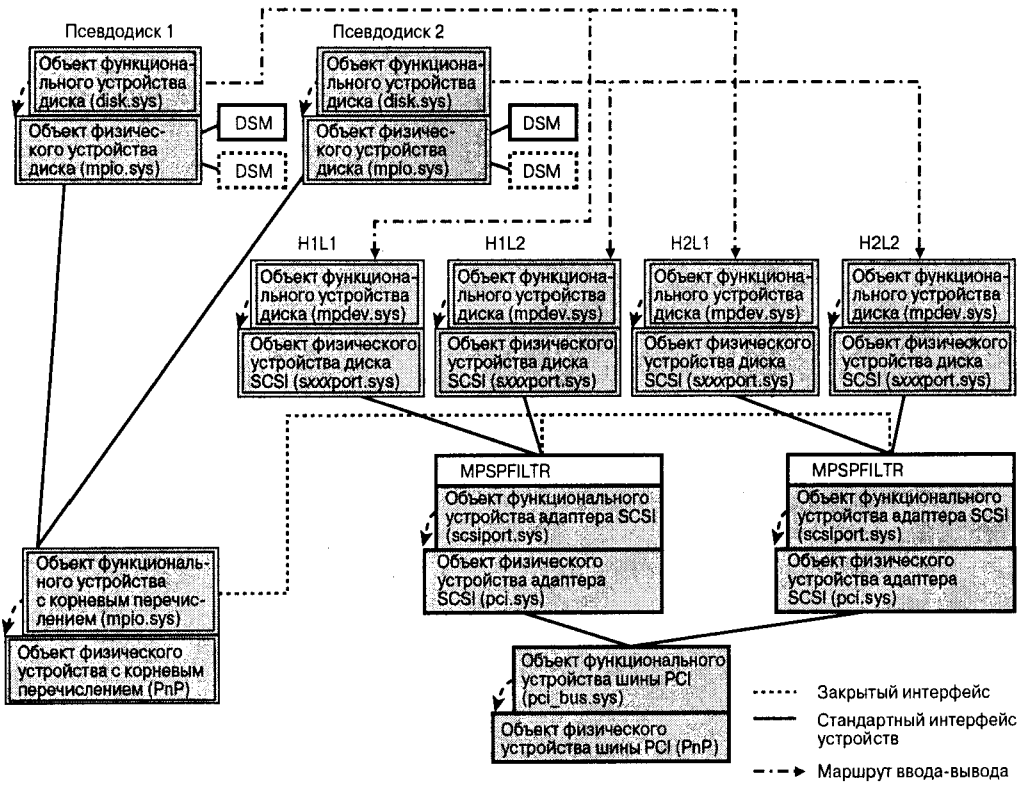


Рис. 9.10. Дерево объектов устройств для предоставления группового ввода-вывода

- Предоставление метода, который позволит использовать до 32 маршрутов на один номер LUN и поддерживает технологии Fibre Channel/SCSI.

На рис. 9.10 показано подробное дерево устройств Windows NT с поддержкой группового ввода-вывода для конфигурации, представленной на рис. 9.9. Дерево драйверов устройств включает в себя различные драйверы фильтрации и связанные с ними объекты устройств, которые вместе формируют архитектуру группового ввода-вывода Microsoft.

Архитектура включает в себя четыре различных компонента.

1. Драйвер фильтрации верхнего уровня, который называется *MPSPFLTR* и предоставляется Microsoft.
2. Драйвер класса *MPDEV*, предоставляемый Microsoft.
3. Драйвер псевдошины *MPIO*, предоставляемый Microsoft.
4. Модуль *DSM*, который должен предоставляться производителем, создающим и продающим систему. Этот производитель лицензирует инстру-

ментарий разработки MPIO у компании Microsoft. Инструментарий разработки уже содержит перечисленные три драйвера и предоставляет всю необходимую информацию (включая заголовочные файлы и пример кода) для создания DSM.

Первое, что бросается в глаза на рис. 9.10, это два различных стека устройств: логический (слева) и физический (справа). Программное обеспечение MPIO формирует мост между этими стеками устройств.

Любопытно отметить схожесть дерева устройств при использовании томов как базовых, так и динамических дисков (базовые и динамические диски рассматриваются в главе 6). Это неудивительно, так как тома являются логическими элементами, содержащими несколько LUN или фрагмент отдельного LUN, а инфраструктура MPIO стремится связывать видимые LUN через несколько путей с одним LUN. Возможности диспетчера разделов при обработке разделов весьма напоминают функции драйвера MPSPFLTR. Как первый, так и второй драйвер особое внимание уделяют пакету `IRP_MN_QUERY_DEVICE_RELATIONSHIPS` и передают подробную информацию об объектах соответствующим партнерам — диспетчеру томов в одном случае и драйверу псевдошины группового ввода-вывода MPIO — в другом. Диспетчер разделов и драйвер MPSPFLTR принимают ответственность за информирование партнеров (диспетчера томов и драйвера псевдошины MPIO) о событиях подсистем PnP и управления питанием.

Сравнивая рис. 9.9 и рис. 9.10, можно заметить, что MPIO являет собой драйвер фильтрации верхнего уровня, размещенный над объектом функционального устройства адаптера. Еще одно различие состоит в паре PDO–FDO, создаваемой для драйвера псевдошины MPIO подсистемой PnP и самим драйвером MPIO. Обратите внимание на закрытый канал взаимодействия между драйвером MPSPFLTR и драйвером псевдошины MPIO. Далее, в верхнем левом углу рис. 9.10, представлены два объекта физического устройства для псевдодисков, созданных драйвером шины MPIO. Таким образом, драйвер шины MPIO получает возможность обрабатывать ввод-вывод и, в свою очередь, вызывать DSM.

К каждому объекту физического устройства, созданному драйвером MPIO, подключены два объекта DSM. Один активно используется, а второй показан в другом прямоугольнике, чтобы подчеркнуть факт сосуществования объектов DSM от разных производителей. Обратившись к правой части рис. 9.10, можно заметить, что четыре объекта физического устройства создаются обычным образом драйвером порта (SCSIPort или Storport). Но подключаемые к ним объекты функционального устройства создаются драйвером класса MPDEV, а не драйвером класса диска.

Файл `Mpdev.sys` представляет собой замену драйвера класса диска с некоторыми изменениями. Драйвер класса диска MPDEV может обрабатывать

только запросы SCSI и не поддерживает функции пакетов IRP. Другими словами, драйвер MPDEV обрабатывает только ограниченное подмножество функций пакетов IRP, самой важной из которых является запрос `IRP_MJ_SCSI`. Драйвер MPDEV не поддерживает базовые функции пакетов IRP, например пакеты чтения и записи (`IRP_MJ_READ` и `IRP_MJ_WRITE`). Это означает, что приложения пользовательского режима не могут получить доступ непосредственно к стеку физических устройств, так как имеют возможность отправлять только запросы управления вводом-выводом (IOCTL). Конечно, драйверы режима ядра могут отправлять драйверу MPDEV блоки команд SCSI (CDB), чем и занимается драйвер класса диска.

Таким образом, драйвер MPDEV может обрабатывать запросы из стека MPIO (показанные штрих-пунктирной линией на рис. 9.10), так как эти запросы приходят от драйвера класса диска (расположенного над объектом физического устройства, созданного драйвером MPIO), преобразующего пакеты IRP (пакеты чтения/записи) в блоки команд SCSI. Более того, компания Microsoft создала строгие списки управления доступом для обеспечения безопасности объектов устройств, принадлежащих драйверу класса MPDEV.

9.3.1.1 Модуль DSM

Модуль DSM (Device-Specific Module) проектировался для предоставления важных функций, описанных далее.

- Обработка инициализации, относящейся к конкретному устройству.
- Предоставление функций, позволяющих выяснить, не являются ли на самом деле два LUN, к которым осуществляется доступ по разным путям, одним LUN. Для этого рекомендуется использовать встроенный идентификатор устройства хранения, а не программную метку носителя. Универсальный модуль DSM, предоставленный Microsoft, выполняет проверку с помощью страницы серийного номера (80h) или страницы идентификации устройства (83h), определенных в наборе команд SCSI. Поставщики устройств не ограничены использованием только этих двух механизмов.
- Обработка специальных команд SCSI, в основном связанных с управлением устройствами и опросом их возможностей, например `Read_Capacity`, `Reserve`, `Release` и `Start_Stop_Unit`, а также принятием решения об отправке этих команд по всем путям или только по одному.
- Принятие решений о маршрутизации запросов ввода-вывода.
- Обработка ошибок.
- Обработка запросов подсистемы PnP и подсистемы управления питанием с помощью библиотечных функций, реализованных Microsoft в драйвере псевдошины группового ввода-вывода.

- Обработка запросов управления, которые поступают к драйверу в виде пакетов IRP через интерфейс WMI, который рассматривался в главе 7. При получении такого запроса драйвер псевдошины группового ввода-вывода вызывает соответствующие процедуры DSM. Драйвер псевдошины группового ввода-вывода может самостоятельно находить и вызывать эти процедуры.

Модуль DSM внедряется с помощью инструментария MPIO, который можно лицензировать у компании Microsoft. Модуль DSM используется в качестве устаревшего (legacy) драйвера, который экспортирует интерфейс для драйвера псевдошины MPIO.

9.3.1.2 Драйвер псевдошины группового ввода-вывода

Драйвер псевдошины группового ввода-вывода загружается обычным образом, в качестве элемента Windows NT, как только будет установлен соответствующий программный пакет поставщика устройства.

При инициализации драйвер псевдошины группового ввода-вывода начинает взаимодействие с драйвером фильтрации MPSPFLTR, который размещен над объектом функционального устройства SCSIPort (см. рис. 9.10), что позволяет создать псевдоустройство для каждого логического устройства, которое подключено к нескольким путям. Для каждого такого псевдоустройства драйвер псевдошины группового ввода-вывода предлагает модулям DSM принять или отвергнуть право владения этим устройством.

Для каждого запроса ввода-вывода драйвер псевдошины группового ввода-вывода обращается к модулю DSM через определенную процедуру. Модуль DSM имеет доступ к каждому пакету IRP и может инициировать в случае необходимости процедуру завершения пакета IRP. Для запросов управления устройством, например `Reserve` или `Release`, модуль DSM может перенаправлять ввод-вывод по всем маршрутам к устройству. Обычные запросы ввода-вывода модуль DSM перенаправляет по любому из маршрутов ввода-вывода в зависимости от того, какая балансировка нагрузки проводится — динамическая или статическая. Если запрос ввода-вывода завершается ошибкой, драйвер псевдошины группового ввода-вывода в определенной точке входа вызывает модуль DSM, который может попытаться перенаправить ввод-вывод по другому маршруту, обеспечивая сохранение целостности данных.

9.3.2 Существующие системы группового ввода-вывода

Несколько компаний предоставляют системы группового ввода-вывода, которые, как минимум, обеспечивают сохранение, а некоторые даже восстановление целостности данных, а также балансировку нагрузки.

Эти системы довольно успешно работают, однако со временем стали очевидными некоторые их недостатки.

1. Конфигурация может быть сложной и запутанной, так как системы не полностью интегрированы с подсистемой PnP, поэтому динамический поиск не обеспечивается.
2. Системы не поддерживают взаимодействия с системами других производителей. Иными словами, если определенный сервер Windows работает с системой от одного производителя, на тот же сервер Windows невозможно установить систему других производителей.

9.3.2.1 Технология PowerPath от компании EMC

Компания EMC разработала технологию сохранения целостности данных и балансировки нагрузки для Windows NT. На рис. 9.11 показана соответствующая архитектура.

В отличие от других архитектур, в архитектуре EMC драйвер фильтрации размещается между диспетчером томов и драйвером класса порта SCSI-Port или RAID. Для каждого существующего логического тома перечисляется N логических томов, где N — число независимых маршрутов доступа к устройству.

Для каждого устройства, имеющего N независимых маршрутов доступа, Windows NT обнаружит N логических устройств. Если ввод-вывод будет осуществляться через все маршруты одновременно, данные могут быть повреждены. Таким образом, система PowerPath отключает N-1 таких устройств, доступ и управление вводом-выводом для которых будут невозможны. Утилита администрирования с графическим интерфейсом отображает одно активное устройство и N-1 устройств выделены серым цветом, которым отмечаются неактивные устройства. От администратора требуется серьезно продумать конфигурацию, особенно, если необходимо обеспечить безопасность путем ограничения адаптеров шины, имеющих доступ к устройству. Подобную схему безопасности можно внедрить с помощью системы EMC Symmetrix, посредством которой администратор укажет имя WWN адаптеров шины, имеющих доступ к определенным LUN на компьютере с установленной EMC Symmetrix.

Администратор имеет возможность указать политику балансировки нагрузки. Ниже приводится описание возможных политик.

- Запросы на ввод-вывод распределяются по всем маршрутам по очереди.
- Следующий запрос ввода-вывода отправляется по тому маршруту, на котором в очереди размещено меньшее количество запросов.
- Следующий запрос ввода-вывода отправляется по тому маршруту, на котором в очереди находится меньше блоков.

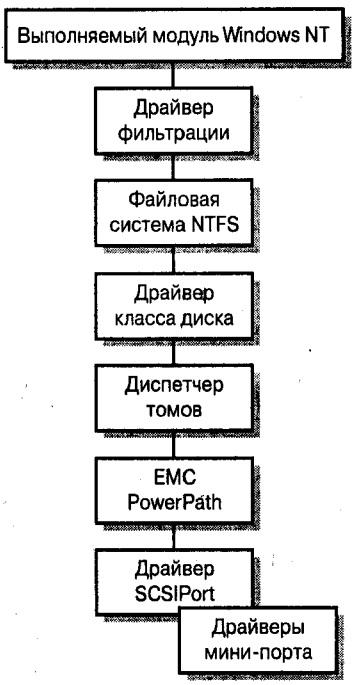


Рис. 9.11. Архитектура EMC PowerPath

- Используется режим оптимизации EMC Symmetrix, при котором следующий запрос ввода-вывода отправляется по маршруту с наименьшим ожидаемым временем завершения.

9.3.2.2 Технология SecurePath от компании HP (Compaq)

Компания HP (Compaq) предлагает систему группового ввода-вывода SecurePath для Windows NT, которая поддерживает сохранение целостности данных и балансировку нагрузки. Реализации этой системы-для Windows NT 4.0 и Windows 2000 несколько различаются.

На рис. 9.12 показана архитектура HP (Compaq) SecurePath для Windows 2000. Архитектура включает в себя драйвер фильтрации блочного устройства хранения, который расположен над драйвером порта (SCSIPort или Storport) и под драйвером класса диска. Служба пользовательского режима и приложения пользовательского режима формируют оставшиеся фрагменты головоломки, которые принимают участие в администрировании и отправке уведомлений.

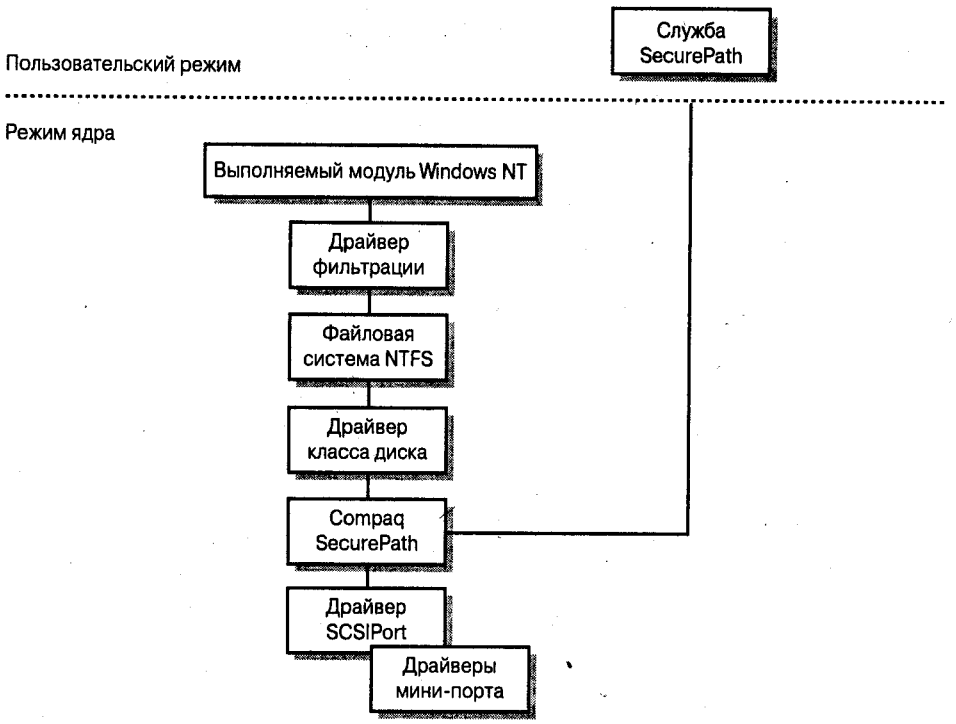


Рис. 9.12. Архитектура HP (Compaq) SecurePath для Windows 2000

В Windows NT 4.0 технология HP (Compaq) SecurePath требует использования драйвера класса диска, который создан компанией HP и называется HSZDisk (рис. 9.13). Кроме того, предоставляется и драйвер фильтрации.

В Windows 2000 сохранение целостности и балансировка нагрузки обеспечиваются драйвером фильтрации *Raidisk* от компании HP. В Windows 2000 драйвер класса диска, предоставленный компанией Microsoft, не заменяется другими драйверами. Драйвер Raidisk обеспечивает:

- сохранение целостности данных;
- балансировку нагрузки (для некластерного системного окружения);
- восстановление целостности после исправления отказавшей системы;
- проверку маршрута к томам хранилищ.

Служба пользовательского режима SecurePath для Windows NT предоставляет возможности по администрированию и взаимодействует с драйвером фильтрации SecurePath с помощью закрытых кодов управления вводом-выводом (IOCTL).

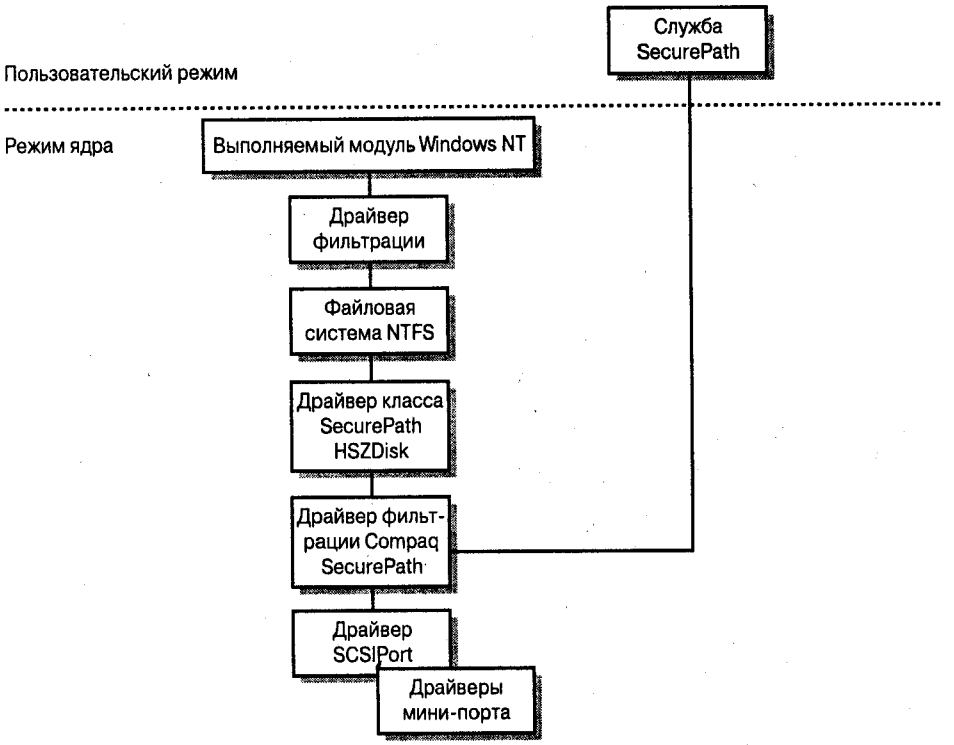


Рис. 9.13. Архитектура HP (Compaq) SecurePath для Windows NT 4.0

9.3.2.3 Технология AutoPath от компании HP

Эта технология обеспечивает динамическую балансировку нагрузки и автоматическое сохранение целостности данных ввода-вывода для Windows NT. Как показано на рис. 9.14, компания HP реализовала систему AutoPath с помощью драйвера фильтрации, размещенного между драйвером класса диска и драйвером порта.

Балансировка нагрузки AutoPath выполняется в соответствии с политикой, установленной администратором. Возможные политики перечислены ниже.

- Круговой доступ (round-robin), в котором данные ввода-вывода распределяются по всем маршрутам.
- Отсутствие балансировки нагрузки; при этом данные ввода-вывода для определенного устройства хранения статически отправляются по выбранному администратором маршруту.
- Данные ввода-вывода отправляются на маршрут, который имеет самую короткую очередь ожидающих запросов.

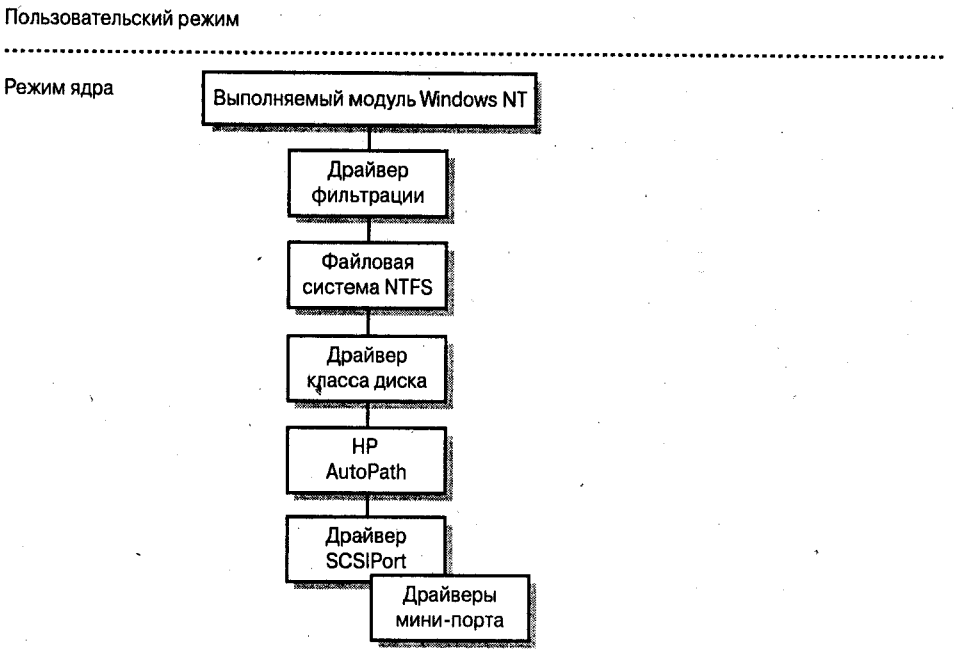


Рис. 9.14. Архитектура AutoPath

- Данные ввода-вывода отправляются на маршрут, который имеет наименьший объем данных, ожидающих ввода-вывода.
- Данные ввода-вывода отправляются на маршрут, который обеспечит наименьшее время обслуживания. Время рассчитывается в виде суммы всех запросов, находящихся в очереди. Данные ввода-вывода направляются по маршруту с наименьшей суммой.

9.4 Локальное и удаленное зеркальное отражение

Зеркальное отражение уже упоминалось в разделе 9.1, но подробно не рассматривалось. *Зеркальное отражение* представляет собой процесс создания дубликата доступных данных, что позволяет обеспечить доступность данных при отказе в работе основного хранилища.

К популярным причинам использования зеркального отражения относятся следующие:

- предоставление метода восстановления данных после различных неисправностей;
- распределение нагрузки или самих данных;
- обеспечение отказоустойчивости;

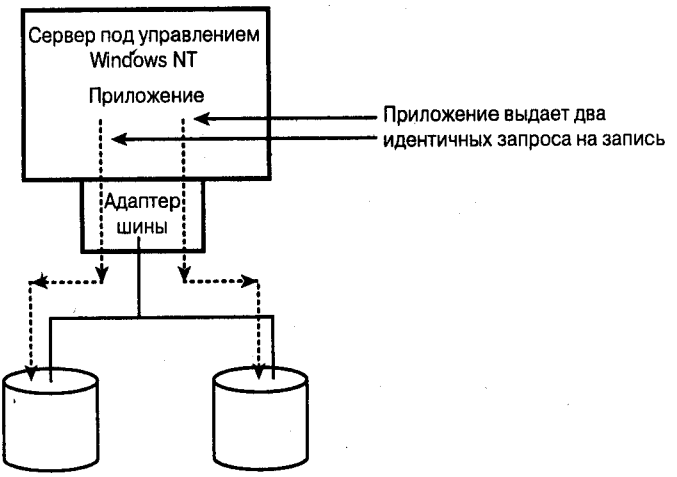


Рис. 9.15. Зеркальное отражение на уровне приложения

- использование вторичного тома для создания хранилища данных, резервного копирования или тестирования с помощью реальных данных.

Зеркальное отражение может быть локальным или удаленным. Локальное отражение выполняется на нескольких уровнях.

- *На уровне приложения серверного узла.* Приложение выдает два идентичных запроса на запись, предназначенные различным дискам (рис. 9.15).
- *На уровне программного массива RAID.* Приложение выдает единственный запрос на запись. Программный массив RAID, реализованный в драйвере диспетчера тома, преобразует единственный запрос в две идентичные операции записи (рис. 9.16).
- *На уровне адаптера шины.* Приложение выдает единственный запрос на запись, который преобразуется в два идентичных запроса на запись на уровне адаптера шины, установленного на сервере (рис. 9.17).
- *На уровне контроллера хранилища.* Такой тип зеркального отражения выполняется внутри подсистемы хранения (рис. 9.18).

Все эти методы относятся только к локальному отражению, имеющему ряд недостатков. Шина имеет ограничение на длину; например, устройства SCSI могут находиться друг от друга на расстоянии не далее нескольких десятков футов. Конечно, для разнесения устройств на большее расстояние можно воспользоваться SAN на базе Fibre Channel. С другой стороны, репликацию можно провести с помощью средств как источника, так и точки назначения данных (рис. 9.19).

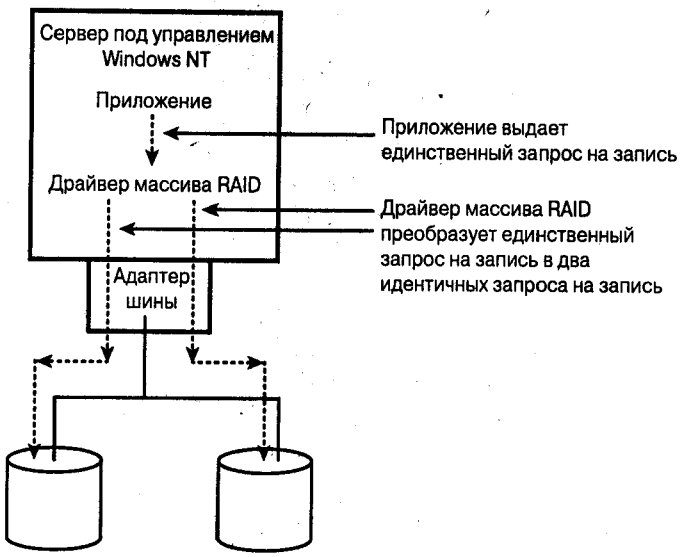


Рис. 9.16. Зеркальное отражение на уровне программного массива RAID

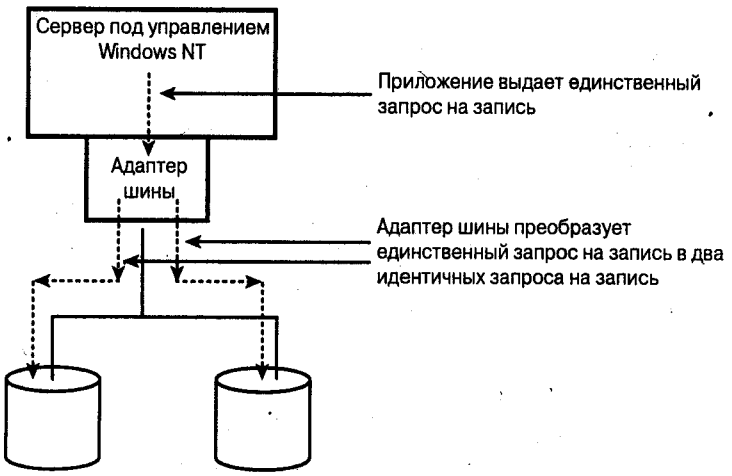


Рис. 9.17. Зеркальное отражение на уровне адаптера шины

Такой подход предоставляет весьма интересные возможности, например:

- выполнение синхронной или асинхронной репликации;
- выполнение двунаправленной репликации, повышающей эффективность и позволяющей серверам дублировать друг друга;
- восстановление после ошибок и тестирование контрольных точек.

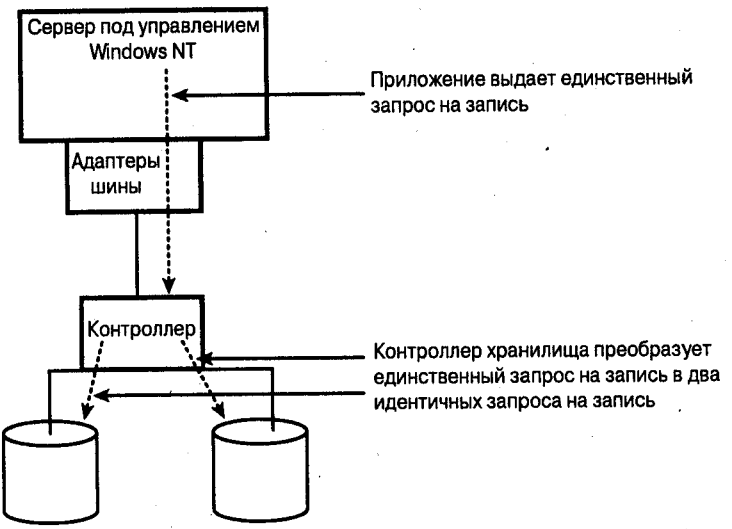


Рис. 9.18. Зеркальное отражение на уровне подсистемы хранения

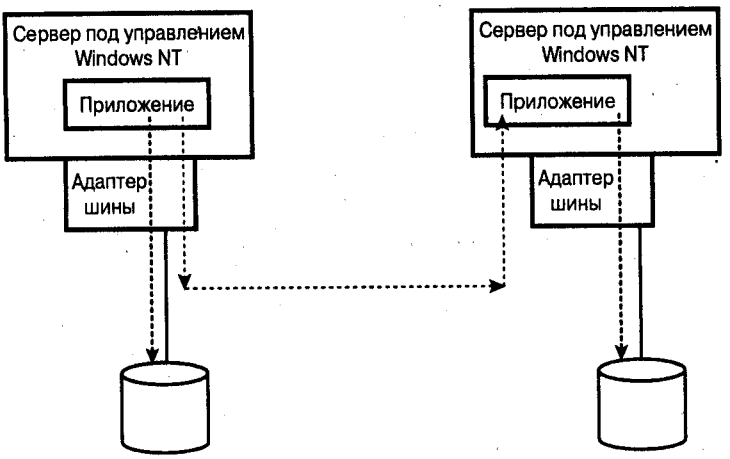


Рис. 9.19. Удаленное зеркальное отражение с помощью средств обеих точек репликации

Репликация может выполняться на уровне блоков дисков (другими словами, уровнем ниже файловой системы) или средствами драйвера фильтрации файловой системы, который может реплицировать файлы и каталоги.

Существует синхронная и асинхронная репликация. При *синхронной репликации* операция записи выполняется на обоих узлах (на источнике и вторичном узле, на который выполняется репликация), при этом от обоих серве-

ров получается подтверждение об успешной записи. Таким образом, страдает производительность приложений. При *асинхронной репликации* запрос записи на удаленный узел размещается в очереди, а запрос записи на локальный узел выдается сразу после возврата управления приложению, благодаря чему операции записи проводятся асинхронно.

Синхронная репликация выполняет синхронизацию данных за счет производительности приложения. Асинхронная репликация позволяет повысить производительность приложения за счет временной рассинхронизации данных на двух дисках. Как только данные поступают и записываются на удаленный диск, зеркальные диски вновь становятся синхронизированными. И синхронный и асинхронный метод репликации сохраняет порядок операций записи на устройство хранения.

Независимые производители программного обеспечения разработали несколько систем, обеспечивающих отказоустойчивость в среде Windows NT средствами репликации. Обратите внимание, что некоторые из этих систем автоматически выбирают синхронную или асинхронную репликацию. Работа начинается с синхронной репликации, но если на удаленной системе возникает ошибка, том отмечается как несинхронизированный и система переключается на асинхронную модель репликации.

В разделах 9.4.1–9.4.3 рассматриваются популярные программы репликации, доступные в продаже. Следует отметить, что представленные сведения не являются рекомендацией. В данном случае программы выбирались на основе двух характеристик.

1. Доступность сведений о программе.
2. Поддержка программами необходимых возможностей.

9.4.1 Программы Volume Replicator и Storage Replicator от компании VERITAS

Один из независимых производителей программного обеспечения — компания VERITAS — специализируется на хранилищах данных, их управлении и технологиях по восстановлению данных для разных платформ, включая Windows NT.

Программа Volume Replicator выполняет репликацию томов и работает с данными из блоков дисков. При этом между драйвером файловой системы и драйвером класса диска уже помещен драйвер фильтрации — VERITAS Volume Manager. Новый драйвер фильтрации подключается к этому драйверу диспетчера томов и перенаправляет запросы на удаленный том.

Программа Storage Replicator выполняет репликацию данных из файлов, а не из блоков дисков. Как показано на рис. 9.20, драйвер фильтрации файловой системы расположен над файловой системой. Программа Storage Repli-

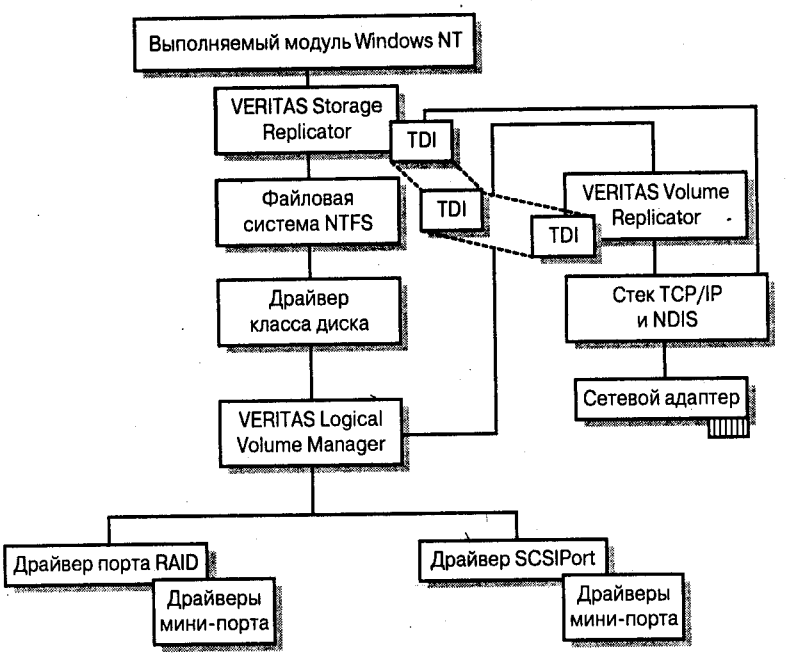


Рис. 9.20. Архитектура программ Volume Replicator и Storage Replicator компании VERITAS

cator сохраняет порядок записываемых данных. Операционная система Windows NT может находиться как на первичном, так и на вторичном узле, т.е. Windows может выступать в роли как источника (генерирующего данные для репликации), так и точки назначения (получающей данные репликации от другой системы, поддерживающей работу Storage Replicator). При внесении в конфигурацию определенных изменений для обеспечения корректной работы программы Storage Replicator необходимо перезапустить соответствующие системы службы. Для Volume Replicator перезагрузка не требуется. На рис. 9.20 показана архитектура продуктов Volume Replicator и Storage Replicator.

Как Volume Replicator, так и Storage Replicator используют интерфейсную библиотеку TDI (Transport Driver Interface) для отправки и приема данных. Программа Storage Replicator представляет собой драйвер фильтрации файловой системы, расположенный над самой файловой системой. В то же время Volume Replicator — это драйвер фильтрации диска (а не файловой системы). На рис. 9.20 демонстрируется, что TDI — подключаемая библиотека, а не отдельный драйвер.

9.4.2 Программа RepliStor от компании LEGATO

Программа RepliStor, разработанная компанией LEGATO (после приобретения компании Octopus), предназначена для синхронной репликации данных на уровне файлов и каталогов. Программа пытается копировать файлы (и синхронизировать их) для сохранения производительности и обеспечения отказоустойчивости. Кроме того, предоставляется возможность репликации разделов системного реестра.

Программа RepliStor поддерживает указание списков файлов, каталогов или общих ресурсов, а также исключений из этих списков, которые не будут подвергаться репликации. Кроме того, RepliStor поддерживает репликацию открытых файлов, если они были указаны до открытия другими приложениями. Также поддерживается репликация общих ресурсов распределенной файловой системы Windows NT 4.0, однако репликация общих ресурсов распределенной файловой системы Windows 2000 не обеспечивается. Версия RepliStor для Windows 2000 поддерживает интеграцию в Active Directory. После репликации файла RepliStor копирует только изменения, внесенные в файл.

Программа RepliStor может быть настроена на использование выделенного сетевого адаптера, что дает возможность изолировать сетевые данные RepliStor. Более того, передаваемые по сети данные шифруются, а каждый сетевой пакет поддерживает цифровую подпись.

Описываемая программа поддерживает репликацию "точка–точка" или "точка–множество". Точкой назначения может быть другой сервер под управлением Windows NT (работающий с другим аппаратным обеспечением) или член кластера. Источником репликации также может быть член кластера. Программа RepliStor проводит репликацию асинхронно, т.е. запросы записи на удаленный узел размещаются в очереди, а запросы записи на локальный узел выполняются без подтверждения с удаленного узла.

Программа RepliStor дополняет другие системы компании LEGATO, например Co-StandbyServer, которая не в состоянии зеркально отражать системные диски (диски, с которых загружается Windows NT). Поскольку RepliStor обеспечивает репликацию на уровне файлов и каталогов, поддерживается копирование и системных дисков.

Кроме все прочего, предоставляется настраиваемый таймер, по которому вторичная система отправляет тестовые эхо-пакеты системе-источнику. Если доставка тестового пакета завершится неудачно, вторичная система сначала отправляет тестовый эхо-пакет по протоколу ICMP; это позволяет убедиться, что основная система действительно неисправна.

Если обращение к основной системе завершилось неудачно, RepliStor предоставляет следующие настраиваемые варианты действий:

- на вторичной системе запускаются необходимые службы;

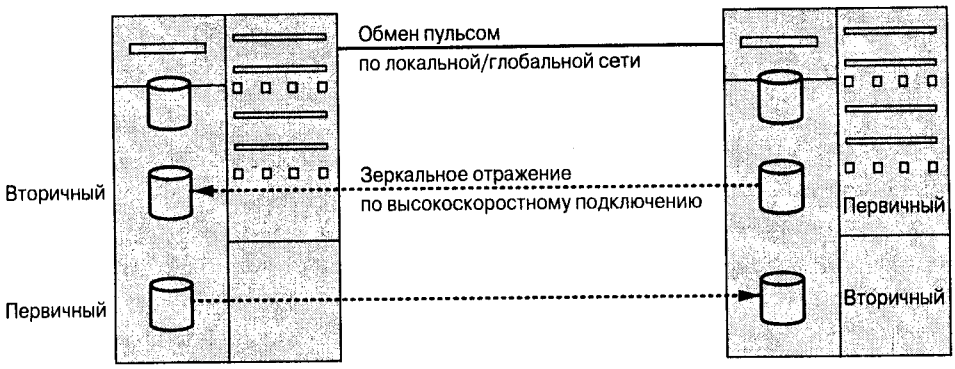


Рис. 9.21. Архитектура программы Co-StandbyServer

- параметры IP-адреса и маски подсети перенаправляются вторичной системе;
- на вторичной системе выполняются определенные команды, в том числе с помощью командных файлов.

9.4.3 Программа Co-StandbyServer

Программа Co-StandbyServer от LEGATO представляет собой кластерную систему для платформы Windows NT. Каждый сервер в кластере может быть активным, и изменения каждого диска реплицируются в обе стороны. Если один из серверов в кластере откажет в работе, Co-StandbyServer перенаправляет такие параметры, как IP-адреса, общие ресурсы, имена серверов и другие ресурсы, работающему серверу.

Как показано на рис. 9.21, Co-StandbyServer обеспечивает двунаправленную синхронную репликацию на уровне блоков между двумя серверами Windows NT, которые могут находиться на расстоянии до 10 км друг от друга. Обратите внимание, что оба сервера могут быть активными и взаимозаменяемыми. Таким образом, на рис. 9.21 показано, что во время репликации одного диска слева направо, второй диск реплицируется справа налево.

Два сервера, между которыми выполняется зеркальное отражение на уровне блоков, не должны быть идентичными. Единственное требование заключается в работе серверов под управлением Windows NT и использовании аппаратного обеспечения, входящего в список совместимого с Windows NT. Функция зеркального отражения не может быть реализована для загрузочного диска.

Существуют определенные проблемы при использовании Co-StandbyServer совместно с конфигурациями чередования на базе Windows 2000 LDM. Программа Co-StandbyServer предоставляет возможности двух типов.

1. Зеркальное отражение, при котором записи на один сервер дублируются на другой сервер.
2. Режим совместного хранилища, при котором отказ в работе хранилища одного сервера приводит к предоставлению хранилища второго сервера первому серверу, что позволяет ему продолжать работу.

В числе преимуществ Co-StandbyServer можно выделить следующие:

- не требуется специального аппаратного обеспечения, и оба сервера могут работать на разном аппаратном обеспечении; например, один сервер может быть многопроцессорным, а второй — однопроцессорным;
- программа Co-StandbyServer поддерживает различные серверы под управлением Windows 2000, включая контроллеры домена, рядовые серверы и т.д.;
- обеспечивается поддержка сценариев для реализации режима защиты целостности для популярных приложений Windows NT, включая IIS, SQL и Exchange.

9.5 Сложности практической реализации

Использование массивов RAID является весьма привлекательным решением, поскольку позволяет избежать значительных потерь производительности и обеспечить отказоустойчивость. Было создано несколько схем массивов RAID, которые предназначены для реализации различных задач. На базе платформы Windows NT возможно создание программных массивов RAID. В частности, Windows 2000 и Windows Server 2003 поставляются вместе с драйвером LDM, который обеспечивает работу программных RAID. Программные массивы RAID могут задействовать немало ресурсов центрального процессора, так как каждая операция ввода-вывода должна транслироваться дважды. Первая трансляция выполняется на уровне файловой системы, когда смещение в пределах файла преобразуется в смещение в пределах тома. Следующее преобразование осуществляется на уровне массива RAID (если необходимый ввод-вывод выполняется на этом массиве), где смещение в пределах тома транслируется в смещение в пределах диска.

Компания Microsoft предоставила производителям инструментарий разработки MPIO, который предназначен для создания отказоустойчивых высокопроизводительных систем для группового ввода-вывода. Готовые системы предоставляются клиентам только производителями, а не самой Microsoft. Инструментарий MPIO позволяет системам нескольких производителей совместно работать на одном сервере под управлением Windows, что ранее было невозможно. В настоящий момент существует тенденция к снижению цен в этом секторе рынка. Более того, системы на основе инструментария MPIO

позволяют потребителям не зависеть от одного производителя, так как возможно совместное применение систем нескольких поставщиков. Стоит проследить, не введет ли Microsoft аналог сертификации на соответствие логотипу, а также обратить внимание на строгость требований для такой сертификации.

Несколько производителей также создали схемы программного зеркального отражения и репликации. Рассматривая возможности этого программного обеспечения, не стоит забывать о новой архитектуре теневого копирования томов, которая рассматривается в главе 5. Иногда теневое копирование томов более подходит для решения задач зеркального отражения данных.

Кроме того, этот сегмент рынка внимательно наблюдает за развитием технологии IP Storage, так как весьма вероятно, что несколько производителей создадут соответствующие системы на базе протокола iSCSI.

9.6 Резюме

В главе описано несколько схем программных массивов RAID, отличающихся по стоимости, производительности и доступности. В Windows NT 4.0 для реализации программных массивов RAID использовался драйвер FtDisk. В свою очередь, в Windows 2000 для реализации программных массивов RAID применяется драйвер LDM, поставляемый в составе самой операционной системы.

Архитектура группового ввода-вывода от Microsoft предоставляет стандартизированный подход для реализации отказоустойчивых и высокопроизводительных систем для платформ Windows 2000 и Windows Server 2003. Эта архитектура поддерживает баланс между нагрузкой и сохранением целостности маршрутов ввода-вывода. Компания Microsoft предоставляет инструментарий для разработки законченных систем, лицензии на который передаются другим производителям. Конечные пользователи могут получить готовые системы только от независимых производителей, а не от самой Microsoft.

Разработкой программ зеркального отражения и репликации для платформы Windows NT занимается сразу несколько известных производителей.

Глава 10

Возможности подсистемы хранения данных в различных версиях Windows NT

В предыдущих главах рассматривалась архитектура Windows в рамках определенных возможностей подсистемы хранения данных. Эта глава предназначена для профессионалов, использующих подсистемы хранения в повседневной работе и желающих получить информацию об их функциях в разных версиях Windows.

Кроме того, в здесь приводятся некоторые прогнозы, которые могут оправдаться в ближайшем будущем. Предполагается, что обсуждаемые возможности будут включены в определенную версию Windows, однако нет никакой гарантии, что прогнозы совпадут с реальностью[1]. Читатель должен помнить об этом, когда будет принимать решения на основе информации, предоставленной в книге.

В этой главе обсуждаются возможности подсистемы хранения в контексте разных версий Windows, однако процесс выпуска новых версий делает такое обсуждение несколько запутанным. Зачастую ключевые функции разрабатываются для определенной версии Windows NT, однако после разработки становятся доступными и для более ранних версий. В таком случае функции рассматриваются в контексте именно новой версии Windows NT.

10.1 Windows NT 4.0

Операционная система Windows NT 4.0 используется в основном как файловый сервер и сервер печати, а также в качестве платформы для приложе-

[1]Компания Microsoft указывает на тот факт, что согласование функций, доступных в новой версии Windows, осуществляется на основе множества факторов, поэтому наличие в ней тех или иных возможностей не гарантируется. Оптимальный вариант заключается в изучении функций операционной системы только после выпуска ее окончательной версии.

ний. Кроме того, эта версия Windows обладает рядом возможностей, позволяющих работать в сетях хранения данных. Тем не менее по сравнению с Windows NT 4.0 в Windows 2000 были существенно улучшены подсистемы хранения данных.

Возможности Windows NT 4.0 и их взаимосвязь с подсистемой хранения рассматриваются в разделах 10.1.1–10.1.4.

10.1.1 Расширенные возможности доступа к единицам хранения

В Windows NT 4.0 Service Pack 5 стала доступна поддержка функции *Large LUNs*, позволяющей использовать до 255 LUN на одно устройство SCSI. До пакета обновлений Service Pack 5 в Windows NT поддерживалось только восемь LUN на одной целевое устройство.

10.1.2 Поддержка клиентов CIFS, NFS, NetWare и Macintosh

Операционная система Windows NT 4.0 отлично справлялась с управлением хранилищами, подключенными к сети, так как могла выступать в роли файлового сервера и сервера печати для различных клиентов. К ним относятся клиенты Windows, использующие протокол совместного доступа к сетевым ресурсам SMB или CIFS, клиенты UNIX, применяющие протокол NFS, клиенты NetWare и Macintosh.

10.1.3 Программные интерфейсы приложений для дефрагментации

В Windows NT 4.0 были предоставлены API для дефрагментации, которые позволяли независимым производителям программного обеспечения создавать приложения дефрагментации для современных и будущих версий Windows NT. Эти интерфейсы создавались Microsoft в тесном сотрудничестве с самими производителями, что позволило удовлетворить практически все требования последних.

10.1.4 Распределенная файловая система

Распределенная файловая система (DFS) позволяет администраторам создавать иерархию именования, отражающую корпоративную организационную структуру, благодаря чему пользователям не нужно отслеживать имена серверов и расположение файлов. Кроме того, распределенная файловая система поддерживает гетерогенную операционную среду, в которой могут участвовать серверы, работающие под управлением операционных систем,

отличных от Windows. Другими словами, DFS предоставляет возможность организации и управления сетевыми ресурсами, подобно тому как файловая система позволяет организовывать файлы и управлять ими.

Клиенты, работающие под управлением Windows, обращаются к распределенной файловой системе по протоколу CIFS, но как только клиент получает ссылку на сервер, где фактически хранится необходимый файл, он может подключаться к серверу по любому другому протоколу, например NFS или NCP (Network Control Protocol). Это означает, что единственный сервер под управлением Windows NT, исполняющий роль сервера DFS, может расширить преимущества распределенной файловой системы и на другие гетерогенные серверы. Распределенная файловая система использует функции службы репликации файлов. Данные о связывании физического расположения с логическим путем к файлу обрабатываются клиентом и сохраняются в системном реестре.

Распределенная файловая система обладает следующими преимуществами:

- обеспечение эффективного администрирования, так как ресурсы сервера могут быть перенесены без нарушения пользовательского доступа; пользователи могут получать доступ к ресурсу, перенаправленному администратором на другой сервер или на другой ресурс сервера;
- пользователям предоставляется уровень абстракции, в котором не требуется отслеживать физическое расположение необходимых файлов.

10.2 Windows 2000

Наиболее заметное улучшение Windows 2000 связано с возможностью использования платформы Windows NT в среде SAN и NAS. В разделах 10.2.1–10.2.13 рассматриваются улучшенные возможности подсистемы хранения в Windows 2000.

10.2.1 Расширенный доступ к единицам хранения

Операционная система Windows 2000 оптимизирует доступ к единицам хранения, а именно:

- значительно уменьшает количество ситуаций, в которых после изменения конфигурации единицы хранения требуется перезагрузка операционной системы, например при добавлении, удалении, увеличении или уменьшении томов;
- значительно увеличивает количество поддерживаемых единиц хранения.

Операционная система Windows NT 4.0 требовала перезагрузки перед получением доступа к новому устройству хранения (в терминологии подсистем хранения оно называется *целевым устройством*). Подсистема PnP Windows 2000 поддерживает динамическое добавление и удаление целевых устройств подсистемы хранения. Драйвер шины сообщает о событии, которое называется `BusChangeDetected`, а операционная система отвечает на это последовательностью действий, которая включает в себя повторное сканирование всех устройств, подключенных к шине. Тем не менее к новому устройству возможен доступ со стороны операционной системы и драйверов без перезагрузки. Кроме того, Windows не требует повторной инициализации шины подсистемы хранения (например, SCSI или Fibre Channel), что занимает немало времени. Таким образом, добавление нового целевого устройства или динамическое предоставление нового LUN для существующего целевого устройства не потребует повторной инициализации шины.

Однако Windows NT (включая Windows 2000) все еще принимает на себя владение всеми обнаруженными LUN. В настоящий момент Microsoft разрабатывает технологию, позволяющую администратору указывать доступные или недоступные LUN для Windows NT. Это даст возможность сделать LUN видимыми для Windows NT (и невидимыми для приложений), причем Windows не будет использовать эти единицы, что поможет избежать возможного повреждения данных LUN.

Операционная система Windows 2000 поддерживает значительно улучшенную схему подключения устройств хранения, которая позволяет адресовать до 128 целевых устройств (идентификаторов SCSI) и до 255 LUN на одно целевое устройств. Для сравнения: Windows NT 4.0 поддерживает только восемь целевых устройств и по восемь LUN на одно целевое устройство. Операционная система Windows NT 4.0 Service Pack 5 поддерживает 255 целевых устройств и восемь LUN на одно целевое устройство.

10.2.2 Новые механизмы управления томами и дисками

В Windows 2000 была представлена новая концепция *динамических дисков*. Термин *динамический диск* указывает на логическую архитектуру физического диска. Физический диск остается базовым диском с таблицей разделов, которая использовалась в Windows NT 4.0, пока не будет явно преобразован в динамический диск. Иными словами, динамические диски — это физические диски с другим форматом таблицы разделов, который допускает динамическое увеличение и уменьшение размера разделов. Новая таблица разделов с поддержкой избыточности обеспечивает доступность при повреждении кластеров основной копии. Более подробная информация по этой теме приводится в главе 6.

Динамические диски распознаются только в Windows 2000 и более поздних версиях Windows NT. Они хранят таблицу разделов старого формата, чтобы защитить данные от повреждения при просмотре раздела в Windows NT 4.0 или другой операционной системе, если она поддерживает только старый формат таблицы разделов. Однако такая таблица содержит лишь одну запись, указывающую на использование всего объема диска. Таблица разделов старого формата не отражает логической организации диска.

В Windows 2000 добавлены два новых компонента режима ядра. *Диспетчер монтирования* (Mount Manager) обрабатывает операции монтирования для базовых и динамических дисков, включая назначение букв дискам. *Диспетчер разделов* (Partition Manager) обрабатывает операции подсистемы PnP, а также операции управления питанием и взаимодействия с интерфейсом WMI, связанные с дисковыми разделами. Комбинация этих подсистем режима ядра позволяет использовать в одной системе одновременно два диспетчера томов. Одним из них является старый драйвер FtDisk, который в определенном смысле можно считать "классическим" диспетчером томов. Второй — это принятый по умолчанию диспетчер LDM (Logical Disk Manager), который поставляется в составе Windows 2000, или LVM (Logical Volume Manager), который предоставляется компанией VERITAS.

Наряду с улучшениями в подсистеме PnP управление дисками обрело ряд новых возможностей.

- Динамическое увеличение и уменьшение размера томов без останова и перезагрузки операционной системы. Приложениям не запрещен доступ к томам во время динамического изменения размера последних.
- Тома могут монтироваться и размонтироваться "на лету". Конечно, размонтированный том недоступен, однако системе не требуется перезагрузка после отключения или добавления тома.
- Тома могут быть перенастроены с одного класса RAID на другой класс без перезагрузки операционной системы. Допустима настройка, разбивка и повторная сборка массивов RAID 0, RAID 1 и RAID 5.
- Поддерживаются устаревшие тома, включая те, которые используют массив RAID с помощью старого диспетчера томов FtDisk.
- Динамические диски содержат всю необходимую информацию для описания структуры диска, т.е. все метаданные хранятся на самом диске. В Windows NT 4.0 часть метаданных хранилась в системном реестре. Это означает, что Windows 2000 допускает перенос динамических дисков между различными компьютерами без дополнительной настройки.
- Для управления дисками доступны утилиты как с графическим интерфейсом, так и поддерживающие командную строку.

- Добавлены новые API, которые позволяют создавать приложения управления томами. В качестве примера можно привести интерфейс для перечисления всех томов (`FindFirstVolume` для первого тома, `FindNextVolume` для следующего тома и `FindVolumeClose` для завершения перечисления и освобождения связанных с ним ресурсов). Функция `GetVolumeInformation` возвращает информацию о томе, включая поддержку томом точек повторной обработки (см. раздел 10.2.10).

10.2.3 Оптимизация распределенной файловой системы

Распределенная файловая система (DFS) впервые появилась в Windows NT 4.0. Она позволяет администратору эффективно управлять ресурсами сети и предоставляет пользователям возможность работать с названиями ресурсов и каталогов, которые отражают действительное рабочее окружение. При этом не требуется знать физическую структуру сети для доступа к данным. Например, пользователи компании могут получать доступ к ресурсу `\\Моя_Компания\Накладные` для загрузки всех файлов и каталогов, которые относятся к бухгалтерии, для чего им не придется задумываться об имени физического сервера и названиях его ресурсов.

В Windows 2000 распределенная файловая система была существенно модифицирована.

- *Повышенная надежность и отказоустойчивость.* В Windows 2000 предоставляется поддержка кластеров для DFS, что исключает отказ файловой системы при неисправности сервера, выполняющего роль корня файловой системы. Для этого DFS интегрирована в инфраструктуру Active Directory, поэтому данные топологии DFS хранятся в базе данных Active Directory. Более того, изменения топологии DFS вступают в силу без остановки и перезапуска службы DFS. Распределенная файловая система взаимодействует со службой репликации файлов Windows 2000 для дублирования изменений в репликах DFS.
- *Оптимизация администрирования.* Теперь для администрирования распределенной файловой системы предоставляется оснастка MMC. Кроме того, DFS поддерживает больше конфигурационных параметров, например допускается указание времени, в течение которого будет кэшироваться запись DFS. Распределенная файловая система в Windows 2000 интегрирована в службу каталогов Active Directory, благодаря чему системный администратор получает больше возможностей по управлению DFS.
- *Повышение производительности.* Клиенты под управлением Windows 2000 выбирают копии DFS, которые находятся в пределах локально-

го сайта, а не случайные копии, возможно находящиеся в удаленных сайтах. Если в локальном сайте доступно несколько копий DFS, клиенты Windows 2000 выбирают копии случайным образом. Клиенты под управлением Windows NT 4.0 выполняют значительно больше операций при доступе к DFS. Еще более старые клиенты DFS вообще о ней не "подозревают" и стремятся преобразовать путь, предполагая, что он содержит действительное имя сервера. Только после неудачной попытки подключиться к несуществующему серверу клиент использует службы драйвера `dfs.sys` для преобразования пути. Клиенты Windows 2000 изначально поддерживают использование пути DFS и задействуют драйвер `dfs.sys` для преобразования пути к файлу.

- *Балансировка нагрузки.* Так как несколько серверов зачастую содержат один и тот же набор файлов, клиент может получить доступ к любому из серверов, в то время как несколько клиентов имеют возможность получить доступ к одному и тому же файлу, хранящемуся на различных серверах.

10.2.4 Автономные папки и клиентское кэширование

Операционная система Windows 2000 позволяет администратору отметить общие ресурсы сервера как содержащие кэшируемые файлы. Файлы такого ресурса (например, данные и выполняемые файлы программ или динамически подключаемые библиотеки) загружаются на клиентские компьютеры и сохраняются локально. Такая загрузка выполняется при первом открытии файла. Неоткрываемые файлы на клиентские компьютеры не загружаются. Синхронизация файлов осуществляется в фоновом режиме либо при регистрации или завершении сеанса пользователя. Если синхронизация приводит к конфликту (когда локальные изменения конфликтуют с изменениями на сервере), он решается при содействии пользователя. Основные преимущества кэшируемых файлов таковы:

- повышение производительности, так как чтение с локального диска выполняется намного быстрее, чем считывание файла по сети;
- увеличение отказоустойчивости, так как пользователь может продолжать работу на клиентском компьютере даже при отказе в работе сервера;
- возможность отключения от сети и работы в автономном режиме.

Администратор может установить следующие параметры автономных папок:

- ручное кэширование документов, когда администратор явно указывает файлы документов, которые могут кэшироваться пользователями;

- автоматическое кэширование документов, когда все файлы документов, открытых пользователем, автоматически кэшируются на локальном компьютере;
- автоматическое кэширование файлов программ, при котором кэшируются выполняемые файлы и файлы динамически подключаемых библиотек.

10.2.5 Служба репликации файлов

В Windows 2000 была представлена двунаправленная служба репликации файлов (FRS), использующая журнал изменений NTFS для обнаружения измененных файлов и каталогов. Служба репликации файлов активно задействуется Active Directory. Кроме того, FRS применяется распределенной файловой системой для создания нескольких копий файлов и метаданных на нескольких серверах, что позволяет обеспечивать балансировку нагрузки и отказоустойчивость.

10.2.6 Индексация содержимого файловой системы

В Windows 2000 Server предоставляется встроенная служба индексации и связанные с ней утилиты.

- Окно Найти файлы и папки (Find Files or Folders) предоставляет доступ к возможностям службы индексации.
- Служба индексации может индексировать содержимое файлов и атрибуты файлов, включая указанные пользователем.
- Служба индексации поддерживает индексацию автономных данных, которыми управляют службы удаленного хранилища (RSS).
- На томах с NTFS служба индексации использует журнал изменений для отслеживания изменений с момента последнего запуска.
- Служба индексации использует возможность массовой проверки списков управления доступом и возвращает только те файлы, которые разрешено просматривать пользователю, выполняющему поиск. Файлы, к которым пользователю доступ запрещен, служба индексации из результатов поиска удаляет.
- Служба индексации поддерживает тома с файловой системой FAT, однако снижение эффективности при этом неизбежно, так как в FAT не поддерживается журнал изменений.

10.2.7 Оптимизация установки и настройки

Операционная система Windows 2000 может создавать собственные тома NTFS. Ранее для этого требовалось создавать раздел FAT, который впоследствии преобразовывался в раздел NTFS. Обратите внимание на взаимозаменяемое использование терминов *раздел* и *том*. Как отмечается в главе 6, разделы и тома относятся к одной логической концепции с несколько разными особенностями. В частности, том, в отличие от раздела, поддерживает динамическое изменение размера.

10.2.8 Оптимизация файловой системы

С выходом Windows 2000 в файловую систему были внесены определенные улучшения. Хотя основные модификации относятся к NTFS, Windows 2000 поддерживает расширенные возможности файловых систем для оптических носителей, а также для "ветерана" FAT. Компания Microsoft заявила, что в дальнейшем FAT будет лишь поддерживаться и новых модификаций не предвидится. Улучшения NTFS настолько существенны, что заслужили более подробного описания (см. раздел 10.2.9).

10.2.8.1 Оптимизация FAT

Операционная система Windows 2000 поддерживает файловую систему FAT32. Это первая версия Windows NT, работающая с FAT32, поскольку до этого FAT32 была реализована только на платформе Windows $9x$. В FAT32 поддерживаются тома размером до 2 Тбайт с максимальным размером файла, равным 4 Гбайт. Из соображений производительности, утилиты для работы с файловыми системами ограничивают размер тома FAT до 32 Гбайт, что, впрочем, относится к возможностям утилит, а не самой файловой системы. В FAT32 реализован меньший размер кластера диска, что позволяет более эффективно использовать дисковое пространство.

10.2.8.2 Файловые системы оптических носителей

Операционная система Windows NT 4.0 поддерживала файловую систему CDFS (CD-ROM File System), что также справедливо и для Windows 2000. В Windows 2000 появилась поддержка файловой системы UDF (Universal Disk Format), разработанной ассоциацией OSTA (Optical Storage Technology Association).

10.2.9 Оптимизация NTFS

Файловая система NTFS с выходом Windows 2000 претерпела значительные изменения, в том числе касающиеся дисковой структуры NTFS. Как только Windows 2000 в первый раз монтирует устаревший раздел с NTFS (*устаревший*, т.е. созданный средствами более ранней версии Windows NT), файловая система подвергается незаметному для пользователя преобразованию. Преобразование выполняется весьма эффективно, и время преобразования не зависит от размера раздела. Операционная система Windows NT 4.0 SP4 и более поздние версии Windows NT поддерживают новую версию NTFS.

Реализация этих возможностей потребовала изменения дисковой структуры NTFS. Для существующих томов дисковая структура меняется при первом монтировании в Windows 2000. Для системного тома эта операция выполняется при модернизации Windows NT 4.0 до Windows 2000.

Компания Microsoft провела немало тестирований Windows 2000 и NTFS в вычислительной среде корпоративного уровня предприятия.

- Максимальный размер файла, который использовался при тестировании, составляет 2^{44} – 64 Кбайт, что составляет примерно 16 Тбайт. С другой стороны, максимальный теоретический размер файла составляет 2^{64} – 1 байт, что примерно равно 17 Эбайт.
- Максимальное количество файлов на одном томе при тестировании составило 34 миллиона. С другой стороны, теоретический предел составляет 2^{32} – 1, т.е. примерно 4 миллиарда.

10.2.9.1 Базовые наборы свойств

Операционная система Windows 2000 впервые предоставляет поддержку базовых наборов свойств — определенных пользователем метаданных, связанных с файлом. Эти метаданные могут индексироваться службой индексации Windows 2000 Server. Например, можно указать автора документа или предполагаемую аудиторию документа. Затем пользователи могут осуществлять поиск документа, воспользовавшись значениями указанных тегов или метаданных. Файловая система NTFS обрабатывает файлы в качестве наборов пар "атрибут–значение". Определенные пользователем свойства сохраняются в дополнительных атрибутах файла.

10.2.9.2 Сканирование файлов определенного владельца

Операционная система Windows NT отслеживает учетные записи пользователей по уникальному идентификатору безопасности (SID). Файловая система NTFS в Windows 2000 может сканировать том и определять все файлы,

которые принадлежат определенному идентификатору безопасности. Этот процесс намного упрощает администрирование; например, при увольнении работника сканирование по идентификатору безопасности позволяет быстро обнаружить и удалить принадлежащие ему файлы.

10.2.9.3 Оптимизация проверки списков управления доступом

Файловая система NTFS в Windows NT 4.0 сохраняла списки управления доступом (ACL) отдельно для каждого файла и каталога. Если пользователю принадлежало 50 файлов, идентичные списки управления доступом потенциально могли сохраняться 50 раз, по одному разу в каждом файле. Файловая система NTFS в Windows 2000 сохраняет списки управления доступом в каталоге и проводит индексацию этих списков. Другими словами, в описанном случае список управления доступом будет сохранен один раз, а каждый из 50 файлов получит "указатель", который дает возможность определить необходимый список управления доступом.

Описанный метод позволяет обеспечить массовую проверку списков управления доступом службой индексирования. Если пользователь осуществляет поиск файлов, служба индексирования создает список файлов. Перед возвратом списка пользователю проводится проверка ACL и удаление из списка всех файлов, к которым пользователь не имеет доступа. Таким образом, пользователь получит список только тех файлов, на доступ к которым у него есть разрешение.

Массовая проверка списков управления доступом может применяться и в других случаях. Например, для определения действий, которые пользователь может выполнять с указанным набором файлов.

10.2.9.4 Файл журнала изменений

Файловая система NTFS отслеживает все вносимые изменения по двум причинам.

1. Для восстановления целостности данных в случае неисправностей в работе, что обеспечивается занесением в журнал информации обо всех операциях перед их выполнением.
2. Для предоставления приложениям информации об изменившихся файлах и каталогах.

В журнале изменений записываются только модификации, внесенные в файловую систему. Информации из журнала недостаточно для отмены выполненной операции. Файл журнала изменений "выживает" на протяжении нескольких перезагрузок операционной системы. Если этот файл окажется

переполненным или удаленным, приложения смогут определить, что некоторая информация об изменениях будет потеряна. Затем приложения проводят, например, повторное сканирование всех необходимых файлов и каталогов. Описанный механизм имеет особое значение для приложений репликации и резервного копирования.

10.2.9.5 Переименование потоков в NTFS

Файловая система NTFS в Windows NT всегда предоставлялась с поддержкой нескольких потоков данных на один файл. Примером приложения, поддерживающего многопоточный ввод-вывод, служит программный сервер Windows NT Macintosh. До появления Windows 2000 не существовало способа переименования потока данных после его создания. Можно было создать новый файл с новым именованным потоком данных и скопировать в него содержимое старого файла, однако такой подход недостаточно эффективен. Файловая система NTFS в Windows 2000 предоставляет специальный интерфейс, который позволяет приложениям переименовывать существующие именованные потоки данных.

10.2.9.6 Идентификаторы объектов и отслеживание ссылок

Операционная система Windows 2000 позволяет отслеживать ссылки. Это могут быть ярлыки файлов или объекты OLE, в частности документы Excel или PowerPoint, включенные в другой файл, например в документ Word. Приложение может отслеживать ссылки, даже если документ — источник ссылки перемещается различными способами, например:

- документ переносится в рамках того же тома на компьютере под управлением Windows NT;
- документ переносится между томами одного и того же компьютера под управлением Windows NT;
- документ переносится с одного компьютера под управлением Windows NT на другой компьютер под управлением Windows NT в рамках одного домена;
- весь том, содержащий документ, переносится с одного на другой компьютер под управлением Windows NT в рамках одного домена;
- переименовывается сервер под управлением Windows NT с монтированным томом, который содержит документ;
- переименовывается общий сетевой ресурс сервера под управлением Windows NT, который содержит документ;

- переименовывается документ;
- выполняется любая комбинация описанных действий.

Каждый файл в Windows 2000 (и более поздних версиях Windows NT) может иметь (но не обязательно) уникальный идентификатор объекта. Для отслеживания файла приложение ссылается на него по уникальному идентификатору. Если ссылка на файл не может быть найдена, на помощь вызывается служба отслеживания ссылок пользовательского режима (служба вызывается Windows NT). Служба стремится найти файл с помощью идентификатора объекта методом "проб и ошибок", обрабатывая все описанные выше сценарии.

10.2.9.7 Разреженные файлы NTFS

В Windows NT 3.51 была реализована функция сжатия файловой системы. Предполагалось, что будут сжиматься файлы, содержащие большие диапазоны, состоящие из одних нулей. Но сжатие и распаковка занимали определенное время, и сжатые данные все равно занимали некий объем жесткого диска. Рассмотрим файл, который имеет логический размер 1 Гбайт, но содержит только 4 Кбайт данных в начале файла и 4 Кбайт данных — в конце. В Windows 2000 такой файл, если его отметить в качестве разреженного, будет занимать всего 8 Кбайт дискового пространства (при соответствующем размере дискового кластера).

Файловая система NTFS упорядочивает внутренние структуры данных для распознавания отсутствующих блоков и при получении от приложения запроса на чтение данных из этих блоков и заполняет нулями буфер приложения. Приложение даже не "догадывается", что файл обладает определенными свойствами, и проводит с ним обычные операции чтения/записи. Разреженные файлы обладают атрибутом, управляемым пользователем, который может быть установлен для индикации разреженности файла. Приложения при этом работают обычным образом, однако система функционирует менее эффективно, так как определенное время затрачивается на возврат нулей для данных, которые находятся в невыделенных блоках файла. Приложения, обрабатывающие атрибут разреженного файла, могут пропускать операцию чтения невыделенных кластеров файла.

Функция перечисления файлов и каталогов (например, `FindFirst` и `FindNext`) возвращает флаг `FILE_ATTRIBUTE_SPARSE_FILE`. Функции Win32 API `BackupRead`, `BackupWrite`, `CopyFile` и `MoveFile` обновлены для работы с разреженными файлами (например, функция `CopyFile` сохраняет разреженное состояние файла и не выполняет операции чтения и записи, которые преобразуют файла в неразреженное состояние).

10.2.9.8 Дисковые квоты

Отслеживание и управление дисковыми квотами реализованы в NTFS, которая предоставляется в Windows 2000. Дисковые квоты отслеживаются для пользователей и для томов. При этом системный администратор может:

- отключить эту функцию;
- использовать ее для отслеживания объемов дискового пространства;
- задействовать ее для отслеживания и принудительного применения политик, которые ограничивают объем дискового пространства, выделенного пользователю.

Дисковые квоты отслеживаются на основе идентификатора безопасности пользователя (SID). Этот уникальный идентификатор назначается каждому пользователю, зарегистрированному на компьютере под управлением Windows NT (в качестве пользователя может выступать другой компьютер или системная учетная запись — по сути, тот же пользователь с набором привилегий, отличных от набора привилегий для стандартного пользователя). Каждый файл и каталог содержит идентификатор безопасности своего владельца. Дисковые квоты указываются для каждого тома, а утилиты администрирования позволяют реплицировать политику квот на другие тома.

Как уже неоднократно отмечалось, Windows 2000 позволяет возвращать список файлов, принадлежащих определенному пользователю. Эта функция предоставляет собой элемент технологии отслеживания дисковых квот, так как при отслеживании квот вычисляется общий объем дискового пространства, которое выделено определенному пользователю на хранение его файлов.

10.2.9.9 Оптимизация утилиты CHKDSK

В файловой системе NTFS для Windows 2000 уменьшено количество случаев, требующих запуска утилиты `CHKDSK`, что намного сокращает время работы данной утилиты. Будет уместным вспомнить выражение "конечный результат может отличаться от продемонстрированного", так как реальные результаты зависят от размера тома и типа повреждения. Для томов, содержащих миллионы файлов, утилита `CHKDSK` может работать быстрее на порядок.

10.2.9.10 Дефрагментация

В Windows 2000 предоставляется утилита дефрагментации, которая на самом деле является "облегченной" версией утилиты DiskKeeper от компании Executive Software. Как и в Windows NT 4.0, в Windows 2000 поддерживаются

специальные API дефрагментации. Встроенная утилита дефрагментации обладает определенными ограничениями, которые не свойственны большинству полноценных приложений дефрагментации, например:

- встроенная утилита не может дефрагментировать главную таблицу файлов NTFS (MFT) или файл подкачки;
- встроенная утилита не может дефрагментировать каталоги;
- встроенная утилита не поддерживает систему Microsoft Cluster Server.

10.2.10 Точки повторной обработки

Появление точек повторной обработки (reparse points) значительно усовершенствовало архитектуру файловой системы NTFS и подсистемы ввода-вывода Windows NT. Обратите внимание, что реализация точек повторной обработки потребовала изменения подсистемы ввода-вывода и NTFS. Возможности точек можно внедрить и в другие файловые системы, отличные от NTFS. Кроме того, точки повторной обработки обеспечивают работу следующих важнейших функций Windows:

- символьные ссылки;
- точки соединения каталогов;
- точки монтирования томов;
- служба SIS (Single Instance Storage);
- служба удаленного хранения HSM (Hierarchical Storage Management).

Все эти возможности рассматриваются в разделах 10.2.10.1–10.2.10.4.

Точки повторной обработки — это объект каталога или файла NTFS. Точка может быть создана, удалена или изменена приложением, использующим Win32 API в целом и функции `CreateFile`, `ReadFile`, `WriteFile` в частности. Вспомните, что Win32 API предоставляет приложениям возможность создавать и менять определенные пользователем атрибуты файла или каталога. Поэтому точки повторной обработки можно представить в виде определенного пользователем атрибута, который обрабатывается специальным образом. Такая обработка включает в себя обеспечение уникальности элементов объекта атрибута и обработку в подсистеме ввода-вывода. Независимые разработчики программного обеспечения обычно создают следующие приложения:

- утилиты пользовательского режима для создания, управления и удаления точек повторной обработки;
- драйверы фильтрации файловой системы, реализующие возможности точек повторной обработки.

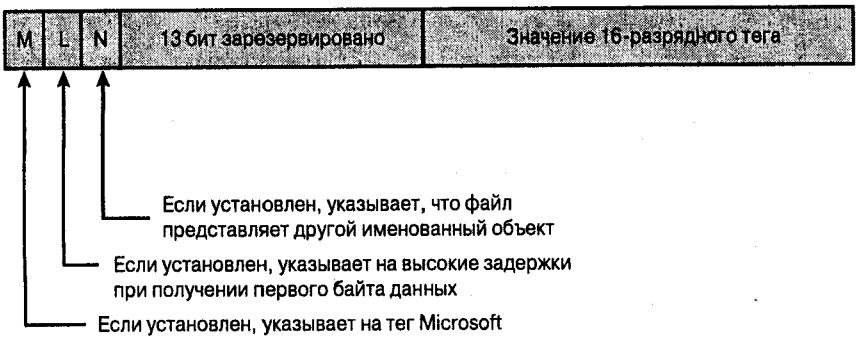


Рис. 10.1. Структура тега точки повторной обработки

Каждая точка повторной обработки состоит из тега и объекта данных. Тег — это уникальное 32-разрядное число, назначаемое непосредственно Microsoft. Независимые производители могут запросить назначение специального тега. На рис. 10.1 показана структура тега повторной обработки, включая следующие элементы:

- бит (M), который указывает, что тег принадлежит драйверу устройства от Microsoft;
- бит (L), который указывает на значительные задержки при получении драйвером первого байта данных; в качестве примера можно указать службу HSM, при использовании которой получение данных с неинтерактивного носителя может осуществляться достаточно долго;
- бит (N), который указывает, что файл или каталог представляют собой ссылку/перенаправление на другой файл или каталог;
- зарезервированные биты;
- фактическое 16-разрядное значение тега.

Объект данных точки повторной обработки может иметь размер до 16 Кбайт. Файловая система NTFS обеспечивает предоставление этого объекта данных драйверу устройства, созданному производителем. Объект данных предоставляется в процессе использования то́чки повторной обработки в подсистеме ввода-вывода.

На рис. 10.2 рассматривается последовательность рабочих операций и реализация точек повторной обработки. Предполагается, что пользователь обладает необходимыми привилегиями для выполнения запрошенной операции. Кроме того, на рис. 10.2 показан только драйвер фильтрации файловой системы, что позволяет сконцентрироваться исключительно на точках повторной обработки.

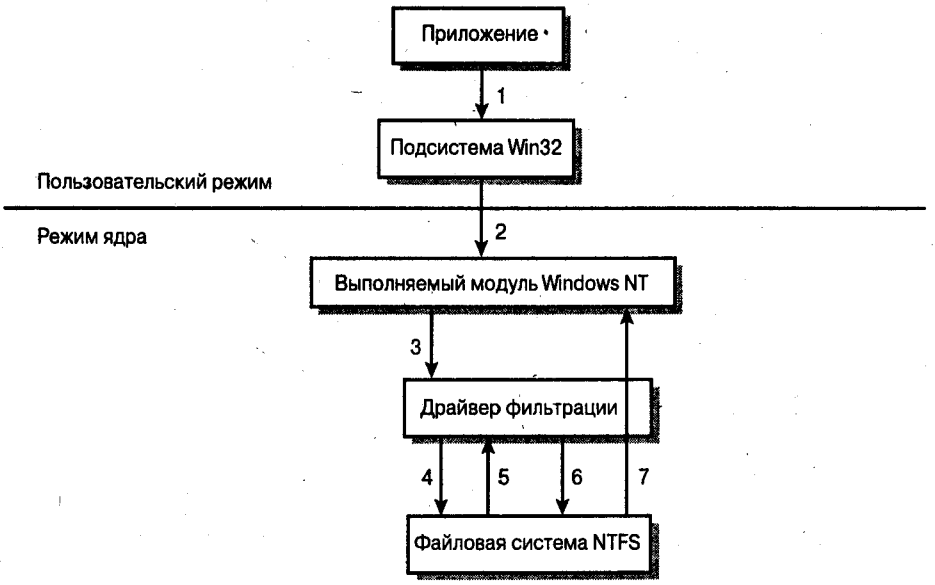


Рис. 10.2. Использование точки повторной обработки

Как показано на рис. 10.2, для реализации возможностей точки повторной обработки применяется несколько этапов.

1. Средствами подсистемы Win32 приложение отправляет запрос на открытие файла.
2. После проверок подсистема Win32 отправляет запрос выполняемому модулю Windows NT.
3. Диспетчер ввода-вывода Windows NT создает пакет IRP (`IRP_MJ_OPEN`) и отправляет его NTFS. Пакет IRP перехватывается драйвером точки повторной обработки в файловой системе.
4. Драйвер фильтрации перехватывает пакет IRP, определяет процедуру завершения, которая будет вызвана после завершения обработки пакета IRP, и средствами диспетчера ввода-вывода отправляет пакет IRP драйверу NTFS.
5. Пакет IRP достигает файловой системы, которая обрабатывает пакет `IRP_MJ_OPEN`, находит необходимый файл или каталог и отмечает ассоциированный с ними тег точки повторной обработки. Затем NTFS помещает тег и данные повторной обработки в пакет IRP и неудачно завершает обработку пакета IRP со специальным кодом ошибки.
6. Далее подсистема ввода-вывода вызывает каждый драйвер фильтрации (по одному за раз), который зарегистрировал процедуру завершения

для этого пакета IRP. Процедурой завершения каждого драйвера фильтрации рассматривается код ошибки и тег повторной обработки в пакете IRP. Если драйвер не распознает тег как собственный, он вызывает диспетчер ввода-вывода, который должен вызвать процедуру завершения следующего драйвера. Предположим, что один из драйверов распознал тег точки повторной обработки. После этого драйвер может использовать данные точки для повторной отправки пакета IRP, измененного в соответствии с данными, предоставленными точкой повторной обработки; например, перед повторной отправкой пакета IRP может быть изменен путь к файлу.

7. Файловая система NTFS завершает обработку повторно отправленного пакета IRP. Типичным примером может служить изменение пути и успешное завершение запроса. Диспетчер ввода-вывода завершает запрос на открытие файла, после чего через процедуры завершения может быть вызван каждый драйвер фильтрации файловой системы. Наконец обработка пакета IRP завершается, и приложение получает дескриптор файла.

Если ни один драйвер фильтрации не распознает тег точки повторной обработки, запрос на открытие файла или каталога завершается неудачно.

Одним приложениям требуются сведения о наличии точек повторной обработки, в то время как другим они совершенно не нужны. К последним, например, относятся приложения Microsoft Office, поэтому при открытии документа Word, PowerPoint или Excel функции точки повторной обработки, которая перенаправляет запрос на открытие на другой том, во внимание не принимаются. Тем не менее некоторым приложениям, которые рекурсивно просматривают дерево каталогов, требуются сведения о возможности создания зацикленных путей. Приложения могут аннулировать функции точек повторной обработки, передав соответствующие параметры (`FILE_OPEN_REPARSE_POINT`) в запросах `CreateFile`, `DeleteFile` и `RemoveDir`. Именно таким образом можно создавать, удалять и модифицировать точки повторной обработки. Функция `GetVolumeInformation` возвращает флаг `FILE_SUPPORTS_REPARSE_POINTS`. Функции `FindFirstFile` и `FindNextFile` возвращают флаг `FILE_ATTRIBUTE_REPARSE_POINT` для указания на наличие точки повторной обработки.

Все точки повторной обработки тома NTFS индексируются в файле, который называется `$Index` и находится в каталоге `\$Extend`. Таким образом, приложение может быстро обнаружить все точки повторной обработки, которые существуют на томе.

Следует подчеркнуть, что в этом разделе описываются точки повторной обработки, которые являются неотъемлемой частью NTFS. Хотя файловая система FAT не поддерживает точки повторной обработки, независимые про-

изводители программного обеспечения или компания Microsoft могут создать другую файловую систему, которая будет отличаться от NTFS и поддерживать точки повторной обработки. Это весьма нетривиальная задача, но стоит обратить внимание, что точки повторной обработки должны быть реализованы в трех областях.

1. Файловая система, например NTFS.
2. Подсистема ввода-вывода и набор функций Win32 API.
3. Утилиты и инструменты.

Очевидно, что Microsoft выполнила необходимый объем работ во всех трех областях и высока вероятность, что новая файловая система (в случае ее появления) также будет поддерживать точки повторной обработки.

В разделах 10.2.10.1–10.2.10.4 рассматриваются конкретные сферы применения точек повторной обработки.

10.2.10.1 Точки монтирования томов

В Windows NT 4.0 для монтирования томов требовалось использование буквы диска. Это ограничивало количество томов (всего можно было монтировать до 26 томов), которые могли использоваться в системе. В Windows 2000 разрешено монтирование тома без использования буквы диска. Но существуют и ограничения:

- том может монтироваться только на локальный каталог, т.е. том не может быть смонтирован на сетевой ресурс;
- том может быть смонтирован только на пустой каталог;
- пустой каталог должен размещаться в томе NTFS (на данный момент только NTFS поддерживает точки повторной обработки).

Приложения, которые получают доступ к каталогу, содержащему точку монтирования, не "замечают" ничего особенного, если только приложение специально не запрашивает информацию о каталоге.

Были добавлены и модифицированы API, которые предоставляют приложениям поддержку точек монтирования томов. Функция `GetVolumeInformation` с помощью флага показывает, поддерживает ли том точки монтирования. Функции `FindFirstVolumeMountPoint` и `FindNextVolumeMountPoint` используются для поиска точек монтирования томов. Функция `FindVolumeMountPointClose` применяется для освобождения ресурсов, задействованных функциями `FindFirstVolumeMountPoint` и `FindNextVolumeMountPoint`. И наконец, функция `GetVolumeNameForMountPoint` возвращает имя тома, в который преобразуется точка монтирования.

10.2.10.2 Точки соединения каталогов

Точки соединения каталогов тесно связаны с точками монтирования томов. Различие между ними заключается в том, что точки монтирования томов связывают каталог с новым томом, а точки соединения каталогов связывают каталог с каталогом на этом же локальном томе. Точки соединения каталогов могут создаваться с помощью утилиты `linkd.exe`, которая предоставляется в составе пакета Resource Kit.

10.2.10.3 Служба SIS

Служба SIS (Single Instance Storage) позволяет значительно снизить объем дискового пространства, которое требуется для хранения дублированных файлов. При этом дублированные файлы заменяются специальной ссылкой. Служба SIS используется совместно со службами удаленной установки (RIS), которые делают возможной загрузку Windows NT с сетевого ресурса.

Используя службы удаленной установки, администратор может создать несколько образов, например образ для бухгалтерии, конструкторского отдела и т.д. Каждый образ содержит множество файлов, дублирующих файлы из других образов. Но простого использования ссылки в данном случае будет недостаточно. Если администратор внесет изменения в один из файлов, все остальные образы, имеющие ссылку на этот файл, также будут содержать изменения. Именно в этом случае пригодится SIS.

Служба SIS вместо дублированного файла использует разреженный файл, для которого не выделено дисковое пространство и создана точка повторной обработки. Эта точка содержит достаточно информации для поиска необходимой резервной копии файла (если дублировано четыре файла, один файл остается как резервная копия, а остальные три файла превращаются в разреженные с точкой повторной обработки). Размер разреженного файла будет совпадать с размером оригинального файла. Когда приложение открывает разреженный файл, драйвер фильтрации перехватывает запрос на открытие файла, открывает резервную копию файла и передает дескриптор файла вызывающему приложению. После этого ввод-вывод осуществляется с помощью резервной копии файла.

10.2.10.4 Удаленное хранилище (HSM)

Удаленное хранилище (remote storage) — это служба управления хранилищем, которая предоставляет возможности прямого переноса файлов между интерактивным (например, жестким диском) и неинтерактивным (например, магнитной лентой) носителем. При переносе файла на неинтерактивный носитель на жестком диске остается так называемый "файл-заглушка". Благода-

ря точке повторной обработки NTFS файл отмечается, как особый, а также указывается точное местонахождение оригинального файла на неинтерактивном носителе. Службы удаленного хранилища (Remote Storage Services — RSS) зачастую называют службами управления иерархическим хранилищем (Hierarchical Storage Management), поскольку они взаимодействуют с двумя уровнями хранения данных.

Администратор может настроить специальные системные политики, например указать объем дискового пространства, который должен оставаться свободным, и типы файлов, которые должны обрабатываться RSS. Службы RSS периодически проверяют дисковое пространство на предмет соответствия установленной администратором политике. Кроме того, проводится поиск файлов, доступ к которым не осуществлялся некоторое время (обычно 30 дней). Эти файлы периодически копируются на магнитную ленту, причем их базовые версии остаются на жестком диске и с помощью точки повторной обработки отмечаются, как предварительно перемещенные. Как только дисковое пространство сокращается до указанного в политике объема, RSS удаляет файлы с диска и вносит изменения в точку повторной обработки, указывая, что файл теперь полностью перемещен на неинтерактивный носитель.

Службы RSS не поддерживают скрытые, системные, шифрованные или разреженные файлы, а также файлы, которые обладают расширенными атрибутами. Службы RSS тесно интегрированы с другими компонентами операционной системы.

- Службы RSS взаимодействуют со службой индексации.
- Службы RSS используют планировщик заданий Windows 2000 для планирования заданий по резервному копированию.
- Программа резервного копирования Windows распознает перемещенные на неинтерактивный носитель файлы и не пытается восстановить такие файлы на диск только для их резервного копирования.
- Службы RSS также интегрированы с системой безопасности NTFS; например, RSS вызывает файл с неинтерактивного носителя только в том случае, когда пользователю разрешен доступ к этому файлу.
- Программа Проводник (Windows Explorer) также взаимодействует с RSS. Перемещенные файлы отмечаются специальной пиктограммой.
- Журнал изменений NTFS взаимодействует с RSS, поэтому для выделения перемещенных между носителями файлов используется специальный флаг. Приложение может использовать этот флаг для определения того, что файл не изменился.

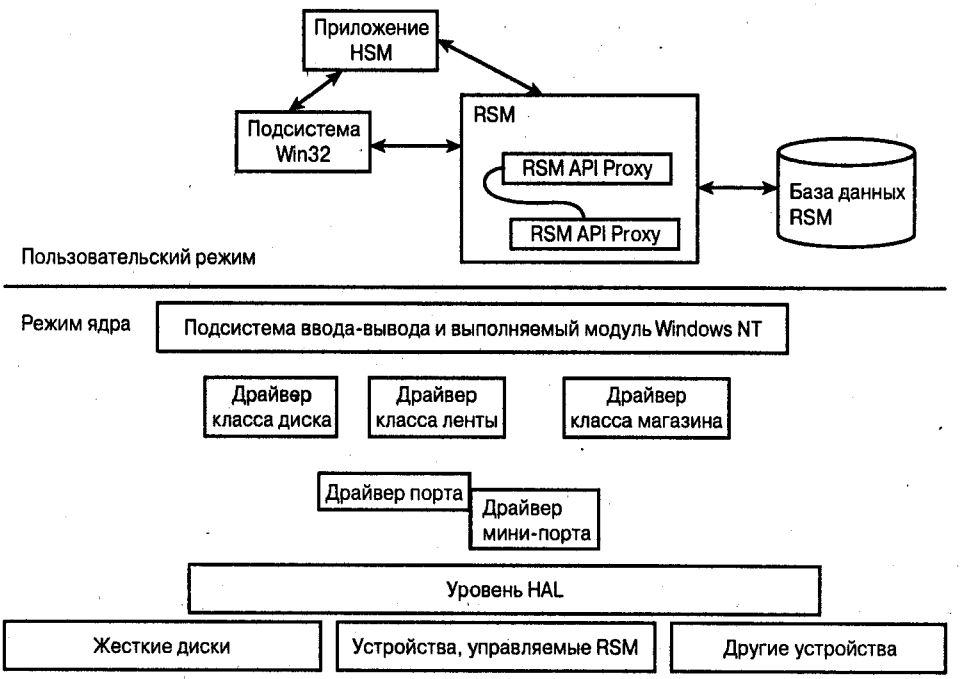


Рис. 10.3. Архитектура HSM и RSM

На рис. 10.3 показана схема RSS и архитектура управления сменными носителями (Removable Storage Management — RSM). Приложение HSM использует программный интерфейс приложений RSM (см. раздел 10.2.11). Служба RSM — это подсистема пользовательского режима, обладающая собственной базой данных для отслеживания носителей и устройств.

10.2.11 Управление сменными носителями

Служба управления сменными носителями (RSM) позволяет приложениям управлять всеми сменными носителями в корпоративной сети, вне зависимости от типа сменного носителя. Интерактивные носители обычно находятся в различных устройствах, включая магазины, ленточные автоматы и роботизированные библиотеки. Неинтерактивные носители, как правило, лежат на полке.

На рис. 10.4 показана архитектура и преимущества службы управления сменными носителями. Приложение работает с единственным интерфейсом, который предоставляется RSM. Сама RSM взаимодействует с интерфейсами новых и временно подключаемых устройств.

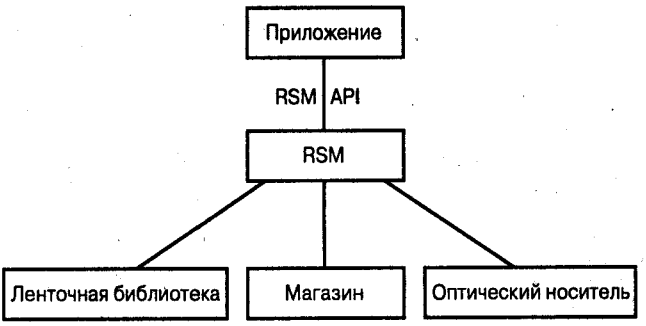


Рис. 10.4. Управление сменными носителями

В службе RSM используется концепция *пула носителей* (media pool) для организации и управления носителями. Пул носителей представляет собой набор носителей. Пул может быть двух типов: системный пул и пул приложений. Системный пул (system pool) содержит носители, предназначенные для системных задач, а также нераспознанные или неназначенные носители. Системный пул делится на следующие категории:

- *свободный пул* содержит носители, не используемые в данный момент и доступные для назначения;
- *пул импорта* содержит только что добавленные распознанные носители; носители в пуле импорта могут находиться под управлением приложений и переноситься в пул приложений, если приложению понадобится использовать данный носитель, или в свободный пул, если приложение определит, что носитель более не требуется;
- *нераспознанный пул* содержит новые носители, которые не были распознаны; как и в случае пула импорта, приложение может принять решение о перемещении носителя в пул приложения или в свободный пул.

Пул приложения содержит носители, созданные приложениями резервного копирования и удаленного хранилища.

10.2.12 Шифрованная файловая система

В Windows 2000 была впервые представлена шифрованная файловая система (Encrypting File System — EFS), которая на самом деле является службой, размещенной уровнем выше NTFS. Эта служба обеспечивает шифрование данных перед их записью на диск и дешифрование после чтения, но перед передачей данных запросившему приложению. Кроме того, EFS реализует систему управления ключами, необходимыми для шифрации, а также клю-

чами восстановления, которые позволяют авторизованному объекту (кроме владельца файла) восстанавливать данные из файлов.

Данные шифруются симметричным шифром (в частности, одной из разновидностей DES), и случайно сгенерированный ключ, использованный для симметричного шифрования, записывается в файл после шифрования асимметричным шифром с помощью открытого ключа пользователя. Для обеспечения возможности по восстановлению файла (например, когда работник увольняется и отказывается сотрудничать) случайно сгенерированный ключ симметричного шифра также шифруется открытым ключом авторизованного объекта и записывается в файл.

Шифрованные файлы не могут быть сжаты, т.е. файл может быть или сжат или зашифрован — третьего не дано. Шифрованная файловая система обеспечивает сохранение шифрования, даже если доступ к диску будет осуществляться в обход NTFS и ее механизмов безопасности. Шифрованная файловая система может использоваться для файлов или для каталогов.

Дополнительная информация о шифрованной файловой системе приводится в главе 6.

10.2.13 Защита системных файлов

Одна из проблем, долгое время мешавшая корректной работе Windows, была связана с повреждением и заменой системных файлов. Производители приложений обычно заменяли системные файлы Windows своими собственными или более новыми версиями. Такая практика в лучшем случае приводит к проблемам при управлении системой, а в худшем — к снижению ее надежности. Начиная с Windows 2000 компания Microsoft реализовала механизм защиты системных файлов (System File Protection — SFP).

Благодаря SFP определенные системные файлы защищены с помощью фонового механизма. Список контролируемых файлов сохраняется в папке кэша. Там же хранятся копии и других файлов. Если файл недоступен, он загружается с носителя (установочного компакт-диска Windows). Фоновый процесс вычисляет сигнатуру файла и сравнивает ее с сигнатурой файла, сохраненного в кэше. Если сигнатуры не совпадают, файл обновляется с помощью кэшированной копии. Утилиты установки пакетов обновлений и оперативных исправлений содержат специальный код, который позволяет обновлять как кэшированную, так и обычную копию файла.

10.3 Windows Server 2003

В Windows Server 2003 сохранены сильные стороны предыдущих версий Windows NT и добавлены собственные возможности, относящиеся к подсисте-

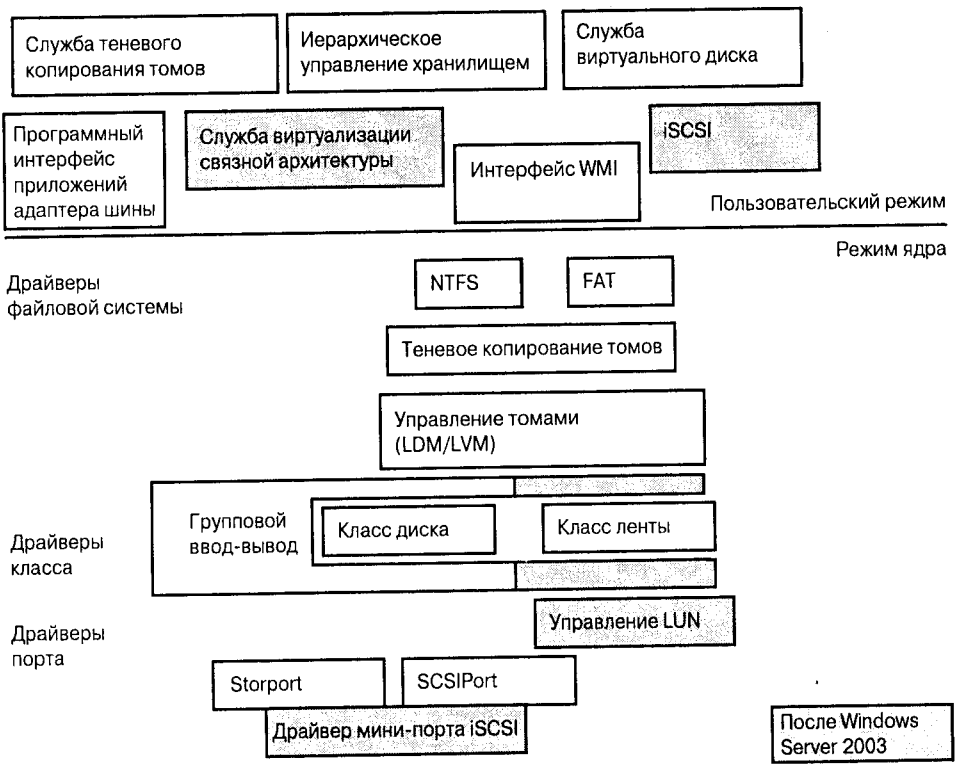


Рис. 10.5. Возможности подсистемы хранения данных в Windows Server 2003 и будущих версиях

ме хранения данных. Чтобы проиллюстрировать объем работы, выполненный разработчиками Microsoft, обратите внимание на рис. 10.5.

Незакрашенные прямоугольники на рис. 10.5 демонстрируют возможности подсистемы хранения Windows Server 2003, рассматриваемые в этой главе. Закрашенные прямоугольники показывают компоненты, которые с высокой долей вероятности появятся в будущих версиях Windows Server. Конкретная версия будущей операционной системы еще не известна. Судя по сложившимся традициям Microsoft, возможно несколько вариантов.

- Поставка вместе со следующей основной версией Windows NT (на данный момент она имеет кодовое наименование Microsoft Longhorn и будет выпущена в 2005 году).
- Поставка в составе пакета Windows Resource Kit или пакета обновлений (Service Pack).
- Поставка в виде обновления, доступного для загрузки с Web-узла Microsoft (например, через службу Windows Update).

- Поставка через независимых производителей аппаратного и программного обеспечения; например, Microsoft предоставит соответствующий программный пакет партнерам, которые используют его в собственных системах.

10.3.1 Модель драйверов Storport

Как отмечается в главе 1, Windows NT предоставляет уровень драйверов устройств, который позволяет достичь эффективного и масштабируемого ввода-вывода.

Дополнительная информация о стеке ввода-вывода Windows и всех его уровнях подробно описана в главе 1. Достаточно сказать, что одним из этих уровней является уровень драйверов порта. Драйвер порта отвечает за получение запросов ввода-вывода от верхних уровней, подготовку блоков команд SCSI и передачу запросов непосредственно устройству. Драйверы порта отвечают за хранение информации, которая позволяет осуществлять взаимодействие с устройством. Создавая блоки команд SCSI и обрабатывая результаты их выполнения, драйверы порта используют службы драйвера мини-порта, который создается производителем устройства.

До выхода Windows Server 2003 компания Microsoft предоставляла драйвер SCSIPort и оставляла производителям возможность создавать драйвер мини-порта SCSIPort, отвечающий за поддержку устройств Fibre Channel и SCSI. Отдельный установочный образ Windows NT мог содержать несколько драйверов мини-порта от различных производителей. Драйвер SCSIPort перенаправлял запросы соответствующему драйверу мини-порта.

При использовании этого подхода возникал ряд проблем.

- В описываемой модели предполагается, что устройства Fibre Channel обладают возможностями, аналогичными функциям устройств SCSI, однако это не соответствует действительности, как и то, что новые устройства SCSI-3 подобны более старым устройствам SCSI.
- Чтобы упростить задачу создания драйвера мини-порта, драйвер SCSIPort поддерживает однопоточную, а не дуплексную модель ввода-вывода. Это не позволяет системе достичь необходимой и возможной скорости обмена данными.
- Драйвер порта содержал определенную информацию, которая не передавалась драйверу мини-порта, что требовало от него получения этих данных с помощью нескольких вызовов. В частности, это относится к спискам сборки/разборки.
- Не желая мириться с ограничениями стандартной модели, производители устройств Fibre Channel создавали единый драйвер, в котором

объединялись функции драйверов порта и мини-порта, или заменяли драйвер порта собственным драйвером. Поскольку этот процесс требовал тщательного анализа функций драйвера порта, такие попытки в лучшем случае работали не совсем стабильно. Кроме того, ситуация еще более усложнялась, если возникала необходимость запустить драйвер мини-порта от одного производителя вместе с драйвером порта от другого производителя.

В Windows Server 2003 предоставлена новая модель драйверов Storport. Теперь от производителей адаптеров шины (HBA) ожидается создание драйвера мини-порта, который подключается к драйверу Storport, а не к драйверу SCSIPort. Чтобы минимизировать усилия по созданию новых драйверов, компания Microsoft сохранила обратную совместимость между моделями Storport и SCSIPort. Это позволяет производителям использовать некоторые (но не все) преимущества новой модели. Чтобы использовать все преимущества, производители должны заново переписать драйвер мини-порта, а не просто выполнить перекомпиляцию и подключить новую библиотеку.

Новая архитектура обладает перечисленными ниже преимуществами.

- Модель Storport обеспечивает более высокую производительность за счет полнодуплексного ввода-вывода. Недостаток этого подхода состоит в том, что драйверу мини-порта требуется обрабатывать процедуру выполнения. Таким образом, высокое быстродействие достигается за счет повышенной сложности всей модели.
- Модель Storport оптимизирует интерфейс между драйвером порта и драйвером мини-порта. Например, новый интерфейс позволяет драйверу мини-порта собирать информацию о списках сборки/разборки за один вызов, а не с помощью нескольких вызовов в цикле. *Списки сборки/разборки* — это универсальный термин, описывающий ситуацию, когда ввод-вывод выполняется в несколько отдельных (и несмежных) буферов одновременно.
- Модель Storport позволяет интерфейсу соответствовать запросам производителей высокоуровневых подсистем хранения, в частности производителей устройств RAID и Fibre Channel. Например, старая модель драйверов SCSIPort предоставляла слишком мало возможностей по управлению очередью, в то время как новым устройствам требуются более развитые механизмы управления. Модель Storport поддерживает до 254 запросов к одной логической единице адаптера. Максимальное количество запросов к адаптеру ограничивается только количеством логических единиц, подключенных к этому адаптеру.
- Модель Storport поддерживает многоуровневую иерархию сброса для восстановления после ошибок. В старой модели при появлении ошиб-

ки выполнялось обнуление данных шины (что нарушало работу других устройств и приложений). В новой модели проводится минимально возможное обнуление (сначала LUN, затем целевое устройство и лишь в качестве последнего средства — обнуление шины).

- Новая модель поддерживает отдельный расширенный интерфейс управления.
- Новая модель предоставляет интерфейс, который не требует создания "фантомного" устройства. Модель SCSIPort не позволяла приложениям опрашивать возможности устройства без его монтирования. Поэтому производители создали фантомное устройство для ответа на определенные запросы. Модель Storport не требует создания фантомного устройства, поддерживая опрос даже без монтирования и подключения объекта мини-порта.
- Все эти возможности предоставляются максимально незаметно благодаря обратной совместимости с моделью драйверов SCSIPort. Производители могут просто перекомпилировать существующий драйвер и подключить новую библиотеку. Это позволит работать с новой моделью драйверов, но не даст использовать полный спектр возможностей модели Storport. Устаревшие устройства SCSI могут продолжать работать, используя старый драйвер SCSIPort.

Более подробная информация по этой теме представлена в главе 2.

10.3.2 Служба теневого копирования томов

В Windows Server 2003 впервые представлена служба теневого копирования томов и связанная с ней инфраструктура. Теневые копии томов также называют "моментальными снимками" томов. Различная терминология используется для обеспечения сохранности прав на интеллектуальную собственность. Архитектура теневого копирования показана на рис. 10.6. Она включает в себя компоненты четырех типов, три из которых существуют в нескольких экземплярах.

1. Служба теневого копирования томов.
2. Модули записи теневого копирования.
3. Модули запроса теневого копирования.
4. Поставщики моментальных снимков.

Служба теневого копирования позволяет создавать целостные копии данных, действительные на определенный момент времени, и имеет ряд особенностей.

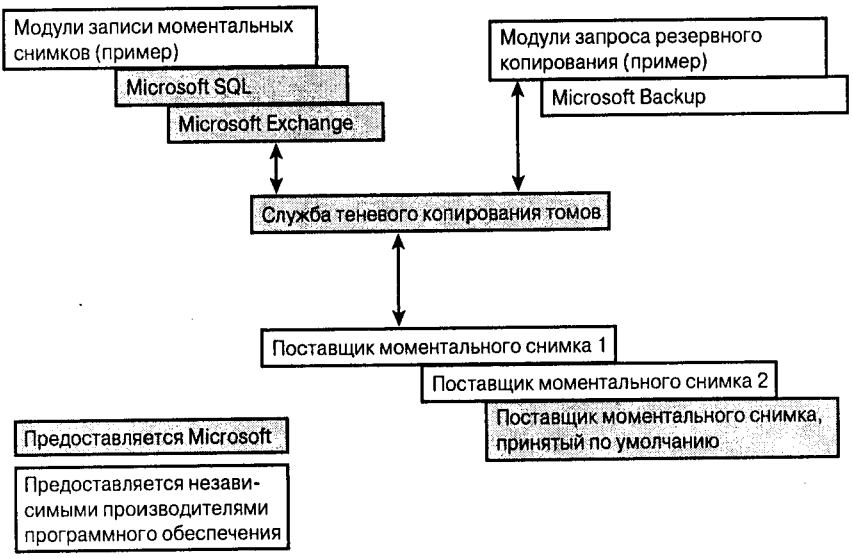


Рис. 10.6. Служба теневого копирования томов

- *Служба теневого копирования томов* создана компанией Microsoft и предоставляет API приложениям резервного копирования, позволяющий запросить создание моментального снимка. Служба обеспечивает задержку операций ввода-вывода, обнуление кэшей и приостановку ввода-вывода системы и приложений, что позволяет создать копию тома в определенный момент времени.
- *Модули записи теневого копирования* — это приложения, например Microsoft SQL Server или Microsoft Exchange, имеющие специальный код для интеграции в службу теневого копирования. Желательно, чтобы все приложения, включая базы данных и приложения планирования ресурсов уровня предприятия, предоставляли поставщика записи теневого копирования для интеграции со службой создания моментальных снимков. Компания Microsoft предоставляет такие модули для Active Directory, SQL Server и Microsoft Exchange.
- *Модули запроса теневого копирования* — это приложения резервного копирования или приложения, которые инициируют создание копии тома, например копии данных для тестирования. Одни из этих модулей создаются компанией Microsoft, другие — независимыми разработчиками программного обеспечения. Приложения резервного копирования, которые предоставляются в Windows Server 2003, также являются модулями запроса теневого копирования.

- *Поставщики моментальных снимков* — это компоненты, которые фактически создают моментальный снимок или копию тома. Компания Microsoft предоставляет поставщика, принятого по умолчанию, который создает копию тома с помощью метода копирования при записи. Ожидается, что производители систем хранения будут создавать собственных поставщиков для создания моментальных снимков. Архитектура допускает использование различных схем для создания моментального снимка, включая разбивку зеркала на аппаратном уровне.

Обратите внимание, что инфраструктура, разработанная Microsoft, позволяет создавать только один моментальный снимок. Тем не менее она соответствует требованиям независимых производителей программного обеспечения, так как предоставляет возможность управления несколькими моментальными снимками и позволяет монтировать моментальные снимки только для чтения.

Дополнительная информация о службе теневого копирования томов приведена в главе 5. Набор SDK для службы теневого копирования томов доступен от компании Microsoft при условии подписания соглашения о неразглашении.

10.3.3 Служба виртуального диска

Служба виртуального диска (Virtual Disk Service — VDS) представляет собой интерфейс управления, интегрированный в Windows Server 2003 и предназначенный для предоставления уровня абстракции при виртуализации дисков независимо от ресурса виртуализации.

Перед описанием архитектурных особенностей VDS стоит рассмотреть причины появления этой службы. Основным мотивом была необходимость в предоставлении администратору возможности автоматизированного и ручного управления следующими операциями:

- формирование тома хранилища, создание на его основе массива RAID 5 объемом не менее 10 Гбайт;
- формирование тома хранилища, увеличение его объема до 10 Гбайт и создания простого тома без возможностей массива RAID.

Другими словами, администратор должен иметь возможность выделять хранилище под моментальные снимки, резервировать снимок и освобождать хранилище, возвращая его в пул свободных хранилищ. Служба виртуального диска предоставляет метод выполнения этих задач независимо от уровня виртуализации и применяемого аппаратного обеспечения, а также различных вариантов подключения устройств хранения.

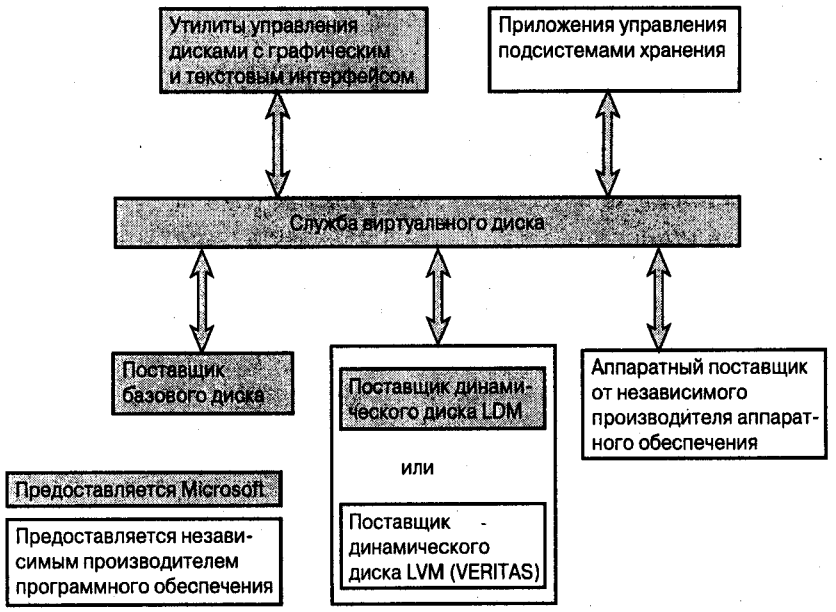


Рис. 10.7. Служба виртуального диска

На рис. 10.7 показана архитектура службы виртуального диска. Закрашенные прямоугольники представляют компоненты, созданные компанией Microsoft и поставляемые в составе операционной системы. Ожидается, что независимые производители аппаратного обеспечения будут создавать аппаратных поставщиков VDS. Служба виртуального диска предоставляет уровень абстракции, что позволяет создавать приложения управления на основе единого интерфейса, вне зависимости от характеристик аппаратного обеспечения на нижних уровнях. Кроме того, VDS дает возможность не вносить изменения в программное обеспечения управляющих приложений, даже если после выхода приложения будет выпущено новое аппаратное обеспечение. Производители аппаратного обеспечения для хранилищ данных могут быть уверены, что их новые системы будут должным образом инициализированы и получат управление посредством VDS.

Компания Microsoft предоставляет двух поставщиков VDS, которые работают с базовыми и динамическими дисками (базовые и динамические диски рассматриваются в главе 6).

Ожидается, что производители аппаратного обеспечения хранилищ также будут создавать поставщиков VDS. Каждый поставщик — это сервер COM, вызываемый VDS. От аппаратных поставщиков ожидается реализация следующих функций:

- обнаружение LUN;
- создание, уничтожение и другие операции по управлению LUN;
- создание объектов COM для LUN, контроллеров хранилища, накопителей и даже самих поставщиков;
- удаленный доступ, обеспечивающий запуск управляющего приложения на рабочей станции и связь этого приложения со службой виртуального диска, работающей на сервере.

На данный момент интерфейсы, необходимые для создания поставщиков, передаются компанией Microsoft после подписания соглашения о неразглашении. Дополнительная информация о службе виртуального диска приведена в главе 7.

10.3.4 Групповой ввод-вывод

Групповой ввод-вывод необходим для обеспечения надежности серверов под управлением семейства Windows NT. Он был впервые представлен в Windows Server 2003 и доступен в Windows 2000 после установки пакета обновлений SP2 и более поздних версий пакетов обновлений. Компания Microsoft предлагает инструментарий разработки для группового ввода-вывода производителям компьютеров и независимым поставщикам аппаратного/программного обеспечения, которые могут использовать этот инструментарий для создания собственных систем, предназначенных для конечных пользователей. Ниже приведены характеристики этих систем.

- Поддерживаются Windows 2000 и Windows Server 2003.
- Достаточно сложная архитектура включает в себя три драйвера от Microsoft и один мини-драйвер, созданный производителем.
- Предоставляется возможность сохранения целостности данных, восстановления целостности и балансировки нагрузки. Кроме того, для одной единицы хранения поддерживается до 32 маршрутов.
- Архитектура основана на уведомлениях PnP и не требует статического определения конфигураций.
- Архитектура совместима с Microsoft Cluster Server.

В функции созданного производителем мини-драйвера (Device Specific Module — DSM) входит следующее:

- идентификация различных маршрутов к одной единице хранения;
- назначение начального маршрута (с использованием балансировки нагрузки, предпочтительного маршрута или с помощью другого алгоритма);

- принятие решения о необходимости повтора ввода-вывода, если возникла ошибка;
- принятие решения о выполнении операции сохранения целостности данных, при которой выбирается альтернативный маршрут;
- определение условий, при которых следует выполнить операцию восстановления целостности;
- выполнение инициализации, специфичной для устройства;
- обработка команд выборки, например `Reserve` и `Release`, а также принятие решения об отправке команд по всем маршрутам или по одному из них.

Групповой ввод-вывод рассматривается в главе 9.

10.3.5 Улучшенное управление

В Windows 2000 ряд инструментов управления используются как в командной строке, так и помощью графического интерфейса. В Windows Server 2003 эта тенденция сохранилась и несколько утилит командной строки позволяют выполнять следующие операции:

- управлять возможностями файловой системы, включая дефрагментацию;
- управлять службой теневого копирования томов;
- управлять томами;
- управлять службами удаленного хранилища.

Кроме того, в Windows Server 2003 для предоставления информации о производительности и управлении системой активно используется интерфейс WMI. Все больше подсистем операционной системы модифицируются для предоставления информации управляющим приложениям через интерфейс WMI. Примерами таких подсистем являются драйверы Storport, служба теневого копирования томов и распределенная файловая система.

10.3.6 Управление томами с поддержкой SAN

Читателям, знакомым с UNIX, давно известно, что в Windows отсутствует эквивалент таблицы монтирования UNIX. Таким образом, Windows стремится монтировать каждый обнаруженный том. Если файловая система тома не распознается, владение томом принимает на себя "чистая" файловая система. Перед подключением сервера под управлением Windows к сети хранения данных администратору следует внимательно настроить маскировку LUN,

зонирование и другие методы управления, чтобы сервер под управлением Windows воспринимал только единицы хранилища, доступ к которым разрешен (и принадлежащие этому серверу). В Windows Server 2003 эта ситуация изменилась.

Компания Microsoft изменила драйвер диспетчера монтирования (он рассматривается в главе 6), улучшив поддержку среды SAN. В частности, драйвер диспетчера монтирования может быть настроен на монтирование только тех томов, которые монтировались ранее, тогда как все новые тома будут игнорироваться. Самый простой способ управления параметрами конфигурации — использование утилиты командной строки `mountvol`.

10.3.7 Создание приложений SAN

В Windows Server 2003 существует ряд возможностей, которые позволяют независимым производителям программного обеспечения создавать эффективные приложения для управления хранилищами в среде SAN. Некоторые из них описаны далее.

- Можно монтировать тома только для чтения.
- Приложения могут использовать новые API, которые вместе со службой теневого копирования томов позволяют приложениям выполнять чтение указанного *моментального снимка тома*. Таким образом, независимые производители аппаратного обеспечения имеют возможность создавать приложения, использующие многоуровневые зеркала и проверяющие целостность данных.
- Новый API позволяет приложениям указывать действительную длину файла. Таким образом, независимые производители программного обеспечения могут создавать распределенную файловую систему, а также приложения резервного копирования и восстановления, записывающие копии уровня блоков на диск и затем устанавливающие действительную длину файла.
- Администраторы могут управлять томами с использованием привилегий безопасности более низкого уровня.
- Добавлены новые интерфейсы, которые позволяют создателям драйверов фильтрации увеличивать функциональность управления. Создатели драйверов фильтрации файловых систем могут также использовать улучшенное управление "кучей" памяти (heap) и процедуры управления объектами устройств, а также новые процедуры управления безопасностью. Кроме того, предоставляются новые интерфейсы управления томами.

10.3.8 Улучшения NTFS

В Windows Server 2003 в файловую систему NTFS были внесены существенные изменения:

- производительность NTFS выросла на 10–15%;
- значительно расширились возможности API, отвечающего за дефрагментацию;
- NTFS стала поддерживать монтирование томов в режиме только для чтения;
- усилены списки управления доступом, принятые по умолчанию, что позволило обеспечить бóльшую безопасность.

10.3.9 Улучшение дефрагментации

В Windows Server 2003 используется интерфейс дефрагментации, который поддерживается подсистемой ввода-вывода Windows в целом и NTFS в частности. Вот некоторые из улучшений системы дефрагментации:

- поддерживается дефрагментация главной таблицы файлов NTFS (MFT), перемещение записей MFT, даже если файл, соответствующий записи в MFT, открыт другим приложением;
- шифрованные файлы могут дефрагментироваться без открытия и чтения, что позволяет усилить безопасность;
- несжатые файлы дефрагментируются по границе дисковых кластеров, а не по границе страницы памяти;
- файловая система NTFS поддерживает дефрагментацию файлов, даже если размер кластера превышает 4 Кбайт;
- файловая система NTFS поддерживает дефрагментацию не только неименованных потоков данных, но и точек повторной обработки и атрибутов файлов.

10.3.10 Улучшения EFS

Шифрованная файловая система была впервые представлена в Windows 2000 и существенно улучшена в Windows Server 2003. Далее описываются основные нововведения.

- В Windows Server 2003 с помощью EFS поддерживается доступ нескольких пользователей к одному файлу. Шифрованная файловая система применяет симметричный алгоритм шифрования (один и тот же ключ используется для шифрации и дешифрации); симметричный ключ

шифруется асимметричным ключом. В частности, симметричный ключ шифруется открытым ключом пользователя и сохраняется в том же файле. В Windows Server 2003 допускается хранение в файле нескольких копий симметричного ключа, зашифрованных открытыми ключами различных пользователей.

- Поддерживается полная проверка по списку отозванных пользовательских сертификатов. Это позволяет ограничить доступ тем пользователям, которые ранее имели доступ к файлам.
- Поддерживается большее количество алгоритмов шифрования. Для этого применяются поставщики служб шифрования.
- Допустимо сквозное шифрование поверх WebDAV. В Windows 2000 перед передачей по сети через интерфейс WebDAV содержимое файлов расшифровывалось. В Windows Server 2003 дешифрация проводится локально на клиенте.
- Допускается хранение в зашифрованном виде автономных файлов. В Windows 2000 шифрование кэша автономных файлов не поддерживалось.
- Шифрованные файлы могут сохраняться в Web-папке.

10.3.11 Службы RSS

Системой управления иерархическим хранилищем (HSM), поддерживаемой службами RSS в Windows 2000, в качестве вторичного носителя допускалось использование только накопителя на магнитной ленте (служба RSM в Windows 2000 поддерживает и другие носители, например оптические накопители, ленточные магазины и автоматы). В Windows Server 2003 предоставляется поддержка и других носителей для HSM.

10.3.12 Улучшение процесса загрузки

Операционные системы Windows XP и Windows Server 2003 оптимизированы для сокращения времени их загрузки. Считывая файлы с диска загрузчик (`ntldr`) использует по одной операции ввода-вывода на файл. Кроме того, операционная система перекрывает ввод-вывод данных на диск инициализацией устройств, а также откладывает инициализацию системных процессов и служб, которые не важны в процессе загрузки.

10.3.13 Улучшение программы CHKDSK

Повышению надежности Windows 2000 способствовало сокращение количества ситуаций, при которых необходим запуск утилиты `CHKDSK`. Кроме

того, сокращено время, необходимое CHKDSK для завершения работы. Эта же тенденция сохранилась и в Windows Server 2003. Во время проведения теста с тремя миллионами файлов завершение работы утилиты CHKDSK на сервере под управлением Windows Server 2003 заняло в 12 раз меньше времени, чем на сервере под управлением Windows 2000.

10.3.14 Улучшение кэширования

В Windows XP тесты производительности ввода-вывода показали, что скорость обмена данными с дисками SCSI намного ниже аналогичных показателей в Windows 2000. И это связано с очень интересной проблемой.

Операционная система и производители жестких дисков поддерживают функции кэширования для повышения производительности и пропускной способности подсистемы ввода-вывода. В Windows NT существует диспетчер кэша, который предоставляет эти функции всем файловыми системам. Кроме того, некоторые жесткие диски также обеспечены быстродействующей кэш-памятью. Хотя кэширование и позволяет повысить быстродействие, данные записываются в кэш, а не на носитель. Если данные не будут вовремя перенесены из кэша на носитель, они могут быть потеряны. Чтобы предоставить приложениям возможность управлять кэшем, Windows NT поддерживает описанные ниже операции.

- Приложение может указать параметр FILE_FLAG_WRITE_THROUGH при вызове функции CreateFile; это говорит о запрете завершения операции записи до записи данных на диск. От драйверов Windows NT ожидается передача этой функции устройствам хранения с помощью флага SCSI Force Unit Access (FUA). С помощью флага FUA, определенного в стандартах SCSI, кэширование отключается для отдельных операций ввода-вывода.
- Приложение может указать параметр FILE_FLAG_NO_BUFFERING при вызове функции CreateFile, что указывает на запрет кэширования на уровне файловой системы.
- Приложение может использовать функцию FlushFileBuffers для сброса всех данных открытого дескриптора файла из системного кэша и отправки жесткому диску сигнала о необходимости обнуления собственного кэша. Обратите внимание: вопреки названию функции, сбрасываются все данные, находящиеся в кэше жесткого диска.

Проблема заключается в том, что операционная система некорректно обрабатывала запросы приложений на отказ от кэширования и Windows Server 2003 стала первой системой семейства Windows, которая полноценно работает с запросами сквозной записи. Это означает, что показатели производительности Windows NT 4.0 и Windows 2000 были искусственно завышены. Кроме

того, некоторые администраторы применяли утилиту настройки от производителя хранилища данных, которая позволяла отключать кэш накопителя. Но в этом случае существуют и другие сложности, описанные далее.

- В Windows XP ошибка исправлена для базовых, но не для динамических дисков. Таким образом, некоторое время складывалось впечатление, что компания Microsoft не спешила в полной мере поддерживать динамические диски. Концепция базовых и динамических дисков, описывающая запись на диск информации о логической структуре диска, рассматривается в главе 6.
- В некоторых книгах издательства Microsoft Press авторам приложений некорректно рекомендовалось использовать флаг `FILE_FLAG_WRITE_THROUGH` функции `CreateFile` в тех случаях, когда требовалась более высокая производительность. Компания Microsoft пообещала исправить собственные приложения, утилиты и инструменты, удалив флаг `FILE_FLAG_WRITE_THROUGH` для достижения максимальной эффективности. Кроме того, предполагалось создать уровень совместимости приложений, позволяющий должным образом работать тем приложениям, которые еще не были модифицированы для удаления параметра `FILE_FLAG_WRITE_THROUGH`.

10.3.15 Автоматическое восстановление системы

Для повышения надежности Windows Server 2003 использован механизм восстановления после критических отказов в работе. В частности, сервер может быть восстановлен после отказа в работе диска в одно действие, которое приведет к восстановлению всей информации операционной системы и системных данных. Автоматическое восстановление системы основано на службе теневого копирования томов и требует наличия дисковода для гибких дисков.

10.3.16 Улучшения DFS

Впервые распределенная файловая система стала доступна в Windows NT 4.0 и была заметно модифицирована в Windows 2000. В Windows Server 2003 она также была несколько усовершенствована.

- Реализована поддержка распределенной файловой системой нескольких корней. Это означает, что компания может объединить различные ресурсы в общее пространство имен, при этом сохраняя несколько пространств имен для обеспечения безопасности и упрощения администрирования. Эта возможность будет особенно востребована в больших корпорациях, организационная структура которых состоит из нескольких

подразделений (например, компания — разработчик программного обеспечения может поставлять как обычные клиентские программы, так и системы для обеспечения безопасности). Кроме того, новые возможности DFS понадобятся после приобретения компании другой корпорацией, причем после слияния предполагается раздельное администрирование корпоративных ресурсов двух компаний.

- Улучшена репликация файлов.
- Улучшена балансировка нагрузки.
- Пользователи могут выбирать серверы, которые физически расположены ближе, что позволяет повысить производительность.

10.3.17 Перенаправитель WebDAV

Серверы и клиенты под управлением Windows 2000 предоставляли различные способы подключений, в том числе поддержку систем CIFS, NFS, NetWare и Macintosh. В Windows Server 2003 добавлен клиент WebDAV (Web Distributed Authoring and Versioning). Это стандартный протокол, который позволяет использовать HTTP в качестве основного транспортного механизма. Например, пользователь, получающий файл Excel с сервера, может не подозревать об использовании протокола CIFS, но для этой же цели можно применять и протокол WebDAV.

10.3.18 Улучшение инфраструктуры драйверов

В Windows Server 2003, как и в Windows XP, создателям драйверов предоставляется расширенный инструментарий. Кроме того, ужесточена процедура тестирования и проверки на соответствие стандартам Windows, что сделало драйверы более надежными. Создатели драйверов не только получают расширенный инструментарий и дополнительную информацию, но и доступ к обратной связи, что позволяет отлаживать существующие драйверы и создавать новые версии.

10.3.19 Поддержка API адаптера шины

Ассоциация SNIA определила API на языке C, который позволяет приложениям управления хранилищем администрировать адаптеры шины Fibre Channel. Этот интерфейс поддерживает запрос и назначение параметров конфигурации адаптера шины, а также измерение параметров производительности адаптера.

Хотя Microsoft и члены ассоциации SNIA поддерживают одинаковый API адаптера шины, используемые подходы несколько различаются. Архитектура, предложенная SNIA (рис. 10.8), включает три обязательных компонента.

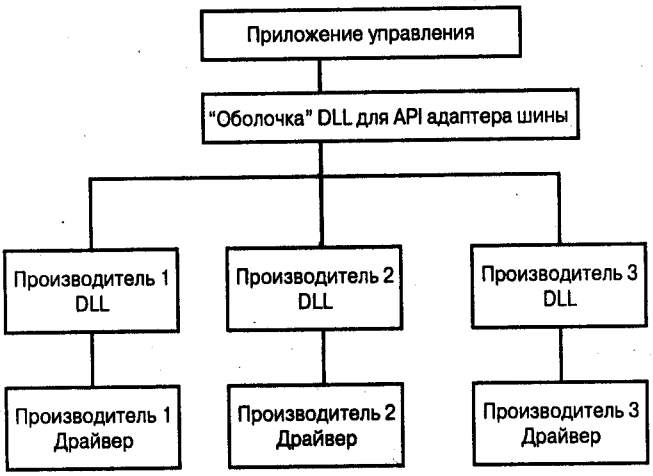


Рис. 10.8. Программный интерфейс адаптера шины от ассоциации SNIA

1. Универсальная динамически подключаемая библиотека API адаптера шины, поддерживаемая SNIA. В ней предоставляется стандартный интерфейс для приложений управления. На нижнем уровне библиотека взаимодействует с различными библиотеками, созданными производителями устройств.
2. Динамически подключаемая библиотека от производителя адаптера, которая подключается к DLL программного интерфейса адаптера шины. Библиотека от производителя предоставляет информацию для управления и взаимодействует с драйвером производителя посредством закрытых вызовов управления вводом-выводом.
3. Драйвер устройства, созданный производителем для данного адаптера шины.

Хотя такой подход обладает определенными преимуществами, компания Microsoft отмечает некоторые связанные с ним проблемы.

- Невозможно эффективно управлять распространением и учетом версий динамически подключаемых библиотек. Это еще один пример ситуации, когда несколько приложений устанавливают собственные библиотеки, потенциально перезаписывая библиотеки, установленные другими приложениями.
- Производитель адаптера шины должен создать не только драйвер устройства и закрытый интерфейс управления вводом-выводом для этого драйвера, но и собственную динамически подключаемую библиотеку, а также внести определенные изменения в библиотеку — оболочку

адаптера шины, чтобы предоставить необходимые интерфейсы созданной DLL.

- Тестирование и сертификация драйверов производителей, в которых используются закрытые вызовы управления вводом-выводом, представляет собой исключительно сложный процесс; например, как можно проверить, что некорректные параметры вызова управления вводом-выводом не приведут к переполнению буфера?
- Архитектура на первый взгляд выглядит расширяемой, но при более детальном рассмотрении очевидно, что производители адаптеров шины всегда будут "догонять" разработчиков приложений управления, добавляя код, который работает с расширениями, характерными для определенного производителя.
- Не поддерживается связь и управление в режиме ядра. В архитектуре SNIA требуется, чтобы приложение управления ожидало момента полной работоспособности подсистемы пользовательского режима Windows NT. Но для некоторых операций управления, например для маскировки LUN, необходимо наличие драйвера в режиме ядра, который будет выполнять определенные действия в процессе загрузки операционной системы еще до того, как будет запущена подсистема пользовательского режима.

Компания Microsoft предлагает несколько иной метод, который демонстрируется на рис. 10.9 и описан ниже.

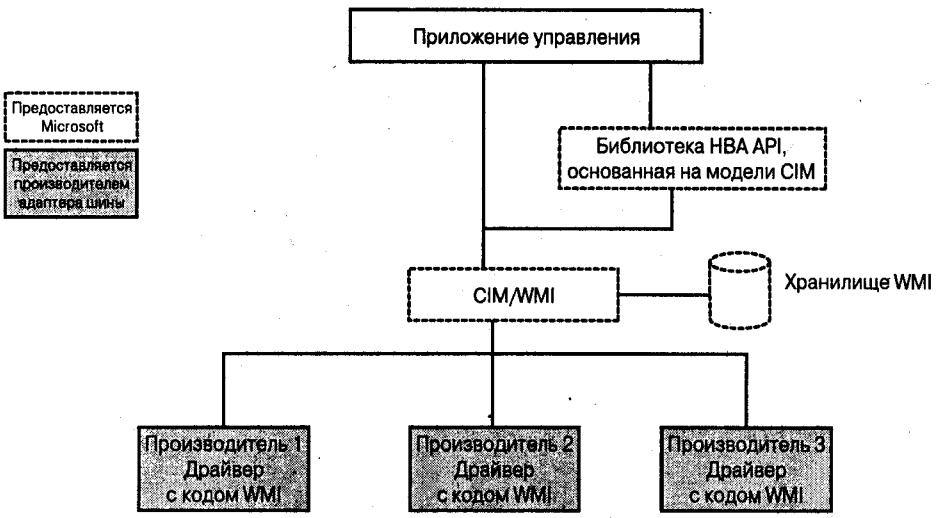


Рис. 10.9. Программный интерфейс адаптера шины от компании Microsoft

- Универсальная динамически подключаемая библиотека программного интерфейса приложений для адаптера шины от ассоциации SNIA. Эта DLL предоставляет интерфейс для приложений управления более высокого уровня. На более низком уровне библиотека подключается к интерфейсу WMI, который представляет собой реализацию модели CIM (Common Information Model) от Microsoft. Как уже отмечалось, CIM — это объектно-ориентированная модель управления, принятая ассоциацией SNIA и рабочей группой DMTF.
- Драйвер устройства для адаптера шины, созданный производителем; этот драйвер реализует интерфейс WMI и предоставляет интерфейс управления и настройки в репозитории WMI. Поскольку WMI является двунаправленным интерфейсом, драйвер обеспечивает поддержку функций пакетов IRP для WMI, что позволяет приложению управления устанавливать конфигурационные параметры драйвера.
- Динамически подключаемая библиотека, которая осуществляет преобразование данных между интерфейсом WMI и API адаптера шины от SNIA.

Метод Microsoft обладает определенными преимуществами.

- Все представленные интерфейсы стандартизированы, тогда как при использовании схемы SNIA интерфейс между универсальной библиотекой HBA API и библиотекой производителя закрыт и определяется производителем. Метод Microsoft соответствует общепринятой модели CIM.
- Поддержка расширяемости, так как производитель может расширить класс WMI или определить новый класс и поместить в нем управляющую информацию. В данном случае также прослеживается соответствие модели CIM.
- Основным преимуществом является возможность использования модели CIM или API адаптера шины от ассоциации SNIA. Если приложение использует API адаптера шины от SNIA, оно может работать без изменений, так как код WMI драйвера и DLL от Microsoft преобразуют информацию WMI в данные API адаптера шины от SNIA.
- Описанная архитектура позволяет компоненту режима ядра опрашивать драйвер производителя и выполнять управляющие действия.

Обратите внимание, что интерфейсы WMI, необходимые для создания драйвера адаптера шины, предоставляются в Windows 2000.

10.3.20 Диски с таблицами разделов GUID

Существует разновидность Windows Server 2003 для 64-разрядных процессоров, которая поддерживает промышленный стандарт EFI (Extensible

Firmware Interface). Этот стандарт стал заменой устаревших программ BIOS, которые до сих пор используются в индустрии персональных компьютеров.

Стандартом EFI определяется таблица разделов GUID (GUID Partition Table — GPT). Точная структура таблицы GUID (Globally Unique Identifier — глобально-уникальный идентификатор) рассматривается в главе 16 спецификации EFI, которая доступна по адресу: `http://developer.intel.com/technology/efi/download.htm`.

Диск с GPT может иметь до 2^{64} логических блоков. В спецификации EFI используется термин *логический блок* (logical block) для описания того, что обычно называется *дисковым кластером* (disk cluster) — наименьшей единицей выделения дискового пространства. Поскольку в EFI указывается размер логического блока, равный 512 байт, максимальный размер диска составляет 18 Эбайт. Диск с GPT может иметь любое количество разделов, как и динамический диск. Кроме того, наравне с динамическими дисками, диск GPT обладает самоописанием, т.е. сведения о логической структуре диска размещены на самом диске. Диски GPT хранят информацию о разделах избыточным образом, что позволяет добиться устойчивости к ошибкам. В то время как диски GPT являются промышленным стандартом, динамические диски Windows 2000 — стандарт закрытый. Правда, по сравнению с компьютерами, использующими BIOS, выпущено сравнительно мало компьютеров, поддерживающих стандарт EFI.

Чтобы сохранить от повреждения данные на диске GPT при подключении к устаревшим системам, на диске содержится главная загрузочная запись (MBR), описывающая весь диск. Таким образом, компьютер на основе BIOS получит сведения о разбивке диска на разделы.

Загрузочные диски GPT имеют новый тип раздела, который называется *системный раздел EFI* (EFI System Partition — ESP). Этот раздел содержит файлы, необходимые для загрузки операционной системы, например `ntldr`, `hal.dll`, `boot.txt` и драйверы. Раздел ESP может располагаться на диске GPT или MBR, как указано в спецификации EFI. Операционная система Windows Server 2003 для 64-разрядных процессоров требует, чтобы раздел ESP находился на диске GPT.

Кроме того, существует еще один тип раздела — *зарезервированный раздел Microsoft* (Microsoft Reserved Partition — MSR). Диски GPT запрещают использование скрытых секторов, поэтому для компонентов, которые раньше использовали скрытые секторы, предназначен раздел MSR. Он создается при первой разбивке диска у производителя компьютера или при установке Windows Server 2003. На дисках объемом менее 16 Гбайт раздел MSR не превышает 32 Мбайт. Для дисков объемом более 16 Гбайт используется раздел MSR, объем которого составляет 128 Мбайт.

Следует подчеркнуть, что реализация спецификации EFI от Intel предлагается *только* для компьютеров с 64-разрядной архитектурой центрального процессора. Хотя стандартом и не запрещено создание 32-разрядной версии EFI, таковой не существует. Таким образом, реализация кода нижнего уровня и кода загрузки Windows для 32- и 64-разрядной платформ будет иметь существенные отличия.

Версия Windows Server 2003 для 64-разрядной архитектуры должна загружаться с диска GPT, причем доступ к старым дискам сохраняется (однако загрузка с них не поддерживается). В Windows Server 2003 для 32-разрядной архитектуры используются диски формата MBR.

10.4 После Windows Server 2003

Не желая почивать на лаврах после внесения в семейство Windows NT описанных возможностей по работе с подсистемами хранения, компания Microsoft разрабатывает новые технологии для следующей после Windows Server 2003 системы. В частности, речь идет о модификации стека iSCSI и дальнейшем усовершенствовании инфраструктуры управления хранилищем.

10.4.1 Служба виртуализации связной архитектуры

Эта служба уже рассматривалась ранее; в ней реализована попытка предоставить единый интерфейс для управления коммутаторами связной архитектуры. Такой интерфейс не должен зависеть от производителя коммутатора, модели коммутатора и его возможностей.

Рассмотрим последовательность действий системного администратора для проведения резервного копирования.

1. Создание тома (может потребоваться управление дисками или массивами RAID).
2. Обеспечение доступности тома для механизма создания моментальных снимков (может потребоваться перенастройка зонирования).
3. Создание моментального снимка.
4. Том с моментальным снимком делается доступным для сервера резервного копирования.
5. Проведение резервное копирования.
6. Возвращение снимка в пул свободных ресурсов хранилища, для чего может также потребоваться дополнительная настройка коммутатора и LUN.

Обычно все происходит таким образом: администратор хранилища выполняет часть действий вручную, запускает программное обеспечение, выполняет еще несколько ручных операций, запускает другое программное обеспечение и т.д.

Служба виртуализации связной архитектуры при использовании с VDS (см. раздел 10.3.3) позволяет полностью автоматизировать эти задачи. Служба виртуализации не имеет обозначенной даты выпуска, и Microsoft по-прежнему не дает сведений на эту тему.

10.4.2 Управление LUN

Операционная система Windows NT исключительно "агрессивно" реализует поддержку подсистемы хранения, монтируя диск сразу после его обнаружения. С каждого обнаруженного диска считывается таблица разделов, после чего монтируются найденные файловые системы. При такой схеме вероятность повреждения данных достаточно велика.

Во время работы над этой книгой функция *маскировки LUN* поддерживалась сторонними производителями аппаратного и программного обеспечения, но отсутствовала в Windows. Одной из проблем, связанных с использованием этих функций, является возможное отключение загрузки соответствующего драйвера или утилиты стороннего производителя, когда операционная система загружается в безопасном режиме. Таким образом, вероятность повреждения данных все равно остается. Маскировка LUN поддерживается и аппаратным обеспечением, например массивами RAID или коммутаторами Fibre Channel, однако возможность программного управления отсутствует. Обычно аппаратное обеспечение предоставляется с графической утилитой управления, которую можно запускать удаленно.

В будущих версиях Windows NT ожидается реализация маскировки LUN в рамках стека драйверов Microsoft, в частности в драйвере Storport.

Что касается заявлений с последней торговой конференции (WinHEC), от Microsoft ожидается предоставление следующих возможностей:

- вызовы управления вводом-выводом, которые позволят приложениям управлять маскировкой LUN;
- список включения, настраиваемый вызовами управления вводом-выводом (IOCTL); должен содержать устройства, которые будут предоставляться приложениям;

- возможность обхода маскировки LUN[2].

Во время подготовки этой книги к печати появилась информация, что Microsoft работает над реализацией маскировки LUN в драйвере порта. Преимущество переноса такой функции в драйвер порта состоит в обязательной загрузке драйвера, что значительно сокращает возможность обхода маскировки LUN. Вероятность загрузки некорректного драйвера порта намного ниже вероятности загрузки некорректного драйвера мини-порта.

Не следует предполагать, что все производители аппаратного обеспечения предоставляют одинаковые возможности по динамической конфигурации любого типа порта. Одни устройства обладают встроенными функциями по переконфигурированию портов "на лету", без перезагрузки и отключения питания, в то время как другим для этого требуется модернизация прошивки или замена платы расширения для определенного порта, что может потребовать отключения питания.

10.4.3 Поддержка iSCSI

Протокол iSCSI стал попыткой реализации блочного хранилища поверх существующей централизованной сетевой инфраструктуры. Хотя от iSCSI не ожидается такой производительности, как от других технологий взаимодействия с хранилищами данных, например Fibre Channel, основное внимание уделяется системам, которым не требуется высокая производительность и понадобится существующая сетевая инфраструктура. Ожидается создание аппаратных ускорителей для стека протоколов TCP/IP, в которых накладные расходы на обработку определенных протоколов переносятся с центрального процессора сервера на специальное аппаратное обеспечение сетевой карты, что будет только способствовать распространению протокола iSCSI.

Компания Microsoft дала понять, что ведется активная реализация протокола iSCSI, которая со временем станет доступной в Windows NT. Определенной даты не сообщалось; предположительно поддержка iSCSI будет обеспечена до выхода Windows Longhorn в 2005 году.

На рис. 10.10 представлена архитектура реализации протокола iSCSI в Windows NT. Инициатор iSCSI реализован в виде драйвера мини-порта, который может работать как под управлением драйвера SCSIPort, так и под управлением драйвера Storport.

[2]Не зная подробностей реализации, можно предположить, что этим будет заниматься один из компонентов ядра. Это не несет в себе угрозы безопасности, так как философия Windows заключается в доверии программному обеспечению, работающему на уровне ядра, и в ограничении возможностей по установке программного обеспечения для данного режима. Таким образом, администратор может использовать только проверенное и заведомо надежное программное обеспечение.

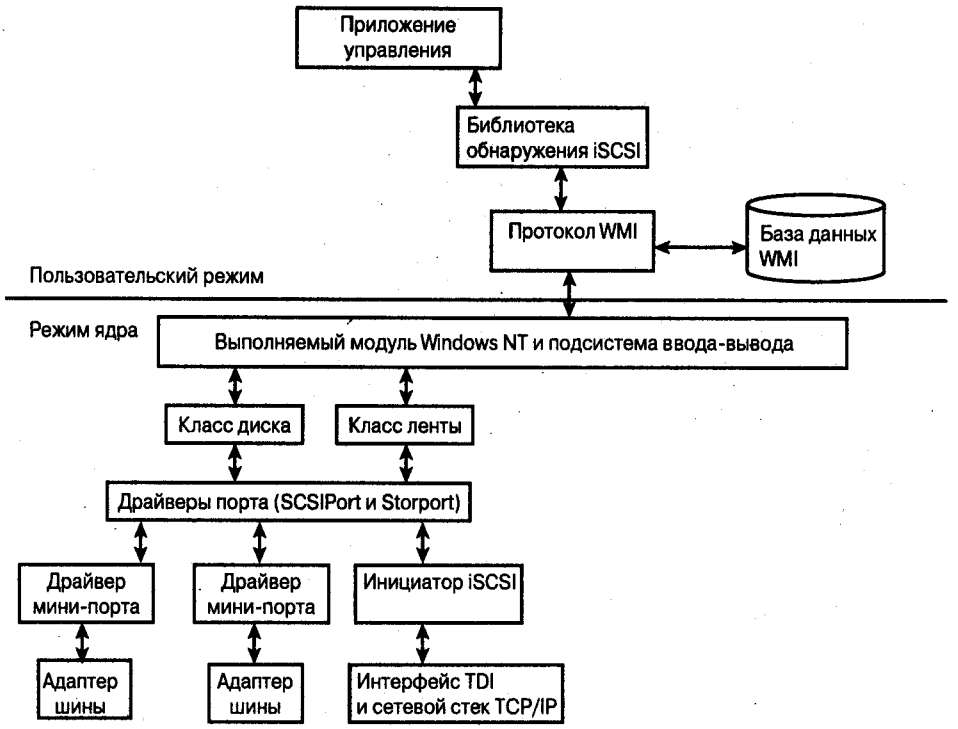


Рис. 10.10. Архитектура протокола iSCSI в Windows NT

Библиотека обнаружения динамически отслеживает все изменения и работает как единый репозиторий для всех LUN, обнаруженных любым способом, включая уведомления клиента или порта службой iSNS (Internet Storage Name Service). Библиотека обнаружения предоставляет API для приложений управления, который может использоваться для обнаружения новых LUN и, если возможно, регистрации в обнаруженной LUN.

Планы Microsoft по реализации протокола iSCSI включают несколько пунктов.

- Основное внимание будет уделяться реализации протокола iSCSI в Windows Server 2003. Ожидается, что iSCSI будет доступен в Windows 2000 и Windows XP.
- Будет предоставлен код для сервера и клиента iSNS.
- В качестве основного механизма защиты будет использоваться протокол IPSec.
- Особое внимание получит реализация инициатора iSCSI. На данный момент не существует планов по реализации целевого устройства iSCSI для Windows NT.

- Передача информации между библиотекой обнаружения и инициатором iSCSI будет осуществляться с помощью интерфейса WMI.

Кроме того, драйвер порта предоставляет маршрут взаимодействия с мини-портом HCA, а также диспетчерами InfiniBand, например диспетчерами подсетей и подключений. Диспетчер подсети отвечает за управление топологией связной архитектуры через программирование коммутаторов и установку атрибутов HCA. Кроме того, диспетчер подсети предоставляет точку соединения политик управления, связанных с управлением доступом, производительностью и устойчивостью к ошибкам (групповой ввод-вывод). Диспетчер подключений реализует возможности InfiniBand, ориентированные на подключения, а также основанные на политиках методы сохранения целостности данных.

10.4.4 Групповой ввод-вывод в Windows 2000/Server 2003

Групповой ввод-вывод данных поддерживается в Windows 2000 и Windows Server 2003. Компания Microsoft предоставляет производителям инструментарий разработки, поэтому готовые системы доступны только у определенных компаний. Ожидается, что Microsoft внедрит поддержку группового ввода-вывода и для сменных носителей (на данный момент поддерживаются только жесткие диски). Поэтому часть прямоугольника группового ввода-вывода на рис. 10.5 закрашена другим цветом.

10.5 Чего не хватает?

Компания Microsoft признала необходимость добавления некоторых возможностей, но никак не обозначила намерения по их реализации. В этом разделе рассматривается ряд технологий и делается попытка прогноза их дальнейшего развития.

10.5.1 Загрузка через SAN

Операционная система Windows 2000 поддерживает загрузку через SAN, но в этом процессе существует множество "подводных камней". Компания Microsoft обеспечивает ограниченную поддержку загрузки через SAN. В частности, существуют описанные ниже проблемы.

- Все используемое программное и аппаратное обеспечение должно быть совместимо с Windows 2000.
- Невозможно использовать кольцо Fibre Channel с разделением доступа (Fibre Channel Arbitrated Loop — FC-AL), так как в ядро Windows NT

не вносились изменения, позволяющие обрабатывать значительные задержки среды FC-AL при повторной инициализации кольца.

- Операционная система Windows 2000 требует эксклюзивного доступа к LUN, с которого проводится загрузка. Соответствующая маскировка выполняется на коммутаторе или через управление адаптерами шины.
- Поставщик оборудования сети SAN должен принимать участие в процессе планирования и гарантировать поддержку в случае возникновения каких-либо проблем.
- Рекомендуется держать файл подкачки на локальном жестком диске.
- Практически все системы требуют отключения группового ввода-вывода в процессе загрузки.

Поддержка загрузки через SAN в операционной системе имеет множество преимуществ. В частности, уменьшится задержка передачи данных в сети. Использование протокола iSCSI позволит выполнять удаленную загрузку по сети хранения данных с помощью платы Gigabit Ethernet. Кроме того, неплохо бы иметь возможность маскировки LUN на ранних этапах загрузки операционной системы.

10.5.2 Сокращение количества уровней в стеке драйверов подсистемы хранения

Модель драйверов Windows NT построена на принципах модульности и использует несколько уровней, что позволяет задействовать любые необходимые драйвера. Возможно, модель слишком успешно реализует поставленную задачу. Обычный сервер под управлением Windows 2000 применяет около десятка драйверов между приложением и физическим устройством. Рассмотрим набор драйверов в стеке:

- драйвер мини-порта Storport или SCSIPort;
- драйвер порта Storport или SCSIPort;
- драйвер класса диска;
- драйвер LDM, который на самом деле включает в себя несколько драйверов;
- драйвер `diskperf`;
- драйвер фильтрации антивирусного сканера;
- драйвер файловой системы;
- драйвер фильтрации файловой системы, который также может состоять из нескольких драйверов, например драйверы фильтрации точек повторной обработки удаленного хранилища, SIS, символьных ссылок и EFS.

Возможно, исправить ситуацию поможет добавление функций одного драйвера в другой. В Windows XP предоставлен ограниченный механизм обратного вызова для драйверов фильтрации. Остается лишь наблюдать за развитием этой тенденции.

10.5.3 Групповой ввод-вывод для протокола iSCSI

Набор разработки для систем группового ввода-вывода не поддерживает работу с iSCSI. С расширением сферы применения iSCSI существует вероятность, что Microsoft предоставит отказоустойчивые и высокопроизводительные модули поддержки iSCSI.

10.6 Сложности практической реализации

Компания Microsoft намерена превратить Windows в полноценную платформу для организации хранилищ данных. Возможности новых версий Windows NT ясно подчеркивают эту тенденцию. Безусловно, Microsoft будет играть важную роль в индустрии корпоративных систем хранения данных.

Менеджерам компаний, принимающим решения о закупках, стоит обратить внимание на возможности, связанные только с Windows Server 2003 и Windows 2000. Поэтому следует внимательно просмотреть планы независимых производителей аппаратного и программного обеспечения по поддержке этих операционных систем.

10.7 Резюме

С развитием платформы Windows NT существенно улучшалась поддержка подсистем хранения данных. Операционные системы Windows NT 4.0 и Windows 2000 предоставляли все больше средств для обеспечения надежности, масштабируемости и отказоустойчивости. Операционная система Windows Server 2003 предлагает новые возможности по обеспечению отказоустойчивости (групповой ввод-вывод), производительности (модель драйверов Storport) и управлению (служба виртуального диска, поддержка приложений и диспетчер монтирования SAN, интерфейс управления на основе командной строки). После выхода Windows Server 2003 компания Microsoft планирует добавить расширенную поддержку протокола iSCSI в линии операционных систем Windows NT.

Список основных источников информации

Глава 1

- Nagar R. *Windows NT File System Internals: A Developer's Guide.* — Cambridge, England : OReilly,1997.
- Oney W. *Programming the Microsoft Windows Driver Model*, 2nd Ed. — Redmond, WA : Microsoft Press, 2002.
- Solomon D. A., Russinovich M. E. *Inside Microsoft Windows 2000*, 3rd Ed. — Redmond, WA : Microsoft Press, 2000.
- Viscarola P., Mason W. A. *Windows NT Device Driver Development.* — Indianapolis, IN : MacMillan Technical Pub., 1999.
- Набор разработки Windows Driver Development Kit: `http://www.microsoft.com/ddk`.
- Набор IFS для Windows XP SP1: `http://www.microsoft.com/ddk/IFSkit`.

Глава 2

- Статья базы знаний Microsoft KB310072 *Adding Support for More Than Eight LUNs in Windows Server*: `http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q310072`.
- Web-узел ассоциации ANSI T10: `http://www.t10.org`.
- Данные о переносе драйверов мини-порта на модель Storport: `http://www.microsoft.com/hwdev/tech/storage/default.asp`.
- Sawert B. *The Programmer's Guide to SCSI* — Reading, MA : Addison-Wesley,1998.
- Schmidt F. *The SCSI Bus and IDE Interface: Protocols, Applications, and Programming.* — Reading, MA : Addison-Wesley, 1995.

- Web-узел торговой ассоциации SCSI: `http://www.scsita.org`.
- Web-узел Microsoft с данными о корпоративных системах хранения, представленными на выставке WinHEC в 2001 году: `http://www.microsoft.com/winhec/winhec2001.mspx`.

Глава 3

- Web-узел MSDN, посвященный протоколу CIFS: `http://msdn.microsoft.com/library/default.asp?url=/downloads/list/windevwin.asp`.
- Статья базы знаний Microsoft KB161372 *How to Enable SMB Signing in Windows NT*: `http://support.microsoft.com/support/kb/articles/Q161/3/72.asp`.
- Leach P. J., Naik D. C. *A Common Internet File System (CIFS/1.0) Protocol: Preliminary Draft* [ныне устаревший документ RFC]. — Ассоциация IETF, 1997: `http://www.ubiqx.org/cifs/rfc-draft/draft-leach-cifs-v1-spec-02.html`.
- Leach P., Perry D. *CIFS: A Common Internet File System.* — Web-узел Microsoft Interactive Developer, 1996: `http://www.microsoft.com/Mind/1196/CIFS.htm`.
- Программа по разработке коммуникационных протоколов Microsoft: `http://www.microsoft.com/legal/protocols`.
- Web-узел NFS версии 4, поддерживается компанией SUN Microsystems: `http://www.nfsv4.org`.
- Pawlowski B., Shepler S., Beame C., Callagahn B., Eisler M., Noveck D., Robinson D., Thurlow R. *The NFS Version 4 Protocol*: `http://www.nluug.nl/events/sane2000/papers/pawlowski.pdf`.
- Протоколы для межсетевого взаимодействия: SMB, версия 2: `http://www.opengroup.org/products/publications/catalog/c209.htm`.
- Shepler S. *NFS Version 4 Design Considerations.* — Группа NWG, документ RFC 2624, июнь 1999: `http://www.ietf.org/rfc/rfc2624.txt`.
- Shepler S., Callaghan B., Robinson D., Thurlow R., Beame C., Eisler M., Noveck D. *NFS Version 4 Protocol.* — Группа NWG, документ RFC 3010, декабрь 2000: `http://www.ietf.org/rfc/rfc3010.txt`.
- Официальный Web-узел ассоциации SNIA: `http://www.snia.org`.

- Наборы для разработки драйверов Windows: `http://www.microsoft.com/ddk`.

Презентации на Web-узле SNIA

- Конференция CIFS 2000: `http://www.snia.org/education/presentations/cifs_2000`.
- O'Shea G. *Operating Systems Case Study Windows 2000 (NT)*: `http://research.microsoft.com/~gregos/MScNT.ppt`.

Глава 4

- Комитет ANSI T11: `http://www.t11.org` (комитет курирует разработку всех протоколов, связанных с интерфейсом Fibre Channel).
- CERN Fibre Channel: `http://hsi.web.cern.ch/HSI/fcs`.
- Clark T. *Designing Storage Area Networks.* — Reading, MA : Addison-Wesley, 1999.
- Fibre Channel — учебники и ресурсы лаборатории InterOperability Lab.: `www.iol.unh.edu/training/fc/fc_tutorial.html`.
- Официальный Web-узел ассоциации FCIA: `http://www.fibrechannel.org`.
- Разработка технологий хранилищ данных на платформе Windows: `http://www.microsoft.com/hwdev/tech/storage/default.asp`.

Глава 5

- Evaluator Group Report: An Analysis of the IBM TotalStorage NAS Products 200/300/300Gateway 300: `http://www.storage.ibm.com/snetwork/nas/analysis.html`.
- Griswold B. *Enterprise Storage*: `http://www.microsoft.com/winhec/presents2001/EntStor.ppt`.
- *I/O Subsystem Enhancements.* — Web-узел MSDN: `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/appendix/hh/appendix/enhancements5_33oz.asp`.
- Moore G. E. *The Continuing Silicon Technology Evolution inside the PC Platform.* — Intel: `http://www.intel.com/update/archive/issue2/feature.htm`.

- Спецификации NDMP (Network Data Management Protocol): http://www.ndmp.org/download.
- Протокол NDMP: http://www.ndmp.org.
- Документ *Preventing Data Loss during Backups Due to Open Files*: http://www.stbernard.com/products/docs/ofm_whitepaper.pdf.
- Russinovich M., Solomon D. *Windows XP: Kernel Improvements Create a More Robust, Powerful, and Scalable OS* // MSDN Magazine. — December 2001: http://msdn.microsoft.com/msdnmag/issues/01/12/XPKernel/XPKernel.asp.
- Рабочая группа по разработке технологий резервирования ассоциации SNIA: http://www.snia.org/tech_activities/workgroups/backup.
- Версии спецификации Extended Copy от T10 и SNIA: http://www.snia.org/tech_activities/workgroups/backup/docs.
- Williams S., Kindel C. *Component Object Model: A Technical Overview.* — Web-узел MSDN: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncomg/html/msdn_comppr.asp.
- Документы и презентации конференции WinHEC 2002: http://microsoft.com/winhec/sessions2002/default.mspx.

Глава 6

- Статья базы знаний Microsoft KB223316 *Best Practices for Encrypting File System*: http://support.microsoft.com/support/kb/articles/Q223/3/16.asp.
- Bolosky W. J., Corbon S., Goebel D., Douceur J. R. *Single Instance Storage in Windows 2000*: http://research.microsoft.com/sn/Farsite/WSS2000.pdf.
- *Comparison: Microsoft Logical Disk Manager (LDM) and VERITAS Volume Manager for Microsoft Windows.* — Mountain View, CA : VERITAS Software Corporation, 2002: http://www.veritas.com/de/produkte/datasht/ldm_vm_vergleich_wp.pdf.
- Статья базы знаний Microsoft KB299726 *A Description of the Functions of the Single Instance Storage Filter and the Groveler Service*: http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q299726.
- *Development Considerations for Storage Applications in Windows 2000.* — Redmond, WA : Microsoft Corporation, 1999: http://www.microsoft.com/windows2000/docs/StorDev.doc.

- Статья базы знаний Microsoft KB237853 *Dynamic Disk Configuration Unavailable for Server Cluster Disk Resources*: `http://support.microsoft.com/default.aspx?scid=KB;EN-US;q237853`.
- Статья базы знаний Microsoft KB175761 *Dynamic vs. Basic Storage in Windows 2000*: `http://support.microsoft.com/default.aspx?scid=kb;en-us;Q175761`.
- Документ *Encrypting File System for Windows 2000*, июль 1999: `http://www.microsoft.com/windows2000/techinfo/howitworks/security/encrypt.asp`.
- Статья базы знаний Microsoft KB226545 *How Single Instance Storage Identifies Which Volumes to Manage*: `http://support.microsoft.com/default.aspx?scid=KB;en-us;Q226545`.
- Статья базы знаний Microsoft KB241201 *How To: Back Up the Recovery Agent Encrypting File System Private Key in Windows 2000*: `http://support.microsoft.com/support/kb/articles/Q241/2/01.asp`.
- Статья базы знаний Microsoft KB234048 *How Windows 2000 Assigns, Reserves, and Stores Drive Letters*: `http://support.microsoft.com/default.aspx?scid=KB;en-us;Q234048`.
- Документ к версии Windows 2000 Datacenter Server: `http://www.microsoft.com/windows2000/en/datacenter/help/default.asp?url=/windows2000/en/datacenter/help/sag_mscs2planning_28.htm`.
- Презентации Microsoft, относящиеся к системам хранения данных: `http://www.microsoft.com/winhec`.
- Файловая система NTFS 5.0: `http://www.ccs.neu.edu/home/lieber/com3336/f99/lectures/111/ntfs/ntfs5.ppt`.
- Статья базы знаний Microsoft KB262797 *Reparse Point Support in Windows 2000-Based Clusters*: `http://support.microsoft.com/default.aspx?scid=kb;en-us;Q262797`.
- Richter J., Cabrera L. F. *A File System for the 21st Century: Previewing the Windows NT 5.0 File System* // Microsoft Systems Journal. — November 1998: `http://www.microsoft.com/msj/defaultframe.asp?page=/msj/1198/ntfs/ntfs.htm\&nav=/msj/1198/newnav.htm`.
- Статья базы знаний Microsoft KB171052 *Software FT Sets Are Not Supported in Microsoft Cluster Server*: `http://support.microsoft.com/default.aspx?scid=kb;en-us;Q171052`.
- Solomon D. A., Russinovich M. *Inside Windows 2000*, 3rd Ed. — Redmond, WA : Microsoft Press, 2000.

- *StorNext File System: High-Performance File Sharing for SAN Environments*: `http://www.adic.com/ibeCCtpSctDspRte.jsp?minisite=10000\&respid=22372\&section=10121`.
- *Windows and GPT FAQ, Version 1.1.* — Web-узел программной разработки платформы Windows: `http://www.microsoft.com/hwdev/tech/storage/GPT_FAQ.asp`.

Глава 7

- Спецификация Bluefin и инициатива SMI ассоциации SNIA: `http://www.snia.org/tech_activities/SMI/bluefin`.
- Модель CIM, Web-узел ассоциации SNIA со ссылками на различные документы и презентации: `http://www.snia.org/tech_activities/cim`.
- HBA API, Web-узел ассоциации SNIA со ссылками на спецификации и программный код: `http://www.snia.org/tech_activities/hba_api`.
- Joshi N. *Windows 2000 Storage Management Overview*: `http://www.microsoft.com/Seminar/Includes/Seminar.asp?Url=/Seminar/1033/20000413Win2KStorageNJ1/Portal.xml`.
- *Removable Storage Management and Windows*: `http://www.microsoft.com/hwdev/archive/storage/RSM.asp`.
- Документ *Windows Management Instrumentation: Background and Overview*, ноябрь 1999: `http://www.microsoft.com/windows2000/techinfo/howitworks/management/wmioverview.asp`.
- Документ *Windows Management Instrumentation: Provider Programming*, декабрь 1999: `http://www.microsoft.com/windows2000/techinfo/howitworks/management/wmiprov.asp`.
- Документ *Windows Management Instrumentation Scripting*, январь 2000: `http://www.microsoft.com/windows2000/techinfo/howitworks/management/wmiscripts.asp`.
- Самоучитель по интерфейсу WMI: `http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/msdn-files/027/001/574/msdncompositedoc.xml`.

Глава 8

Технология InfiniBand

- Connor D. *Microsoft Cancels InfiniBand Development* // Network World Fusion. — July 2002: http://www.nwfusion.com/news/2002/0730msin.html.
- Торговая ассоциация InfiniBand: http://www.infinibandta.org.

Технология IP Storage

- Clark T. *IP SANs: A Guide to iSCSI, iFCP, and FCIP Protocols for Storage Area Networks.* — Boston : Addison-Wesley, 2002.
- Web-узел конференции IEEE: http://www.storageconference.org.
- Web-узел рабочей группы IETF IP Storage: http://www.ietf.org/html.charters/ips-charter.html.
- IP Storage, университет Карнеги–Меллон: http://www.ece.cmu.edu/~ips/index.html.
- Форум SNIA IP Storage: http://www.snia.org/tech_activities/ip_storage.

TCP Offload

- Статья базы знаний Microsoft KB260978 *Description of NDIS Features in Windows 2000*: http://support.microsoft.com/support/kb/articles/Q260/9/78.asp.
- *NDIS 5.0 Overview*: http://www.microsoft.com/hwdev/tech/network/ndis5.asp.
- *Windows Network Task Offload*: http://www.microsoft.com/hwdev/tech/network/taskoffload.asp.

Глава 9

- Edelmann S. *SANworks Secure Path: The Secret of Data Availability*: http://www.decus.de/slides/sy2001/vortraege_2504/2d07.pdf.
- Технология HP (Compaq) SecurePath: http://www.mcoecn.org/WhitePapers/COMPAQ-SECURE-PATH.PDF.
- Kammer K. R. *Dell EMC PowerPath: Improving Storage Availability*, август 2002: http://ftp.us.dell.com/app/3q02-Kam.pdf.
- Web-узел технологии RAIDWeb: http://www.raidweb.com/whatis.html.

Групповой ввод-вывод

- Edelmann S. *SANworks Secure Path: The Secret of Data Availability*: http://www.decus.de/slides/sy2001/vortraege_2504/2d07.pdf.
- *EMC PowerPath Enterprise Storage Software: Product Description Guide*: http://www.emc.com/products/product_pdfs/pdg/powerpath_pdg.pdf.
- Презентация Microsoft Enterprise Storage: http://www.microsoft.com/winhec/winhec2001.mspx.
- Статья базы знаний Microsoft KB293778 *Multiple-Path Software May Cause Disk Signature to Change*: http://support.microsoft.com/default.aspx?scid=KB;en-us;Q293778.
- Web-узел Microsoft Storage Technologies: http://www.microsoft.com/hwdev/tech/storage/default.asp.
- Документ *StorageWorks Secure Path, Version 3.0, for Windows NT: A High-Availability Solution for Intel Platforms Product Description*: http://www.compaq.com/products/storageworks/techdoc/storagemgtsoftware/AA-RL54A-TE.html.
- Документы по продуктам компании VERITAS для платформы Windows: http://www.veritas.com/us/partners/microsoft/whitepapers.html.

Глава 10

- *Development Considerations for Storage Applications in Windows 2000.* — Redmond, WA: Microsoft Corporation, 1999: http://www.microsoft.com/windows2000/docs/StorDev.doc.
- *Enterprise-Class Storage in Microsoft Windows 2000.* — Microsoft, март 2000: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnentdevgen/html/enterstor.asp.
- Службы хранилищ данных Windows 2000: http://microsoft.com/windows2000/technologies/storage/default.asp.
- Обзор характеристик Windows Server 2003: http://microsoft.com/windows.netserver/evaluation/features/featuresorterresults.aspx?Technology=Storage+Management.
- *Technical Overview of File Services.* — Microsoft : Windows Server 2003, июль 2002: http://www.microsoft.com/windows.netserver/docs/fileoverview.doc.

Предметный указатель

Научно-популярное издание

Дайлип Наик

Системы хранения данных в Windows

Литературный редактор *Т.П. Кайгородова*
Верстка *А.Н. Полинчик*
Художественные редакторы *В.Г. Павлютин*, *О.Л. Василенко*
Корректоры *З.В. Александрова*, *Л.А. Гордиенко*, *Л.В. Пустовойтова*

Издательский дом "Вильямс".
101509, Москва, ул. Лесная, д. 43, стр. 1.

Подписано в печать 03.12.2004. Формат 70×100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 34,8. Уч.-изд. л. 23,3.
Тираж 3000 экз. Заказ № 9033.

Отпечатано с диапозитивов в ФГУП "Печатный двор"
Министерства РФ по делам печати,
телерадиовещания и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.

"Книга Дайлипа Наика *Системы хранения данных в Windows* представляет собой превосходное издание по технологиям хранения данных, которое должно быть на рабочем столе как разработчиков, так и конечных пользователей подобных систем".

Том Кларк, начальник отдела маркетинга технических систем, компания Nishan Systems

Рынки Windows и корпоративных систем хранения данных постепенно объединяются. Начиная с серверов нижнего уровня, Windows шаг за шагом становится идеальной операционной платформой для критических приложений. Рынок корпоративных систем хранения данных охватывает не только промышленные системы, но и серверы среднего уровня. Благодаря интегрированным функциям хранения данных технологии Microsoft Windows постепенно приобретают всеобщее признание. Системные администраторы, программисты и менеджеры технических отделов благодаря этой книге получат великолепную возможность ознакомиться со всем потенциалом корпоративных систем хранения данных на базе Windows.

В книге *Системы хранения данных в Windows* впервые представлен всеобъемлющий обзор новых и постоянно развивающихся технологий хранения данных для систем компании Microsoft. Сначала приводится обзор индустрии хранения данных и архитектуры операционной системы Windows Server, включая подсистему ввода-вывода данных Windows NT. Далее обсуждаются все плюсы и минусы текущих функциональных возможностей Windows, будущих версий Windows Server и программных продуктов сторонних разработчиков.

В книге подробно описываются различные технологии хранения данных.

- Технология серверной дисковой памяти (Direct Attached Storage — DAS), включая новую программную модель драйверов Windows Storport
- Архитектура серверных хранилищ данных (Storage Area Networks — SAN) на базе интерфейса Fibre Channel
- Архитектура сетевых хранилищ данных (Network Attached Storage — NAS), включая стек сетевых драйверов Windows и обзор технологии CIFS
- Технологии резервирования и восстановления данных, в частности служба теневого копирования томов (Volume Shadow Copy) в Windows XP и Windows Server 2003
- Файловые системы и виртуализация дисков, в том числе детальный обзор NTFS и файловой системы кластеризации Windows
- Управление системами хранения данных, а также новая служба Windows Virtual Disk Service
- Технологии IP Storage и Infiniband
- Технологии обеспечения постоянного доступа к данным, включая зеркальные тома RAID и функции многопоточного ввода-вывода

Книга завершается рассмотрением функций хранения данных Windows NT в соответствии с хронологией их появления в операционных системах Windows NT, Windows 2000 и Windows Server 2003. Кроме того, описываются будущие возможности, которые могут быть реализованы в следующих версиях Windows Server. Таким образом, книга *Системы хранения данных в Windows* поможет профессионалам в области информационных технологий осознать главенствующую роль, которую серверы Windows в скором времени будут играть в сфере корпоративных систем хранения данных.

Дайлип Наик более 12 лет работает на различных должностях в компании Microsoft, в том числе системным инженером, менеджером проектов и техническим консультантом. К его заслугам относится написание кода протоколов CIFS/SMB и связанных с ними документов RFC, а также кода и документации для инсталляционного системного набора файловой системы Windows NT. Дайлип также принимал участие в разработке программной среды WMI (Windows Management Instrumentation) и различных функций управления/оптимизации (включая управление хранилищами данных) для Windows. Кроме того, Дайлип представляет компанию Microsoft в нескольких организациях по стандартизации промышленных систем.

ВИЛЬЯМС

Addison-Wesley
Pearson Education

www.williamspublishing.com
www.awprofessional.com

ISBN 5-8459-0746-2