ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»

А.Н. Губин
Ф. В. Филиппов

СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ

УЧЕБНОЕ ПОСОБИЕ

СПбГУТ)))

САНКТ-ПЕТЕРБУРГ
2015

УДК 004.43

Рецензент
кандидат технических наук, доцент кафедры робототехники и автоматизации производственных систем Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» *А. В. Шевченко*

Утверждено редакционно-издательским советом СПбГУТ
в качестве учебного пособия

Губин, А.Н.
Схемы хранения данных: учебное пособие / А.Н. Губин, Ф. В. Филиппов – СПб.: Изд-во СПбГУТ, 2015. - 76 с.

Рассматриваются основные принципы и технологии построения систем хранения данных.

Пособие предназначено для бакалавров и магистров специальности 230400 – Информационные системы и технологии, а также аспирантов и специалистов в области информационных технологий.

УДК 004.43

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Хранение информации является важнейшей составляющей современных информационных технологий. Рост количества производящих контент устройств обусловливает существенный (экспоненциальный) рост объемов как структурированных (данные организованные в определенные структуры и строго форматированные по содержанию), так и неструктурированных данных (веб-страницы, изображения, аудио/видео потоки, тексты и др.). Возможность обработки данных предполагает их хранение и обеспечение доступности для приложений, использующих эти данные. Причем, в последнее время все большую актуальность приобретает возможность хранения и обработки неструктурированных данных.

Данные представляют собой некоторую совокупность фактов представленных в формализованном виде.

Традиционные базы данных, как правило, ориентированы на выполнение жестко определенных операций по обработке данных. Возможности обработки нерегламентированных запросов в таких базах данных существенно ограничены.

В то же время, необходимость выполнения по отношению к системе хранения данных таких операций как

- формирование запросов произвольной формы;
- интегрирование данных из различных генерирующих контент систем;
- обработка больших объемов данных,

вызвало появление новых технологий организации баз данных, а именно технологии хранилищ данных.

В основе концепции хранилища данных лежат две основные идеи:

- интеграция разъединенных детализированных данных в едином хранилище;
- разделение наборов данных и приложений, используемых для оперативной обработки и применяемых для решения задач анализа.

Определение понятия "хранилище данных"(Data Warehouse) первым дал Уильям Г. Инмон в 2002 году. Он определил хранилище данных как "предметно-ориентированную, интегрированную, содержащую исторические данные, не разрушаемую совокупность данных, предназначенную для поддержки принятия управленческих решений".

Концептуально модель хранилища данных можно представить в виде схемы, представленной на рис. 1.

Данные из различных источников помещаются в хранилище данных, а описания этих данных в репозиторий метаданных. Конечный пользователь,

используя различные инструменты (средства визуализации, построения отчетов, статистической обработки и т.д.) и содержимое репозитория, анализирует данные в хранилище. Результатом его деятельности обычно является информация в виде готовых отчетов, найденных скрытых закономерностей, каких-либо прогнозов. Так как средства работы конечного пользователя с хранилищем данных могут быть самыми разнообразными, то теоретически их выбор не должен влиять на структуру и функции поддержания в актуальном состоянии хранилища данных.

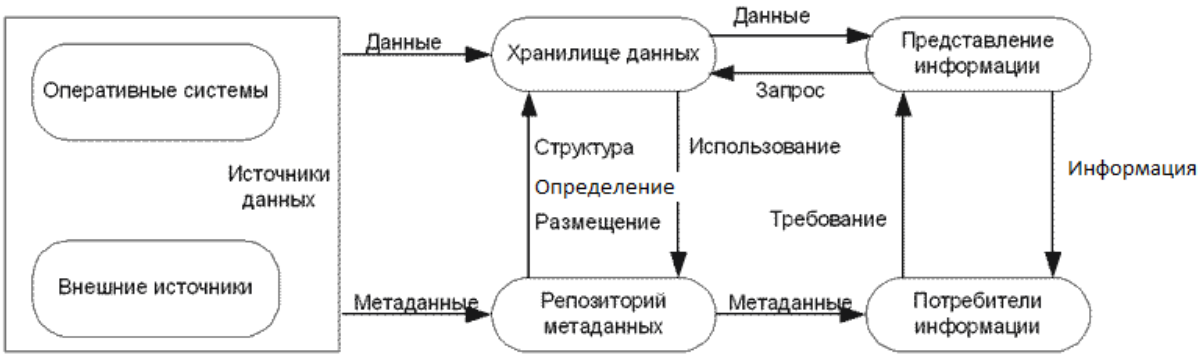


Рис. 1. Концептуальная модель хранилища данных

Структурированные и неструктурированные данные, представленные в смысловой форме, представляют собой информацию. То есть информация – это сведения и знания, извлекаемые из данных.

Метаданные представляют собой набор структурированных данных, представляющих собой характеристики хранимых сущностей для целей их идентификации, поиска, оценки, управления ими.

Система хранения данных (СХД) - это программно-аппаратное решение по организации надёжного хранения данных с использованием принципов построения хранилищ данных и предоставления к ним гарантированного доступа.

В учебном пособии рассмотрены основные вопросы реализации систем хранения данных.

Ограниченный объем пособия не позволяет достаточно подробно осветить все вопросы касающиеся всех важнейших проблем разработки и реализации СХД, поэтому для получения дополнительных сведений рекомендуется обратиться к соответствующим источникам, список которых включает 3 наименования. При работе над пособием авторы неоднократно осуществляли заимствование из этих источников определений, примеров и методов изложения, без отдельных ссылок в тексте.

1. ОСНОВНЫЕ КОМПОНЕНТЫ СХД

1.1. СХД – основные типы

В случае отдельного ПК под системой хранения данных можно понимать внутренний жесткий диск или систему дисков (RAID массив). Если же речь заходит о системах хранения данных масштаба предприятий и корпораций, то традиционно можно выделить три технологии организации систем хранения данных:

- Direct Attached Storage (DAS);
- Network Attach Storage (NAS);
- Storage Area Network (SAN).

Устройства DAS (Direct Attached Storage) – решение, когда устройство для хранения данных подключено непосредственно к серверу, или к рабочей станции, как правило, через интерфейс по протоколу SAS.

Архитектура системы хранения DAS представлена на рис.2.

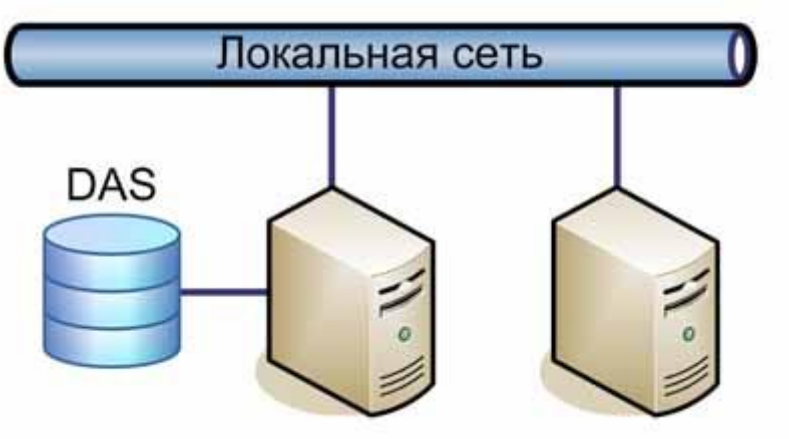


Рис. 2. Архитектура СХД Direct Attached Storage

К основным преимуществам DAS систем можно отнести их низкую стоимость (в сравнении с другими решениями СХД), простоту развертывания и администрирования, а также высокую скорость обмена данными между системой хранения и сервером. Этим объясняется большая популярность их использования в сегменте малых офисов, хостинг-провайдеров и небольших корпоративных сетей.

По сути, данный тип СХД представляет собой дисковую полку (корзину) с жесткими дисками, вынесенную за пределы сервера.

Дисковая полка - это внешний программно-аппаратный массив жестких дисков с набором дополнительных функций, подключаемый к хосту через

высокоскоростную магистраль передачи данных (SCSI U320 Мбит / сек или оптический интерфейс).

К недостаткам DAS-систем следует отнести неоптимальную утилизацию ресурсов, поскольку каждая DAS система требует подключения выделенного сервера и позволяет подключить максимум 2 сервера к дисковой полке в определенной конфигурации. Кроме того, СХД DAS обладает низкой надежностью, так как при выходе из строя сервера, к которому подключено хранилище данных, данные становятся недоступными.

Устройства NAS (Network Attached Storage) – отдельно стоящая интегрированная дисковая система, по-сути, NAS-сервер, со своей специализированной ОС и набором полезных функций быстрого запуска системы и обеспечения доступа к файлам.

Архитектура системы хранения NAS представлена на рис.3.

Рис. 3. Архитектура СХД Network Attached Storage

Технология NAS (сетевые подсистемы хранения данных) развивается как альтернатива универсальным серверам, реализующим множество функций (печати, приложений, факс сервер, электронная почта и т.п.). В отличие от них NAS-устройства исполняют только одну функцию - файловый сервер.

NAS подключаются к ЛВС и обеспечивают доступ к данным для неограниченного количества гетерогенных клиентов (клиентов с различными ОС) или других серверов. В настоящее время практически все NAS устройства ориентированы на использование в сетях Ethernet (Fast Ethernet, Gigabit Ethernet) на основе протоколов TCP/IP. Доступ к устройствам NAS производится с помощью специальных протоколов доступа к файлам. Наиболее распространенными протоколами файлового доступа являются протоколы CIFS, NFS и DAFS. Внутри подобных серверов стоят специализированные ОС, такие как MS Windows Storage Server.

К достоинствам NAS следует отнести:

• Дешевизну и доступность ресурсов СХД не только для отдельных серверов, но и для любых компьютеров предприятия или корпорации.

• Простоту коллективного использования ресурсов.

• Простоту развёртывания и администрирования

• Универсальность для клиентов (один сервер может обслуживать клиентов MS, Novell, Mac, Unix)

Недостатками NAS являются:

• Реализация доступа к данным через протоколы “сетевых файловых систем” что медленнее, чем доступ к локальному диску.

• Отсутствие возможности (для большинства недорогих NAS-серверов) обеспечить скоростной и гибкий метод доступа к данным на уровне блоков, а не на уровне файлов.

Storage Area Network (SAN) –это специальная выделенная сеть, объединяющая устройства хранения данных с серверами приложений, обычно строится на основе протокола Fibre Channel или протокола iSCSI.

Рис. 4. Архитектура СХД Storage Area Network

В отличие от NAS, SAN не имеет понятия о файлах: файловые операции выполняются на подключенных к SAN серверах. SAN оперирует блоками, как некий большой жесткий диск. Идеальный результат работы SAN - возможность доступа любого сервера под любой операционной системой к лю-

бой части дисковой емкости, находящейся в SAN. Оконечные элементы SAN - это серверы приложений и системы хранения данных (дисковые массивы, ленточные библиотеки и т. п.). А между ними, как и в обычной сети, находятся адаптеры, коммутаторы, мосты, концентраторы.

ISCSI является более «дружелюбным» протоколом, поскольку он основан на использовании стандартной инфраструктуры Ethernet – сетевых карт, коммутаторов, кабелей. Более того, именно системы хранения данных на базе iSCSI являются наиболее популярными для виртуализированных серверов, в силу простоты настройки протокола. Архитектура системы хранения SAN представлена на рис.4.

К достоинствам SAN относятся:

- Высокая надёжность доступа к данным, находящимся на внешних системах хранения. Независимость топологии SAN от используемых СХД и серверов.
- Централизованное хранение данных (надёжность, безопасность).
- Удобное централизованное управление коммутацией и данными.
- Перенос интенсивного трафика ввода-вывода в отдельную сеть (разгружается LAN).
- Высокое быстродействие (обеспечивается скоростной и гибкий метод доступа к данным на уровне блоков).
- Масштабируемость и гибкость логической структуры SAN
- Возможность организации резервных, удаленных СХД и удаленной системы бэкапа и восстановления данных.
- Возможность строить отказоустойчивые кластерные решения без дополнительных затрат на базе имеющейся SAN.

К недостаткам SAN относятся:

- Более высокая стоимость
- Сложность в настройке FC-систем
- Необходимость сертификации специалистов по FC-сетям (iSCSI является более простым протоколом)
- Более жесткие требования к совместимости компонентов SAN.

1.2. Хранение данных на магнитных дисках

Устройство хранения данных – важнейший компонент в среде систем хранения данных. Устройства хранения используют магнитные, оптические или полупроводниковые средства запоминания данных. Жесткие диски, ленты и дискеты используют магнитные принципы запоминания информации. CD-ROM, CD-RW – пример устройств, использующих оптические средства

хранения данных, съемные карты флэш-памяти – пример использования полупроводниковых средств хранения данных.

Наиболее популярными устройствами для хранения данных, используемым в современных компьютерах для хранения и доступа к данным средствами высокопроизводительных приложений в режиме реального времени, являются дисковые устройства.

Дисковое устройство использует для записи/чтения данных быстровращающийся плоский диск, покрытый магнитным материалом. Данные записываются на диск и считываются с диска через головку чтения/записи. Несколько собранных вместе магнитных дисков с блоком головок чтения/записи и устройством управления (контроллером) образуют устройство, которое получило название жесткого диска (рис.5).

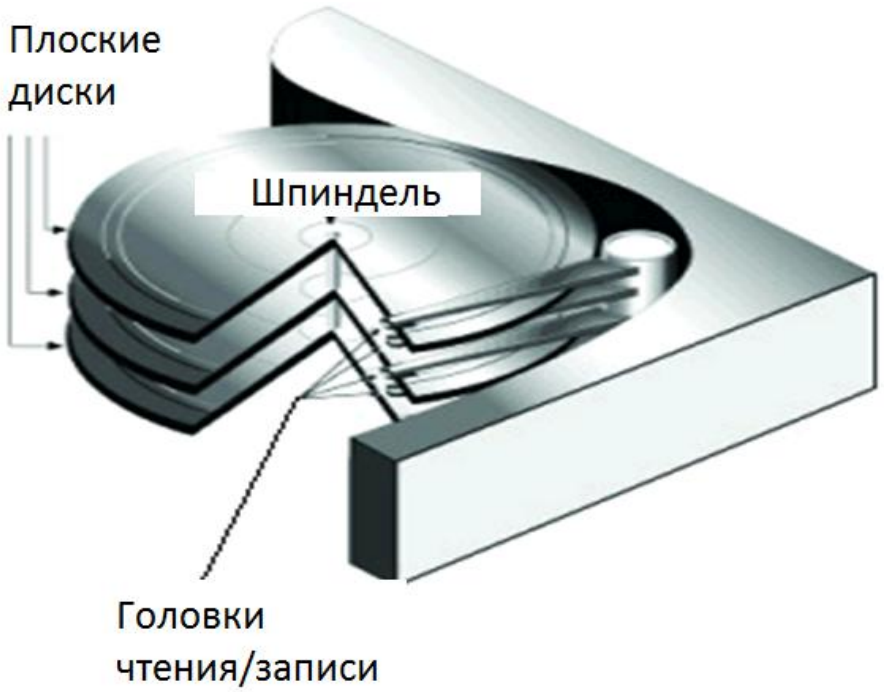


Рис. 5. Общая структура жесткого диска

На магнитных дисках (МД) информация записывается на концентрических дорожках, разделенных на секторы. Дорожка образуется как намагниченная узкая область поверхности МД (рис. 6).

Каждая дорожка делится на секторы. Сектор – дуга дорожки с фиксированным угловым размером.

Начальная разметка диска на дорожки и секторы производится на заводе изготовителе с помощью программы форматирования на низком уровне.

Совокупность дорожек, одновременно находящихся под блоком головок считывания, называется цилиндром.

Информационная емкость одного сектора равна 512 байт (на любой дорожке).

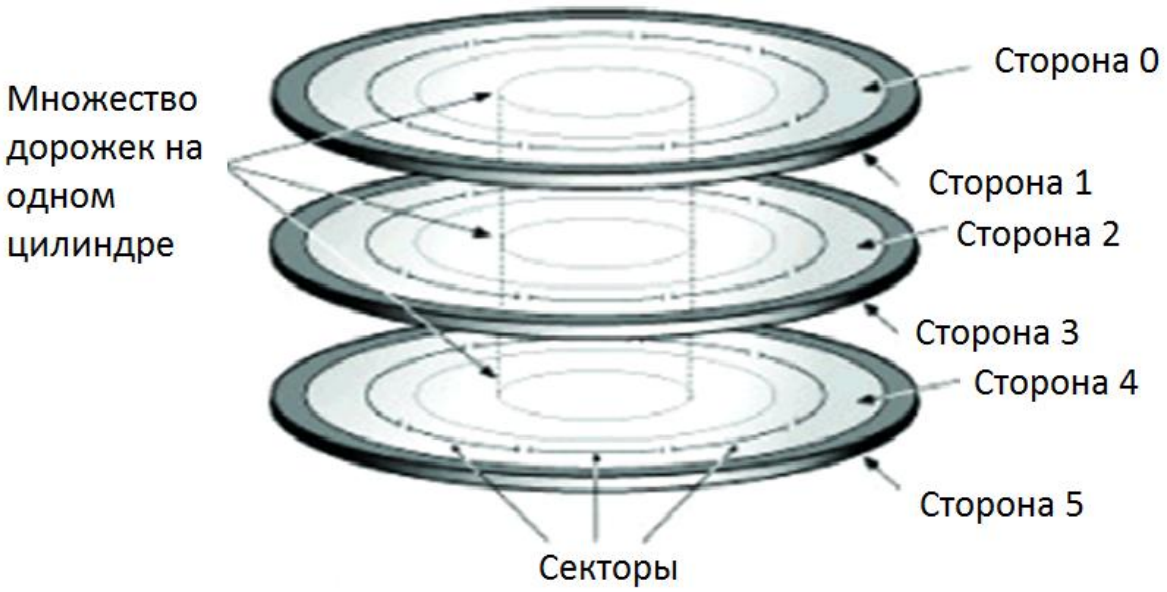


Рис. 6. Общая структура организации хранения информации на жестком диске

Вначале количество секторов на дорожках диска было одинаковым и следовательно плотность записи на них разная: чем ближе к центру, тем выше плотность записи (рис.7).

Рис. 7. Общая структура магнитного диска без зон

Пространство магнитного диска в этом случае используется неэффективно. Для повышения эффективности использования дискового пространства используют зонную запись секторов, которая предусматривает расположение секторов по зонам и выравнивание плотности записи информации по магнитным дорожкам (рис.8).

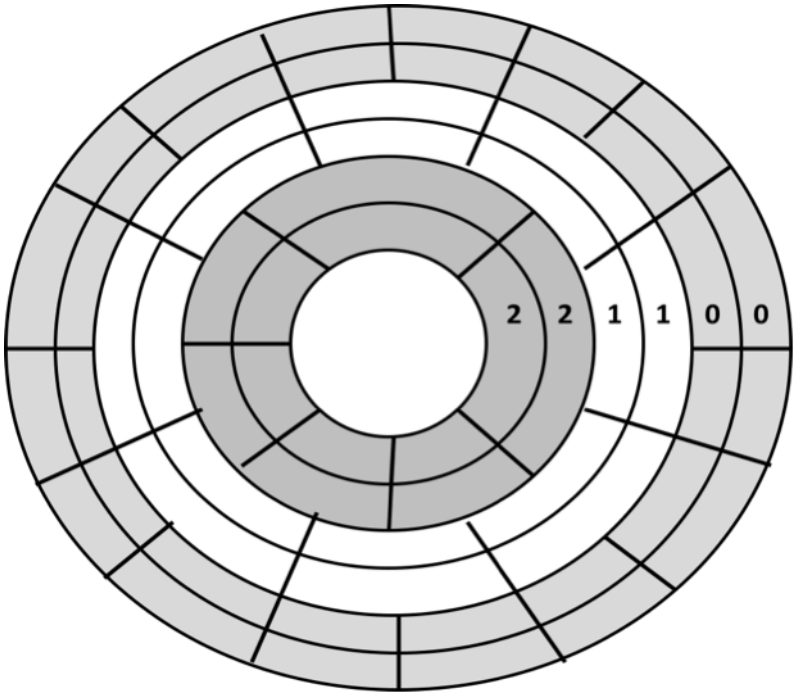


Рис. 8. Общая структура магнитного диска с зонами

Зоны нумеруются, начиная с крайней на диске, которая получает нулевой номер. Каждой зоне присваивается соответствующее количество секторов на магнитной дорожке. Все дорожки одной зоны имеют одинаковое количество секторов.

Каждый сектор состоит из двух частей. В первой части (по направлению вращения диска) записывается адрес сектора и дополнительная служебная информация (16 байт). Информационная часть сектора содержит фрагменты файлов и всегда содержит 512 байт (рис.9).

Адрес сектора – это три числа: < номер головки> <номер дорожки> <номер сектора> (ГДС).

Современные магнитные диски используют адресацию логических блоков (Logical block addressing -LBA).

Механизм LBA предусматривает адресацию и доступ к блоку данных на магнитном диске, когда контроллеру нет необходимости учитывать геометрию самого жесткого диска (количество цилиндров, сторон, секторов на цилиндре). При этом каждый блок, адресуемый на жёстком диске, имеет

свой номер, целое число, начиная с нуля (то есть первый блок LBA=0, второй LBA=1, ...).

Рис. 9. Общая структура сектора на магнитном диске

Однако для осуществления физического доступа к любому блоку информации необходимо преобразовать номер блока в адрес сектора (ГДС). Преобразование осуществляется в соответствии со следующими выражениями:

$$\mathbf{Г = [(LBA-(C-1))/С_{кол.}]mod\ Г_{кол.},}$$

$$\mathbf{Д=(LBA-(C-1)-ГС_{кол.})/С_{кол.}\ Г_{кол.},}$$

$$\mathbf{C=(LBA\ mod\ С_{кол.})+1\ ,}$$

где **Г**- номер головки, **Д**-номер дорожки, **С**- номер сектора, $\mathbf{С_{кол.}}$ - количество секторов на дорожке, $\mathbf{Г_{кол.}}$ –количество головок, **mod** - операция взятия остатка от деления.

Обратное преобразование адреса сектора в номер блока производится следующим образом

$$\mathbf{LBA= (Д\ Г_{кол.}+Г)\ С_{кол.} + С-1.}$$

После форматирования МД на низком уровне диск становится доступным для записи и считывания информации. Такой доступ с использованием ГДС называется адресным.

Для обеспечения доступа пользователя к информации, записанной на МД в виде файлов, используется файловый доступ к МД. При файловом доступе пользователю достаточно знать только имя файла (его физическое расположение на МД пользователю неизвестно).

Преобразование адресного доступа в файловый доступ обеспечивает файловая система (рис.10).

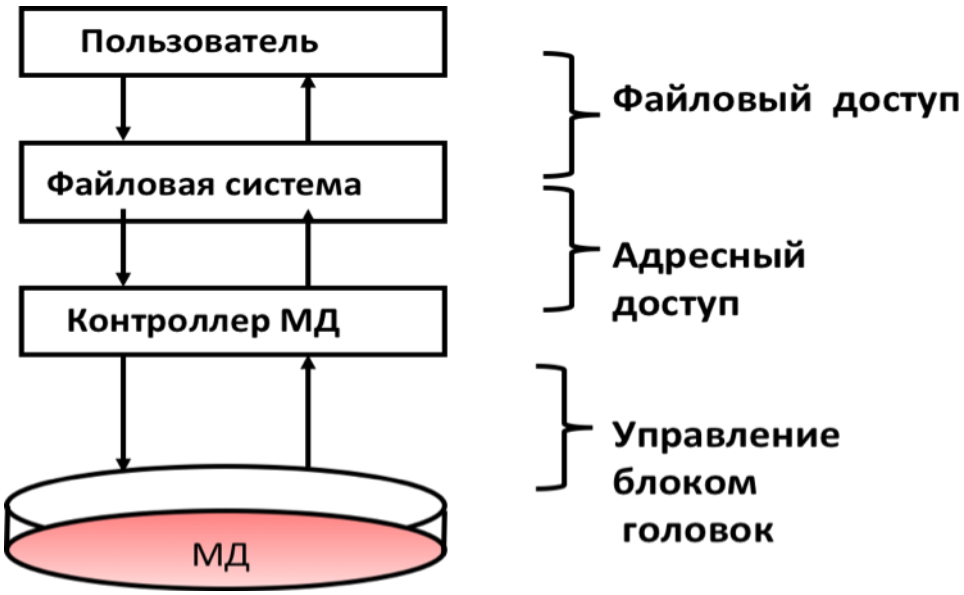


Рис. 10. Общая структура сектора на магнитном диске

Файловая система появляется на МД в результате выполнения операций высокоуровнего форматирования магнитного диска.

При выполнении форматирования диска появляется новый параметр, который требуется задать или выбрать – это размер **кластера.**

Кластером называется группа расположенных подряд секторов (рис.11).

Объем кластера определяется количеством входящих в него секторов (определяется при форматировании) и может быть:

1(0,5К); 2(1К); 4(2К); 8(4К); 16(8К); 32(16К); 64(32К).

Все кластеры нумеруются в порядке натурального ряда чисел (на диске миллионы кластеров). Каждому номеру кластера соответствует адрес его начального сектора (Г-Д-С).

Независимо от своей природы и содержания все файлы представляют собой цепочки связанных кластеров.

В реальных файловых системах набор ссылок на кластеры, которые составляют файл, обычно отрывается от элементов хранения информации и помещается в отдельную область диска в виде управляющей таблицы.

Такая таблица называется таблицей размещения файлов (File Allocation Table – FAT).

FAT состоит из такого количества ячеек – сколько кластеров размещено на диске.

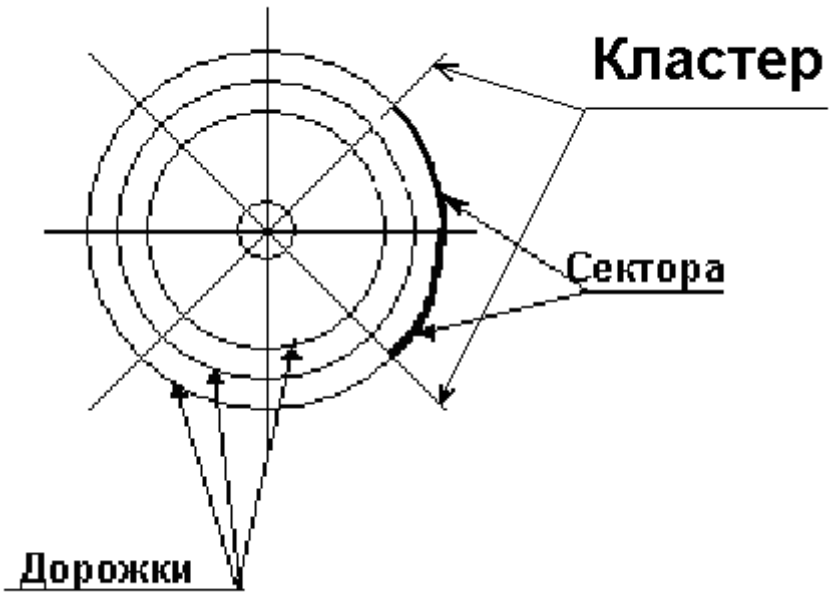


Рис. 11. Общая структура кластера на магнитном диске

На магнитном диске кроме таблицы FAT помещают таблицу, содержащую список имен файлов и точки входа этих файлов. Такая таблица называется директорией (каталогом, папкой).

Основное значение каталога – указание начальных кластеров файлов (точек входа), их связь с именами файлов и определение атрибутов файлов.

1.3. Расчет времени считывания данных с магнитного диска

Рассмотрим расчет времени считывания файлов с магнитного диска с FAT –файловой системой при следующих допущениях:

- временем поиска имени файла в каталоге и временем построения цепочки кластеров файла пренебрежем;
- время передачи данных по шине не учитывается;

- время перехода головки на другую дорожку определяется как время перехода на одну дорожку, умноженное на число пересекаемых дорожек.

Рассмотрим учебный магнитный диск с параметрами, указанными в табл.1.

Таблица 1. Параметры магнитного диска.

Наименование параметра	Обозначение параметра	Значение параметра
Число поверхностей	*Ns*	1
Число дорожек (цилиндров)	*Ntrk*	4
Число секторов в дорожке	*Nsect*	20
Число секторов в кластере	*Nscl*	4
Объём сектора в байтах	*Vsect*	512
Время перемещения головки считывания/записи на одну дорожку в мс	*T1*	4
Количество оборотов в мин	*Nob*	3600

На этом диске в корневом каталоге хранятся 5 файлов, объём которых и начальный кластер каждого файла (точка входа) заданы в табл.2.

Таблица 2. Параметры хранящихся на диске файлов

Имя файла	Объём в байтах (Vf)	Начальный кластер файла
A	1500	12
B	4800	5
C	450	6
D	7000	1
E	6500	10

Составим таблицу распределения файлов (FAT) для заданных выше условий. Для этого определим объемы файлов в кластерах. Объем кластера для заданных условий определяется как

$$Vcl = Vsect \bullet Nscl = 512 \bullet 4 = 2048 \text{ байт}$$

или 2 килобайта. Соответственно объемы файлов в кластерах определится как как целая часть следующего выражения

$$Vfcl = Vf / Vcl + 1$$

Результаты расчетов представлены в табл. 3.

Таблица 3. Объем хранящихся на диске файлов в кластерах

Имя файла	Объём в кластерах (Vfcl)
A	1
B	3
C	1
D	4
E	4

Общее количество позиций в таблице распределения файлов определяется емкостью диска в кластерах. Для решаемой задачи это значение можно рассчитать как

Ndcl = Ntrk • Nsect / Nscl = 4 • 20 / 4 = 20

Таблица расположения файлов для заданного диска, с учетом объема каждого из файлов и их точек входа будет иметь следующий вид.

Таблица 4. Таблица расположения файлов на диске

№ кластера										10
№ ссылки				eof		eof		eof		11
№ кластера	1	2	3	4	5	6	7	8	9	20
№ ссылки	3	eof	4	eof						0

В таблице 4 во второй строке указан или номер следующего кластера входящего в состав файла, или признак конца файла - eof (последний входящий в состав файла кластер), или признак свободного кластера -0 (кластер не входит в состав файла).

Из таблицы 4 можно получить информацию о цепочках кластеров составляющих файлы:

A = 12,
B = 5 → 7→ 8,
C = 6,
D = 1→2→3→4,
E= 10→11→13→14.

Следующим этапом решения поставленной задачи является построение карты диска, которая отражает расположение кластеров по дорожкам диска. Количество дорожек задано и равно 4, общий объем диска в кластерах равен 20, следовательно, на каждой дорожке диска расположено 5 кластеров.

Карта диска будет иметь следующий вид (табл.5).

Таблица 5. Карта диска

№ дорожки	Кластеры				
1	1 D1	2 D2	3 D3	4 D4	5 B1
2	6 C	7 B2	8 B3	9 0	10 E1
3	11 E2	12 A	13 E3	14 E4	15 0
4	16 0	17 0	18 0	19 0	20 0

Каждая клетка карты диска содержит номер кластера диска (число в левой части клетки) и обозначение входящего в состав файла кластера с указанием его номера (B3- третий кластер файла B).

Определим время, за которое диск делает один оборот

$$Tob = 60 \bullet 1000/Nob = 60000/3600 = 16{,}7 \text{ мс},$$

где 60 – количество секунд в минуте, 1000 количество миллисекунд в секунде.

За это время под головкой считывания/записи диска проходят все кластеры одной дорожки, для нашего случая - это четыре кластера, следовательно, время считывания одного кластера составит

$$Tcl = Tob/4 = 16{,}7/4 = 4{,}18 \text{ мс}$$

Для считывания первого кластера файла головка должна быть установлена на дорожку, где расположен этот кластер. Номер дорожки определяется по карте диска.

Время перемещения головки на одну дорожку задано, значит, легко можно определить время позиционирования головки на дорожку как:

$$Tpoz = T1 \bullet N = 3 \bullet N \text{ мс},$$

где $T1 = 4$ *msec* - время перемещения головки на одну дорожку (задано),

N - число переходов с дорожки на дорожку при перемещении головки считывания/записи.

Когда головка установится на нужной дорожке, необходимо потратить какое-то время на ожидание появления под головкой нужного кластер.

Поскольку этот процесс случайный, то, предполагая равномерный закон распределения, можно принять за время ожидания величину полуоборота диска, то есть:

Toj=Tob/2=8.35 мс.

Определим время считывания файла А. Файл занимает один кластер, который расположен на третьей дорожке.

Ta = Tpoz + Toj + Tcl = 3•4 + 8,35 + 4,18 = 24,53 мс.

При расчете времени считывания файла В, необходимо учесть, что файл состоит из трех кластеров, которые расположены на первой и второй дорожках диска.

Время считывания первого кластера файла В определим следующим образом

Tb1 = Tpoz + Toj + Tcl = 1•4 + 8,35 + 4,18 = 16,53 мс.

Для считывания второго кластера файла В необходимо переместить головку считывания с первой на вторую дорожку, учесть время ожидания прохождения нужного кластера под головкой считывания и считать кластер.

Tb2 = Tpoz + Toj + Tcl = 1•4 + 8,35 + 4,18 = 16,53 мс.

Для считывания третьего кластера файла В перемещения на головки считывания на другую дорожку не требуется, не требуется и ожидания кластера, так как кластер b3 располагается на той же дорожке сразу за кластером b2, следовательно

Tb3 = Tcl = 4,18 мс.

Общее время считывания файла В определится как сумма времен считывания всех кластеров, входящих в состав этого файла.

Tb=Tb1+Tb2+Tb3=16,53+16,53+4,18=37,24 мс.

Аналогично рассчитывается время считывания данных для остальных файлов.

Анализ результатов расчетов показывает, что основные затраты времени при считывании информации с магнитных дисков определяются временем позиционирования головок на требуемую дорожку диска и временем ожидания прохода нужного кластера под головкой считывания.

1.4. Производительность дисковых систем

Для анализа производительности дисковой системы целесообразно представить ее в виде черного ящика на входе, которого формируется очередь из запросов на выполнение операций по вводу/выводу информации, а на выходе – поток обработанных запросов. Источником запросов на ввод/вывод информации является приложение, решающее основную задачу обработки данных (рис. 12).

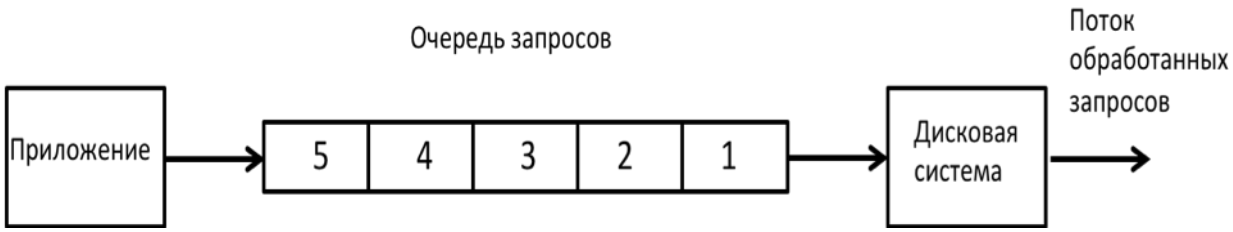


Рис. 12. Обработка запросов дисковой системой

Определим основные показатели производительности дисковой системы.

Запросы на выполнение операций ввода/вывода поступают на вход дисковой системы со скоростью работы приложения. Эта скорость называется частотой поступления.

Общее количество запросов в очереди и на обработке в дисковой системе - **N** определяется частотой поступления - a и средним временем отклика системы - R (общее время от прибытия запроса в очередь и до отправки обработанного запроса) в соответствии со следующим выражением

$$N = a \cdot \mathrm{R}$$

При этом уровень загрузки дисковой системы - U определится как

$$U = a \cdot R_S,$$

где R_S - среднее время обслуживания запроса на ввод/вывод информации.

Уровень загрузки системы может рассматриваться как коэффициент использования дисковой системы, его значение варьируется от 0 до 1.

Если рассчитать среднее время между запросами как

$$R_a = \frac{1}{a},$$

Тогда уровень загрузки определится как

$$U = \frac{R_s}{R_a},$$

Среднюю скорость отклика системы можно определить как разницу между скоростью обслуживания запросов и скоростью поступления этих запросов в очередь.

Тогда среднее время отклика можно определить как величину обратную средней скорости отклика системы.

$$R = 1/(\frac{1}{R_S} - \frac{\mathbf{1}}{R_a})$$

или

$$R = 1/(\frac{1}{R_S} - a)$$

и далее

$$R = R_S/(\mathbf{1} - aR_S)$$

и

$$R = R_S/(\mathbf{1} - U)$$

Если загрузка дисковой системы стремится к 1 (дисковая система работает на максимуме загрузки), то время отклика стремится к бесконечности.

Количество запросов в очереди (средний размер очереди) можно определить как

$$N_Q = N - U$$

После преобразований N_Q определится как

$$N_Q = U^2(1 - U)$$

Достаточно важным параметром дисковой системы является значение времени, которое затрачивается запросом в очереди на обслуживание.

Это время определяется как разность среднего времени отклика системы и времени затраченном на обработку запроса. То есть,

$$R_Q = R_S/(1 - U) - R_S$$

или после преобразований

$$R_Q = UR$$

Рассмотрим работу дисковой системы, когда на ее вход поступают запросы на обслуживание со скоростью a = 100 запросов в секунду, а время обслуживания запроса составляет $R_S = 8$ миллисекунд.

Определим такие параметры производительности дисковой системы как коэффициент загрузки, суммарное время отклика на запрос, средний размер очереди, общее время проведенное запросом в очереди.

Время между запросами можно определить как

$$R_a = \frac{1}{a} = 10 \text{ мс}.$$

Коэффициент использования дисковой системы вычисляется следующим образом

$$U = \frac{R_s}{R_a} = \frac{8}{10} = 0{,}8$$

или 80% .

Определим время отклика дисковой системы

$$R = \frac{R_S}{1 - U} = \frac{8}{1 - 0{,}8} = 40 \text{ мс.}$$

Средний размер очереди составит

$$N_Q = U^2(1 - U) = 0{,}8^2(1 - 0{,}8) = 3{,}2.$$

Время, проведенное запросом в очереди

$$R_Q = UR = 0{,}8 \cdot 40 = 32 \text{ мс.}$$

Чтобы оценить динамику зависимости показателей функционирования дисковой системы от ее параметров увеличим мощность дисковой системы вдвое, то есть, предположим, что

$$R_S = 4 \text{ мс}.$$

Тогда

$$U = \frac{R_S}{R_a} = \frac{4}{10} = 0{,}4,$$

$$R = \frac{R_S}{1-U} = \frac{4}{1-0{,}4} = 6{,}67 \text{ мс},$$

$$N_Q = U^2(1-U) = 0{,}4^2(1-0{,}4) = 0{,}26.$$

$$R_Q = UR = 0{,}4 \cdot 6{,}67 = 2{,}67 \text{ мс.}$$

Очевидно, что при увеличении мощности дисковой системы (уменьшение времени обслуживания системы вдвое) существенно уменьшается время отклика системы (почти в шесть раз).

Исследования подобных систем показывают, что связь между уровнем загрузки системы и временем отклика имеет нелинейный характер и может быть представлена следующим графиком (рис. 13).

При загрузке дисковой системы выше, чем $U = 0{,}7$ время отклика дисковой системы начинает увеличиваться в геометрической прогрессии относительно загрузки системы.

То есть параметры дисковой системы при реализации СХД следует выбирать так, чтобы их загрузка (коэффициент использования) не превышал 0,7.

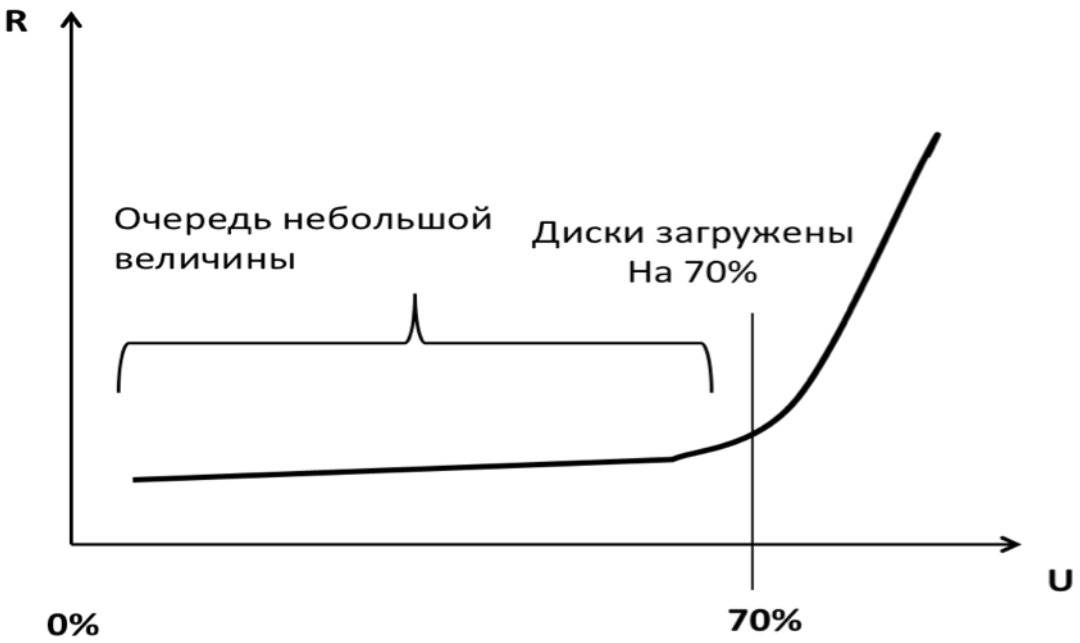


Рис. 13. Зависимость времени отклика от загрузки дисковой системы

1.5. Структура дисковых систем

Структура используемых дисковой системы определяется результатами анализа потребностей приложения.

Объем запоминающего устройства (емкость магнитного диска) достаточно легко рассчитывается исходя из объема подлежащей хранению информации, количества файловых систем, компонентов баз данных и других компонентов ИУС.

Требуемые значения временных характеристик дисковой системы также определяется на основании потребностями приложения.

Важнейшей характеристикой работы дискового устройства является значение IOPS (Input/Output Operations Per Second) – количество операций ввода/вывода в секунду.

Время обслуживания запроса приложения на запись или на чтение информации (*Ra)* можно определить как

$$R_a = E + L + X, \text{ где}$$

E – среднее время поиска блока информации,

L – среднее время задержки (время ожидания прохода блока информации под головкой)

X – среднее время передачи данных по внутренним шинам дисковой системы.

Рассмотрим работу дисковой системы, которая обладает следующими характеристиками.

E =5 мс.

Скорость вращения диска равна 15000 об/мин.

Средняя скорость передачи данных по внутренним шинам диска – 40 Мб/с.

Объем передаваемых блоков информации составляет 32 Кб.

Заданная скорость вращения диска позволяет определить среднее время ожидания нужного блока данных как время, необходимое для половины оборота диска

$$L = \frac{0{,}5 \ 60 \ 1000}{15\,000} = 2 \text{ мс}$$

Среднее время передачи блока данных можно определить как

$$X = \frac{32 \text{ Кб}}{40\,000 \text{Кб}/c} = 0{,}8 \text{ мс}$$

Тогда максимальное количество операций ввода/вывода в одну секунду при принятых характеристиках составит

$$IOPS = \frac{1}{R_a} = \frac{1}{E + L + S} = \frac{\mathbf{1}}{(5 + 2 + 0{,}8)\text{мс}} = 128 \ 1/c$$

Изменяя объем блоков, используемых при выполнении операций ввода/вывода, можно изменять значение IOPS для рассматриваемого диска.

В табл. 6 представлены результаты расчетов IOPS для различных значений объема обрабатываемых блоков

Таблица 6. Результаты расчета IOPS

Объем блока (Кб)	Время обслуживания диска (R_a, мс)	IOPS (1/с)
4	7,1	140
8	7,2	139
16	7,4	135
32	7,8	128
64	8,6	116

Общее количество дисков (N), необходимых для приложения, рассчитывается следующим образом

$$N = Max\,(C, I),$$ где

C – количество дисков необходимых для выполнения требований по объему запоминающего устройства,

I - количество дисков необходимых для достижения требуемого значения IOPS.

Пример. Требования по объему запоминающего устройства для приложения составляет 1,46 Тб.

Максимальная работоспособность приложения составляет 9000 операций IOPS.

Поставщик оборудования предлагает диски со следующими характеристиками:

- емкость диска 146 Гб;
- скорость вращения диска 15000 об/мин;
- при загрузке 70% диск обеспечивает 180 операций IOPS.

Необходимо определить количество приобретаемых дисков.

Чтобы удовлетворить требования по объему запоминающего устройства потребуется

$$C = 1{,}46\ \text{Тб} / 146\ \text{Гб} = 10$$

дисков.

Для выполнения 9000 операций IOPS потребуется

$$I = (9000)/(180) = 50$$

дисков.

Таким образом, количество дисков, которое удовлетворяет всем запросам приложения составляет

$$N = Max\,(10{,}50) = 50.$$

Рассмотрим еще один пример определения параметров функционирования дисковой системы в составе ИСУ.

Пример. Запросы в системе обрабатываются со скоростью 80 операций IOPS. Среднее время обслуживания запросов R_S (обычно задается поставщиком)

$R_S = 6$ мс.

Необходимо определить:
- загруженность контроллера ввода/вывода дисковой системы (U);
- суммарное время отклика на запрос (R);
- средний размер очереди(N_Q);
- суммарное время, проведенное запросом в очереди.

Время, затраченное диском на обработку запроса определяется как

$$R_a = \frac{\mathbf{1}}{IOPS} = \frac{1}{80\ (\frac{1}{c})} = 12{,}5\ \text{мс}\,.$$

Загруженность контроллера составит

$$U = \frac{R_S}{R_a} = \frac{6}{\mathbf{12{,}5}} = 0{,}4 = 40\%\,.$$

Ссуммарное время отклика на запрос определится как

$$R = \frac{R_S}{(1-\mathrm{U})} = \frac{6}{0{,}52} = 11{,}53\ \text{мс}\,.$$

Средний размер очереди рассчитывается как

$$N_Q = \frac{U^2}{(1-\mathrm{U})} = \frac{0{,}23}{0{,}52} = 0{,}44\,.$$

Суммарное время, проведенное запросом в очереди составит

$$R_Q = UR = 0{,}48 \ \bullet 11{,}53 = 6\ \text{мс}\,.$$

Вопросы для самопроверки

1. К какому типу систем хранения данных относятся системы, предусматривающие непосредственное подключение дисковых массивов к серверу?
2. Чем отличается архитектура системы хранения данных типа DAS от архитектуры системы хранения данных типа NAS?
3. Каким образом дисковые массивы подключаются к серверам в системах хранения данных типа SAN?
4. На каком уровне форматирования магнитных дисков обеспечивается разметка дисковых поверхностей на дорожки и сектора?
5. Чем отличается структура магнитного диска с зонами от структуры магнитного диска без зон?

6. Какая из подсистем операционной системы обеспечивает преобразование файлового доступа к данным к адресному доступу?
7. На каком уровне форматирования магнитного диска определяется емкость кластера?
8. Как определить время, затрачиваемое дисковой системой на обработку одного запроса, если параметр IOPS=50?

2. ТЕХНОЛОГИЯ RAID-массивов

Неотъемлемой частью современных вычислительных систем являются жесткие магнитные диски (HDD). Надежность работы HDD характеризуется среднем временем безотказной работы (это наработка на отказ). Современные информационные центры используют в своих системах хранения данных тысячи жестких дисков. Чем больше дисков, тем больше вероятность выхода из строя одного из используемых дисков. Так если СХД использует 100 дисков, то при среднем времени безотказной работы для каждого диска 750 000 часов, средний период безотказной работы всего массива дисков составит 7 500 часов (750 000/100). То есть в среднем через 7 500 часов (через каждые 10-11 месяцев) один из жестких дисков системы выходит из строя.

Кроме того, быстродействие HDD существенно меньше быстродействии я остальных компонентов СХД.

Технология RAID-массивов позволяет в некоторой степени преодолеть указанные недостатки. RAID-массив (redundant array of independent disks — избыточный массив независимых дисков) - это технология позволяющая использовать многочисленные диски как целое устройство, защищая данные от выхода из строя единичных дисков и повышая производительность обработки запросов на ввод/вывод информации.

2.1. Компоненты RAID-массивов

Существует два типа реализации RAID: на аппаратном и программном уровнях.

Программная реализация RAID использует для управления информацией функции, реализованные на основе программ. Программное обеспечение RAID-технологий разрабатывается на уровне операционных систем без использования аппаратных контроллеров для управления RAID-массивов.

Использование программного RAID более рентабельно: не надо тратиться на приобретение дополнительных аппаратных средств, эксплуатация программных средств более просто и менее затратное по сравнению аппа-

ратными средствами. Однако программный RAID обладает следующими недостатками:

- прграммно реализованный RAID существенно снижает общую производительность информационной системы. Это происходит из-за дополнительной нагрузки на процессор, который должен производить дополнительную работу по реализации функций RAID-массива;

- не все уровни RAID-массивов реализуются на программном уровне;

- программное обеспечение RAID связано с операционной системой хоста. Обновление программного обеспечение хоста вызывает необходимость проверки программного обеспечения на совместимость.

Аппаратная реализация RAID подразумевает разработку специализированного контроллера RAID-массива для хоста или для дискового массива. Тип реализации контроллера определяет способ взаимодействия дискового массива с хостом.

Плата RAID-контроллера устанавливается в хост, жесткие диски подключаются непосредственно к контроллеру, взаимодействие с жесткими дисками осуществляется через PCI-шины.

Внешний RAID-контроллер – это отдельное устройство, функционирующее как интерфейс между хостом и дисковым массивом. Основными функциями контроллера являются:

- управление группами дисков и контроль состояния дисков;

- перевод запросов на ввод/вывод между логическими и физическими дисками;

- восстановление данных при сбоях дискового массива.

Общая структура RAID-массива может быть представлена в следующем виде (рис. 14).

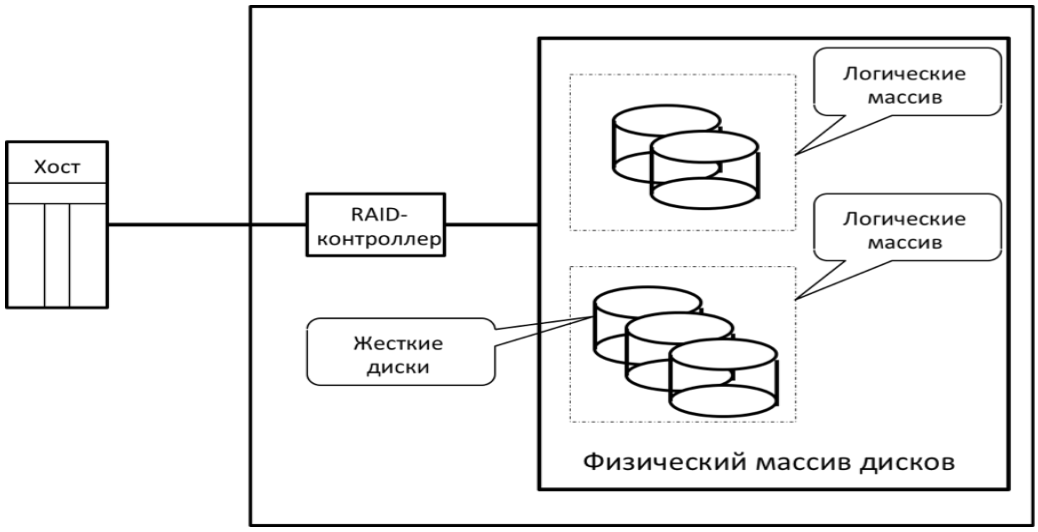


Рис. 14. Основные компоненты RAID-массива

2.2. RAID-уровни

RAID-уровни формируются с использованием следующих технологий:

- распределения;

- зеркалирования;

- контроля четности.

Эти технологии определяют доступность данных, производительность массива и надежность хранения данных. Некоторые RAID-массивы используют одну из перечисленных технологий, другие могут использовать одновременно несколько технологий. Выбор уровня RAID-массива определяются требованиями производительности приложения доступности и надежности хранения данных.

RAID 0 (распределение)

Минимальное количество дисков 2.

Объем массива = число дисков * размер наименьшего диска.

При записи данные разбиваются на блоки и пишутся на каждый диск массива. Избыточность отсутствует, выход одного из дисков приводит к полной потери информации. Используется только для ускорения чтения/записи, применяется там, где не важна надежность, а нужна скорость и низкая стоимость. Например, как массив для виртуальных машин. При увеличении количества дисков в массиве увеличивается производительность, так как появляется возможность одновременного считывания и записи большего количества блоков данных. Не рекомендуется для хранения важных данных, так как надежность массива в целом ниже, чем у отдельно взятого жесткого диска. При восстановление такого массива главное выстроить образы дисков в правильном порядке. На рис. 15 представлена общая структура распределения данных по дисковому массиву при использовании RAID нулевого уровня.

RAID 1 (зеркалирование)

Минимальное количество дисков 2.

Объем массива = размер наименьшего диска.

Зеркалирование – это технология хранения данных на двух разных дисках (на втором диске хранится копия данных, записанных на первом диске). Производительность данного уровня RAID при записи равна производитель-

ности самого медленного диска в массиве, а вот при чтении возможен выигрыш, читать можно одновременно со всех дисков. При этом это самый дорогой вариант RAID массива, так как дисковое пространство используется крайне не эффективно.

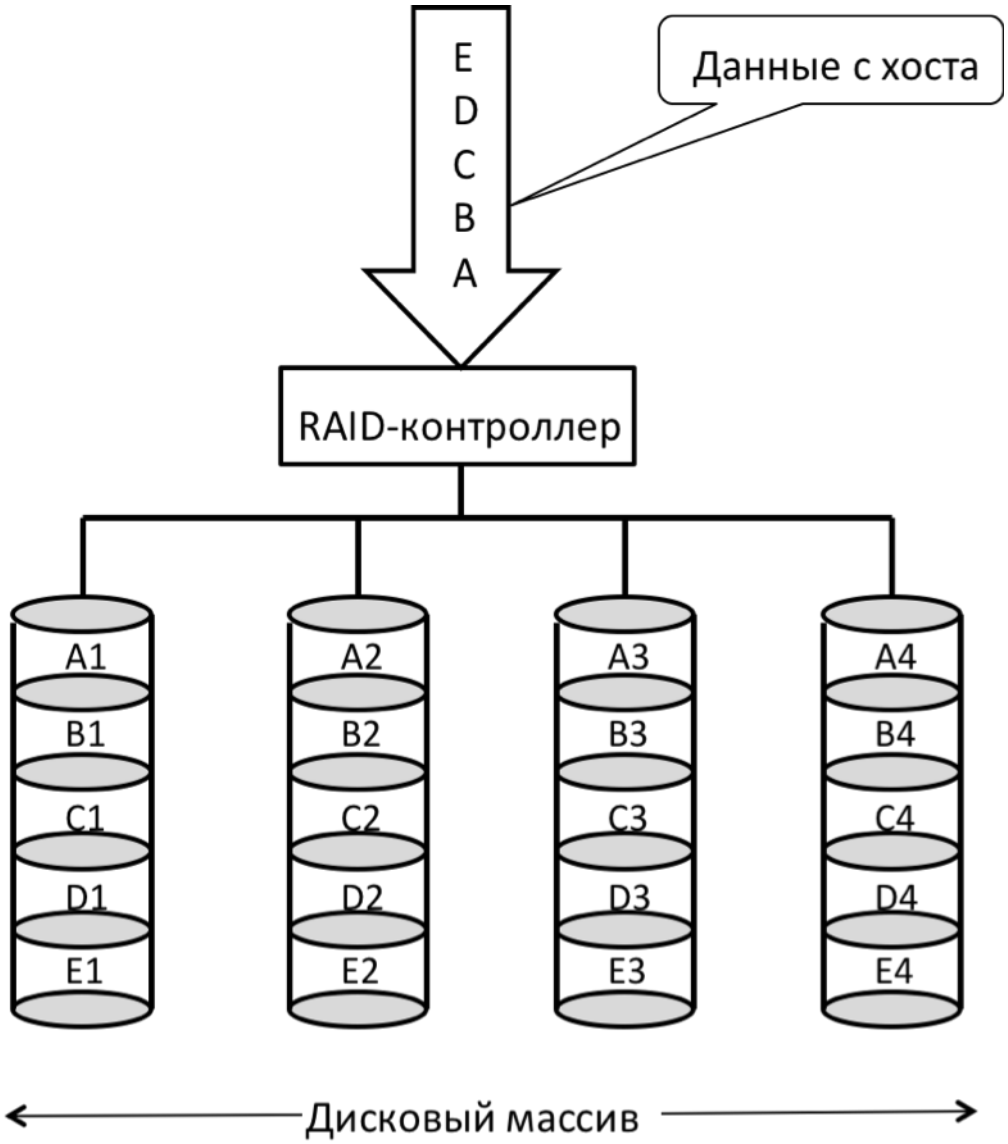


Рис. 15. Общая структура распределения данных при использовании RAID нулевого уровня

В случае сбоев в работе массива, восстановление данных осуществляется с минимальными по сравнению с другими RAID массивами затратами. RAID-контроллер использует для восстановления данных диск с зеркалированными данными. Конфигурация RAID 1 рекомендуется для использования с приложениями, требующими высокой надежности хранения данных. На рис. 16 представлена общая структура распределения данных по дисковому массиву при использовании RAID первого уровня.

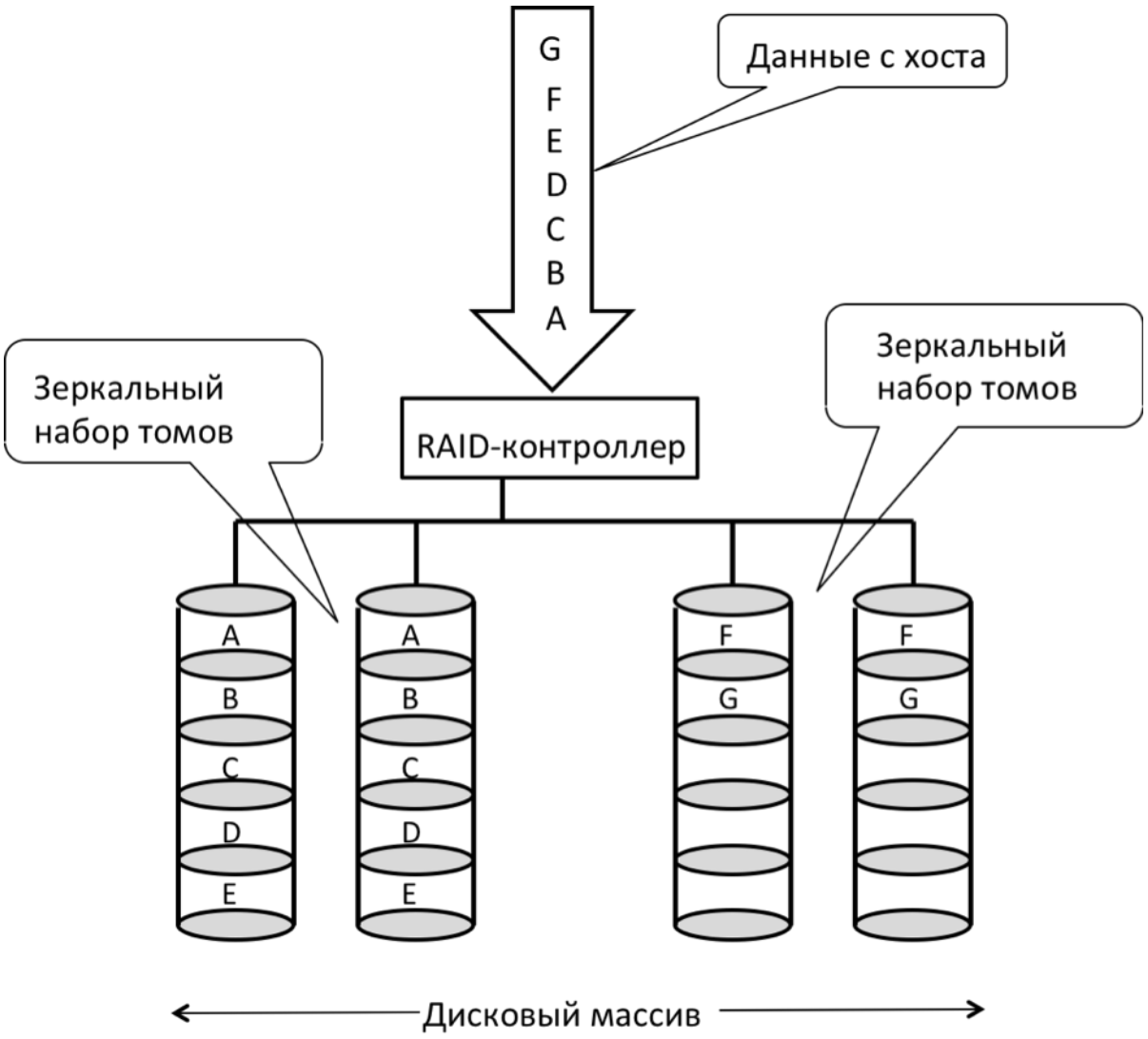


Рис. 16. Общая структура распределения данных при использовании RAID первого уровня

RAID 2 (коррекция ошибок)

В массивах такого типа диски делятся на две группы - для данных и для кодов коррекции ошибок, причем если данные хранятся на n дисках, то для складирования кодов коррекции необходимо n-1 дисков. Данные записываются на соответствующие винчестеры так же, как и в RAID-0, они разбиваются на небольшие блоки по числу дисков, предназначенных для хранения информации. Оставшиеся диски хранят коды коррекции ошибок, по которым в случае выхода какого-либо винчестера из строя возможно восстановление информации. В данной конфигурации надёжность достигается за счет применения кодов коррекции по алгоритму У. Хэмминг (Richard W. Hamming), который позволяет обнаруживать и исправлять битовые ошибки на лету (Error

Detection and Correction). Однако держать ради этого громоздкую структуру из почти двойного количества дисков никому не хотелось, и этот вид RAID-массива не получил распространения.

RAID 3 (контроль четности)

RAID 3 распределяет данные по дисковому массиву для увеличения производительности и применяет контроль четности для повышения отказоустойчивости системы.

Структура массива RAID-3 такова: в массиве из n дисков данные разбиваются на блоки размером 1 байт и распределяются по n-1 дискам, а еще один диск используется для хранения блоков четности. В RAID-2 для этой цели стояло n-1 дисков, но большая часть информации на этих дисках использовалась только для коррекции ошибок на лету, а для простого восстановления в случае поломки диска, хватает и одного выделенного винчестера.

Соответственно, отличия RAID-3 от RAID-2 очевидны: невозможность коррекции ошибок на лету и меньшая избыточность. Преимущества таковы: скорость чтения и записи данных высока, а для создания массива требуется совсем немного дисков, всего три. Но массив этого типа хорош только для однозадачной работы с большими файлами, так как наблюдаются проблемы со скоростью при частых запросах данных небольшого объема. На рис. 17 представлена общая структура распределения данных по дисковому массиву при использовании RAID третьего уровня.

RAID 4 (контроль четности)

RAID-4 похож на RAID-3, но отличается от него тем, что данные разбиваются на блоки, а не на байты. Таким образом, удалось устранить проблему низкой скорости передачи данных небольшого объема. Запись же производится медленно из-за того, что четность для блока генерируется при записи и записывается на единственный диск. Используются массивы такого типа очень редко.

Структура распределения данных по дисковому массиву при использовании RAID четвертого уровня не отличается от структуры, представленной на рис. 17.

RAID 5 (распределенное хранение информации о четности)

Минимальное количество дисков 3.

Объем массива = (Число дисков - 1) * Размер наименьшего диска.

Конфигурация RAID 5 похожа на конфигурацию RAID 4, поскольку использует сегментирование при независимом доступе к полосам. Разница за-

ключается в размещении контрольных данных. В RAID 4 они записываются на отдельный диск. Этот диск становится узким местом при записи. В RAID 5 контрольная информация распределяется по всем дискам.

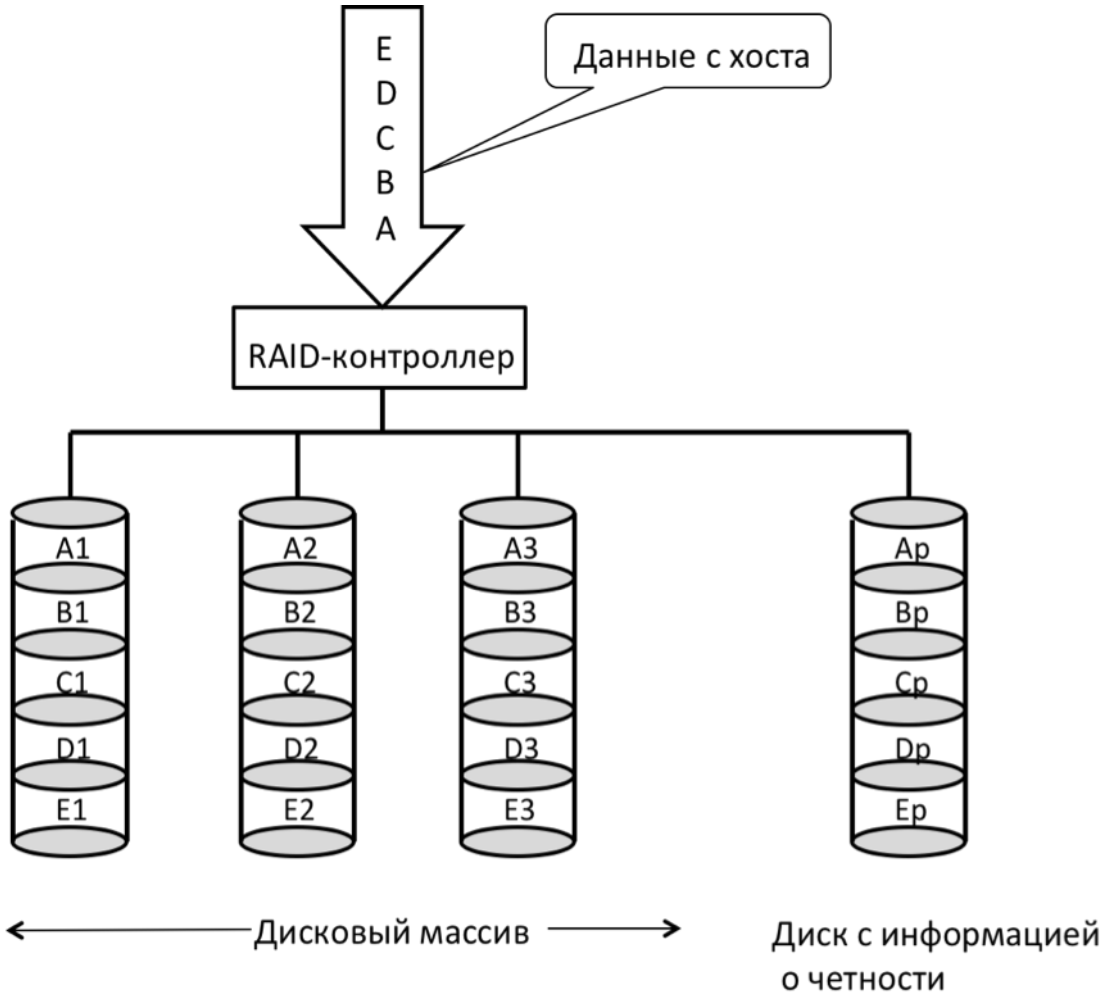


Рис. 17. Общая структура распределения данных при использовании RAID третьего уровня

Под контрольной информацией подразумевается результат операции *xor* (исключающее или). Логическая операция *xor* обладает особенностью, которая даёт возможность заменить любой операнд результатом, и, применив алгоритм *xor*, получить в результате недостающий операнд. Например: *a xor b = c* (где *a, b, c* — три диска рейд-массива), в случае если *a* откажет, можно получить информацию диска *a*, поставив на его место *c* и проведя *xor* между *c* и *b*: *c xor b = a*. Это применимо вне зависимости от количества операндов: *a xor b xor c xor d = e*. Если отказывает *c* тогда *e* встаёт на его место и проведя *xor* в результате получаем *c: a xor b xor e xor d = c*. Этот метод, по сути, обеспечивает отказоустойчивость 5 версии. Для хранения результата *xor* требует-

ся всего 1 диск, размер которого равен размеру любого другого диска в RAID.

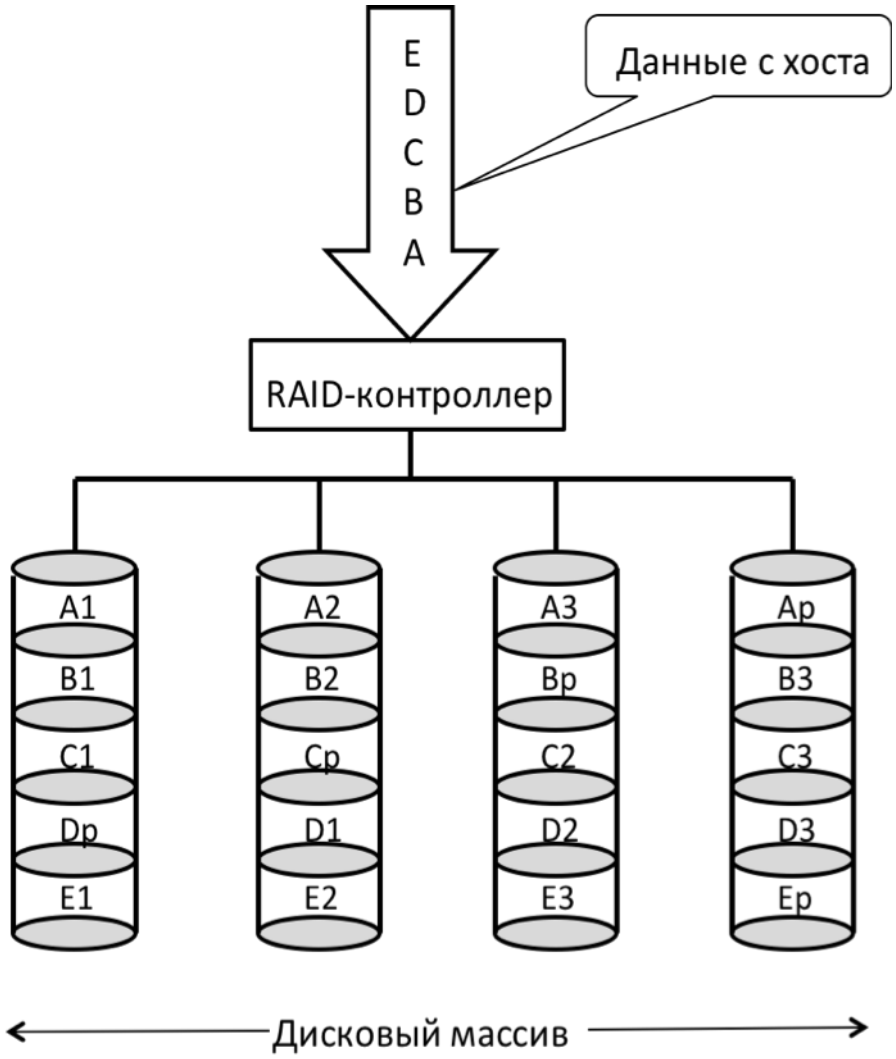


Рис. 18. Общая структура распределения данных при использовании RAID пятого уровня

На рис. 18 представлена общая структура распределения данных по дисковому массиву при использовании RAID пятого уровня. Места хранения контрольной информации обозначены как *Ap, Bp, Cp, Dp, Ep.*

Рассмотрим пример восстановления данных для RAID массива, структура которого изображена на рис.18. Имеется четыре диска для хранения данных и записи контрольных сумм. Необходимо записать на дисковый массив последовательность битов 1101 0011 1100. RAID- контролер разбивает эту последовательность на блоки по четыре бита, то есть A1=1101, A2=0011, A3=1100 и вычисляет контрольную сумму

Ap=A1⊕ A2⊕ A3=1101⊕ 0011⊕ 1100=0010

Эта контрольная сумма, записываемая на четвертый диск.

Если один из дисков, например, третий, вышел из строя, то блок A3 = 1101 оказывается недоступным для считывания данных. Однако его значение легко восстановить по контрольной сумме и по значениям остальных блоков с помощью все той же операции «исключающего ИЛИ»:

A3=A1⊕ A2⊕ Ap =1101⊕ 0011⊕ 0010=1100

В случае RAID 5 все диски массива имеют одинаковый размер, однако общая емкость дисковой подсистемы, доступной для записи, становится меньше ровно на один диск. Например, если четыре диска имеют размер 100 Гбайт, то фактический размер массива составляет 300 Гбайт, поскольку 100 Гбайт отводится на контрольную информацию.

RAID 5 используется в основном для обслуживания систем средней производительности.

Недостатки RAID 5 проявляются при выходе из строя одного из дисков, при этом весь том переходит в критический режим (degrade), все операции записи и чтения сопровождаются дополнительными манипуляциями, резко падает производительность. Уровень надежности снижается до надежности RAID-0 с соответствующим количеством дисков (то есть в n раз ниже надежности одиночного диска). Если до полного восстановления массива произойдет выход из строя, или возникнет невосстановимая ошибка чтения хотя бы еще на одном диске, то массив разрушается, и данные на нем восстановлению обычными методами не подлежат.

Следует принять во внимание, что процесс RAID Reconstruction (восстановления данных RAID за счет избыточности), после выхода из строя диска вызывает интенсивную нагрузку чтения с дисков на протяжении многих часов непрерывно, что может спровоцировать выход какого-либо из оставшихся дисков из строя в этот наименее защищенный период работы RAID, а также выявить ранее необнаруженные сбои чтения в массивах cold data (данных, к которым не обращаются при обычной работе массива, архивные и малоактивные данные), что резко повышает риск сбоя при восстановлении данных.

RAID 6 (распределенное хранение двух контрольных сумм)

Минимальное количество дисков 4.

Объем массива = (Число дисков - 2) * Размер наименьшего диска.

RAID 6 - похож на RAID 5, но имеет более высокую степень надёжности. Под контрольные суммы выделяется ёмкость 2-х дисков, рассчитываются 2 суммы по разным алгоритмам (XOR и алгоритм Рида-Соломона). Тре-

буется более мощный RAID-контроллер. Обеспечивается работоспособность после одновременного выхода из строя двух дисков, то есть, реализована защита от кратного отказа. Для организации массива требуется минимум 4 диска. Обычно использование RAID-6 вызывает примерно 10-15% падение производительности дисковой группы, относительно RAID 5, что вызвано большим объёмом обработки для контроллера (необходимость рассчитывать вторую контрольную сумму, а также читать и перезаписывать больше дисковых блоков при записи каждого блока).

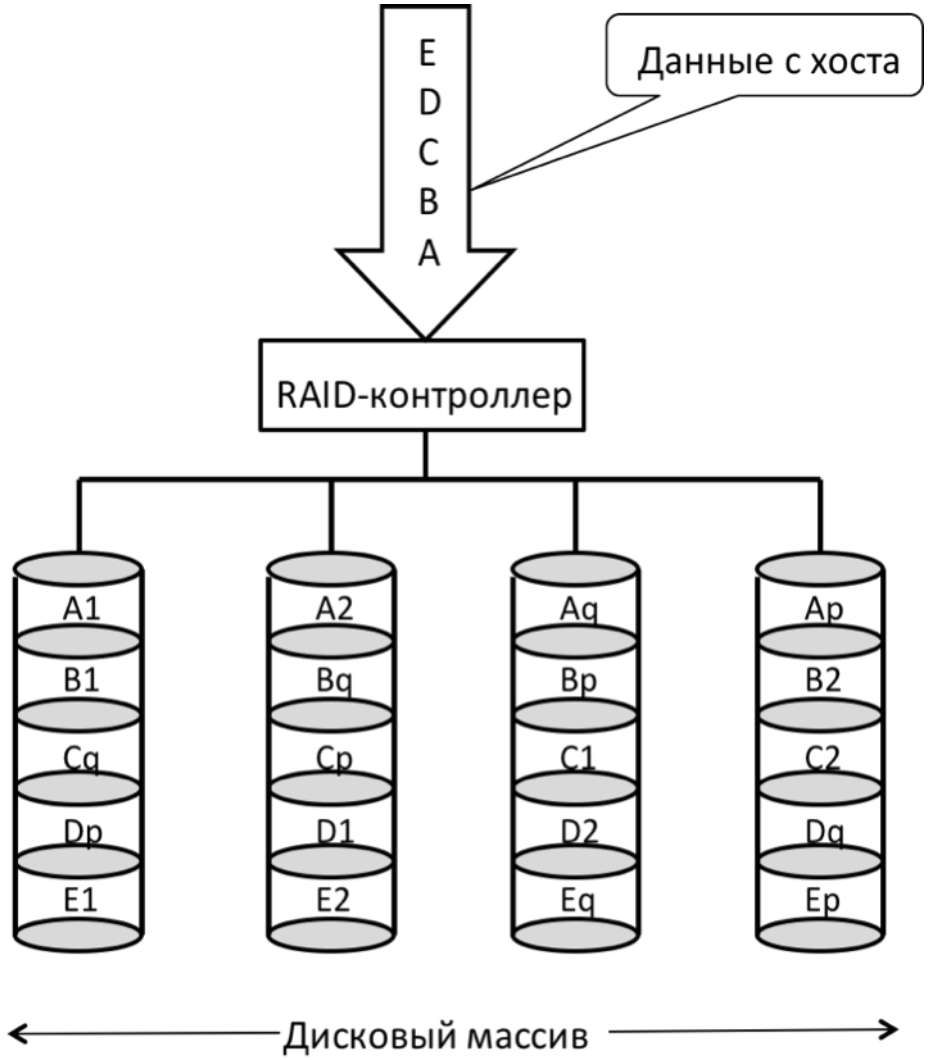


Рис. 19. Общая структура распределения данных при использовании RAID шестого уровня

Из-за своей излишней избыточности и потери производительности не получил широкого распространения и может быть рекомендован к использованию только при повышенных требованиях к надёжности.

Общая структура распределения данных по дисковому массиву при использовании RAID шестого уровня представлена на рис. 19. Места хранения контрольной информации обозначены как *Ap, Bp, Cp, Dp, Ep* – рассчитанные по *xor*-алгоритму и *Aq, Bq, Cq, Dq, Eq* – по алгоритму Рида-Соломона (недвоичные циклические коды, позволяющие исправлять ошибки в блоках данных).

2.3. Комбинированные уровни RAID-массивов

Помимо стандартных («Common RAID Disk Drive Format (DEF) standard») базовых уровней RAID 0 - RAID 6, существуют комбинированные уровни с названиями вида «RAID α+β» или «RAID αβ», что обычно означает «RAID β составленный из нескольких RAID α».

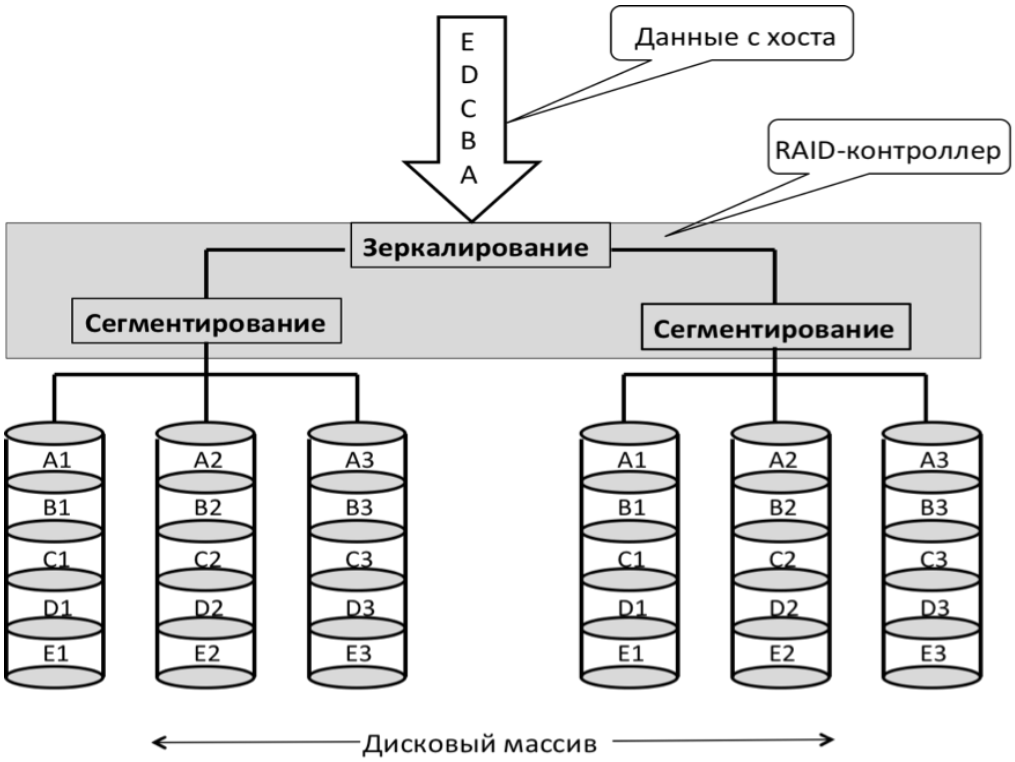


Рис. 20. Общая структура распределения данных при использовании RAID 10 (1+0)

Например:

• RAID 10 (или 1+0) — это RAID 0, составленный из нескольких (или хотя бы двух) RAID 1 (зеркалированных пар). Общая структура организации данных на дисках RAID 10 представлена на рис. 20.

•RAID 01 (или 0+1) — это RAID 1, составленный из нескольких (или хотя бы двух) RAID 0 (зеркалированных пар). Общая структура организации данных на дисках RAID 01 представлена на рис. 21.

• RAID 51 — RAID 1, зеркалирующий два RAID 5 .

При обычных условиях RAID 10 и RAID 01 обладают одинаковыми преимуществами. Различие между ними проявляется при операциях восстановления информации в случае выхода из строя одного из дисковых устройств массива. RAID 10 – сегментированное зеркало. Основным элементом RAID 10 является зеркальная пара. Это значит, что данные сначала зеркалируются, а потом обе копии данных распределяются по разным дискам в RAID 0. Если диск в любом задействованном зеркальном комплекте откажет, то его содержимое может быть получено с оставшегося диска зеркальной пары.

Таким образом, если массив RAID 10 подвергнется максимальному количеству отказов дисков, которое он может перенести, он преобразится в массив RAID 0, не имеющий надежности, но зато очень быстрый.

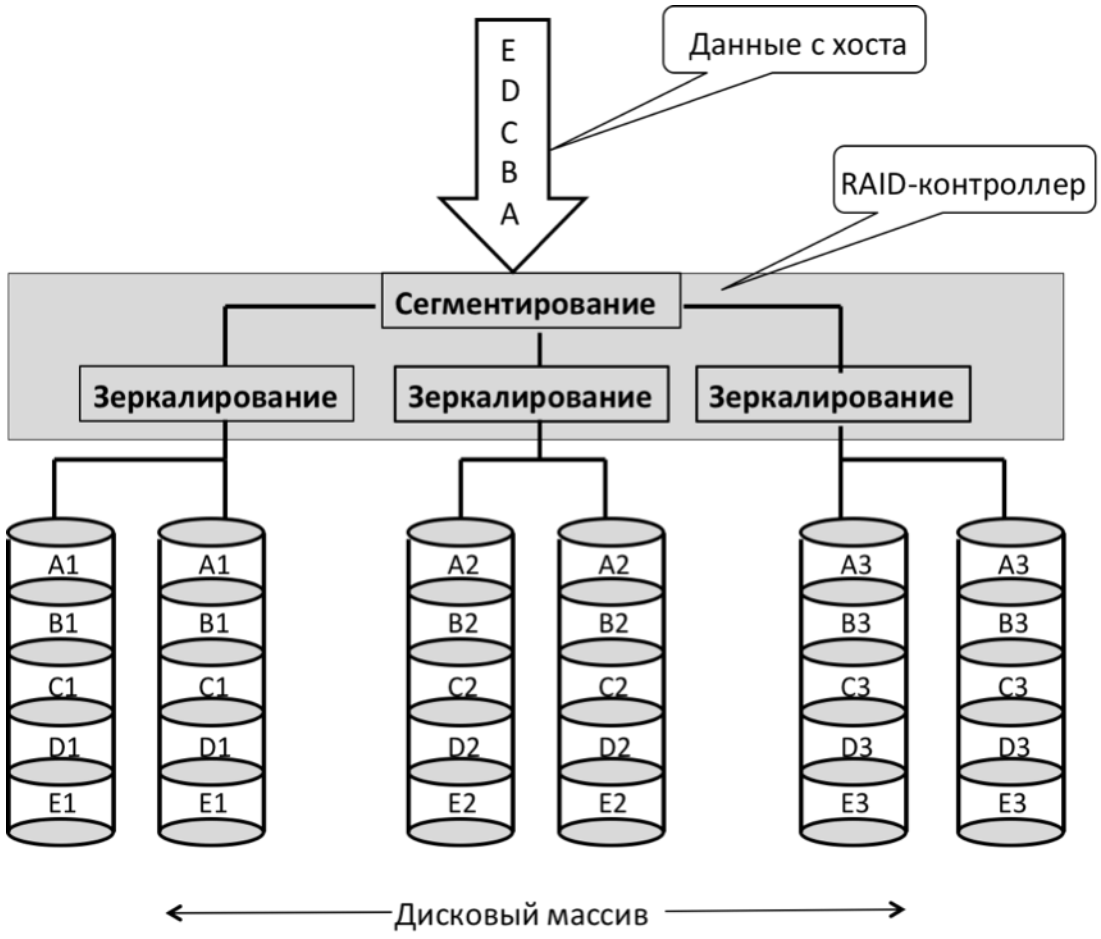


Рис. 21. Общая структура распределения данных при использовании RAID 01 (0+1)

Базовым элементом в RAID 01 является дорожка, то есть сначала данные сегментируются по разным дискам, а потом вся дорожка зеркалируется. При повреждении одного диска повреждается вся дорожка. Операция восстановления копирует всю дорожку на массив с поврежденным диском, что существенно увеличивает время восстановления данных.

Комбинированные уровни наследуют как преимущества, так и недостатки своих «родителей»: появление чередования в уровне RAID 50 нисколько не добавляет ему надёжности, но зато положительно отражается на производительности. Уровень RAID 15, будет очень надёжным, но не самым быстрым и, к тому же, крайне неэкономичным: полезная ёмкость тома меньше половины суммарной ёмкости дисков.

2.4. Расчет конфигурации RAID-массивов

При выборе конфигурации RAID-массива важно учитывать влияние структуры этого массива на производительность дискового массива. Так в RAID-массивах с зеркалированием, а также в массивах с контролем по четности на все операции записи необходимо тратить дополнительные ресурсы дисков, которые называются дополнительными расходами на запись или пенальти.

В RAID 1 все записи должны выполнятся на два диска (основной и зеркало), таким образом, дополнительные расходы на запись составляют одну операцию.

В RAID 5 контроллер при выполнении каждой операции записи должен считывать, рассчитывать и записывать блок данных с информацией по четности. При наличии в RAID 5 четырех дисков, три диска используется для хранения данных, а четвертый – для хранения блока с контрольной информацией по четности (рис. 18). Информация *Ap* рассчитывается следующим образом

$$Ap = A1 \oplus A2 \oplus A3$$

Если хост хочет изменить блок данных, который занимает место только на одном диске *A3* (то есть на одном стрипе в пределах одного страйпа), то RAID контроллер не может просто записать этот небольшой блок данных на *A3* и считать запрос хоста выполненным. Эта операция должна обновить данные четности *Ap*.

Новая информация рассчитывается следующим образом

$$Apnew = Apold - A3old + A3new$$

В последнем выражении должны выполняться операции XOR.

В булевой арифметике минус реализуется стандартной операцией искючающего ИЛИ, то есть

$$Apnew = Apold \oplus A3old \oplus A3new$$

После расчета новой контрольной информации по четности контроллер производит запись новых данных на *A3* и *Ap,* на этом операция записи завершается (получается две операции записи).

Таким образом, для каждой операции записи блока данных, контроллер выполняет две операции считывания и две операции записи, дополнительные расходы на запись при использовании RAID 5 составляют 4 (рис.22).

Для определения количества дисков, требующихся для обеспечения работы приложения необходимо учитывать влияние RAID конфигурации на IOPS дисковой системы.

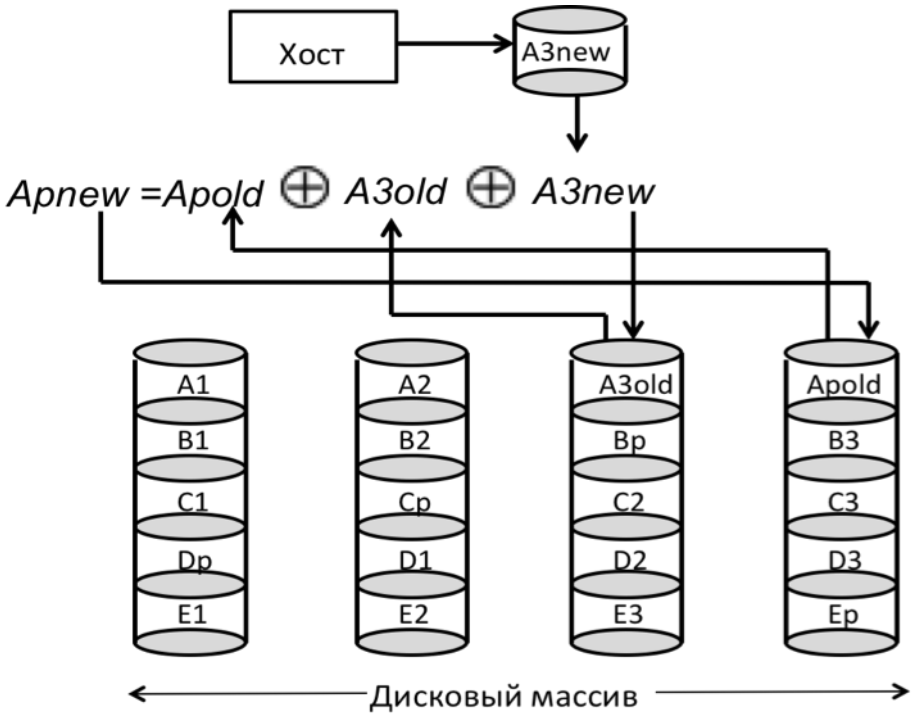


Рис. 22. Дополнительные расходы на запись в конфигурации RAID 5

Рассмотрим приложение, генерирующее 5200 IOPS операций, 60% из которых являются операциями считывания. Нагрузка на дисковую систему при наличии RAID 5 рассчитывается следующим образом.

$$RAID\ 5 = 0{,}6 \cdot 5200 + 4 \cdot (0{,}4 \cdot 5200) = 11\ 440\ IOPS$$

Рассчитанная нагрузка на дисковую систему позволяет определить количество дисков в системе.

Так если в данном примере используются жесткие диски с максимальной рабочей нагрузкой 180 IOPS, то количество дисков определится как

Nhdd = 11 400 / 180 = 64 диска

Если использовать конфигурацию RAID 1, то нагрузка на дисковую систему составит

RAID 1 = 0,6 • 5200 + 2 • (0,4 • 5200) = 7 200 IOPS.

При использовании для реализации RAID 1 жестких дисков с такими же, как и в предыдущем случае, характеристиками их количество определится как

Nhdd = 7 800 / 180 = 42 диска

Здесь для получения результата количество дисков округляется до следующего четного числа.

Вопросы для самопроверки

1. Какова основная функция RAID-массива магнитных дисков?
2. Какой из RAID-массивов обеспечивает только ускорение выполнения операций чтения/записи данных?
3. При какой конфигурации RAID-массива используется зеркалирование данных?
4. Каково минимальное количество дисков, позволяющее реализовать RAID 5?
5. Какой порядок выполнения операций "зеркалирование" и "сегментирования" в RAID 10?
6. Какая из дисковых систем, RAID 1 или RAID 0, обладает большей производительностью?
7. Как формируется контрольная информация в RAID 5, если для реализации RAID-массива используется 4 дисковых устройства?
8. На какое из дисковых устройств записывается контрольная информация в RAID 5, если для реализации RAID-массива используется 4 дисковых устройства?

3. ИНТЕЛЛЕКТУАЛЬНЫЕ СХД

Интеллектуальные системы хранения данных, рассматриваемые в данном разделе, представляют собой полнофункциональные RAID массивы, обеспечивающие оптимальные параметры обработки запросов на ввод/вывод информации. Управление процессами обработки данных в таких массивах осуществляется операционной средой с использованием больших массивов Кэш-памяти.

3.1. Основные компоненты интеллектуальных систем хранения данных

В общем случае интеллектуальная СХД состоит из четырех основных компонентов (рис. 23) – внешнего блока, Кэш-памяти, внутреннего блока и массива физических дисков.

Внешний блок обеспечивает интерфейс связи между СХД и хостом, состоит из внешних портов и контроллеров, реализующих основные функции управления данными.

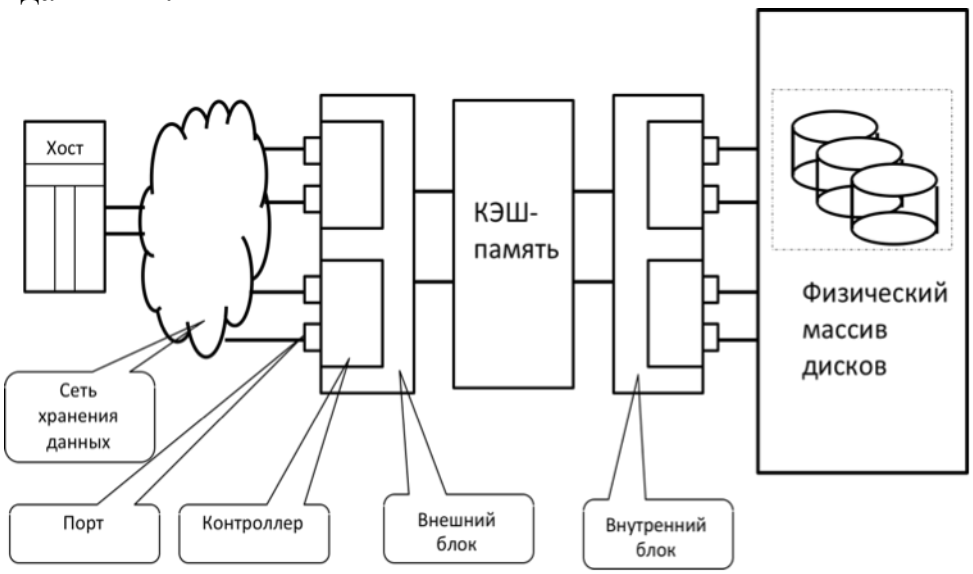


Рис. 23. Основные компоненты интеллектуальной системы хранения данных

Внешние порты работают в соответствии с используемыми транспортными протоколами передачи данных. Это, как правило, SCSI, Fibre Channel или iSCSI.

Для повышения доступности и надежности функционирования СХД входной блок контроллеров снабжается резервными портами.

Кэш-память представляет собой энергозависимую полупроводниковую память, в которую временно помещают данные для уменьшения времени,

требуемого на обслуживание запросов хоста на выполнение операций ввода/вывода.

Внутренний блок обеспечивает выполнение интерфейсных функций между Кэш-памятью и физическими дисками, состоит из внутренних контроллеров и внутренних портов.

Алгоритмы работы внутренних контроллеров определяются необходимостью поддерживать функции RAID, а также обнаружение и исправление ошибок.

Использование нескольких внутренних контроллеров и портов повышает надежность и облегчает балансировку загрузки элементов СХД.

Физические диски предназначены для хранения данных. Перспективным направлением развития устройств хранения данных является использование энергонезависимых твердотельных накопителей (SSD - solid-state drive). В отличие от традиционных жестких дисков флэш-накопители не имеют движущихся частей и выигрывают в быстродействии, бесшумности и механической надежности. В массиве хранения данных флэш-накопители могут хранить до терабайта информации и при этом обеспечивают меньшее (более 30%) энергопотребление по сравнению с традиционными механическими дисковыми накопителями.

3.2. Определение очередности выполнения запросов

Контроллеры внешнего блока оптимизируют обработку запросов ввода/вывода, изменяя очередность выполнения запросов.

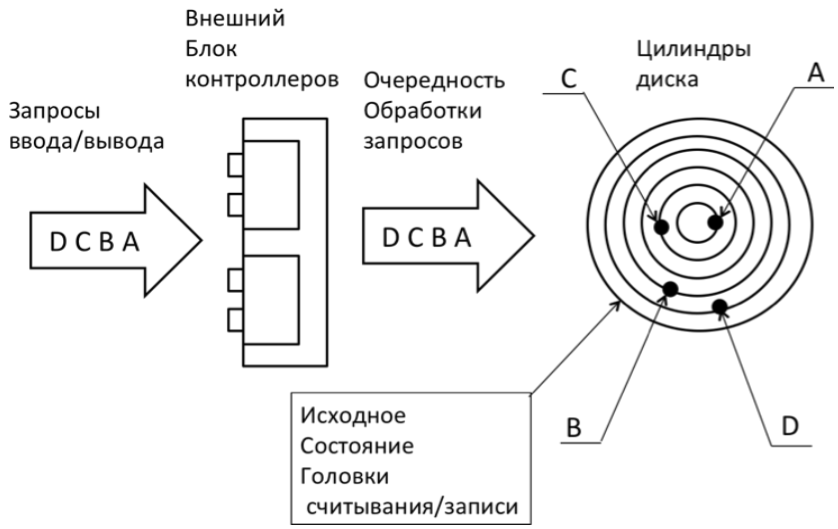


Рис. 24. Обработка запросов согласно алгоритму FIFO

Существует несколько наиболее распространенных алгоритмов обработки запросов в СХД.

Алгоритм "Первый пришел, первым выполнен" (FIFO - first in, first out). Это стандартный алгоритм, при котором запросы выполняются в последовательности, в которой они были получены (рис. 24). То есть очередность выполнения запросов не изменяется.

Оценка общего времени выполнения запросов хоста приведена на рис. 25. На указанном рисунке:

$t_п$ – время прохождения блока головок чтения/записи одной дорожки при выполнении операции позиционирования головок;

$t_о$ – время ожидания момента прохождения нужного блока данных под головкой чтения/записи;

$t_з$ – время чтения/записи блока данных.

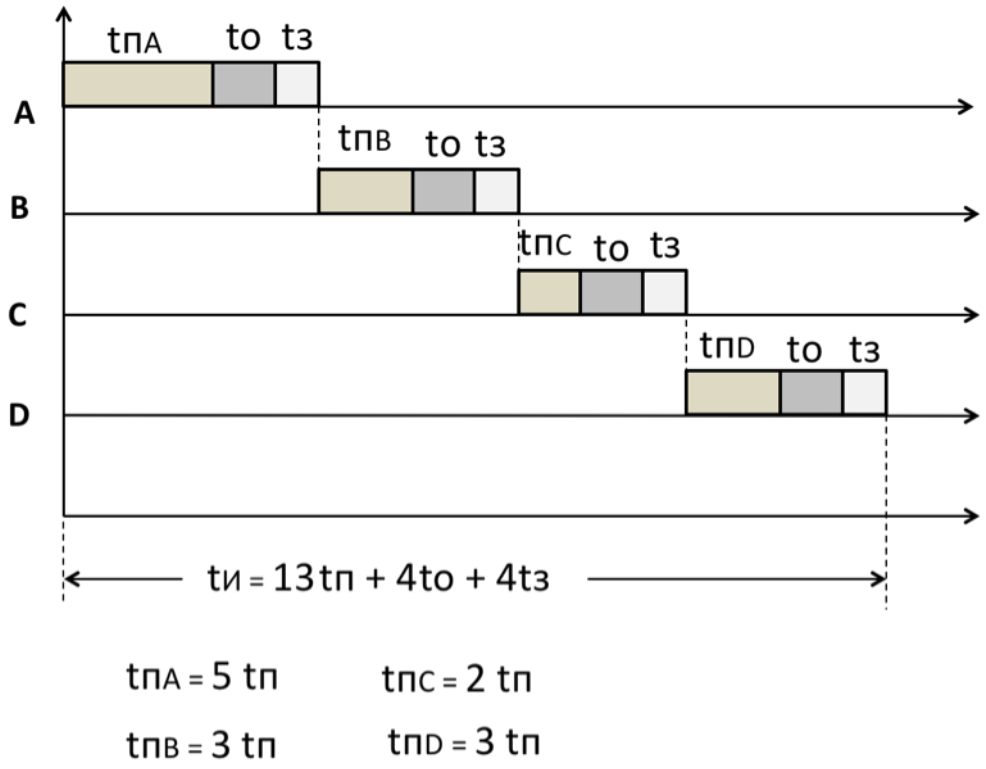


Рис. 25. Оценка времени выполнения запросов согласно алгоритму FIFO

Алгоритм оптимизации времени позиционирования блока головок чтения/записи. При использовании данного алгоритма порядок обработки запросов изменяется согласно рис. 26.

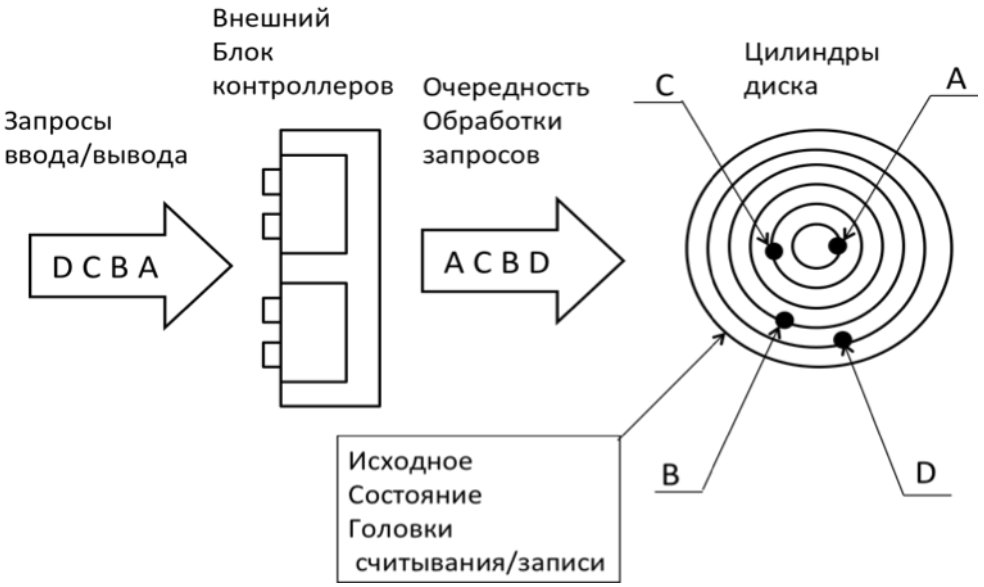


Рис. 26. Обработка запросов согласно алгоритму оптимизации времени позиционирования головок

Оценка общего времени выполнения запросов хоста при использовании алгоритма оптимизации времени позиционирования головок приведена на рис. 27.

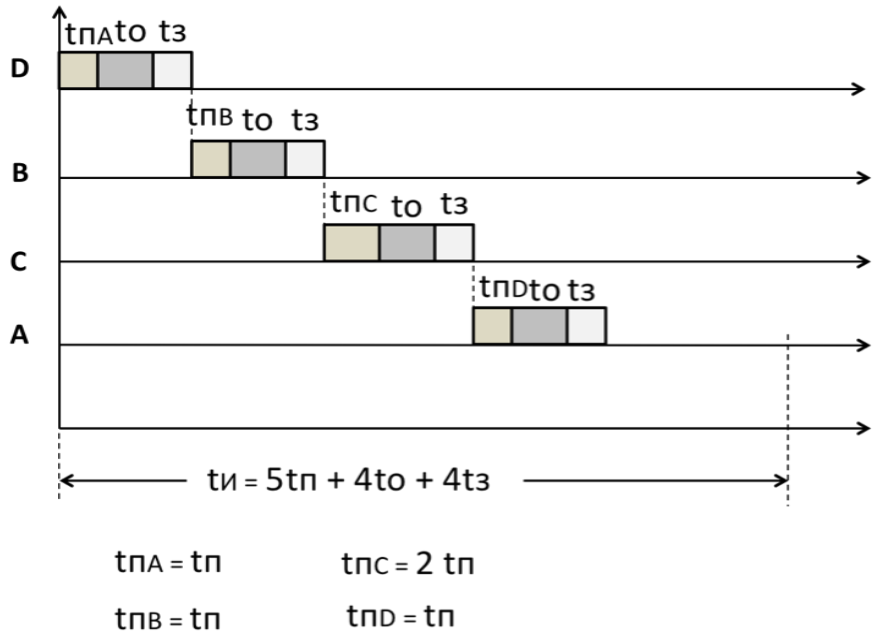


Рис. 27. Оценка времени выполнения запросов согласно алгоритму оптимизации времени позиционирования головок

Таким образом, общее время исполнения запросов A, B, C, D при оптимизации перемещений блока головок записи/считывания уменьшилось на величину 8 tп.

Существуют алгоритмы обработки запросов, которые наряду с оптимизацией времени позиционирования головок считывания/записи, учитывают время ожидания момента прохождения нужного блока под головками считывания/записи.

3.3. Использование Кэш-памяти

Использование Кэш-памяти – это универсальный способ ускорения доступа к данным СХД. Улучшение быстродействия СХД обеспечивается за счет исключения из процесса доступа к данным механических задержек, связанных с физическими дисками, которые являются самыми медленными компонентами интеллектуальных систем хранения данных.

Под Кэш-памятью понимается не только способ ускорения доступа к данным, но и быстродействующее запоминающее устройство.

Обычно Кэш-память организована в виде страниц (слотов), размер которых конфигурируется в соответствии с размерами порций данных, которыми оперирует приложение (рис.28). Вся Кэш-память делится на память для хранения массивов данных и оперативную тэговую память. В массиве данных хранятся данные приложения или дискового массива, а тэговая память используется для управления процессами обработки данных и содержит, как правило, информацию о расположении данных, а также служебную информацию, характеризующую время нахождения данных в КЭШ, актуальность хранимых данных и др.

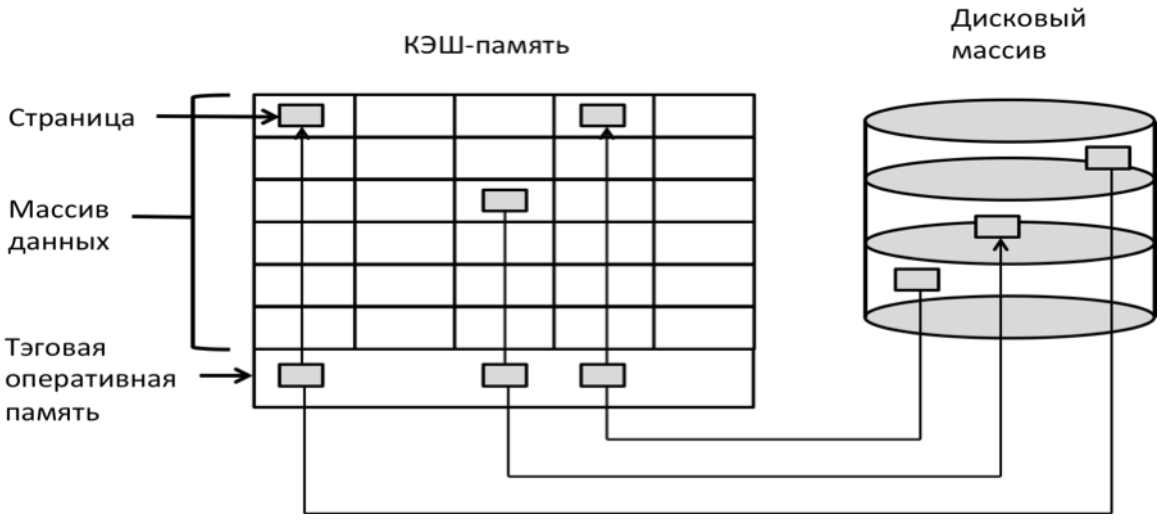


Рис. 28. Общая структура Кэш-памяти СХД

Когда приложение отправляет запрос на чтение информации с дискового массива, то внешний контроллер обращается к тэговой оперативной памяти КЭШ, чтобы определить, имеются ли запрошенные данные в Кэш-памяти. Если запрошенные данные найдены в Кэш-памяти, то такое событие называют КЭШ-попадание при чтении, и данные передаются приложению без обращения к дисковому массиву. Таким образом, обеспечивается быстрый ответ (около миллисекунды) на запрос приложения.

Событие, когда запрошенные данные не найдены в Кэш-памяти называется КЭШ-промахом, при этом запрошенные приложением данные должны быть считаны с дискового массива, помещены в Кэш-память и переданы приложению. При этом время выполнения запроса существенно увеличивается.

При последовательных запросах на чтение в интеллектуальных СХД используется алгоритм предварительной выборки (предварительного считывания). В этом случае при выполнении одного запроса с дискового массива извлекается смежный набор связанных блоков. То есть, с дискового массива считывается и помещается в Кэш-память несколько блоков, которые еще не были запрошены приложением. Когда приложение в дальнейшем запросит эти блоки, то произойдет КЭШ-попадание при чтении.

В интеллектуальных системах хранения данных используются фиксированные и переменные размеры данных предварительного считывания.

Фиксированное предварительное считывание позволяет существенно уменьшить время выполнения запросов на считывание, если размеры ввода/вывода для приложения постоянны.

Переменное предварительное считывание позволяет интеллектуальной системе хранения считывать объем данных, кратный размеру запроса приложения.

Общее время доступа к данным с учетом Кэширования зависит от вероятности КЭШ-попадания.

Так, если время доступа к данным на дисковом массиве равно t1, а время доступа к данным расположенным в Кэш-памяти равно t2, то при вероятности КЭШ-попадания равной p, среднее время доступа к данным определится как

$$t=t1(1-p)+ t2 \cdot p$$

Например, при $p=0,8$

$$t=t1(1-0,8)+ t2 \cdot 0,8$$

Если учесть, что *t2<<t1*, то можно считать, что среднее время доступа к данным уменьшилось в пять раз.

Опыт использования СХД с КЭШ памятью показывает, что при решении стандартных задач процент КЭШ-попаданий достаточно высок (более 90%).

Операции записи данных на дисковый массив с использованием Кэш-памяти также позволяют получить преимущества по быстродействию по сравнению с записью данных напрямую на дисковый массив.

Операции записи данных с использованием Кэш-памяти могу выполняться согласно следующим алгоритмами.

КЭШ с отложенной записью. В этом случае, данные помещаются в Кэш-память и сразу же в приложение отправляется подтверждение о записи. Позднее данные из нескольких запросов на запись переносятся на дисковый массив. Реагирование приложения на запрос о записи данных происходит гораздо быстрее (без учета времени записи данных на дисковый массив).

Запись в обход КЭШ. При использовании этого алгоритма, данные помещаются в Кэш-память и немедленно записываются на дисковый массив с отправлением подтверждения о записи приложению. Так как данные переносятся на дисковый массив сразу после получения запроса, риск потери данных из-за сбоя Кэш-памяти снижается, но время реакции на запрос о записи со стороны приложения увеличивается.

3.4. Управление Кэш-памятью

Кэш-память является дорогим и ограниченным по объему ресурсом СХД. В процессе функционирования интеллектуальной системы хранения данных происходит заполнение данными страниц Кэш-памяти. Для поддержания режимов работы СХД, позволяющих эффективно использовать возможности Кэш-памяти, необходимо обеспечивать периодическое освобождение страниц Кэш-памяти для приема новой информации.

В интеллектуальных СХД используются различные алгоритмы управления процессами освобождения страниц Кэш-памяти для поддержания некоторого количества свободных страниц Кэш-памяти, а также перечня страниц, которые могут быть освобождены в случае необходимости.

Такими наиболее известными и широко используемыми алгоритмами являются LRU (Least Recently Used) и MRU (Most Recently Used).

Least Recently Used – это алгоритм замещает страницы на которых находятся давно не использованные данные. Контроллер Кэш-памяти постоянно отслеживает моменты доступа к данным, определяет давно не использованные страницы и либо освобождает эти страницы, либо маркирует их для повторного использования. Работа LRU алгоритма основывается на предполо-

жении, что данные, не использованные в течение некоторого времени, не будут запрошены приложением и в ближайшее время. Если такая страница содержит метку о том, что данные этой страницы не записаны на диск, то перед повторным использованием страницы ее содержимое записывается на дисковый массив.

Most Recently Used – это алгоритм противоположный по действию LRU. Страницы Кэш-памяти, которые были только что использованы, освобождаются или помечаются для повторного использования. Работа алгоритма основывается на предположении, что недавно использованные данные некоторое время не будут затребованы приложением.

По мере заполнения Кэш-памяти данными интеллектуальная система хранения данных должна выполнять сброс "грязных" страниц (страниц с данными, записанными в Кэш-память, но не записанными на дисковый массив). Сброс представляет собой процесс передачи на диск данных, находящихся в Кэш-памяти.

Для управления процессом сброса в Кэш-памяти задают два пороговых уровня заполнения страниц: верхний уровень и нижний уровень.

Верхний уровень – это уровень заполнения Кэш-памяти при котором начинается быстрый сброс данных, находящихся в Кэш-памяти.

Нижний уровень - это уровень заполнения Кэш-памяти при котором СХД прекращает быстрый или принудительный сброс данных и переходит на фоновый режим сброса.

Сброс в фоновом режиме происходит постоянно в небольшом объеме, когда уровень заполнения Кэш-памяти находится между высоким и нижнем уровнями заполнения.

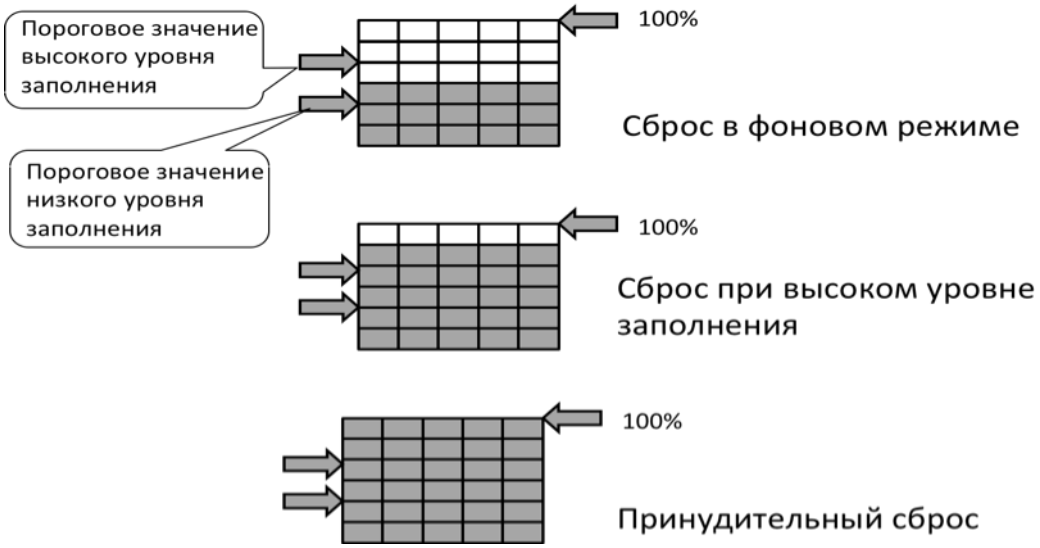


Рис. 29. Типы сброса Кэш-памяти

Сброс при высоком уровне заполнения Кэш-памяти активизируется, если при использовании Кэш-памяти достигается верхний уровень заполнения памяти, при этом СХД выделяет дополнительные ресурсы на выполнение сброса. Этот режим имеет минимальный уровень воздействия на процессы обработки запросов приложения.

Принудительный сброс происходит в случае получения запросов на большой объем ввода/вывода, когда Кэш-память заполняется на 100%. В этом случае данные всех грязных страниц Кэш-памяти принудительно перезаписываются на дисковый массив (рис. 29).

КЭШ является энергозависимой памятью и перебои в электропитании оборудования СХД может привести к потере данных, еще не переданных на дисковый массив. Такой риск потери данных можно существенно уменьшить, используя зеркальное кэширование или резервирование Кэш-памяти.

Зеркальное кэширование предусматривает хранение каждой записи данных на двух независимых картах Кэш-памяти. В случае сбоя в работе Кэш-памяти данные для записи на дисковый массив сохраняются в зеркально отображенной странице Кэш-памяти. Что касается операции считывания, то в случае сбоя в работе Кэш-памяти данные могут быть повторно считаны в Кэш-память с дискового массива.

Таким образом, зеркальному кэшированию подлежат только данные предназначенные для записи на дисковый массив.

При использовании методов зеркального кэширования возникает проблема синхронизации Кэш-памяти, то есть необходимо обеспечивать идентичность данных на двух разных страницах Кэш-памяти. Эту задачу решает операционная среда интеллектуальных систем хранения данных.

Резервирование Кэш-памяти предусматривает использование дополнительных магнитных дисков, которые используются для записи содержимого Кэш-памяти при возникновении аварийных ситуаций. При восстановлении работоспособности СХД данные с этих дисков переписываются в Кэш-память, а затем на дисковый массив.

Вопросы для самопроверки

1. Назовите основные компоненты интеллектуальной системы хранения данных?
2. За счет выполнения каких операций по управлению запросами на обслуживание повышается производительность интеллектуальной системы хранения данных?
3. Как вычислить среднее время доступа к данным в интеллектуальной системе хранения данных, если время доступа к данным на диске равно 10 мс, время доступа к данным в КЭШ равно 0,1 мс, а вероятность КЭШ-попадания равна 0,95?
4. Какие данные Кэш-памяти подлежат зеркалированию при использовании в интеллектуальной системе хранения данных резервирование Кэш-памяти?

4. СЕТИ ХРАНЕНИЯ ДАННЫХ

Сеть хранения данных (SAN - Storage Area Network) это высокоскоростная сеть, которая связывает компьютерные системы (хост-серверы) с высокопроизводительными подсистемами хранения информации (хранилищами данных).

SAN состоят из адаптеров шины, специализированных устройств для поддержки маршрутизации трафика (концентраторы и коммутаторы) и дисковых массивов хранения данных.

Технологии SAN выполняют операции чтения/записи информации на уровне блоков данных.

Чаще всего для реализации SAN используется инфраструктура оптоволоконных каналов Fibre Channel.

1.1. Fibre Channel

Fibre Channel - это открытый промышленный стандарт высокоскоростного последовательного интерфейса. Он обеспечивает подключение серверов и сторедж-систем на расстоянии до 10 км (при использовании стандартного оснащения) на скорости 100 MB/s, 200 MB/s, 400 MB/s.

Изначально технология Fibre Channel предполагала поддержку только волоконно-оптических линий (fiber optic). Однако когда добавилась поддержка меди, было принято решение название в принципе сохранить, но для отсылки на стандарт использовать британское слово Fibre.

В технологии Fibre Channel предпринята попытка объединить лучшее из двух базовых разделов техники связи — 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.

Другой подход состоит в использовании центрального хранилища данных (канальная система) и расширения существующих технологических функций. На базе этого метода создавалась технология Fibre Channel.

Одним из важнейших преимуществ Fibre Channel (FC) наряду со скоростными параметрами (которые, кстати, не всегда являются главными для пользователей SAN и могут быть реализованы с помощью других технологий) является возможность работы на больших расстояниях и гибкость топологии, которая пришла в новый стандарт из сетевых технологий. Таким образом, концепция построения топологии сети хранения данных базируется на тех же принципах, что и традиционные сети, как правило, на основе концентраторов и коммутаторов, которые помогают предотвратить падение скорости при возрастании количества узлов и создают возможности удобной организации систем без единой точки отказов.

Коммуникационный протокол FC (рис. 30) представляет собой объединение пяти уровней реализации функций обработки данных: от FC-0 до FC-4 (за исключением уровня FC-3, который не реализуется).

Рис. 30. Пакет протоколов Fibre Chanel

FC-0 определяет физический интерфейс и среду передачи данных. Спецификация FC-0 включает в себя кабели, разъемы, оптические и электрические параметры среды передачи данных.

FC-1 определяет протокол передачи данных, который включает в себя правила последовательного кодирования и декодирования информации, использования специальных символов и управление ошибками. На передающем узле 8-разрядный символ кодируется в 10-разрядный передаваемый блок и передается на приемный узел. На приемном узле 10-разрядный блок передается на уровень FC-1, который декодирует 10-разрядный блок в оригинальный 8-разрядный символ.

FC-2 представляет собой транспортный уровень и обрабатывает данные отражающие адреса портов источника и приемника, а также информацию по управлению каналом связи и передаваемую информацию. Данный уровень обеспечивает работу с адресами FC, кроме того, здесь производится структуризация данных (кадры, последовательности, обмены), управление потоками и маршрутизация.

FC-4 определяет интерфейсы приложений и способы, которыми протоколы верхнего уровня отображаются на более низких уровнях FC. Стандарт FC определяет несколько протоколов, которые могут работать на уровне FC-4. К этим протоколам относятся:

- SCSI (Small Computer Systems Interface) - протокол для физического подключения и передачи данных между компьютерами и периферийными устройствами. SCSI стандарты определяют команды, протоколы и электрические и оптические интерфейсы;
- HIPPI (High-Performance Peripheral Interface) – обеспечивает высокоскоростной обмен между самыми мощными компьютерами (например, суперкомпьютеры);
- ESCON (Enterprise Systems Connection) – обеспечивает связь систем в масштабах предприятия;
- ATM (Asynchronous Transfer Mode) - высокопроизводительная технология коммутации и мультиплексирования, основанная на передаче данных в виде ячеек (cell) фиксированного размера;
- IP (Internet Protocol) – стек протоколов, который объединяет сегменты сети в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов (маршрутизаторов).

4.2. FC-архитектура

FC-архитектура поддерживает три основных топологии использования основных компонентов SAN: "точка-точка" (FC-PTP - Fibre Channel point-to-point), "управляемая петля" (FC-AL – Fibre Channel Arbitrated Loop) и "коммутируемое соединение" (FC-SW - Fibre Channel Switch).

Архитектура "точка-точка" предусматривает непосредственное подключение двух устройств друг к другу – передатчик одного устройства соединяется с приемником другого (рис. 31). Все отправленные одним устройством кадры предназначены для второго устройства.

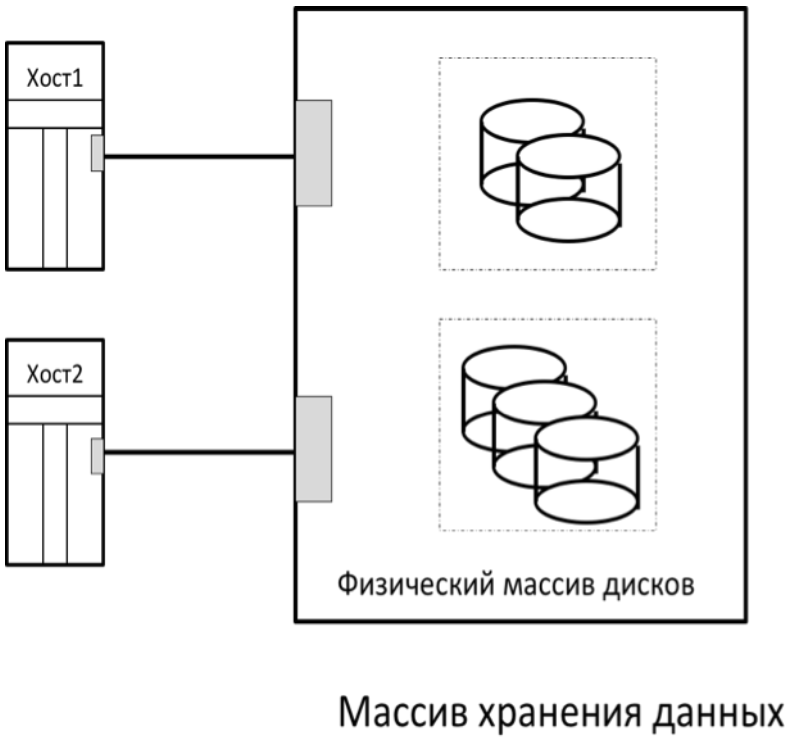


Рис. 31. Топология "точка-точка"

Эта конфигурация обеспечивает выделенное соединение для обмена данными между узлами, однако обладает ограниченными возможностями масштабирования системы.

Архитектура "управляемая петля" предусматривает объединение устройств в петлю — передатчик каждого устройства соединяется с приемником следующего устройства (рис. 32). Перед тем, как петля будет выполнять функции передачи данных, устройства составляющие петлю осуществляют "арбитраж"- договариваются о праве контроля над петлей. Для передачи данных по петле устройство должно завладеть «эстафетой» (token). В любой момент времени только одно устройство может выполнять операции ввода/вывода по петле. Для построения управляемой петли используют концен-

траторы, которые способны размыкать или замыкать петлю при добавлении нового устройства или выходе устройства из петли.

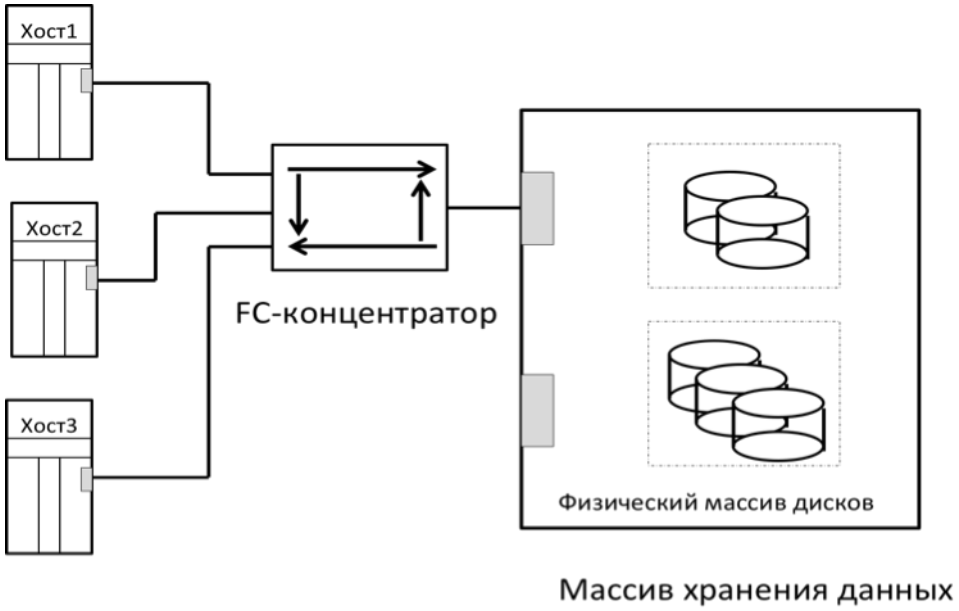


Рис. 32. Топология "управляемая петля" FC

Конфигурация FC-AL использует 8-разрядную адресацию и может поддерживать до 127 устройств в петле.

Добавление или удаление устройства в структуру петли приводит к повторной инициализации петли, что может вызывать кратковременные паузы в передаче данных по петле.

Архитектура "коммутируемое соединение" (коммутируемая сеть, коммутируемая фабрика) базируется на применении коммутаторов (рис. 33). Такая структура позволяет подключать большее количество устройств, чем в управляемой петле, при этом добавление новых устройств не влияет на передачу данных между уже подключёнными устройствами. Так как на основе коммутаторов можно строить сложные сети, на коммутаторах поддерживаются распределённые службы управления сетью (fabric services), отвечающие за маршруты передачи данных, регистрацию в сети и присвоение сетевых адресов и проч. Fibre Channel изначально разрабатывался как высокоскоростная сеть, пригодная для работы в реальном времени. В транспорте Fibre Channel заложены механизмы регулирования потока (flow control), синхронизации портов по времени и возможность повтора сбойной информации без обращения к протоколу верхнего уровня. В структурах Fibre Channel при подключении порта обязательным является выполнение операции login, так что коммутатор всегда знает о всех портах сети - какой порт где находится и какие функции может выполнять. Когда в коммутатор Fibre Channel приходит кадр данных, то коммутатор уже знает, где находится адресат и куда этот кадр

маршрутизировать (в отличие от Ethernet, в котором коммутатор после прихода кадра сначала ищет, где находится адресат и только после его ответа посылает ему этот кадр, и, если истекло время старения, коммутатор Ethernet вновь будет искать маршрут для другого кадра данных от того же источника к тому же адресату, хотя оба порта были online). Очевидно, что подход Fibre Channel требует больше ресурсов, поэтому коммутаторы по этой технологии значительно дороже, чем для Ethernet.

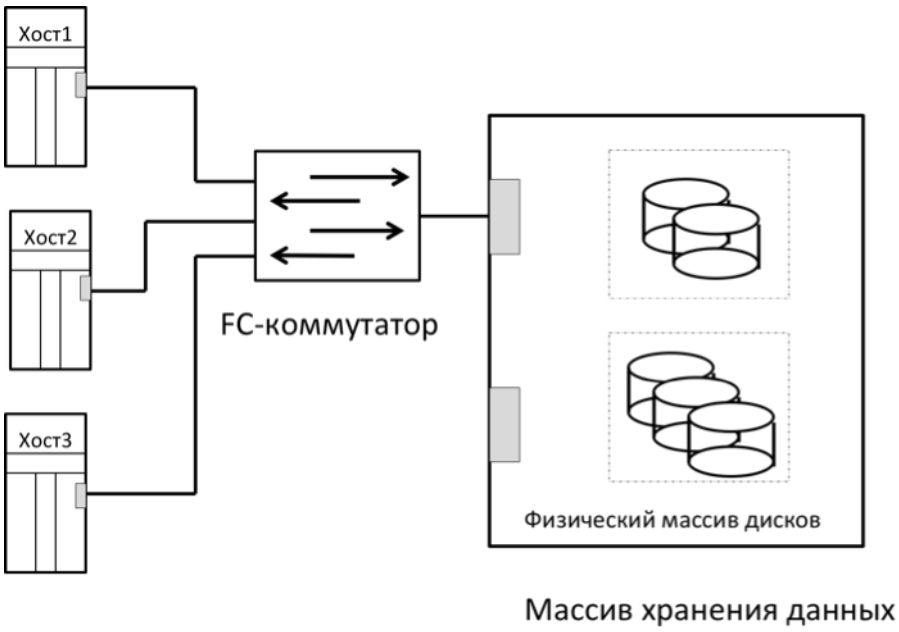


Рис. 33. Топология "коммутируемая фабрика" FC

Каждый порт структуры имеет уникальный 24-разрядный FC-адрес, обеспечивающий возможность выполнения операций коммутации между портами.

Развитие архитектуры коммутируемых фабрик FC привело к появлению двухуровневых и трехуровневых структур FC. Количество уровней в структуре FC определяется количеством коммутаторов между двумя, расположенными на наибольшем расстоянии друг от друга, узлами. На рис. 34 показаны двух- и трехуровневые архитектуры FC.

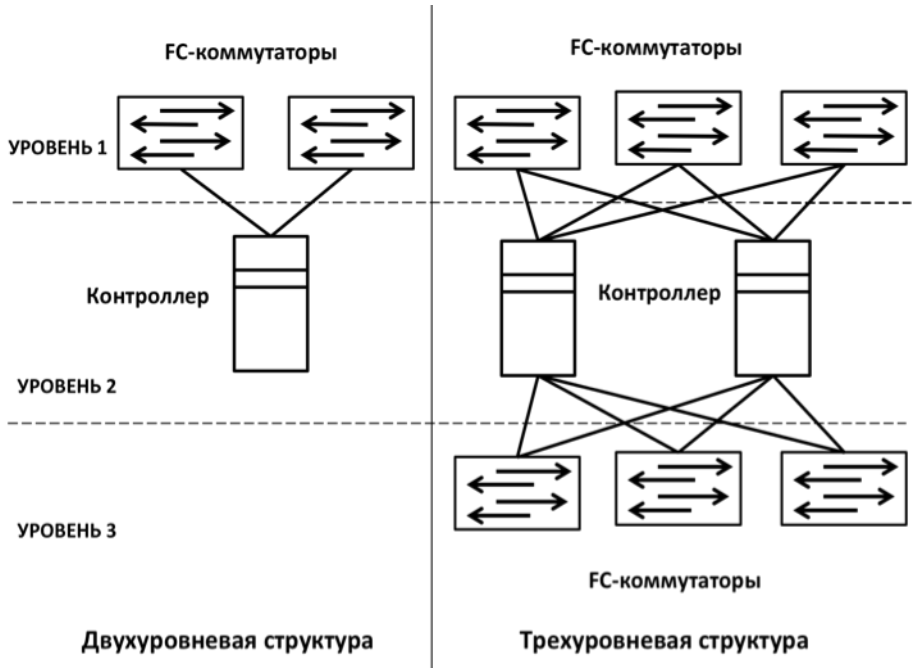


Рис. 34. Многоуровневая топология FC-SW

4.3. Порты SAN

Базовыми структурными элементами топологии FC являются порты, которые в составе SAN могут быть следующими.

Порты узлов:

- N_port (Node port) – порт устройства с поддержкой топологии FC-P2P («Точка-Точка») или FC-SW (с коммутатором). Конечная точка в топологии FC. Этот порт является узловым, обеспечивает подключение порта главной шины хоста или порта массива хранения данных к FC;
- NL_port (Node Loop port) – узловой порт с поддержкой петли. Обеспечивает подключение к FC-концентратору хостов и хранилищ данных.

Порты топологии FC-SW:

- E_Port (Expansion port), порт расширения. Используется для соединения FC-коммутаторов. Может быть соединён только с портом типа E_Port. Когда E_port одного коммутатора соединяется с E_port другого коммутатора, то между коммутаторами организуется межкоммутаторный канал (ISL- interswitch links). ISL является

одним из основных механизмов, обеспечивающих масштабирование СХД;

- F_port (Fabric port) - порт «фабрики» (switched fabric — коммутируемая связная архитектура). Используется для подключения портов типа N_Port к коммутатору. Не поддерживает топологию петли – порт коммутатора, обеспечивающий связь с узловым N_port;
- FL_port (Fabric Loop port) – порт FC-коммутатора, обеспечивающий соединение коммутатора с NL_port FC-концентратора. При наличии такого соединения все NL_ports в структуре FC-AL могут участвовать в коммуникациях по всей сети FC-SW. Такая конфигурация называется открытой петлей. Конфигурация управляемой петли без коммутатора получила название частной петли. Частная петля содержит узлы с портами NL_ports и не имеет портов типа FL_ports;
- EX_port - порт для соединения FC-маршрутизатора и FC-коммутатора. Со стороны коммутатора он выглядит как обычный E_port, а со стороны маршрутизатора это EX_port.
- TE_port (Trunking Expansion port (E_port) - внесен в Fibre Channel компанией CISCO, сейчас принят как стандарт. Это расширенный ISL или EISL. TE_port предоставляет помимо стандартных возможностей E_port маршрутизацию множественных VSANs (Virtual SANs). Это реализовано применением нестандартного кадра Fibre Channel (vsan тегирование).

Универсальные порты:

- G_port (Generic port) – универсальный порт, который может работать как E_port, N_Port или NL_Port. Тип порта определяется автоматически во время инициализации устройства;
- L_Port (Loop port), любой порт устройства с поддержкой топологии «Петля» — NL_port или FL_port.

На рис. 35 показано использование основных FC-портов при построении СХД.

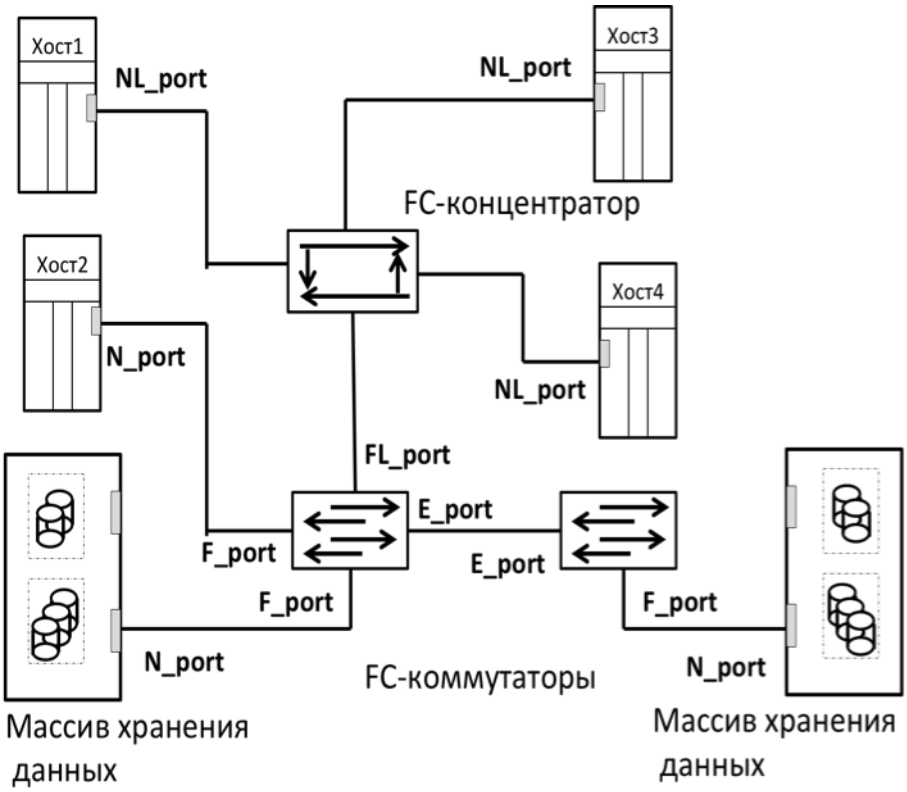


Рис. 35. Типы портов Fibre Channel

4.4. Адресация Fibre Channel

FC-адрес порта назначается динамически при подключении порта к FC фабрике. Формат FC-адреса определяется типом порта подключаемого узла СХД. Это могут быть порты типа N_port и NL_port в открытой петле или NL_port в частной петле.

Формат N_port содержит три восмиразрядных поля (рис.36).

Первое поле определяет идентификатор домена FC-коммутатора. Из 256 возможных значений идентификаторов доменов могут быть использованы только 239 значений, остальные 17 адресов зарезервированы под специальные сервисы FC структур. Так адрес FFFFC зарезервирован для сервера имен, а адрес FFFFFE – для сервиса входа в FC фабрику.

Второе поле – идентификатор области, используется для идентификации группы портов. Примером группы портов является плата коммутатора с несколькими портами.

Третье поле определяет адрес порта в пределах группы портов.

Максимальное число адресов портов типа N_port вычисляется как произведение

239 (доменов) • 256 (областей) • 256 (портов) = 15 663 104.

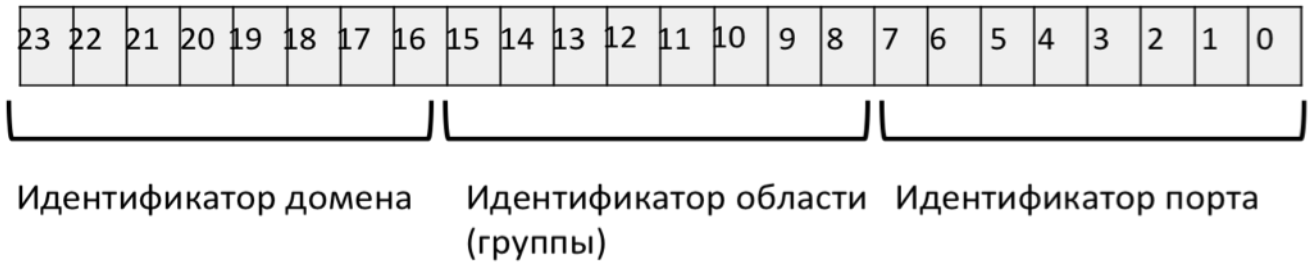


Рис. 36. 24-разрядный FC-адрес порта типа N_port

Формат адресов портов типа NL_port зависит от структуры СХД. Для частной петли два верхних байта не используются (в них заданы нулевые значения), младший байт используется для идентификации физического адреса управляемой петли (рис. 37).

Рис. 37. 24-разрядный FC-адрес порта типа NL_port в частной петле

Если управляемая петля подключается к фабрике коммутаций через FL_port, то она становится открытой. В этом случае NL_port обеспечивает вход в фабрику и два верхних байта адреса используются для идентификации петли. Идентификатор петли одинаков для всех NL_port в данной петле (рис. 38).

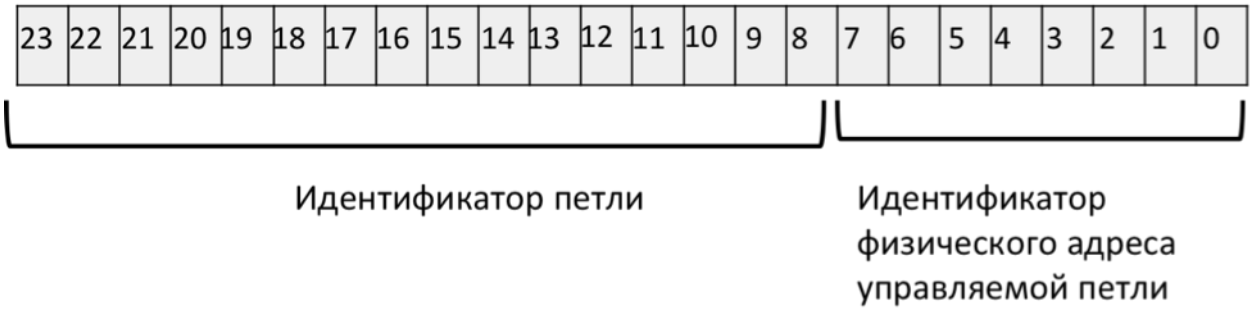


Рис. 38. 24-разрядный FC-адрес порта типа NL_port в открытой петле

4.5. Имена в глобальной сети

Каждому устройству в среде FC назначается уникальный 64-разрядный идентификатор, который получил название имя в глобальной сети (WWN- World Wide Name). Используется два типа имен: имя узла в глобальной сети (WWNN- World Wide Node Name) и имя порта в глобальной сети (WWPN- World Wide Port Name). В отличие от FC-адреса, который назначается динамически, имя устройства в глобальной сети является статическим и уникальным для каждого устройства. Имена назначаются производителям комитетом IEEE и встраиваются в устройство на этапе изготовления. Имена в глобальной сети аналогичны MAC-адресам (Media Access Control) в IP сетях.

На рис. 39 показана структура имени в глобальной сети для устройства хранения данных и для адаптера главной шины

Имя в глобальной сети для устройства хранения данных															
5	0	0	6	0	1	6	0	0	0	6	0	0	1	B	2
0101	0000	0000	0110	0000	0001	0110	0000	0000	0000	0110	0000	0000	0001	1011	0010
	Идентификатор компании 24 бита							Идентификатор модели 32 бита							

Порт

Имя в глобальной сети для адаптера главной шины															
1	0	0	6	0	1	6	0	0	0	6	0	0	1	B	2
0101	0000	0000	0110	0000	0001	0110	0000	0000	0000	0110	0000	0000	0001	1011	0010
	Резерв 12 бит				Идентификатор компании 24 бита						Определяется компанией 24 бита				

Рис. 39. Формат представления имен в глобальной сети

4.6. Управление в среде Fibre Channel

Как правило, структуры FC содержат несколько хостов (Initiator) и несколько единиц оборудования хранения данных (Target), в этих условиях возникает необходимость ограничить влияние некоторых хостов на устройства и подсистемы хранения данных. Ограничение доступа может быть реализовано с использованием механизмов маскирования LUN (Logical Unit Number) и зонирования.

Маскирование LUN

В FC структурах используется следующая система адресации устройств: шина (Bus) – адрес (ID) – подадрес (LUN).

Понятие LUN введено для обеспечения доступа к отдельным устройствам СХД в том случае, когда к одному адресу подключается много устройств. Например, внешние дисковые устройства подключаются к серверу одним кабелем к одному порту. Чтобы различать по этому адресу отдельные устройства и вводится понятие подадреса (LUN). В частности LUN может представлять собой не только отдельные логические диски, но и участки RAID-массивов, которые контроллер представляет операционной среде в качестве отдельного физического диска. Стандарт SCSI-2 поддерживает до 64 LUN на один порт.

С функциональной точки зрения маскирование LUN позволяет определенному хосту получать доступ только к конкретному под устройству хранилища данных, которое обладает определенным значением LUN. И, наоборот, с помощью механизма маскирования LUN можно запретить доступ к определенным LUN для определенных компьютеров и серверов.

Маскирование LUN используется как один из механизмов поддержки целостности данных в среде SAN.

Существует несколько способов обеспечения маскировки LUN. Обычно, маскировка выполняется следующими средствами:

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

Маскирование 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 настроенным BIOS адаптера шины) не «замечает» существования LUN, значения которых не установлены в таблице BIOS.

К недостаткам такого метода следует отнести необходимость проведения корректной настройки адаптера. Кроме того все устройства, 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

Маскирование 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. Маскирование 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 этот способ обычно не используется.

Маскирование LUN контроллерами подсистем хранения данных Fibre Channel

Этот метод маскирования LUN является принудительным для подключенных узлов или требует от узла минимального участия. Маскирование LUN реализуется контроллером подсистемы хранения данных или маршрутизатором (с помощью соответствующей прошивки). Эти устройства настроены на поддержку таблицы имен устройств (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.

Маскирование LUN программным обеспечением узла

Маскирование 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 устройства к LUN.

Такая маскировка может выполняться в виде функции операционной системы или драйвера адаптеров шины.

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.

Зонирование (Zoning)

Основной задачей зонирования является разграничение потоков ввода/вывода информации в сетях хранения данных.

Зонирование можно воспринимать в качестве аналога настройки виртуальных локальных сетей (VLAN) в IP сетях. В виртуальной локальной сети только устройства, входящие в состав одной и той же VLAN "видят" друг друга. Устройства, находящиеся в разных VLAN друг друга не "видят", хотя находятся в той же физической локальной сети.

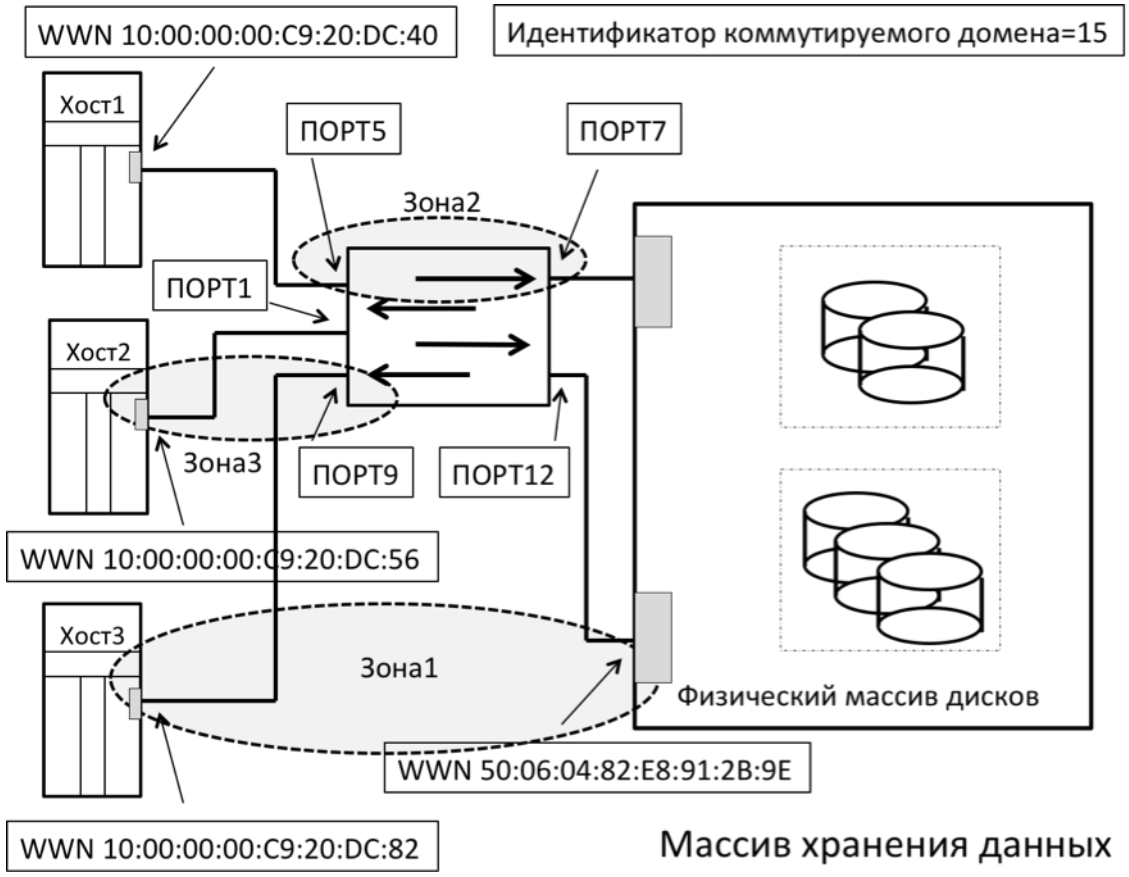
Точно так же зонирование ограничивает возможности компонентов SAN (особенно инициаторов), предоставляя доступ к определенным (входящим в одну и ту же зону) единицам хранения информации, даже если в этой же физической сети хранения данных размещены и другие устройства хранения данных.

Зонирование является ресурсом всей фабрики и конфигурации зон автоматически передаются на все ее коммутаторы.

Выделяют три основных вида зонирования.

1. Зонирование по портам. Для определения зон используются FC-адреса физических портов фабрики. Устройства, включенные в порты, отнесенные к одной зоне, видят друг друга. Доступ к устройствам из других зон невозможен. Такое зонирование называется жестким. Этот метод обеспечивает достаточно высокий уровень безопасности, однако требует перенастройки зонирования в случаях, когда меняется структура фабрики (например, добавляются новые порты). Это связано с тем, что FC-адреса назначаются динамически при регистрации порта в фабрике. Поэтому любое изменение в конфигурации фабрики затрагивает зонирование.

2. Зонирование по имени в глобальной сети. Для формирования зон используются имена устройств в глобальной сети (WWN). Устройства с WWN, отнесенными к одной зоне, видят друг друга вне зависимости от того, в какой порт они подключены. Доступ из других зон возможен, если известен WWN устройства внутри зоны. Основным преимуществом зонирования данного типа является его гибкость. Возможно перевподключение устройств СХД без реконфигурирования информации по зонированию. Это объясняется тем, что WWN устройств являются статичными и определяются заводами изготовителями. Такой метод зонирования называется мягким.



Зона 1 (Зона по имени в глобальной сети) = 10:00:00:00:C9:20:DC:82; 50:06:04:82:E8:91:2B:9E
Зона 2 (Зона по портам) = 15,5; 15,7
Зона 3 (Зона смешанного типа) = 10:00:00:00:C9:20:DC:56; 15,9

Рис. 40. Основные типы зонирования

3. Зонирование смешанного типа. Объединяет способы зонирования предыдущих методов. Позволяет привязать определенный порт FC-структуры к имени узла в глобальной сети.

На рис. 40 показаны три типа зонирования в сети FC.

Зонирование используется для управления доступом серверов к хранилищу данных. Часто зонирование используется совместно с маскированием LUN. Следует подчеркнуть, что это два разных типа реализации контроля доступа – зонирование реализуется на уровне фабрики, а маскирование – на уровне массива хранения данных.

Вопросы для самопроверки

1. Сколько уровней содержит коммуникационный пакет протоколов Fibre Channel?
2. Назовите базовые топологии использования компонентов SAN ?
3. Чем отличаются порты узлов от портов топологии в составе систем хранения данных типа SAN ?
4. Как определяется тип универсального порта в составе систем хранения данных типа SAN ?
5. Каково основное назначение операций маскирования LUN в составе систем хранения данных типа SAN ?
6. Назовите основные виды зонирования в составе систем хранения данных типа SAN?

5. ОБЛАЧНЫЕ СХД

5.1. Общая структура облачных СХД

Облачные системы хранения данных – это метод организации хранилищ данных, при котором данные, принадлежащие предприятию, хранятся не в центре обработки данных (ЦОД) или на отдельных серверах этого предприятия, а на некотором множестве виртуальных серверов. Причем эти виртуальные сервера принадлежат компаниям, сдающим в аренду или продающим пользователям доступное дисковое пространство. Доступ к данным, которые хранятся в облачных СХД обеспечивается из любого места, где есть доступ к сети Интернет.

Общая структура взаимодействия пользователей с облачными системами хранения данных представлена на рис. 41.

Такие решения очень привлекательны для малых и средних предприятий, которым необходимо обеспечивать обработку и хранение больших объемов данных. В этом случае потребители виртуальных услуг оплачивают лишь используемый объем дискового пространства. Им не приходится оплачивать аппаратное обеспечение, обслуживание ЦОД и другие накладные расходы по содержанию центра обработки данных.

Рис. 41. Общая структура организации облачных систем хранения данных

В общем случае облачные вычисления (облачные СХД можно рассматривать как частный случай реализации облачных вычислений) — это модель предоставления по требованию пользователя сетевого доступа к совместно используемому пулу конфигурируемых вычислительных ресурсов (например, сетей, серверов, систем хранения, приложений и сервисов), которые могут быть предоставлены с минимальными усилиями по управлению и минимальным взаимодействием с поставщиком услуг.

Под сетевым доступом понимается обеспечение доступности сервисов облачных вычислений по сети посредством стандартных механизмов, поддерживающих использование гетерогенных платформ (таких как мобильные телефоны, ноутбуки, стационарные ПК, мониторы со встроенным «тонким клиентом»).

Пул ресурсов — Это совокупность вычислительных ресурсов поставщика услуг, которые объединяются в группы для преставления различным потребителям в рамках многопользовательской модели, при этом физические и виртуальные ресурсы могут назначаться и переназначаться в соответствии с потребностями клиентов. Потребитель сервиса как правило не знает и не контролирует точное физическое расположение предоставляемых ресурсов, но может на более высоком уровне абстракции специфицировать их размещение.

Возможны следующие модели представления услуг облачных вычислений.

Программное обеспечение как услуга (Software as a Service, SaaS). Потребителю предоставляется возможность использования приложений поставщика, работающих в облачной инфраструктуре. Приложения доступны с различных клиентских устройств посредством тонких клиентов, таких как веб-браузер или специализированное клиентское приложение. Потребитель не управляет и не контролирует используемую облачную инфраструктуру, включая сети, серверы, операционные системы, системы хранения и даже некоторые параметры программных приложений, исключением может являться предоставление потребителю возможности управления ограниченным набором пользовательских настроек приложения.

Платформа как услуга (Platform as a Service, PaaS). Потребителю предоставляется возможность развертывания в облачной инфраструктуре собственных приложений, использующих базовую архитектуру (языки программирования, библиотеки, инструментарий) поддерживаемую поставщиком. Потребитель не управляет и не контролирует используемую базовую облачную инфраструктуру (сети, серверы, операционные системы, системы хранения), но управляет развернутыми приложениями и, возможно, рядом системных настроек, связанных с функционированием приложения.

Инфраструктура как услуга (Infrastructure as a Service, IaaS). Потребителю предоставляется возможность получения базовых ресурсов, таких как мощности систем хранения, вычислительные ресурсы, сетевые ресурсы и другие базовые вычислительные ресурсы которые могут использоваться потребителем для работы произвольного программного обеспечения, включая операционные системы и приложения. Потребитель не контролирует и не управляет базовой облачной инфраструктурой, но получает управление над операционными системами, предоставленными ресурсами систем хранения, приложениями и в некоторых случаях ограниченным набором сетевых ресурсов (например, локальный сетевой экран или виртуальный коммутатор).

5.2. Виртуализация ресурсов вычислительных систем

Одним из основных механизмов, обеспечивающих возможность предоставления услуг облачного хранения данных, является виртуализация ресурсов вычислительных систем.

Виртуализация (в области серверов) это совокупность программно-аппаратных средств, позволяющих на логическом уровне отделить вычислительные ресурсы системы от ее аппаратной части.

Обычно на одном сервере может работать только одна операционная система, управляющая работой данного сервера. Все вычислительные ресурсы этого сервера передаются этой операционной системе.

При виртуализации используются программные средства (программная прослойка, среда виртуализации), которые эмулируют заданную часть вычислительных ресурсов сервера в виде изолированного контейнера, представляющего собой виртуальную вычислительную машину (рис.42). Таких контейнеров на сервере может быть несколько и в каждом из них может быть установлена своя операционная система. Эта программная прослойка (программная среда для создания виртуальных серверов) получила название гипервизора или монитора виртуальных машин.

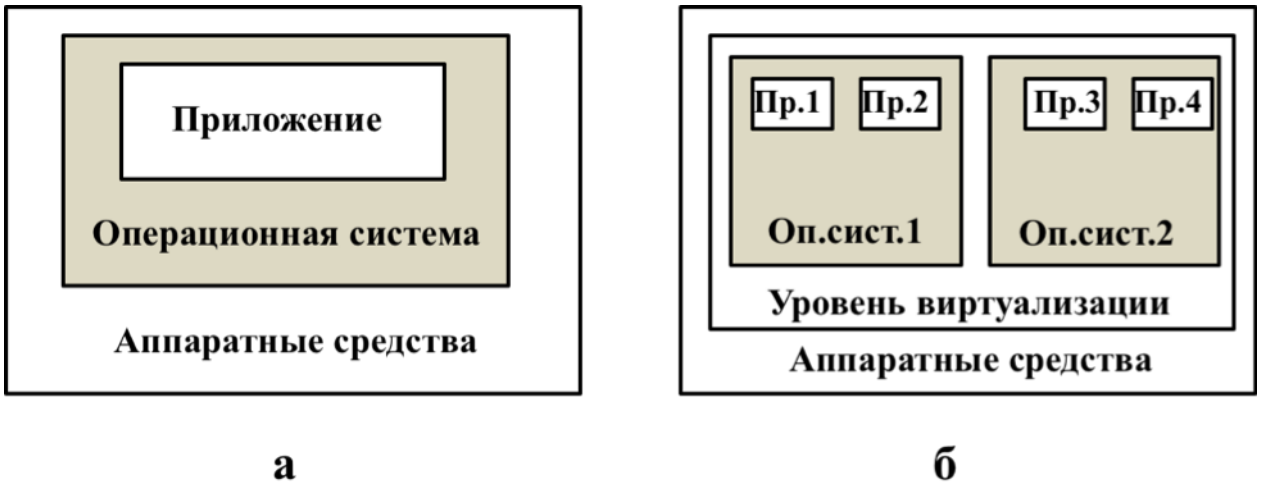


Рис.42. Виртуализация сервера (а- сервер до виртуализации, б- сервер после виртуализации)

В качестве дисковых ресурсов, которые представляются гипервизором для использования операционными системами в контейнерах, обычно используются как дисковые ресурсы локальных физических серверов, так и дисковые ресурсы внешних систем хранения данных.

Сетевая виртуализация систем хранения данных осуществляется с использованием внеполосной и внутриполосной технологий. При использовании внеполосной технологии конфигурация виртуализированной среды хранится в отдельном устройстве вне каналов передачи данных. Эта конфигурация называется также системой с передачей по разделенному каналу. В этом случае происходит разделение канала управления и канала передачи данных (канал управления проходит через устройство виртуализации, а канал передачи данных – не проходит).

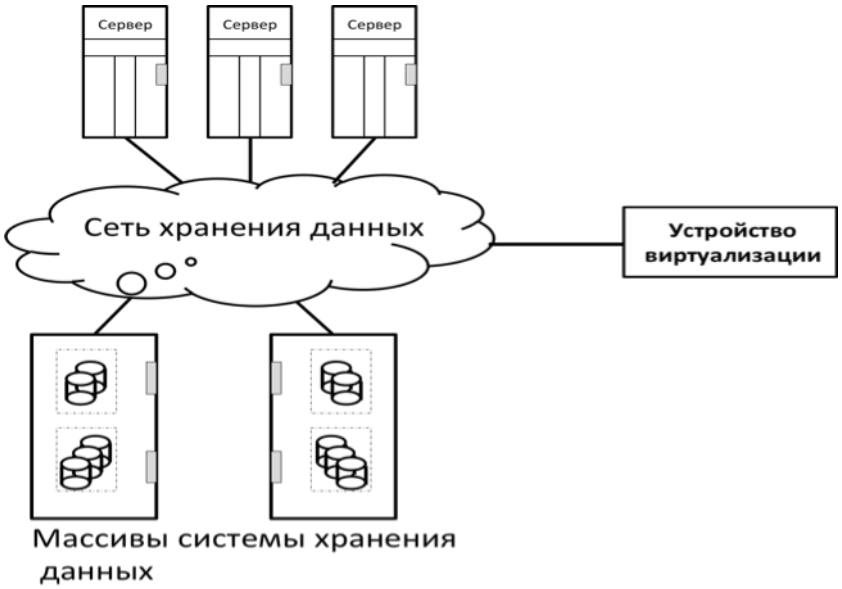


Рис.43. Конфигурация внеполосной виртуализации системы хранения данных

Такая конфигурация (рис. 43) позволит обрабатывать данные с сетевой скоростью при минимальной задержке, необходимой для преобразования информации о виртуальной конфигурации в информацию о физическом устройстве.

Рис.44. Конфигурация внутриполосной виртуализации системы хранения данных

При внутриполосной конфигурации виртуализация осуществляется в канале данных, серверами и устройствами общего назначения (рис. 44). В процессе обработки данных в этом случае пакеты часто кэшируются, чтобы затем после преобразования виртуальной информации их можно было отпра-

вить в соответствующую область физических устройств. Возникает дополнительная задержка времени отклика для приложения, связанная с пребыванием данных в Кэш-памяти перед их отправкой на физический диск.

В качестве примера реализации внеполосной виртуализации можно привести устройство EMC Invista, функционирующей на базе интеллектуальных маршрутизаторов SAN. Основные компоненты Invista и способ подключения к SAN приведены на рис. 45.

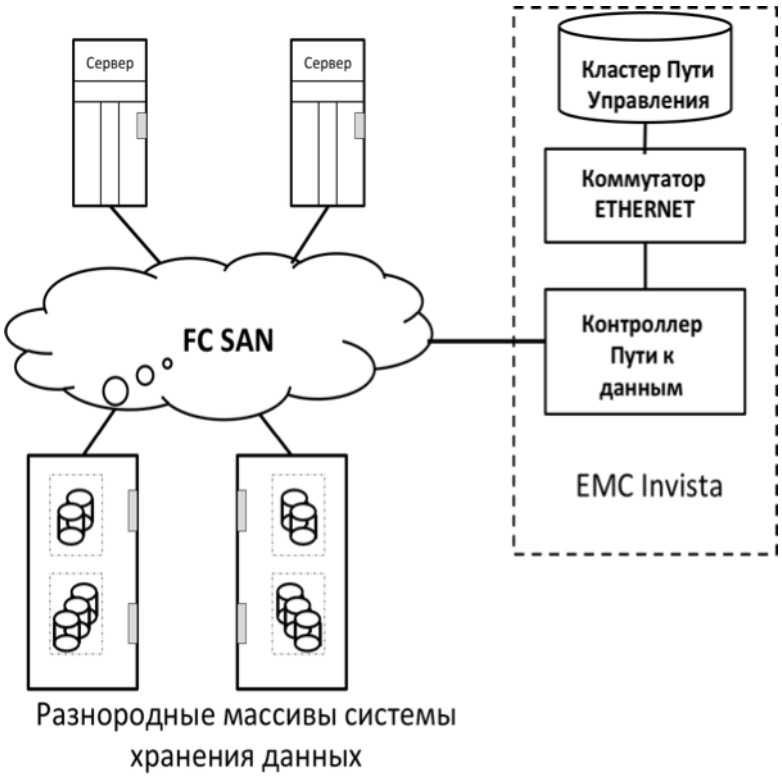


Рис.45. Основные компоненты устройства виртуализации EMC Invista

Кластер пути управления представляет собой специальное запоминающее устройство, обеспечивающее запуск программы Invista и содержащее информацию о параметрах конфигурации Invista. Кроме того блок кластера пути обеспечивает выполнение основных функций контроля и управления виртуальными запоминающими устройствами.

Коммутатор сети Ethernet обеспечивает доступ к основным устройствам Invista для их конфигурирования и контроля трафика.

Контроллер пути к данным выполняет функции маршрутизатора для системы SAN, обеспечивая корректное выполнение операций ввода/вывода данных.

Вопросы для самопроверки

7. В чем отличие внеполосной технологии виртуализации от внутриполосной?

8. Назовите базовые технологии виртуализации информационно-вычислительных систем.
9. Назовите основные задачи, решаемы при виртуализации серверов.
10. Какие возможности получает пользователь при использовании облачных вычислений согласно модели "Платформа как услуга"?
11. Какие возможности получает пользователь при использовании облачных вычислений согласно модели "Программное обеспечение как услуга"?
12. Какие возможности получает пользователь при использовании облачных вычислений согласно модели "Инфраструктура как услуга"?
13.

ЗАКЛЮЧЕНИЕ

Сегодня системы хранения данных работают во многих организациях по всему миру. В основном это организации, операционная деятельность которых предъявляет чрезвычайно высокие требования к доступности и производительности приложений, работающих с данными. Таковыми являются финансовые организации, для которых важны как производительность (поскольку задержки в функционировании основных систем могут стоить миллионы долларов), так и надежность (поскольку потеря важной финансовой информации может повлечь за собой еще более тяжелые последствия). Другим важнейшим потребителем систем хранения данных являются телекоммуникационные компании, где необходимо реализовать биллинг огромного количества информации в пределах заданного времени и, следуя нормативным актам регулирующих органов, надежно сохранять эту информацию длительный период времени.

Следует подчеркнуть, что в учебном пособии рассмотрены лишь основные вопросы реализации систем хранения данных.

В дальнейшем при изучении проблем построения современных систем хранения данных следует обратить внимание на широкое использование в серверах и СХД корпоративного класса устройств долговременного хранения данных на базе флэш-памяти.

СПИСОК ЛИТЕРАТУРЫ

1. От хранения данных к управлению информацией / EMC Education Services ; ред. К. Белецкая [и др.] ; пер. с англ. В. Воротинцев, О. Гринвуд. - СПб. : Питер , 2010. - 522 с.

2. Джордж Джад. Основы проектирования SAN. Из-во "Edvance Edition", М. -2008, с. 589
3. Марк Фарли. Сети хранения данных. Из-во ЛОРИ, 2004

Губин Александр Николаевич
Филиппов Феликс Васильевич

СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ

Учебное пособие

Редактор

План 2013 г., п.

Подписано к печати
Объем усл.-печ. л. Тираж экз. Заказ

Издательство СПбГУТ. 191186 СПб., наб. р. Мойки, 61

Отпечатано в