



第10章 大容量存储系统



第10章 大容量存储系统

10.1 大容量存储结构概述



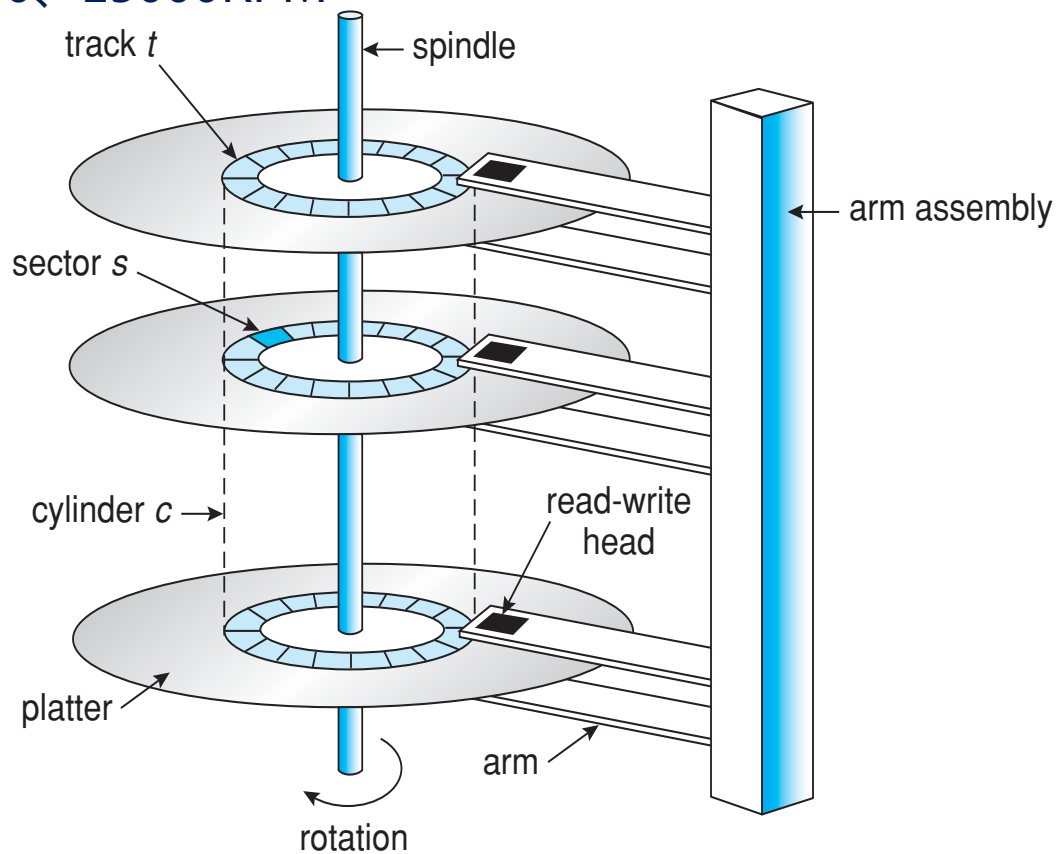
大容量存储结构概述

- 文件的存储设备主要有磁带、磁盘、光盘等。
- 存储设备的特性可以决定文件的存取方法。
- 以磁带为代表的顺序存取设备
- 以磁盘为代表的直接存取设备

磁盘机理

■ 磁盘

- 为现代计算机提供了大容量的二级存储
- 普通驱动器转速60—250次/秒，按每分钟转速为5400、7200、10000、15000RPM





磁盘工作机理

- 磁盘访问时间
 - 传输速率(Transfer rate) : MBps级
 - 指从磁盘上读出数据或向磁盘写入数据所经历的时间
 - 定位时间(Positioning time, random-access time):ms级
 - 寻道时间Seek Time : 移动磁臂到所需柱面/磁道的时间, 由启动磁臂时间和磁头移动多条磁道的时间构成
 - 旋转延迟RPM : 所需扇区旋转到磁头下的时间, 平均旋转延迟时间是每转所需时间的一半
- 磁头碰撞 : 磁头和盘面接触的破坏结果
- 驱动器通过I/O总线与计算机相连
 - BUS : EIDE, ATA, SATA, USB, Fiber Channel, SCSI, SAS, Firewire
 - 主机控制器 : 总线与磁盘控制器对话的部件
 - 磁盘控制器 : 磁盘驱动器内部的部件

磁盘工作机理

- 盘面大小 .85" 到 14"
 - 通常的 3.5" , 2.5" , and 1.8"
- 每个驱动器大小 30GB 到 4TB
- 性能
 - 传输速率：理论 6 Gb/sec
 - 有效传输速率：实际 1Gb/sec
 - 寻道时间：3ms 到12ms , 台式机硬盘9ms
 - 平均寻道时间一般是磁道总数的1/3(统计值)
 - 旋转延迟：基于转轴速度, $1/(RPM/60)=60/RPM$
 - 平均旋转延迟：1/2最大延迟

(From Wikipedia)

Spindle [rpm]	Average latency [ms]
4200	7.14
5400	5.56
7200	4.17
10000	3
15000	2

磁盘性能

- 访问延迟 = 平均访问时间 = 平均寻道时间 + 平均旋转延迟
 - 如上例子，最快： $3\text{ms} + 2\text{ms} = 5\text{ms}$
 - 较慢 $9\text{ms} + 5.56\text{ms} = 14.56\text{ms}$
- 平均I/O时间 = 平均访问时间 + (总传输量 / 传输速率) + 控制器负载时间
- 例：在7200RPM磁盘上，传输4KB数据块，平均寻道时间5ms，传输速率1Gbps，控制器负载0.1ms，则
 - $5\text{ms} + 4.17\text{ms} + 0.1\text{ms} + \text{传输时间}$
 - 传输时间 = $4\text{KB} / 1\text{Gbps} * 8 = 0.0305\text{ms}$
 - 平均I/O时间为9.3005ms

第一个商业硬盘



1956
IBM RAMDAC计算机
采用 IBM Model 350
disk storage system

存储大小: 5M (7 bit)
盘面尺寸: 50 x 24"
访问时间 = < 1s



固态硬盘SSD

- 非易失存储：
 - 有多种版本，DRAM、Flash、SLC、MLC等
- 比HDD更可靠，但是相对更贵
- 寿命更短，但更快
- 采用专用高速接口直接连接NVMe、PCIe
- 没有移动部件、没有寻道时间、没有旋转延迟



磁带

- 曾经是早期的二级存储
- 相对永久，且容量很大
- 访问时间非常慢
 - 随机访问，比磁盘慢1000倍
- 主要用于备份不常用信息，也用于系统之间信息传输的媒介
- 磁带绕在轴上，向前转或向后转，并读写磁头
- 移动磁带到正确位置约几分钟，传输速率140MBps
- 一般存储容量：200GB 到 1.5TB



第10章 大容量存储系统

10.2 磁盘特性



磁盘结构

- 磁盘驱动器被视为逻辑块的大一维数组
 - 逻辑块是最小的传输单元
 - 低级格式化能够在物理介质上创建逻辑块
- 逻辑块一维数组被顺序地映射到磁盘扇区
 - 扇区0是最外面柱面的第一个磁道的第一个扇区。
 - 映射顺序：先按照磁道内扇区顺序，再按照柱面内磁道顺序，再按照从外到内的柱面顺序进行。
 - 逻辑到物理的地址转换并不简单
 - 磁道中可能存在坏扇区
 - 每个磁道扇区量并非常量：恒定线速度 or 恒定角速度



磁盘连接

- 主机连接存储：通过I/O bus与I/O端口通信来访问存储
- SCSI总线：
 - 一条I/O总线最多支持16个设备，ATA只支持2个设备
 - 1个控制器卡，即SCSI引导器
 - 15个存储设备，即SCSI目标
 - 每个SCSI目标提供访问8个逻辑单元能力，
- 光纤通道FC
 - 高速串行体系
 - 具有24位地址空间用于交换，是SAN(storage area networks)的基础技术

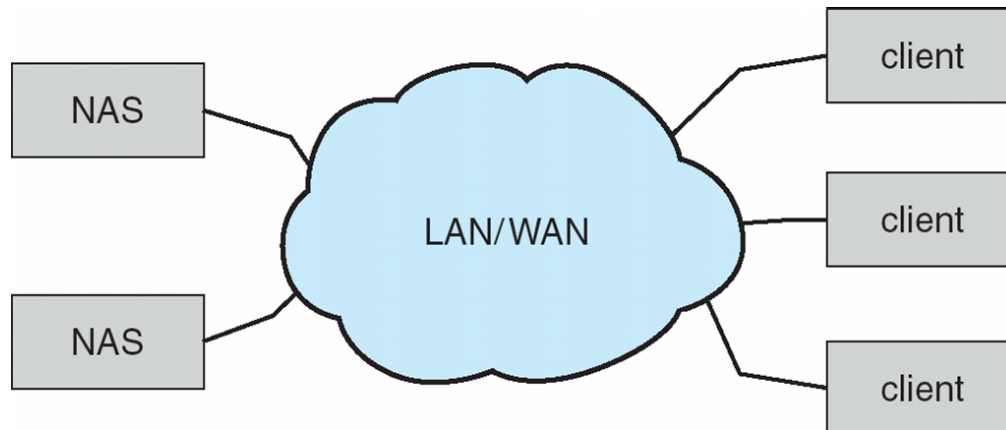


存储矩阵(Storage Array)

- 提供连接多个磁盘和磁盘矩阵
- 存储矩阵拥有控制器
 - 端口，用于连接主机到矩阵
 - 内存，用于运行控制软件
 - 可连接大量的磁盘
 - RAID机制，热交换区等
 - 提供共享存储
 - 提供一些特性：快照、备份、复制等

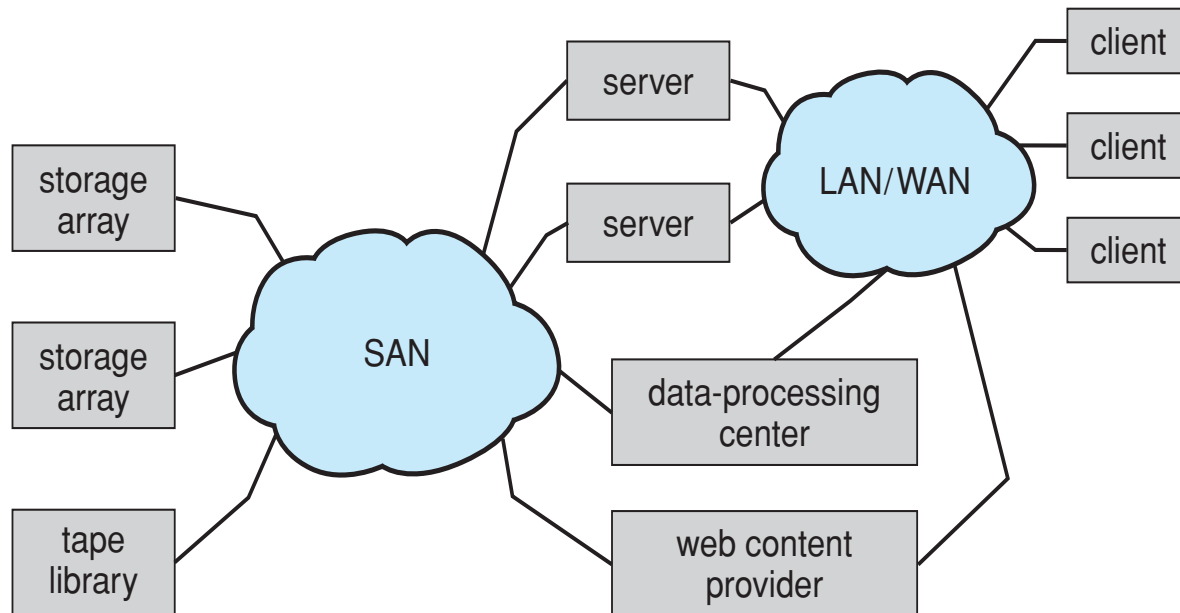
网络连接存储NAS

- NAS是通过网络进行连接的存储
- 典型协议：NFS、CIFS
- 通过远程过程调用，基于TCP、UDP实现
- iSCSI协议可以基于Internet实现SCSI协议



存储区域网络(SAN)

- 适用于更大存储环境
- 采用私有存储协议而非网络协议建立连接
- 提供多个主机连接多个存储矩阵





第10章 大容量存储系统

10.3 磁盘调度



磁盘调度需求

- 操作系统负责高效地使用硬件
 - 对于磁盘，需要更快的访问时间和磁盘带宽
- 目标：最小化寻道时间
- 寻道时间 \approx 寻道距离
- 磁盘带宽 = 传输量的总和 / (最后完成传输时间 - 第一个请求服务时间)



1. 先来先服务--FCFS

- 先来先服务算法按进程请求访问磁盘的先后次序进行调度。
- 特点：简单合理，但未对寻道进行优化。



先来先服务算法例

下一磁道号	移动距离
55	45
58	3
39	19
18	21
90	72
160	70
150	10
38	112
184	146

从100号磁道开始，磁盘访问请求为：55、58、39、18、90、160、150、38、184

平均寻道长度为：55.3



2. 最短寻道时间优先--SSTF

- 最短寻道时间优先算法选择与当前磁头所在磁道距离最近的请求作为下一次服务的对象。
- 特点：寻道性能比FCFS好，但不能保证平均寻道时间最短，还可能会使某些请求总也得不到服务。



最短寻道时间优先算法例

下一磁道号	移动距离
90	10
58	32
55	3
39	16
38	1
18	20
150	132
160	10
184	24

从100号磁道开始，磁盘访问请求为：55、58、39、18、90、160、150、38、184

平均寻道长度为：27.6



3. 扫描算法--SCAN

- 进程饥饿：进程的请求总也得不到服务。
- SCAN算法：在磁头当前移动方向上选择与当前磁头所在磁道距离最近的请求作为下一次服务的对象。**磁头方向可以双向移动。当走到尽头才掉头。**因这种算法中磁头移动规律颇似电梯的运行，故又称为**电梯调度算法(*)**。
- 特点：具有较好的寻道性能，能避免进程饥饿，但不利于两端磁道的请求。



扫描算法例

下一磁道号	移动距离
150	50
160	10
184	24
90	94
58	32
55	3
39	16
38	1
18	20

从100号磁道开始，向磁道号增加方向移动。磁盘访问请求为：55、58、39、18、90、160、150、38、184（假设磁头最远为184）

平均寻道长度为：27.8



4. 循环扫描算法CSCAN

- CSCAN算法是SCAN算法的改良，它规定磁头单向移动。例如，自里向外移动，当磁头移到最外磁道时立即又返回到最里磁道，如此循环进行扫描。
- 特点：该算法消除了对两端磁道请求的不公平。



循环扫描算法例

下一磁道号	移动距离
150	50
160	10
184	24
18	166
38	20
39	1
55	16
58	3
90	32

从100号磁道开始，向磁道号增加方向移动。磁盘访问请求为：55、58、39、18、90、160、150、38、184

平均寻道长度为：35.8



5. Look & C-Look算法

- SCAN & CSCAN的改良
- 磁头不再走到磁道尽头，而是走到请求的尽头即掉头。
- 注意：在一些国内教材，包括考研时，SCAN & CSCAN是不走到磁道尽头的，且不介绍Look & C-look算法，这时候，就默认SCAN&CSCAN，为只走到请求的尽头即掉头



6. N-Step-SCAN

- 若多个进程反复请求对某一磁道的访问，则磁臂可能停留在某处不动，这一现象称为**磁臂粘着**。
- N-Step-SCAN算法：
 - 将磁盘请求队列分成若干个长度为N的子队列
 - 磁盘调度按FCFS算法依次处理这些子队列
 - 处理每个队列时按SCAN算法进行，一个队列处理完后，再处理其他队列。



7. FSCAN算法

- FSCAN算法是N-Step-SCAN算法的简化
- 它只将磁盘请求队列分成两个子队列。
 - 一个是当前所有请求磁盘I/O的进程形成的队列，由磁盘调度按SCAN算法进行处理，
 - 另一个队列则是在扫描期间新出现的磁盘请求。



8.如何选择磁盘调度算法

- SSTF 较为普遍且常用
- SCAN 和 C-SCAN 对于磁盘负荷较大的系统有优势，可以避免饥饿问题
- 对于任意算法，性能依赖于请求的数量与类型
- 请求可能会被文件分配所影响：连续分配、索引分配
- 由于以上复杂性，磁盘调度算法应可替换。SSTF和LOOK时较为合理的默认算法
- 调度除了寻道，还应该考虑旋转时间，但是OS很难计算，往往由控制器直接优化。
- 服务请求可能因应用上下午存在优先级问题，OS需要进一步考虑



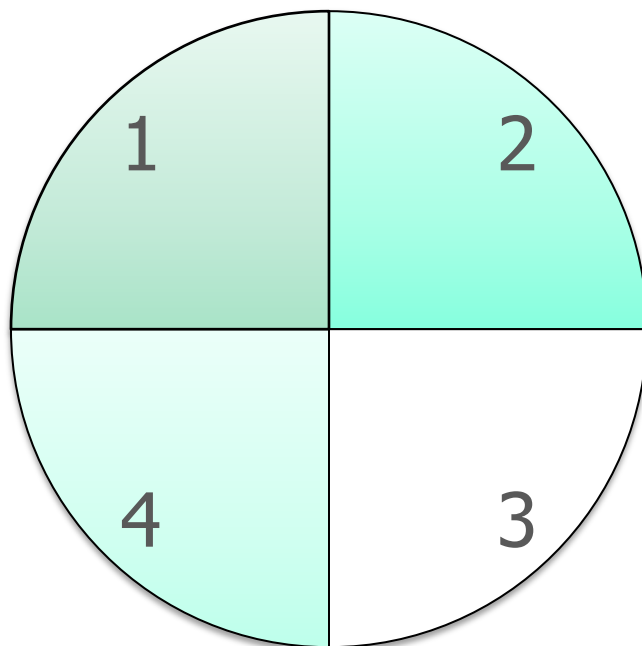
第10章 大容量存储系统

10.4 磁盘存储优化分布

磁盘存储优化分布

- 考虑磁道保存4个记录的旋转型设备，假定收到四个I/O请求。假设每周为20ms。假定平均用1/2周定位，1/4周读出记录。

记录信息



■ 位置1 □ 位置2 ■ 位置3 ■ 位置4



多种I/O请求排序方法

■ 情形1

- I/O请求次序为读记录4、3、2、1，初始位置未知，考虑一般情况，则平均用 $1/2$ 周定位，再加上 $1/4$ 周读出记录。
- $1/2 + 1/4 + 3 \times 3/4 = 3$ 周，即60毫秒。

■ 情形2

- 如果果次序为读记录1、2、3、4。
- 总处理时间等于 $1/2 + 1/4 + 1/4 + 1/4 + 1/4 = 1.5$ 周，即30毫秒。

■ 情形3

- 如果知道当前读位置是记录4，可采用次序为读记录4、1、2、3。总处理时间等于1周，即20毫秒

优化分布问题

- 考虑10个逻辑记录A, B....., J, 被存于旋转型设备上, 每道存放10个记录, 安排如下:

物理块

1-10

逻辑纪录

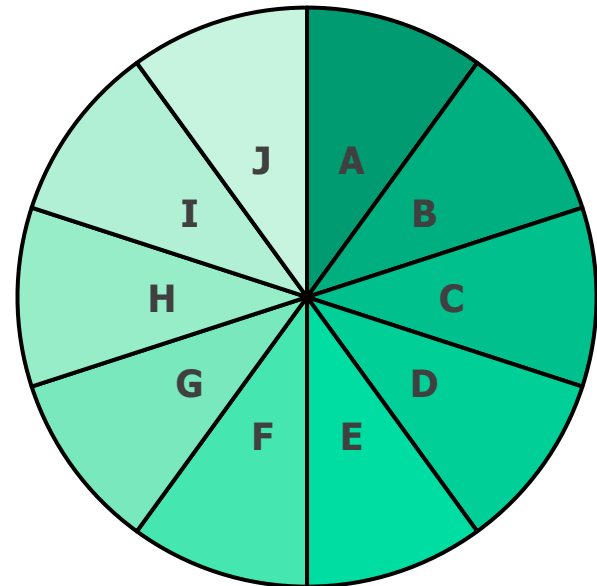
A-J (A B C D E))

- 设磁盘匀速转动, 旋转一周的时间为20毫秒, 读出时间为2毫秒, 处理时间为4毫秒
- 按A B C D EJ的顺序, 处理10个记录的时间为多少?

信息简单顺序分布

- 处理10个记录的总时间：
10ms(移动到记录A的平均时间)
+ 2ms(读记录A)
+ 4ms(处理记录A)
+ $9 \times [16\text{ms}(\text{访问下一记录}) + 2\text{ms}(\text{读记录}) + 4\text{ms}(\text{处理记录})] = 214\text{ms}$

扇区信息分布



如果按照下面方式对信息优化分布

物理块

逻辑纪录

1

A

2

H

3

E

4

B

5

I

6

F

7

C

8

J

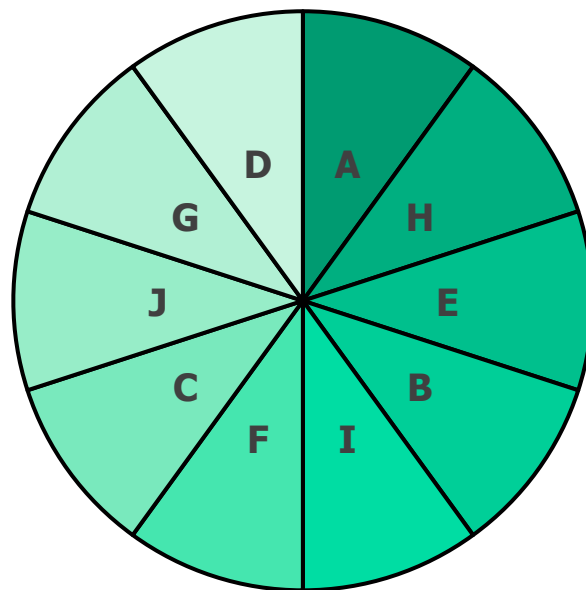
9

G

10

D

扇区信息分布



10ms(移动到记录A的平均时间)
+10× 2ms (读记录)
+10× 4ms (处理记录)]
=70毫秒



第10章 大容量存储系统

10.5 磁盘管理



磁盘管理

- 低级格式化：将磁盘分成多个扇区，以满足磁盘控制器的读写
 - 每一个分区都具有头信息、数据和纠错码
- 为了使用磁盘存储文件，OS需要将自己的数据写到磁盘上
 - 逻辑分区：将磁盘分成一个或者多个柱面组成的分区，可将每个分区作为独立的逻辑磁盘
 - 逻辑格式化：即创建文件系统，将初始的文件系统数据结构存储到磁盘上。
 - 操作单位：
 - 磁盘I/O：多扇区为单位block
 - 文件I/O：分簇cluster

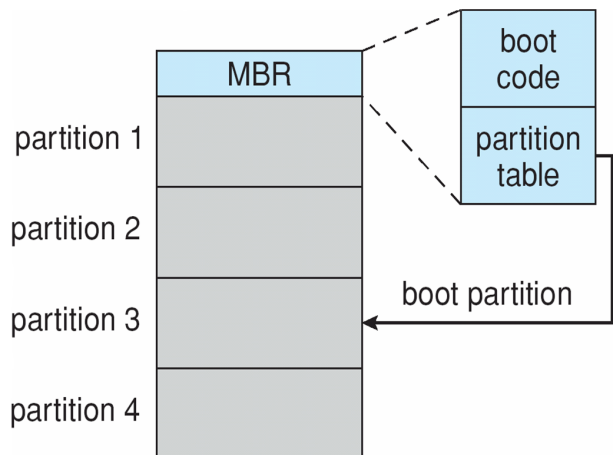
磁盘管理

- Raw disk:

- APP试图建立自己独特的分区管理而创建的特殊磁盘格式，只是一个逻辑块的大数组，如数据库

- 引导块

- 最基本的Bootstrap被存储在计算机ROM中
- 扩展的bootstrap存储在引导分区的引导块中



- 坏块

- 格式化时检查坏扇区，并建立坏快链表，可用备用块提代坏块

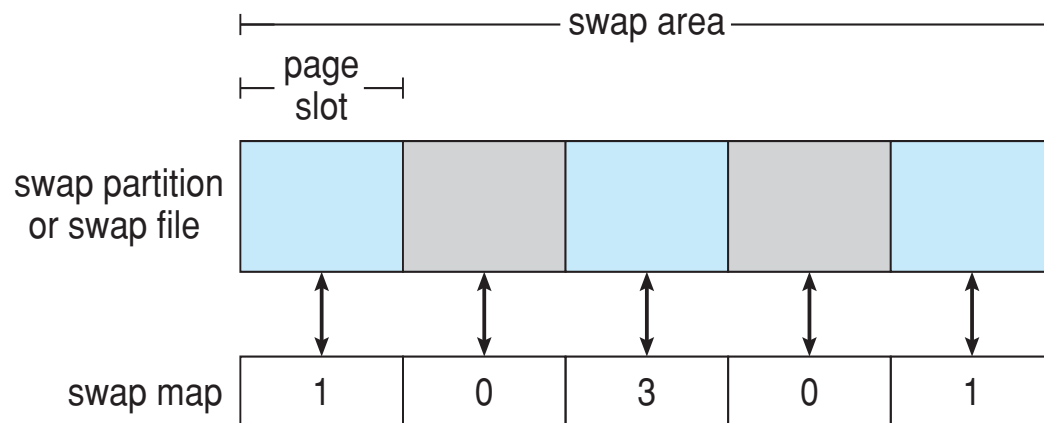


第10章 大容量存储系统

10.6 交换空间管理

交换空间管理

- 交换空间：虚拟内存所使用的磁盘空间
- 交换空间可在普通文件系统中创建，也可以是一个独立的磁盘分区
 - 文件系统简单大文件：
 - 创建简单，但访问效率低
 - 依赖文件系统的已有数据结构访问，带来更多的磁盘访问
 - 可能产生外部碎片，带来更长交换时间
 - 独立的磁盘分区：
 - 不需文件系统，
 - 管理起可以用适当算法进行优化。
 - 交换分区的访问比文件系统访问更频繁，使得内部碎片存在时间短





第10章 大容量存储系统

10.7 磁盘容错技术



磁盘容错技术

- 容错技术：通过在系统中设置冗余部件来提高系统可靠性的一种技术。
- 磁盘容错技术：通过增加冗余磁盘驱动器、磁盘控制器等方法来提高磁盘系统可靠性的一种技术。也称为系统容错技术。
- 系统容错分为三级：
 - 第一级容错技术：低级磁盘容错技术
 - 第二级容错技术：中级磁盘容错技术
 - 第三级容错技术：高级系统容错技术



第一级容错技术

- 第一级容错技术
 - 最基本的一种磁盘容错技术
 - 主要用于防止因磁盘表面缺陷所造成的数据丢失。
- 它包含
 - 双份目录
 - 双份文件分配表
 - 写后读校验等措施。



双份目录和双份文件分配表

- 目录和文件分配表是文件管理的重要数据结构，为防止它们被破坏，可在不同的磁盘上或在磁盘的不同区域中建立双份目录和文件分配表，一份称为主目录或主文件分配表，另一份称为备份文件目录及备份文件分配表。
- 一旦主目录或主文件分配表被破坏，则启用备份文件目录及文件分配表。
- 系统启动时也要对两份数据结构进行检查，以验证它们的一致性。



热修复重定向和写后读校验

- 当磁盘出现较少缺陷时，可采用以下两种补救措施：
 - 热修复重定向：将磁盘中的一部分作为热修复重定向区，用于存放当发现磁盘有缺陷时的待写数据，并对写入该区的所有数据进行登记，便于以后对数据进行访问。
 - 写后读校验：每次向磁盘中写入一个数据块后又立即从磁盘上读出该数据块，与写入数据进行比较，若相同则写下一块，否则重写。若重写后仍不一致，则认为该盘块有缺陷，此时便应将该块数据写入热修复重定向区。

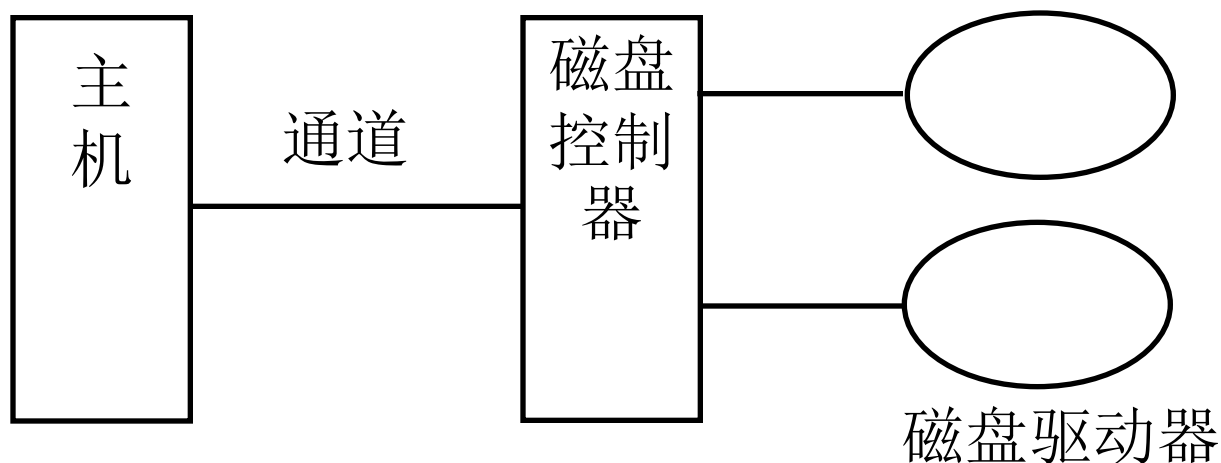


第二级容错技术

- 第一级容错技术只能用于防止由磁盘表面部分故障造成的数据丢失。若磁盘驱动器发生故障，则应采用第二级容错技术。
- 第二级容错技术主要用于防止由磁盘驱动器及磁盘控制器故障所导致的系统不能正常工作。

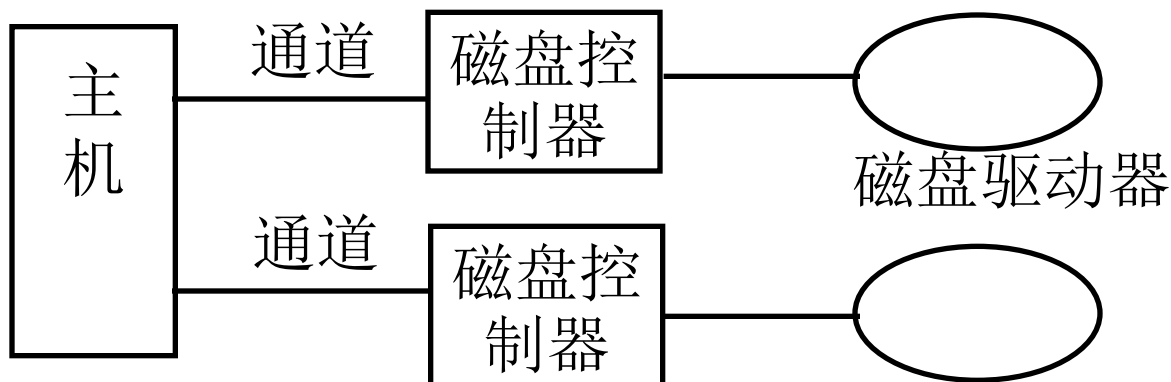
磁盘镜像

- 磁盘镜像：在同一磁盘控制器下，再增设一个完全相同的磁盘控制器。
- 在每次向主磁盘写入数据后，都采用写后校验方式，将数据再同样写到备份磁盘上，使两个磁盘上有完全相同的位像图。当主磁盘发生故障时，启用备份磁盘并发出警告。



磁盘双工

- 磁盘双工：将两台磁盘驱动器分别接到两个磁盘控制器上，同样使这两台磁盘驱动器镜像成对。
- 在磁盘双工时，文件服务器同时将数据写到两个处于不同控制器下的磁盘上，使两者有完全相同的位像图，如果其中的一台磁盘发生故障，另一台仍然可以工作，同时发出警告。





第三级容错技术

- 第三级系统容错是在提供一、二级容错的基础上，提供文件服务器镜像功能。
- 主服务器与从服务器是配置完全相同的两台计算机。每台服务器除了按常规加插网卡外，还需插入一块镜像服务器接口卡，然后用光缆将两块镜像服务器接口卡连接起来。
- 主服务器是当前正在为工作站提供网络服务的服务器。系统自动将主服务器的内存和硬盘中的数据复制到从服务器。当主服务器发生故障时，从服务器成为网中的主服务器，使网络不受影响地正常工作。当故障排除后，两台服务器重新同步。

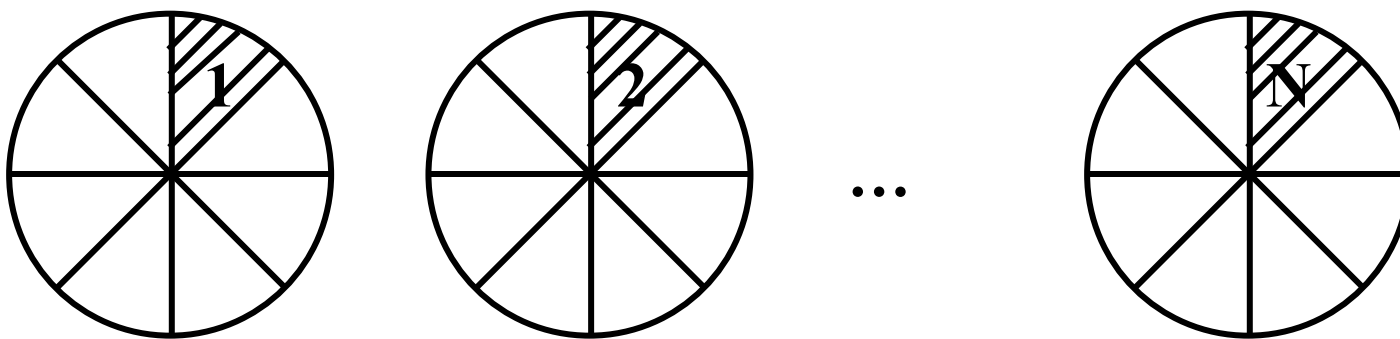


独立磁盘冗余阵列（RAID）

- 独立磁盘冗余阵列 (Redundant Array of Independent Disk, RAID)是利用一台磁盘阵列控制器来统一管理和控制一组磁盘驱动器，组成一个高速可靠的，快速的大容量磁盘系统。也称为廉价磁盘冗余阵列。
- 在RAID中，数据通过分拆均匀地写到每一个驱动器上。
- RAID提供了与磁盘双工及磁盘镜像类似的冗余。冗余级别取决于RAID级别。

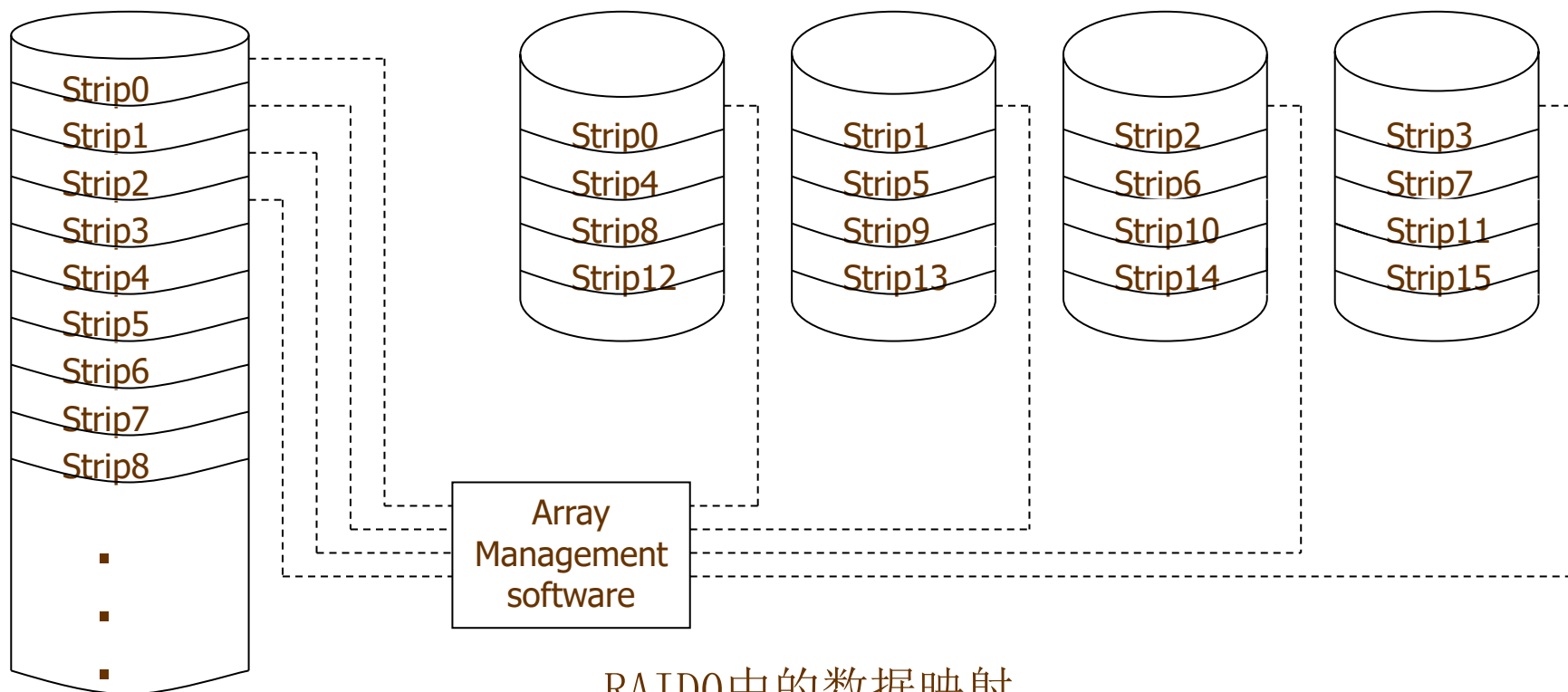
并行交叉存取

- 在并行交叉磁盘存储系统中，有多台磁盘驱动器，系统将每一盘块中的数据分为若干个子盘块的数据，再把每一子盘块的数据分别存储到各个不同磁盘中的相同位置。以后要读写数据时，采取并行传输方式，将各个盘块中的子盘块数据同时向内存中传输，从而使传输时间大大减少。



RAID 0级

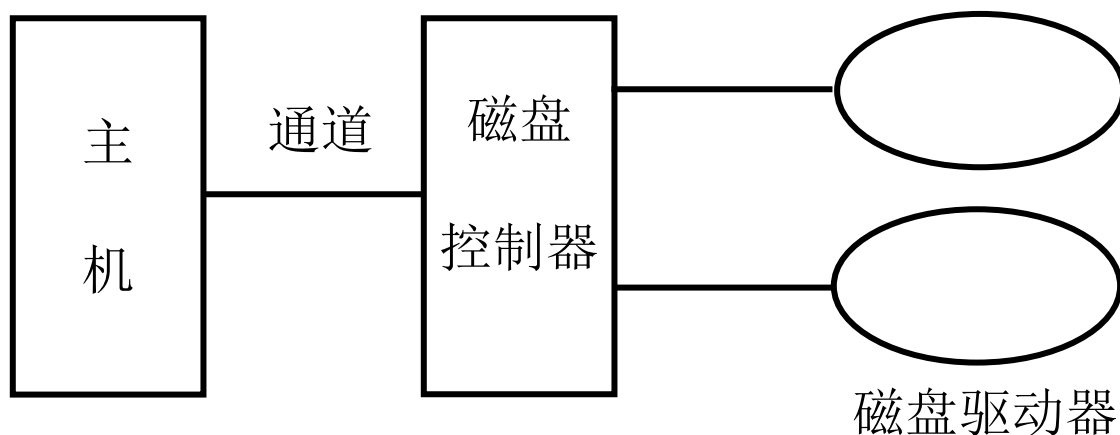
- RAID 0级：按照**数据条带**(一个或多个扇区)，轮转写入，并采用并行交叉存取，。它虽能有效地提高磁盘I/O速度，但无冗余校验功能，致使磁盘系统的可靠性不好。



RAID0中的数据映射

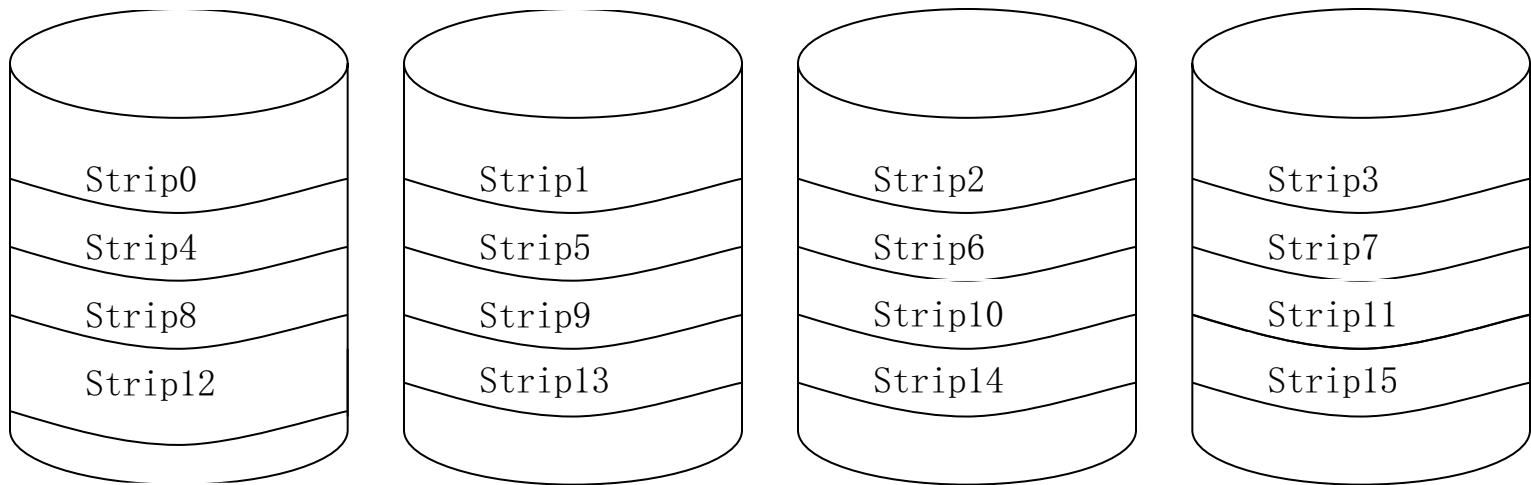
RAID 1 级

- RAID1级：具有**磁盘镜像**功能，可利用并行读写特性，将数据分块并同时写入主盘和镜像盘。其速度比传统镜像盘快，但磁盘利用率只有50%。磁盘镜像的示意图：

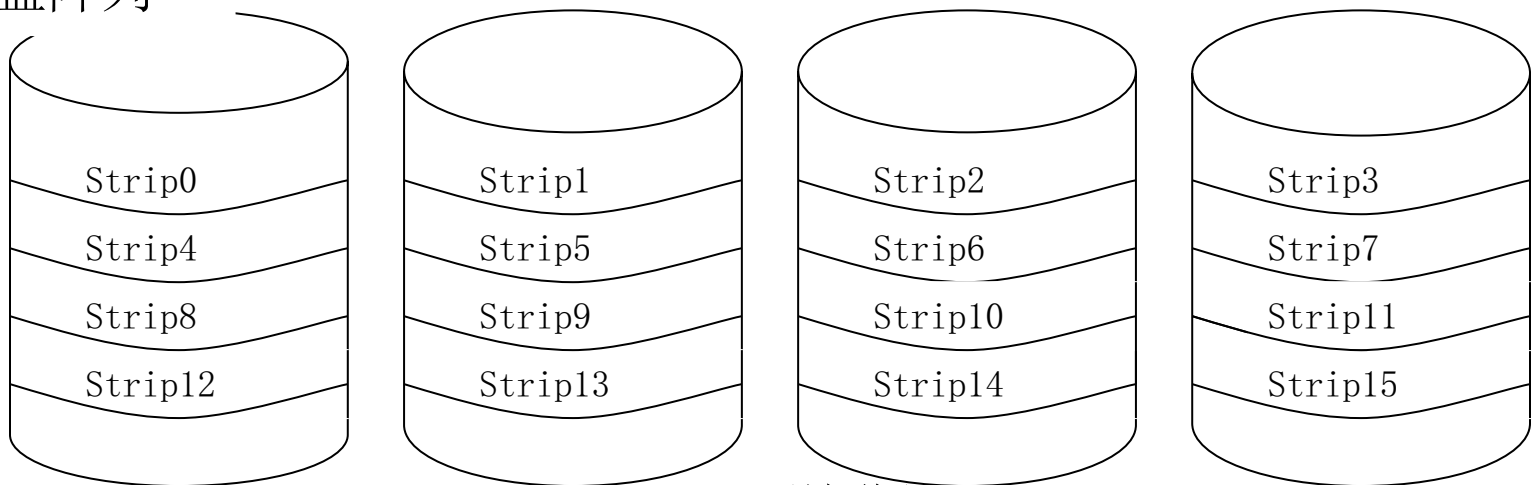




主盘阵列



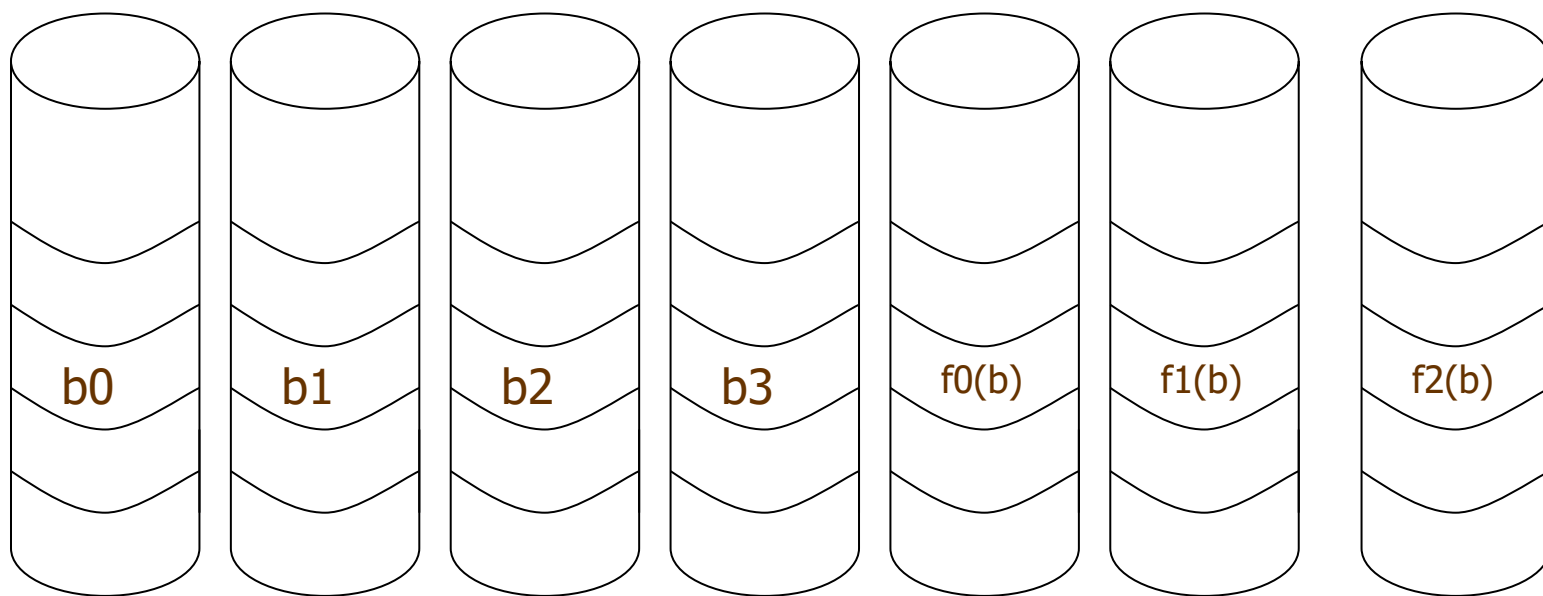
镜像盘阵列



RAID Level 1 (镜像)

RAID 2

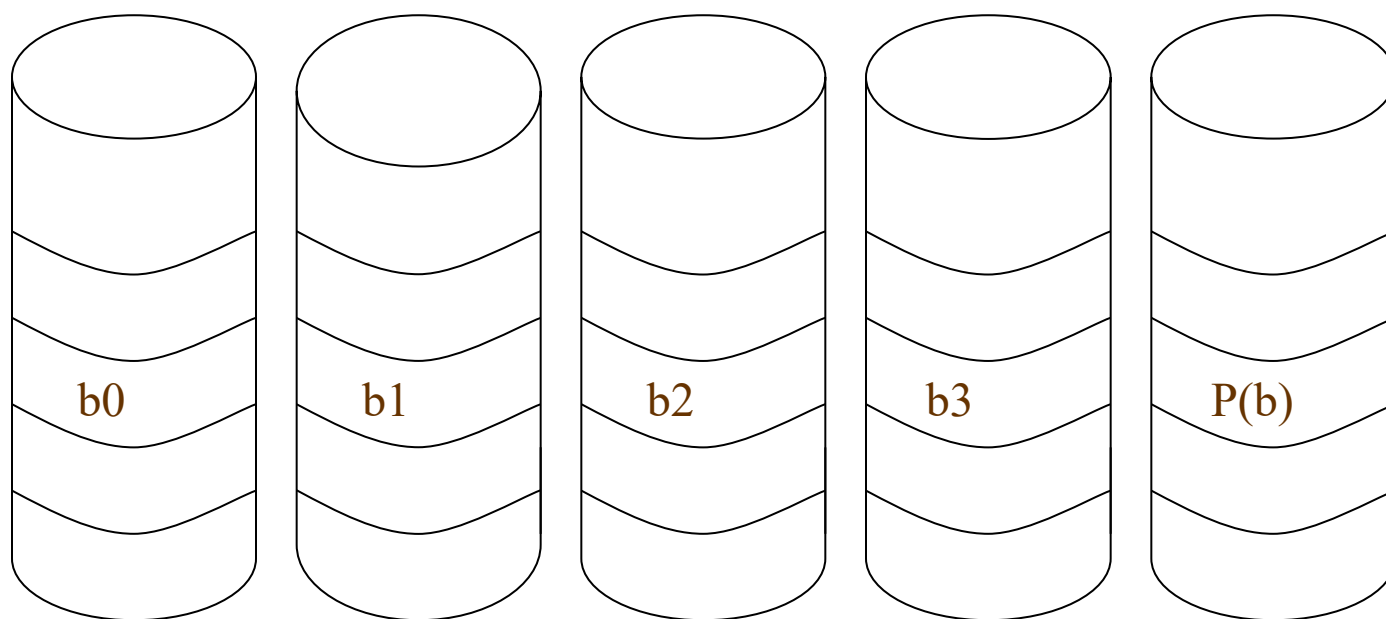
- RAID2级：按照数据字或者字节存取，具有并行传输功能的磁盘阵列。**用编码技术来提供错误检查及恢复**，实施较复杂。在商业环境中很少使用。



RAID Level 2 (冗余校验海明码)

RAID 3

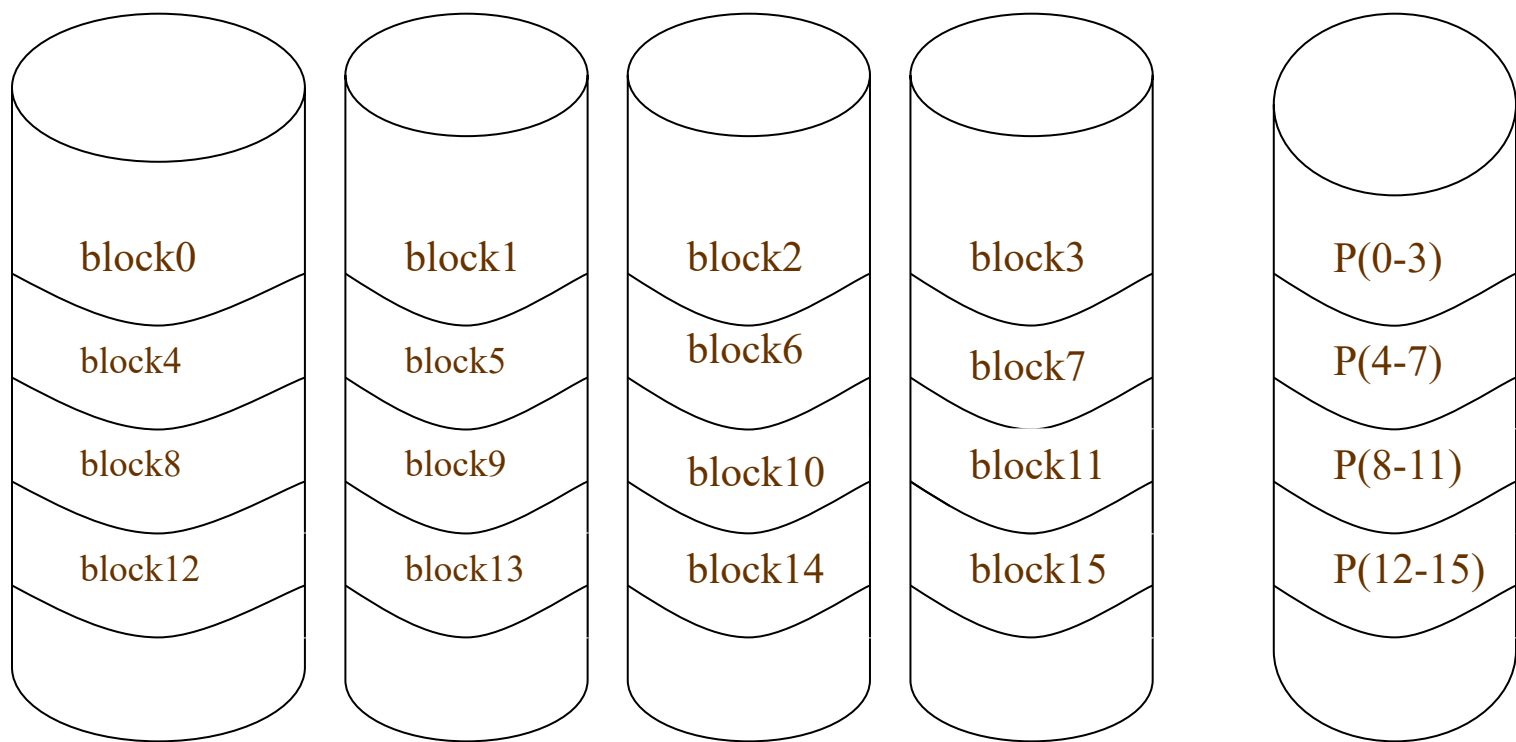
- RAID3级：利用一台**奇偶校验盘**来完成容错。是RAID2的简化，常用于科学计算和图像处理。



RAID Level 3 (奇偶校验)

RAID 4级

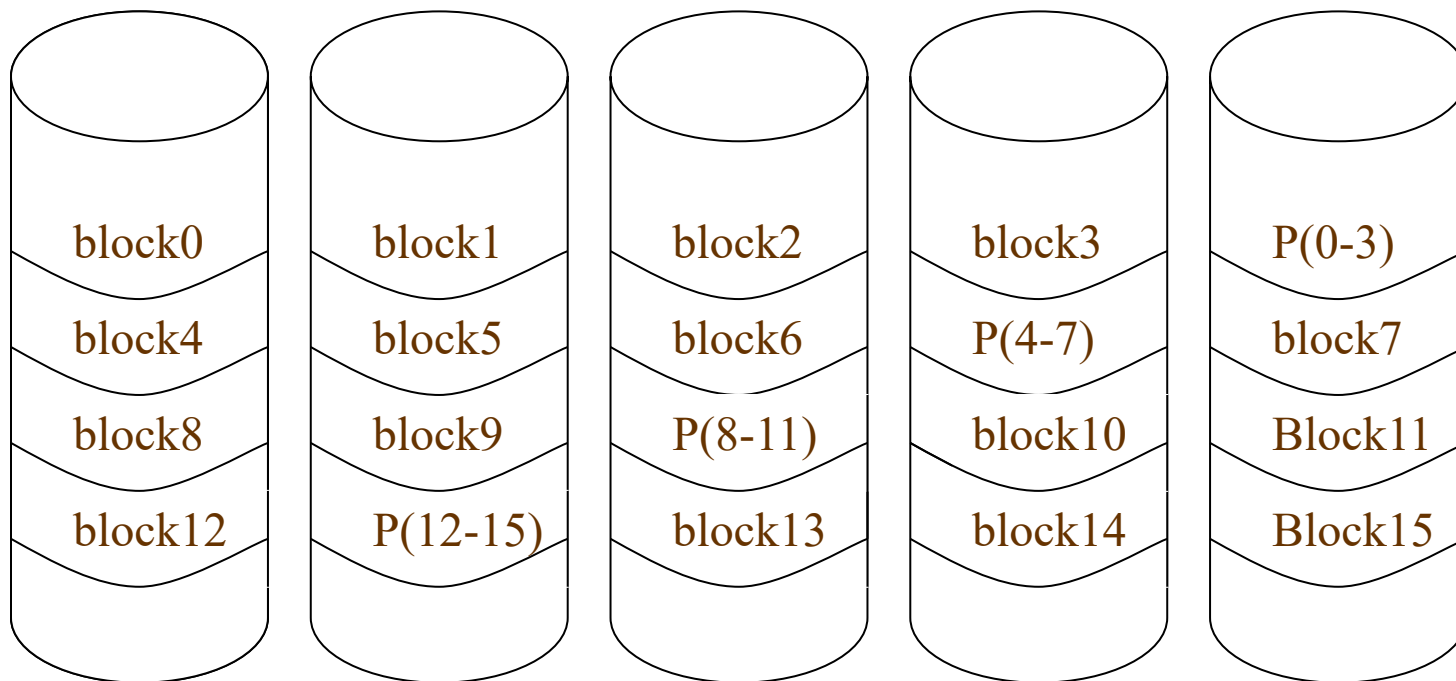
- RAID4级同RAID3级类似，但**数据是按Block拆分**，在商业环境中也很少使用。



RAID Level 4 (Block 级奇偶校验)

RAID 5级

- RAID5级：是一种具有独立数据传送功能的磁盘阵列。每个驱动器都有自己独立的数据通道，**奇偶校验码分布在不同的盘上**，独立地进行读写，纠错信息散布在所有数据盘上。常用于I/O繁忙的事务处理。





RAID 6、7级

- 在RAID6级中，设置了一个**专用的可快速访问的异步校验盘**，该盘具有独立的数据访问通道，比RAID3、RAID5级性能更好。
- RAID7：自身带有**智能化实时操作系统和用于存储管理的软件工具**，可完全独立于主机运行，不占用主机CPU资源。



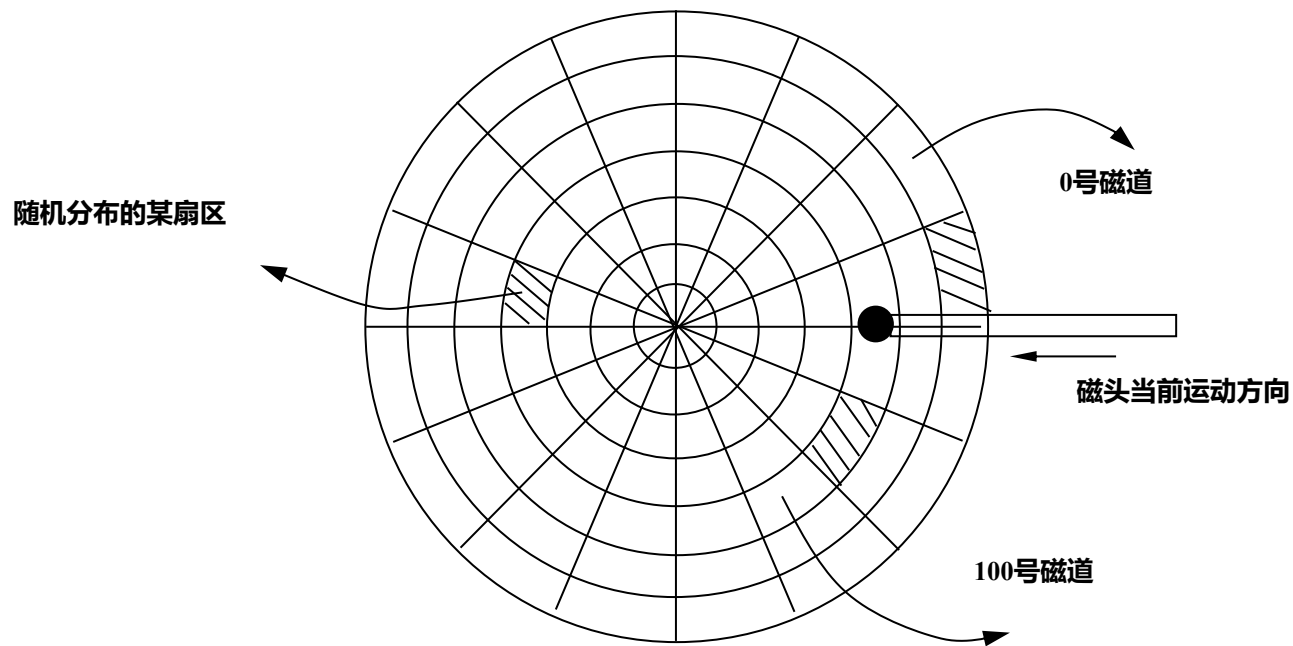
自己练习

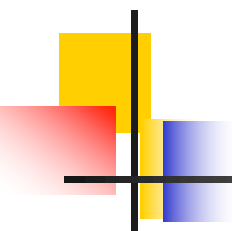
- 假定磁盘有200个磁道，编号0~199，当前存取臂的位置在143号磁道上，并刚刚完成了125号磁道的服务请求，如果请求队列的先后顺序是：86，147，91，177，94，150，102，175，130；试问：为完成上述请求，下列算法存取臂移动的总量是多少？并算出存取臂移动的顺序。
 - (1) 先来先服务算法FCFS；
 - (2) 最短查找时间优先算法SSTF；
 - (3) 扫描算法Look。

考研真题

- 假设计算机系统采用CSCAN(循环扫描)磁盘调度策略，使用2KB的内存空间记录16384个磁盘块的空闲状态。
 - (1) 请说明在上述条件下如何进行磁盘块空闲状态的管理。
 - (2) 设某单面磁盘旋转速度为每分钟6000转，每个磁道有100个扇区，相邻磁道间的平均移动时间为1ms。若在某时刻磁头位于100号磁道处，并沿着磁道号增大的方向移动(如下图所示)，磁道号请求队列为50,90,30,120，对请求队列中的每一个磁道需读取1个随机分布的扇区，则读完这4个扇区总共需要多少时间？给出计算过程。
 - (3) 如果将磁盘替换为随机访问的Flash半导体存储器(如U盘、SSD等)，是否有比CSCAN更高效的磁盘调度策略？若有，给出磁盘调度策略的名称并说明理由；若无，说明理由。

考研真题





假设计算机系统采用CSCAN(循环扫描)磁盘调度策略，使用2KB的内存空间记录16384个磁盘块的空闲状态。

(1) 请说明在上述条件下如何进行磁盘块空闲状态的管理。

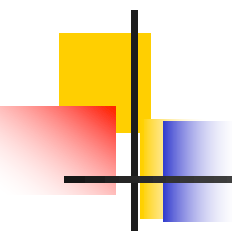
答：用位图表示磁盘的空闲块状态。

- 每一位表示一个磁盘块的空闲状态，共需 $16384/32 = 512$ 个字 $= 512 \times 4$ 个字节 $= 2\text{KB}$ ，正好可放在系统提供的内存中。



(2)

- 采用CSCAN调度算法，（ **这里考虑是到达请求的尽头就回头** ），访问磁道的顺序为120,30,50,90，则从100开始，向磁道增大方向，移动磁道长度为 $20+90+20+40=170$ ，总的移动时间为 $170 \times 1\text{ms}=170\text{ms}$ 。
- 由于转速为6000r/m，换算为0.1r/ms，则平均旋转延迟时间为半圈， $1/2/0.1=5\text{ms}$ ，总的旋转时间为 $5\text{ms} \times 4=20\text{ms}$ 。
- 由于转速为6000r/m，则读取一个磁道上的一个扇区的平均读取时间为 $10\text{ms}/100=0.1\text{ms}$ ，总的读取扇区的时间= $0.1\text{ms} \times 4=0.4\text{ms}$ 。
- 读取上述磁道上的所有4个扇区所花费的总时间= $170\text{ms}+20\text{ms}+0.4\text{ms}=190.4\text{ms}$ 。



(3) 如果将磁盘替换为随机访问的Flash半导体存储器(如U盘、SSD等), 是否有比CSCAN更高效的磁盘调度策略? 若有, 给出磁盘调度策略的名称并说明理由; 若无, 说明理由。

- 采用FCFS(先来先服务)调度策略更高效。
- 因为Flash半导体存储器的物理结构不需要考虑寻道时间和旋转延迟, 可直接按I/O请求的先后顺序服务。