操作系统　Operating System

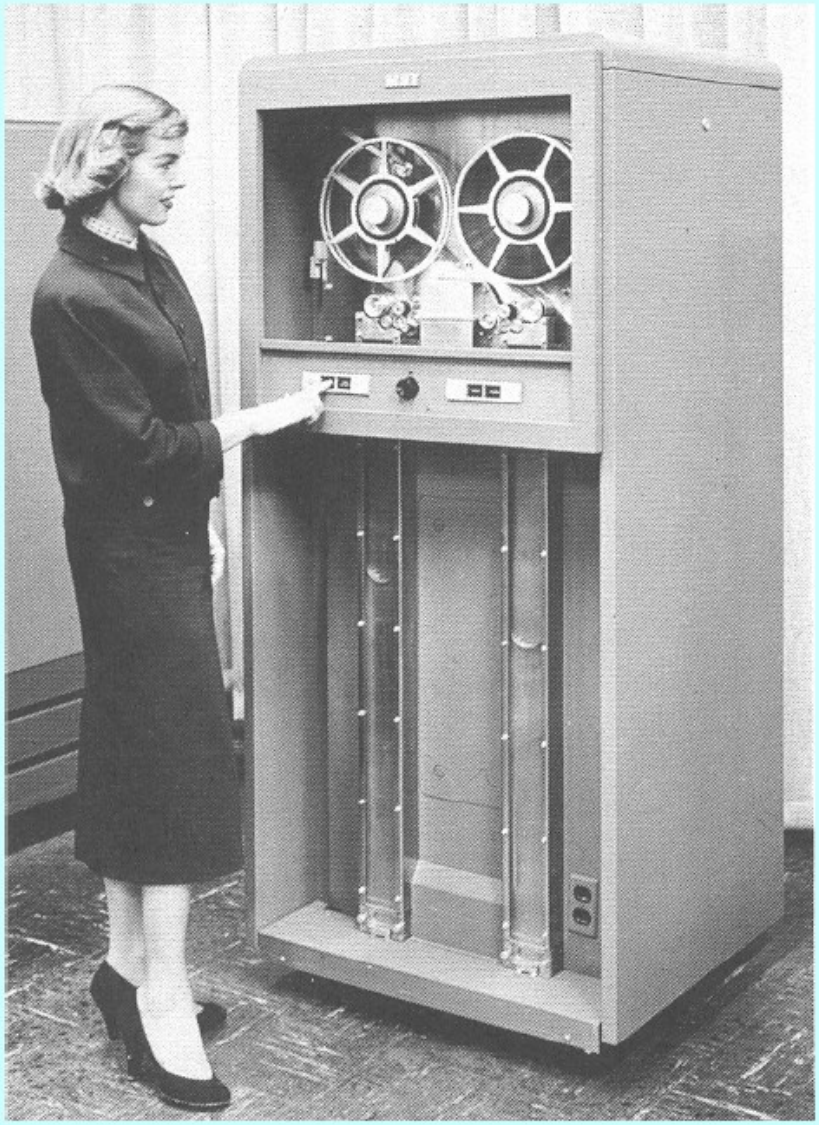
第六章 磁盘存储管理

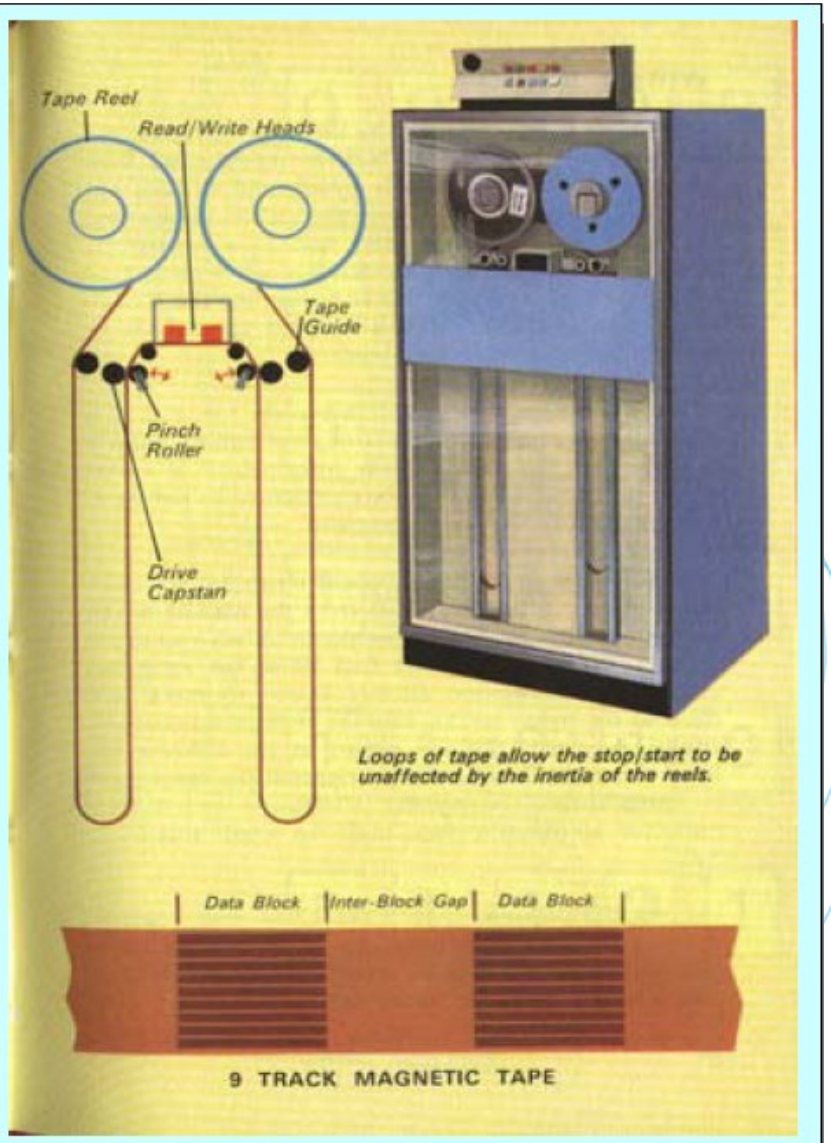
沃天宇
woty@buaa.edu.cn
2021年5月13日

内容提要

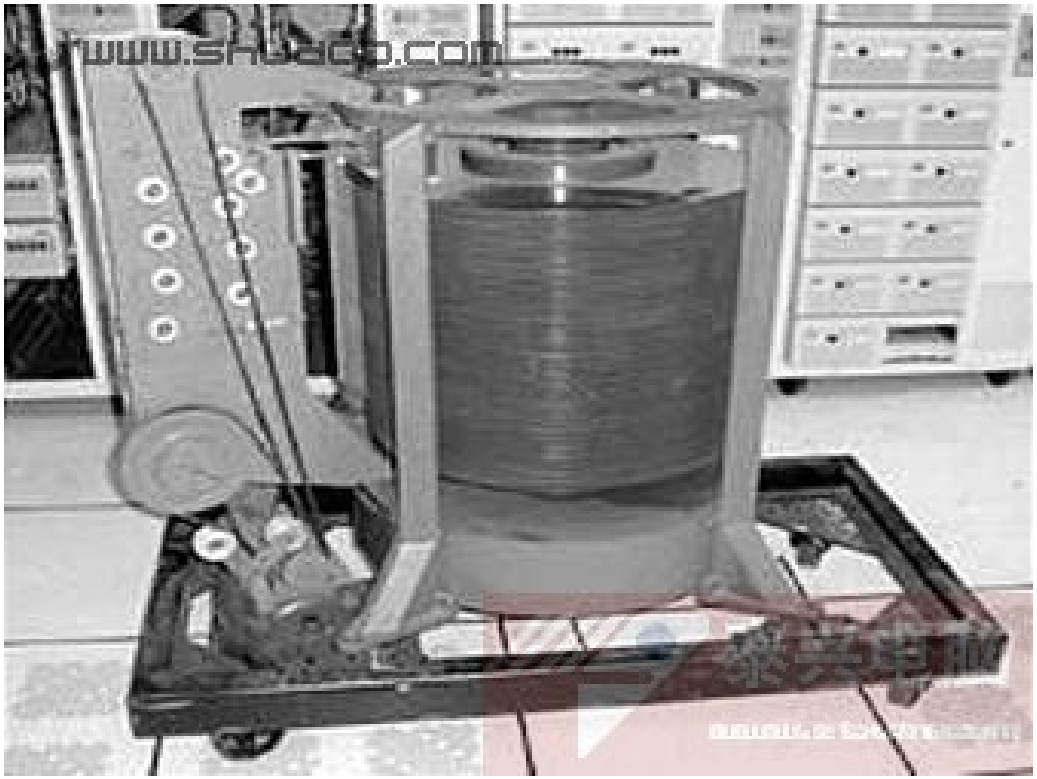
- 磁盘存储的工作原理
- 保证磁盘的可靠性
- 提高I/O访问速度
- 磁盘管理的实例

硬盘诞生前的外存——磁带





IBM System 305 RAMAC(1956年)



第一个使用移动磁头硬盘的商业计算机 （容量：5M字符≈4.7683 MB）

硬盘简史

62~78年
14吋硬盘

78~80年
8吋硬盘

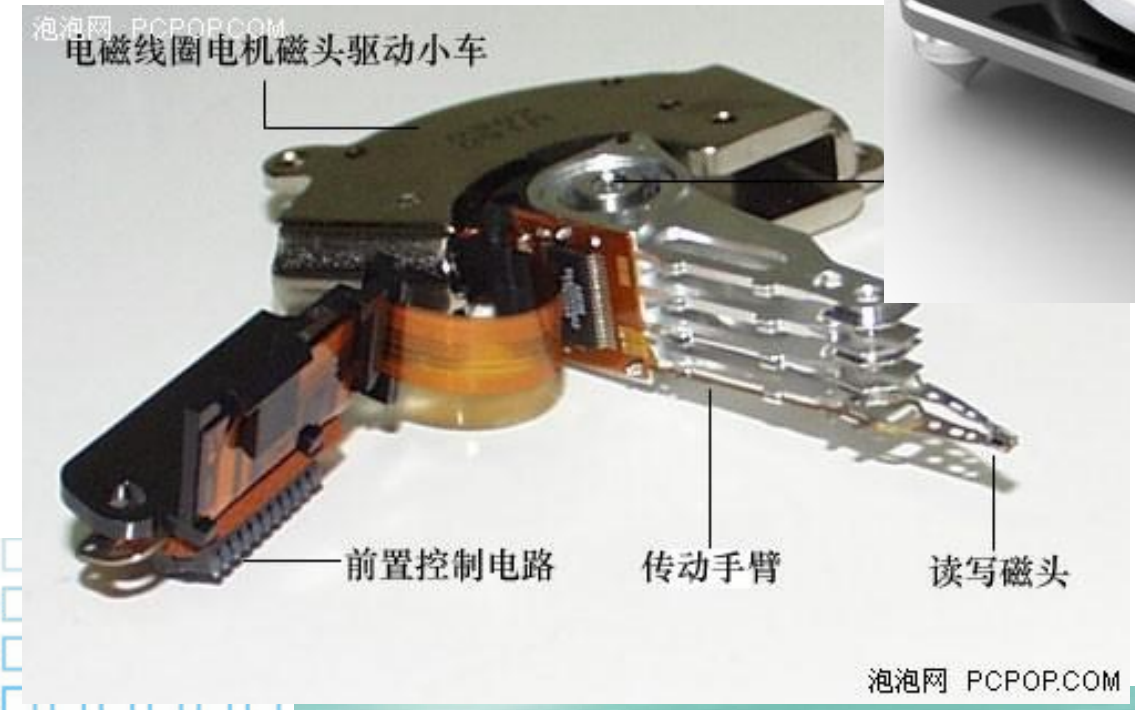
80~84年
5.25吋硬盘

84~89年
3.5吋硬盘

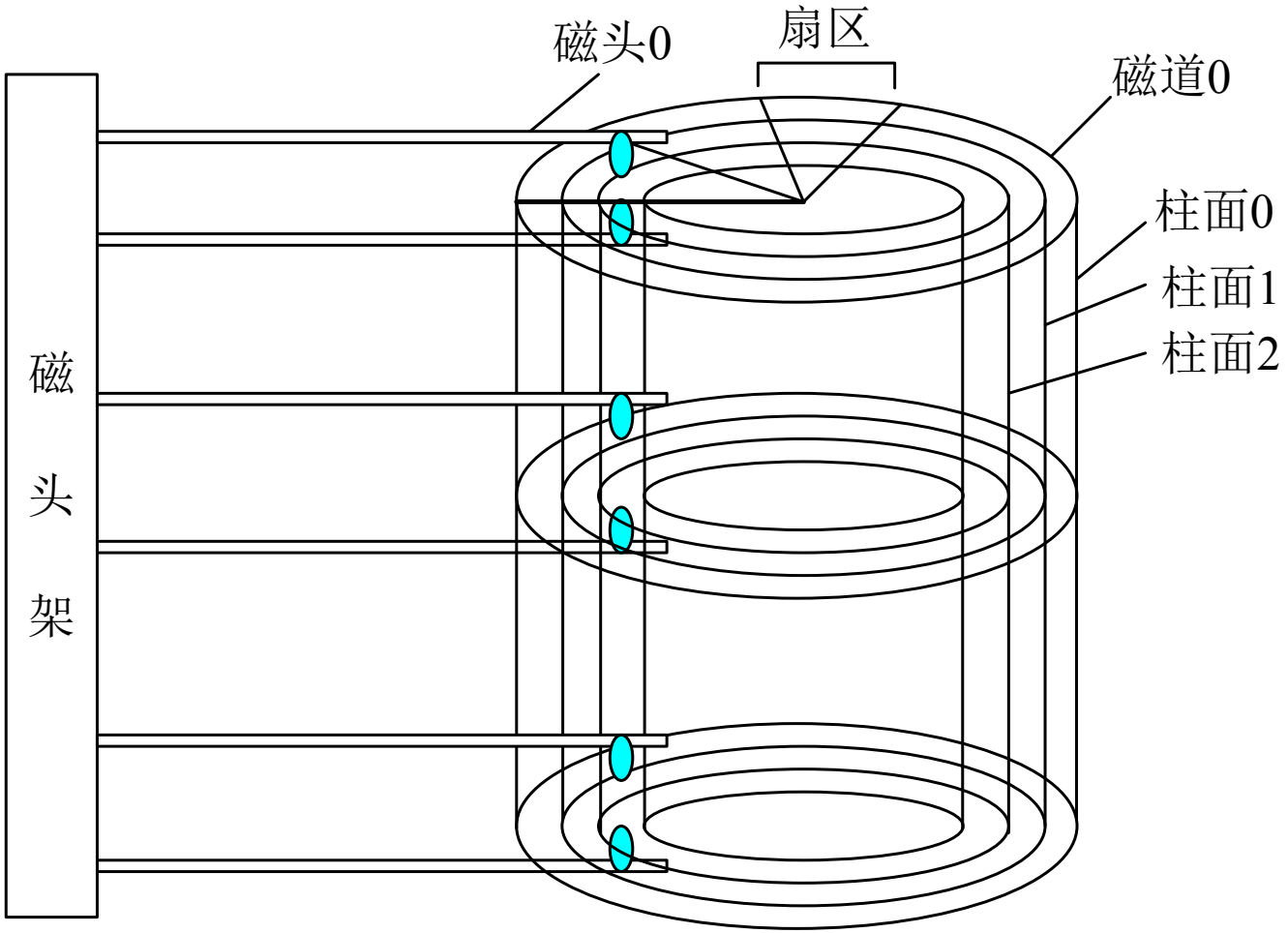
89~92年
2.5吋硬盘

92年~今
1.8吋硬盘

硬盘的机械结构



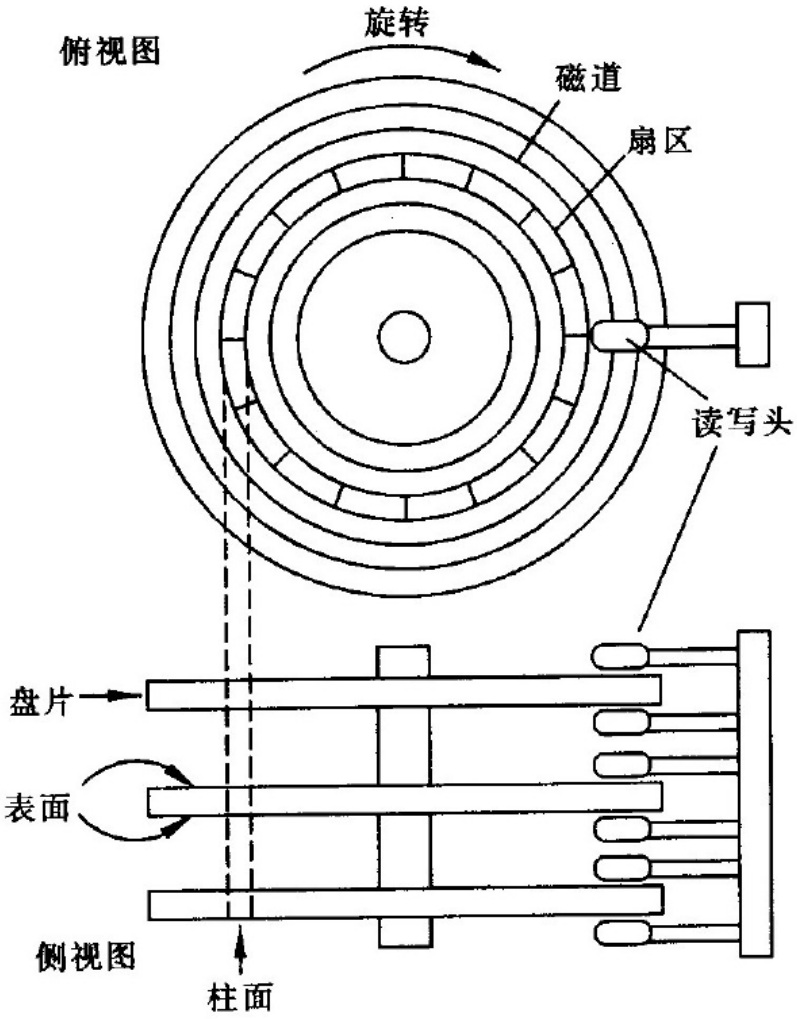
温彻斯特盘



1973年，IBM： 磁头与盘片不接触

基本概念

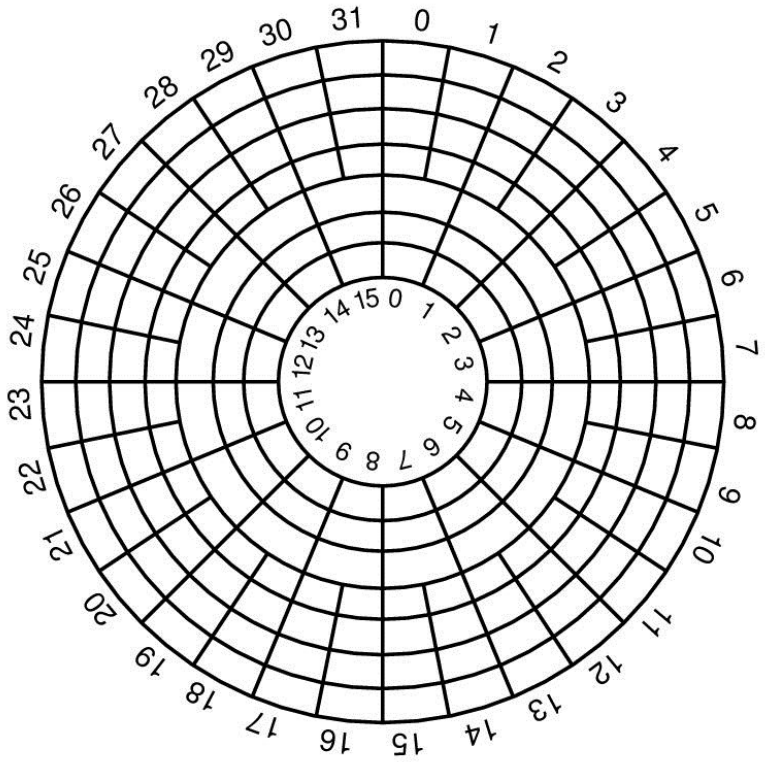
- 扇区（sector）
  - 盘片被分成许多扇形的区域
- 磁道（track）
  - 盘片上以盘片中心为圆心，不同半径的同心圆。
- 柱面（cylinder)
  - 硬盘中，不同盘片相同半径的磁道所组成的圆柱。
- 每个磁盘有两个面，每个面都有一个磁头(head)。



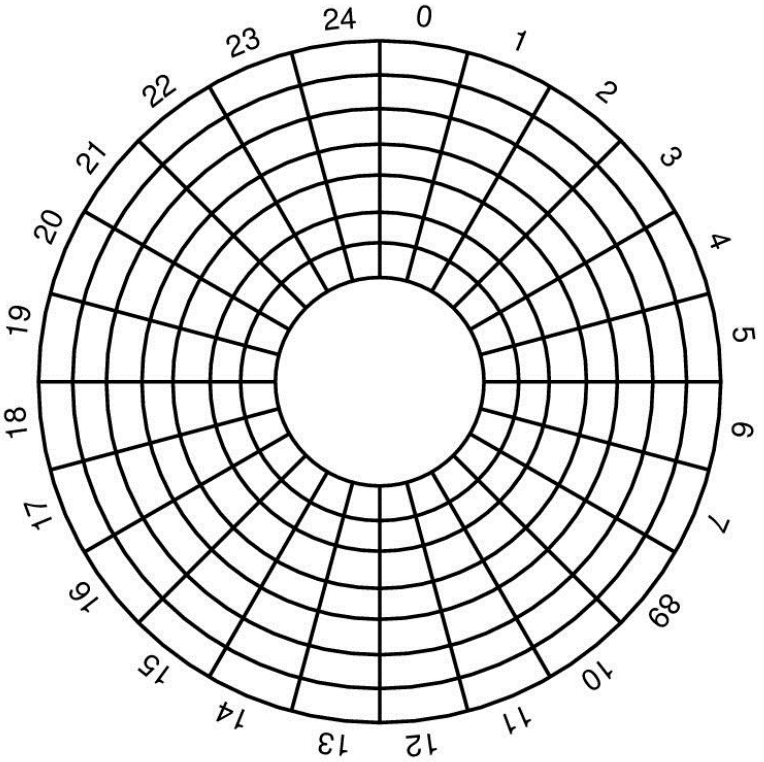
磁盘设备结构示意图

磁盘的组织

- 读一个扇区需要柱面/磁头/扇区
- 现代磁盘驱动器可以看做一个一维的逻辑块的数组，逻辑块是最小的传输单位
- 一维逻辑块数组按顺序映射到磁盘的扇区。
  - 扇区0是最外面柱面的第一个磁道第一个扇区。
  - 该映射是先按磁道内扇区顺序，再按柱面内磁道顺序，再按从外到内的柱面顺序来排序的。
  - 绝大多数磁盘都有一些缺陷扇区，因此映射必须用磁盘上的其他空闲扇区来替代这些缺陷扇区。
  - 对于磁盘，每个磁道的扇区数并不是常量。
    - 磁道的位密度



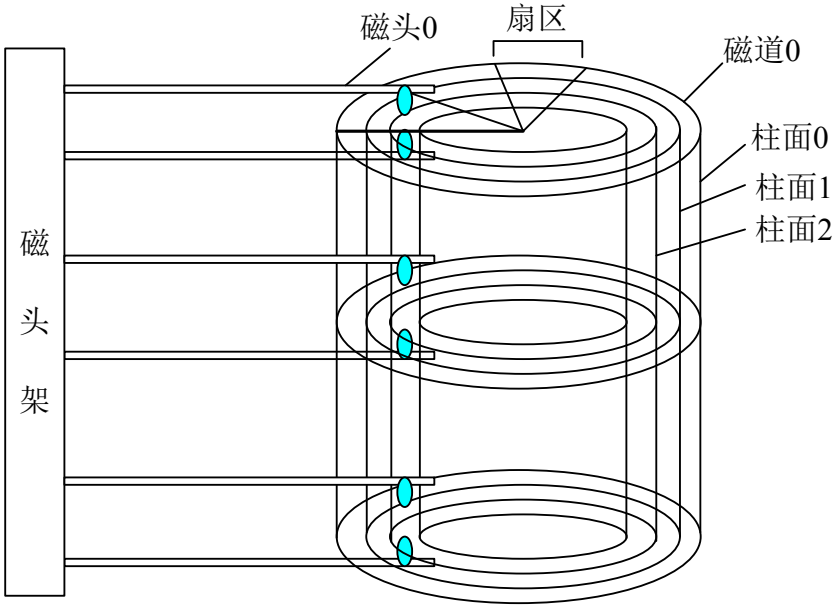
具有两个环带的磁盘物理规格

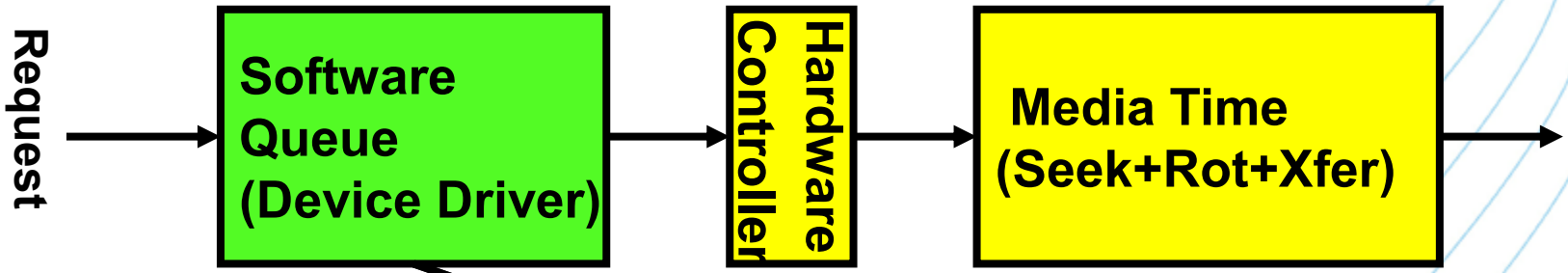


经过虚拟的磁盘几何规格

磁盘访问时间

- 寻道时间
- 旋转延迟时间
- 传输时间



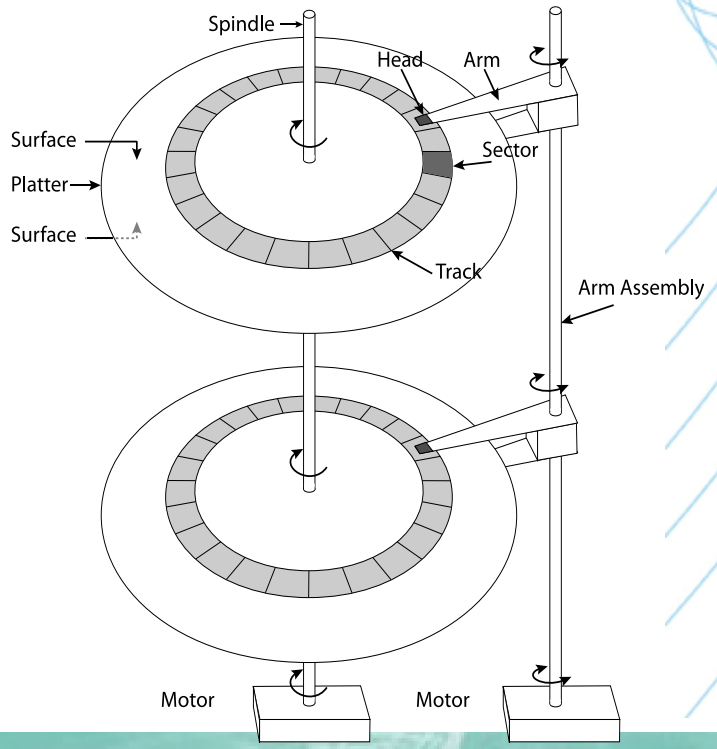


Response Time（磁盘延迟） = Queue + Disk Service Time

寻道+旋转+传输

寻道时间

- 这是把磁臂（磁头）从当前位置移动到指定磁道上所经历的时间。该时间是启动磁盘的时间s与磁头移动n条磁道所花费的时间之和。
- $T_s = m \times n + s$



旋转延迟时间

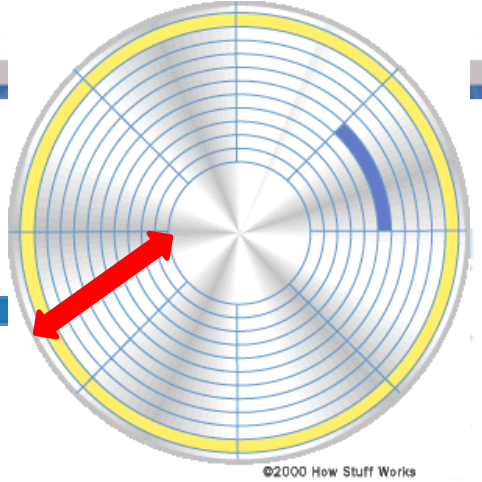
- 对于硬盘，假设旋转速度为3600 RPM，每转需时16.7 ms，平均旋转延迟时间T_r为8.3 ms。对于软盘，其旋转速度为300或600 r/min，这样，平均T_r为50~100 ms。

$$Tr = 1/(2r)$$

传输时间

- T_t是指把数据从磁盘读出，或向磁盘写入数据所经历的时间，T_t的大小与每次所读／写的字节数b，旋转速度r以及磁道上的字节数N有关
- $T_t = b/(rN)$
  - 传输数据的大小 (通常是1个扇区): 512B
  - 旋转速度：3600 RPM ~ 15000 RPM
  - 典型的传输速度：2MB~50 MB/秒

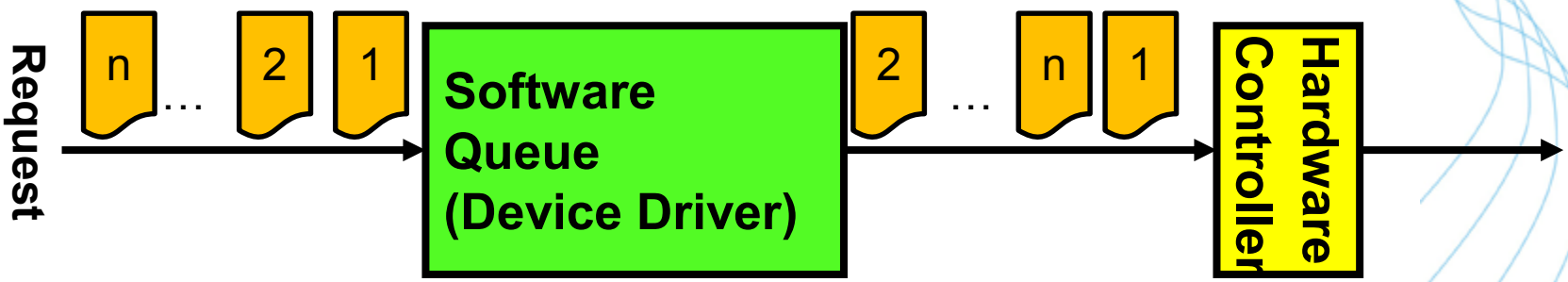
总访问时间T_a



- $T_a = T_s + 1/(2r) + b/(rN)$
- 访问时间=寻道时间+旋转延迟时间+传输时间
- 例如
  - 假定寻道时间和旋转延迟时间平均为30 ms，而磁道的传输速率为1 MB/s，如果传输1K字节，此时总的访问时间为31 ms，传输时间所占比例是相当地小。当传输10K字节的数据时，其访问时间也只是40 ms，即当传输的数据量增加10倍时，访问时间只增加了约30%。 ——抓主要矛盾

磁盘调度算法

- 对于OS的磁盘IO驱动，可能收到IO请求的速度大于磁盘实际能够执行的速度
  - 接收请求的顺序 vs 执行IO的顺序



- 先来先服务
- 最短寻道时间优先

先来先服务算法（FCFS）

- **算法思想：**

  按访问请求到达的先后次序服务。

- **优点：**

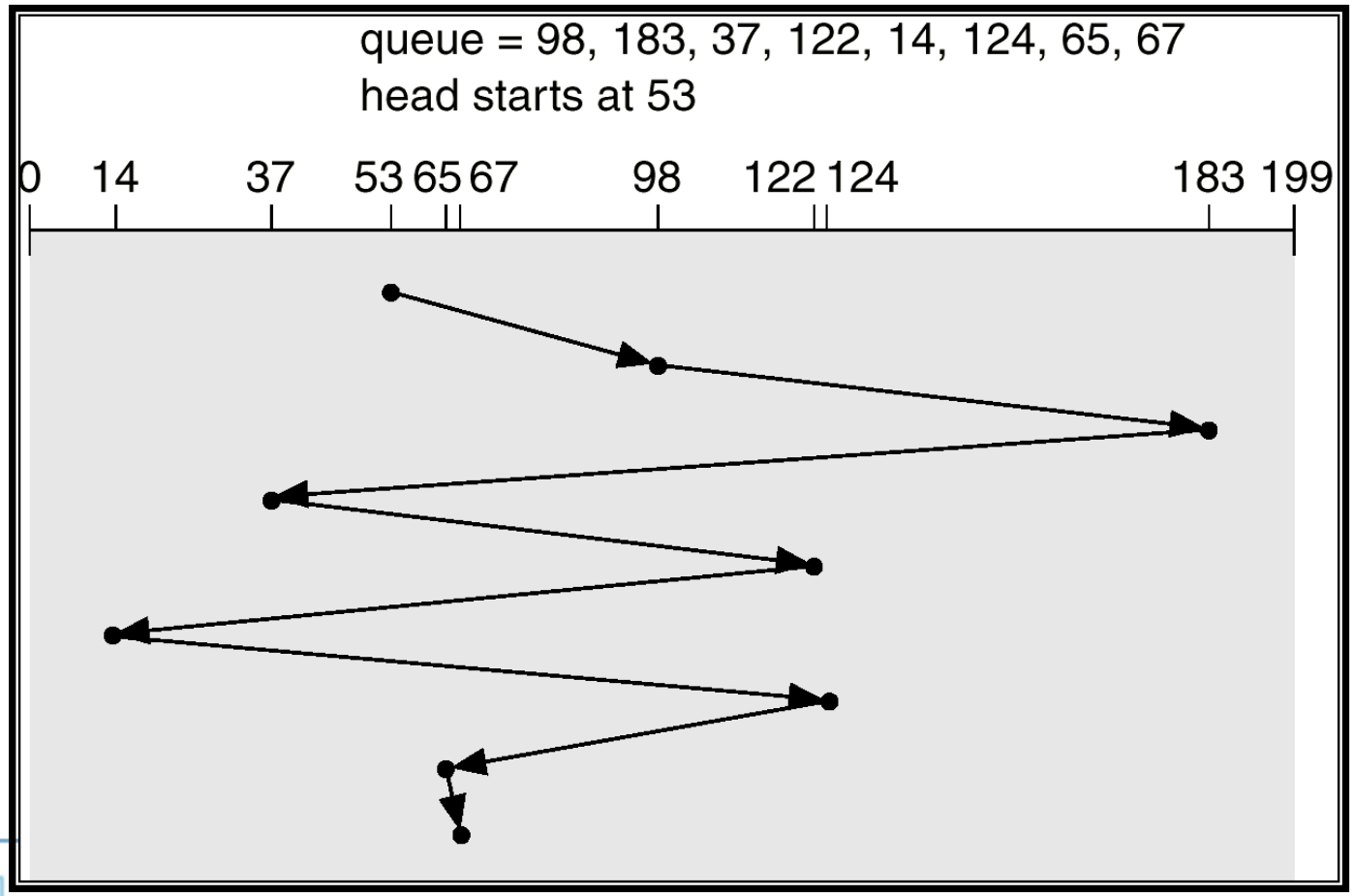
  简单，公平。

- **缺点：**

  效率不高，相邻两次请求可能会造成最内到最外的柱面寻道，使磁头反复移动，增加了服务时间，对机械也不利。

先来先服务

- 总行程（磁头移动距离）：640个柱面



最短寻道时间优先算法（SSTF）

- **算法思想：**

  优先选择距当前磁头最近的访问请求进行服务，主要考虑寻道优先。

- **优点：**

  改善了磁盘平均服务时间。

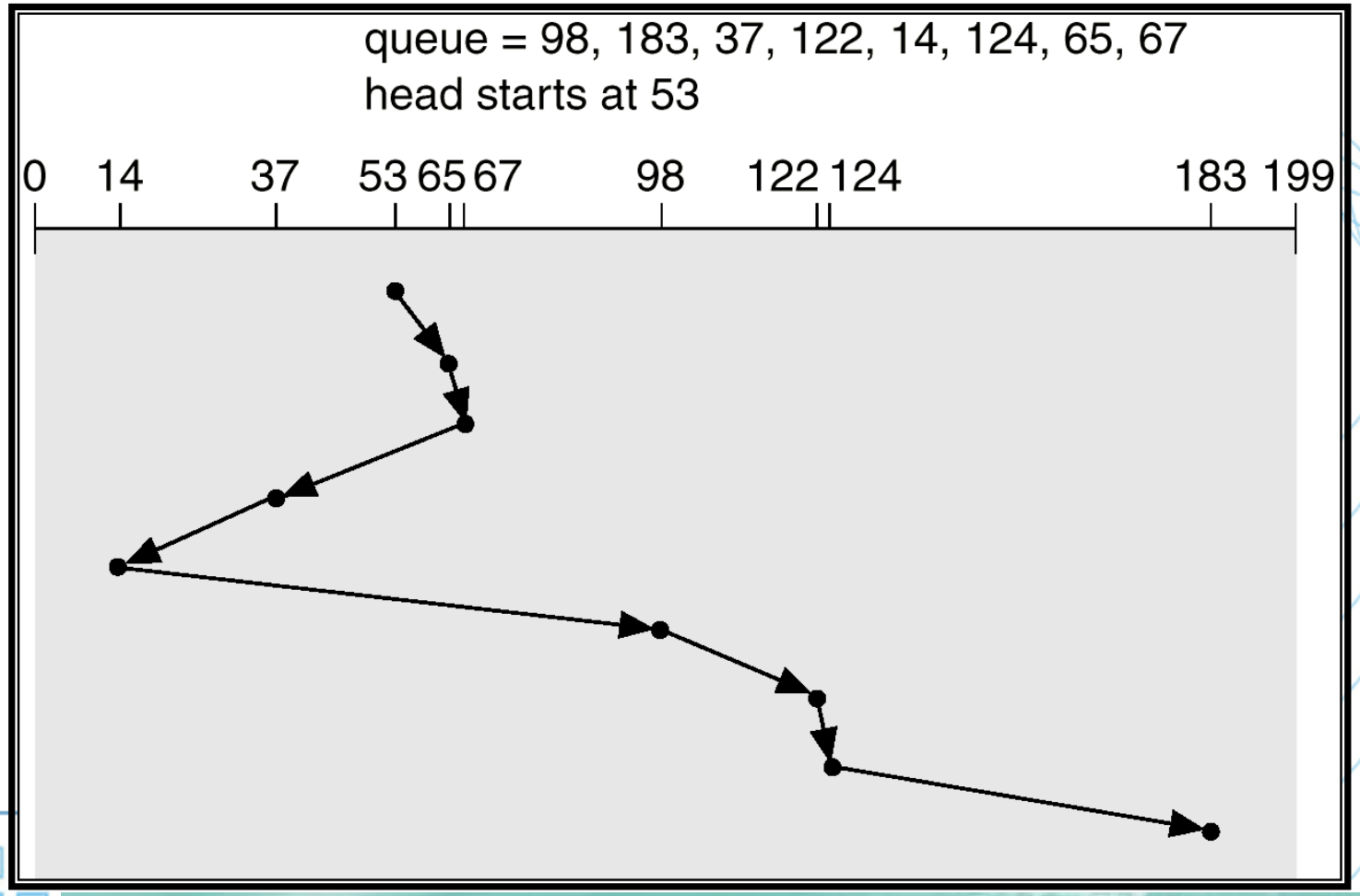
- **缺点：**

  可能产生“饥饿” 现象，造成某些访问请求长期等待得不到服务。

- **调度的两个评价标准：效率＋公平**

最短寻道时间优先 (SSTF)

- 总行程（磁头移动距离）：236个柱面



改进算法

- 不仅考虑磁道的距离，还要考虑磁头移动方向
  - 扫描算法（电梯算法）
  - 循环扫描算法

扫描算法（SCAN）电梯调度

- **算法思想：**

  当有访问请求时，磁头按一个方向移动，在移动过程中对遇到的访问请求进行服务，然后判断该方向上是否还有访问请求，如果有则继续扫描；否则改变移动方向，并为经过的访问请求服务，如此反复。

- **优点：**

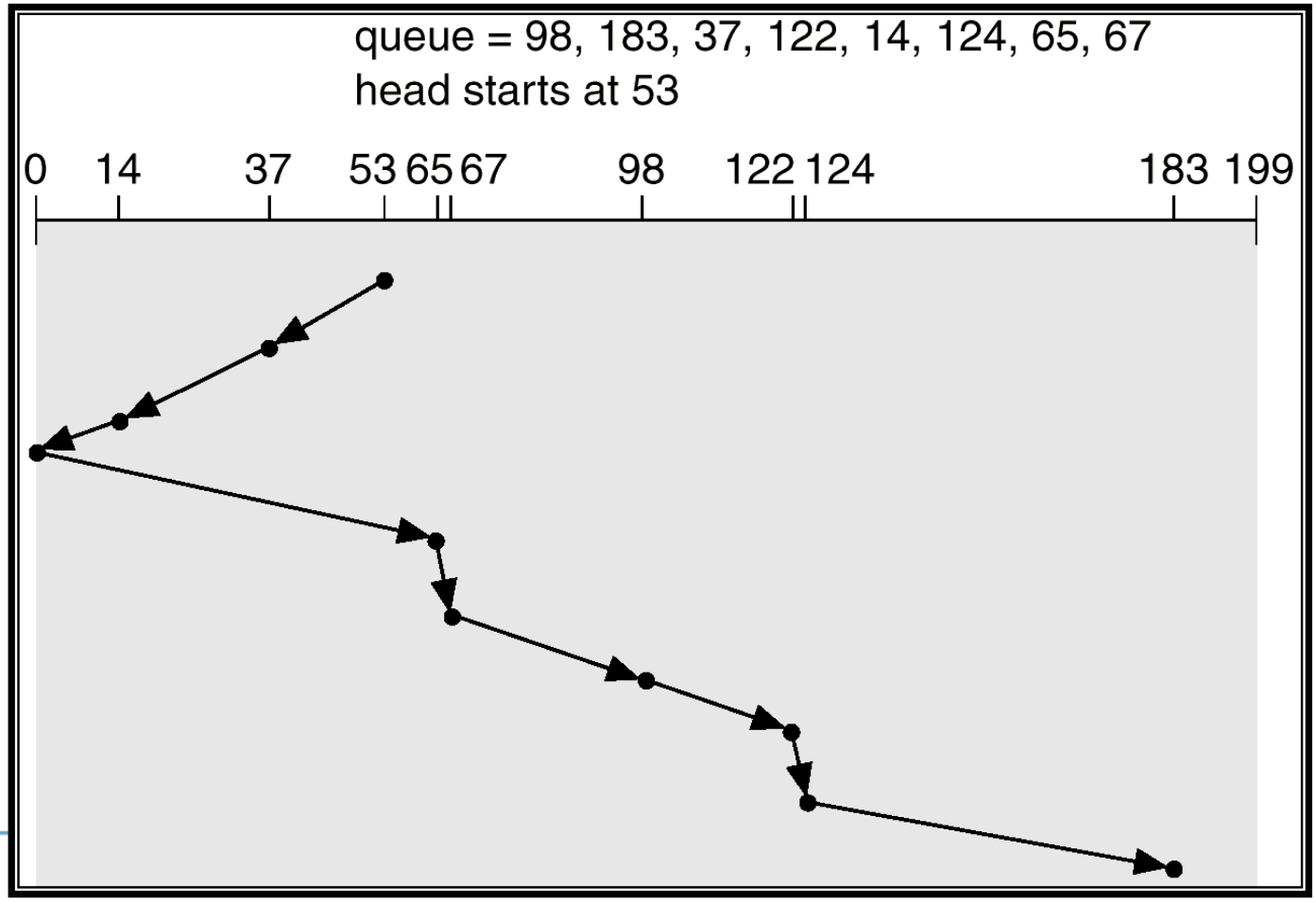
  克服了最短寻道优先的缺点，既考虑了距离，同时又考虑了方向

- **缺点：**

  但由于是摆动式的扫描方法，两侧磁道被访问的频率仍低于中间磁道。

扫描算法（Scan，电梯算法）

- 总行程：208 cylinders.

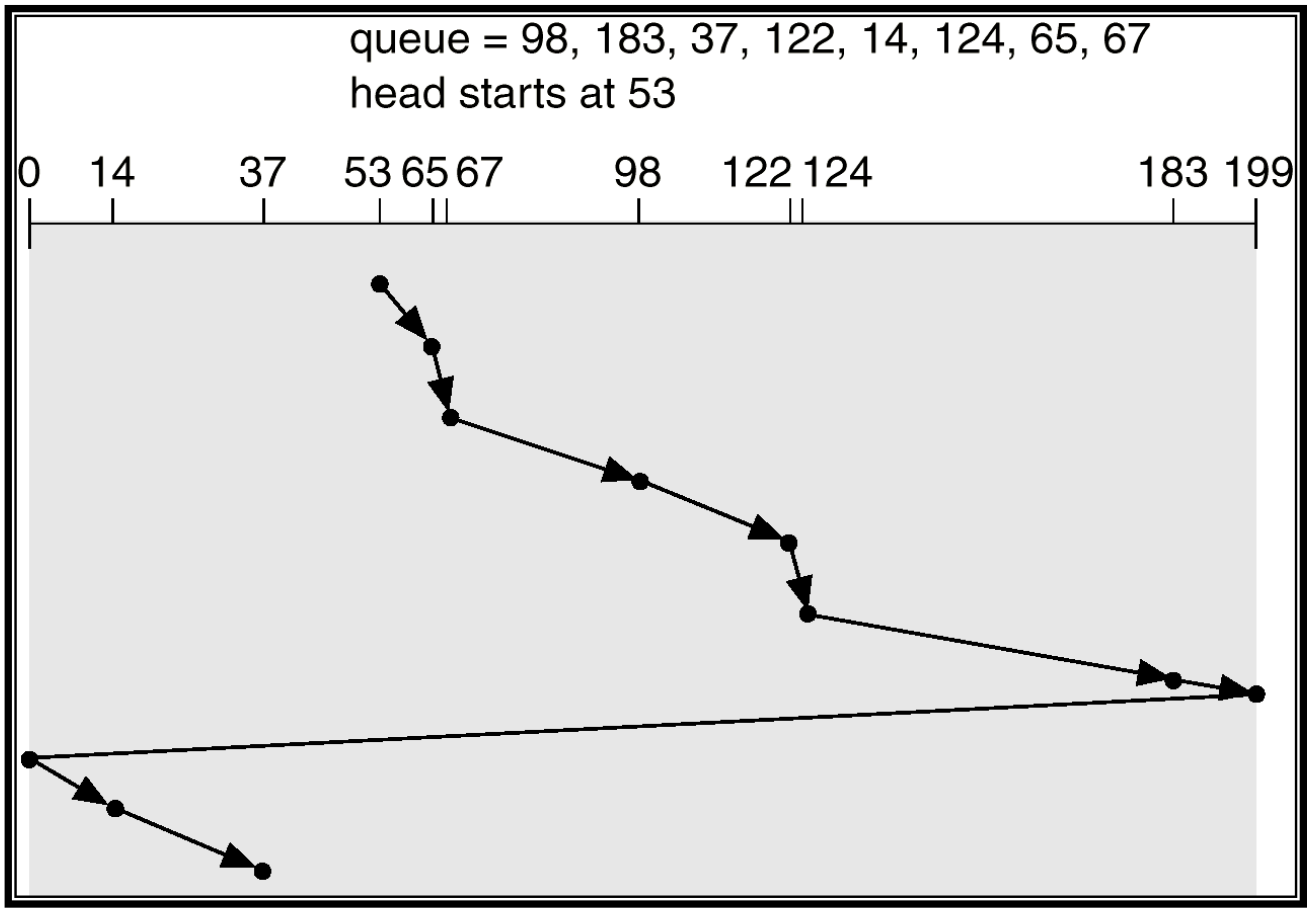


循环扫描算法（CSCAN）

- 算法思想：
  - 按照所要访问的柱面位置的次序去选择访问者。
  - 移动臂到达最后一个柱面后，立即带动读写磁头快速返回到0号柱面。
  - 返回时不为任何的等待访问者服务。
  - 返回后可再次进行扫描。
- 由于SCAN算法偏向于处理那些接近最里或最外的磁道的访问请求，所以使用改进型的C-SCAN算法可避免这个问题。

循环扫描算法（C-Scan）

- 磁头移动距离：382 个柱面

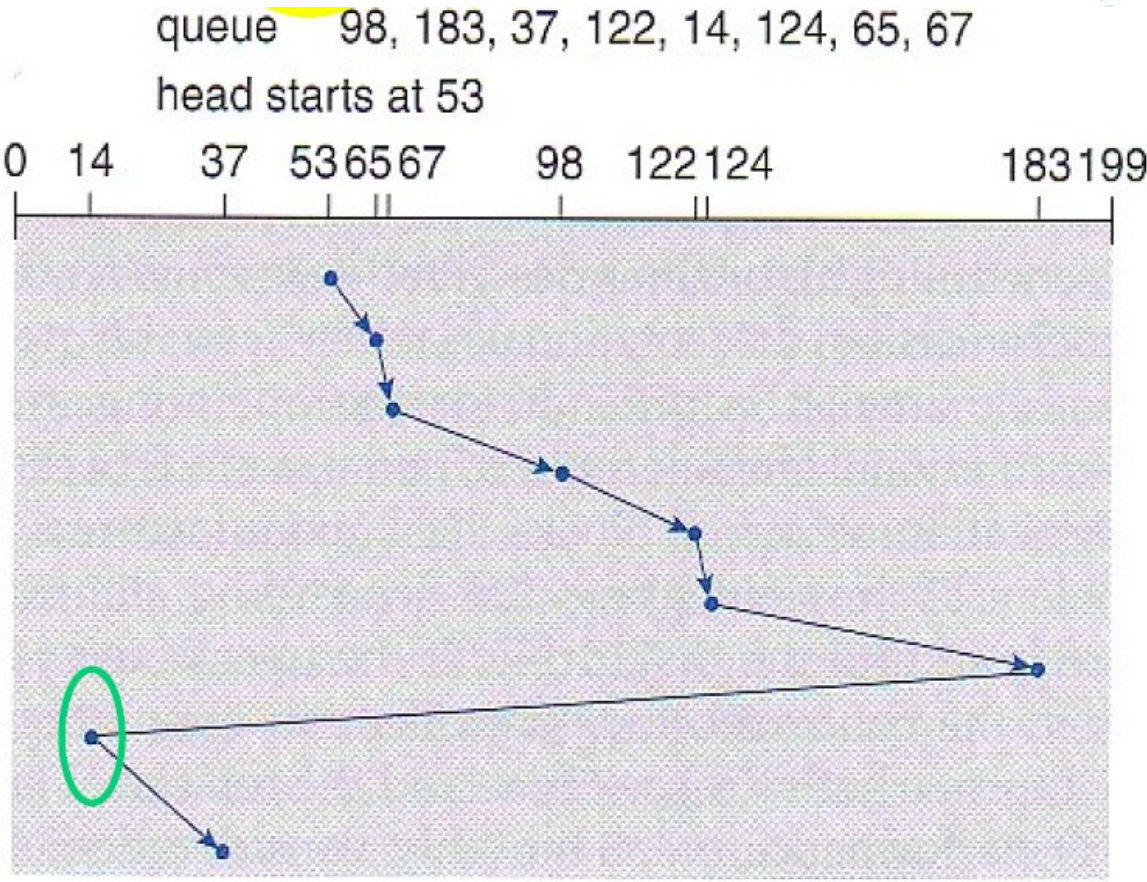


算法比较

	优点	缺点
FCFS算法	公平、简单	平均寻道距离大，仅应用在磁盘I/O较少的场合
SSTF算法	性能比“先来先服务”好	不能保证平均寻道时间最短，可能出现“饥饿”现象
SCAN算法	寻道性能较好，可避免“饥饿”现象	不利于远离磁头一端的访问请求
C-SCAN算法	消除了对两端磁道请求的不公平	--

Scan和C-Scan的问题

- 磁头总是扫描到最里/最外磁道才返回
- 改进：
  - Scan=>Look (查看扫描)
  - C-Scan=>C-Look (循环查看扫描)



C-Look算法

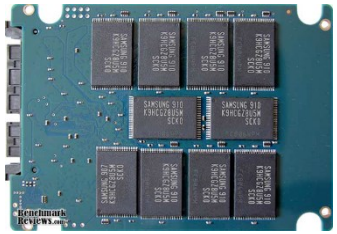
322寻道

其他改进算法

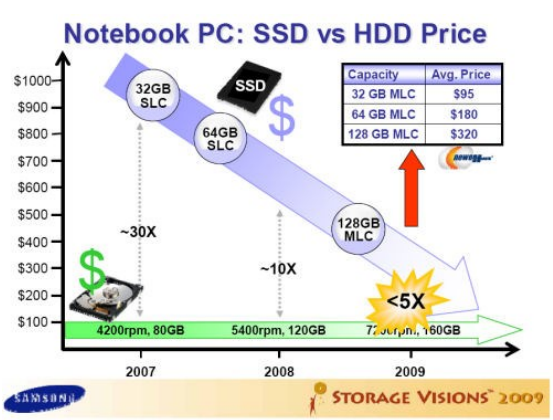
- “磁臂粘着”现象：有一个或几个进程对某一磁道有较高的访问频率，即这个(些)进程反复请求对某一磁道的I/O操作，从而垄断了整个磁盘设备。
- N-Step-SCAN
  - 将请求队列分成分成长度为N的子队列，队列之间采用FCFS，队列内部采用SCAN
  - N很大～SCAN；N＝1～FCFS
- FSCAN
  - 2-Step-SCAN，请求队列分为两个子队列（当前请求和新请求）

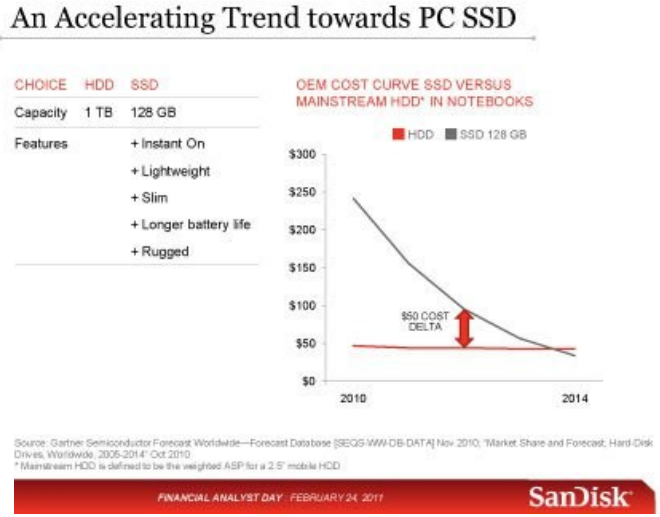
小专题：Flash Disk

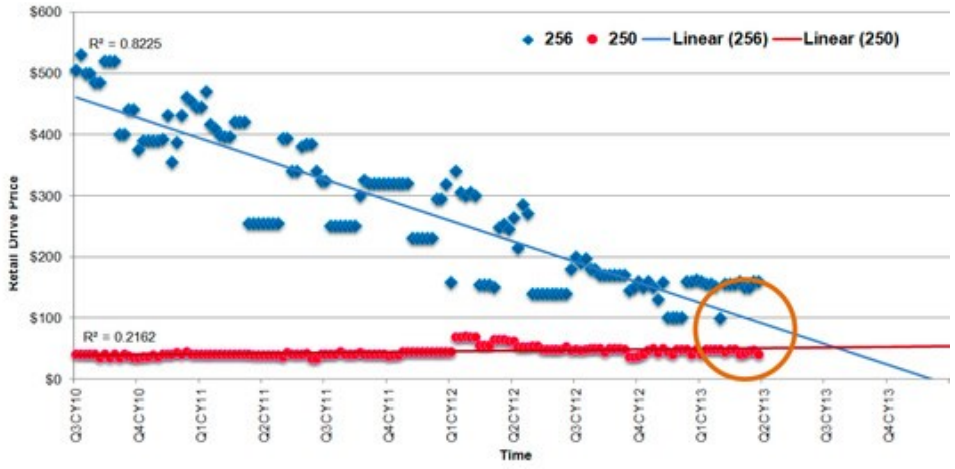
- NOR技术是由Intel公司1988年首先推出
  - NOR则是为了满足程序代码的高速访问，并且支持程序在芯片内部运行
  - 工作电压低、随机读取快、功耗低、稳定性高
- NAND技术是由东芝公司1989年发明
  - NAND技术在设计之初是为了数据存储应用
  - NAND则写回速度快、芯片面积小，特别是容量大有很大优势。



Are we in an inflection point?







SSD vs HDD

Usually 10 000 or 15 000 rpm SAS drives

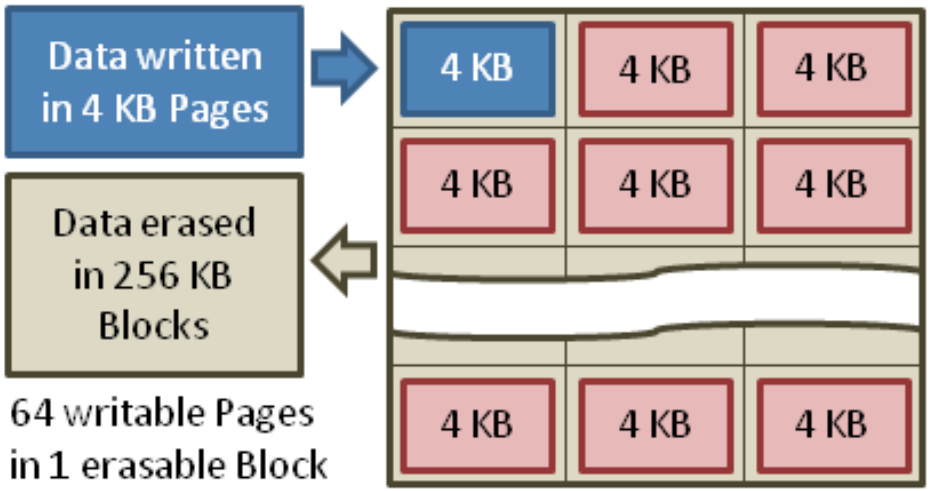
0.1 ms	**Access times** SSDs exhibit virtually no access time	5.5 ~ 8.0 ms
SSDs deliver at least 6000 io/s	**Random I/O Performance** SSDs are at least 15 times faster than HDDs	HDDs reach up to 400 io/s
SSDs have a failure rate of less than 0.5 %	**Reliability** This makes SSDs 4 - 10 times more reliable	HDD''s failure rate fluctuates between 2 ~ 5 %
SSDs consume between 2 & 5 watts	**Energy savings** This means that on a large server like ours, approximately 100 watts are saved	HDDs consume between 6 & 15 watts
SSDs have an average I/O wait of 1 %	**CPU Power** You will have an extra 6% of CPU power for other operations	HDDs' average I/O wait is about 7 %
the average service time for an I/O request while running a backup remains below 20 ms	**Input/Output request times** SSDs allow for much faster data access	the I/O request time with HDDs during backup rises up to 400~500 ms
SSD backups take about 6 hours	**Backup Rates** SSDs allows for 3 - 5 times faster backups for your data	HDD backups take up to 20~24 hours

- NAND的地址分为三部分:块号、块内页号、页内字节号。
  - 一次数据访问，NAND一般是经过三次寻址，先后确定块号、块内页号和页内字节号，至少占用三个时钟周期，因此随机字节读写速度慢。
- NOR的地址线足够多，且存储单元是并列排布，可以实现一次性的直接寻址。另外，由于NOR的数据线宽度很大，即使容量增大，它的数据寻址时间基本上是一个常量，而NAND则比较困难。

- 在数据传输速度上，NOR无论是在随机读取还是连续传输速度上都比NAND快，但相对而言，在连续大数据传输速度上，二者差异较小。
- 从产品成本来看，NOR Flash的平均每MB成本是NAND Flash成本的三到四倍。

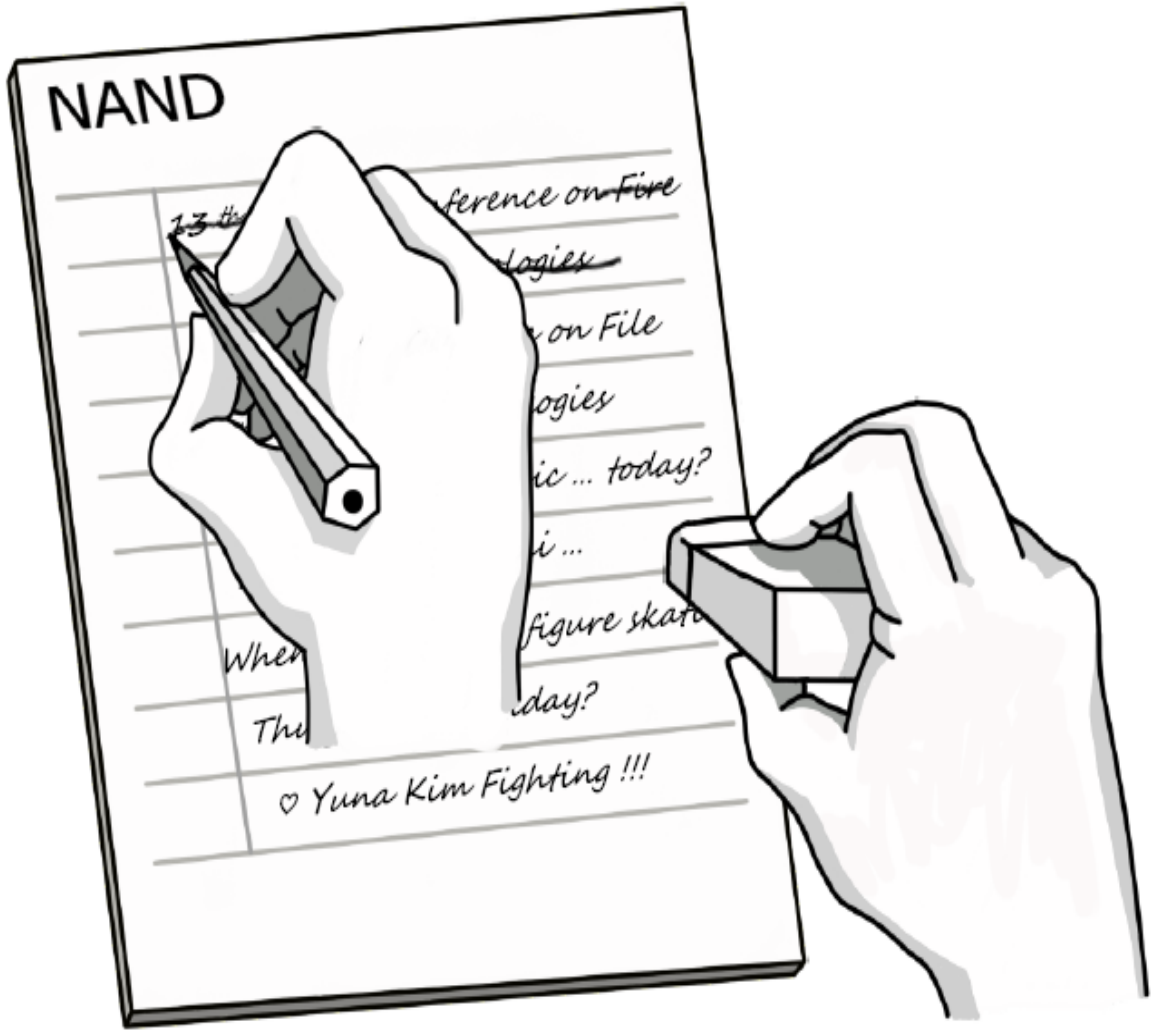
- Flash Disk的特征是低功耗、大容量、数据访问速度高。
  - 没有高速的机械部件，可以在剧烈振动的环境中工作;零下45摄氏度到零上85摄氏度
- Flash Disk执行一次数据读写的周期比硬盘短。
  - 即使是较慢的NAND型Flash Memory作为存储载体的Flash Disk执行一次数据访问的周期大致为15-30μs，对比硬盘的5ms。
- 问题：容量、可靠性。

writes 10x reads,
erasure 10x writes



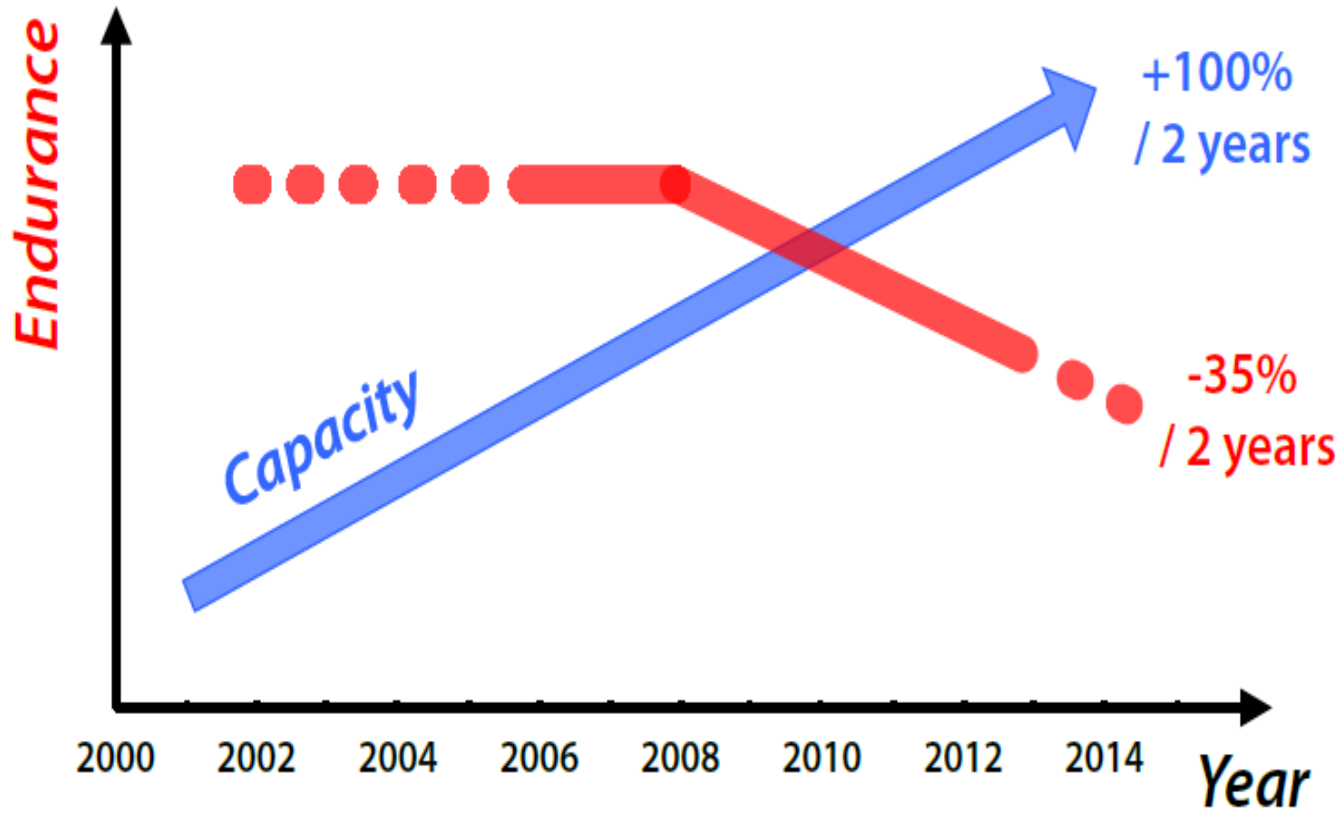
Typical NAND Flash Pages and Blocks

Writing Letters and Erasing Paper



Jaeyong Jeong *et.al.* Lifetime Improvement of NAND Flash-based Storage Systems Using Dynamic Program and Erase Scaling, FAST 2014

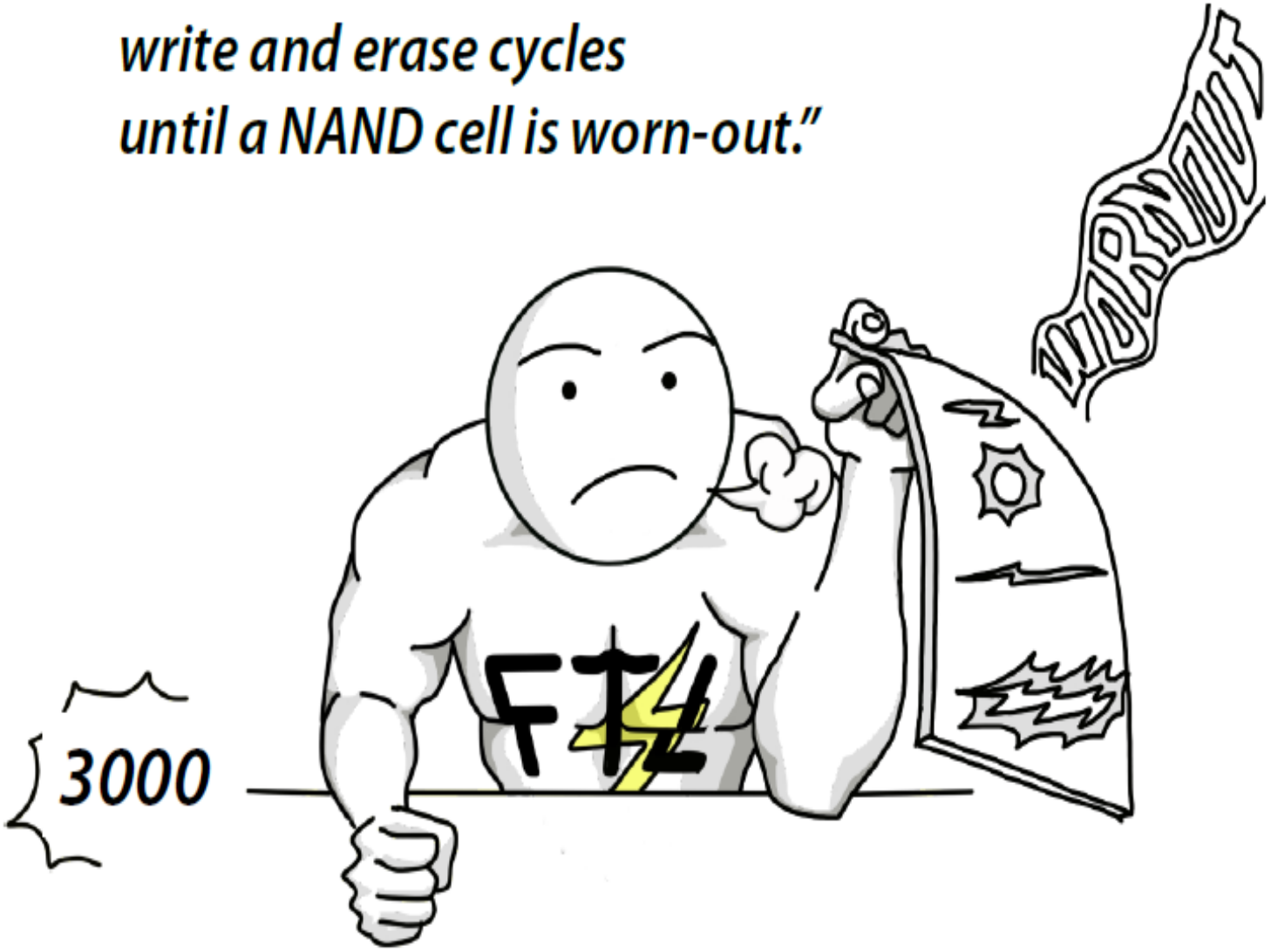
Trend of NAND Device Technologies



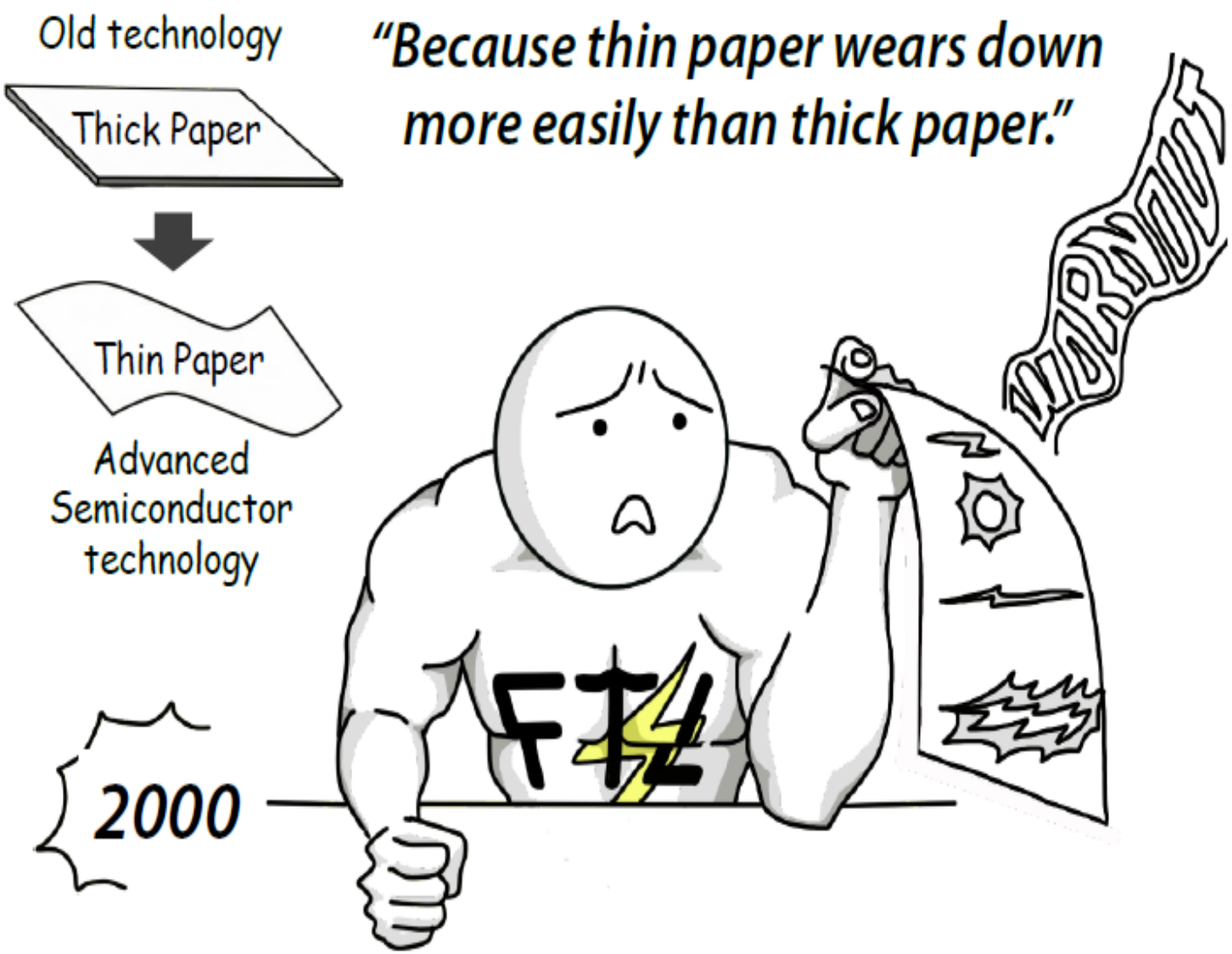
But, NAND endurance is drastically decreased last 6 years.

What is the NAND Endurance ?

"The maximum number of write and erase cycles until a NAND cell is worn-out."



Why is the NAND Endurance Decreased ?



思考

- 磁盘调度算法对于Flash Disk还适用么？

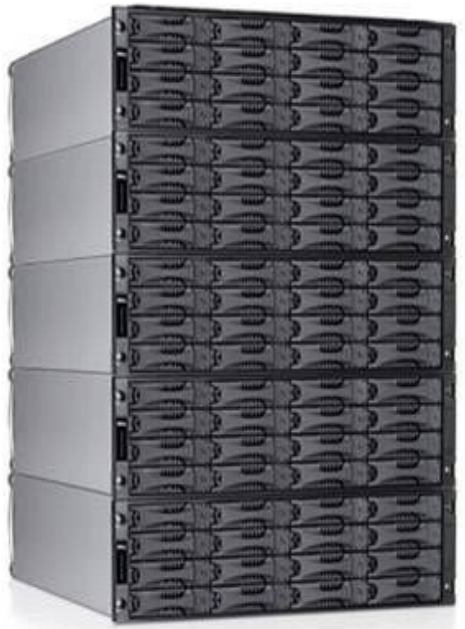
内容提要

- 磁盘的工作原理
- 保证磁盘的可靠性
- 提高I/O访问速度
- 磁盘管理的实例

RAID

- 廉价冗余磁盘阵列,Redundant arrays of inexpensive disks
  - 利用冗余技术提高可靠性
  - 利用并行提高性能
- 具有N个磁盘的磁盘组中某些磁盘比特定磁盘具有较高的出错率。假设单个磁盘的平均损坏时间为100000小时，具有100个磁盘的阵列中至少一个磁盘的平均损坏时间则为100000/100=1000小时，大约是41.66天

Prof. David A. Patterson



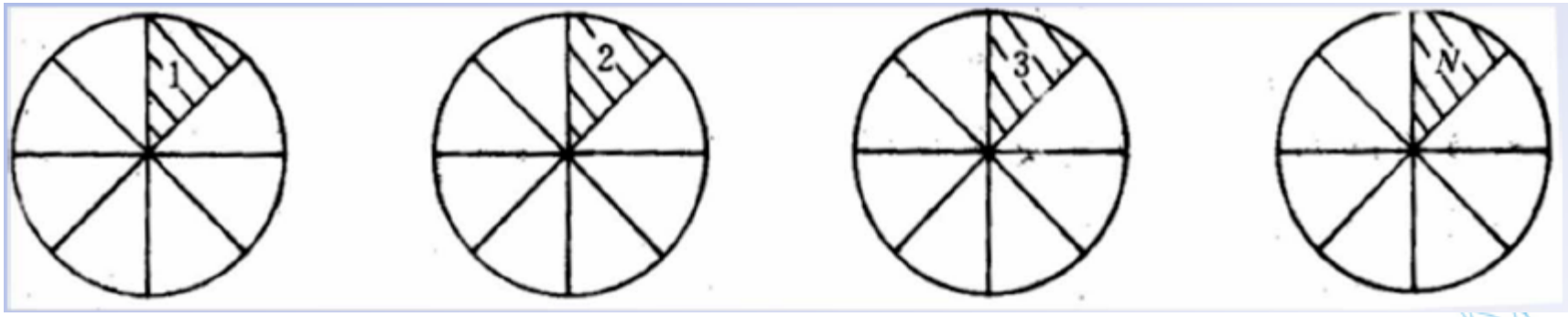
RAID

- 一种把多块独立的硬盘（物理硬盘）按照不同方式组合起来形成一个硬盘组（逻辑硬盘），从而提供比单个硬盘更高的存储性能和提供数据冗余的技术。
- 组成磁盘阵列的不同方式称为RAID级别(RAID Levels)。
- 数据冗余的功能是在用户数据一旦发生损坏后，利用冗余信息可以使损失数据得以恢复，从而保障了用户数据的安全性。

RAID的优点

- 成本低，功耗小，传输速率高
  - RAID比起传统的大直径磁盘驱动器来，在同样的容量下，价格要低许多。
  - RAID让很多磁盘驱动器并行传输数据，比单个磁盘驱动器提高几倍、几十倍甚至上百倍的速率。有效缓解了快速的CPU与慢速的磁盘之间的矛盾
- 可提供容错功能
  - 普通磁盘驱动器只能通过CRC(循环冗余校验)码提供简单的容错，RAID建立在每个磁盘驱动器的硬件容错功能之上，可提供更高的安全性。

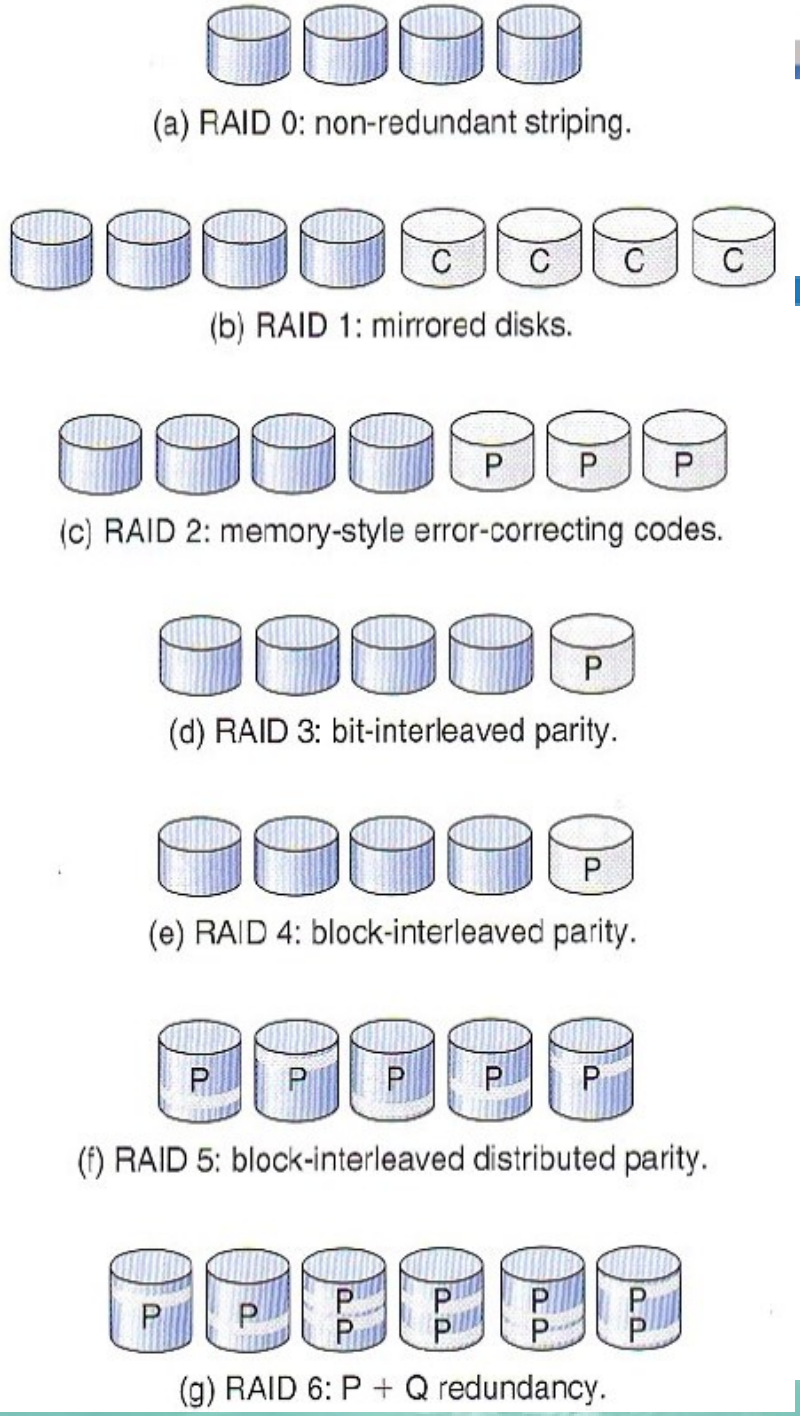
数据分段并行交叉存取



- 数据分段：把一个文件的数据分成多个条带（striping）写到多个硬盘，每个条带的大小可以按需调整。

RAID分级

- RAID有六个级别,其级别分别是0、1、2、3、4、5,6,后来还出现了RAID0+1(也称RAID10)。



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

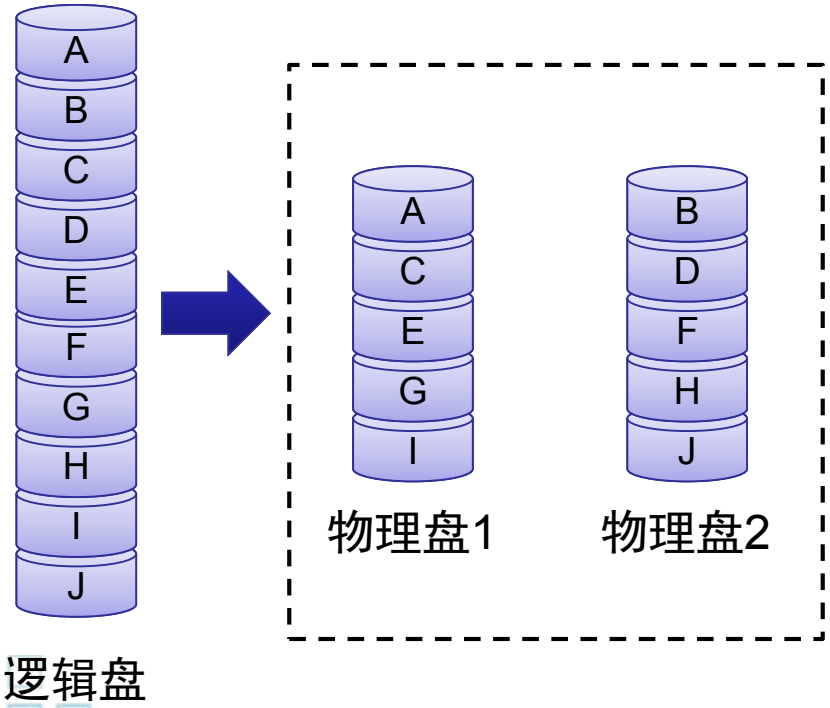
(f) RAID 5: block-interleaved distributed parity.

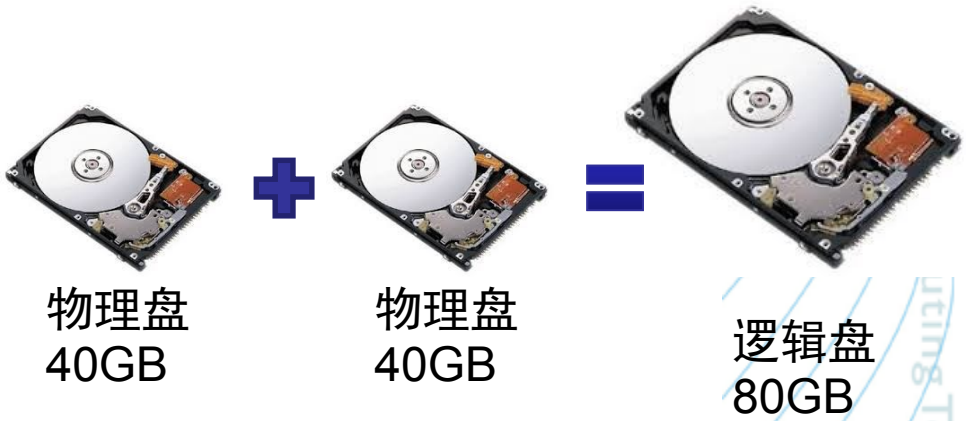
(g) RAID 6: P + Q redundancy.

演示：
http://www.fujitsu.com/global/products/computing/storage/eternus/glossary/raid/raid0-w.html

RAID 0

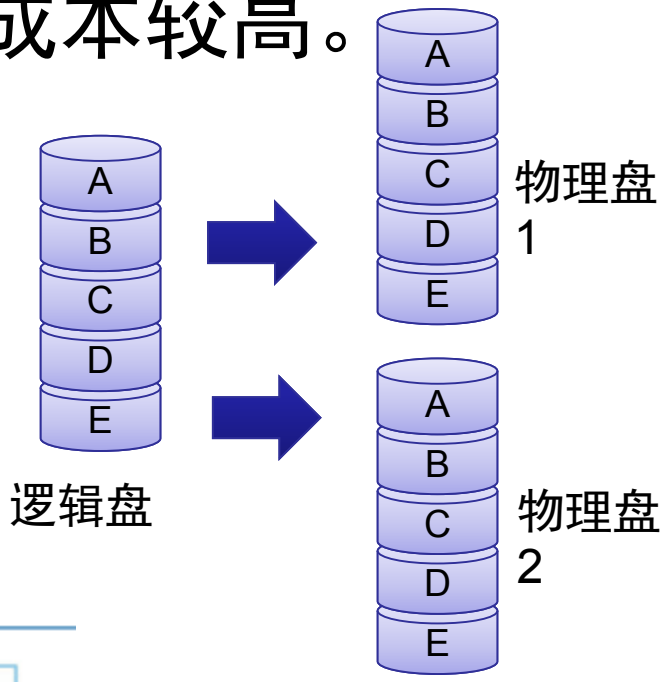
- 该级仅提供了并行交叉存取。它虽然有效提高了磁盘I/O速度，但并无冗余校验功能。





RAID1

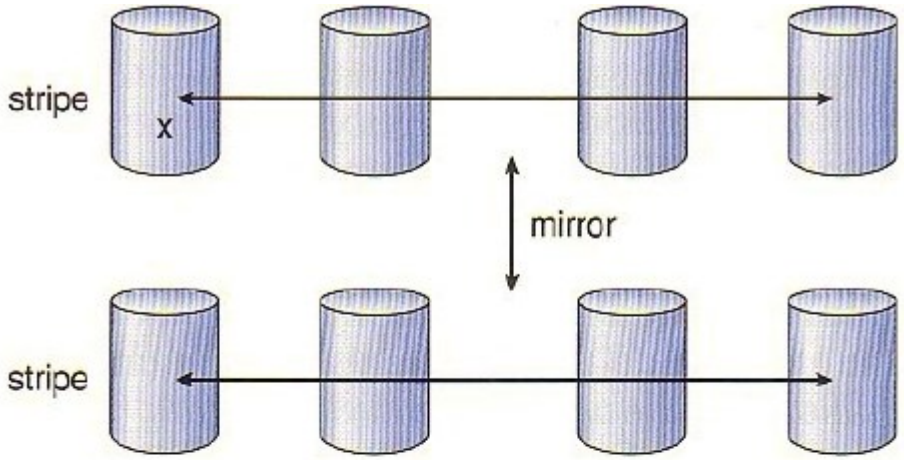
- 镜像磁盘冗余阵列，将每一数据块重复存入镜像磁盘，以改善磁盘机的可靠性。镜像盘也称拷贝盘，使有效容量下降了一半，成本较高。



读性能好，写性能由最差的磁盘决定；
可靠性高；
代价较高。

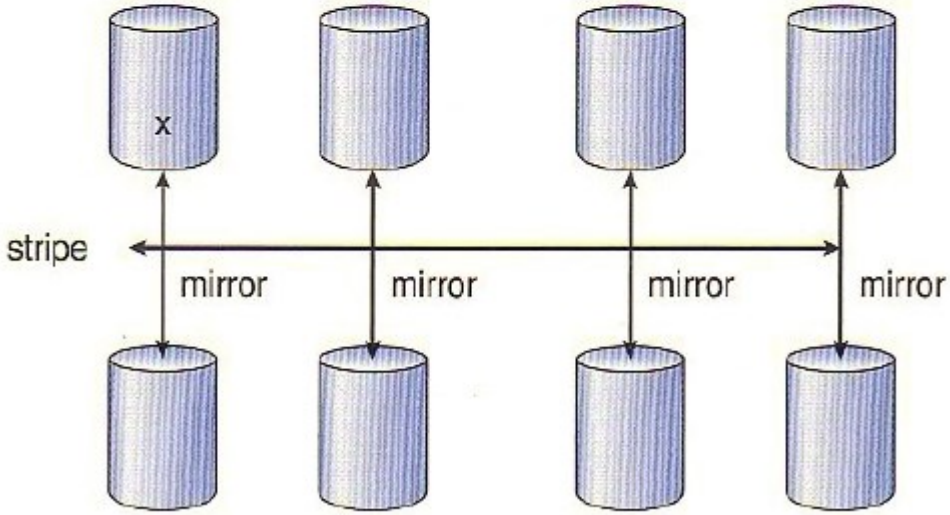
RAID 10

- 综合RAID 0和RAID 1的特点



a) RAID 0 + 1 with a single disk failure.

先分块后镜像

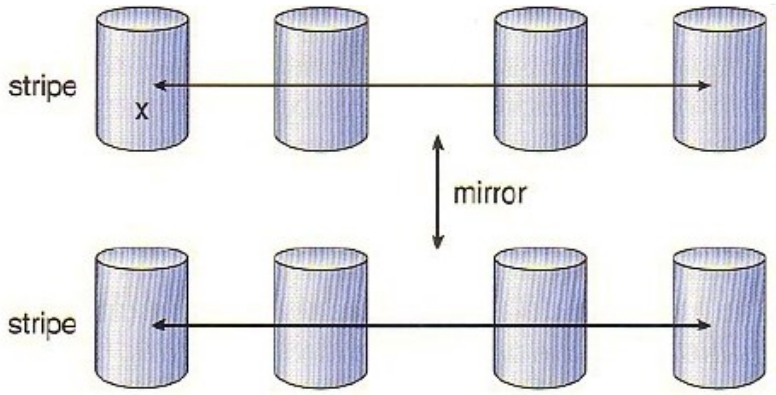


b) RAID 1 + 0 with a single disk failure.

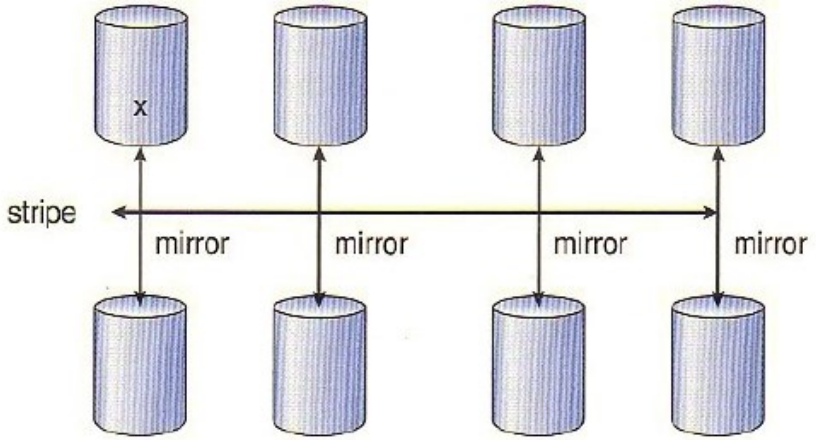
先镜像后分块

RAID 0+1 vs RAID 1+0

- n块盘中1块盘故障
  - 都不会引起数据丢失
- 2块盘故障引起数据丢失
  - RAID 0+1
    - n/(2n-2)
  - RAID 1+0
    - 1/(n-1)
- 3块盘？
- n/2块盘？



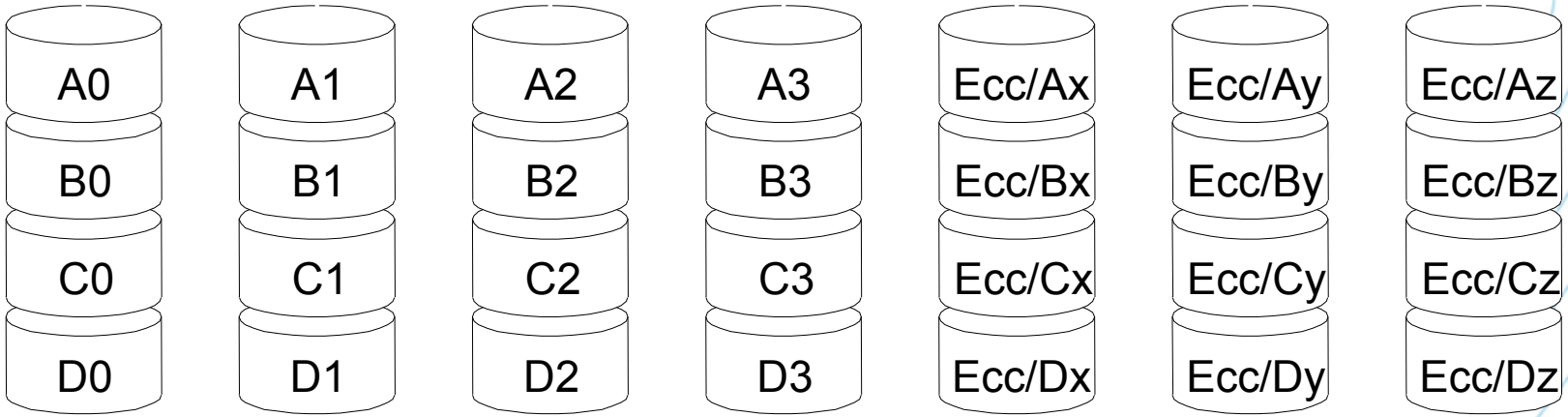
a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.

RAID2

- 采用海明码纠错的磁盘阵列，将数据位交叉写入几个磁盘中。按位条带化。



校验码

- 奇偶校验
  - 偶校验：$P = a_0 \oplus a_1 \oplus a_2 \oplus \cdots \oplus a_{n-1}$
  - 奇校验：$P = not\ (a_0 \oplus a_1 \oplus a_2 \oplus \cdots \oplus a_{n-1})$
- 海明码
  - 要能纠正信息字中的单个错误，所需的最少码距为3。实现这种纠正的方法之一是海明码。它是一种多重(复式)奇偶检错系统。它将信息用逻辑形式编码，以便能够检错和纠错。
  - 以4位数据为例，需要3个校验位 $2^P \geq P+D+1$
  - $A=B_1 \oplus B_3 \oplus B_5 \oplus B_7=0$ 得 $P_1=D_1 \oplus D_2 \oplus D_4$
  - $B=B_2 \oplus B_3 \oplus B_6 \oplus B_7=0$ 得 $P_2=D_1 \oplus D_3 \oplus D_4$
  - $C=B_4 \oplus B_5 \oplus B_6 \oplus B_7=0$ 得 $P_3=D_2 \oplus D_3 \oplus D_4$

XOR数据恢复原理

- 多个数字XOR的时候，有两个特点：
  a) 结果与运算顺序无关。也就是
  (a XOR b) XOR c = a XOR (b XOR c)。
  b) 各个参与运算的数字与结果循环对称。（证明？）
  如果 a XOR b XOR c = d，那么
  a = b XOR c XOR d;
  b = a XOR c XOR d;
  c = a XOR b XOR d。

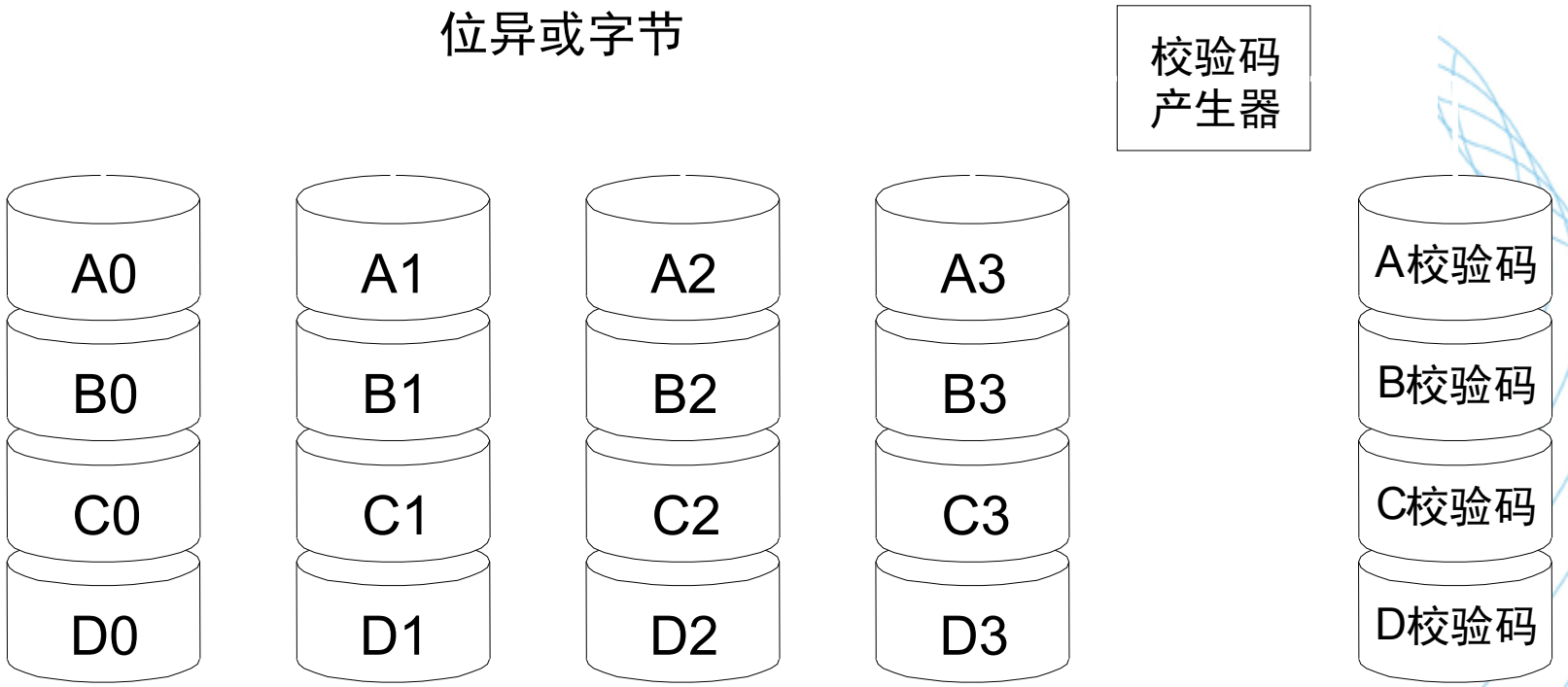
RAID容错就是利用了XOR运算的这些特点。a、b、c、d就可以看作是四颗磁盘上的数据，其中三个是应用数据，剩下一个是校验。碰到故障的时候，不管哪个找不到了，都可以用剩下的三个数字XOR一下算出来。在实际应用中，阵列控制器一般要先把磁盘分成很多条带，然后再对每组条带做XOR。

RAID2的特点

- 并行存取，各个驱动器同步工作。
- 使用海明编码来进行错误检测和纠正，数据传输率高。
- 需要多个磁盘来存放海明校验码信息，冗余磁盘数量与数据磁盘数量的对数成正比。
- 是一种在多磁盘易出错环境中的有效选择，并未被广泛应用，目前还没有商业化产品。

RAID 3

- 采用奇偶校验冗余的磁盘阵列，也采用数据位交叉，阵列中只有一个校验盘。



RAID3的特点

- 将磁盘分组，采用字节级别的条带，读写要访问组中所有盘，每组中有一个盘作为校验盘。
- 校验盘一般采用奇偶校验。
- 简单理解：先将分布在各个数据盘上的一组数据加起来，将和存放在冗余盘上。一旦某一个盘出错，只要将冗余盘上的和减去所有正确盘上的数据，得到的差就是出错的盘上的数据。【怎么知道那个盘出错？】
- 缺点：
  - 恢复时间较长
  - 读写性能的水桶效应瓶颈

RAID4

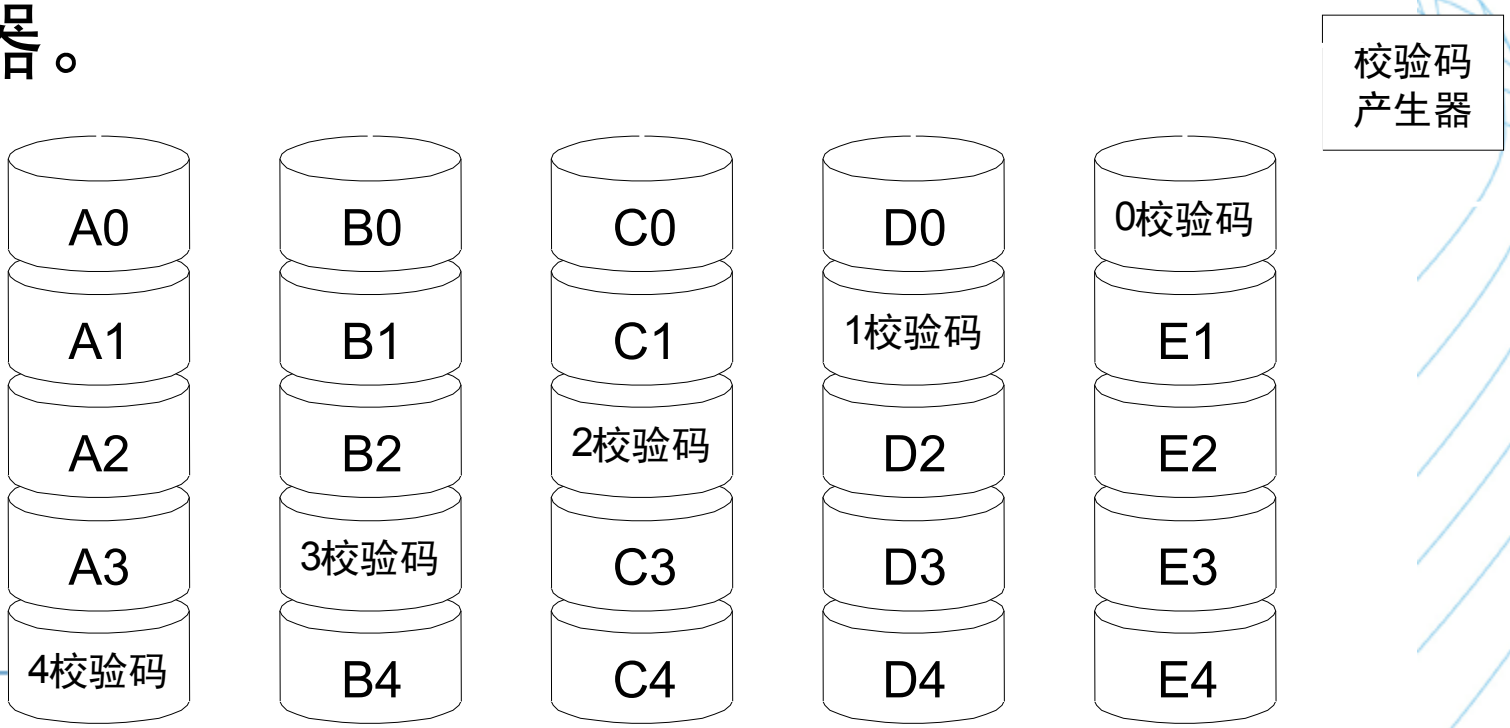
- 并行处理磁盘阵列：一种独立传送磁盘阵列，采用数据块交叉，用一个校验盘。将数据按块交叉存储在多个磁盘上。

RAID4的特点

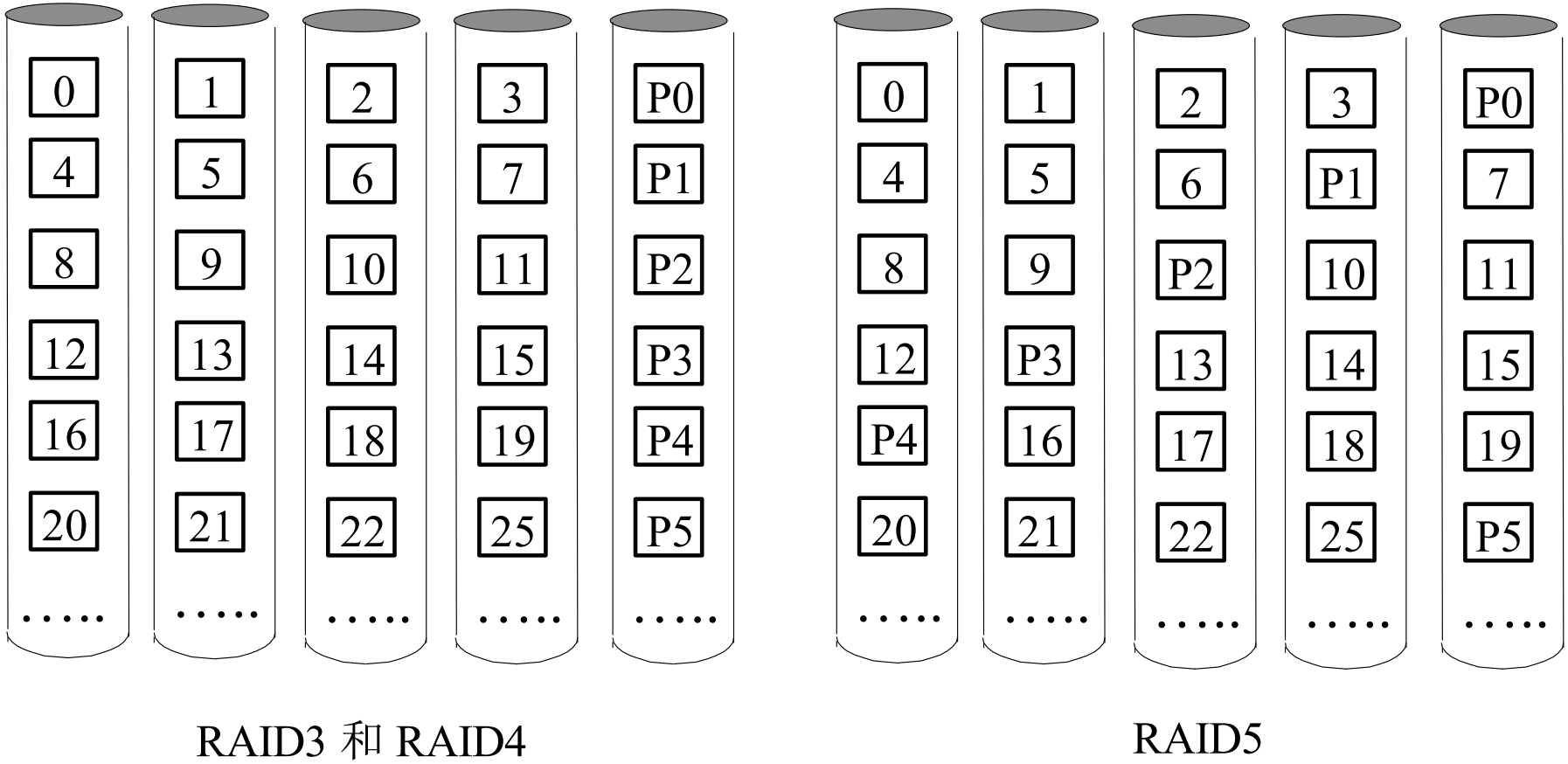
- 冗余代价与RAID3相同
- 访问数据的方法与RAID3不同
- 在RAID3中，一次磁盘访问将对磁盘阵列中的所有磁盘进行（同步）操作。
- RAID4出现的原因：希望使用较少的磁盘参与操作，以使磁盘阵列可以并行进行多个数据的磁盘操作。
- 随机读快，随机写慢（竞争同一个校验盘）

RAID5

- 一种独立传送磁盘阵列，采用数据块交叉和分布的冗余校验，将数据和校验都分布在各个磁盘中，没有专门的奇偶校验驱动器。

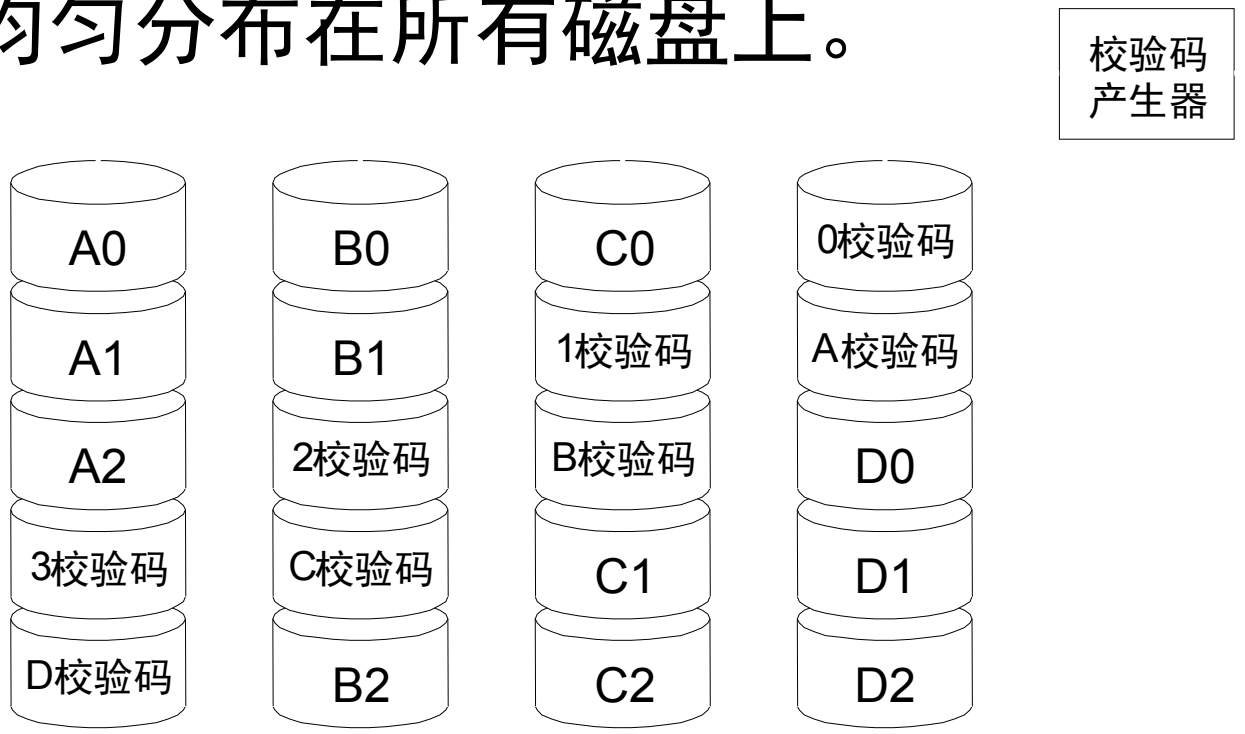


RAID4和RAID5中的信息分布



RAID6

- 双维校验独立存取盘阵列，数据以块（块大小可变）交叉方式存于各盘，检、纠错信息均匀分布在所有磁盘上。



RAID6的特点

- 写入数据要访问1个数据盘和2个冗余盘；
- 可容忍双盘出错；
- 存储开销是RAID5的两倍（多一个冗余盘）。

RAID小结

- 条带化：一个字节块可能存放在多个数据盘上
  - 优点：并行存取，性能好，磁盘负载均衡
  - 缺点：可靠性、不同IO请求需要排队
- 镜像：数据完全拷贝一份
  - 优点：可靠性
  - 缺点：存储开销
- 校验：数据通过某种运算（异或）得出，用以检验该组数字的正确性
  - 优点：可靠性，快速恢复
  - 缺点：开销

RAID比较

级别	说明	特点	磁盘数量	最少磁盘数量
0	条带化	速度加倍，可靠性减半，容量不变	n	2
1	镜像	速度不变，可靠性加倍，容量减半	n*2	2
0+1	组内条带化、组间镜像	速度加倍，可靠性加倍，容量减半	n*2	4
1+0	组内镜像，组间条带化	速度加倍，可靠性加倍，容量减半	2*n	4
2	海明码校验	校验独立存储，数据位交叉（条带化），自动纠正1个盘错	n+log(n)	7*
3	奇偶校验	校验独立存储，数据位交叉（条带化）容1块盘错	n+1	3
4	奇偶校验	校验独立存储，数据块交叉，容1块盘错	n+1	3
5	奇偶校验	校验交叉存储，数据块交叉，容1块盘错	n+1	3
6	奇偶校验	校验交叉存储，数据块交叉，容2块盘错	n+2	4

内容提要

- 磁盘的工作原理
- 保证磁盘的可靠性
- 提高I/O访问速度
- 磁盘管理的实例

提高I/O速度的主要途径

- 选择性能好的磁盘
- 并行化
- 采用适当的调度算法
- 设置磁盘高速缓冲区

提高磁盘I/O速度——缓存

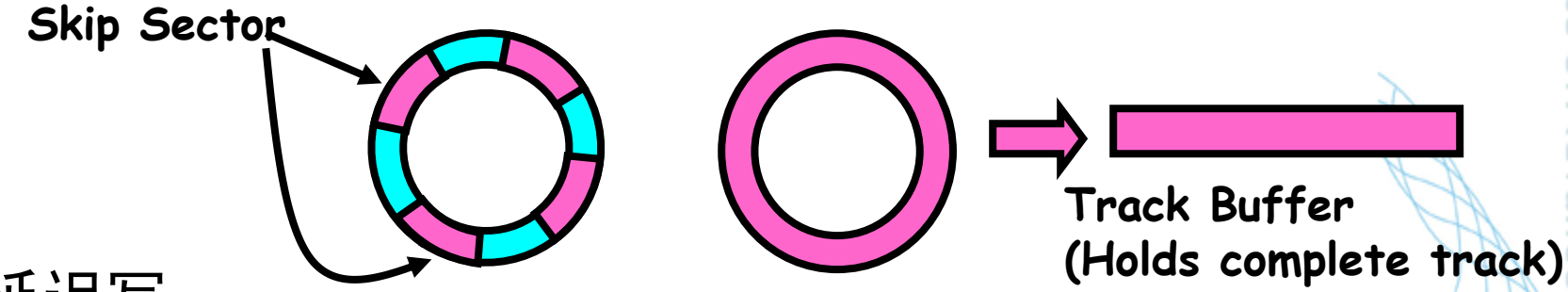
- 磁盘高速缓存的形式
  - 独立缓存（固定大小）
  - 以虚拟内存为缓存（弹性大小）
- 数据交付
  - 直接交付（copy开销）
  - 指针交付（内存管理复杂）
- 置换算法（LRU）
- 周期性写回
  - 周期性地将disk cache中被修改过的内容写回磁盘

优化数据布局

- 优化物理块的分布
  - 连续摆放，使磁头的移动距离最小，比如：安排一个文件的两块数据块在同一磁道，消除磁头在磁道间的移动（Defreg）
- 优化索引节点的分布
  - 访问文件时，先访问索引节点，然后再去访问盘块
  - 将磁盘上的所有磁道分成若干个组，在每个组中都含有索引节点、盘块和空闲盘块表，使得索引节点盘块与存放文件的盘块间距离最小（如：Linux的磁盘文件逻辑结构）

提高磁盘I/O速度的其他方法

- 提前读
  - 顺序访问时，常采用提前读入下一块到缓冲区中



- 延迟写
  - 将本应立即写回磁盘的数据挂到空闲缓冲区的队列的末尾。直到该数据块移到链头时才将其写回磁盘，再作为空闲区分配出去（可靠性？）
- 虚拟盘
  - 利用内存空间去仿真磁盘（RAM盘）
  - Vitual disk 与disk cache的区别是：
    - Vitual disk的存放的内容由用户完全控制
    - Disk cache中的内容完全是由操作系统控制

内容提要

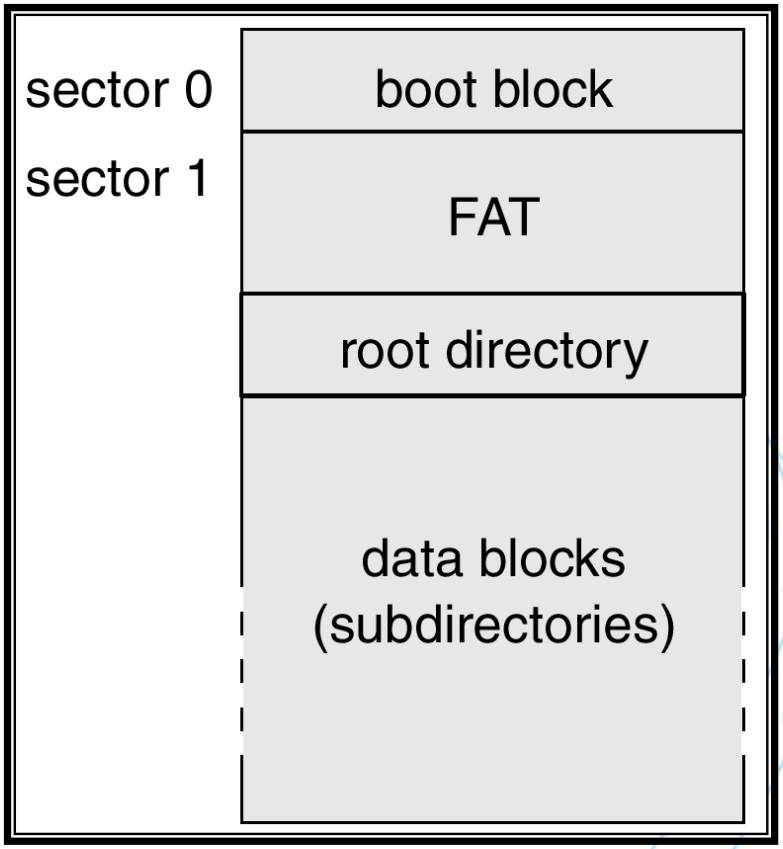
- 磁盘的工作原理
- 保证磁盘的可靠性
- 提高I/O访问速度
- 磁盘管理的实例

Windows2000磁盘管理系统

- 跨磁盘管理（DISK SPANNING）
- 容错（FAULT TOLERANCE）

MS—DOS的分区方式缺点

- 对大多磁盘设置的改变需要重起操作系统才能生效。
- WINDOWS NT的注册表中为MS—DOS方式的分区保存了多分区盘的配置信息
- 每个卷有一个唯一的从A到Z的驱动器名

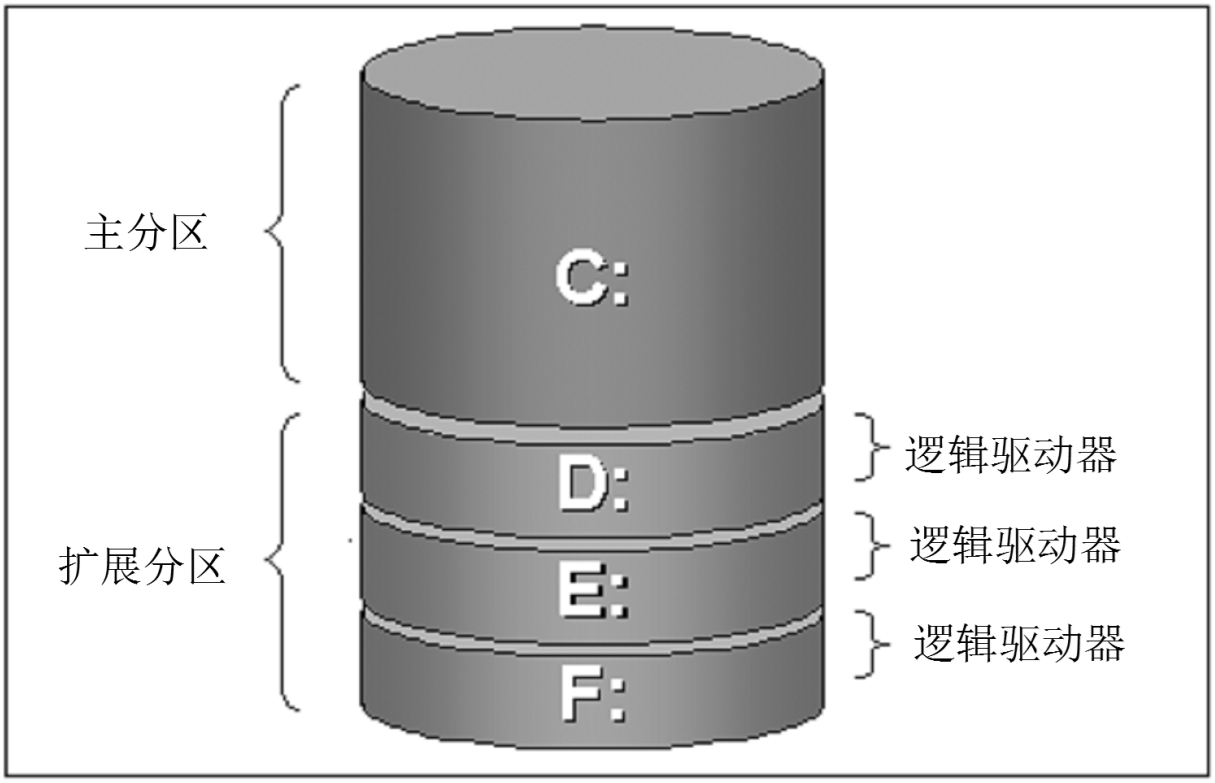


基本术语

- 盘（DISKS）—种物理存储设备，如硬盘，3.5英寸软盘，光盘。
- 盘被分为扇区（SECTOR），可寻址的大小固定的块。扇区的大小是由硬件决定的。硬盘扇区大小通常为512字节。光盘的扇区一般为2048字节。未来的X86系统可能支持更大的硬盘扇区。
- 分区是盘上连续扇区的集合，分区表或其他盘管理数据库中保存了分区的起始扇区，大小，和其他特性。
- 简单卷（SIMPLE VOLUME）是这样一种对象，它代表文件系统驱动程序作为一个独立单元所管理的、来自一个分区的扇区。
  - 简单卷是物理磁盘的一部分，但它工作时就好像是物理上的一个独立单元。简单卷是相当于 Windows NT 4.0 及更早版本中的主分区的动态存储。当您只有一个动态磁盘时，简单卷是您可以创建的唯一卷。
- 多分区卷（MULTIPATITION VOLUME）是这样一种对象，它代表文件系统驱动程序作为一个独立单元所管理的，来自多个分区的扇区。多分区卷提供简单卷所不支持的更好的性能，可靠性，大小的特性。

基本盘和动态盘的概念

- WINDOWS 2000 把基于MS—DOS分区方式的盘称为**基本盘**。使用这种方式管理磁盘时，首先要将磁盘划分成一个或多个磁盘分区，然后才可以向磁盘中存储数据。分区的类型分为主分区、扩展分区和逻辑驱动器。

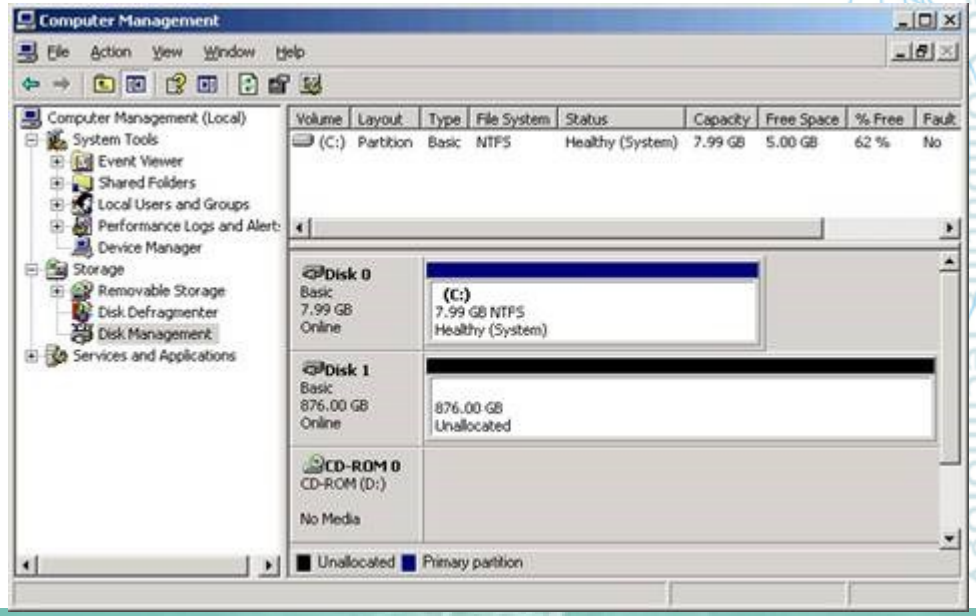


- 动态盘对WINDOWS 2000来说是一个新的概念，实现了比基本盘更具适应性的分区机制。实现许多基本磁盘不能实现的功能，如提高磁盘的访问速度、提供容错功能等。动态磁盘通过将磁盘划分为卷的方式来管理磁盘空间，卷的类型分为简单卷、带区卷、跨区卷、镜像卷和RAID-5卷。

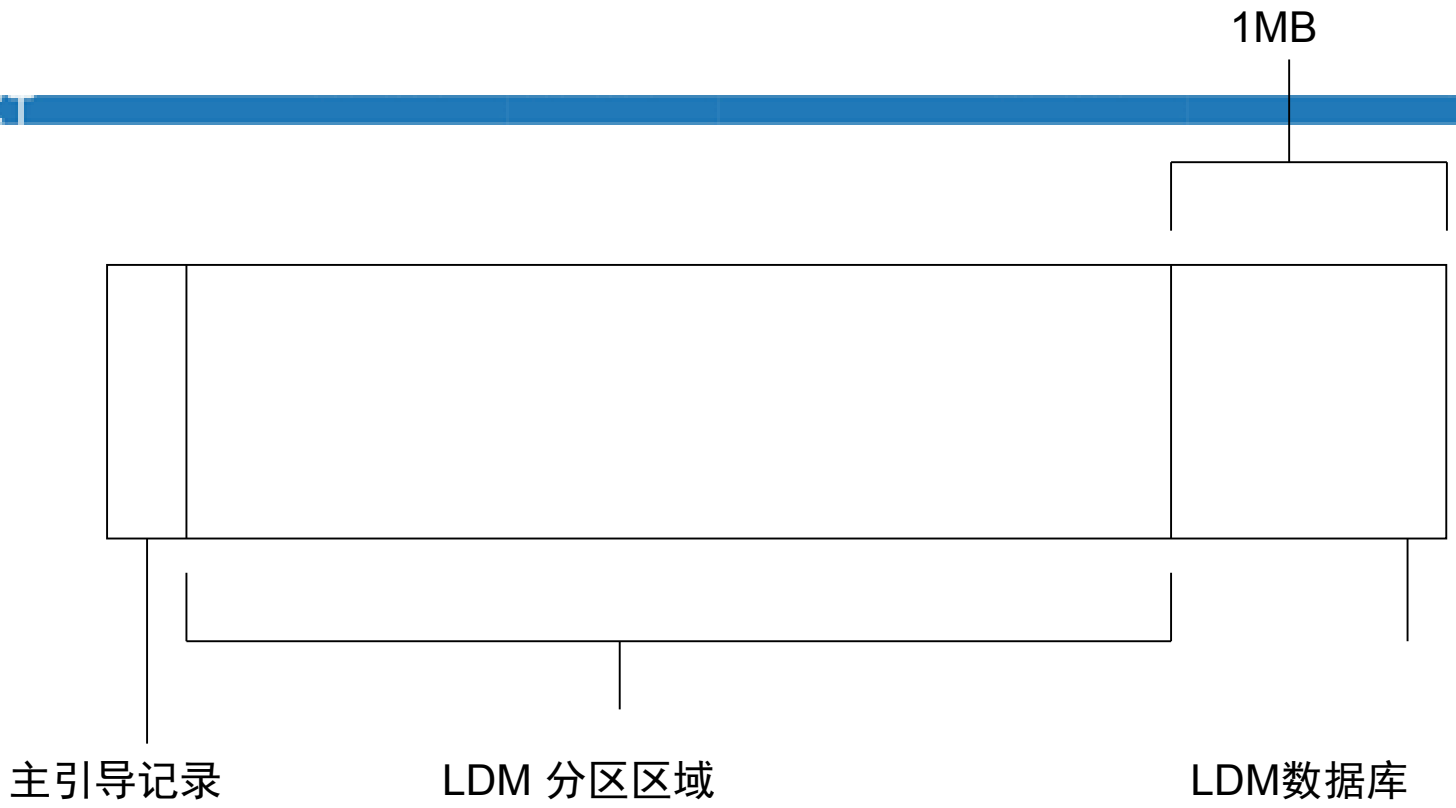
基本盘和动态盘之间不同点

- 动态盘支持创建新的多分区卷。
- 基本盘的多分区卷的配置信息保存在注册表中。动态盘的多分区卷的配置信息保存在磁盘中。
- 动态盘的不足：它所应用的分区的格式是专有的，并且不和其它的操作系统兼容

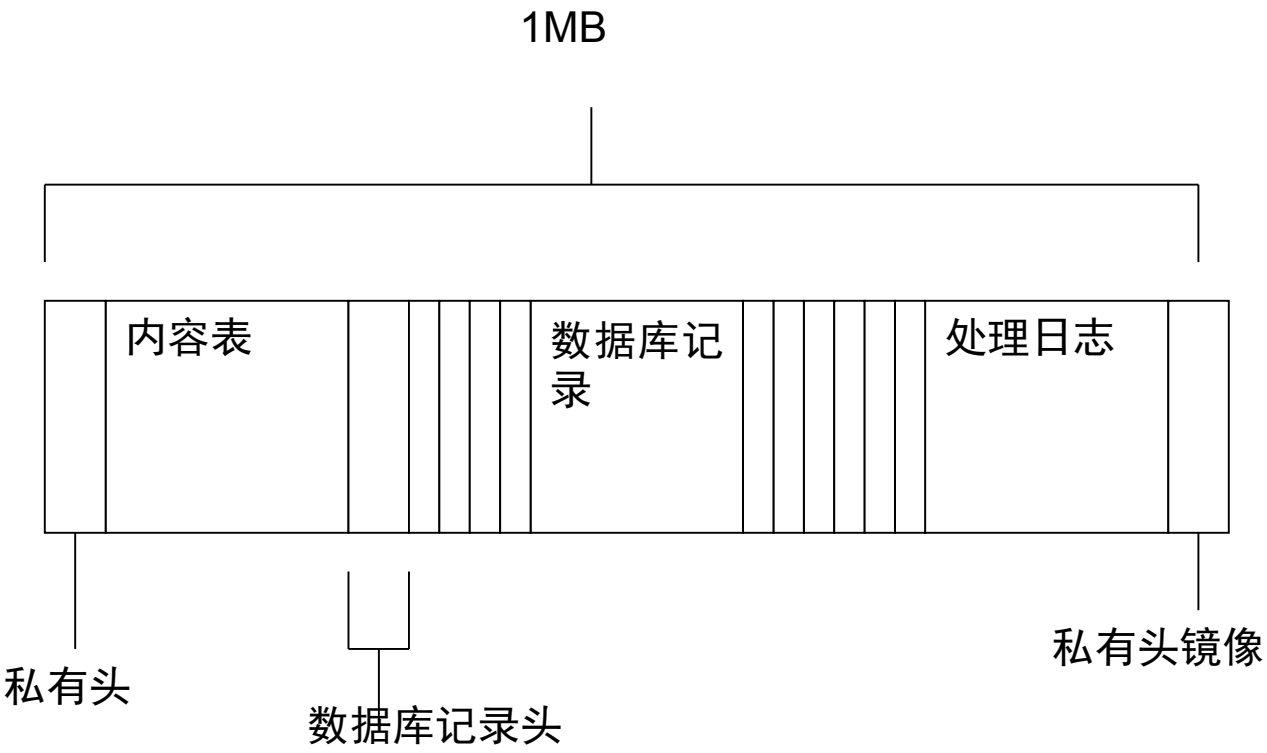
- 系统卷是WINDOWS2000存放引导文件的地方，包括引导程序（NTLDR）和NTDETECT，
- 引导卷是WINDOWS2000存放系统文件，如NTOSKRNL.EXE核心内核文件的地方



- LDM（Logical Disk Manager）的分区在一个盘的MSDOS分区表中并没有体现出来，所以被称为软分区，而MSDOS分区被称为硬分区。



动态盘的内部组织



数据库结构

- 私有头：GUID，磁盘组的名字（该名字是由Dg0和计算机的名字一起组成，例如SusanDg0，意味着计算机的名字是Susan）和一个指向数据库内容表的指针。为了保证可靠性，LDM在磁盘的最后一个扇区保存了私有头的拷贝。
- 数据库内容表有16个扇区大小，其中包含关于数据库布局的信息。
- 数据库记录区域紧接着内容表，并将内容表后第一个扇区作为数据库记录头。这个扇区中存储了数据库记录区的信息，包括其所包含的记录个数，数据库相关的磁盘组的名字和GUID，以及LDM用于创建下一项的序列号。

- 数据库中的每一项可以是如下四种类型之一：分区，磁盘，组件，卷。LDM把每一项与内部对象的标识符联系到一起。
- 在最低的级别，分区项描述软分区，它是在一个盘上的连续区域。存储在分区项中的标识符把这个项与一个组件和一个磁盘项联系起来。
- 磁盘项代表一个磁盘组中的动态盘，包括磁盘的GUID。
- 组件项像一条链子把一个或多个分区项和与分区相连的卷项联系起来。
- 卷项存放这个卷的GUID，卷的大小和状态，驱动器的名字。比一个数据库记录大的磁盘项占用多个记录的空间，分区项、组件项和卷项很少占用多个记录的空间。

- LDM需要三个项来描述一个简单卷：分区项、组件项和卷项。
- 分区项描述系统分配给某个卷的磁盘上的一个区域，组件项把一个分区项和一个卷项联系起来，卷项中包含Windows Server 2003内部用来识别卷的GUID。
- 多分区卷需要的项数多于三个。例如，一个条带卷包括最少两个分区项，一个组件项和一个卷项。
- 唯一一种含有一个以上组件项的卷的类型是：镜像卷。镜像卷含有两个组件项，每个只表示这个镜像的一半。
- LDM为每个镜像卷使用两个组件项的目的是：当一个镜像破坏时LDM能够在组件一级将他们分割开来，并创建两个各含有一个组件项的卷。
- 因为简单卷需要三个项，而1MB数据库空间大约可以容纳8000个项，所以在Windows Server 2003中可以创建的卷数目的有效上界大约是2500个。

- LDM数据库的最后部分是事务处理日志区，它包含的几个扇区在数据库信息改变时用来存储备份信息。这样确保在系统崩溃或断电时，LDM能够利用日志把系统恢复到一个正确的状态。

管理工具

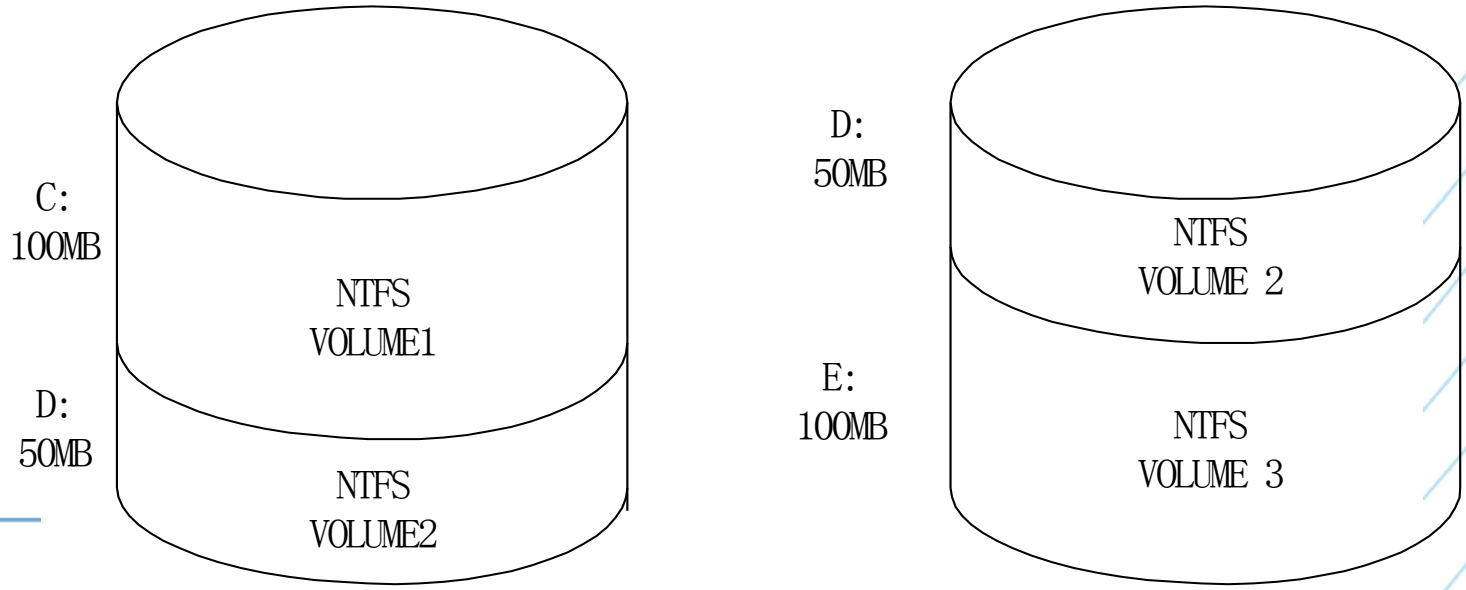
- FtDisk和 DMIO负责识别文件系统驱动程序管理的卷，并将I/O直接从卷映射到组成卷的底层分区。
- 对简单卷来说，通过把卷的偏移量加上卷在磁盘中的起始地址，卷管理器可以保证卷的偏移量被转换成盘的偏移量。
- 对于多分区卷这就复杂多了，因为组成卷的分区可以是不邻接的分区，甚至可以在不同的磁盘中。有一些多分区卷使用数据冗余技术，所以它们需要更多的卷到磁盘的转换工作。

多重分区管理

- 跨分区卷(spanned volume)
- 条带卷（striped volume）
- 镜像卷（mirrored volume）
- 廉价冗余磁盘阵列5卷（RAID-5 volume）

跨分区卷

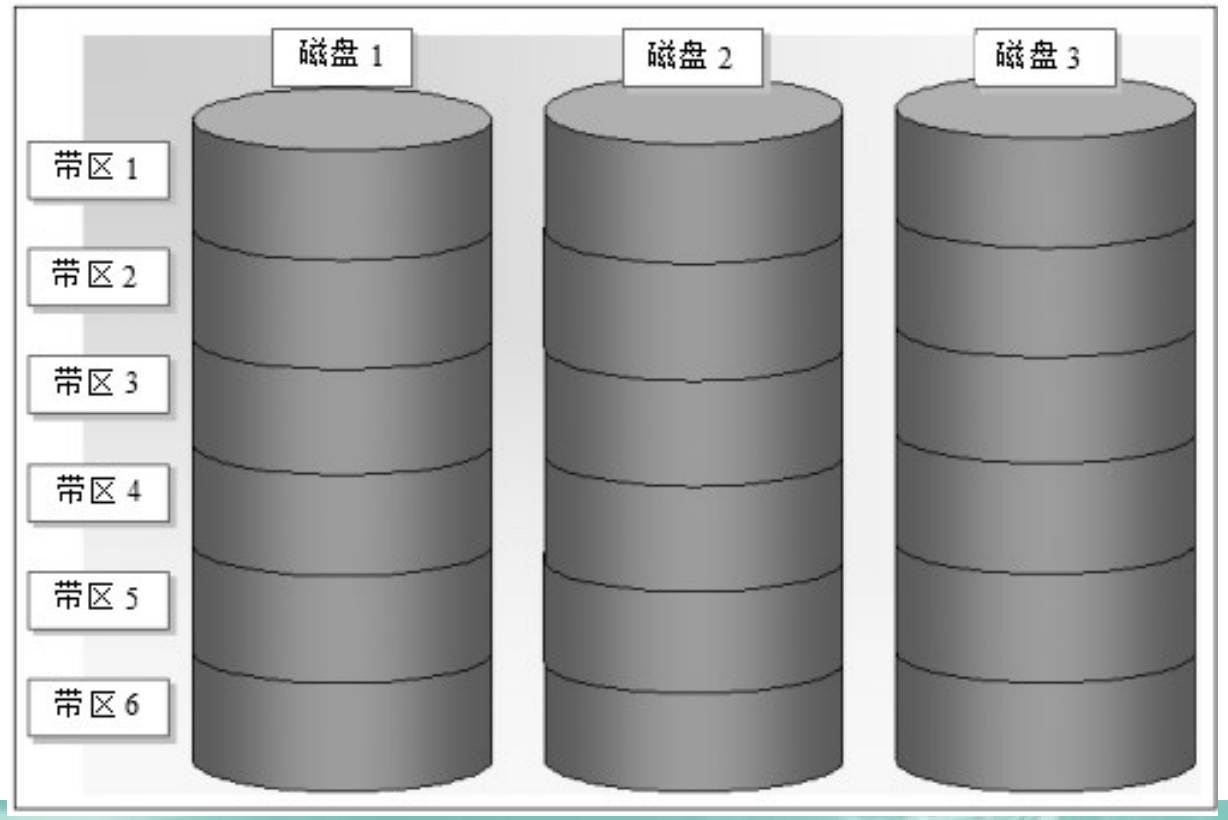
- 跨分区卷可以将多个磁盘的未划分空间合并在一个卷中（至少2个磁盘，最多支持32个磁盘），并赋予一个驱动器号，像访问一个驱动器那样访问跨区卷。



- 一个单独的逻辑卷，最多由在一个或多个磁盘上的32个空闲分区组成。
- 跨分区卷可以用来把小的磁盘空闲区域，或者把两个或更多的小磁盘组成大的卷。
- 卷管理器对Windows Server 2003的文件系统隐藏了磁盘物理配置信息。

条带卷（RAID-0卷）

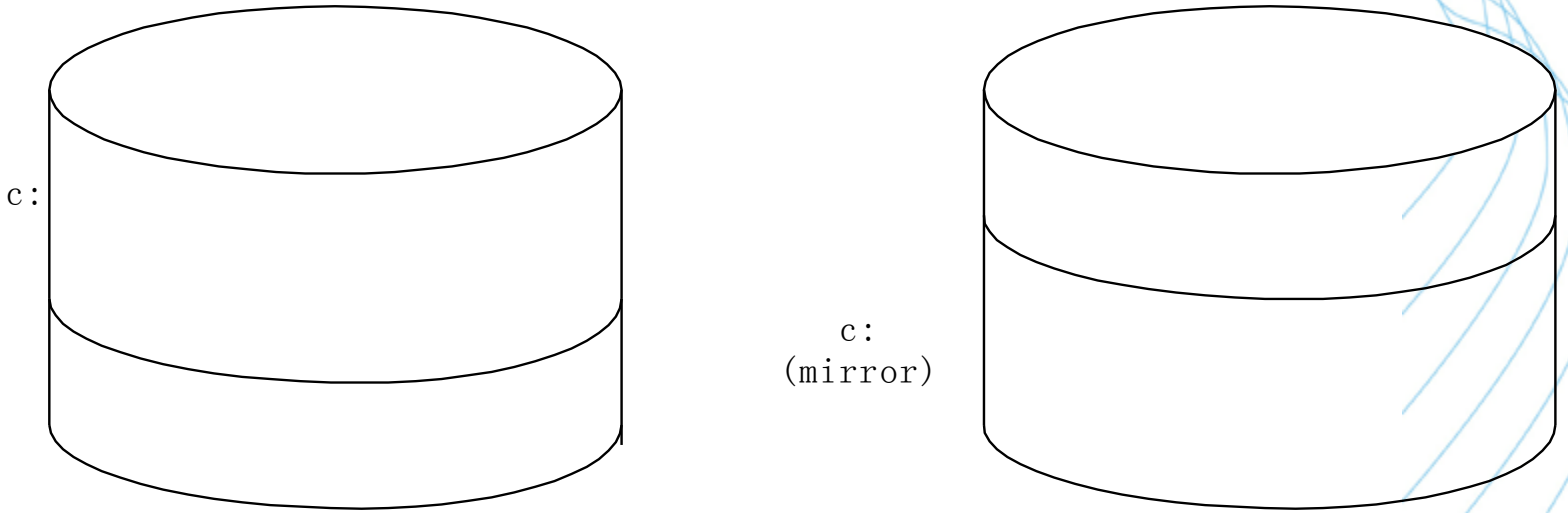
- 通过将多个磁盘的空间组合在一起（至少2个磁盘，最多支持32个磁盘），形成条带卷



- 一系列分区组成的单独的逻辑卷，最多有32个分区并且每个盘一个分区。
- 条带卷中的一个分区不需要占据整个磁盘，唯一的限制是每个盘上的分区大小相同
- 数据能够被平均分配到每个磁盘上

镜像卷

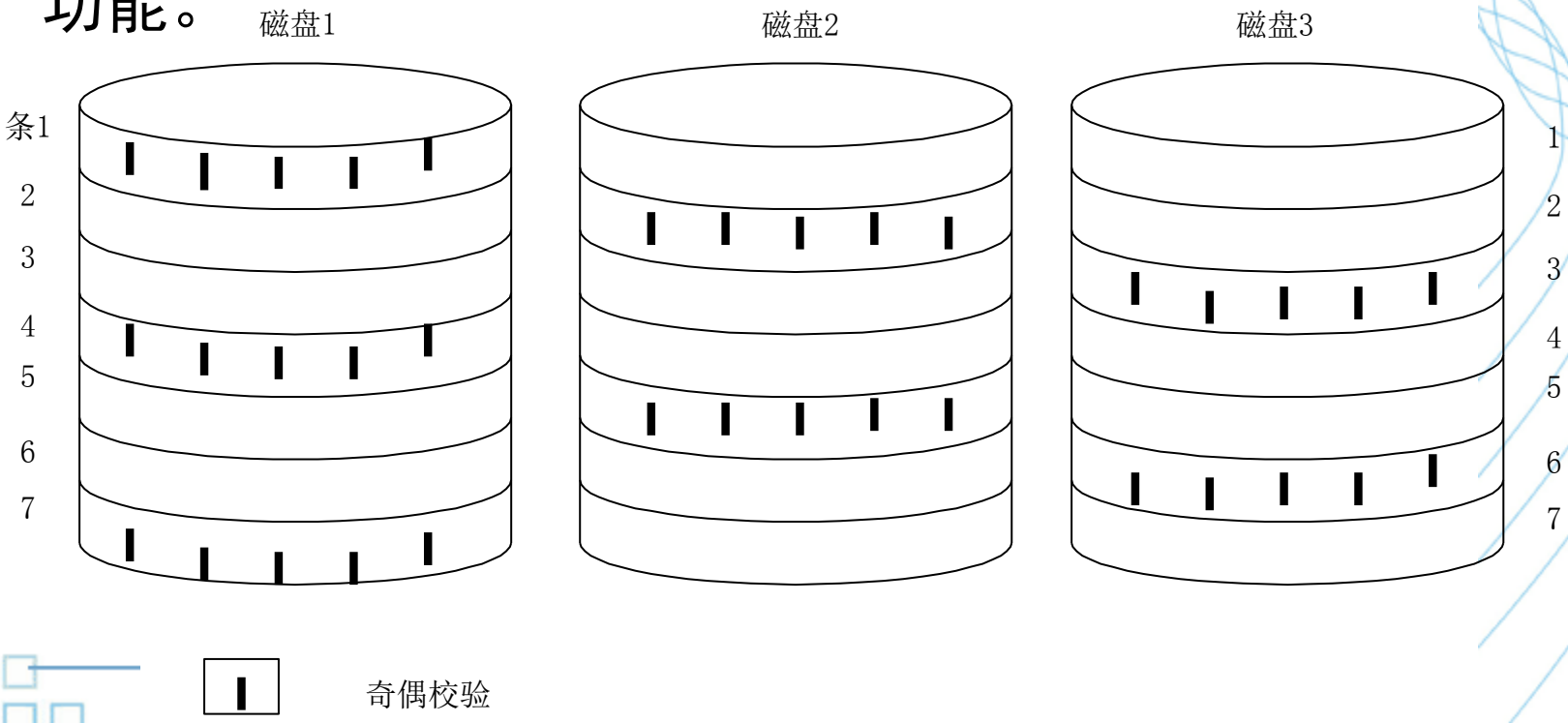
- 可以将两个硬盘合成一个卷来使用。在向镜像卷保存数据时，会将同样的数据同时保存到两个磁盘中，以实现容错功能。



- 一个磁盘上分区的内容被复制另一个磁盘与它等大小的分区中。镜像卷有时也被称为RAID-1。
- 镜像卷能够在主分区和镜像分区之间平衡I/O操作。两个读操作可以同时进行，所以理论上只用一半时间就可以完成。当修改一个文件时，必须写入镜像卷的两个分区，但是磁盘写操作可以异步进行，所以用户态程序的性能一般不会被这种额外的磁盘更新所影响。
- 镜像卷是唯一一种支持系统卷和引导卷的多分区卷。

RAID-5卷

- RAID-5卷：RAID-5卷首先将磁盘划分成带区，在向RAID-5卷保存数据时，会先利用数学算法算出校验码同时写入硬盘中。这样，当RAID-5卷中的一个硬盘损坏时，可利用其他硬盘将数据恢复过来，从而实现容错功能。

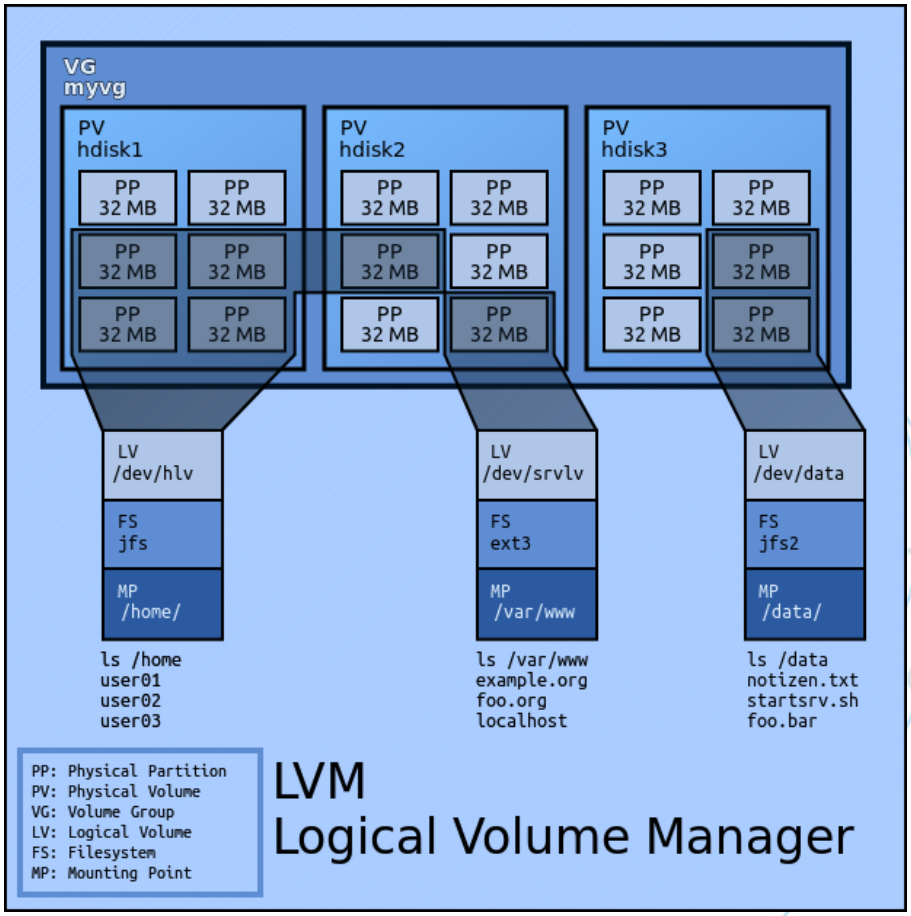


Linux的逻辑卷

- **Logical Volume Manager** (**LVM**) 在Linux Kernel中包含的一个device mapper target，提供逻辑卷管理功能。
  - The **device mapper** is a framework provided by the Linux kernel for mapping physical block devices onto higher-level *virtual block devices*. It forms the foundation of **LVM2**, software **RAIDs** and **dm-crypt** disk encryption, and offers additional features such as **file system snapshots**.
- 目前主流的Linux发行版都支持使用逻辑卷承载根文件系统（操作系统所在文件系统）

几个概念

- 物理卷PV
- 卷组VG
- 逻辑卷LV
- 区块Extents



检错与纠错

码距

- 一个编码系统中任意两个合法编码（码字）之间不同的二进数位（bit）数叫这两个码字的码距，而整个编码系统中任意两个码字的的最小距离就是该编码系统的码距。（图1的码距为1，图2的码距为2）

信息序号	二进码字		
	a_2	a_1	a0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

图 1

信息序号	二进码字			
	a_3	a_2	a_1	a_0
0	0	0	0	0
1	1	0	0	1
2	1	0	1	0
3	0	0	1	1
4	1	1	0	0
5	0	1	0	1
6	0	1	1	0
7	1	1	1	1

图 2

码距与纠错的关系

- 码距越大，纠错能力越强，但数据冗余也越大，即编码效率低了。所以，选择码距要取决于特定系统的参数。数字系统的设计者必须考虑信息发生差错的概率和该系统能容许的最小差错率等因素。要有专门的研究来解决这些问题。

码距	码能力	
	检错	纠错
1	0	0
2	1	0
3	2 或 1	
4	2 加 1	
5	2 加 2	
6	3 加 2	
7	3 加 3	

图3

为了使一个系统能检查和纠正一个差错，码间最小距离必须至少是“3”。最小距离为3时，或能纠正一个错，或能检二个错，但不能同时纠一个错和检二个错。

问题？