中国科学院大学
University of Chinese Academy of Sciences

磁盘、SSD和RAID

中国科学院大学计算机与控制学院
中国科学院计算技术研究所

2019-12-04

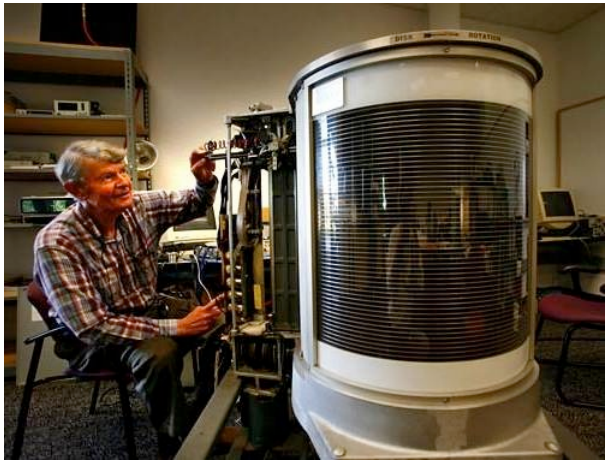
内容提要

- 磁盘
  - 内部结构
  - 性能特性
  - 磁盘调度
- 固态硬盘SSD
- RAID (磁盘阵列)

第一块磁盘

1956，IBM 305 RAMAC

- 50个盘片（disc）
- 总容量5MB
- 每个盘片 24”
- 体积：1.9 m^3
- 重量：910 kg



https://en.wikipedia.org/wiki/IBM_305_RAMAC

磁盘（Hard Disk）

- 持久化的、大容量的、低成本的存储设备：机械，速度慢
- 多种尺寸： 3.5”, 2.5” (曾经 5 ¼” , 1.8”, …)
- 多种容量：100GB ~ 14TB
- 多种接口

2.5”
5TB

.5-1”×4”×5.7”
0.5~6TB

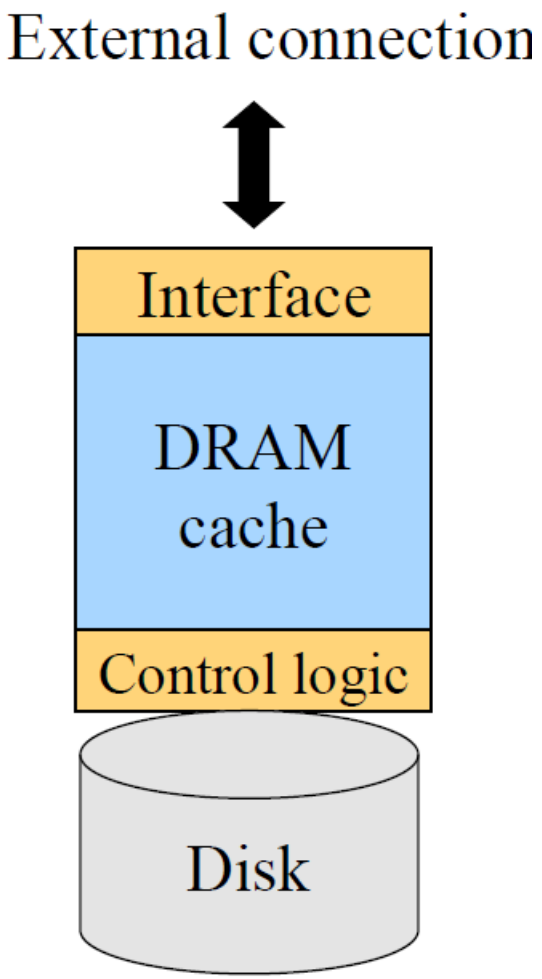
.4-.7”×2.7”×3.9”
0.5~2TB

24mm×32mm×2.1mm
1~256GB

3.5”
12TB

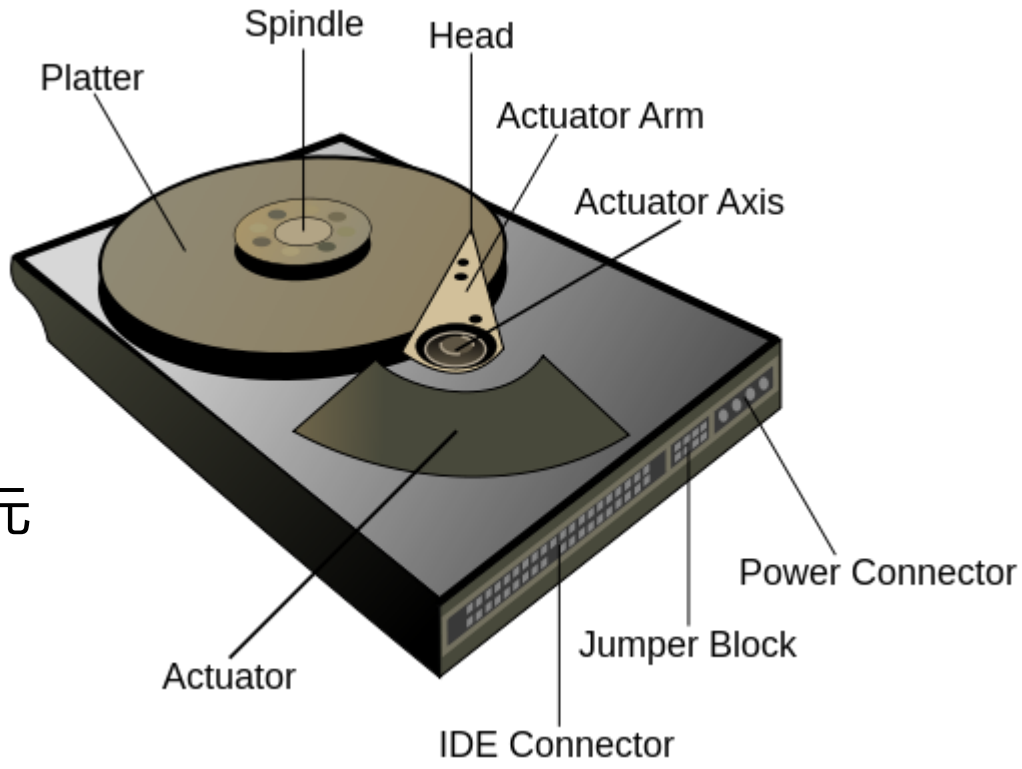
典型的磁盘控制器

- 与主机的接口
  - SATA (1.0, 2.0, 3.0) , ATA
    - 面向对延迟、吞吐率要求不高，大容量的场景
  - SAS (1, 2, 3)， SCSI / Ultra-SCSI
    - 面向低延迟，高吞吐率，高可靠场景
  - FC: Fiber channel
- 缓存
  - 缓冲数据
- 控制逻辑
  - 读写操作
  - 请求调度
  - 缓存替换
  - 坏块检测和重映射

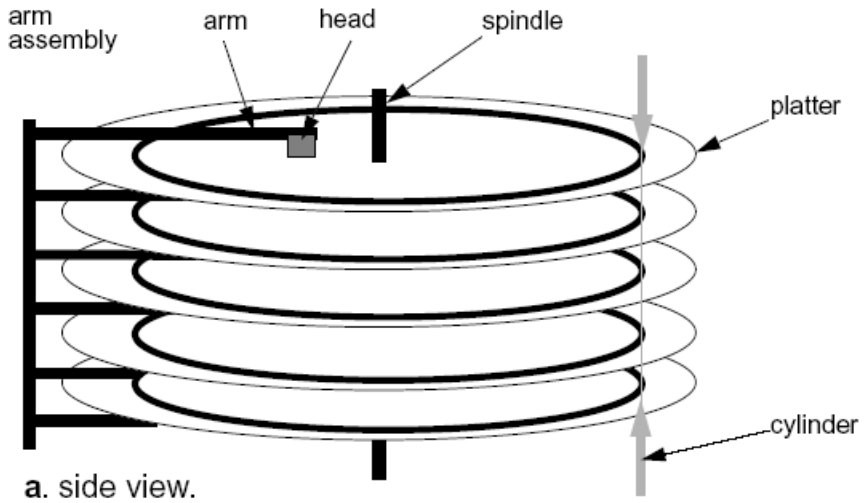


磁盘的结构

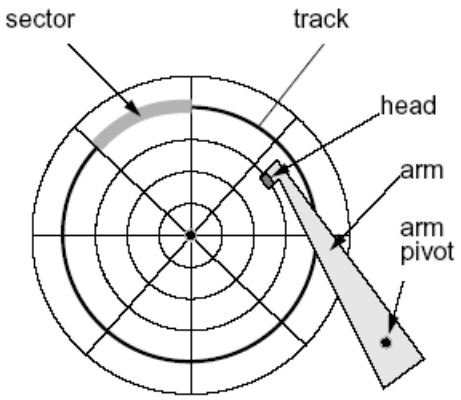
- 盘片：一组
  - 按一定速率旋转
- 磁道 (Track)
  - 位于盘片表面的同心圆
  - 用于记录数据的磁介质
  - bit沿着每条磁道顺序排列
- 扇区 (Sector)
  - 磁道划分为固定大小的单元
    一般为512字节
- 磁头：一组
  - 用于读写磁道上的数据
- 磁臂：一组
  - 用于移动磁头（多个）



磁盘的结构



a. side view.

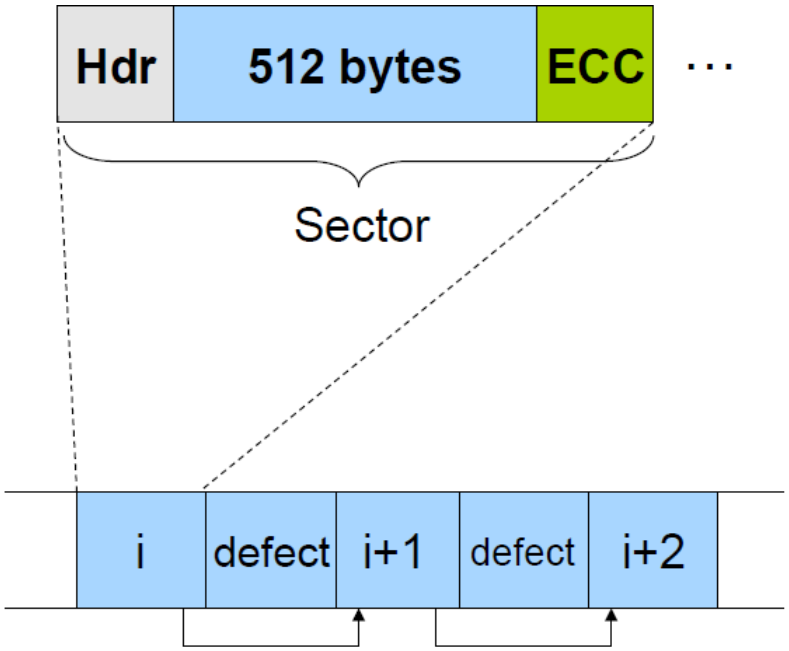


b. top view.

- 柱面 (Cylinder)
  - 由所有盘片上半径相同的磁道组成
- Zone
  - 不同磁道的扇区个数不同：外道多，内道少
  - 所有柱面划分为zone，同一zone中每条磁道的扇区数相同
  - 1000-5000个柱面/zone，其中几个为备用柱面（spare cylinder）

磁盘扇区（Sector）

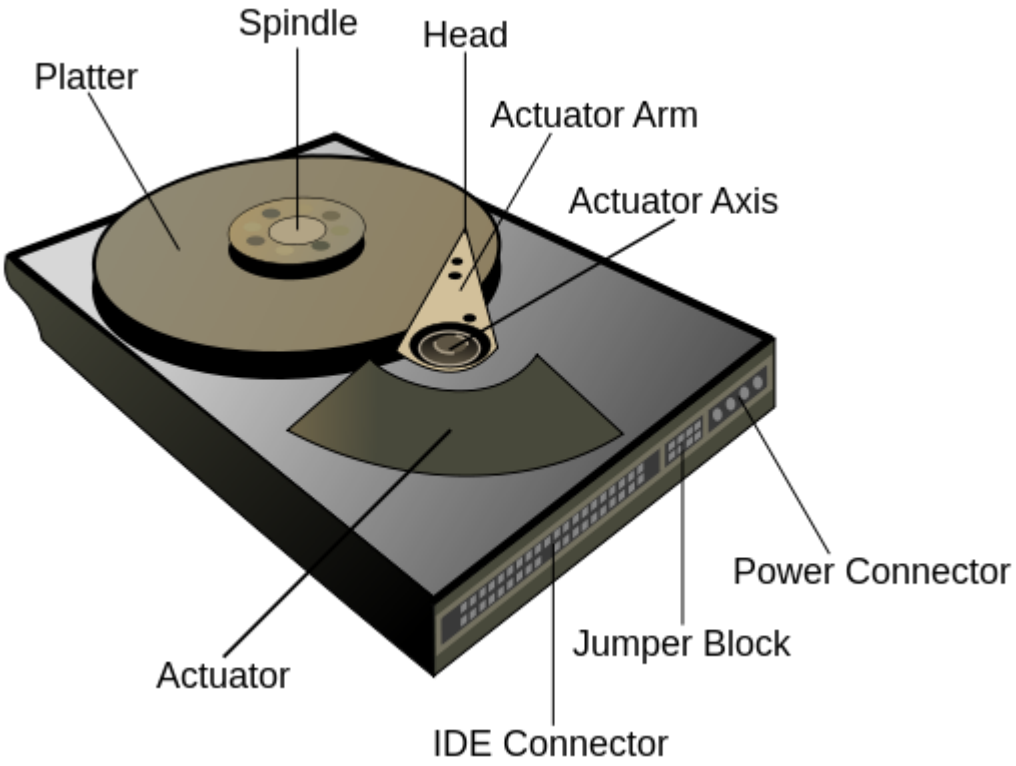
- 扇区的创建
  - 磁盘格式化
  - 逻辑块地址映射到物理块地址
- 扇区的格式
  - 头部：ID，损坏标志位，…
  - 数据区：实际用于存储数据的区域，典型大小为512B
  - 尾部：ECC校验码
- 坏扇区
  - 发现坏扇区 → 先用ECC纠错
  - 如果不能纠错，用备用扇区替代
  - 坏扇区不再使用
- 磁盘容量
  - 格式化损耗20%左右：每个扇区的头部和尾部 + 坏扇区



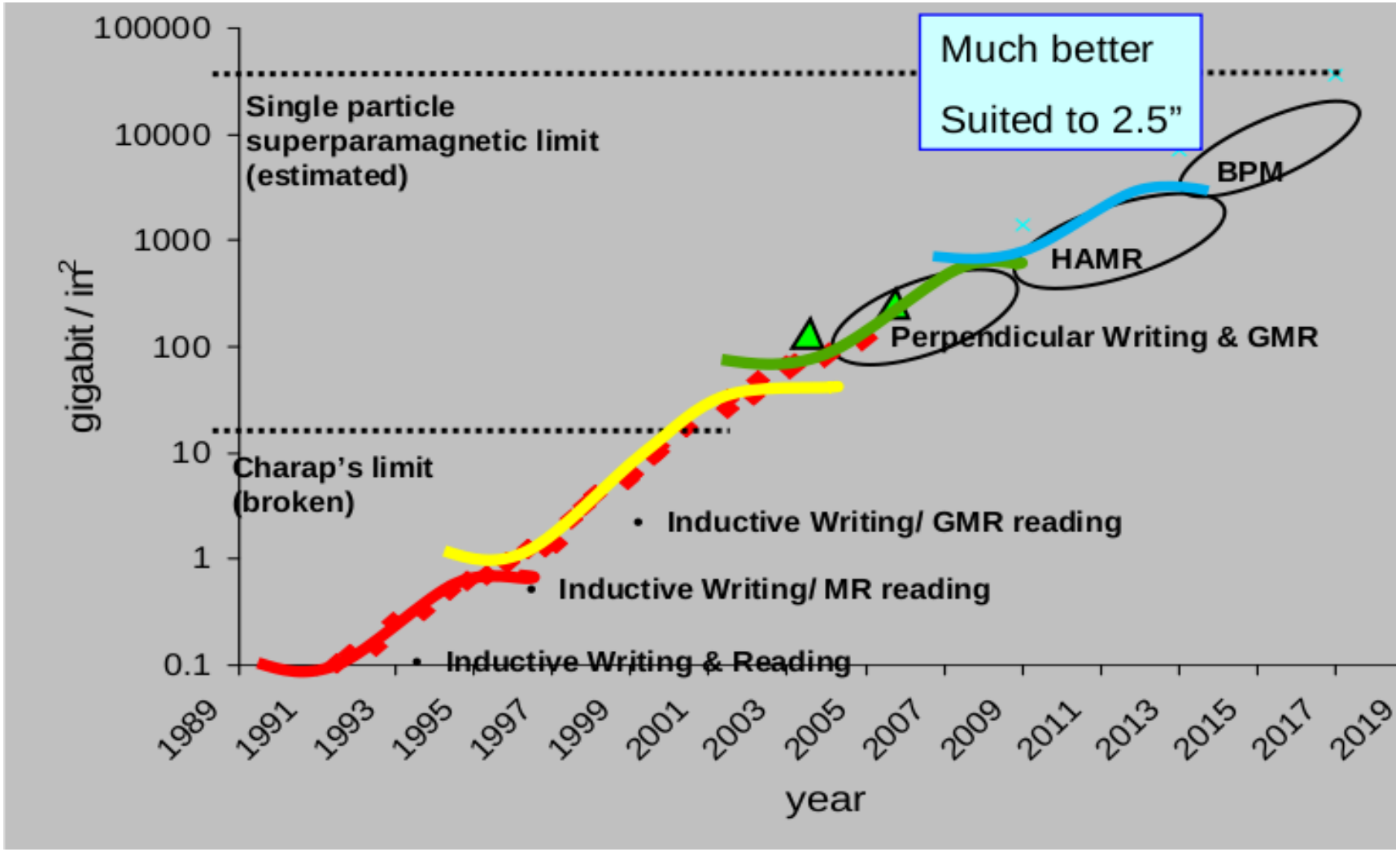
读写操作

读写某个柱面的某个扇区

- 定位柱面，移动磁臂使磁头对准柱面 → 寻道 seek
- 等待扇区旋转到磁头下方 → 旋转 rotation
- 进行数据读写 → 数据传输 transfer



存储密度



Source: Dave Anderson of Seagate - via David Rosenthal - LOCKSS

60年的变化（来源Mark Kryder @SNW 2006）

	IBM RAMAC (1956)	Seagate ST4000NM0035 (2019)	变化
容量	5MB	4TB	800,000 ↑
价格	$1,000/MB	￥0.25/GB	↓
转速	1,200 PRM	7,200 RPM	6 ↑
寻道时间	600 ms	4.16 ms	144 ↓
传输速率	10KB/s	226MB/s	22,600 ↑
功耗	5000 W	6.9W	725 ↓
重量	910 KG	680 g	↓

磁盘性能

- 有效带宽 = 数据量 /耗时
- 耗时
  - 寻道时间（seek time）
    - 把磁头移动到目标柱面的时间
    - 典型：3.5~9.5ms
  - 旋转延迟（rotation delay)
    - 等待目标扇区旋转到磁头下方的时间
    - 典型：7,200 ~ 15,000 RPM
  - 数据传输时间（data transfer time）
    - 典型传输带宽：70~250 MB/sec
- 例子：
  - 假设BW=100MB/s, seek=5ms, rotation=4ms
  - 访问1KB数据的总时间= 5ms + 4ms + (1KB/100MB/s) = 9.01ms
  - 寻道时间和旋转延迟占99.9%
  - 有效带宽 = ?
  - 访问1MB数据的有效带宽呢？

磁盘性能

- 一次传输多少数据才能达到磁盘带宽的90%？
  - 假设磁盘BW=100MB/s, seek=5ms , rotaion=4ms
  - BW×90% = size / (size/BW + rotation + seek)
  - size = BW× (rotation + seek) × 0.9/(1-0.9)
    = 100MB×0.009*9 = 8.1MB

Block Size (Kbytes)	% of Disk Transfer Bandwidth
9Kbytes	1%
100Kbytes	10%
0.9Mbytes	50%
8.1Mbytes	90%

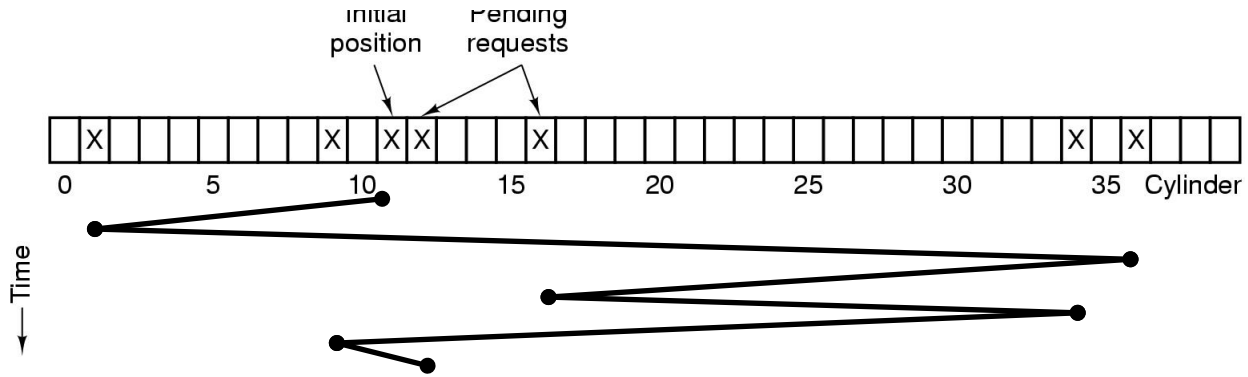
- 对于小粒度的访问，时间主要花在寻道时间和旋转时间上
  - 磁盘的传输带宽被浪费
  - 缓存：每次读写邻近的多个扇区，而不是一个扇区
  - 调度算法：减少寻道开销

磁盘缓存

- 方法
  - 用少量DRAM来缓存最近访问的块
    - 典型大小为 64~256MB（对于一块磁盘）
  - 由控制器管理，OS无法控制
  - 块替换策略：LRU
- 优点
  - 如果访问具有局部性，读性能受益
- 缺点
  - 需要额外的机制来保障写的可靠性

磁盘寻道算法FIFO (FCFS)

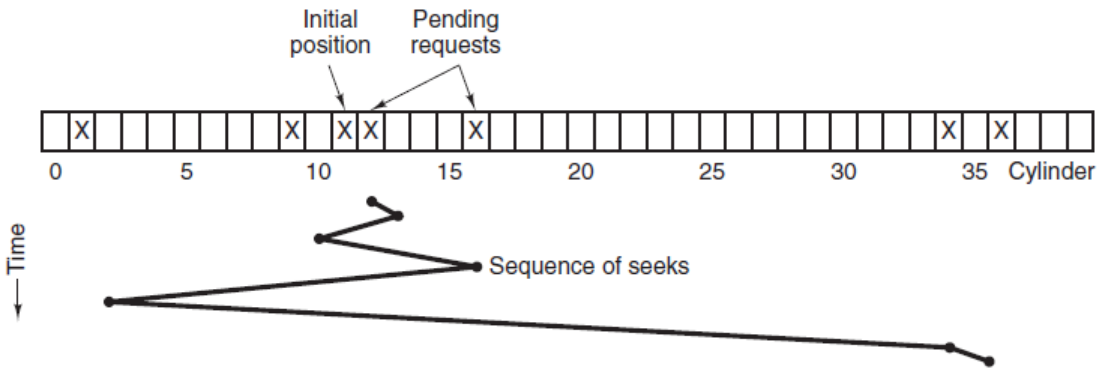
- 例子



  - 请求到达顺序：11→1→36→16→34→9→12（柱面编号）
  - FIFO服务顺序：11→1→36→16→34→9→12
  - FIFO总寻道距离：10+35+20+18+25+3 = 111
- 好处
  - 公平性
  - 服务顺序是应用预期的
- 坏处
  - 请求到来的随机性，经常长距离的寻道
  - 可能发生极端情况：比如横扫整个磁盘

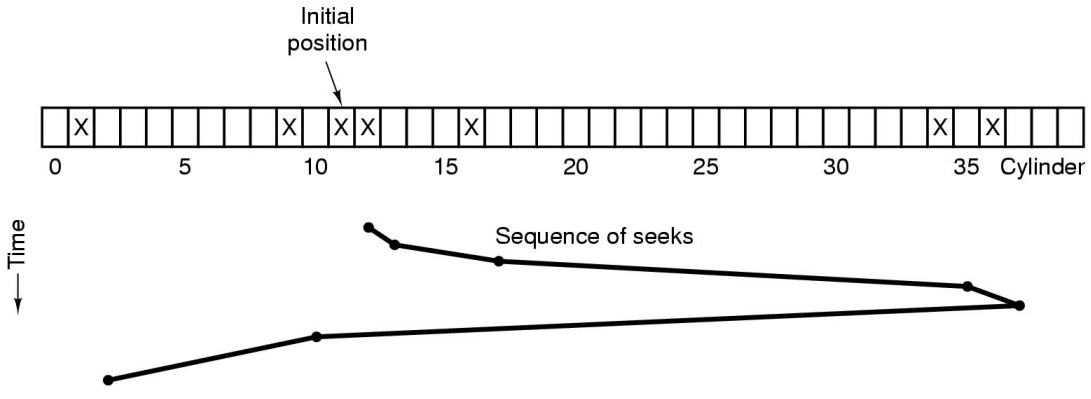
磁盘调度SSF (Shortest Seek First)

- 方法
  - 选择磁头移动距离最短的请求
  - 计入旋转时间
  - 请求到达顺序：11→1→36→16→34→9→12（柱面编号）
  - SSF服务顺序：11→12→9→16→1→34→36
  - SSF总寻道距离：1+3+7+15+33+2 = 61
- 好处
  - 试图减少寻道时间
- 坏处
  - 产生饥饿



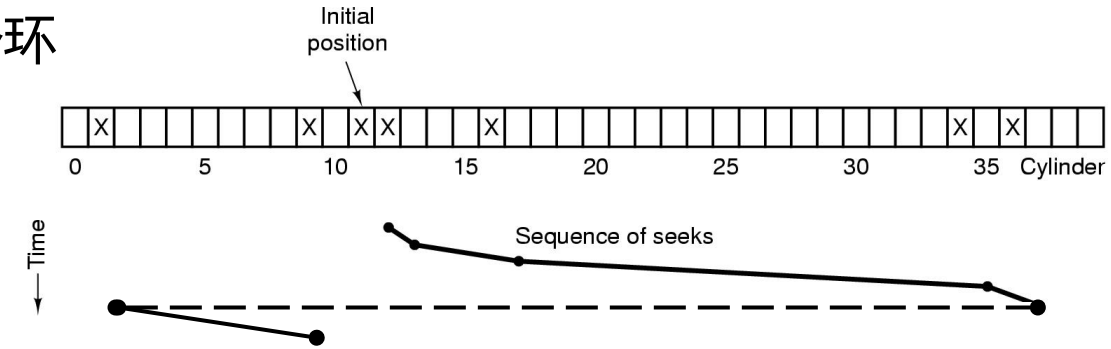
电梯调度 (SCAN/LOOK)

- 方法
  - 磁头按一个方向到另一端，再折回，按反方向回到这端，不断往返
  - 只服务当前移动方向上寻道距离最近的请求
  - LOOK：如果磁盘移动方向上没有请求，就折回
  - 请求到达顺序：11→1→36→16→34→9→12 （柱面编号）
  - SSF服务顺序：11→12→16→34→36→9→1
  - SSF总寻道距离：1+4+18+2+27+8 = 60
- 好处
  - 消除饥饿：请求的服务时间有上限
- 坏处
  - 反方向的请求需等待更长时间



C-SCAN/C-LOOK (Circular SCAN)

- 方法
  - 将SCAN改为折回时不服务请求，立即回到
  - 寻道类似连起来成一个环
  - C-LOOK
- 好处
  - 服务时间趋于一致
- 坏处
  - 折回时不干事



磁盘调度算法

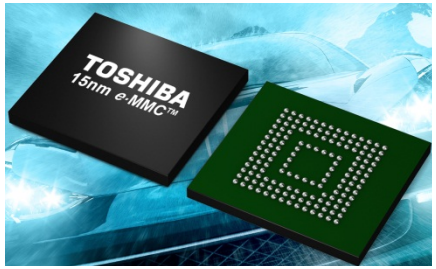
- 调度算法
  - FIFO：实现简单，但寻道时间长
  - SSF：贪心算法，可能造成饥饿现象（距离初始磁头位置较远的请求长期得不到服务）
  - SCAN/LOOK：减少饥饿
  - C-SCAN/C-LOOK：减少SCAN算法返回时的扫描开销
- 磁盘I/O请求缓冲
  - 把请求缓冲在控制器缓冲区
  - 缓冲时间取决于缓冲区大小
- 进一步的优化
  - 既寻道最短，又旋转延迟最短

内容提要

- 磁盘
- 固态硬盘SSD
  - 闪存组织
  - SSD FTL机制
    - 地址映射
    - 磨损均衡
- RAID (磁盘阵列)

闪存（Flash memory）

- 1984: NOR flash, 日本东芝公司, Fujio Masuoka
- 1987: NAND flash,日本东芝公司
- 全电子器件，无机械部件
- 非易失性存储
- 1992：SSD原型，SandDisk



NOR闪存 vs. NAND闪存

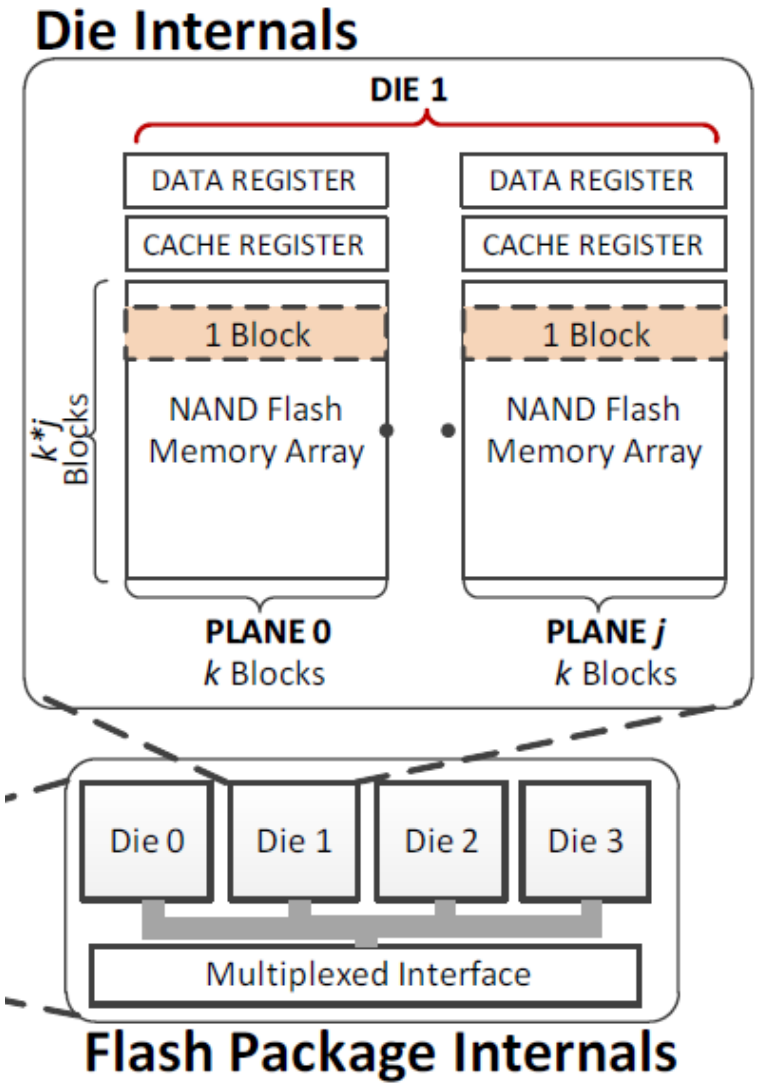
- NOR是字节寻址，NAND是页寻址
- NOR读延迟是比NAND低100x
- NOR的擦除时间比NAND高300x
- NOR用于取代 ROM，存可执行代码
- NAND用于大容量持久化存储设备

闪存

- 信息存储方式
  - SLC: 1 bit/cell, 2个值：0/1
  - MLC: 2 bits/cell, 4个值：00/01/10/11
  - TLC: 3 bits/cell, 8个值
  - QLC: 4 bits/cell, 16个值

NAND闪存组织

- Flash package
  - 多个Die
- Die
  - 多个Plane/Bank
- Plane/Bank
  - 很多块（擦除块）
  - 一些寄存器
- **块** (Block/Erase Block)
  - 很多页
- **页** (Page)
  - 很多cell



闪存组织

- 页
  - 由数据区与OOB区构成
  - 数据区用于存储实际数据
  - OOB区用于记录
    - ECC
    - 状态信息：Erased/Valid/Invalid
    - Logic page number
  - 页大小
    - SLC通常2KB~8KB，TLC通常4KB~16KB
  - 64+页/块
  - 块大小
    - SLC通常128KB、256KB…
    - TLC通常2MB，4MB，8MB…

闪存的操作接口

- Read, Erase, Program (写)
- 读：read a page
  - 读的粒度是页
  - 读很快，读延迟在几十微秒 (us)
  - 读延迟与位置无关，也与上一次读的位置无关 (opp. 磁盘)
- 擦除：erase a block
  - 把整个块写成全1
  - 擦除的粒度是块，必须整块擦除
  - 很慢：擦除时间为几个毫秒 (ms)
  - 需软件把块内有效数据拷贝到其它地方
- 写：program a page
  - 擦除后才能写，因为写只能把1变成0
  - 写的粒度是页
  - 写比读慢，比擦除快，写延迟在几百微秒 (us)

页的状态

Invalid, Erased, Valid

- 初始状态为Invalid
- 读：不改变页的状态
- 擦除：块内所有页的状态变为Erased
- 写
  - 只能写状态为Erased的页
  - 写完成，页状态变为Valid

闪存的性能和可靠性

- 性能
  - 写延迟比读高10倍以上
  - 写延迟波动幅度大
  - 擦除很慢：~ 磁盘定位延迟
  - 延迟随密度增加而增长

Device	Read (µs)	Write (µs)	Erase (ms)	P/E Cycles
SLC	25	200-300	1.5-2	100,000
MLC	50	600-900	~3	10,000
TLC	~75	~900-1350	~4.5	1,000
QLC				200

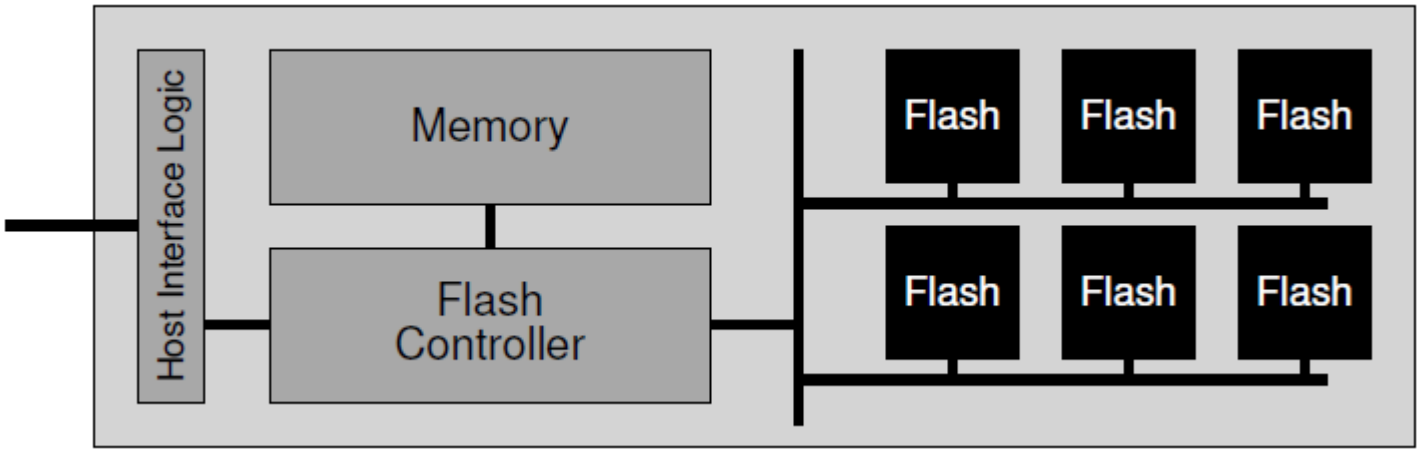
- 可靠性
  - 磨损
    - 擦写次数有上限，随密度增加而减少
  - 干扰
    - 读写一个页，相邻页中一些位的值发生翻转

闪存特性：

1. 读延迟很低：随机读的性能远优于磁盘
2. 写慢：必须先擦除再写，ms级 ~ 磁盘写
3. 磨损：每个块擦写次数有上限

基于闪存的SSD

- 用很多闪存芯片来构成一个持久化存储设备SSD



- 多个闪存芯片：并行I/O，提高 I/O性能
- 与主机的接口：提供标准块设备接口
- 数据缓存和缓冲：SRAM/DRAM
- 闪存控制器FTL：控制逻辑
  - 主机命令转换成闪存命令 (Read/Erase/Program)
  - 逻辑块地址转换成闪存的物理地址 (页/块)
  - 缓存替换
  - 减少写干扰：块内的页顺序写

最简单的FTL：直接映射

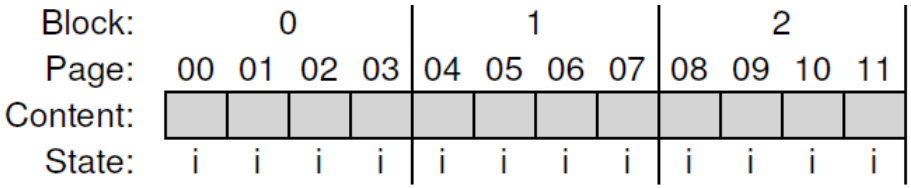
- Direct mapping
  - 逻辑块的第N块直接映射到物理的第N页 (假设逻辑块与物理页都为4KB)
- 读操作容易：读逻辑第k块
  - 读物理第k页
- 写操作麻烦：写逻辑第k块
  - 第k页所在闪存块 (记为B0)
  - 把B0整个块读出来
  - 把B0整个块擦除
  - B0中的旧页和新的第k页：以顺序方式一页一页再写入B0
- 缺陷：写性能极差
  - 每写一个页，要读整个块、擦除整个块、写整个块
  - 写放大
  - 小粒度随机写性能比磁盘还差

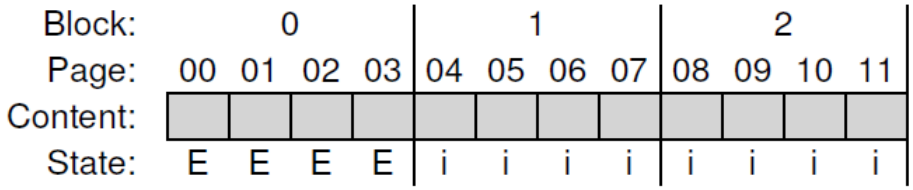
FTL改进: 异地更新

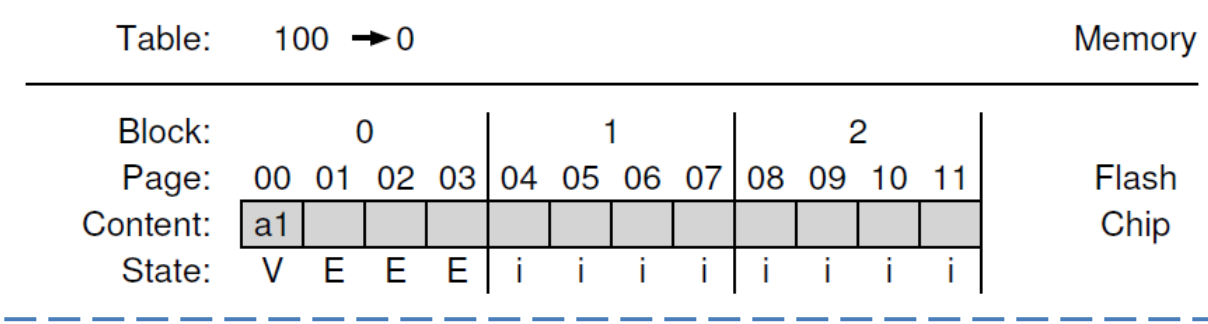
- 核心思想：异地更新 (out-of-place update)
  - 不再执行原地更新
  - 每次写页，写到一个新位置（新的物理页地址）
- 页级映射
  - 映射表： LBN → 物理页地址PPN, 页级映射表
- 写一个逻辑块k
  - 寻找一个空闲页p（例如当前擦除块中下一个空闲页p）
  - 在映射表中记录: 逻辑块 k→物理页 p
- 读一个逻辑块k
  - 查映射表，获得逻辑块k对应的物理页地址p
  - 读物理页p

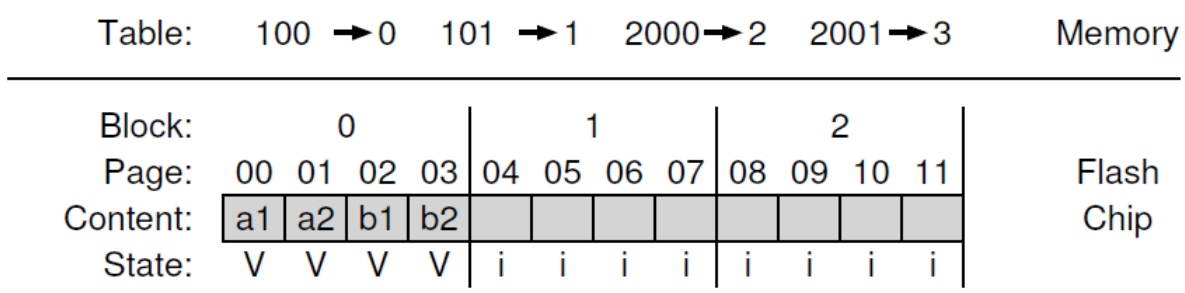
页级映射

例子：依次写逻辑块100, 101, 2000和2001 (a1, a2, b1, b2)







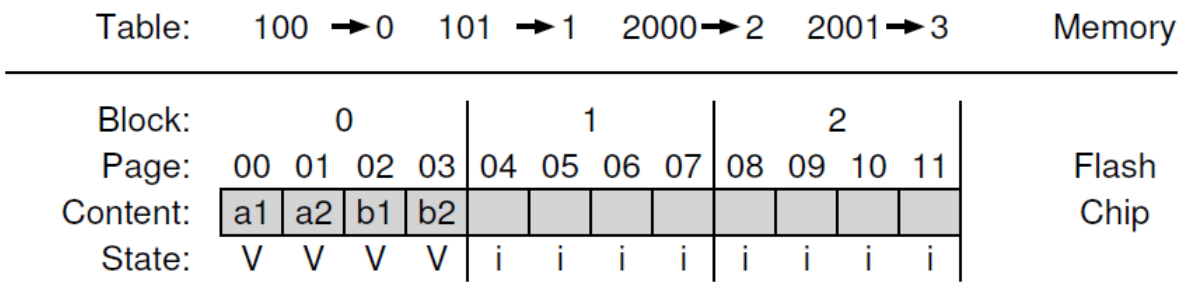


页级映射

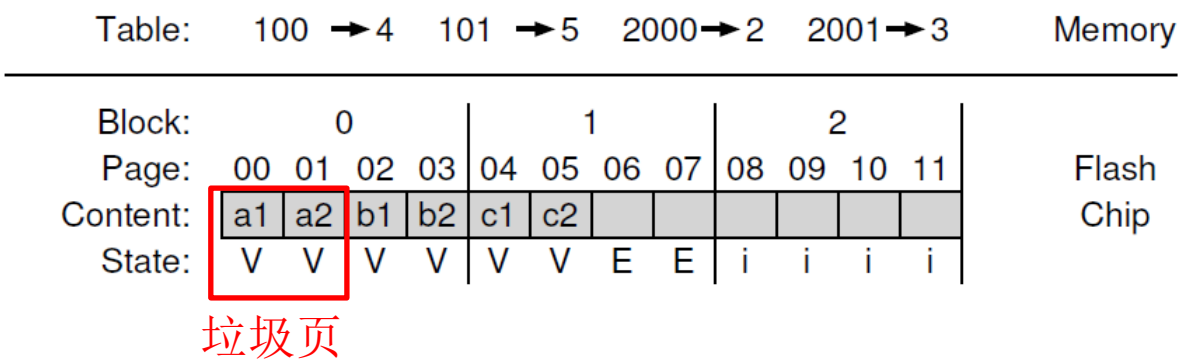
- 页级映射表：LBN → PPN
  - 整个放在内存中
  - 持久化：利用页的OOB区来保存映射表
  - 随着写页而被写到闪存
  - 掉电或重启，扫描OOB区恢复映射表
- 优点
  - 性能好：减少写放大
  - 可靠性好：映射关系被自动写入闪存
- 问题
  - 重写产生垃圾页
    - 每次写到新位置，导致原先页的内容无效
  - 内存开销大
    - 映射表全部放内存
    - 映射表的大小与SSD容量成正比

垃圾页 (garbage/dead page)

例子：依次写逻辑块100, 101, 2000和2001 (a1, a2, b1, b2)

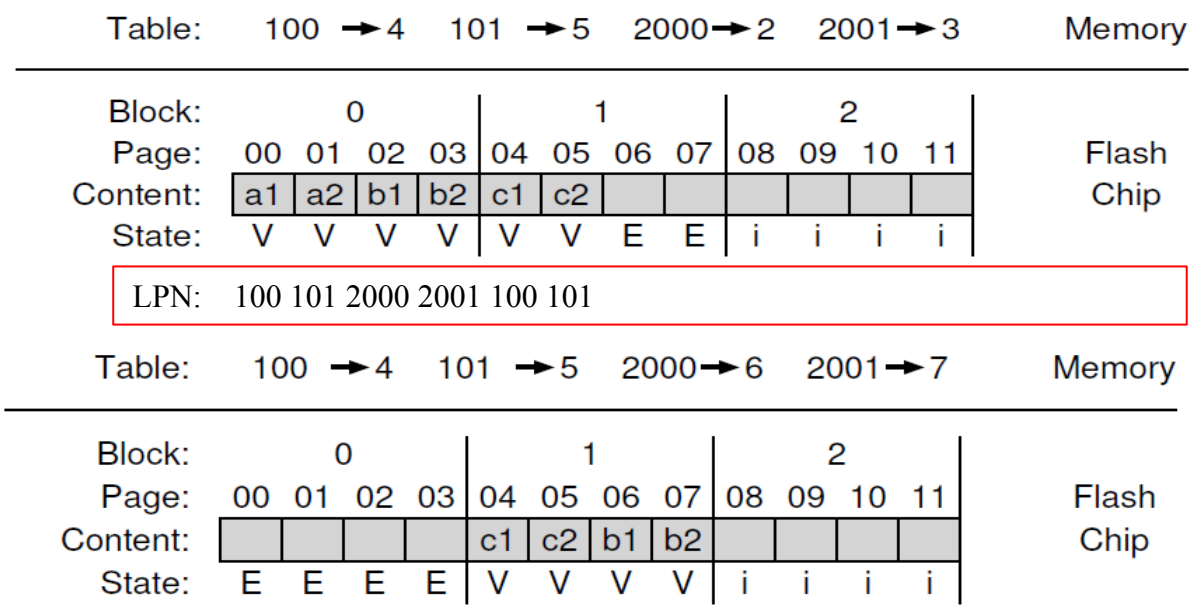


再写逻辑块100和101 (c1, c2)



垃圾回收

- 思想
  - 选择一个含垃圾页的块
  - 把其中的活页 (live page)拷贝到其他块中 (读 & 重写)
  - 回收整个块，并把它擦除
- 如何判断活页？(live page vs. garbage page)
  - 每页记录它对应的逻辑块地址 (OOB区)
  - 查映射表, 如果映射表记录的PPN == 该页，是 live page



垃圾回收后

垃圾回收

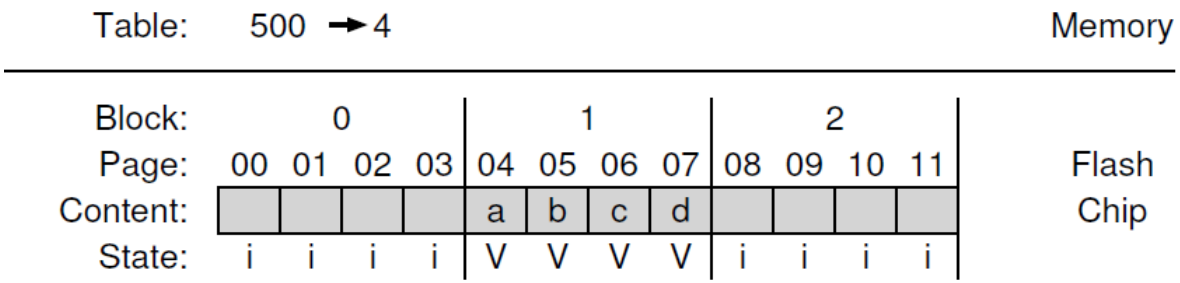
- 问题：开销非常大
  - 活页需拷贝：读 & 写
  - 开销与活页所占的比例成正比

- 解决办法：超配（over-provisioning）
  - 实际物理空间比用户所见空间更大：多15%~45%
    - 例如，用户看到100GB的SSD, 实际上内部是120GB
    - GC时将数据写入over-provisioning space，减少对性能的影响
  - GC一般在SSD后台执行，尽量在设备不忙时执行，受限于空闲页数量

块级映射：block-level mapping

- 块级映射
  - 逻辑地址空间划分为chunk，chunk size=擦除块（物理块）size
  - 映射表：chunk# → 擦除块（物理块）地址PBN
- 读一个逻辑块
  - 逻辑块地址 = chunk# || 偏移
  - 用chunk#查映射表，获得对应的擦除块地址PBN
  - 物理页地址 = PBN || 偏移

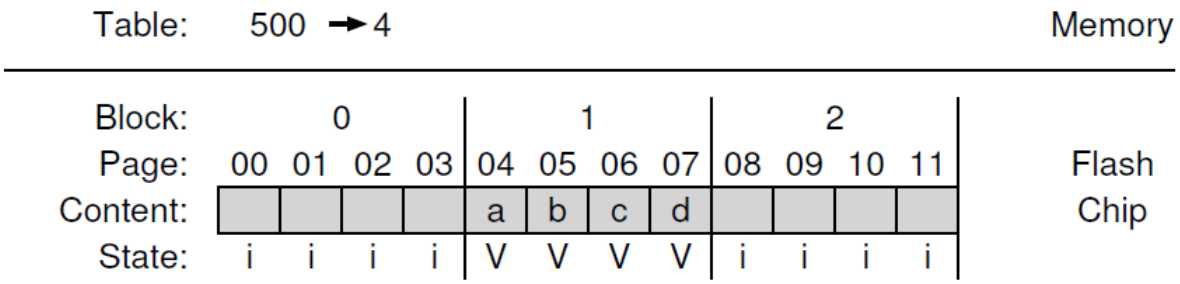
例：依次写逻辑块2000, 2001, 2002, 2003 (a, b, c, d)



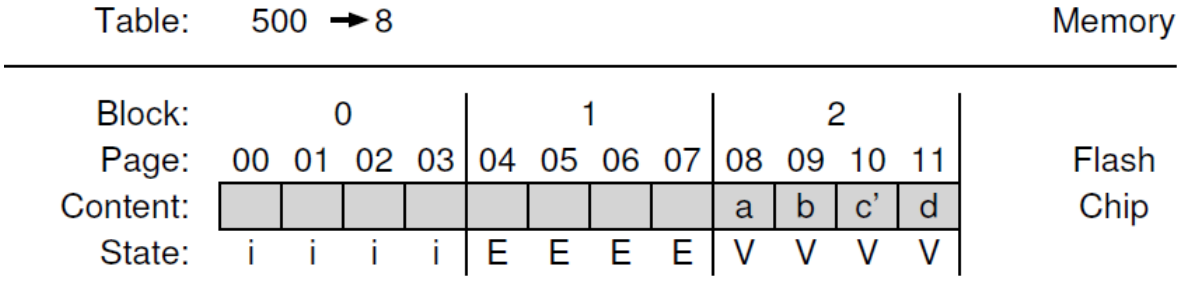
块级映射：block-level mapping

- 问题：小写性能差
  - 写粒度小于擦除块：拷贝活页 (读 & 写), 写放大
  - 小写很常见：擦除块大 (> 256KB)

例：依次写逻辑块2000, 2001, 2002, 2003 (a, b, c, d)



再写逻辑块2002 (c')



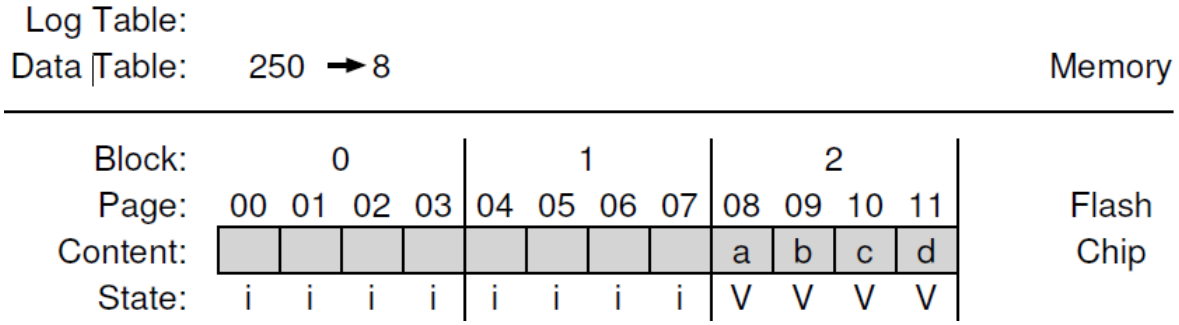
混合映射：hybrid mapping

- 思想
  - 将擦除块（物理块）划分为两类：数据块和日志块
  - 写逻辑块时都写入日志块
  - 数据块采用块级映射，数据映射表
  - 日志块采用页级映射，日志映射表
  - 适当的时候把日志块合并为数据块

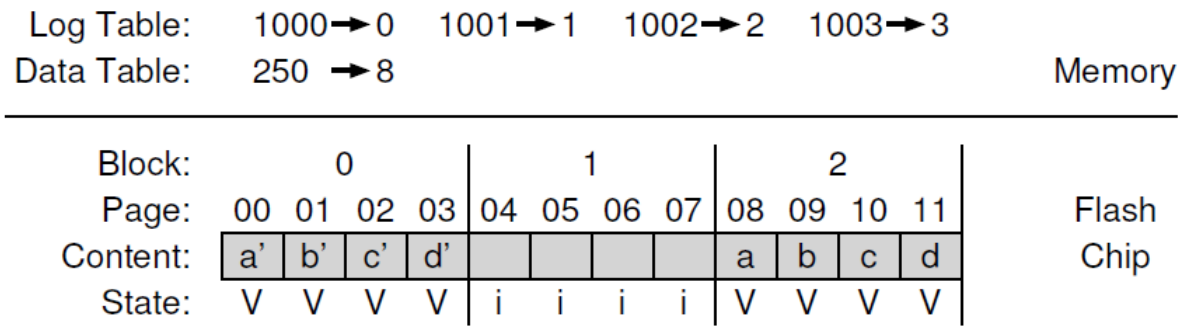
- 读一个逻辑块
  - 先查日志映射表，按页级映射的方法
  - 如果没找到，再查数据映射表，按块级映射的方法

混合映射：hybrid mapping

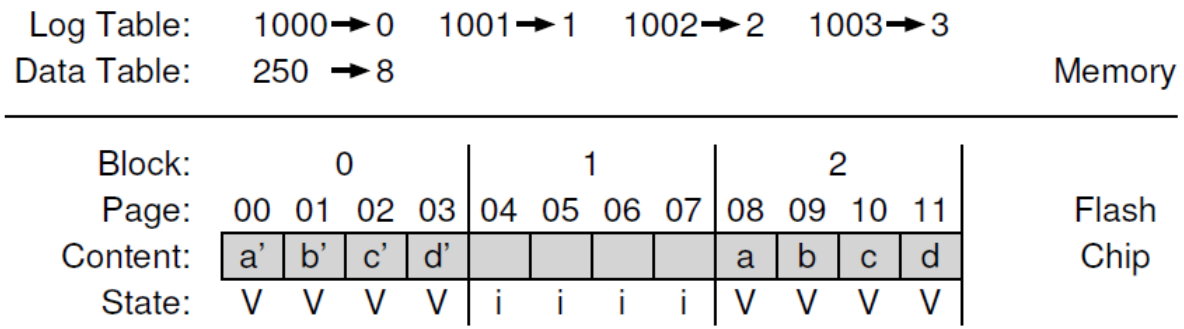
例：当前逻辑块1000, 1001, 1002, 1003中数据分别为a,b,c,d



依次写逻辑块1000, 1001, 1002, 1003 (a', b', c', d')

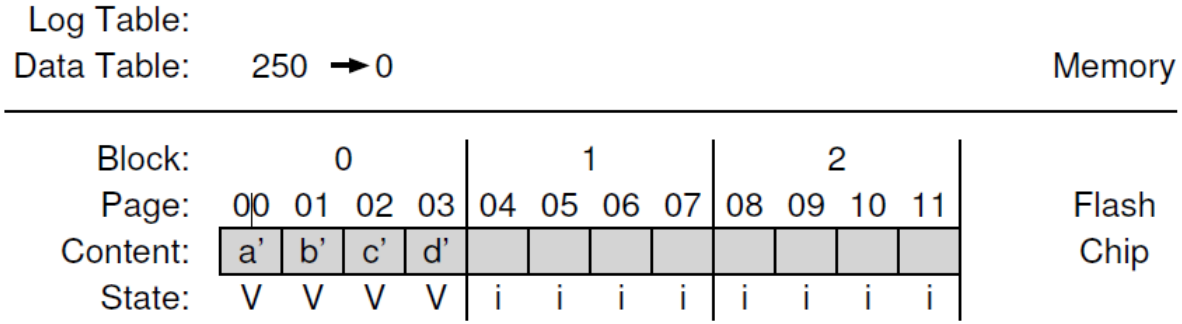


混合映射：hybrid mapping

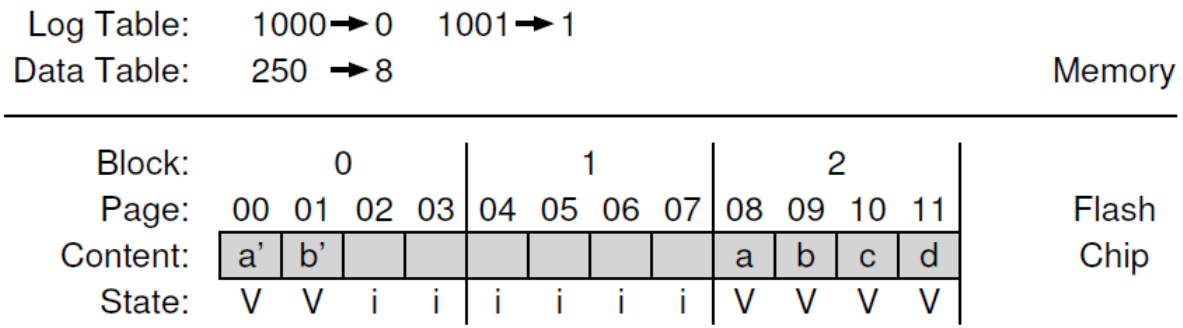


- Switch merge
  - 直接把日志块转成数据块: 前提是整个日志块的页序与chunk一致
  - 把原来的数据块回收擦除
  - 优点：开销低，只修改映射表信息，无数据拷贝

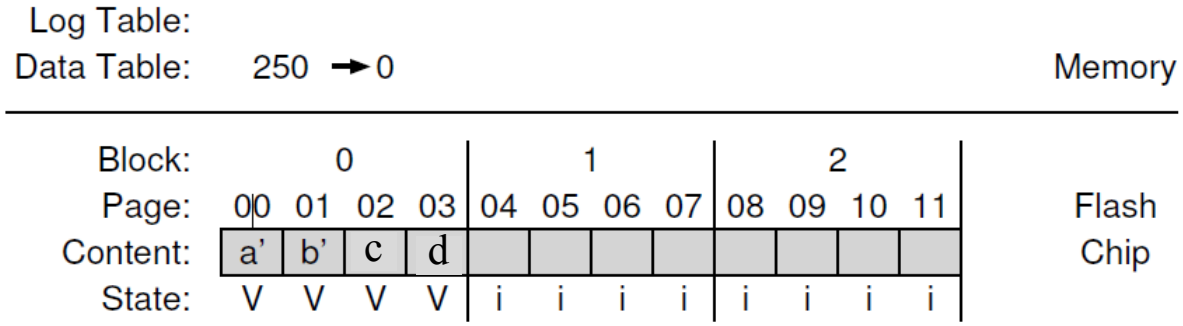
合并后



混合映射：hybrid mapping

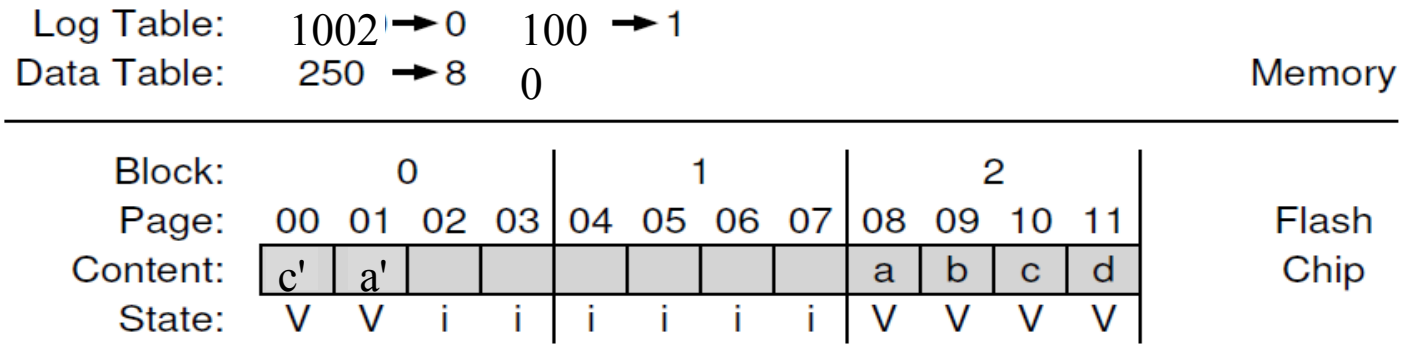


- Partial merge
  - 把数据块中有效页拷贝到日志块：日志块中页序与chunk一致
  - 把日志块转成数据块，把原来的数据块回收擦除
  - 有数据拷贝开销

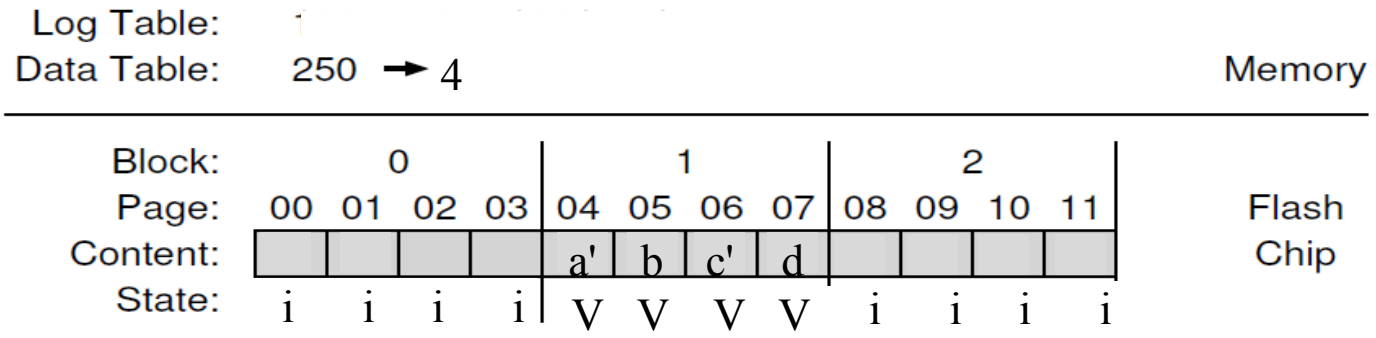


合并后

混合映射：hybrid mapping



- Full merge
  - 分配一个新的日志块，从数据块和日志块分别拷贝有效页到新日志块
  - 把新日志块转成数据块
  - 把原来的数据块和日志块都回收擦除
  - 开销很大：需要拷贝整个物理块的数据（读 & 写）



磨损均衡

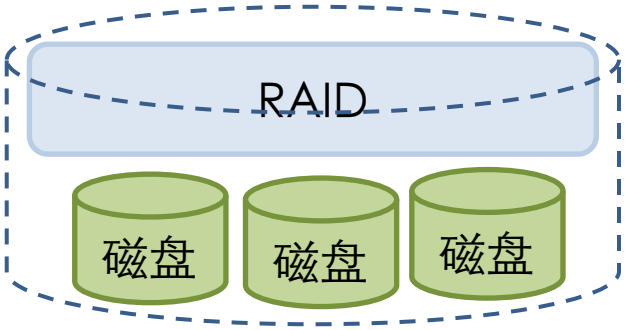
- 目标
  - 让所有块被擦除的次数近似
- 动态磨损均衡
  - 每次写时，选择擦除次数较少或最少的空闲块
  - 局限性：不同数据的修改频率不同
    - 例子：只写一次的数据（static data），很少写的数据（cold data)
- 静态磨损均衡
  - 动态磨损均衡不考虑不会被回收的物理块，例如长时间不被修改的逻辑块（写冷块）
  - 不再被写，不再有磨损
  - 解决办法：FTL定期重写冷块

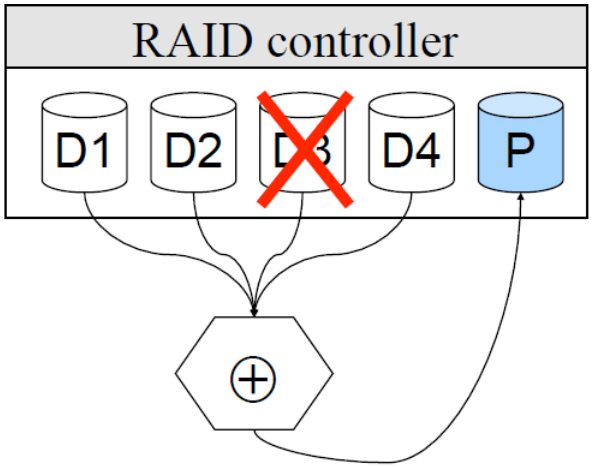
内容提要

- 磁盘
- 固态硬盘SSD
- RAID (磁盘阵列)

RAID (Redundant Array of Independent Disks)

- 主要思想
  - 由多个磁盘构成一个存储设备
- 好处
  - 提高性能：多个磁盘并行工作
  - 增加容量：聚合多个磁盘的空间
  - 提高可靠性：数据冗余，
    磁盘损坏，数据不损坏
- 坏处
  - 成本
  - 控制器变得复杂
- 牵涉的问题
  - 块映射：逻辑块LBN → <磁盘#, 块#>
  - 冗余机制

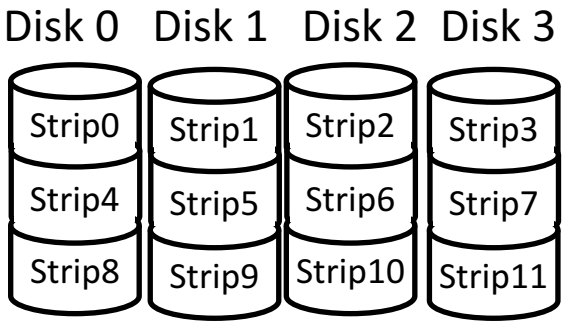




$P = D1 \oplus D2 \oplus D3 \oplus D4$

$D3 = D1 \oplus D2 \oplus P \oplus D4$

RAID-0



Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

RAID Level 0

- 以条带（strip）为粒度映射到N块磁盘（轮转方式）
- 1 strip = N个块
- 无冗余

容量

- N × 单个磁盘容量

可靠性

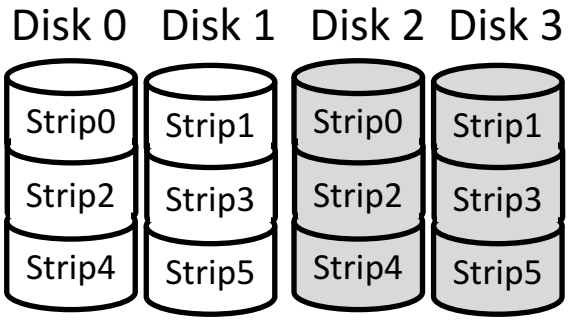
- （单个磁盘可靠性）^{N}

性能

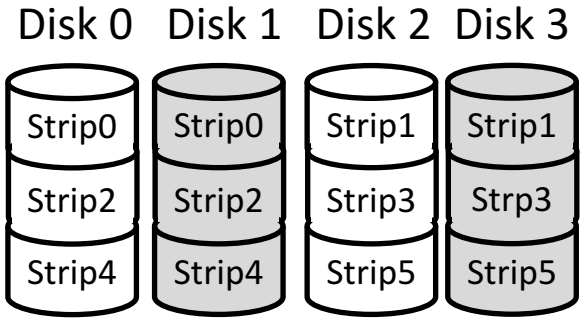
- 带宽 = N × 单个磁盘带宽
- 延迟 = 单个磁盘的延迟

RAID-1

RAID-01 / RAID-0+1



RAID-10 / RAID-1+0



Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

RAID Level 1

- 镜像
- 镜像级别R：数据存R份
- 通常与RAID-0结合使用
  RAID-01或RAID-10

容量

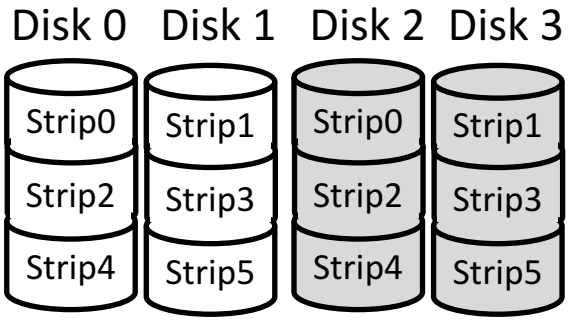
- （N ×单个磁盘容量）/ R

可靠性（R=2)

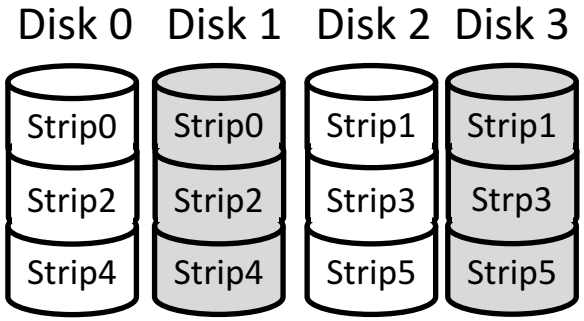
- 容忍任何一个磁盘坏
- 特殊情况下可容忍N/2个磁盘坏

RAID-1

RAID-01 / RAID-0+1



RAID-10 / RAID-1+0



Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

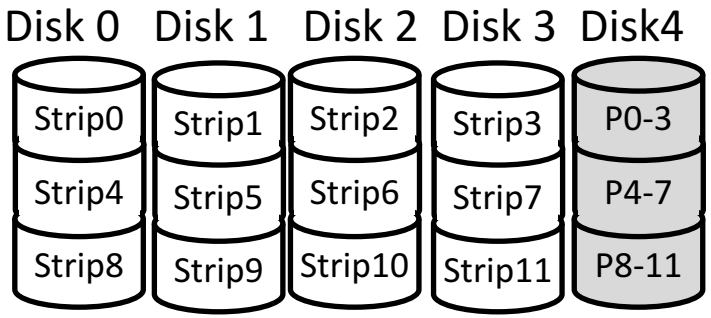
带宽

- 写带宽
  - (N × 单个磁盘写带宽) / R
- 读带宽
  - N × 单个磁盘读带宽

延迟

- ~单个磁盘的延迟

RAID-4



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

校验块计算使用XOR

Block0	Block1	Block2	Block3	Parity
00	10	11	10	11
10	01	00	01	10

RAID Level 4

- 条带化 + 1个校验块
- 所有校验块在同一块磁盘上(校验盘)
- 缺点：校验盘为写性能瓶颈，易坏

每次写都更新校验块

方法一：读所有数据盘

1、并行读所有磁盘的对应块

2、计算新校验块

3、并行写新块和新校验块

方法二：读一个数据盘和校验盘

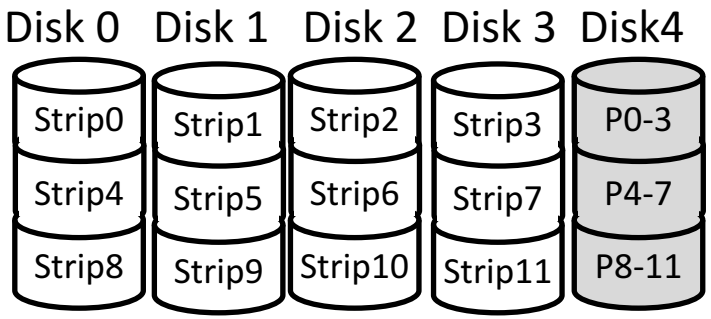
1、并行读旧块和旧校验块

2、计算新校验块

$Pnew = (Bold \oplus Bnew) \oplus Pold$

3、并行写新块和新校验块

RAID-4



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

容量

- (N-1) ×单个磁盘容量

可靠性

- 容忍任何一块磁盘坏
- 用XOR重构坏盘数据

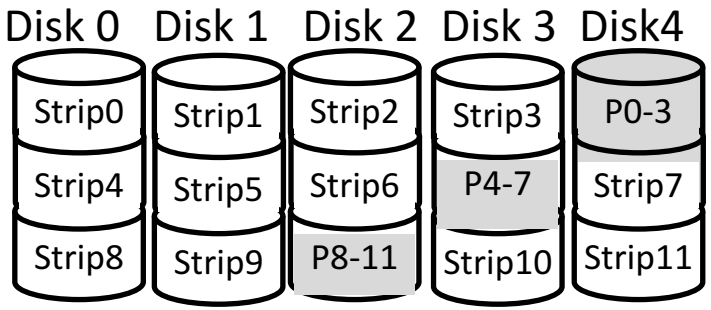
延迟

- 读延迟等于单个磁盘的延迟
- 写延迟约等于2倍单个磁盘延迟

带宽

- 读带宽=(N-1) × 单个磁盘带宽
- 校验盘为写瓶颈，所有校验块串行写

RAID-5



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

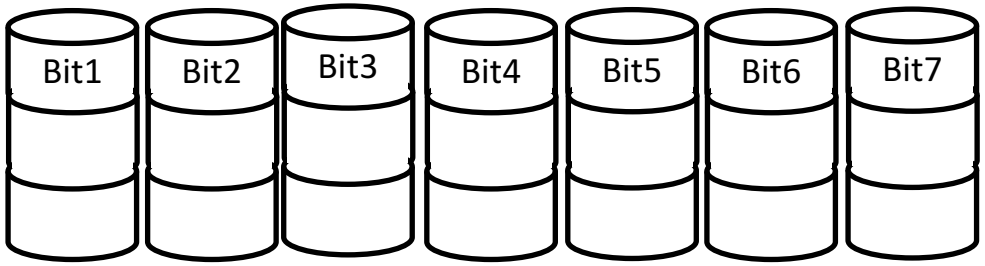
RAID Level 5

- 条带粒度映射 + 1个校验块
- 校验块分散在不同磁盘上
- Rebuild：复杂 & 速度慢

写带宽

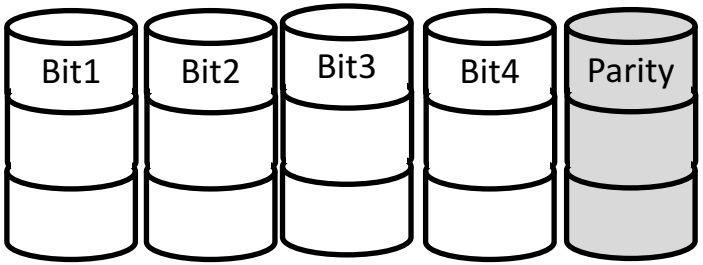
- 写并行：校验块并行写
- 写带宽=(N × 单个磁盘带宽) / 4

其它RAID级别



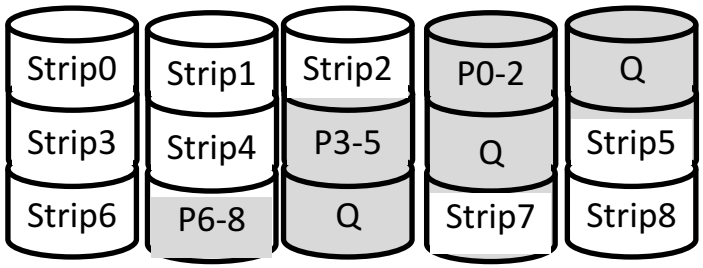
RAID Level 2

- 按位为粒度映射 + ECC
- 每4位 + 3位海明码
- 所有磁盘同步读写：寻道+旋转



RAID Level 3:

- 按位为粒度映射 + Parity位
- 已知坏磁盘时，可纠错

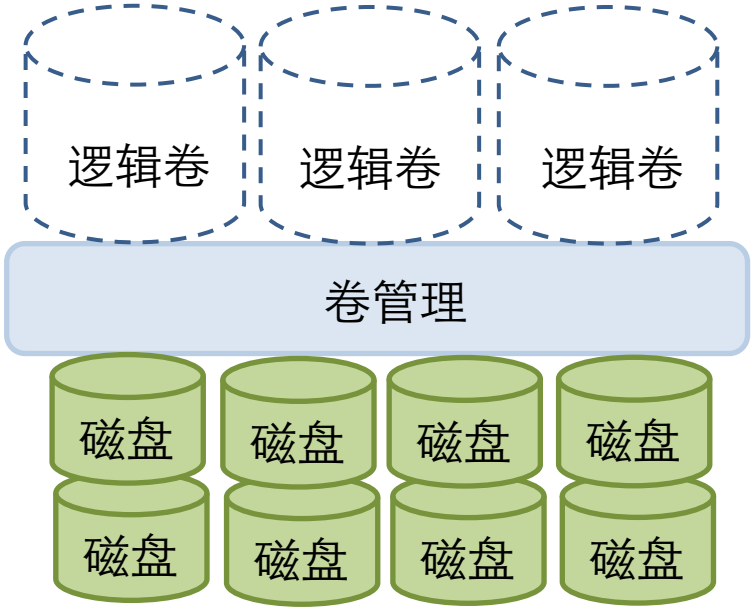


RAID Level 6

- 能容忍两块磁盘同时坏
- 条带化 + 2个校验块

卷管理（Volume manager）

- 虚拟块设备
  - 在多个磁盘上创建一个或多个逻辑卷
  - 逻辑卷：一个虚拟块设备
  - 采用RAID技术将逻辑卷的块地址映射到物理设备
- 好处
  - 提供虚拟的容量和性能
  - 容错
- 实现
  - OS内核的逻辑卷管理：
    - Windows, MacOS, Linux等
  - 存储设备控制器（存储系统）
    - EMC, Hitachi, HP, IBM, NetApp
    - 接口：PCIe, iSCSI, FC, …



总结

- 磁盘
  - 机械设备，内部很复杂，读写性能受限
  - 进行大块读写可以获得高带宽
  - 需要磁盘调度来减少寻道开销
- 闪存的特性
  - 读延迟很低：读性能远优于磁盘
  - 写慢，擦除慢（ms级）：必须先擦除再写
  - 磨损：每个块擦写次数有上限
- SSD FTL主要功能

总结

- RAID提高可靠性和I/O带宽
  - RAID 0
  - RAID 10，RAID 01
  - RAID 5
- 卷管理提供虚拟块设备
  - 通常基于RAID管理底层物理块设备