OPERATING SYSTEMS

Mass Storage Systems

Operating Systems

郑贵锋 博士
中山大学计算机学院
zhenggf@mail.sysu.edu.cn
https://gitee.com/code_sysu

中山大學 SUN YAT-SEN UNIVERSITY

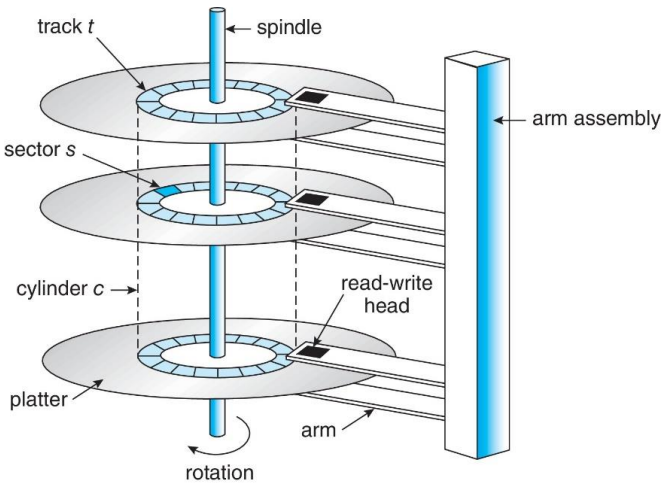
- 目录
  - 概述
    - 磁盘结构
    - 磁盘连接
  - 磁盘调度
  - 磁盘管理
    - 交换空间管理
    - RAID结构
  - 稳定的存储实现

大容量存储结构概述

- 磁盘为现代计算机提供了大量的辅助存储
  - 驱动器以每秒60到250次的速度旋转。
  - 传输速率是驱动器和计算机之间的数据传输速率。
  - 定位时间（随机访问时间）是将磁臂移动到所需圆柱的时间（寻道时间）和所需扇区旋转到磁头下的时间（旋转延迟）
  - 磁头碰撞是由于某些磁头与磁盘表面接触造成的。正常情况下它不能被修复。
- 驱动器通过I/O总线连接到计算机
  - 总线有很多种，包括EIDE、ATA、SATA、USB、光纤通道、SCSI、SAS、Firewire.
  - 计算机中的主机控制器使用总线与内置在驱动器或存储阵列中的磁盘控制器进行通信。

硬盘

- 移动磁头式磁盘构造



硬盘

- 盘片Platter尺寸范围为0.85至14英寸（历史上）
  - 通常为3.5英寸、2.5英寸和1.8英寸
- 每个驱动器的容量范围从420M到3TB或更大。
- 性能
  - 传输速率 – 理论值 6 Gb/sec
  - 有效传输速率 – 实际值 1Gb/秒
  - 寻道时间通常为3毫秒到12毫秒 – 通常桌面驱动器是9毫秒
  - 平均寻道时间基于1/3轨道测量或计算
  - 基于主轴速度的最大延迟
    - 以秒为单位的旋转周期
    - 1/(RPM/60) = 60/RPM
  - 平均延迟 = ½延迟

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

硬盘

- 硬盘性能
  - 访问一次磁盘需要的时间

    访问延迟 = 平均访问时间

    = 平均寻道时间 + 平均延迟
  - 对于慢速磁盘(4200rpm)，9ms + 5.56ms = 14.56ms
  - 对于速度最快的磁盘(15000rpm)，3ms + 2ms = 5ms
  - SSD(非磁性，无活动部件)可能下降至0.05毫秒
  - 完成一次I/O的时间的平均值

    平均I/O时间 = 平均访问时间

    + (传输数据总量 / 传输速率)

    + 控制器开销

- **硬盘**
  - 硬盘性能
    - 实例
      - 在7200rpm磁盘上传输4KB块，平均寻道时间为5ms，传输速率为1Gb/s，控制器开销为0.1ms。

        平均寻道时间 = 5ms

        平均延迟 = 60000/7200/2 = 4.17ms

        平均访问时间 = 5ms + 4.17ms = 9.17ms

        控制器开销 = 0.1ms

        传输数据总量 = 4KB = 32Kb

        传输速率 = 1Gb/s = 2^{20} Kb/s

        传输时间 = 32Kb / 2^{20} Kb/s =3.1×10^{-5} s = 0.031ms

        平均I/O时间 = 9.17 + 0.031 + 0.1 = 9.301ms

        （访问并传输一个4KB页需要的平均时间）

- **硬盘**
  - 第一个商用磁盘驱动器
    - 1956年，IBM RAMDAC计算机包括的IBM Model 350型磁盘存储系统
      - 5M(7位)字符
      - 50 × 24英寸盘片
      - 存取时间 <= 1秒

固态磁盘

- 固态磁盘(SSD)是非易失性存储器，像硬盘一样使用。
  - 许多技术变化
- 可以比硬盘更可靠
- 每MB更贵
- 也许他们的寿命更短
- 容量较小
- 总线可能太慢了
  - 直接连接到PCI
- 无运动部件，因此无寻道时间或旋转延迟
  - 快得多

磁带

- 磁带是早期的辅助存储介质
  - 从开放式卷轴演变为盒式磁带
- 相对永久，并保存大容量数据
- 访问时间慢
- 随机访问比磁盘慢1000倍
- 主要用于备份、存储不常用数据、系统间传输介质
- 保存在卷轴中，并缠绕或倒绕读写磁头
- 一旦数据在磁头下，传输速率可与磁盘媲美
  - 140MB/秒及以上
- 典型存储容量200GB到1.5TB
- 常见的技术有LTO-{3,4,5}和T10000

硬盘结构

- 硬盘驱动器以逻辑块(block)的大型一维数组方式寻址，其中逻辑块是最小的传输单元。
  - 低级格式化在物理介质上创建逻辑块。
- 逻辑块的一维数组按顺序映射到磁盘的扇区。
  - 扇区0是最外层柱面上第一个磁道的第一个扇区。
  - 映射依次通过该磁道，然后是该柱面中的(下面盘片的)其余磁道，然后从最外层到最内层通过其余柱面
  - 逻辑到物理地址的映射应该很容易
    - 除了坏扇区
    - 通过恒定角速度访问每条磁道上的非恒定扇区数（内磁道扇区数少）

磁盘连接

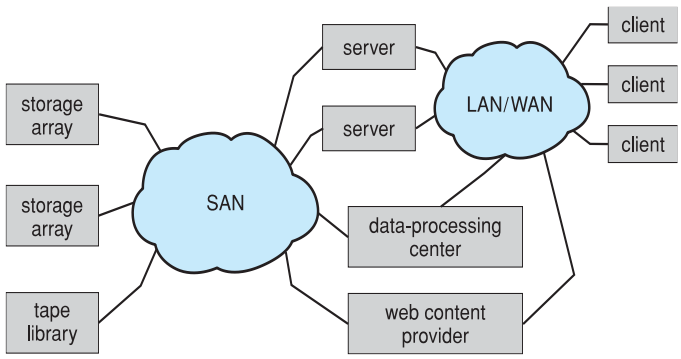
- 连接主机的存储通过I/O端口访问
  - 与I/O总线对话
- SCSI本身是一条总线，一根电缆上最多有16个设备，SCSI启动器请求操作，SCSI目标执行任务。
  - 每个目标最多可以有8个逻辑单元（连接设备控制器的磁盘）
- 光纤通道(FC)是一种高速串行结构
  - 可以在24位地址空间切换通道 – 存储区域网络(SAN)的基础，其中许多主机连接到许多存储单元。
- I/O定向到总线ID
  - 设备ID，逻辑单元(LUN).

磁盘连接

- 存储阵列
  - 可以只连接磁盘或磁盘阵列
  - 存储阵列具有控制器，为连接的主机提供功能。
    - 用于将主机连接到阵列的端口
    - 内存、控制软件（有时为NVRAM等）
    - 几到数千个磁盘
    - RAID、热备盘、热插拔
    - 共享存储
      - 更有效率
    - 某些文件系统中的功能
      - 快照、克隆、精简资源调配、复制、重复数据消除等

存储区域网

- 存储区域网络(SAN)在大型存储环境中很常见。
- 多台主机连接到多个存储阵列 – 灵活

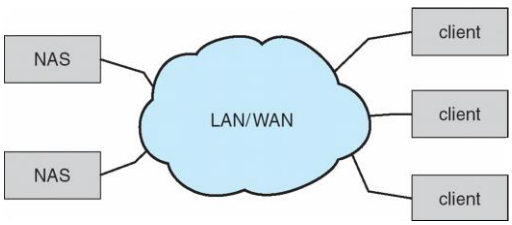


存储区域网

- SAN是一个或多个存储阵列。
  - 连接到一个或多个光纤通道交换机
- 主机也连接到交换机。
- 通过LUN掩蔽(masking)将存储从特定阵列提供给特定服务器。
- 易于添加或删除存储，添加新主机和分配存储
  - 使用低延迟光纤通道结构
- 为什么有分离的存储网络和通信网络？
  - 考虑iSCSI，FCOE

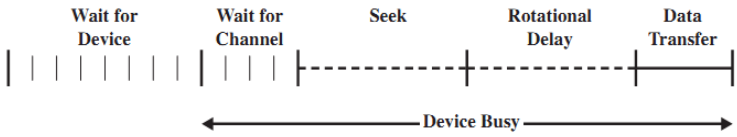
网络连接存储

- 网络连接存储(NAS)是通过网络而不是通过本地连接（如总线）提供的存储。
  - 远程连接到文件系统
- NFS和CIFS是常见的协议。
- 通过主机和存储之间的远程进程调用(RPC)在IP网络上通过TCP或UDP实现
- iSCSI协议使用IP网络承载SCSI协议
  - 远程连接到设备（块）



HDD Scheduling

- **硬盘调度**
  - 操作系统负责有效地使用硬件
    - 对于磁盘驱动器，这意味着具有快速的访问时间和磁盘带宽。
  - 访问时间的两部分
    - 寻道时间是磁盘将磁头移动到包含所需扇区的柱面的时间。
    - 旋转延迟是等待磁盘将所需扇区旋转到磁头的额外时间。
    - 假设寻道时间 ≈ 寻道距离
  - 磁盘带宽是传输的总字节数，除以第一次服务请求和最后一次传输完成之间的总时间。



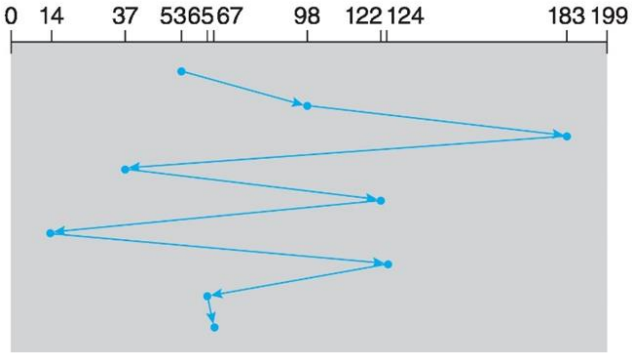
HDD Scheduling

- **硬盘调度**
  - 磁盘I/O请求的来源很多
    - 操作系统
    - 系统进程
    - 用户进程
  - I/O请求包括输入或输出模式、磁盘地址、内存地址、要传输的扇区数。
  - 操作系统维护每个磁盘或设备的请求队列。
  - 空闲磁盘可以立即处理I/O请求，繁忙磁盘意味着请求必须排队等待服务。
    - 驱动器控制器具有较小的缓冲区，可以管理I/O请求队列
    - 只有当队列存在时，优化算法才有意义。

HDD Scheduling

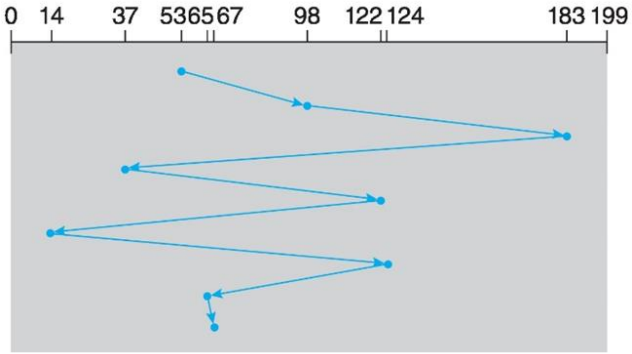
- **先到先得（FCFS）**
  - 如图显示了200个柱面的全部磁头移动
    - 假设初始磁头位置在柱面53中。包含柱面请求的队列为

      98, 183, 37, 122, 14, 124, 65, 67



HDD Scheduling

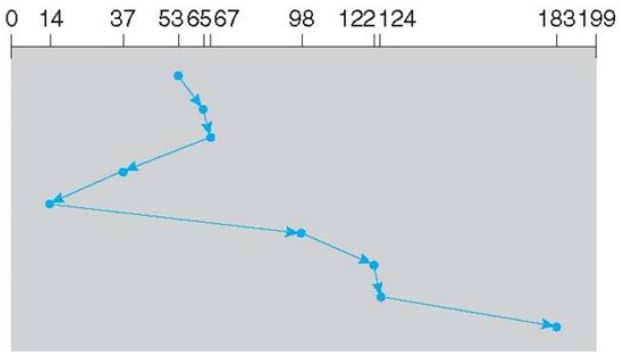
- **先到先得（FCFS）**
  - 对FCFS的评论
    - 按顺序处理I/O请求
    - 对所有进程公平
    - 如果有许多进程/请求，则在性能上接近随机调度
    - 受全局之字形效应的影响



最短寻道时间优先(SSTF)

- 如图显示了200个柱面的全部磁头移动。
  - 假设初始磁头位置在柱面53中。包含柱面请求的队列为

    98, 183, 37, 122, 14, 124, 65, 67



最短寻道时间优先(SSTF)

- 对SSTF的评论
  - SSTF从当前磁头位置选择具有最小寻道时间的请求。
    - 也称为最短寻道距离优先(SSDF) — 计算距离更容易
    - 是一种最近邻算法
  - 对中间磁道的请求有利。
  - SSTF调度是SJF（最短作业优先）调度的一种形式；可能会导致饥饿（某些请求无法满足）

电梯算法

- 电梯算法基于普通电梯调度原理。
- 电梯算法的四种组合：
  - 双向或仅单向服务。
  - 运行直到最后一个柱面或直到最后一个I/O请求。

运行直到 / 方向	最后一个柱面	最后一个请求
双向服务	SCAN	LOOK
仅单向服务	C-SCAN	C-LOOK

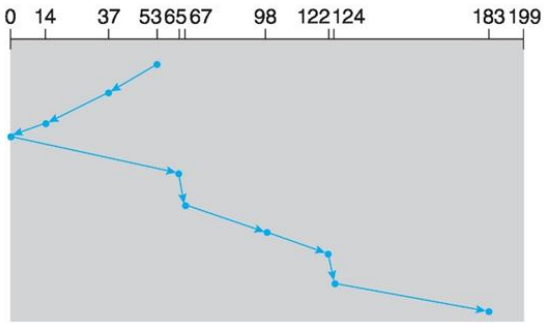
电梯算法

- 扫描调度SCAN
  - 磁臂从磁盘的一端开始向另一端移动，在到达每个柱面时为该柱面的请求提供服务，直到到达磁盘的另一端，然后磁头反向移动，继续服务。
    - 磁臂的行为就像建筑物中的电梯一样，首先为所有向上的请求提供服务，然后反过来为向下的请求提供服务。
  - 它往往更多地停留在末端，因此对极端的要求更公平。
  - 请注意，在柱面请求的均匀分布中，当磁头到达一端并反转方向时，相对较少的请求出现在磁头正前方，因为这些柱面最近已经服务过。请求密度最大的是磁盘的另一端，这些请求等待的时间也最长。

HDD Scheduling

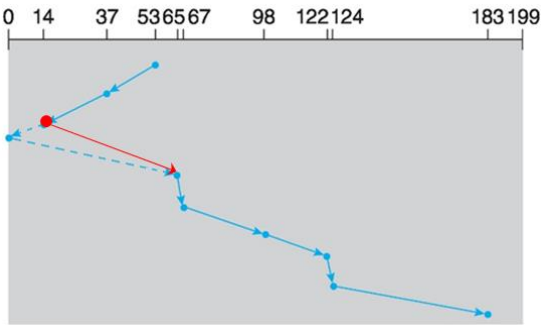
- **电梯算法**
  - 扫描调度SCAN
    - 如图显示了200个柱面的全部磁头移动
      - 假设盘臂向0移动且初始磁头位置在柱面53中。包含柱面请求的队列为

        98, 183, 37, 122, 14, 124, 65, 67



HDD Scheduling

- **电梯算法**
  - 查看调度LOOK
    - 磁臂从磁盘上的第一个I/O请求开始，向某一端移动，在到达每个柱面时为请求提供服务，直到到达该方向上的最后一个I/O请求，此时磁头反向移动，继续服务。
      - 它向两个方向移动。
      - 更倾向于满足中间柱面的请求。



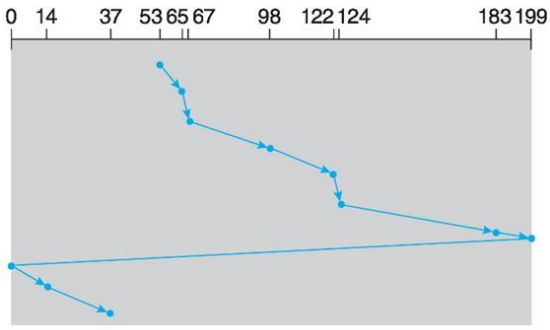
HDD Scheduling

- **电梯算法**
  - 循环扫描调度
    - 循环扫描(C-SCAN)调度是扫描的一种变体，旨在提供更均匀的等待时间。与SCAN一样，C-SCAN将磁头从磁盘的一端移动到另一端，同时为请求提供服务。但是，当磁头到达另一端时，它会立即返回到磁盘的开头，而不处理回程中的请求。
      - 它将所有柱面视为从最后一个柱面到第一个柱面的一个循环列表。

HDD Scheduling

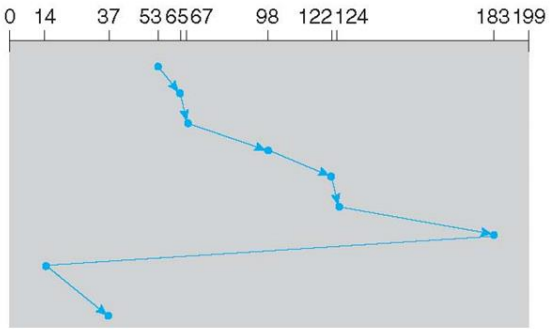
- **电梯算法**
  - 循环扫描调度
    - 如图显示了200个柱面的全部磁头移动。
      - 假设盘臂向0移动，并且初始磁头位于柱面53中。包含柱面请求的队列为

        98, 183, 37, 122, 14, 124, 65, 67



HDD Scheduling

- **电梯算法**
  - 循环查看调度C-LOOK
    - 循环查看(C-LOOK)调度是C-SCAN的LOOK版本
    - 一般来说，循环版本更公平，但因为回程不提供服务，总寻道时间更长。
    - 扫描版本的总寻道时间大于相应的查看版本。



HDD Scheduling

- **磁盘调度**
  - 例子（来自William Stallings的书）

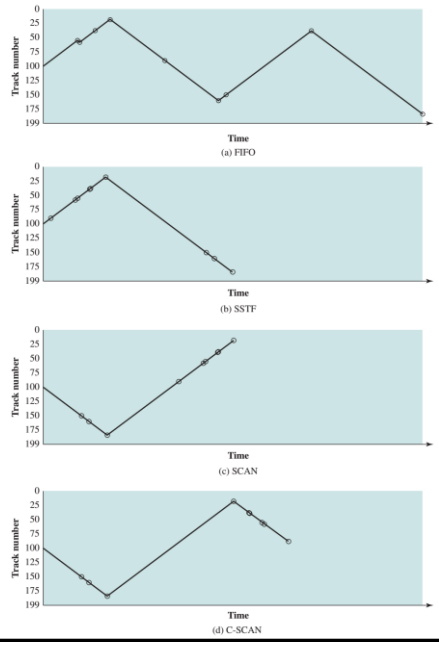
Table 11.2 Comparison of Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	Average seek length	27.5	Average seek length	27.8	Average seek length	35.8

请求序列

HDD Scheduling

- **磁盘调度**
  - 例子(William Stallings)



HDD Scheduling

- **磁盘调度**
  - 其他磁盘调度策略
    - 搭便车Pickup
      - FCFS和LOOK的组合
      - 通过FCFS转到下一个I/O请求，但在到达该请求的进程中为所有现有请求提供服务。
    - 优先级
      - 目标不是优化磁盘使用，而是满足其他目标。
      - 短批处理作业可能具有更高的优先级。
      - 提供良好的交互响应时间。

磁盘调度

- 扫描算法的变种
  - FScan
    - 使用两个队列。
    - 一个队列为空以接收新请求。
  - N步扫描
    - 将磁盘请求队列分段为长度为N的子队列。
    - 使用扫描算法，一次处理一个子队列。
    - 处理某个队列时，新请求添加到其他队列。

选择磁盘调度算法

- 如果磁盘负载较低，则选择FCFS。
- SSTF是常用的，具有天然的吸引力。
- SCAN和C-SCAN对于磁盘负载较大的系统性能更好
  - 减少饥饿。
- 性能取决于请求的数量和类型。
- 对磁盘服务的请求可能会受到文件分配方法和元数据布局的影响。
- 磁盘调度算法应该作为操作系统的一个单独模块编写，如果需要，可以用不同的算法替换。
- SSTF或LOOK是默认算法的合理选择。

HDD Scheduling

- **选择磁盘调度算法**
  - 来自William Stallings

Table 11.3 Disk Scheduling Algorithms

Name	Description	Remarks
Selection according to requestor		
Random	Random scheduling	For analysis and simulation
FIFO	First-in-first-out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last-in-first-out	Maximize locality and resource utilization
Selection according to requested item		
SSTF	Shortest-service-time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of *N* records at a time	Service guarantee
FSCAN	*N*-step-SCAN with N = queue size at beginning of SCAN cycle	Load sensitive

Disk Management

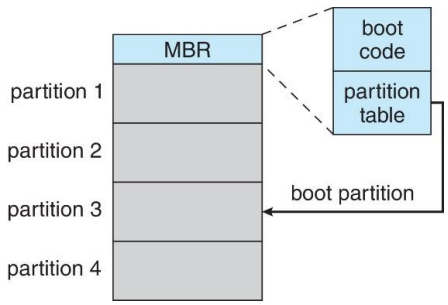
- **磁盘管理**
  - 低级格式化或物理格式化
    - 低级格式化将磁盘划分为磁盘控制器可以读取和写入的扇区。
    - 每个扇区可以保存头部信息、数据和纠错码(ECC).
    - 扇区通常包含512字节的数据，但可以选择。
  - 要使用磁盘保存文件，操作系统仍然需要在磁盘上记录自己的数据结构。
    - 将磁盘分区为一个或多个柱面组，每个柱面组称为逻辑磁盘
    - 逻辑格式化或“创建文件系统”
  - 为了提高效率，大多数文件系统将块分组到聚簇(cluster)中。
    - 磁盘I/O基于块进行
    - 文件I/O基于聚簇进行

Disk Management

- **磁盘管理**
  - 原始磁盘访问
    - 应用程序希望自己进行数据块管理，而不是交给操作系统管理
      - 例如，数据库管理系统。
  - 引导块初始化系统。
    - 引导程序(bootstrap)存储在ROM中。
    - 引导加载程序(bootstrap loader)存储在引导分区的引导块中。
  - 诸如保留扇区之类的方法用于处理坏块。

Disk Management

- **磁盘管理**
  - 从Windows中的磁盘引导



Disk Management 39 / 54

磁盘管理

- Linux的引导过程
  - (1) BIOS（基本输入输出系统）
    - 执行开机自检，开机自检
  - (2) MBR（主引导记录，磁盘中的第一个扇区）
    - 加载GRUB2
  - (3) GRUB2（大统一引导加载程序版本2）
    - 加载vmlinuz内核映像
    - 提取initramfs映像的内容
  - (4) 内核
    - 从initrd映像加载必要的驱动程序模块；Systemd启动系统第一个进程。
    - 内核初始化要点包括：
      - 初始化CPU组件，例如MMU
      - 初始化调度程序(PID 0)
      - 以rw读写模式装载根文件系统
      - 分叉init进程(PID 1)

Disk Management 40 / 54

磁盘管理

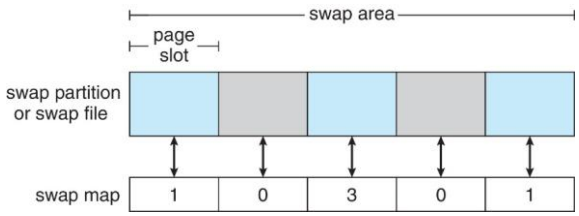
- Linux的引导过程
  - (4) 内核
    - 本质上，内核初始化做两件事：
      - 启动共享资源管理器的核心系统（RAM、处理器和大容量存储）
      - 启动一个进程/sbin/init
    - init进程是第一个进程，它加载所有各种守护进程并装载/etc/fstab下列出的所有分区。
  - (5) SYSTEMD
    - 从/etc/system读取conf配置文件
    - 读取由/etc/systemd/system/default.target链接的文件
    - 使系统进入系统目标定义的状态

Disk Management

- **交换空间管理**
  - 交换空间Swap-space
    - 虚拟内存使用磁盘空间作为主内存的扩展。
    - 由于内存容量增加，现在不太常用。
  - 交换空间可以从普通文件系统中创建，更通常，它也可以位于单独的原始(raw)磁盘分区中。
  - 4.3BSD在进程启动时分配交换空间。
    - 空间中包含程序段和数据段。
    - 内核使用交换映射来跟踪交换空间的使用。
  - Solaris 2仅在脏页被强制移出物理内存时分配交换空间，而不是在首次创建虚拟内存页时分配交换空间。
    - 文件数据写入交换空间，直到请求写入文件系统
    - 其他脏页面也将临时被转移到交换空间
    - 程序段页面将直接丢弃，根据需要再从文件系统中重新读取。
  - 如果系统的交换空间不足怎么办？
    - 有些系统允许多个交换空间

Disk Management

- **交换空间管理**
  - Linux上用于交换的数据结构



Disk Management

- **RAID**
  - RAID
    - “独立磁盘的冗余阵列”
      - 过去的“廉价磁盘冗余阵列”。
    - 多个磁盘驱动器通过冗余提供可靠性。
  - 平均故障间隔时间 – MTBF, 另一故障发生将导致数据丢失的暴露时间。(平均失效时间)
    - N个磁盘中某个故障的机会远高于单个特定磁盘故障的机会
  - 平均修复时间 – 更换发生故障的驱动器和恢复驱动器上的数据所需的时间。(平均修复时间)
  - 基于上述因素（出现故障并修复）的平均数据丢失时间
  - 如果镜像磁盘故障独立，考虑100000个小时的MTBF和10个小时的平均时间来修复单个磁盘。
    - 这种镜像系统的平均数据丢失时间为

      $100000^2 / (2 \times 10) = 500 \times 10^6$（小时），
      - 或者57000年！
  - 经常与NVRAM结合以提高写入性能
  - 磁盘使用技术的几项改进涉及使用多个磁盘协同工作

Disk Management

出现数据丢失要同时满足两个条件：
（1）第一个磁盘损坏。P(A)
（2）在第一个磁盘损坏的前10小时或后10小时之内，第二个磁盘损坏。P(B|A)
每个磁盘平均故障时间是100000小时，也就是说，每过100000小时，条件1预期能被满足一次，在条件1被满足的条件下，发生条件2的概率为 20/100000。可以这样理解，每隔100000小时，条件1被满足1次， 条件2被满足20/100000次，那么发生一次数据丢失，也就是条件2满足1次所需要的时间，就是100000 / (20/100000) = 100000^2/20 小时了。

用概率知识理解:

定义：设A，B 是两个事件，且A不是不可能事件，则称 P(B|A)=P(AB)/P(A) 为在事件A发生的条件下，事件B发生的条件概率。
这里，P(A) = 1/100000，P(B|A) = 20/100000
两条件同时成立（数据丢失）的概率：P(AB) = P(A)×P(B|A) = 20 / 100000^2

概率为P的事件，期望1/P (= 100000^2/20小时)会发生1次！

  - 这种镜像系统的平均数据丢失时间为

    $100000^2 / (2 \times 10) = 500 \times 10^6$（小时），
    - 或者57000年！
  - 经常与NVRAM结合以提高写入性能
  - 磁盘使用技术的几项改进涉及使用多个磁盘协同工作

Disk Management

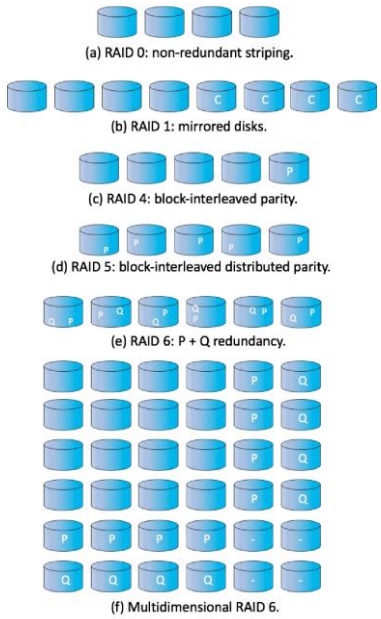
- **RAID**
  - 数据分拆(data striping)
    - 按位分拆
      - 按位分拆每个字节的位到多个驱动器上。
      - 例如，对于8个驱动器组成的阵列，把每个字节的第 i 位写入第 i 个驱动器. 该阵列可被视为一个驱动器，其扇区大小为正常大小的8倍，访问速率为正常速率的8倍。
    - 按块分拆
      - 按块分拆是唯一常用的分拆。它跨多个驱动器分拆文件块
      - 例如，对于n个驱动器，文件的第 i 块将保存到第 (i mod n) 个驱动器。
    - 其他级别的分拆
      - 扇区的字节；块的扇区。
    - 通过分拆在存储系统中实现的并行性是：
      - 通过负载均衡提高多个小型访问（即页面访问）的吞吐量
      - 减少大型访问的响应时间。

Disk Management

- **RAID**
  - RAID分为六个不同的级别
    - 镜像(RAID 1)保留每个磁盘的副本
    - 镜像后分拆(RAID 1+0)或分拆后镜像 (RAID 0+1)提供了高性能和高可靠性。
    - 块交错奇偶校验(RAID 4, 5, 6)使用的冗余要少得多。
  - 如果阵列出现故障，存储阵列中的RAID仍可能出现故障，因此阵列之间的数据自动复制很常见。
    - 通常，少量热备盘未分配，会自动更换故障磁盘并在其上重建数据。

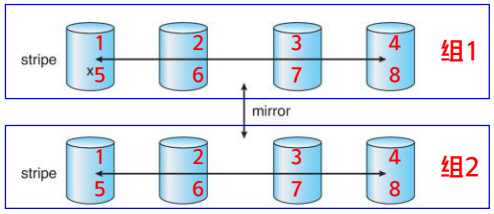
Disk Management

- **RAID**
  - RAID的级别



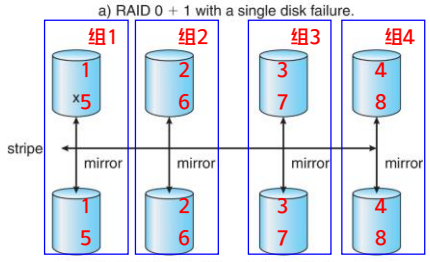
Disk Management

- **RAID**
  - RAID 0+1和RAID 1+0



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

单磁盘故障，整个条带不可访问！



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

单磁盘故障，其镜像仍然可用。（理论上更优）

Disk Management

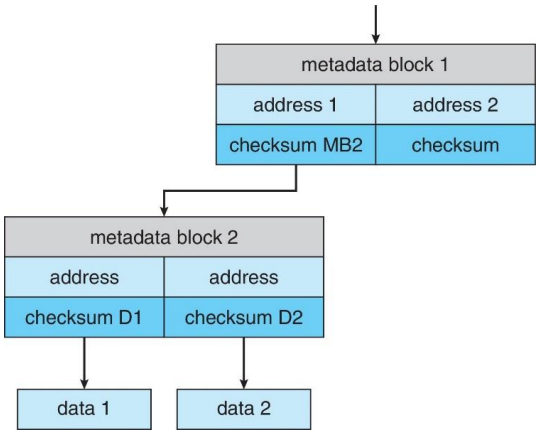
- **其他特性**
  - 无论在何处实现RAID，都可以增加其他有用的功能。
  - 快照是一组更改发生之前的文件系统视图。
    - 在某个时间点
  - 复制是在不同地点之间自动复制写操作。
    - 用于冗余和灾难恢复
    - 可以是同步的，也可以是异步的
  - 热备盘不用于数据，但配置为在驱动器发生故障时用作替换。
    - RAID生产环境会自动使用它来更换发生故障的磁盘，并在可能的情况下重建RAID集。
    - 热备盘缩短了平均修复时间。
  - RAID本身不能防止或检测数据损坏或其他错误，只能防止磁盘故障

Disk Management

- **其他特性**
  - Solaris ZFS增加所有数据和元数据的校验和。
    - 校验和与指向对象的指针一起保存，以检测对象是否正确以及是否更改。
    - 校验和可用于检测和纠正数据和元数据损坏。
    - ZFS将卷和分区消除，通过RAID集组成存储池
      - 磁盘在池中分配的
      - 具有池的文件系统共享其存储池，如内存一样调用malloc()和free()使用和释放空间。

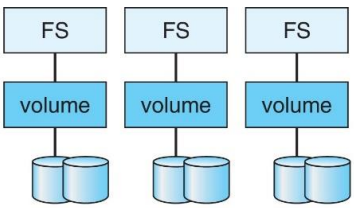
Disk Management

- **其他特性**
  - ZFS对所有元数据和数据进行校验和

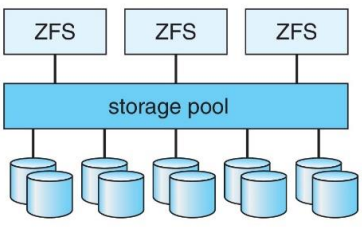


Disk Management

- **其他特性**
  - 传统存储和池存储



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Stable-Storage Implementation

- **稳定存储的实现**
  - 预写日志方案需要稳定的存储。
  - 稳定的存储意味着数据永远不会丢失（由于故障等）
  - 要实现稳定存储，请执行以下操作：
    - 在多个具有独立故障模式的非易失性存储介质上复制信息
    - 以受控的方式更新信息，以确保在数据传输或恢复进程中出现任何故障后，我们能够恢复稳定的数据。
  - 磁盘写入有三种结果之一。
    - 成功完成 – 数据已正确写入磁盘。
    - 部分故障 – 在传输进程中发生故障，因此只有部分扇区写入了新数据，且在故障期间写入的扇区可能已损坏。
    - 完全故障 – 故障发生在磁盘写入开始之前，因此磁盘上以前的数据值保持不变。

Stable-Storage Implementation

- **稳定存储的实现**
  - 若在块写入期间发生故障，恢复过程将块恢复到一致状态。
    - 系统为每个逻辑块维护2个物理块，并执行以下操作：
      1. 写入第一物理块
      2. 成功后，向第二物理块写入相同信息
      3. 仅在第二次写入成功完成后，才声明完成。
    - 系统经常使用NVRAM作为一个物理块设备来加速。