Lecture8 文件系统 2

文件系统的管理

1. 文件系统的可靠性
   1) 可靠性

      抵御和预防各种物理性破坏和人为性破坏的能力

      a) 坏块问题（坏块文件）

      b) 备份：通过转储操作，形成文件或文件系统的多个副本

         - 全量转储：定期将所有文件拷贝到后援存储器
         - 增量转储：只转储修改过的文件，即两次备份之间的修改，减少系统开销
         - 物理转储：从磁盘第 0 块开始，将所有磁盘块按序输出到磁带
         - 逻辑转储（多用但是慢）：从一个或几个指定目录开始，递归地转储自给定日期后所有更改的文件和目录
2. 文件系统一致性
   1) 问题

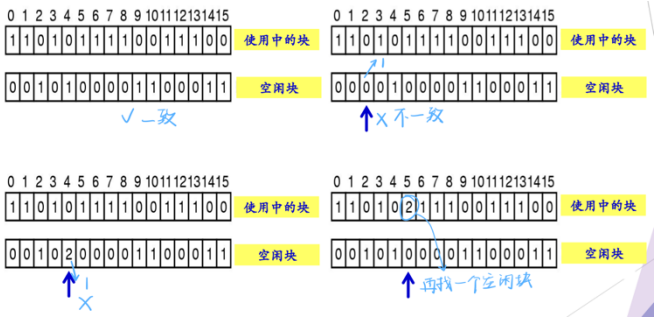
      a) 磁盘块 → 内存 → 写回磁盘块

      b) 若在写回之前，系统崩溃，则文件系统出现不一致
   2) 解决方案

      a) 设计一个实用程序，当系统再次启动时，运行该程序，检查磁盘块和目录系统

      b) 例子：UNIX 一致性检查工作过程

         - 两张表，每块对应一个表中的计数器，初值为 0
         - 表 1：记录了每个磁盘块在文件中出现的次数
         - 表 2：记录了每个磁盘块在空闲块表中出现的次数



3. 文件系统写入方式

应当考虑文件系统的一致性与性能

1) 通写（write-through）

a) 内存中的修改立即写到磁盘

b) 缺点：性能差

c) 例： FAT 文件系统

2) 延迟写（lazy-write）

a) 利用回写（write back）缓存的方法得到高速

b) 缺点：可恢复性差

3) 可恢复写（transaction log）

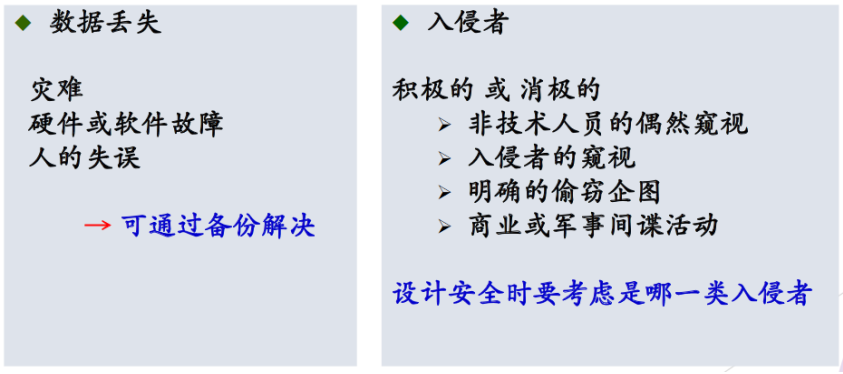
a) 采用事务日志来实现文件系统的写入（日志+回写）

b) 既考虑安全性，又考虑速度性能

c) 例：NTFS

4. 文件系统安全性

1) 安全性：确保未经授权的用户不能存取某些文件



5. 文件的保护机制

1) 作用

a) 用于提供安全性、特定的操作系统机制

b) 对拥有权限的用户，应该让其进行相应操作，否则，应禁止

c) 防止其他用户冒充对文件进行操作

2) 实现

a) 用户身份认证

- 当用户登录时，检验其身份（用户是谁，用户拥有什么，用户知道什么）

- 口令、密码
- 物理鉴定：磁卡，签名分析
- 基于生物特征信息的认证
- CAPTCHA 测试

b) 访问控制（RWX）

主动控制：访问控制表	能力表(权限表)
✓ 每个文件一个 ✓ 放在内核空间 ✓ 记录用户ID和访问权限 ✓ 用户可以是一组用户 ✓ 文件可以是一组文件	✓ 每个用户一个 ✓ 放在内核空间 ✓ 记录文件名及访问权限 ✓ 用户可以是一组用户 ✓ 文件可以是一组文件

3) 目标：保证文件数据不能被随意访问

4) UNIX 的文件保护

a) 审查用户的权限，审查本次操作的合法性

b) 采用文件的二级存取控制

- 第一级：对访问者的识别，对用户分类
  - ✓ 文件主（owner）
  - ✓ 文件主的同组用户（group）
  - ✓ 其它用户（other）
- 第二级：对操作权限的识别，对操作分类
  - ✓ 读操作（r）
  - ✓ 写操作（w）
  - ✓ 执行操作（x）
  - ✓ 不能执行任何操作（-）

例子：rwx rwx rwx
chmod 711 file1 或 chmod 755 file2

6. 数据恢复技术

1) 数据恢复的原理

当磁盘、分区、文件遭到破坏时，其数据未真正被覆盖，只是数据在磁盘上的组织形式被破坏，以至于操作系统或用户不能访问

2) 哪些情况下数据不能被恢复？在 os 数据（如 inode 区）被破坏的情况下

3) 数据恢复包括系统数据恢复和用户数据恢复（利用固定的文件布局）

4) 数据恢复手段：工具和手工（计算机取证技术）

文件系统的性能

1. 简介

   1) 磁盘服务速度和可靠性是系统性能和可靠性的主要瓶颈

   2) 设计文件系统应尽可能减少磁盘访问次数（核心思想）

   3) 提高文件系统性能的方法：

      a) 已经学过的：目录项(FCB)分解、当前目录、磁盘碎片整理

      b) 将要学的：磁盘高速缓存、磁盘调度、提前读取、合理分配磁盘空间、信息的优化分布、RAID 技术（记）

2. 磁盘高速缓存（最重要）

有些系统称为文件缓存、块高速缓存、缓冲区高速缓存

1) 设计思想

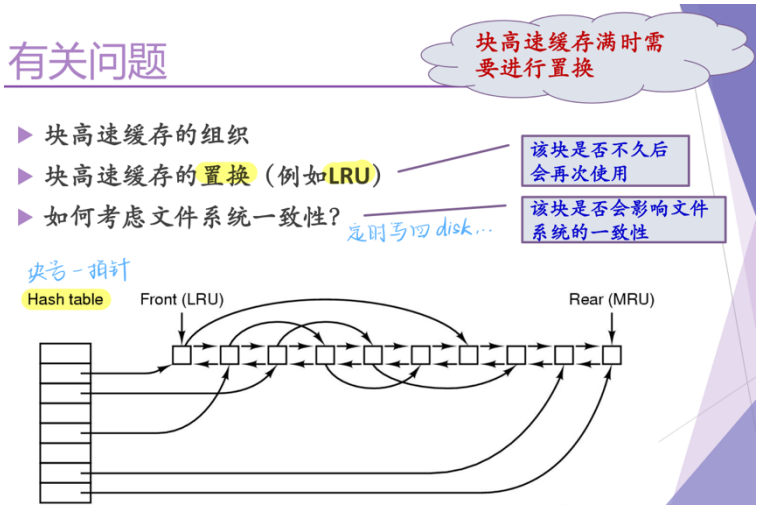
内存中为磁盘块设置的一个缓冲区，保存了磁盘中某些块的副本

2) 实现

   a) 出现对某块的 I/O 请求时，先确定该块是否在磁盘高速缓存中。如果在则直接读；否则，先将数据块读到磁盘高速缓存中，再拷贝（所需的内容，有可能是磁盘的一部分）到所需的地方

   b) 由于访问的局部性原理，当一数据块被读入磁盘高速缓存以满足一个 I/O 请求时，很有可能将来还会再访问到这块数据（可以考虑预取策略等）

3) 有关问题



4) UNIX 系统的磁盘高速缓存

3. 提前读取

1) 简介

a) 思路：每次访问磁盘，多读入一些磁盘块

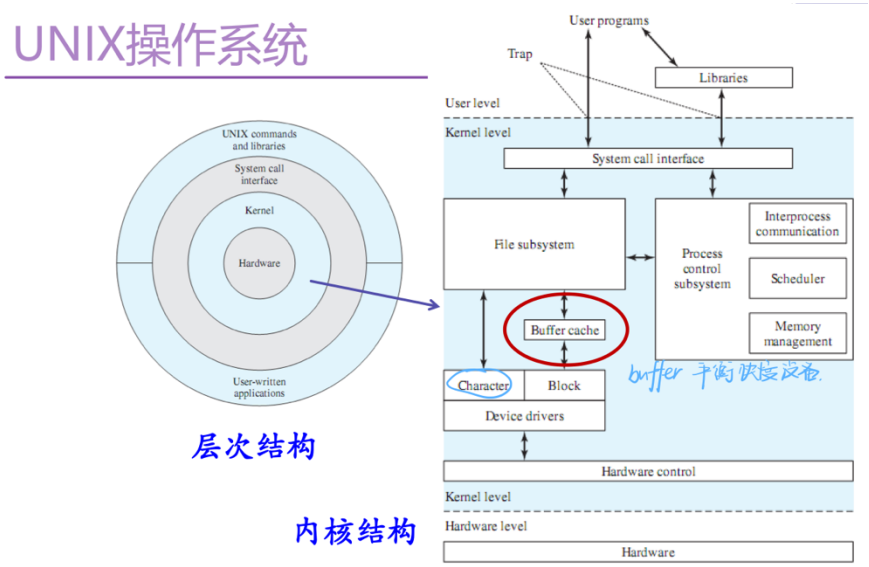
b) 依据：程序执行的空间局部性原理

c) 开销：较小(只有数据传输时间)

d) 具有针对性

2) Windows 的文件访问方式

a) 不使用文件缓存



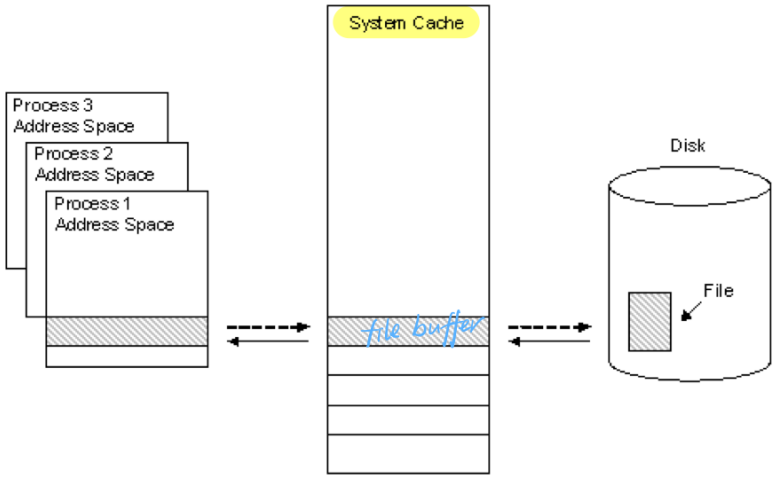
- 普通的方式
- 通过 Windows 提供的 FlushFileBuffer 函数实现

b) 使用文件缓存（默认模式）

- 预读取
  - ✓ 每次读取的块大小、缓冲区大小、置换方式
- 写回
  - ✓ 写回时机选择、一致性问题
- 异步模式
  - ✓ 不再等待磁盘操作的完成
  - ✓ 使处理器和 I/O 并发工作

c) 用户对磁盘的访问通过访问文件缓存来实现

- 由系统的 cache manager（线程）来实现对缓存的控制
  - ✓ 读取数据的时候预取(prefetch)
  - ✓ 在 cache 满的情况下，根据 LRU 原则清除缓存的内容
  - ✓ 定期地更新磁盘上的内容使其与 Cache 一致（1 秒）
- Write-back 机制
  - ✓ 在用户要对磁盘写数据时，只更改 Cache 中的内容
  - ✓ 由 Cache Manager 来决定何时将更新反映到磁盘



阴影部分为需要访问的数据，因此数据在磁盘、系统cache空间和进程空间有3份拷贝，一般情况下用户对数据的修改并不直接反映到磁盘上，而是通过write-back机制由lazy writer定期地更新到磁盘

4. 合理分配磁盘空间

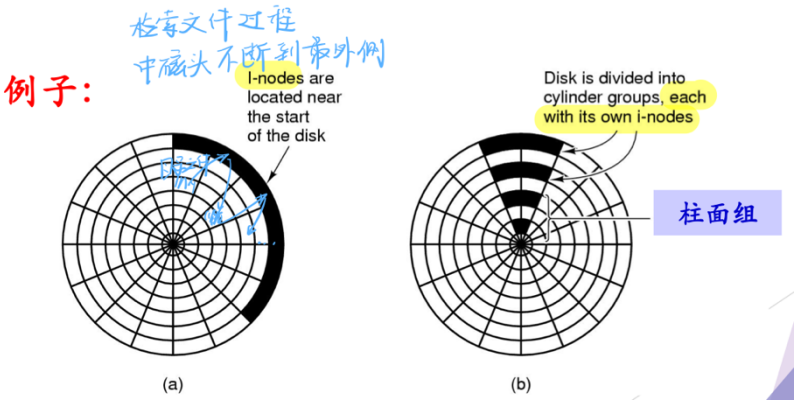
1) 设计思想

分配块时，把有可能顺序存取的块放在一起 → 尽量分配在同一柱面上，从而减少磁盘臂的移动次数

2) 优缺点

减少了寻道时间，但可能会使外部碎片增加

例子：



5. 磁盘调度

1) 简介

当多个访盘请求在等待时，采用一定的策略，对这些请求的服务顺序调整安排

2) 目标

降低平均磁盘服务时间，达到公平、高效

a) 公平：一个 I/O 请求在有限时间内满足

b) 高效：减少设备机械运动所带来的时间浪费

3) 优点

一次访盘时间 = 寻道时间+旋转时间+传输时间

a) 减少寻道时间（旋转时间不易优化）

b) 减少延迟时间

4) 磁盘调度算法

假设磁盘访问序列：98，183，37，122，14，124，65，67

读写头起始位置：53

要求计算：

(1) 磁头服务序列

(2) 磁头移动总距离（道数）

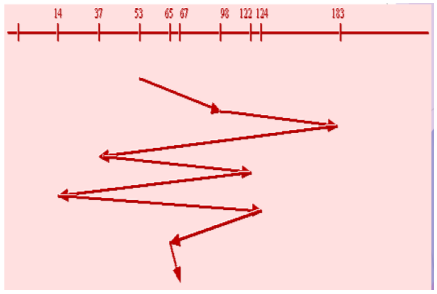
a) 先来先服务

- 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，相临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

假设磁盘访问序列：
98，183，37，122，14，124，65，67

读写头起始位置：53
640 磁道 (平均 80)

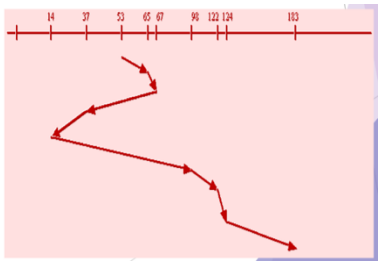


b) 最短寻道时间优先

- 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：造成某些访问请求长期等待得不到服务（饥饿）

假设磁盘访问序列：
98，183，37，122，14，124，65，67

读写头起始位置：53
236 磁道 (平均 29.5)

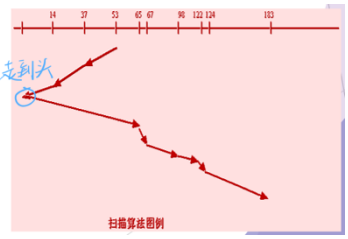


c) 扫描算法（SCAN，电梯算法）

- 折中权衡距离和方向，平衡效率与公平性
- 做法
  - ✓ 当设备无访问请求时，磁头不动（长时间不动可能复位）
  - ✓ 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

假设磁盘访问序列：
98，183，37，122，14，124，65，67

读写头起始位置：53
218 磁道 (平均 27.25)



d) 单向扫描调度算法 C-SCAN

- 做法
  - ✓ 总是从 0 号柱面开始向里扫描
  - ✓ 按照各自所要访问的柱面位置的次序去选择访问者
  - ✓ 移动臂到达最后个一个柱面后，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
- 优点：减少了新请求的最大延迟

e) N-step-SCAN

- 做法
  - ✓ 把磁盘请求队列分成长度为 N 的子队列，每一次用 SCAN 处理一个子队列
  - ✓ 在处理某一队列时，新请求必须添加到其他某个队列中（避免饥饿，否则若同时有新请求访问当前磁头所在磁盘块，会导致其他访问无法得到处理）
  - ✓ 如果在扫描的最后剩下的请求数小于 N，则它们全都将在下一次扫描时处理
  - ✓ 对于比较大的 N 值，其性能接近 SCAN；当 N＝1 时，即 FIFO
- 优点：克服“磁头臂的粘性”，避免饥饿

f) FSCAN

- 做法
  - ✓ 使用两个子队列
  - ✓ 扫描开始时，所有请求都在一个队列中，而另一个队列为空
  - ✓ 扫描过程中，所有新到的请求都被放入另一个队列中
  - ✓ 对新请求的服务延迟到处理完所有老请求之后

- 优点：克服“磁头臂的粘性”，避免饥饿

g) 旋转调度算法

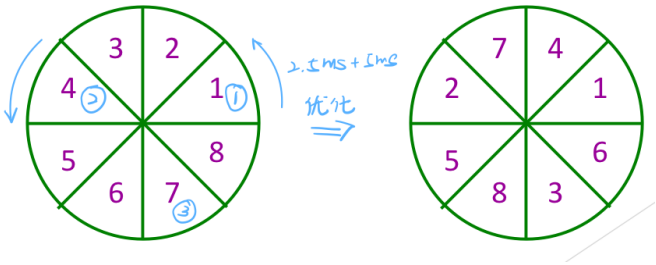
- 根据延迟时间来决定执行次序的调度
- 分析场景
  - ✓ 若干等待访问者请求访问同一磁头上的不同扇区
  - ✓ 若干等待访问者请求访问不同磁头上的不同编号的扇区
  - ✓ 若干等待访问者请求访问不同磁头上具有相同的扇区
- 解决方案
  - ✓ 对于前两种情况：总是让首先到达读写磁头位置下的扇区先进行传送操作（先到哪个扇区就读谁）
  - ✓ 对于第三种情况：这些扇区同时到达读写磁头位置下，可任意选择一个读写磁头进行传送操作
- 注意：所有磁头同步运动且每次只能有一个磁头访问某一个扇区

6. 信息的优化分布

1) 记录在磁道上的排列方式也会影响输入输出操作的时间

2) 例子

处理程序要求顺序处理 8 个记录；磁盘旋转一周为 20 毫秒/周；花 5 毫秒对记录进行处理



7. 记录的成组与分解

1) 记录的成组

a) 把若干个逻辑记录合成一组存放一块的工作

- 进行成组操作时必须使用内存缓冲区，缓冲区的长度等于逻辑记录长度乘以成组的块因子

b) 目的

- 提高了存储空间的利用率
- 减少了启动外设的次数，提高系统的工作效率

2) 记录的分解

a) 从一组逻辑记录中把一个逻辑记录分离出来的操作

- 同样要先把记录读到内存缓冲区中

3) 典型例子—目录文件

8. RAID 技术

1) 设计思想

a) 设计时要考虑的问题

磁盘存储系统的速度、容量、容错、数据灾难发生后的数据恢复以满足不断增长的数据存储需求

b) 解决方案：RAID（独立磁盘冗余阵列）(Redundant Arrays of Independent Disks)

- 多块磁盘按照一定要求构成，操作系统则将它们看成一个独立的

存储设备

- 高性能、容错、高可靠性（且便宜）的存储技术

2) 技术框架

a) 设计思想

通过把多个磁盘组织在一起，作为一个逻辑卷提供磁盘跨越功能

b) 数据的组织和存储

- 通过把数据分成多个数据块 ，并行写入/读出多个磁盘，以提高数据传输率（数据分条 stripe）（基本思路）
- 通过镜像（mirroring）或数据校验（data parity）操作，提高容错能力（冗余）和扩展性
- 条带化、镜像、校验按“字节”或者“位”或者“块”或者 “对象”
- 最简单的 RAID 组织方式：镜像

  最复杂的 RAID 组织方式：块交错校验

  通过不同的方式，组合多个 RAID 级别以获得更高的性能和容错能力

3) RAID 0 – 条带化

a) 做法

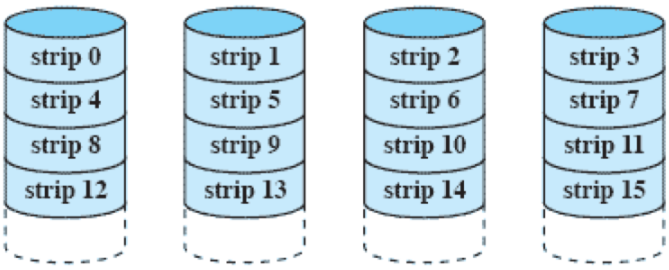
- 数据分布在阵列的所有磁盘上
- 有数据请求时，同时多个磁盘并行操作

b) 优点

- 充分利用总线带宽，数据吞吐率提高
- 驱动器负载均衡且无冗余（即无差错控制）
- 性能最佳

c) 缺点

- 无差错控制，容错率低



4) RAID 1 – 镜像

a) 做法

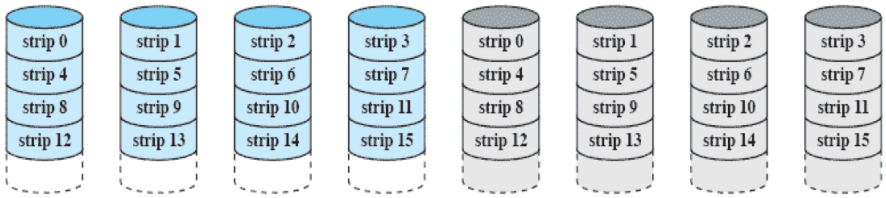
- 所有数据同时存在于两块磁盘的相同位置

b) 优点

- 数据安全性最好
- 最大限度保证数据安全及可恢复性

c) 缺点

- 磁盘利用率 50%



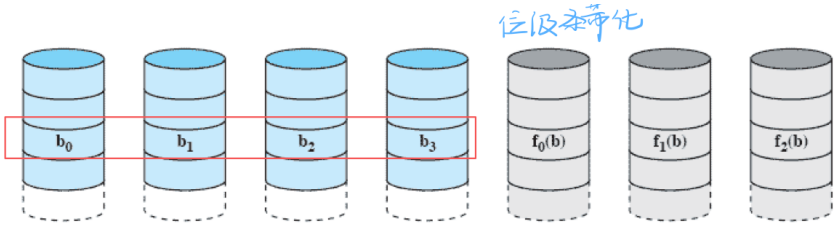
5) RAID 2 并行访问 — 海明码校验

a) 做法

- 将数据条块化分布于不同硬盘（字节或位为单位）
- 加入海明码，在磁盘阵列中间隔写入每个磁盘中
- 数据发生错误时可实施校正以保证输出正确数据
- 存取数据时，整个磁盘阵列一起动作，在各个磁盘的相同位置平行存取，所以有很好的存取时间

b) 优点

- 可以进行数据矫正
- 有很好的存取时间



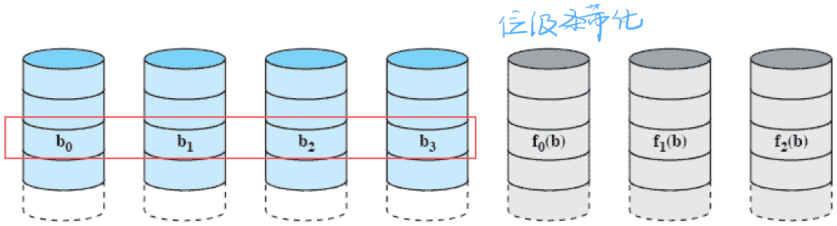
6) RAID 3 交错位奇偶校验

a) 做法

- 类似 RAID2，以字节为单位将数据拆分，并交叉写入数据盘
- 专门设置一个存储校验盘，保存校验码（奇偶校验）

b) 优点

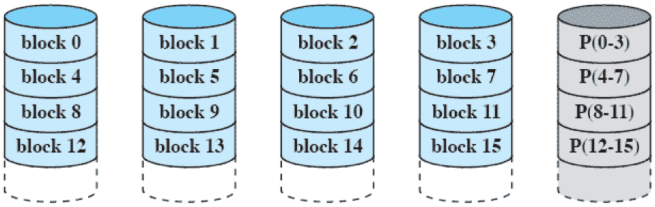
- 可以进行数据矫正
- 有很好的存取时间



7) RAID 4 交错块奇偶校验

a) 做法

- 带奇偶校验
- 与 RAID3 相似，但以数据块为单位



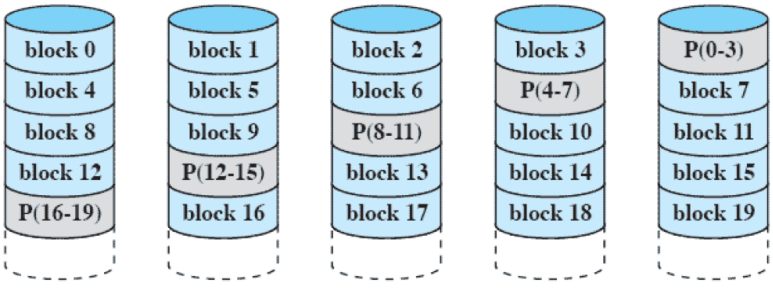
8) RAID 5 交错块分布式奇偶校验

a) 做法

与 RAID4 类似，奇偶校验分散在各个磁盘

b) 优缺点

- 数据读出效率高，写入效率一般
- 磁盘利用率较好，提高了可靠性
- 有写损失



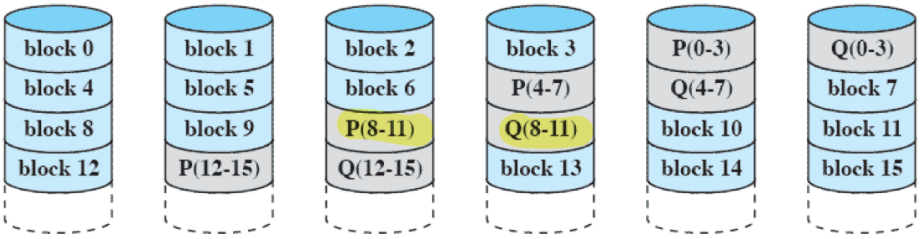
9) RAID 6 交错块双重分布式奇偶校验

a) 做法

在 RAID5 的基础上，设立两个校验码，并将校验码写入两个驱动器

b) 优缺点

- 数据恢复能力增强
- 磁盘利用率降低，写能力降低



10) RAID 7 最优化异步高 I/O 速率及高数据传输率

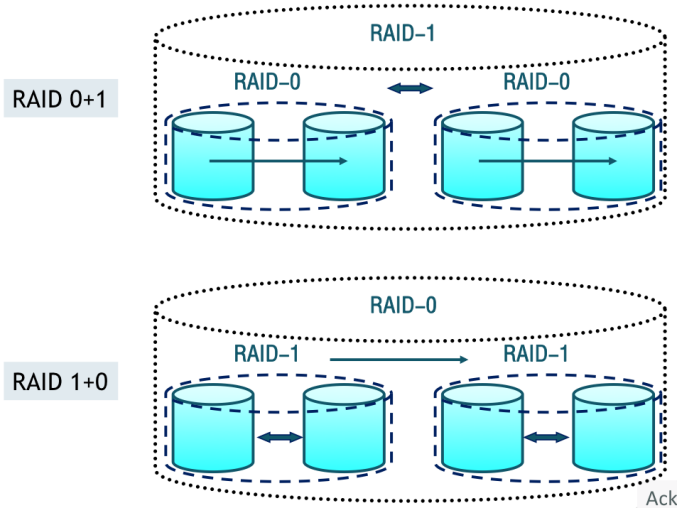
a) 做法

- 自身带有智能化实时操作系统和用于存储管理的管理工具，独立于主机运行
- 每个磁盘有独立的 I/O 通道，与主通道连接
- 操作系统直接对每个磁盘的访问进行控制，可以让每个磁盘在不同的时段进行数据读写

b) 缺点

- 价格高

11) RAID 的嵌套



文件系统的结构设计

1. 文件系统分类
    1) 磁盘文件系统

        实例系统：FAT、NTFS、ext2/3/4、ISO9660 等
    2) Flash 文件系统

        实例系统：F2FS
    3) 数据库文件系统

        实例系统： WinFS
    4) 日志文件系统
    5) 网络/分布式文件系统

        实例系统： NFS、SMB、AFS、GFS
    6) 虚拟文件系统
2. 一些新型的系统
    1) 分布式文件系统 Ceph
        a) 目标

            设计基于 POSIX 的没有单点故障的分布式文件系统，使数据能容错和无缝的复制
        b) 特点

            容错实现和简化海量数据管理的功能
    2) 只读文件系统 EROFS
    3) ReFS（Resilient File System）文件系统
        a) 目标

            提供高度可靠性、数据完整性和扩展性，以满足大规模存储和数据管理的需求
        b) 实现
            - 数据完整性

                使用“integrity streams”的技术来保证数据的完整性。它会计算和存储每个文件的校验和，并在读取文件时验证校验和，以检测和修复数据损坏

- 故障容错

  ReFS 支持故障容错功能，包括数据镜像和数据冗余。可以将数据分布在多个磁盘上，以防止单点故障，并在发生故障时提供自动修复和恢复功能

- 可扩展性

  ReFS 设计为支持大容量存储和高性能工作负载

- 支持稳定写入

  采用“allocate on write”的技术，在进行写操作时不直接修改原始数据，而是创建新的数据副本。这样可以保证写操作的原子性和数据一致性

- 兼容性

  ReFS 与 NTFS 文件系统兼容，并提供逐步迁移的能力。它可以与现有的 NTFS 卷一起使用，逐步将文件系统迁移到 ReFS，而无需重新格式化磁盘

3. 问题

   1) 如何定义文件系统对用户的接口？

      a) 文件及属性

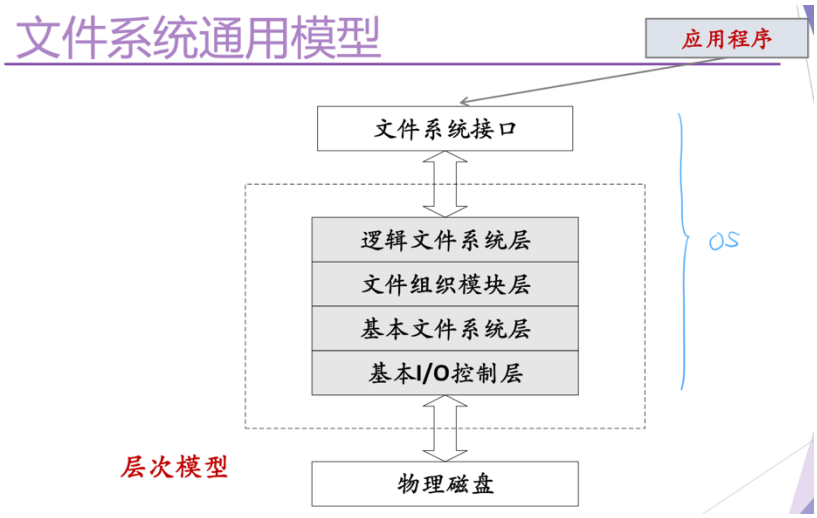
      b) 文件操作

      c) 目录结构

   2) 如何将逻辑文件系统映射到物理磁盘设备上？

      数据结构与算法

3) 文件系统实现时如何分层？



各层的作用

a) 文件系统接口

定义了一组使用和操作文件的方法

b) 逻辑文件系统层

使用目录结构为文件组织模块提供所需的信息，并负责文件的保护和安全

c) 文件组织模块层

负责对具体文件以及这些文件的逻辑块和物理块进行操作

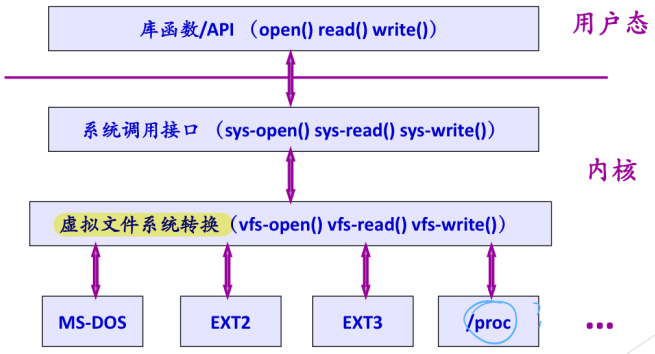
d) 基本文件系统层

主要向相应的设备驱动程序发出读写磁盘物理块的一般命令

e) 基本 I/O 控制层

由设备驱动程序和中断处理程序组成，实现内存和磁盘系统之间的信息传输

4. 虚拟文件系统



5. 日志结构文件系统（LFS- Log-structured File System）

1) 思路

提高磁盘写操作的效率(读操作由文件缓存满足) → 避免寻找写的位置

a) 把整个磁盘看作是一个日志，每次写到其末尾

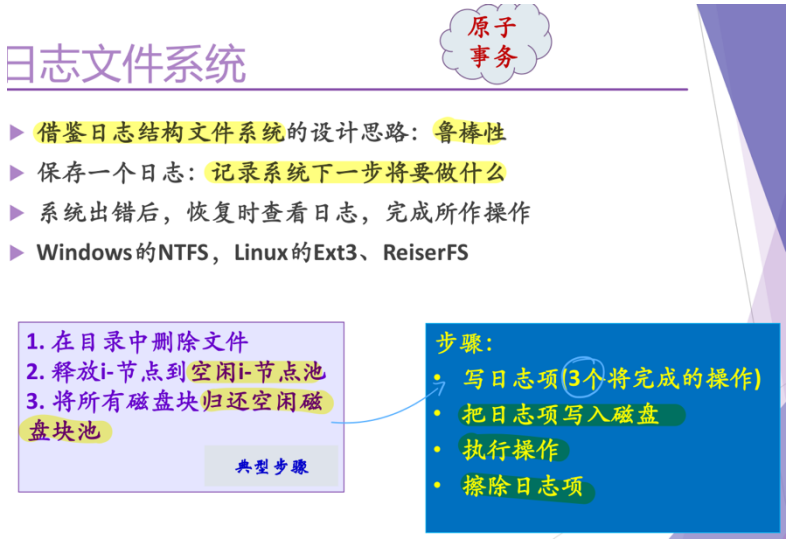
b) 集中(按一段)写入日志末尾

c) 将 I 节点和文件内容一起写入，建立 I 节点表

d) 清理线程：扫描日志，清理，生成新的段

典型的写操作步骤：文件目录 i 节点、目录项、文件的 i 节点、文件本身

6. 日志文件系统



7. 分布式文件系统

1) 分布式计算机系统

a) 简介

- 由多台分散的计算机互连而成的计算机系统
- 强调资源、任务、功能和控制的全面分布
- 各个资源单元（物理或逻辑的）既相互协作又高度自治，能在全系统范围内实现资源管理，动态进行任务分配或功能分配，并行运行分布式程序

b) 工作方式

- 任务分布
- 功能分布

2) 分布式文件系统

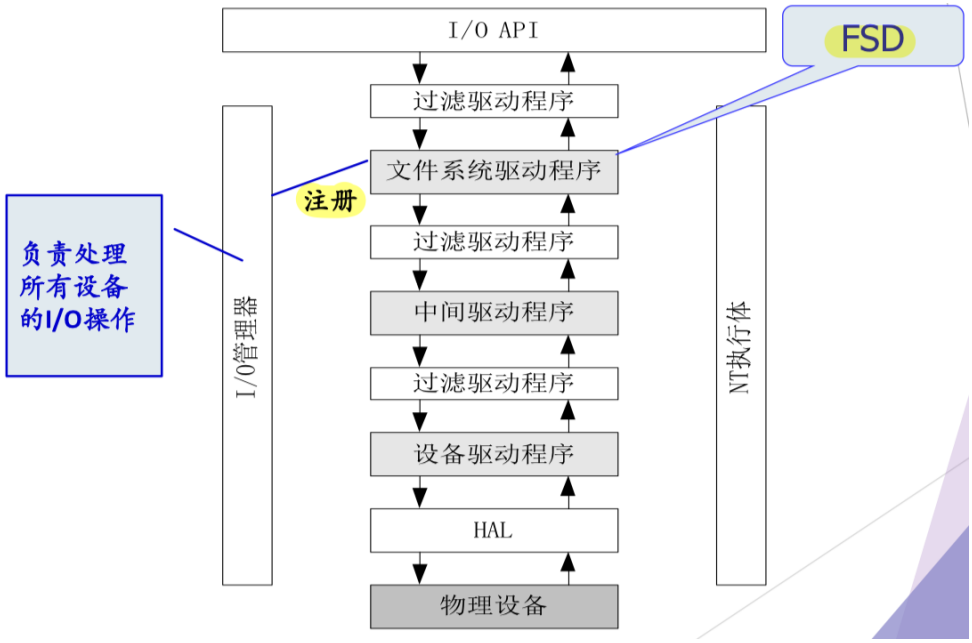
a) 简介

- 完成的功能类似于传统操作系统中的文件系统——永久性存储和共享文件，允许用户直接存取远程文件而不需要将它们复制到本地
- 设计要求

  系统的透明性（transparency）：系统的内部实现细节对用户是隐藏的

  - ✓ 存取透明性
  - ✓ 位置透明性
  - ✓ 故障透明性
  - ✓ 并发存取透明性
  - ✓ 故障透明性
  - ✓ 性能透明性
  - ✓ 复制透明性
  - ✓ 迁移透明性

Windows 文件系统模型



1. FSD（文件系统驱动程序）

   1) 分类

      a) 本地 FSD：允许用户访问本地计算机的数据

      b) 远程 FSD：允许用户通过网络访问远程计算机上的数据

   2) 作用

      Windows 文件系统的有关操作都通过 FSD 来完成

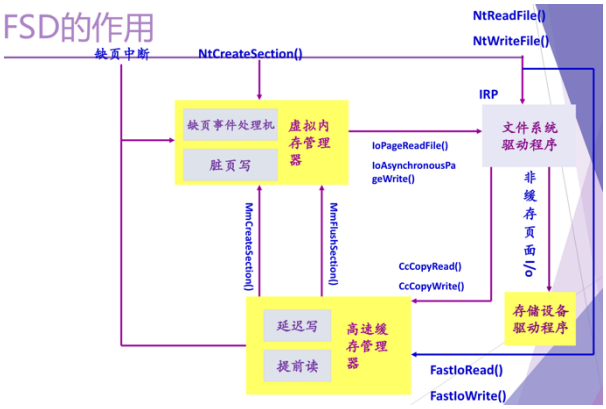
      a) 显式文件 I/O

      b) 高速缓存滞后写

      c) 高速缓存提前读

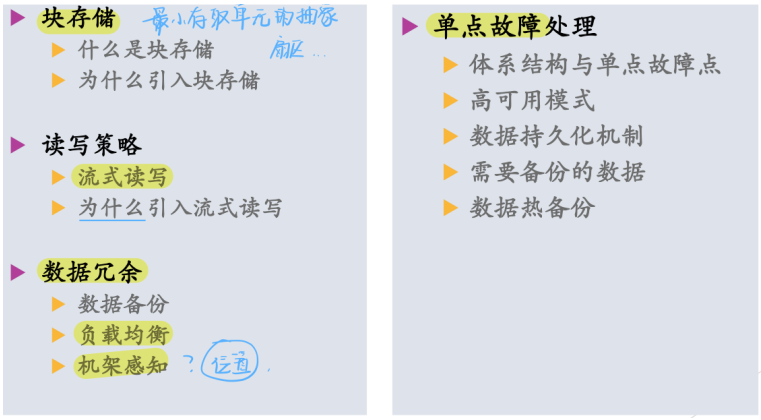
      d) 内存“脏”页写

      e) 内存缺页处理



Hadoop Distributed File System(HDFS)实现机制概述

1. 简介
   1) 分布式文件系统
   2) 部署在低廉的（low-cost）硬件上（和 RAID 思想类似）
   3) 高容错性（fault-tolerant）
   4) 高吞吐量（high throughput）
   5) 适用于超大数据集（large data set）
2. 实现特点



   1) 块存储
      a) 以数据块做基本存储单元，减少了寻址规模
      b) 文件以块为单位存储在不同的磁盘上，文件大小不受限于单个磁盘的容量
   2) 流式读写
      a) 只支持在文件末尾添加数据，不支持在任意位置修改
      b) 并发读文件，不支持并发写文件
      c) 一次写入多次读取；高吞吐量
   3) 数据冗余
      a) 数据备份
      b) 负载均衡
      c) 机架感知
4. 单点故障处理

重点小结

1. 文件系统的管理
    1) 文件系统的备份
    2) 文件系统的一致性
2. 文件系统的性能优化
    1) 块高速缓存
    2) 磁盘调度
    3) RAID 技术
3. 文件系统的结构设计
    1) 文件系统的层次模型
    2) 虚拟文件系统
    3) 日志结构文件系统
    4) 日志文件系统