大容量存储结构

大容量存储结构概述

- 磁盘
  - 磁盘或硬盘 为现代计算机系统提供大量外存。
  - 读写磁头 “飞行” 在一个盘片的表面上方。 磁头附着在磁臂上， 磁臂将所有磁头作为一个整体而一起移动。 盘片的表面逻辑地分成圆形磁道, 再细分为扇区 同一磁臂位置的磁道集合形成了柱面
  - 传输速率 是在驱动器和计算机之间的数据流的速率。 定位时间 或随机访问时间 包括两部分：寻道时间 (移动磁臂到所要柱面的所需时间) 和旋转延迟
- 固态磁盘
  - SSD 为非易失性存储， 可用作硬盘
- 磁带
  - 磁带 曾经用作早期的外存媒介。 虽然它是相对永久的， 可以容纳大量的数据， 但是与内存和磁盘相比， 它的访问时间更长另外， 磁带随机访问要比磁盘随机访问慢千倍，所以磁带对于外存不是很有用

磁 盘 结 构

- 现代磁盘驱动器可以看作逻辑块的一维数组， 这里的逻辑块是最小的传输单位。
- 扇区 0 是最外面柱面的第一个磁道的第一个扇区。这个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进行的。
  - 通过这个映射， 至少从理论上能够将逻辑块号转换为由磁盘内的柱面号、 柱面内的磁道号、 磁道内的扇区号所组成的老式磁盘地址。
- 对于采用恒定线速度 CLV)的媒介， 每个磁道的比特密度是均匀的。 磁道距离磁盘中心越远， 长度越长， 从而也能容纳更多扇区。 当从外部区域移到内部区域， 每个轨道的扇区数量会降低。 最外层的轨道通常比最内层的轨道多拥有 40% 的扇区数。 随着磁头由外磁道移到内磁道， 驱动器增加旋转速度， 以保持传输数据率的恒定。
- 磁盘旋转速度可以保持不变， 因此内部磁道到外部磁道的比特密度不断降低， 以保持数据率不变。 这种方法用于硬盘， 称为恒定角速度 CAV)

磁 盘 连 接

- 主机连接存储
  - 主机连接存储是通过本地 I/O 端口来访问的存储。
- 网络连接存储
  - 网络连接存储 NAS) 设备是一种专用存储系统， 可以通过数据网络来远程访问 客户通过远程过程调用 RPC)
- 存储区域网络
  - 网络连接存储系统的一个缺点是， 存储 I/O 操作消耗数据网络的带宽， 从而增加网络通信的延迟。 这个问题对于大型客户机 服务器环境可能特别严重； 服务器与客户机之间的通信和服务器与存储设备之间的通信， 竞争通信带宽。
  - 存储区域网络 SAN) 为专用网络， 采用存储协议而不是网络协议连接服务器和存储单元

磁盘调度

- 寻道时间是磁臂移动磁头到包含目标扇区的柱面的时间。
- 旋转延迟 是磁盘旋转目标扇区到磁头下的额外时间。
- 磁盘带宽是传输字节的总数除以从服务请求开始到最后传递结束时的总时间。
- FCFS 调度
  - 磁盘调度的最简单形式当然是， 先来先服务 FCFS) 算法。 虽然这种算法比较公平， 但是它通常并不提供最快的服务
- SSTF 调度
  - 在移动磁头到别处以便处理其他请求之前， 处理靠近当前磁头位置的所有请求可能较为合理。 这个假设是最短寻道时间优先 SSTF) 算法的基础。
  - SSTF算法选择处理距离当前磁头位置的最短寻道时间的请求。 换句话说， SSTF 选择最接近磁头位置的待处理请求
  - SSTF 调度本质上是一种最短作业优先 SJF) 调度； 与 SJF 调度一样， 它可能会导致一些请求的饥饿。
- SCAN 调度
  - 对于扫描算法 SCAN , 磁臂从磁盘的一端开始， 向另一端移动； 在移过每个柱面时， 处理请求。 当到达磁盘的另一端时， 磁头移动方向反转， 并继续处理。 磁头连续来回扫描磁盘。 SCAN 算法有时称为电梯算法
- C-SCAN 调度
  - 循环扫描 C SCAN, 是 SCAN 的一个变种， 以提供更均匀的等待时间 C-SCAN 移动磁头从磁盘一端到磁盘另一端 , 并且处理行程上的请求。 然而， 当磁头到达另一端时， 它立即返回到磁盘的开头， 而并不处理任何回程上的请求
- LOOK 调度
  - 磁臂只需移到一个方向的最远请求为止，遵循这种模式的 SCAN 算法和C-SCAN算法分别称为LOOK和C-LOOK调度
- 磁盘调度算法的选择
  - SSTF 是常见的， 并且具有自然的吸引力， 因为它比 FCFS 具有更好的性能。 对于磁盘负荷较大的系统， SCAN 和 C-SCAN表现更好， 因为它们不太可能造成饥饿问题。
  - 然而， 对于任何调度算法， 性能在很大程度上取决于请求的数量和类型。
  - 文件分配方式可以大大地影响磁盘服务的请求。 程序读取连续分配文件时， 生成多个相近位置的磁盘请求， 导致有限的磁头移动。 相比之下， 链接或索引的文件可能包括分散在磁盘上的多个块， 导致更多的磁头移动。
  - 目录和索引块的位置也很重要。 因为每个文件必须打开才能使用， 并且打开文件需要搜索目录结构，所以目录会被经常访问。
  - 由于这些复杂因素， 磁盘调度算法应该作为操作系统的一个单独模块， 这样如果需要，可以用不同的算法来替换。 SSTF 或 LOOK 是默认算法的合理选择。

磁 盘 管 理

- 磁盘格式化
  - 在磁盘可以存储数据之前， 它必须分成扇区， 以便磁盘控制器能够读写。 这个过程称为低级格式化 或物理格式化
  - 在可以使用磁盘存储文件之前， 操作系统仍然需要将自己的数据结构记录在磁盘上。 这分为两步。
    - 第一步是， 将磁盘分为由柱面组成的多个分区操作系统可以将每个分区作为一个单独磁盘。
    - 第二步是逻辑格式化 , 或创建文件系统。 在这一步， 操作系统将初始的文件系统数据结构存储到磁盘上。 这些数据结构包括空闲和已分配的空间和一个初始为空的目录。
- 引导块
  - 为了开始运行计算机， 如打开电源或重启时， 它必须有一个初始程序来运行。 这个初始自举 程序往往很简单。 它初始化系统的所有部分， 从 CPU 寄存器到设备控制器和内存， 接着启动操作系统。
  - 对于大多数计算机， 自举程序处在只读存储器 ROM) 中。 这个位置非常方便，因为 ROM 不需要初始化而且位于固定位置， 这便于处理器在上电或复位时开始执行。
  - 具有启动分区的磁盘称为启动磁盘 或系统磁盘
  - 引导 ROM 内的代码指示磁盘控制器将引导块读到内存 （ 这时不加载设备驱动程序)， 然后开始执行代码。 完整的自举程序比引导 ROM 的自举程序更加复杂。 它可以从非固定的磁盘位置处加载整个操作系统， 并且开始运行操作系统。 即使如此， 完整的自举程序可能很小
- 坏块
  - 对于简单磁盘， 如采用 IDE 控制器的磁盘， 可以手动处理坏块。 一种策略是， 在格式化磁盘时扫描磁盘以便发现坏块。 发现的任何坏块， 标记为不可用， 以便文件系统不再分配它们。
  - 如果在正常操作时块变坏了， 则必须人工运行特殊程序 （ 如 Linux 命令 badlocks)以便搜索坏块并锁定它们。 坏块中的数据通常会丢失。
  - 低级格式化将一些块放在一边作为备用， 操作系统看不到这些块。 控制器可以采用备用块来逻辑地替代坏块。 这种方案称为扇区备用 或扇区转寄

交换空间管理

- 虚拟内存采用磁盘空间作为内存的扩展。 由于磁盘访问比内存访问慢得多， 所以使用交换空间显著降低系统性能， 交换空间的设计和实现的主要目标是， 为虚拟内存提供最佳吞吐量。
- 交换空间的使用
  - 交换空间位置可有两个： 它可以位于普通文件系统之上， 或者它可以是一个单独的磁盘分区。
  - 如果交换空间只是文件系统内的一个大的文件， 则可以采用普通文件系统程序来创建它、 命名它以及分配它的空间。 这种方法虽然实现容易， 但是效率较低。
  - 或者， 可以在单独的原始分区上创建交换空间。 这里不存放文件系统和目录的结构。
  - 相反， 通过单独的交换空间存储管理器， 从原始分区上分配和取消分配块。 这种管理器可以采用针对速度优化 （ 而不是存储效率优化)， 因为 （ 在使用时） 交换空间比文件系统访问更加频繁。
  - 内部碎片可能增加， 但是这种折中是可以接受的， 因为交换空间内的数据的存储时间通常要比文件系统的文件的存储时间更短。

RAID 结构

- 一个系统拥有大量磁盘， 就有机会改善数据的读写速率， 因为磁盘操作可以并行进行。 此外， 这种设置提供能力， 以提高数据存储的可靠性， 因为冗余信息可以存储在多个磁盘上。 因此， 单个磁盘的故障不会导致数据丢失。
- 多种磁盘组织技术统称为磁盘冗余阵列 RAID) 技术， 通常用于处理性能与可靠性问题。
- 通过冗余提高可靠性
  - 可靠性问题的解决是引入冗余 存储额外信息， 这是平常不需要的， 但是在磁盘故障时可以用于重建丢失的信息。 因此， 即使磁盘故障， 数据也不会丢失
  - 最为简单 （ 但最昂贵） 的引入冗余的方法是， 重复每个磁盘。 这种技术称为镜像 由于镜像， 每个逻辑磁盘由两个物理磁盘组成， 并且每次写入都在两个磁盘上进行。 这称为镜像卷
  - 电源故障是一个特别的关注点， 因为它们比自然灾害更为常见。 即使使用磁盘镜像， 如果对两个磁盘写入同样的块， 而在两块完全写入之前电源故障， 则这两块可能处于不一致的状态。
    - 这个问题的一种解决方法是， 先写一个副本， 再写下一个。 另一个是， 为 RAID 阵列添加固态非易失性 RAM 的缓存。 这种写回高速缓存在电源故障时会得到保护；
- 通过并行处理提高性能
  - 采用多个磁盘， 通过将数据分散在多个磁盘上， 也可以改善传输率。 最简单形式是， 数据分条 包括将每个字节分散在多个磁盘上； 这种分条称为位级分条
  - 位级分条可以推广到其他磁盘数量， 它或者是 8 的倍数或者除以 8。 例如， 如果采用 4个磁盘阵列， 则每个字节的位 i和位 4+ i 可存在磁盘； 上。 此外， 分条不必按位级来进行。
  - 例如， 对于块级分条, 文件的块可以分散在多个磁盘上；
  - 磁盘系统的并行化， 通过分条实现， 有两个主要目标：
    - 通过负载平衡， 增加了多个小访问 （即页面访问） 的吞吐量。
    - 降低大访问的响应时间。
- RAID 级别
  - 镜像提供高可靠性， 但是昂贵。 分条提供高数据传输率， 但并未改善可靠性。
  - 在低代价下提供冗余可以有多种方案。 这些方案有不同的性价折中， 并分成不同的级别， 称为 RAID 级别
- RAID 级别的选择
  - 一个考虑是重构性能。 如果磁盘故障， 则重建数据的所需时间可能很大。 如果要求持续提供数据， 如高性能或交互式数据库系统， 那么这可能是个重要因素。 此外， 重建性能影响平均故障时间
- 扩展
  - RAID 概念已扩展到其他存储设备 （ 包括磁带阵列， 甚至无线系统的数据广播。 当应用于磁带阵列时， 即使磁带阵列内的一个磁带损坏， 仍然可以利用 RAID 结构恢复数据。
- RAID的问题
  - RAID 并不总是确保数据对操作系统和使用者是可用的。
  - 大多数的 RAID 实现的另一个问题是缺乏灵活性

稳定存储实现

- 磁盘写入导致三种结果：
  - 成功完成： 数据正确写到磁盘。
  - 部分故障： 在传输中出现故障， 这样有些扇区写了新数据， 而在故障发生时在写的扇区可能已被破坏。
  - 完全故障： 在磁盘写入开始之前发生故障， 因此磁盘上的以前数据值保持不变。
- 要做到这一点， 系统必须为每个逻辑块维护两个物理块。 输出操作执行如下：
  - 将信息写到第一个物理块。
  - 当第一次写入成功完成时， 将同样信息写到第二个物理块。
  - 只有第二次写入成功完成， 才可声明操作完成。