8.3 磁盘管理和 RAID 结构

一、磁盘管理

操作系统还负责磁盘管理有面其他的内容。这里讨论磁盘初始化、引导块、坏块。

1． 磁盘格式化

一个新的磁盘是一个空白板：它只是一些含有磁性记录材料的盘子。在磁盘能存储数据之前，它必须分成扇区以便磁盘控制器能读和写。这个过程称为**低级格式化**（或物理格式化）。低级格式化为磁盘的每个扇区采用特别的数据结构。每个扇区的数据结构通常由头、数据区域（通常为512 B大小）和尾部组成。头部和尾部包含了一些磁盘控制器所使用的信息，如扇区号码和纠错代码(error-correcting code, ECC)。

绝大多数硬盘在工厂时作为制造过程的一部分就已低级格式化了。这一格式化使得制造商能测试磁盘和初始化从逻辑块码到磁盘上无损扇区的映射。

为了使用磁盘存储文件，操作系统还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一个或多个柱面组成的**分区**。操作系统可以将每个分区作为一个独立的磁盘。例如，一个分区可以用来存储操作系统的可执行代码，而其他分区用来存储用户数据。在分区之后，第二步是逻**辑格式化**（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（FAT或者inode）和一个初始为空的目录。

为了提高效率，大多数操作系统将块集中到一大块，通常称作簇(cluster)。磁盘I/O通过块完成，但是文件系统I/O通过簇完成，这样有效确保了I/O可以进行更多的顺序存取和更少的随机存取。

2．引导块

为了让计算机开始运行，如当打开电源时或重启时，它需要运行一个初始化程序。该初始化自举（bootstrap）程序应该很简单。它初始化系统的各个方面，从CPU寄存器到设备控制器和内存，接着启动操作系统。为此，自举程序应找到磁盘上的操作系统内核装入内存，并转到起始地址，从而开始操作系统的执行。

对绝大多数计算机，自举程序保存在只读存储器(ROM)中。这一位置较为方便，由于ROM不需要初始化且位于固定位置，这便于处理器在打开电源或重启时开始执行。而且，由于ROM是只读的，所以不会受计算机病毒的影响。问题是改变这种自举代码需要改变ROM硬件芯片。因此，绝大多数系统只在启动ROM中保留一个很小的自举加载程序，其作用是进一步从磁盘上调入更为完整的自举程序。这一更为完整的自举程序可以容易地进行修改：新版本可写到磁盘上。这个完整的自举程序保存在磁盘的启动块上，启动块位于磁盘的固定位置。拥有启动分区的磁盘称为启动磁盘(boot disk)，或系统磁盘(systemdisk)。

启动ROM中的代码引导磁盘控制器将启动块读入到内存（这时尚没有装入设备驱动程序），并开始执行代码。完整自举程序比启动ROM内的自举加载程序更加复杂，它能从磁盘非固定位置中装入整个操作系统，并开始运行。即使如此，完整的自举程序仍可能很小。

考虑一下Windows 2000中的启动程序。Windows 2000系统将其启动代码放在硬盘上的

第一个扇区(被称为**主引导记录**(master boot record)，或MBR)。此外，Windows 2000中允许硬盘分成一个或多个分区，一个分区为引导分区(boot partition)，包含操作系统和设备驱动程序。 Windows 2000系统通过运行系统ROM上的代码，开始启动。此代码指示系统从MBR读取引导代码。除了含有引导代码，MBR中包含一个硬盘分区列表和一个说明系统引导分区的标志。系统一旦确定引导分区，它读取该分区的第一个扇区(即所谓的引导扇区(boot sector))，并继续余下的启动过程，包括加载各种子系统和系统服务。

3．坏块

由于磁盘有移动部件并且容错能力小（磁头在磁盘表面上飞行），所以容易出问题。有时问题严重，必须替换磁盘，其内容就要从备份介质上恢复到新磁盘上。但更常见的是，一个或多个扇区坏掉。绝大多数磁盘从工厂里出来时就有坏块。根据所使用的磁盘和控制器，对这些块有多种处理方式。

对于简单磁盘，如使用IDE控制器的磁盘，可手工处理坏扇区。例如，MS-DOS format命令执行逻辑格式化，它将扫描磁盘以查找坏扇区。如果format找到坏扇区，那么它就在相应的FAT条目中写上特殊值以通知分配程序不要使用该块。如果在正常使用中块变坏，那么就必须人工地运行一个特殊程序，如chkdsk来搜索坏块，并像前面一样将它们锁在一边。坏块中的数据通常会丢失。

一个典型的坏扇区事务处理可能如下：

·操作系统试图访问逻辑块87。

·控制器计算ECC的值，发现该块是坏的，它将此结果通知操作系统。

·下次操作系统重启时，可以运行一个特殊程序以告诉SCSI控制器用备用块替代坏块。

·之后，每当系统试图访问逻辑块87时，这一请求就转换成控制器所替代的扇区的地址。

这种控制器所引起的重定向可能会使操作系统的磁盘调度算法无效。为此，绝大多数磁盘在格式化时为每个柱面都留了少量的备用块，还保留了一个备用柱面。当坏块需要重新映射时，控制器就尽可能使用同一柱面的备用扇区。

二、交换空间管理

交换空间管理是操作系统的另一底层任务。虚拟内存使用磁盘空间作为内存的扩充。由于磁盘访问比内存访问要慢很多，所以使用交换空间会严重影响系统性能。交换空间设计和实现的主要目的是为虚拟内存提供最佳吞吐量。这里讨论如何使用交换空间，交换空间在磁盘上的什么位置，以及交换空间该如何管理。

1. 交换空间的使用

不同操作系统根据所实现的内存管理算法，可按不同方式来使用交换空间。例如，实现交换的系统可以将交换空间用于保存整个进程映像，包括代码段和数据段。换页系统也可能只用交换空间以存储换出内存的页。系统所需交换空间的量因此会受以下因素影响：物理内存的多少、所支持虚拟内存的多少、内存使用方式等。它可以是数MB到数GB的磁盘空间。

注意，对交换空间数量的高估要比低估更为安全。这是因为如果系统使用完了交换空间，那么可能会中断进程或使整个系统死机。高估只是浪费了一些空间（本可用于存储文件），但并没有造成什么损害。有些系统建议了交换空间的大小。例如，Solaris建议设置交换空间

数量与虚拟内存超出可分页物理内存的数量相等。过去，Linux建议设置交换空间数量是虚拟内存数量的两倍，但是，现在大多数Linux系统使用相当少的交换空间。事实上，现在关于是否设置交换空间，Linux内部有很多争论。

有的操作系统，如Linux，允许使用多个交换空间。这些交换空间通常位于不同磁盘上，这样因换页和交换所引起的I/O系统的负荷可分散在各个系统I/O设备上。

2. 交换空间位置

交换空间可有两个位置：交换空间在普通文件系统上加以创建，或者是在一个独立的磁盘分区上进行。如果交换空间是文件系统内的一个简单大文件，那么普通文件系统程序就可用来创建它、命名它并为它分配空间。这种方式虽然实现简单但是效率较低。遍历目录结构和磁盘分配数据结构需要时间和（可能）过多磁盘访问。外部碎片可能会通过在读写进程镜像时强制多次寻道，从而大大地增加了交换时间。通过将块位置信息缓存在物理内存中，以及采用特殊工具为交换文件分配物理上连续块等技术，可以改善性能，但是遍历文件系统数据结构的开销仍然存在。

另外一种方法是，交换空间可以创建在独立的生(raw)磁盘分区上。这里不需要文件系统和目录结构，只需要一个独立交换空间存储管理器以分配和释放块。这种管理器可使用适当算法以优化速度，而不是优化存储效率，因为交换空间比文件系统访问更频繁。内部碎片可能会增加，但还是可以接受的，这是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短很多。交换空间在启动的时候会初始化，因此任何碎片存在的时间都很短。这种方法在磁盘分区时创建一定量的交换空间。增加更多交换空间可能需要重新进行磁盘分区（可能涉及移动或删除文件系统，以及利用备份以恢复文件系统），或在其他地方增加另外交换空间。

有的操作系统较为灵活，可以使用原始分区空间和文件系统空间进行交换。Linux就是这样的操作系统：策略和实现是分开的，系统管理员可决定使用何种类型。权衡取决于文件系统分配和管理的方便和原始分区交换的住能。

三、RAID 结构

随着磁盘驱动器不断地变得更小更便宜，如今在一台计算机系统上装上大量磁盘从经济上来说已经可行了。一个系统拥有了大量磁盘，它就有机会改善数据读写速度（因为磁盘操作可并行进行）。而且，这种设置也使系统有机会改善数据存储的可靠性，因为可在多个磁盘上存储冗余信息。因此，一个磁盘损坏并不会导致数据丢失。这里的多种磁盘组织技术，通常统称为磁盘冗余阵列(RAID)技术，通常用于提高性能和可靠性。

过去，RAID 是由许多小的、便宜磁盘组成，可作为大的、昂贵磁盘的有效替代品。现在，RAID 的使用主要是因为其高可靠性和高数据传输率，而不是经济原因。因此，RAID 中的 I 表示“独立(independent)"而不是“便宜（inexpensive）”。

最为简单（但最为昂贵）的引入冗余的方法是复制每个磁盘。这种技术称为**镜像**(mirroring)。因此每个逻辑磁盘由两个物理磁盘组成，每次写都要在两个磁盘上进行。如果一个磁盘损坏，那么可从另一磁盘中恢复。只有在第一个损坏磁盘没有替换之前而第二个磁盘又出错，那么数据才会丢失。

对于磁盘镜像，读请求处理的速度可以加倍，这是因为读请求可以发送给任一磁盘（只要两个磁盘都能工作，情况几乎总是这样）。每个读的传输率与单个磁盘系统一样，但是单位时间内读数量加倍了。

对于多个磁盘，通过在多个磁盘上分散数据，可以改善传输率。最简单形式是，**数据分散**是在多个磁盘上分散每个字节的各个位，这种分散称为位级分散。例如，如果有 8 个磁盘，可将每个字节的位 i 写到第 i 个磁盘上。这 8 个磁盘可作为单个磁盘使用，其扇区为正常扇区的 8 倍，更为重要的是它具有 8 倍的传输率。对于这种结构，每个磁盘都参与每次访问（读或写），这样每秒所能处理的访问数与单个磁盘的一样，但每次访问可在同样时间内读相当于单个磁盘系统 8 倍的数据。

位级分散可扩展到其他数量的磁盘，只要该数量为 8 的倍数或能除以 8。例如，如果有 4 个磁盘，每个字节的位 i 和位 4+i 可存在磁盘 i 上。另外，分散不必在字节的位级上进行：例如，对于块级分散，一个文件的块可分散在多个磁盘上；对于 n 个磁盘，一个文件的块 i 可存在磁盘（i mod n）+1。其他分散级别，如扇区字节和块的扇区也是可能的。块级分散是最常见的。

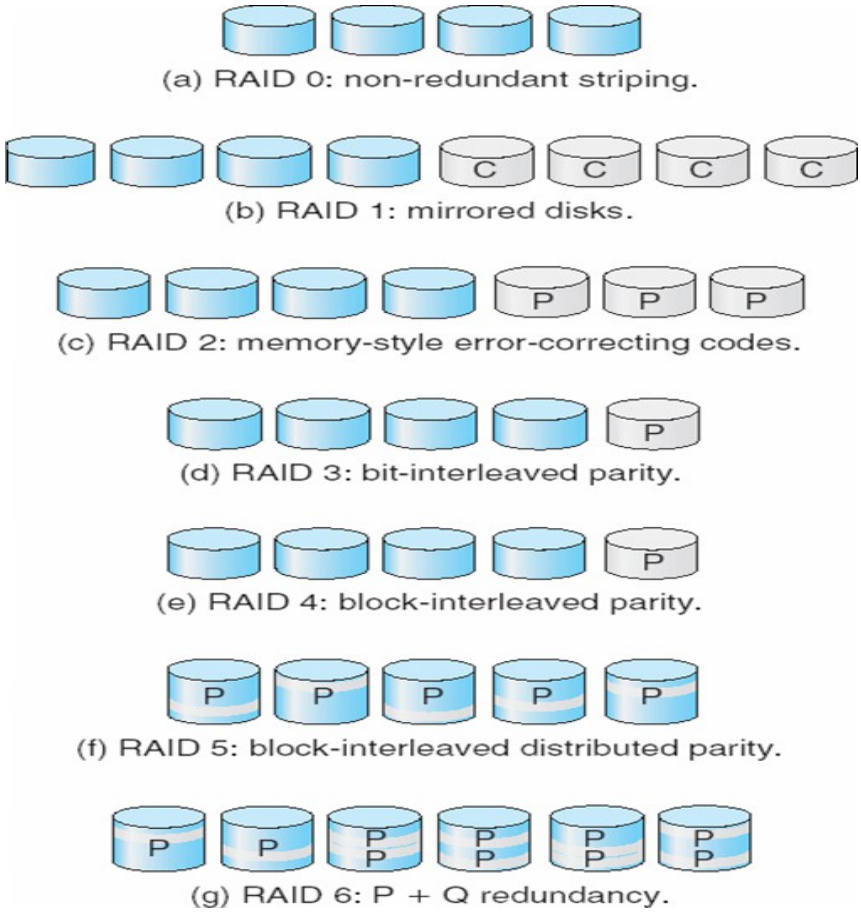
总之，磁盘系统并行访问有两个主要目的：

① 通过负荷平衡，增加了多个小访问（即页访问）的吞吐量。

②降低大访问的响应时间。

RAID 级别

镜像提供高可靠性，但昂贵；分散提供了高数据传输率，但并未改善可靠性。通过磁盘分散和“奇偶”位可以提供多种方案以在低代价环境下提供冗余。这些方案有不同的性价折中，可分成不同级别，称为 RAID 级别。这里讨论各种级别。图 8.8 描述了这些结构（图 8.8 中，P 表示差错纠正位，C 表示数据的第二副本）。在图 8.8 中所描述的各种情况中，4 个磁盘用于存储数据，其他磁盘用于存储冗余信息以便从差错中恢复。



8.8 RAID 的级别

RAID 级别 0：RAID 级别 O 指按块级别分散的磁盘阵列，但没有冗余（如镜像或奇偶位）。图 8.8(a)显示了大小为 4 的磁盘阵列。

RAID 级别 1：RAID 级别 l 指磁盘镜像。图 8.8(b)表示了一个镜像组织。

RAID 级别 2：RAID 级别 2 也称为内存方式的差错纠正代码结构。内存系统一直实现了基于奇偶位的错误检测。内存系统的每个字节都有一个相关奇偶位，以记录字节中置为 l 的个数是偶数(parity =O)或是奇数（parity=1）。如果字节的 1 个位损坏（或是 l 变成 0，或是 O 变成 1），那么字节的奇偶也将改变，因此与所存储的奇偶位就不匹配。类似地，如果所存储的奇偶位损坏了，那么它就与所计算的奇偶位不匹配。因此，单个位差错可为内存系统所检测。差错纠正方案存储两个或多个额外位，当单个位出错时可用来重新构造数据。ECC 的思想可直接用于将字节分散在磁盘上的磁盘阵列。注意，对于 4 个磁盘的数据，RAID 级别 2 只用了三个额外磁盘，而 RAID 级别 1 则需要用 4 个额外磁盘。

RAID 级别 3：RAID 级别 3 或基于**位交织奇偶结构**对级别 2 做了改进。与内存系统不同，磁盘控制器能检测到一个扇区是否正确读取，这样单个奇偶位就可用于差错检测和差错纠正。这种方案如下：如果一个扇区损坏，那么知道是哪个扇区，通过计算其他磁盘扇区相应位的奇偶值可得出所损坏位是 l 还是 O。如果其他位的奇偶值等于存储奇偶值，那么缺少位为 O；否则，就为 l。RAID 级别 3 与级别 2-样好，但在额外磁盘数量方面要更便宜（它只有一个额外磁盘），这样级别 2 在实际中并不使用。这种方案如图 8.8(d)所示。

RAID 级别 4：RAID 级别 4 或块交织奇偶结构采用与 RAID O 一样的块级分散，另外在一独立磁盘上保存其它 N 个磁盘相应块的奇偶块。这种方案如图 8.8(e)所示。如果一个磁盘出错，那么奇偶块可以与其他磁盘的相应块一起用于恢复出错磁盘的块。

读块只访问一个磁盘，可以允许其他磁盘处理请求。因此，每个访问的数据传输速度较慢，但是多个读访问可以并行处理，产生了更高的总的 I/O 速度。大量读的数据传输速度高，这是因为所有磁盘可以并行读：大量数据的写操作传输速度也高，这是因为数据和奇偶可以并行写。

RAID 级别 5：RAID 级别 5 或块交织分布奇偶结构。不同于级别 4，它是将数据和奇偶分布在所有 N+1 块磁盘上，而不是将数据存在 N 个磁盘上而奇偶存在单个磁盘上。对于每一块，一个磁盘存储奇偶，而其他的存储数据。例如，对于 5 个磁盘的阵列，第 n 块的奇偶保存在磁盘(n mod 5)+1 上；其他 4 个磁盘的第 n 块保存该奇偶块对应的真正数据。这种方案如图 8.8(f)所示，其中 P 分布在所有磁盘上。奇偶块不能保存在同一磁盘上，这是因为一个磁盘出错会导致所有数据及奇偶丢失，因而无法恢复。通过将奇偶分布在所有磁盘上，RAID5 避免了 RAID4 方案的对单个奇偶磁盘的过度使用。RAID5 是最常见的奇偶校验 RAID 系统。

RAID 级别 6：RAID 级别 6，也称为 P+Q 冗余方案，与 RAID 级别 5 很类似，但是保存了额外冗余信息以防止多个磁盘出错。不是使用奇偶校验，而是使用了差错纠正码如 Read-Solomon 码。在如图 8.8(g)所示的方案中，每 4 个位的数据使用了 2 个位的冗余数据，而不是像级别 5 那样的一个奇偶位，这样系统可以容忍两个磁盘出错。

RAID 级别 0+1: RAID 级别 0+1 指 RAID 级别 0 和级别 l 的组合。 RAIDO 提供了性能，而 RAID l 提供了可靠性。通常，它比 RAID 5 有更好的性能。它适用于对性能和可靠性都要求高的环境。然而，这增加了用于存储的磁盘数量，所以也更为昂贵。对 RAID 0+l，一组磁盘被分散成条，每一条再镜像到另一条。

另一现在正商业化的 RAID 选择是 RAID l+0，即磁盘先镜像，再分散。这种 RAID 与 RAID 0+1 相比有一些理论上的优点。例如，如果 RAID 0+1 中的一个磁盘出错，那

么整个条就不能访问，虽然所有其他条可用。对于 RAID l+O，如果单个磁盘不可用，但其镜像仍如其他磁盘一样可用(见图 8.9)。

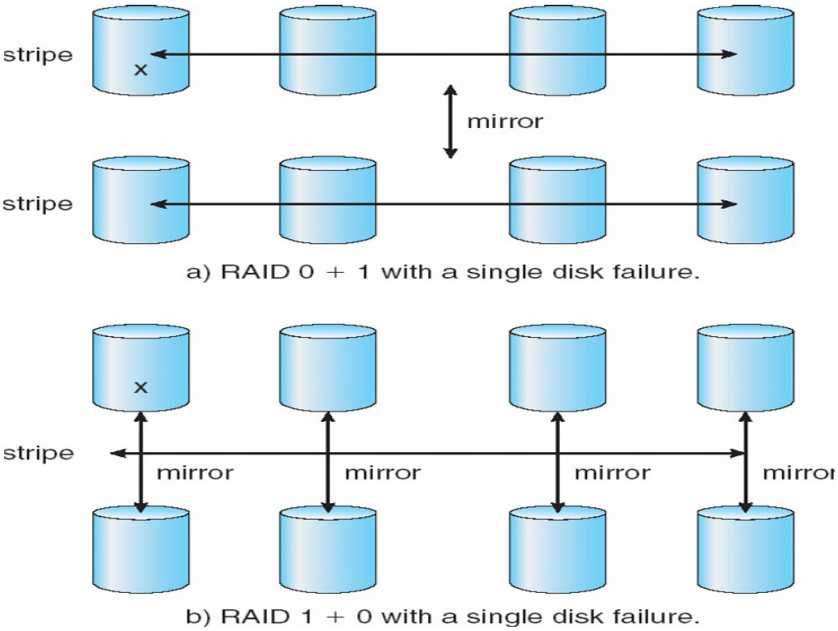


图 8.9 RAID 0+1 和 1+0