输入输出系统——完成计算机与外界信息交换或给计算机提供大容量外部存储器

1.输入/输出系统的性能

- 系统的响应时间
  - 从用户发出请求到系统作出响应的时间
- **可靠性**
  - 系统从某个初始参考点开始一直连续提供服务的能力
  - 衡量标准——**平均失效前时间**（Mean Time to Failure，MTTF）
  - MTTF的倒数就是系统的失效率
  - 中断服务的时间用**平均修复时间**（Mean Time to Repair，MTTR）
- **可用性**

$$\text{可用性}=\frac{\text{MTTF}}{\text{MTTF}+\text{MTTR}}$$

  - 系统正常工作的时间在连续两次正常服务间隔时间中所占的比率
  - 其中MTTF+MTTR可以用**平均失效间隔时间**（Mean Time Between Falirue）
- **可信性**
  - 服务的质量——在多大程度上可以认为服务是可靠的
  - 不可衡量

例 6.1 假设磁盘子系统的组成部件和它们的 MTTF 如下：

(1) 磁盘子系统由 10 个磁盘构成，每个磁盘的 MTTF 为 1 000 000 小时。

(2) 1 个 SCSI 控制器，其 MTTF 为 500 000 小时。

(3) 1 个不间断电源，其 MTTF 为 200 000 小时。

(4) 1 个风扇，其 MTTF 为 200 000 小时。

(5) 1 根 SCSI 连线，其 MTTF 为 1 000 000 小时。

假定每个部件的生存期服从指数分布，同时假定各部件的故障是相互独立的，求整个系统的 MTTF。

解 整个系统的失效率为

$$\text{系统失效率}=10\times\frac{1}{1\,000\,000}+\frac{1}{500\,000}+\frac{1}{200\,000}+\frac{1}{200\,000}+\frac{1}{1\,000\,000}=\frac{23}{1\,000\,000}$$

系统的 MTTF 为系统失效率的倒数，即

$$\text{MTTF}=\frac{1\,000\,000}{23}=43\,500\ \text{小时}$$

- 提高系统组成部件可靠性的方法
  - 有效构建方法——在构建系统的过程中消除故障隐患，这样建立起来的系统就不会出现故障
  - 纠错方法——纠错方法是指在系统构建中采用容错的方法，即使出现故障，也可以通过容错信息保证系统正常工作

2.廉价磁盘冗余阵列

- **磁盘阵列**
  - 使用多个磁盘来代替大容量磁盘存储系统
  - 好处：大容量、并行可以提高性能
  - **交叉存放**：以条带为单位将数据均匀分布到多个磁盘上，因而可以并行地处理多个数据读/写请求，从而提高I/O性能
    - 多个独立的请求可以由多个磁盘来并行地处理，减少了I/O请求排队等待时间
    - 一个请求如果是访问多个块，也可以由多个磁盘合作并行处理，提高了单个请求的数据传输率
  - **磁盘数量越多，性能提高越多，可靠性下降**——可靠性将为单个磁盘的1/N
  - 可以在磁盘阵列中加设**冗余信息盘**来解决此问题——当某个磁盘出问题的时候可以利用冗余盘中的

信息重新构建，只有两个磁盘都坏了的时候磁盘阵列才不能工作，而修复时间很短，所以可靠性比单个磁盘高很多 -> 廉价磁盘冗余
    - 阵列（Redundant Array of Inexoensive Disks，RAID）
    - 不同磁盘阵列的区分特征——数据交叉存放的粒度、荣誉数据的计算方法及在磁盘阵列中的存放方式
- 细粒度磁盘阵列
    - 把数据分割成较小的单位交叉存放
    - 优点：几乎所有I/O请求都能获得很高的数据传输率
    - 缺点：只有一个逻辑上的I/O在处理当中，每个磁盘都会为每一个请求进行定位而浪费时间
- 粗粒度磁盘阵列
    - 数据以较大的单位交叉存放
    - 优点：规模较小的数据访问只需要访问较少的几个磁盘，只有较大规模的访问才需要访问所有盘，多个较小规模的请求可以同时得到处理，而较大的数据又传输效率挺高
- 奇偶校验码计算冗余信息（也有汉明码或Reed-Solomon码）
- 存放冗余信息的方式——集中存放在少数的几个盘中、冗余信息均匀存放在所有盘中（避免热点问题）
- 减少MTTR的方法——热备份盘（热切换技术）——被备份的盘坏了，直接用备份盘，不拷贝数据，换上来新的盘再做备份盘
- **RAID 0**
    - 实际上是非冗余磁盘阵列，严格来说并不属于RAID
    - 直接把几个磁盘连接在一起成一个更大的盘
    - A、B、C、D构成条带，大小为**条带宽度**
    - 优点：性能高
    - 缺点：无法恢复

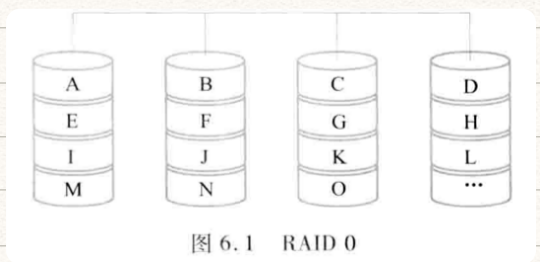

图 6.1 RAID 0

- **RAID 1**
    - 镜像磁盘——每一个盘都有冗余备份
    - 写：两个盘都写，不校验
    - 读：同时读，哪个块要哪个
    - 故障：另一个直接上，不进行数据重建，但是尽快换掉坏的，否则这个再坏了就不行了
    - 优点：实现简单
    - 缺点：成本最高

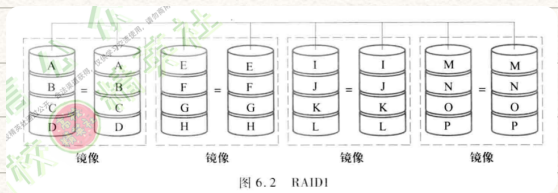

图 6.2 RAID1

- **RAID 2**
    - 存储器式磁盘阵列
    - 冗余磁盘存汉明纠错码
    - 不好，太极端，没有商业化产品
- **RAID 3**
    - 位交叉奇偶校验磁盘阵列
    - 读出数据的时候如果发现某个盘坏了，可以通过该磁盘之外的所有盘中的正确信息恢复故障盘中的数据
    - 优点：

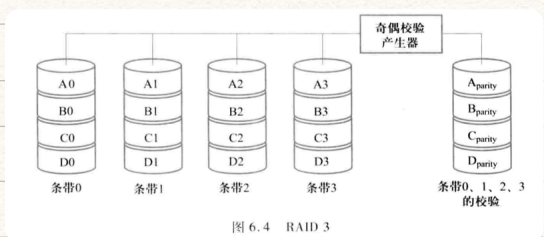

图 6.4 RAID 3

- 空间开销小，不管多少个磁盘都只需要一个校验盘
- 细粒度，不管什么数据都要所有盘为之服务，传输率高，对大数据量的读写有很大优势

- 缺点：
  - 不能同时进行多个I/O请求的处理

- **RAID 4**
  - 块交叉奇偶校验磁盘阵列——粗粒度
  - 写操作👇

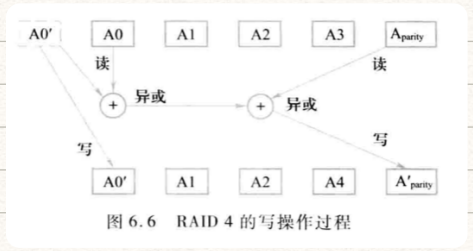

图 6.6 RAID 4 的写操作过程

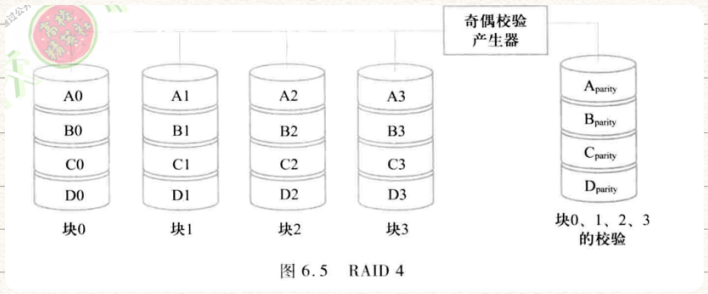

图 6.5 RAID 4

  - 优点：有效处理小规模访问，也能快速处理大规模访问，空间开销小
  - 缺点：控制开销大

- **RAID 5**
  - 块交叉分布奇偶校验磁盘阵列——解决校验盘瓶颈问题
  - 优点：更快得处理小规模写操作
  - 缺点：控制器是经典RAID（1~5）中最复杂的

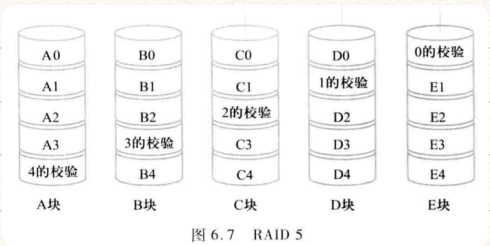

图 6.7 RAID 5

- **RAID 6**
  - P+Q双校验磁盘阵列——能容纳两个盘出错
  - 在RAID 5的基础上又加了一个校验信息，放在另一个盘中
  - 优点：可靠性更强，适合于保存重要数据
  - 缺点：空间开销是RAID 5的两倍

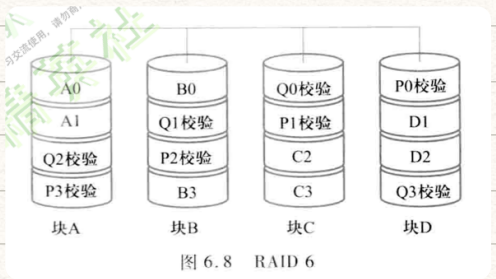

图 6.8 RAID 6

- **RAID 10**
  - RAID 1+0
  - 先镜像后条带存放

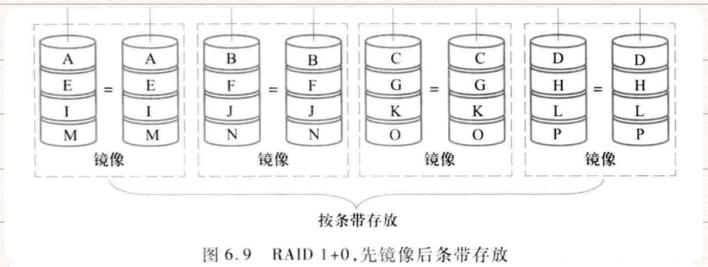

图 6.9 RAID 1+0，先镜像后条带存放

- **RAID 01**
  - RAID 0+1
  - 先条带存放后镜像

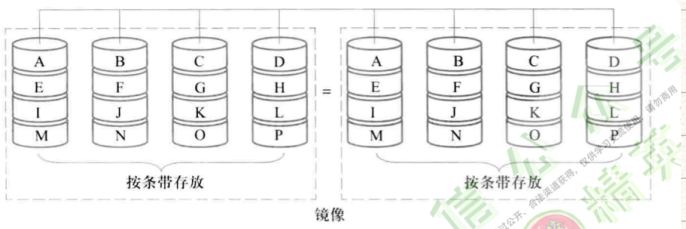

图 6.10 RAID 0+1，先条带存放后镜像

表 6.1　RAID 的分级及其特性

RAID 级别	可以容忍的故障个数以及当数据盘为 8 个时，所需要的检测盘的个数	优点	缺点	公司产品
0 非冗余，条带存放	0 个故障； 0 个检测盘	没有空间开销	没有纠错能力	广泛应用
1 镜像	1 个故障； 8 个检测盘	不需要计算奇偶校验，数据恢复快，读数据快。而且其小规模写操作比更高级别的 RAID 快	检测空间开销最大（即需要的检测盘最多）	EMC，HP（Tandem），IBM
RAID 级别	可以容忍的故障个数以及当数据盘为 8 个时，所需要的检测盘的个数	优点	缺点	公司产品
2 存储器式 ECC	1 个故障； 4 个检测盘	不依靠故障盘进行自诊断	检测空间开销的级别是 $\log_2 m$ 级（m 为数据盘的个数	没有
3 位交叉奇偶校验	1 个故障； 1 个检测盘	检测空间开销小（即需要的检测盘少），大规模读写操作的带宽高	对小规模、随机的读写操作没有提供专门的支持	外存概念
4 块交叉奇偶校验	1 个故障； 1 个检测盘	检测空间开销小，小规模的读操作带宽更高	校验盘是小规模写的瓶颈	网络设备
5 块交叉分布奇偶校验	1 个故障； 1 个检测盘	检测空间开销小，小规模的读写操作带宽更高	小规模写操作需要访问磁盘 4 次	广泛应用
6 P+Q 双奇偶校验	2 个故障； 2 个检测盘	具有容忍 2 个故障的能力	小规模写操作需要访问磁盘 6 次，检测空间开销加倍（与 RAID 3、4、5 比较）	网络设备

通道处理机

- **定义**：专门负责整个计算机系统的输入输出工作的专用处理机
- **4级层次结构的输入输出系统**：cpu、通道、设备控制器、外设
- 通道的功能

    - 接收CPU发来的I/O指令，根据指令选择制定的外设与通道相连
    - 执行通道程序 -> 从主存中逐条取出通道指令、译码、执行操作
    - 给出外设中要进行读/写操作的数据的所在地址
    - 给出主存缓冲区的首地址
    - 控制外设与主存缓冲区之间的数据传送长度
    - 指定传送工作结束时要进行的操作
    - 检查外设的工作状态是否正常，并将该状态信息送往主存指定单元保存
    - 数据传输过程中完成必要的格式转换
- 主要构成硬件
    - 寄存器：数据缓冲寄存器、主地址寄存器、传输字节计数器、通道命令寄存器、通道状态寄存器
    - 控制逻辑：分时控制、地址分配、数据传送、数据装配和拆分
- **工作过程**：
    - 用户程序用访管指令进入管理程序，尤其编制一个通道程序、开启通道
    - 通道处理机执行通道程序，完成指定的数据输入/输出工作
    - 通道结束后向CPU发送中断请求，CPU再次进入管态
- **通道的种类：**
    - 字节多路通道
        - 简单的共享通道
        - 用于连接**多台**低速或中速设备
        - 以字节为宽度进行输入输出
        - 相邻两次传送之间有较长时间的等待
    - 选择通道
        - 磁盘存储器需要高速外设
        - 多台高速设备请求传送数据时，选择通道按照一定规则选择要服务的设备
        - 一旦选中就进入“忙”状态**直到所有数据传输完**
        - 硬件包括5个寄存器、格式变换部件及通道控制部件
        - 数据缓冲寄存器、设备地址寄存器、主存地址计数器、交换字节数计数器、设备状态/控制寄存器
    - **数组多路通道**
        - 前两者的结合产物
        - **以数据块为单位、分时轮流为多台高速设备提供服务**
        - 适用于高速设备但不像选择通道那样一次把所选设备的全部数据都传送完，而是传送完固定长度的数据块之后，就重新选择别的设备
        - 为什么要分时
            - 因为在读数据的时候，磁盘的传输速率虽然很高，但是辅助操作时间很长，需要磁头定位（寻道时间）、找扇区（等待时间）然后才能读出数据
            - 寻道时间+等待时间=寻址时间
            - 如果通道只为某一个设备服务，那么在其寻址的过程中，通道要一直等待，非常浪费
        - 所以数组多路通道在工作的时候，是发出定位命令 -> 断开，让外设先执行定位工作去 -> 找到

之后再连上，发出找扇区的命令 -> 再断开 -> 找到后再连上传输数据

- **通道流量分析**
  - 定义：一个通道在数据传送期间，单位时间内能传送的数据量
  - 单位：Bps（字节/秒）
  - 也称通道吞吐率，通道数据传输率
  - 通道最大流量——通道在满负荷工作状态下的流量
  - 字节多路通道
    - 总时间 $T=(T_S+T_D)*p*n$
    - 最大流量 $f_{MAX\text{-}BYTE}=\dfrac{P\cdot n}{(T_S+T_D)Pn}=\dfrac{1}{T_S+T_D}$
    - 实际流量 $f_{BYTE}=\sum_{i=1}^{p} f_i$

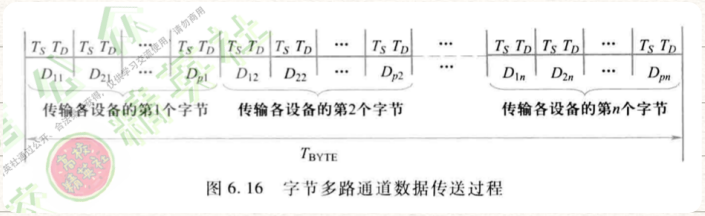

图 6.16　字节多路通道数据传送过程

  - 选择通道
    - 一段时间内只能单独为一台高速外设服务
    - 逐个为物理上连接的几台高速外设传输数据
    - 总时间 $T=pT_S+pnT_D$
    - 最大流量 $T_{MAX\text{-}SELECT}=\dfrac{pn}{pT_S+pnT_D}=\dfrac{1}{\frac{T_S}{n}+T_D}$
    - 实际流量 所有设备中最大的一个

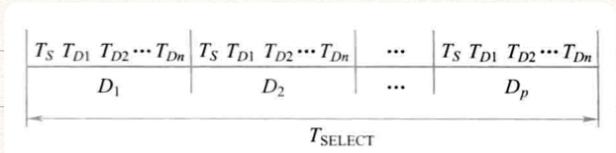

图 6.17　选择通道数据传送过程

  - 数组多路通道
    - 总时间 $T=p\frac{n}{K}T_S+pnT_D$
    - 最大流量 $T_{MAX\text{-}BLOCK}=\dfrac{pn}{p\frac{n}{K}T_S+pnT_D}=\dfrac{1}{\frac{T_S}{K}+T_D}$
    - 实际流量 所有设备中最大的一个

I/O与操作系统