理第二个请求。控制器的责任是分解请求，并且以正确的顺序将适当的命令提供给适当的磁盘，之后还要在内存中将结果正确地装配起来。0级RAID的性能是杰出的而实现是简单明了的。

对于习惯于每次请求一个扇区的操作系统，0级RAID工作性能最为糟糕。虽然结果会是正确的，但是却不存在并行性，因此也就没有增进性能。这一结构的另一个劣势是其可靠性潜在地比SLED还要差。如果一个RAID由四块磁盘组成，每块磁盘的平均故障间隔时间是20 000小时，那么每隔5000小时就会有一个驱动器出现故障并且所有数据将完全丢失。与之相比，平均故障间隔时间为20 000小时的SLED的可靠性要高出四倍。由于在这一设计中未引入冗余，实际上它还不是真正的RAID。

下一个选择——1级RAID如图5-20b所示，这是一个真正的RAID。它复制了所有的磁盘，所以存在四个主磁盘和四个备份磁盘。在执行一次写操作时，每个条带都被写了两次。在执行一次读操作时，则可以使用其中的任意一个副本，从而将负荷分布在更多的驱动器上。因此，写性能并不比单个驱动器好，但是读性能能够比单个驱动器高出两倍。容错性是突出的：如果一个驱动器崩溃了，只要用副本来替代就可以了。恢复也十分简单，只要安装一个新驱动器并且将整个备份驱动器复制到其上就可以了。

0级RAID和1级RAID操作的是扇区条带，与此不同，2级RAID工作在字的基础上，甚至可能是字节的基础上。想象一下将单个虚拟磁盘的每个字节分割成4位的半字节对，然后对每个半字节加入一个汉明码从而形成7位的字，其中1、2、4位为奇偶校验位。进一步想象如图5-20c所示的7个驱动器在磁盘臂位置与旋转位置方面是同步的。那么，将7位汉明编码的字写到7个驱动器上，每个驱动器写一位，这样做是可行的。

Thinking Machine公司的CM-2计算机采用了这一方案，它采用32位数据字并加入6个奇偶校验位形成一个38位的汉明字，再加上一个额外的位用于汉明字的奇偶校验，并且将每个字分布在39个磁盘驱动器上。因为在一个扇区时间里可以写32个扇区的数据，所以总的吞吐量是巨大的。此外，一个驱动器的损坏不会引起问题，因为损坏一个驱动器等同于在每个39位字的读操作中损失一位，而这是汉明码可以轻松处理的事情。

不利的一面是，这一方案要求所有驱动器的旋转必须同步，并且只有在驱动器数量很充裕的情况下才有意义（即使对于32个数据驱动器和6个奇偶驱动器而言，也存在19%的开销）。这一方案还对控制器提出许多要求，因为它必须在每个位时间里求汉明校验和。

3级RAID是2级RAID的简化版本，如图5-20d所示。其中要为每个数据字计算一个奇偶校验位并且将其写入一个奇偶驱动器中。与2级RAID一样，各个驱动器必须精确地同步，因为每个数据字分布在多个驱动器上。

乍一想，似乎单个奇偶校验位只能检测错误，而不能纠正错误。对于随机的未知错误的情形，这样的看法是正确的。然而，对于驱动器崩溃这样的情形，由于坏位的位置是已知的，所以这样做完全能够纠正1位错误。如果一个驱动器崩溃了，控制器只需假装该驱动器的所有位为0，如果一个字有奇偶错误，那么来自废弃了的驱动器上的位原来一定是1，这样就纠正了错误。尽管2级RAID和3级RAID两者都提供了非常高的数据率，但是每秒钟它们能够处理的单独的I/O请求的数目并不比单个驱动器好。

4级RAID和5级RAID再次使用条带，而不是具有奇偶校验的单个字。如图5-20e所示，4级RAID与0级RAID相类似，但是它将条带对条带的奇偶条带写到一个额外的磁盘上。例如，如果每个条带k字节长，那么所有的条带进行异或操作，就得到一个k字节长的奇偶条带。如果一个驱动器崩溃了，则损失的字节可以通过读出整个驱动器组从奇偶驱动器重新计算出来。

这一设计对一个驱动器的损失提供了保护，但是对于微小的更新其性能很差。如果一个扇区被修改了，那么就必须读取所有的驱动器以便重新计算奇偶校验，然后还必须重写奇偶校验。作为另一选择，它也可以读取旧的用户数据和旧的奇偶校验数据，并且用它们重新计算新的奇偶校验。即使是对于这样的优化，微小的更新也还是需要两次读和两次写。

结果，奇偶驱动器的负担十分沉重，它可能会成为一个瓶颈。通过以循环方式在所有驱动器上均匀地分布奇偶校验位，5级RAID消除了这一瓶颈，如图5-20f所示。然而，如果一个驱动器发生崩溃，重新构造故障驱动器的内容是一个非常复杂的过程。