下述三种操作：

1) 稳定写（stable write）。稳定写首先将块写到驱动器1上，然后将其读回以校验写的是正确的。如果写的不正确，那么就再次做写和重读操作，一直到n次，直到正常为止。经过n次连续的失败之后，就将该块重映射到一个备用块上，并且重复写和重读操作直到成功为止，无论要尝试多少个备用块。在对驱动器1的写成功之后，对驱动器2上对应的块进行写和重读，如果需要的话就重复这样的操作，直到最后成功为止。如果不存在CPU崩溃，那么当稳定写完成后，块就正确地被写到两个驱动器上，并且在两个驱动器上得到校验。

2) 稳定读（stable read）。稳定读首先从驱动器1上读取块。如果这一操作产生错误的ECC，则再次尝试读操作，一直到n次。如果所有这些操作都给出错误的ECC，则从驱动器2上读取对应的数据块。给定一个成功的稳定写为数据块留下两个可靠的副本这样的事实，并且我们假设在合理的时间间隔内相同的块在两个驱动器上自发地变坏的概率可以忽略不计，那么稳定读就总是成功的。

3) 崩溃恢复（crash recovery）。崩溃之后，恢复程序扫描两个磁盘，比较对应的块。如果一对块都是好的并且是相同的，就什么都不做。如果其中一个具有ECC错误，那么坏块就用对应的好块来覆盖。如果一对块都是好的但是不相同，那么就将驱动器1上的块写到驱动器2上。

如果不存在CPU崩溃，那么这一方法总是可行的，因为稳定写总是对每个块写下两个有效的副本，并且假设自发的错误决不会在相同的时刻发生在两个对应的块上。如果在稳定写期间出现CPU崩溃会怎么样？这就取决于崩溃发生的精确时间。有5种可能性，如图5-31所示。

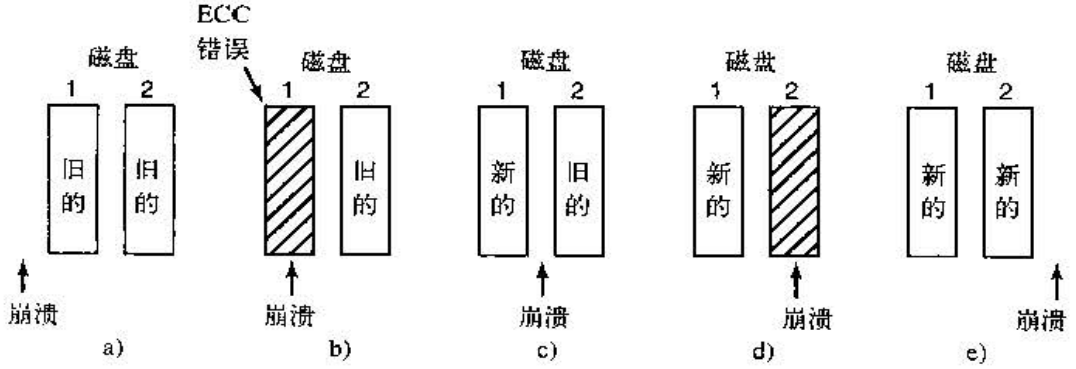

图5-31 崩溃对于稳定写的影响的分析

在图5-31a中，CPU崩溃发生在写块的两个副本之前。在恢复的时候，什么都不用修改而旧的值将继续存在，这是允许的。

在图5-31b中，CPU崩溃发生在写驱动器1期间，破坏了该块的内容。然而恢复程序能够检测出这一错误，并且从驱动器2恢复驱动器1上的块。因此，这一崩溃的影响被消除并且旧的状态完全被恢复。

在图5-31c中，CPU崩溃发生在写完驱动器1之后但是还没有写驱动器2之前。此时已经过了无法复原的时刻：恢复程序将块从驱动器1复制到驱动器2上。写是成功的。

图5-31d与图5-31b相类似：在恢复期间用好的块覆盖坏的块。不同的是，两个块的最终取值都是新的。

最后，在图5-31e中，恢复程序看到两个块是相同的，所以什么都不用修改并且在此处写也是成功的。

对于这一模式进行各种各样的优化和改进都是可能的。首先，在崩溃之后对所有的块两个两个地进行比较是可行的，但是代价高昂。一个巨大的改进是在稳定写期间跟踪被写的是哪个块，这样在恢复的时候必须被检验的块就只有一个。某些计算机拥有少量的非易失性RAM（nonvolatile RAM），它是一个特殊的CMOS存储器，由锂电池供电。这样的电池能够维持很多年，甚至有可能是计算机的整个生命周期。与主存不同（它在崩溃之后就丢失了），非易失性RAM在崩溃之后并不丢失。每天的时间通常就保存在这里（并且通过一个特殊的电路进行增值），这就是为什么计算机即使在拔掉电源之后仍然知道是什么时间。

假设非易失性RAM的几个字节可供操作系统使用，稳定写就可以在开始写之前将准备要更新的块的编号放到非易失性RAM里。在成功地完成稳定写之后，在非易失性RAM中的块编号用一个无效的块编号（例如-1）覆盖掉。在这些情形下，崩溃之后恢复程序可以检验非易失性RAM以了解在崩溃期间