第二个问题独立于前一个问题。如果关系到文件系统一致性（除数据块之外，其他块基本上都是这样）的某块被修改，都应立即将该块写回磁盘，不管它是否被放在LRU链表尾部。将关键块快速写回磁盘，将大大减少在计算机崩溃后文件系统被破坏的可能性。用户的文件崩溃了，该用户会不高兴，但是如果整个文件系统都丢失了，那么这个用户会更生气。

尽管用这类方法可以保证文件系统一致性不受到破坏，但我们仍然不希望数据块在高速缓存中放很久之后才写入磁盘。设想某人正在用个人计算机编写一本书。尽管作者让编辑程序将正在编辑的文件定期写回磁盘，所有的内容只存在高速缓存中而不在磁盘上的可能性仍然非常大。如果这时系统崩溃，文件系统的结构并不会被破坏，但他一整天的工作就会丢失。

即使只发生几次这类情况，也会让人感到不愉快。系统采用两种方法解决这一问题。在UNIX系统中有一个系统调用sync，它强制性地把全部修改过的块立即写回磁盘。系统启动时，在后台运行一个通常名为update的程序，它在无限循环中不断执行sync调用，每两次调用之间休眠30s。于是，系统即使崩溃，也不会丢失超过30秒的工作。

虽然目前Windows有一个等价于sync的系统调用——FlushFileBuffers，不过过去没有。相反，Windows采用一个在某种程度上比UNIX方式更好（有时更坏）的策略。其做法是，只要被写进高速缓存，就把每个被修改的块写进磁盘。将缓存中所有被修改的块立即写回磁盘称为通写高速缓存（write-through cache）。同非通写高速缓存相比，通写高速缓存需要更多的磁盘I/O。

若某程序要写满1KB的块，每次写一个字符，这时可以看到这两种方法的区别。UNIX在高速缓存中保存全部字符，并把这个块每30秒写回磁盘一次，或者当从高速缓存删除这一块时，写回磁盘。在通写高速缓存里，每写入一字符就要访问一次磁盘。当然，多数程序有内部缓冲，通常情况下，在每次执行write系统调用时并不是只写入一个字符，而是写入一行或更大的单位。

采用这两种不同的高速缓存策略的结果是：在UNIX系统中，若不调用sync就移动（软）磁盘，往往会导致数据丢失，在被毁坏的文件系统中也经常如此。而在通写高速缓存中，就不会出现这类情况。选择不同策略的原因是，在UNIX开发环境中，全部磁盘都是硬盘，不可移动。而第一代Windows文件源自MS-DOS，是从软盘世界中发展起来的。由于UNIX方案有更高的效率它成为当然的选择（但可靠性更差），随着硬盘成为标准，它目前也用在Windows的磁盘上。但是，NTFS使用其他方法（日志）改善其可靠性，这在前面已经讨论过。

一些操作系统将高速缓存与页缓存集成。这种方式特别是在支持内存映射文件的时候很吸引人。如果一个文件被映射到内存上，则它其中的一些页就会在内存中，因为它们被要求按页进入。这些页面与在高速缓存中的文件块几乎没有不同。在这种情况下，它们能被以同样的方式来对待，也就是说，用一个缓存来同时存储文件块与页。

2. 块提前读

第二个明显提高文件系统性能的技术是：在需要用到块之前，试图提前将其写入高速缓存，从而提高命中率。特别地，许多文件都是顺序读的。如果请求文件系统在某个文件中生成块k，文件系统执行相关操作且在完成之后，会在用户不察觉的情形下检查高速缓存，以便确定块$k+1$是否已经在高速缓存。如果还不在，文件系统会为块$k+1$安排一个预读，因为文件系统希望在需要用到该块时，它已经在高速缓存或者至少马上就要在高速缓存中了。

当然，块提前读策略只适用于顺序读取的文件。对随机存取文件，提前读丝毫不起作用。相反，它还会帮倒忙，因为读取无用的块以及从高速缓存中删除潜在有用的块将会占用固定的磁盘带宽（如果有“脏”块的话，还需要将它们写回磁盘，这就占用了更多的磁盘带宽）。那么提前读策略是否值得采用呢？文件系统通过跟踪每一个打开文件的访问方式来确定这一点。例如，可以使用与文件相关联的某个位协助跟踪该文件到底是“顺序存取方式”还是“随机存取方式”。在最初不能确定文件属于哪种存取方式时，先将该位设置成顺序存取方式。但是，查找一完成，就将该位清除。如果再次发生顺序读取，就再次设置该位。这样，文件系统可以通过合理的猜测，确定是否应该采取提前读的策略。即便弄错了一次也不会产生严重后果，不过是浪费一小段磁盘的带宽罢了。

3. 减少磁盘臂运动

高速缓存和块提前读并不是提高文件系统性能的惟一方法。另一种重要技术是把有可能顺序存取的