页面也如（1）一样移动到后备链表或已修改链表。

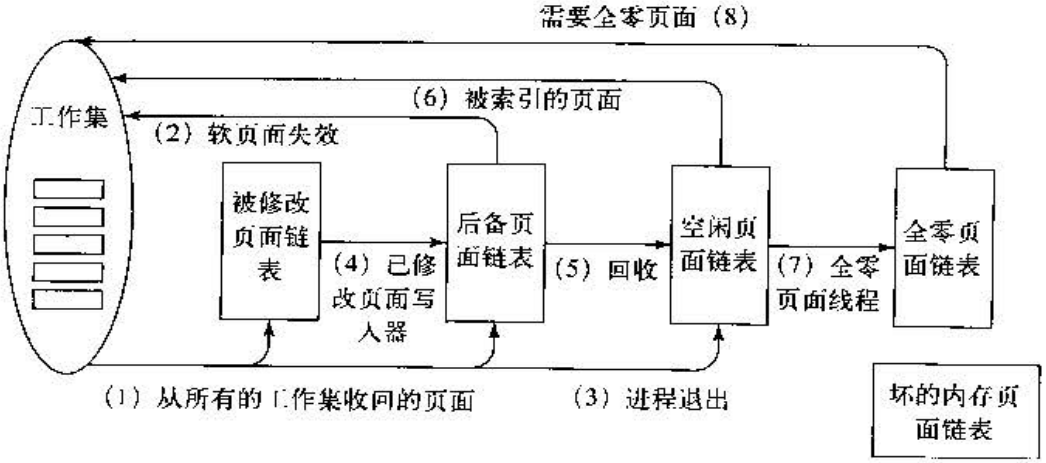

图11-36 不同的页面链表以及它们之间的转变

两个系统线程——映射页面写入器（mapped page writer）和已修改页面写入器（modified page writer），周期性地被唤醒来检查是否系统中有足够的干净页面。如果没有，这两个线程从已修改链表的顶部取出页面，写回到磁盘，然后将这些页面插入后备链表（4）。前者处理对于映射文件的写，而后者处理页面文件的写。这些写的结果就是将已修改（脏）页面移到后备（干净）链表中。

之所以使用两个线程是因为映射文件可能会因为写的结果增长，而增长的结果就需要对磁盘上的数据结构具有相应的权限来分配空闲磁盘块。当一个页面被写入时如果没有足够的内存，就会导致死锁。另一个线程则是解决向页面文件写入页时的问题。

下面说明图11-36中另一个转换。如果进程解除页映射，该页不再和进程相关从而进入空闲链表（5），当该页是共享的时候例外。当页面失效会请求一个页框给将要读入的页，此时该页框会尽可能从空闲链表中取下（6）。由于该页会被全部重写，因此即使有机密的信息也没有关系。

栈的增长则是另一种情况。这种情况下，需要一个空的页框，同时安全规则要求该页全零。由于这个原因，另一个称为零页面线程（ZeroPage thread）的低优先级内核线程（参见图11-28）将空闲链表中的页面写全零并将页面放入全零页链表（7）。全零页面很可能比空闲页面更加有用，因此只要当CPU空闲且有空闲页面，零页面线程就会将这些页面全部写零，而在CPU空闲的时候进行这一操作也是不增加开销的。

所有这些链表的存在导致了一些微妙的策略抉择。例如，假设要从磁盘载入一个页面，但是空闲链表是空的，那么，要么从后备链表中取出一个干净页（虽然这样做稍后有可能导致缺页），要么从全零页面链表中取出一个空页（忽略把该页清零的代价），系统必须在上述两种策略之间做出选择。哪一个更好呢？

内存管理器必须决定系统线程把页面从已修改链表移动到后备链表的积极程度。有干净的页面后备总比有脏页后备好得多（因为如有需要，干净的页可以立即重用），但是一个积极的净化策略意味着更多的磁盘I/O，同时一个刚刚净化的页面可能由于缺页中断重新回到工作集中，然后又成为脏页。通常来讲，Windows通过算法、启发、猜测、历史、经验以及管理员可控参数的配置来做权衡。

总而言之，内存管理需要一个拥有多种数据结构、算法和启发性的十分复杂、重要的构件。它尽可能地自我调整，但是仍然留有很多选项使系统管理员可以通过配置这些选项来影响系统性能。大部分的选项和计数器可以通过工具浏览，相关的各种工具包在前面都有提到。也许在这里最值得记住的就是，在真实的系统里，内存管理不仅仅是一个简单的时钟或老化的页面算法。

11.6 Windows Vista的高速缓存

Windows高速缓存（cache）通过把最近和经常使用的文件片段保存在内存中的方式来提升文件系统的性能。高速缓存管理器管理的是虚拟寻址的数据块，也就是文件片段，而不是物理寻址的磁盘块。这种方法非常适合NTFS文件系统，如11.8节所示。NTFS把所有的数据作为文件来存储，包括文件系统的元数据。