虚拟内存是在 20 世纪 60 年代早期发明的，远在 CPU-内存之间差距的加大引发产生 SRAM 缓存之前。因此，虚拟内存系统使用了和 SRAM 缓存不同的术语，即使它们的许多概念是相似的。在虚拟内存的习惯说法中，块被称为页。在磁盘和内存之间传送页的活动叫做交换(swapping)或者页面调度(paging)。页从磁盘换入(或者页面调入)DRAM 和从 DRAM 换出(或者页面调出)磁盘。一直等待，直到最后时刻，也就是当有不命中发生时，才换入页面的这种策略称为按需页面调度(demand paging)。也可以采用其他的方法，例如尝试着预测不命中，在页面实际被引用之前就换入页面。然而，所有现代系统都使用的是按需页面调度的方式。

9.3.5　分配页面

图 9-8 展示了当操作系统分配一个新的虚拟内存页时对我们示例页表的影响，例如，调用 `malloc` 的结果。在这个示例中，VP5 的分配过程是在磁盘上创建空间并更新 PTE 5，使它指向磁盘上这个新创建的页面。

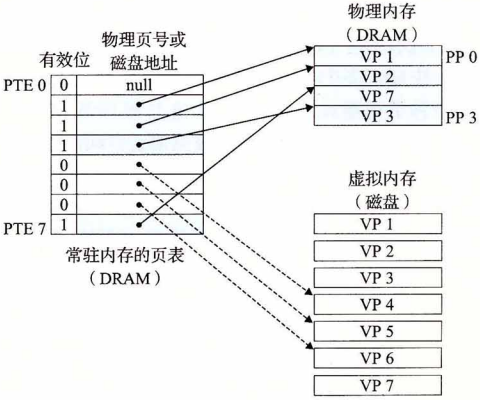

图 9-8　分配一个新的虚拟页面。内核在磁盘上分配 VP 5，并且将 PTE 5 指向这个新的位置

9.3.6　又是局部性救了我们

当我们中的许多人都了解了虚拟内存的概念之后，我们的第一印象通常是它的效率应该是非常低。因为不命中处罚很大，我们担心页面调度会破坏程序性能。实际上，虚拟内存工作得相当好，这主要归功于我们的老朋友局部性(locality)。

尽管在整个运行过程中程序引用的不同页面的总数可能超出物理内存总的大小，但是局部性原则保证了在任意时刻，程序将趋向于在一个较小的活动页面(active page)集合上工作，这个集合叫做工作集(working set)或者常驻集合(resident set)。在初始开销，也就是将工作集页面调度到内存中之后，接下来对这个工作集的引用将导致命中，而不会产生额外的磁盘流量。

只要我们的程序有好的时间局部性，虚拟内存系统就能工作得相当好。但是，当然不是所有的程序都能展现良好的时间局部性。如果工作集的大小超出了物理内存的大小，那么程序将产生一种不幸的状态，叫做抖动(thrashing)，这时页面将不断地换进换出。虽然虚拟内存通常是有效的，但是如果一个程序性能慢得像爬一样，那么聪明的程序员会考虑是不是发生了抖动。

旁注　统计缺页次数

你可以利用 Linux 的 `getrusage` 函数监测缺页的数量(以及许多其他的信息)。

9.4　虚拟内存作为内存管理的工具

在上一节中，我们看到虚拟内存是如何提供一种机制，利用 DRAM 缓存来自通常更大的虚拟地址空间的页面。有趣的是，一些早期的系统，比如 DEC PDP-11/70，支持的是一个比物理内存更小的虚拟地址空间。然而，虚拟地址仍然是一个有用的机制，因为它