等。如图10-15，这些信息是如下组织的。

首先，Linux维护一个页描述符数组，称为mem_map，其中页描述符是page类型的，而且系统当中的每个物理页框都有一个页描述符。每个页描述符都有个指针，在页面非空闲时指向它所属的地址空间，另有一对指针可以使得它跟其他描述符形成双向链表，来记录所有的空闲页框和一些其他的域。在图10-15中，页面150的页描述符包含一个到其所属地址空间的映射。页面70、页面80、页面200是空闲的，它们是被链接在一起的。页描述符的大小是32字节，因此整个mem_map消耗了不到1%的物理内存（对于4KB的页框）。

因为物理内存被分成区域，所以Linux为每个区域维护一个区域描述符。区域描述符包含了每个区域中内存利用情况的信息，例如活动和非活动页的数目，页面置换算法（本章后面介绍）所使用的高低水位，还有许多其他的域。

此外，区域描述符包含一个空闲区数组。该数组中的第i个元素标记了2^i个空闲页的第一个块的第一个页描述符。既然可能有多块2^i个空闲页，Linux使用页描述符的指针对把这些页面链接起来。这个信息在Linux的内存分配操作中使用。在图10-15中，free_area[0]标记所有仅由一个页框组成的物理内存空闲区，现在指向页面70，三个空闲区当中的第一个。其他大小为一个页面的空闲块也可通过页描述符中的链到达。

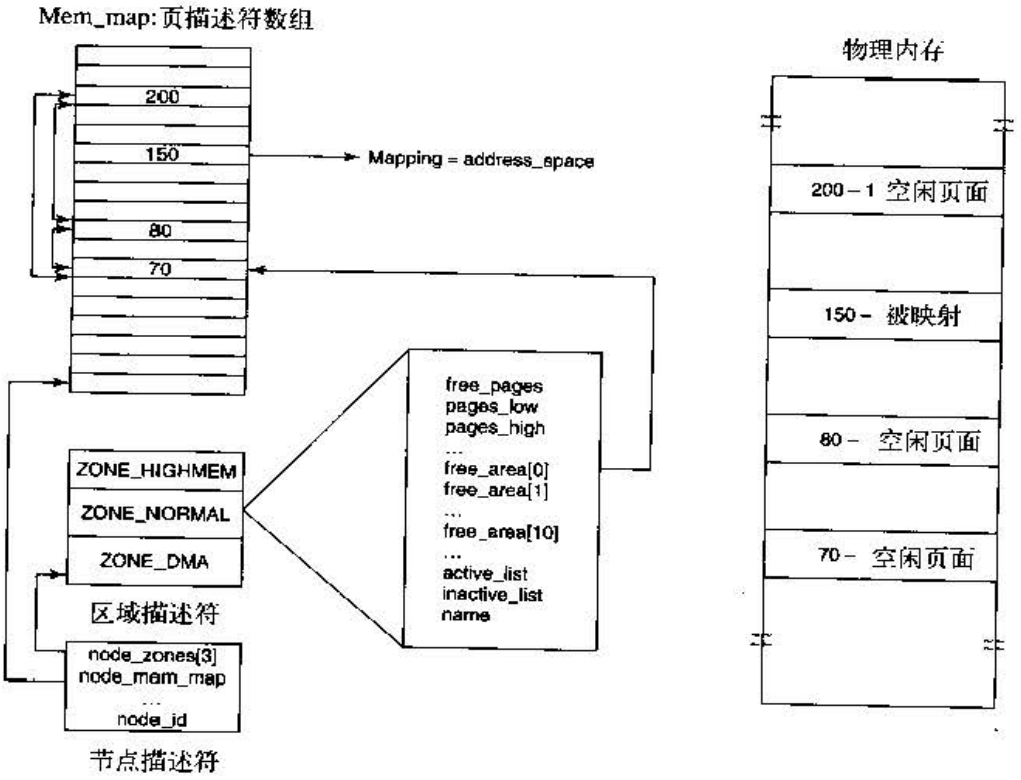

图10-15 Linux内存表示

最后，Linux可以移植到NUMA体系结构（不同的内存地址有不同的访问时间），为了区分不同节点上的物理内存（同时避免跨节点分配数据结构），使用了一个节点描述符。每个节点描述符包含了内存使用的信息和该节点上的区域。在UMA平台上，Linux用一个节点描述符描述所有的内存。每个页描述符的最初一些位是用来指定该页框所属的节点和区域的。

为了使分页机制在32位和64位体系结构下高效工作，Linux采用了一个四级分页策略。这是一种最初在Alpha系统中使用的三级分页策略，在Linux 2.6.10之后加以扩展，并且从2.6.11版本以后使用的一个四级分页策略。每个虚拟地址划分成五个域，如图10-16。目录域是页目录的索引，每个进程都有一个私有的页目录。找到的值是指向其中一个下一级目录的一个指针，该目录也由虚拟地址的一个域索引。中级页目录表中的表项指向最终的页表，它是由虚拟地址的页表域索引的。页表的表项指向所需要的页面。在Pentium处理器（使用两级分页）上，每个页的上级和中级目录仅有一个表项，因此总目录项就