现在，假设第二台虚拟机启动，希望把它的虚拟页面4、5、6分别映射到物理页面10、11、12，并加载指向页表的控制寄存器。管理程序捕捉到了这次陷入，但是它会做什么呢?它不能进行这次映射，因为物理页面10、11、12正在使用。它可以找到其他空闲页面，比如说20、21、22并使用它们，但是在此之前，它需要创建一个新的页表完成虚拟页面4、5、6到物理页面20、21、22的映射。如果还有其他的虚拟机启动，继续请求使用物理页面10、11、12，管理程序也必须为它创建一个映射。总之，管理程序必须为每一台虚拟机创建一个影子页表（shadow page table），用以实现该虚拟机使用的虚拟页面到管理程序分配给它的物理页面之间的映射。

但更糟糕的是，每次客户操作系统改变它的页表，管理程序必须相应地改变其影子页表。例如，如果客户操作系统将虚拟页面7重新映射到它所认为的物理页面200（不再是物理页面10了）。管理程序必须了解这种改变。问题是客户操作系统只需要写内存就可以完成这种改变。由于不需要执行敏感指令，管理程序根本就不知道这种改变，所以就不会更新它的由实际硬件使用的影子页表。

一种可能的（也很笨拙的）解决方式是，管理程序监视客户虚拟内存中保存顶级页表的内存页。只要客户操作系统试图加载指向该内存页的硬件寄存器，管理程序就能获得相应的信息，因为这条加载指令是敏感指令，它会引发陷入。这时，管理程序建立一个影子页表，把顶级页表和顶级页表所指向的二级页表设置成只读。接下来客户操作系统只要试图修改它们就会发生缺页异常，然后把控制交给管理程序，由管理程序来分析指令序列，了解客户操作系统到底要执行什么样的操作，并据此更新影子页表。这种方法并不好，但它在理论上是可行的。

在这方面，将来的VT技术可以通过硬件实现两级映射从而提供一些帮助。硬件首先把虚拟页面映射成客户操作系统所认为的“物理页面”，然后再把它（硬件仍然认为它是虚拟页面）映射到物理地址空间，这样做不会引起陷入。通过这种方式，页表不必再被标记成只读，而管理程序只需要提供从客户的虚拟空间到物理空间的映射。当虚拟机切换时，管理程序改变相应的映射，这与普通操作系统中进程切换时系统所做的改变是相同的。

在准虚拟化的操作系统中，情况是不同的。这时，准虚拟化的客户操作系统知道当它结束的时候需要更改进程页表，此时它需要通知管理程序。所以，它首先彻底改变页表，然后调用管理程序例程来通知管理程序使用新的页表。这样，当且仅当全部的内容被更新的时候才会进行一次管理例程调用，而不必每次更新页表的时候都引发一次保护故障，很明显，效率会高很多。

### 8.3.6 I/O设备的虚拟化

了解了处理器和内存的虚拟化，下面我们来研究一下I/O的虚拟化。客户操作系统在启动的时候会探测硬件以找出当前系统中都连接了哪种类型的I/O设备。这些探测会陷入到管理程序。那么管理程序会怎么做呢？一种方法是向客户操作系统报告设备信息，如磁盘、打印机等真实存在的硬件。于是客户操作系统加载相应的设备驱动程序以使用这些设备。当设备驱动程序试图进行I/O操作时，它们会读写设备的硬件寄存器。这些指令是敏感指令，将会陷入到管理程序，管理程序根据需要从硬件中读取或向硬件中写入所需的数据。

但是，现在我们有一个问题。每一个客户操作系统都认为它拥有全部的磁盘分区，而同时实际上虚拟机的数量比磁盘分区数多得多（甚至可能是几百个）。常用的解决方法是管理程序在物理磁盘上为每一个虚拟机创建一个文件或区域作为它的物理磁盘。由于客户操作系统试图控制真正的物理磁盘（如管理程序所见），它会把需要访问的磁盘块数转换成相对于文件或区域的偏移量，从而完成I/O操作。

客户操作系统正在使用的磁盘也许跟真实的磁盘不同。例如，如果真实的磁盘是带有新接口的某些新品牌、高性能的磁盘（或RAID），管理程序会告知客户操作系统它拥有的是一个旧的IDE磁盘，让客户操作系统安装IDE磁盘驱动。当驱动程序发出一个IDE磁盘命令时，管理程序将它们转换成新磁盘驱动的命令。当硬件升级、软件不做改动时，可以使用这种技术。事实上，虚拟机对硬件设备重映射的能力证实VM/370流行的原因：公司想要买更新更快的硬件，但是不想更改它们的软件。虚拟技术使这种想法成为可能。

另一个必须解决的I/O问题是DMA技术的应用。DMA技术使用的是绝对物理内存地址。我们希望，管理程序在DMA操作开始之前介入，并完成地址的转换。不过，带有I/O MMU的硬件出现了，它按照