This is crucial for protecting processes from one another in a regular operating system. Therefore the kernel must ask the hypervisor to carry out these operations on its behalf, using a mechanism called a *hypercall*. A hypercall is similar to a system call (from a user process to the kernel), except that it is used for communication between a kernel and the hypervisor, and it typically implements lower-level operations.

Virtual memory is used to ensure that processes cannot interfere with the data or code of other processes. Each process is given a virtual *address space*, which ensures that that process can access only its allocated memory. The kernel is responsible for creating the virtual address space, by maintaining *page tables*, which map virtual addresses to the physical addresses that identify the actual location of data on the memory chips. When it is running in a virtual machine, the kernel does not have carte blanche to manage these tables, as it could conceivably make a mapping to memory that belongs to another virtual machine. Therefore, Xen must validate all updates to the page tables, and the kernel must inform the hypervisor when it wants to change any page table. This could be very inefficient if the hypervisor were involved in every page table update (for example, when a new process starts and its page tables are first built). However, it turns out that these cases are relatively rare, and Xen can amortize the cost of going to the hypervisor by batching the update requests or "unhooking" the page table while it is being updated.

Look at Figure 7-2. Each virtual machine has a share of the total physical memory.[‡] However, it might not be contiguous, and, in most cases, it will not start at physical address zero. Therefore, each virtual machine kernel deals with two types of addresses: physical (or *machine*) and *pseudophysical*. The physical addresses correspond to the actual location of data on the memory chips, whereas the pseudophysical addresses provide the virtual machine with the illusion of a contiguous physical address space that starts at zero. Pseudophysical addresses may be useful for certain algorithms and subsystems that rely on this assumption and would otherwise need to be paravirtualized.

To be of any practical use, it must be possible to interact with a virtual machine. At a bare minimum, the virtual machine needs a disk (more properly known as a *block device*), and a network card.[§] Since most operating systems include support for at least one block device and network card, it might seem tempting for the hypervisor to emulate these devices so that the original drivers could be used. However, the software implementation would struggle to emulate the performance of the real devices, and the emulated device models may have to go through contortions (such as implementing hardware protocols) that are unnecessary and inefficient when providing a device's function in software.

[‡] Note that Xen does not support overcommitting physical memory, so there is no swapping of virtual machines. However, the memory footprint of a virtual machine can be altered using a process called *ballooning*.

[§] You might think that a mouse, keyboard, and video output would be necessary for interactivity, but these can be provided by a remote desktop client such as VNC. Nevertheless, recent versions of Xen have included support for these virtual devices.