SOLUTIONS TO CHAPTER 7 PROBLEMS

1. There are numerous reasons, among them consolidating servers to save hardware investment cost, rack space, and electrical power, and make management of thousands of servers easier.

2. If the hardware configuration was upgraded, virtualization could hide this and allow old software to continue working.

3. There are various reasons. A key one is to have many platforms such as Windows 7, Windows 8, Linux, FreeBSD, OS X, etc. available on a single desktop machine to test the software being developed. Also, rebooting a virtual machine after a crash induced by a software bug is much faster.

4. After upgrading to a new computer and operating system, the person might want to run some software that he had on the old one. Virtualization makes it possible to run the old system and new one on the same computer, thus preserving the old software.

5. Very few programmers had access to an IBM mainframe. Starting on the 1980s, the Intel x86 series dominated computing and it was not virtualizable. While binary translation could solve that problem, that idea was not thought of until the late 1990s.

6. Any instruction changing the page tables or memory map is certainly sensitive, as well as anything involving I/O. Any instruction capable of reading the true state of the machine is also sensitive.

7. There are many, including moves, arithmetic instructions, jump and call instructions, shifts, etc.

8. Full virtualization means emulating the hardware exactly so every operating system running on the virtual machine behaves exactly as it would on the bare metal. Paravirtualization consists of changing the operating system so it does not do anything that is hard to virtualize. Full virtualization in the absence of hardware support is complicated on any architecture that is complex, like the x86. It is easier on RISC machines. If virtualization hardware is present, full virtualization is not so difficult. So, which is harder probably depends on whether hardware support is available. If it is, then paravirtualizing an operating system is probably more work. If there is no hardware support, it may be easier to change the operating system to be more friendly. If there are many operating systems that have to be paravirtualized, that could be more work.

9. Yes, of course. Linux has been paravirtualized precisely because the source code is available. Windows has been paravirtualized by Microsoft (which has the source code), but has not released any paravirtualized versions.

10. Virtual machines have nothing to do with disk partitions. The hypervisor can take a disk partition and divide it up into subpartitions and give each virtual machine one of them. In principle, there can be hundreds. It can either statically partition the disk into *n* pieces or do this on demand. In hosted virtual machines, it is common to use files on the host to store disk images of the guest.

11. An application or process is virtualized during runtime, by using a virtualization layer between the application and the OS. This layer executes the application's instructions, modifying them as required prior to execution. The application is transparent to the presence of the underlying layer. Windows Emulator (WINE) is an example, where Microsoft Windows binary executables can be executed on another operating system such as Linux. This is done using on-the-fly mapping of Windows API calls to POSIX calls.

12. Type 1 hypervisors generally require changing the boot procedure of the computer to load the hypervisor first, then create virtual machines, and then install operating systems in them. At data centers run by expert system administrators, this is not a problem, but for most ordinary users, doing this is far too complicated. Type 2 hypervisors were invented to make installing a hypervisor no more difficult than installing an application program, something that users frequently do. Also, by using a host operating system to service local peripherals, it was not necessary for the hypervisor to have drivers for all of them since it could use the ones inside the host OS.

13. Yes. When a guest OS does I/O, for example, the virtualization hardware catches it and gets control to the type 2 hypervisor, which then figures out what to do. Usually this will involve making a request to the host OS to perform the I/O, but not having to worry about trapping the I/O instruction definitely simplifies matters for the hypervisor.

14. It was invented in the early days, before virtualization hardware existed. It was necessary to prevent guest operating systems, which were running in user mode, from executing sensitive instructions that were not privileged. Going forward, this is less necessary since modern hardware traps when a user-mode program executes a sensitive instruction. However, in some circumstances, binary translation is faster than trapping. Nevertheless, as the hardware improves, the need for binary translation will decrease.

15. Typically, ring 0 (with the highest set of privileges) is used for running in kernel mode; and ring 3 for user mode. Some hypervisors used ring 0 for running the hypervisor in kernel mode; the guest OS was run in ring 1. When the guest OS invokes privileged instructions, it could trap to the hypervisor that runs these instructions after verifying the access rights, permissions, etc. Other ways are also possible.

16. It has been shown that the VT-enabled CPU approach results in a lot of traps due to the use of trap-and-emulate approach. Since traps are expensive to handle, there are instances where translation based approaches outperform the hardware based approach.

17. When the guest OS issues a ''Clear Interrupt'' instruction (such as CLI), doing it in hardware can be very time consuming in some CPUs such as those with deep pipelines. On the other hand, in a virtualized system, the hypervisor need not actually disable interrupts in hardware and simply use a variable to indicate that the specified guest OS's ''Interrupt Flag'' is set to zero, making it faster to execute the CLI instruction.

18. It could translate the entire program in advance. The reason for not doing so is that many programs have large pieces of code that are never executed. By translating basic blocks on demand, no unused code is ever translated. A potential disadvantage of on-demand translation is that it might be slightly less efficient to keep starting and stopping the translator, but this effect is probably small. In addition, static analysis and translation of x86 code is complicated due to indirect branches (branches whose targets are computed at run time). This is made worse by the variable-size instructions on the x86. Thus you may not be sure which instructions to translate. Finally, there is the issue of self-modifying code.

19. A pure hypervisor just emulates the real hardware and nothing else. A pure microkernel is a small operating system that offers basic services to the programs using it. The virtual machines running on a pure hypervisor run traditional operating systems such as Windows and Linux. On top of a microkernel are generally processes that implement operating system services but in a decentralized way.

20. If multiple guest OSes all allocate what they think is physical page *k* to one of their processes, there is a problem. Some way is needed to perform a second mapping between pages because the guests do not really control the physical pages, despite what they may think. This is why nested page tables are needed.

21. Not only does the machine need memory for the normal (guest) operating system and all its applications, but it also needs memory for the hypervisor functions and data structures needed to execute sensitive instructions on behalf of the guest OS. Type 2 hypervisors have the added cost of the host operating system. Moreover, each virtual machine will have its own operating system, so there will be *N* operating system copies stored in memory. One way to reduce memory usage would be to identified "shared code" and keep only one copy of this code in memory. For example, a Web hosting company may run multiple VMs, each running an identical version of Linux and an identical copy of the Apache web server code. In this case, the code segment can be shared across VMs, even though the data regions must be private.

22. Each guest OS will maintain a page table that maps its virtual page numbers to physical frame numbers (on its share of the virtualized memory). In order to prevent different guest operating systems from incorrectly referring to the same physical page number, the hypervisor creates a shadow page table that maps the virtual machine's virtual page number to the physical frame number provided by the hypervisor.

23. Page tables can be modified only by the guest operating system, not the application programs in the guest. When the guest OS is finished modifying the tables, it must switch back to user mode by issuing a sensitive instruction like RETURN FROM TRAP. This instruction will trap and give the hypervisor control. It could then examine the page tables in the guest OS to see if they had been modified. While this could work, all the page tables would have to be checked on every system made by a guest application, that is, every time the guest OS returned to user mode. There could be thousands of these transitions per second, so it is not likely to be as efficient as using read-only pages for the page table.

24. When a hypervisor runs out of pages, it has no way of figuring out which pages the guest operating systems really value. The solution is to cheat and include balloon drivers in the guests. The hypervisor then signals the balloon drivers to expand their memory usage, forcing the guest operating systems to decide which pages to evict. This is definitely cheating because the hypervisor is not supposed to talk to specific pieces of the guest operating systems. It is not supposed to know what is going on in the virtual machines at all. But this technique solves a problem in a simple way, so everyone pretends there is nothing iffy going on.

25. Balloon drivers do not work if the hypervisor does not know anything about the guest operating systems running on its virtual machines. It also does not work if there is no way to include a balloon driver in them, for example, if they do not support loadable drivers and the source code is not available so they cannot be recompiled to include the balloon driver.

26. Consider a case where multiple virtual machines copies of the same guest OS reside in a system. In this case, it is not necessary to maintain multiple copies of the read-only portion of the OS (such as code segments) in memory. Only one copy needs to be maintained, thereby reducing memory requirements and allowing more virtual machines on a system. This technique is called deduplication. VMware calls this ''transparent page sharing.''

27. Yes. Early DMA hardware used absolute memory addresses. If a guest operating system started a DMA operating to what it thought was physical address *k*, this would probably not go to the buffer it was supposed to go to and might overwrite something important. Early hypervisors had to rewrite code that used DMA to use addresses that would not cause trouble.

28. Using cloud services means you do not have to set up and maintain a computing infrastructure. You may also be able to outsource making backups. Furthermore, if your computing needs change rapidly, you can add or remove machines easily. On the downside, the cloud provider could easily steal your confidential data, and the promised expandability might be illusory if you need extra capacity just at the moment Walmart or some other big customer decides to grab 10,000 machines. Also, the bandwidth between you and the cloud might be an issue. It is likely to be far less than the local bandwidth, so if a lot of data needs to move between you and the cloud, that could be an issue. Also, if you are doing real-time work, the bandwidth between you and the cloud could vary wildly from moment to moment, causing trouble.

29. Obviously there are many, but a provider offering empty virtual x86 machines would be offering IAAS. A provider offering Windows 8 or Linux machines would be offering PAAS. A provider offering a word-processing program, such as Microsoft Word, running in the cloud would be offering software as a service.

30. Suppose many virtual machines were started up on a single server. Initially, all of them did about the same amount of work and required the same resources and the situation was fine. Then all of a sudden, one of them began using massive resources (CPU, memory, etc.) disturbing all the other virtual machines. This might be a good time to migrate it to a dedicated server of its own.

31. Physical I/O devices still present problems because they do not migrate with the virtual machine, yet their registers may hold state that is critical to the proper functioning of the system. Think of read or write operations to devices (e.g., the disk) that have been issued but have not yet completed. Network I/O is particularly difficult because other machines will continue to send packets to the hypervisor, unaware that the virtual machine has moved. Even if packets can be redirected to the new hypervisor, the virtual machine will be unresponsive during the migration period, which can be long because the entire virtual machine, including the guest operating system and all processes executing on it, must be moved to the new machine. As a result packets can experience large delays or even packet loss if the device/hypervisor buffers overflow.

32. In order to migrate a specific process, process state information has to be stored and then transferred, including open files, alarms, signal handlers, etc. Errors may creep in during the state capture task leading to potentially incorrect, incomplete or inconsistent state information. In the case of VM migration, the entire memory and disk images are moved to the new system, which is easier.

33. Standard (dead) migration consists of stopping the virtual machine and saving its memory image as a file. The file is then transported to the destination, installed in a virtual machine, and restarted. Doing so causes the application to stop for a little while during transport. In many circumstances having the ap-

plication stop is undesirable. With live migration, the pages of the virtual machine are moved while it is running. After they all arrive at the destination, a check is made to see if any of them have changed since being migrated. If so, they are copied again. This process is repeated until all the pages at the destination are up to date. Working this way (live migration) means applications can be moved with no downtime.

34. The three main requirements were: Compability (ability to run an existing guest OS without any modifications as a virtual machine); Performance (minimal overhead during VM execution; else, users would not choose to run their applications inside a VM) and Isolation (protecting the hardware resources from malicious or otherwise unauthorized access).

35. There was no way that VMware could have drivers for the thousands of different I/O devices in existence. By having VMware Workstation be a type 2 hypervisor, it could solve the problem by indirectly using the drivers already installed in the host OS.

36. VMware ESXi has been made small so it can be put into the firmware of the servers. When the server is turned on, the BIOS can then copy itself to RAM and start creating virtual machines. This greatly simplifies the booting and startup process.

37. Several examples can be found at: *virtualboximages.com*. These include various distributions of preinstalled Open Source Operating Systems. For example, rather than get an ISO for a new Linux flavor go through the install process and then get the VM running, it is easier to download the preinstalled VDI. There are similar appliances that run on VMWare. Other examples can be found at: *http://www.turnkeylinux.org*