**14.4 PROCESSOR ISSUES**

The QEMU and the Android Emulator in the Android SDK. They have the benefit of being easily transportable since they are not platform dependent, but they are not very efficient from a performance standpoint, as the emulation process is resource intensive. The second model doesn’t virtualize processors but provides segments of processing time on the physical processors (pCPUs) of the virtualization host to the virtual processors of the virtual machines hosted on the physical server. This is how most of the virtualization hypervisors offer processor resources to their guests. When the operating system in a virtual machine passes instructions to the processor, the hypervisor intercepts the request. It then schedules time on the host’s physical processors, sends the request for execution, and returns the results to the VM’s operating system. When multiple VMs are contending for processor, the hypervisor acts as the traffic controller, scheduling processor time for each VM’s request as well as directing the requests and data to and from the virtual machines. In a physical server, typically the application has exclusive use of all the compute resources configured in the system. In reality, from a processor standpoint, most servers are vastly underutilized, which is a strong driver for consolidation through virtualization as discussed earlier. When applications are migrated to virtual environments, one of the larger topics of discussion is how many virtual processors should be allocated to their virtual machines. Since the physical server they are vacating had sixteen cores, often the request from the application team is to duplicate that in the virtual environment, regardless of what their actual usage was. In addition to ignoring the usage on the physical server, another overlooked item is the improved capabilities of the processors on the newer virtualization server. To help “right-size” the virtual machine configurations, there are tools available that will monitor resource (processor, memory, network, and storage I/O) usage on the physical servers then make recommendations for the optimum VM sizing. One basic rule during VM creation is to begin with one vCPU and monitor the application’s performance. Another good practice is not to overallocate the number of vCPUs in a VM. A matching number of pCPUs need to be scheduled for the vCPUs in a VM. Native operating systems manage hardware by acting as the intermediary between application code requests and the hardware. key function of the operating system is to help prevent malicious or accidental system calls from disrupting the applications. Protection rings describe level of access or privilege inside of a computer system:

* The most trusted layer is often called Ring 0 (zero) and is where the operating system kernel works and can interact directly with hardware.
* Rings 1 and 2 are where device drivers execute while user applications run in the least trusted area, Ring 3. In practice, though, Rings 1 and 2 are not often used, simplifying the model to trusted and untrusted execution spaces.
* This separation prevents unprivileged code from causing untrusted actions such as a system shutdown or an unauthorized access of data from a disk or network connection.

**14.5 MEMORY MANAGEMENT**

Memory resources are usually the first bottleneck that virtual infrastructures reach as they grow. Also, like the virtualization of processors, memory usage in virtual environments is more about the management of the physical resource rather than the creation of a virtual entity. The virtual machine is configured with fewer resources than the physical host contains. When the virtual machine uses memory resources, the hypervisor manages the memory requests through the use of translation tables, so the guest (VM) operating system addresses the memory pace at the addresses that they expect. This is a good first step, but problems remain like processor, application owners ask for memory allocations that mirror the physical infrastructures they migrated from, regardless of whether the size of the allocation is warranted or not. This leads to overprovisioned virtual machines and wasted memory resources usage. On hosts where the guests are running the same operating system and the same applications, between 10 and 40% of the actual physical memory can be reclaimed. At 25%, an 8-GB server could host two additional 1-GB virtual machines. Since the hypervisor manages page sharing, the virtual machine operating systems are unaware of what is happening in the physical system.

Another strategy for efficient memory use is akin to thin provisioning in storage management. This allows an administrator to allocate more storage to a user than is actually present in the system.

**14.6 I/O MANAGEMENT**

Application performance is often directly linked to the bandwidth that a server has been allocated. Whether it is storage access that has been bottlenecked, or constrained. traffic to the network, either case will cause an application to be perceived as underperforming. In this way, during the virtualization of workloads, I/O virtualization is a critical item. The architecture of how I/O is managed in a virtual environment. In the virtual machine, the operating system makes a call to the device driver as it would in a physical server. The device driver then connects with the device; though in the case of the virtual server, the device is an emulated device that is staged and managed by the hypervisor. These emulated devices are usually a common actual device, such as an Intel e1000 network interface card or simple generic SGVA or IDE controllers. This virtual device plugs into the hypervisor’s I/O stack that communicates with the device driver that is mapped to a physical device in the host server, translating guest I/O addresses to the physical host I/O addresses. The hypervisor controls and monitors the requests from the virtual machine’s device driver, through the I/O stack, out the physical device, and back again, routing the I/O calls to the correct devices on the correct virtual machines. There are some architectural differences between vendors, but the basic model is similar. The advantages of virtualizing the workload’s I/O path are many. It enables hardware independence by abstracting vendor-specific drivers to more generalized versions that run on the hypervisor. A virtual machine running on an IBM server as a host can be live migrated to an HP blade server host without worrying about hardware incompatibilities or versioning mismatches. This abstraction enables of one of virtualization’s greatest availability strengths: live migration. Sharing of aggregate resources, network paths, for example, is also due to this abstraction. In more mature solutions, capabilities exist to granularly control the types of network traffic and the bandwidth afforded to individual VMs or groups of virtual machines to ensure adequate performance in a shared environment to guarantee a chosen Quality of Service level. In addition to these, there are other features that enhance security and availability. The trade-off is that the hypervisor is managing all the traffic, for which it is designed, but it requires processor overhead. In the early days of virtualization this was an issue that could be a limiting factor, but faster multicore processors and sophisticated hypervisors have all but removed this concern. TCP Offload Engine (TOE) removes the TCP/IP processing from the server processor entirely to the NIC. Other variations on this theme are Large Receive Offloadm(LRO), which aggregates incoming packets into bundles for more efficient processing, and its inverse Large Segment Offload (LSO), which allows the hypervisor to aggregate multiple outgoing TCP/IP packets and has the NIC hardware segment them into separate packets. In addition to the model described earlier, some applications or users will demand a dedicated path. In this case, there are options to bypass the hypervisor’s I/O stack and oversight, and directly connect from the virtual machine’s device driver to physical device on the virtualization host. This provides the virtue of having adedicated resource without any overhead delivering the greatest throughput possible.