vmware®

VMware ESX Server 3

Ready Time Observations

VMware ESX Server is a thin software layer designed to multiplex hardware resources efficiently among virtual machines running unmodified commodity operating systems. In many computing environments, individual servers are underutilized, allowing them to be consolidated as virtual machines on a single physical server with little or no impact on performance. Similarly, many small servers can be consolidated onto fewer larger machines to simplify management and reduce costs.

To achieve best performance in a consolidated environment, you must consider ready time — the time a virtual machine must wait in a ready-to-run state before it can be scheduled on a CPU. This paper provides information to help you understand the factors that influence ready time on an ESX Server 3.0 system. It covers the following topics:

Providing Access to Computing Resources

Typical ESX Server hosts in production environments are likely to host multiple operating system instances on a multiprocessor system. These instances of the operating systems run in virtual machines and are referred to as guest operating systems, or guests. Each guest runs one or more applications. The ESX Server software schedules these guests to use the physical CPUs on the host as and when there is demand on the applications supported by the guests.

The virtual machine that runs a guest operating system has multiple components. These include the virtual CPU that the guest operating system uses. The virtual machine monitor manages the virtual machine's access to such hardware resources as CPU, memory, disk storage, network adapters, mouse, and keyboard. In this paper, discussion of scheduling a guest refers to scheduling the virtual CPU of the virtual machine to use the physical CPU of the host. In addition, the ESX Server software runs several processes to perform maintenance activities. The virtual machine components and these processes can be more complex than normal processes seen on operating systems. The physical CPUs on the ESX Server host are shared by all these maintenance processes as well as the virtual machines.

Whenever a resource is shared, there is a chance that an attempt to use the shared resource will not be fulfilled immediately because the resource is busy. When multiple processes are trying to use the same physical CPU, that CPU may not be immediately available and a process must wait

before ESX Server can allocate a CPU to it. The ESX Server scheduler manages access to the physical CPUs on the host system. The time a virtual machine or other process waits in the queue in a ready-to-run state before it can be scheduled on a CPU is known as ready time.

As overall CPU utilization and the number of virtual machines increase, the scheduler is more likely to require a virtual machine to wait for access to a CPU. Even when a guest operating system is not servicing load, there are maintenance activities that the operating system must perform (for example, it must service clock interrupts to maintain correct time). Thus, even idle guests must be scheduled, consume CPU resources (albeit small), and accumulate ready time. The fact that the scheduler is allocating CPU resources to operating systems — rather than to applications as a normal operating system does — can make the scheduling somewhat more complex than it is in normal operating systems.

As part of their performance and capacity planning, ESX Server administrators have looked at the statistics for processes running on the host and used the ready time metric as one of the inputs. Ready time can be an indicator of saturation on a system. Users sometimes equate the ready time observed with run queues on Unix or Linux. The fact that run queues are reported on a per-processor basis while ready time is reported for each virtual machine (or each virtual CPU in the case of multiprocessor virtual machines) causes the metric to be slightly different. Some users have asked how a virtual machine can accumulate ready time while it appears that CPUs are also accumulating idle time. This document attempts to address some of these questions about ready time.

Several factors affect the amount of ready time seen.

- Overall CPU utilization

  You are more likely to see ready time when utilization is high, because the CPU is more likely to be busy when another virtual machine becomes ready to run.

- Number of resource consumers (in this case, guest operating systems)

  When a host is running a larger number of virtual machines, the scheduler is more likely to need to queue a virtual machine behind one or more that are already running or queued.

- Load correlation

  If loads are correlated — for example, if one load wakes another one when the first load has completed its task — ready times are unlikely. If a single event wakes multiple loads, high ready times are likely.

- Number of virtual CPUs in a virtual machine.

  When co-scheduling for *n*-way Virtual SMP is required, the virtual CPUs can be scheduled only when *n* physical CPUs are available to be preempted.

In multiprocessor systems, an additional factor affects ready time. Virtual machines that have been scheduled on a particular CPU will be given a preference to run on the same CPU again. This is because of performance advantages of finding data in the CPU cache. In a multiprocessor system, therefore, the ESX Server scheduler may choose to let a few cycles on a CPU stay idle rather than aggressively move a ready virtual machine to another CPU that may be idle. Virtual machines can and do move occasionally when deemed beneficial by the scheduler algorithm. Scheduler options can be used to make this migration more aggressive; however, our tests indicate that this results in a lower overall system throughput and may not be desirable.

This adds an additional complication in understanding ready time on multiprocessor systems. Even when a CPU is idle, a virtual machine may be ready and waiting — that is, accumulating ready time.

vmware®

Test Description and Findings

The objective of these tests was to establish patterns correlating ready time, CPU utilization, and use of SMP virtual machines, and to look at scheduler event traces (available only in ESX Server 3.0) to see if CPUs were being used efficiently. For the most part the analysis was done on the beta 2 version of ESX Server 3.0, though test results were also collected on ESX Server 2.5.2.

The tests were run on a Hewlett-Packard DL 585 with four AMD Opteron CPUs at 1.6GHz and 4GB of RAM. We set up 10 virtual machines on this server, each with 256MB of RAM. Six of the virtual machines were set up as uniprocessor virtual machines. The other four had two CPUs each. All the virtual machines were running Red Hat Enterprise Linux 3.

The workload used in the tests was a custom CPU burner program, a single-threaded load generator. The program ran a fixed number of iterations of a CPU-intensive activity, then slept for some time. The amount of time spent in the compute-intensive loop or in sleep was in the same order as the default scheduler quantum (the time a guest can be on a CPU before it is descheduled to give other guests a turn — 50ms by default). On a system that is very busy, the behavior of the program becomes somewhat less predictable. The work element of the program is a fixed quantum. However, the sleep time is dependent on the guest operating system in the virtual machine delivering an interrupt to tell the CPU burner it is time to wake up. As the CPU the virtual machines are running on gets busier, it is possible that the virtual machine does not get scheduled to deliver this interrupt in a timely manner and the actual sleep periods thus get longer.

In a real-world workload, if a guest operating system is receiving a steady stream of work, it tends to accumulate more work, waiting in its queue, while the virtual machine is not active. In this simulation, on the other hand, work does not accumulate while the virtual machine is inactive. Thus in this simulation, the amount of work done falls as the system gets busier.

For the purposes of this test, this decrease in the amount of work done is not a problem, because we want to compare actual utilization with accumulated ready time. However, utilization does not grow in a linear fashion as load is added to the system.

Data for all tests was collected over a 10-minute period. The `esxtop` tool took periodic snapshots of the system. We used those snapshots to calculate how much ready time was being accumulated by virtual machines and other processes on the server. Utilization and ready time used here are aggregates for the virtual machines. Utilization and ready time from all other components of ESX Server are small and were not taken into consideration for this test.

vmware®

Processor Utilization versus Ready Time

The first test (Figure 1) shows how ready time changes with utilization. Six virtual machines were running on the test server. All of them were pinned to a single CPU. Load was generated by the CPU burner program run with a setting of approximately 15 percent, so that if it were the only thing running, the virtual machine would consume close to 15 percent of the capacity of a single CPU. Idle virtual machines also consumed some CPU capacity to perform guest operating system maintenance tasks.

The test started with the CPU burner running in a single virtual machine. After each 10-minute period, another guest started running the CPU burner. The data points in Figure 1 represent the statistics for each increment, as the CPU burner runs in one virtual machine, then in two, and eventually runs in all six. Because of the increase in sleep times, as explained in the previous section, the actual utilization from the CPU burner drops from the initial 15 percent when a single virtual machine is running. The utilization decrease is about 10 to 15 percent, depending on the version of ESX Server and the efficiency with which it delivers the interrupts.

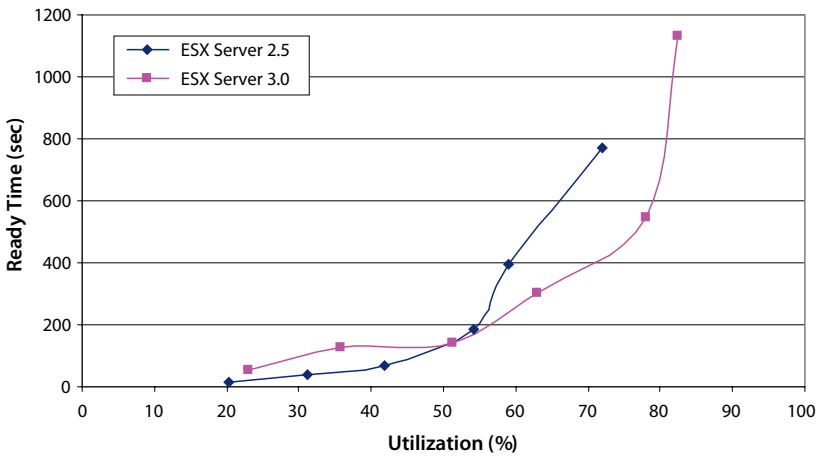

Figure 1: Aggregate ready time compared to utilization

The tests illustrate that with ESX Server 2.5, the ready time starts to increase dramatically between 55 and 60 percent utilization. With the scheduler changes made in ESX Server 3.0, this increase occurs around the same time but is much less sharp. At higher utilizations, the ESX Server 3.0 scheduler does a better job of servicing virtual machines efficiently, thus overall ready time remains lower.

Because ready time is time a process spends waiting when it could be running, it has a direct impact on response time. During the measurement period, because the virtual machines were all pinned to a single CPU, 600 seconds of CPU time were available. In the case in which four of the virtual machines have load, about 300 seconds of ready time were accumulated. An average 50 percent slowdown would therefore be expected. An event with a response time of 1 second may be expected to take 1.5 seconds under this level of load as a direct result of delays caused by ready time. This is not unexpected in observations of multiuser systems under load. The same factors apply to an ESX Server host running multiple virtual machines.

It is important to note that increasing amounts of ready time do not mean that the remaining CPU on the system is unusable. It just means that due to such factors as load synchronicity, there are periods when the CPU has no work to do and other times when it is running one virtual machine but has one to five others ready to run and waiting.

vmware®

Ready Time on SMP and Uniprocessor Virtual Machines

We ran an additional test to verify that there was no significant difference in ready time when running different mixes of SMP and uniprocessor virtual machines. We considered this an important question because of ESX Server co-scheduling (an attempt to give roughly equal time to all the virtual CPUs of an SMP virtual machine). In this test, the virtual machines were not pinned to a CPU, thus migrations could happen.

The test was run on a four-CPU system. Six single processor virtual machines and three SMP virtual machines (each with two virtual CPUs) were running for the duration of the test. The test involved running six instances of the CPU burner program set to consume 50 percent of a CPU — in other words, a total of 6 * 50% = 300% or three CPUs of the four available. The CPU burner was run in various combinations of uniprocessor and SMP virtual machines, starting with the CPU burner running only in uniprocessor virtual machines, then shifting one instance at a time to the SMP virtual machines until all six instances were running in three SMP virtual machines.

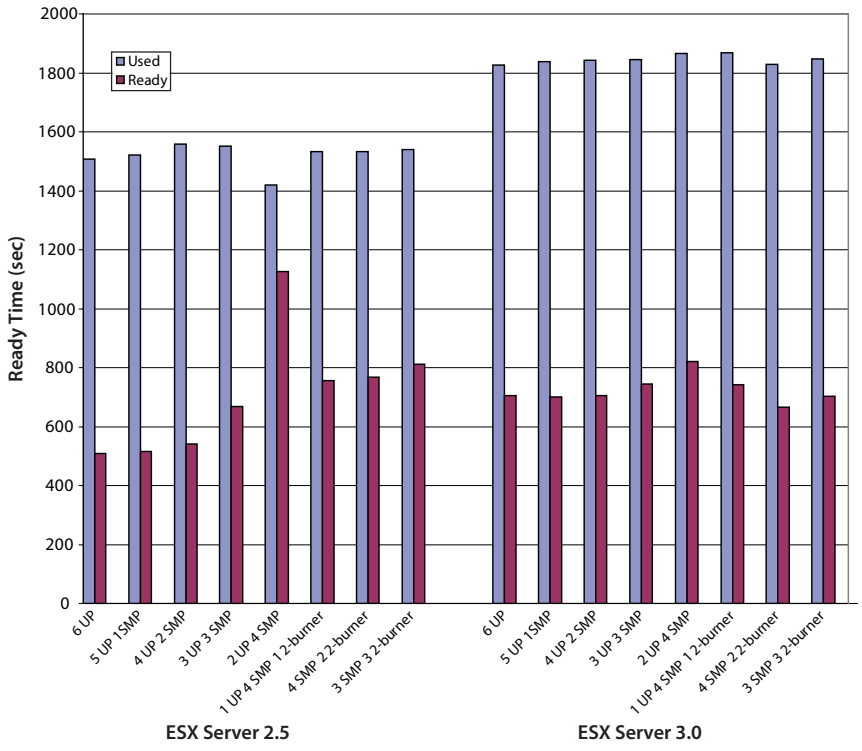

Figure 2: Used and ready time for various configurations

The results indicate that in ESX Server 3.0 there is little difference in ready time, whether this load is running on uniprocessor or SMP virtual machines. The scheduler overall does a better job of ensuring all the virtual machines get scheduled frequently enough, resulting in an actual usage of 75 percent — in contrast to ESX Server 2.5, on which the actual usage stays much lower. Scheduler changes and other improvements in ESX Server 3.0 result in a difference in utilization, thus preventing a direct comparison between ready times for the two versions. It is clear, however, that co-scheduling does not have a large impact on ready time in ESX Server 3.0.

vmware®

Scheduler Trace

ESX Server 3.0 provides a mechanism for looking at scheduler trace data to examine state information on processes and CPUs and to validate decisions made by the scheduler. The mechanism to collect this data is available in the released version of ESX Server 3.0. The data is collected by the `vm-support` script. However, the ability to post-process the script's binary output file is not part of the released product. The following is the sample post-processed output from this logging mechanism:

```
39989 2 46490240550 SETRUNSTATE 1212 WAIT
39990 2 46490240553 QUEUEREMOVE 1212
39991 2 46490240555 SETWAITSTATE 1212 1047688517 UPOL
39992 2 46490240558 VTIMESET 1212 24672 4216061778 4216061945 4215994412 RUN
39993 2 46490240562 VTIMESET 1212 24672 4216061945 4216061790 4215994412 VTLIMIT
39994 2 46490240569 SETRUNSTATE 1180 RUN
39995 3 46490240572 MIGRATE 1180 2 OTHER
39996 3 46490240573 QUEUEREMOVE 1180
39997 2 46490240575 QUEUEADD 1180
39998 2 46490240588 SWITCH 1212 1180
39999 2 46490240631 SETRUNSTATE 1180 WAIT
40000 2 46490240632 QUEUEREMOVE 1180
40001 2 46490240635 SETWAITSTATE 1180 28695732 RPC
40002 2 46490240637 VTIMESET 1180 35246 4216076408 4216076563 4215994419 RUN
40003 2 46490240646 SETRUNSTATE 1174 RUN
40004 1 46490240648 MIGRATE 1174 2 OTHER
40005 1 46490240650 QUEUEREMOVE 1174
40006 2 46490240652 QUEUEADD 1174
40007 2 46490240661 SWITCH 1180 1174
40008 3 46490240692 SETRUNSTATE 1187 WAIT
40009 3 46490240695 QUEUEREMOVE 1187
40010 3 46490240699 SETWAITSTATE 1187 14142156 IDLE
40011 3 46490240702 VTIMESET 1187 35246 4216090583 4216092541 4215994424 RUN
```

A detailed explanation of the log shown above is beyond the scope of this document. However, based on this information it is possible to construct a state diagram of what the CPUs are doing and which processes are ready to run, what CPUs they are waiting on, how long they waited in ready state before they began to run, and other aspects of CPU state. We used a custom script to convert the above data into the following state representation.

```
Event# ElapsedCPU0   CPU1   CPU2   CPU3
       Time
9335   213549 I      1165   1170   1187
9340   213563 1192   1165   1170   1187
9341   213565 1192   1165   1170   1187
9342   213621 I      1165   1170   1187
9347   213634 1184   1165   1170   1187
9348   213636 1184   1165   1170   1187
9351   213654 1184   1165   1170   1187
9352   213655 1184   1165   1170   1187
9353   213706 I      1165   1170   1187
9358   213718 1182   1165   1170   1187
9359   213721 1182   1165   1170   1187
9362   214007 1182   1165   1170   1187
9363   214009 1182   1196   1170   1187
9364   214012 1182   1196   1170   1187
9365   214049 1182   I      1170   1187
9369   214061 1182   1165   1170   1187
9370   214064 1182   1165   1170   1187
9371   214487 1182   1165   I      1187
9376   214507 1182   1165   1202   1187
```

```
9377   214509 1182   1165   1202   1187
9379   214513 1182   1165   1202   1187
9380   214520 1182   1165   1202   1187
9383   214529 1182   1165   1202   1187
9384   214531 1182   1165   1202   1187
9387   214559 1182   1165   1202   1187
9388   214561 1182   1165   1202   1187
9389   214595 1182   1165   I      1187
9393   214612 1182   1165   1172   1187   pr=1172 Wt=10795 Frm=203817, NumberIdle=31
                                           WtOnCpu=3
```

The data above gives us a clear picture of which process (which identifies the virtual machine) is running on the CPU or whether the CPU is idle. Each time a process enters a running state from the ready state it records how long (in microseconds) the process waited in the ready-to-run state before it could be scheduled. We also record the CPU it was waiting on and how many times any CPU was noticed in an idle state during the time this process was ready to run.

As can be seen from the data above, there are periods when a CPU is idle while a process is ready and waiting. In this instance process 1172 waited 10.795ms before it was allowed to run. During this time there were 31 instances when a CPU was idle during a state change but process 1172 was not run. The decision not to migrate the process to the idle CPU is borne out by the fact that the CPUs that were idle were in that state for very short periods of time and processes that had been previously scheduled on that CPU did become ready to run soon thereafter. As a result, the number of cycles lost was very small.

An analysis of this data on ESX Server 3.0 shows that processes are being allocated to CPUs very efficiently and no processes have large waits in ready state, waiting for a particular CPU to become available while another CPU is idle.

Interpreting Ready Time

Ready time for a process in isolation cannot be identified as a problem. The best metrics for examining the health of a server continue to be CPU utilization, response time, and application queues.

It is normal for a system to accumulate some ready time even when overall CPU utilization is low. Take an example of two processes (A and B) that each use 20 percent of a CPU, for an overall utilization of 40 percent. When process B is being scheduled, statistically 80 percent of the time the CPU is idle. The remaining 20 percent of the time process B must wait for process A to finish. The same is true for process A — 20 percent of the time it must wait for process B to finish.

This demonstrates that even under low utilization there is a chance that a shared resource will be busy. Thus some ready time is to be expected and is not a problem. The behavior is no different in the case of an ESX Server host with multiple running virtual machines. It behaves essentially the same way that an operating system does when trying to run multiple tasks concurrently.

The objective of server consolidation is to drive CPU utilization higher. For applications that are purely throughput driven it may be possible to drive the system to full utilization. For systems servicing applications that are somewhat interactive, attempting to drive the utilization beyond 60 to 70 percent may result in a perceptible lag in user activities. If response time thresholds are established for applications, they can give a clear picture of whether service levels demanded from an application are being met.

If response time is not a critical element of an application, queuing within the application may be used as a measure of server capacity. If the application is regularly falling behind in processing it may be time to re-examine server capacity.

One of the ways to collect ready time data is to collect periodic snapshots using the `esxtop` tool in batch mode. It may be best to start the tool in interactive mode, disable all columns not required for these tests, then collect the data with `esxtop` in batch mode. You can take snapshots a few minutes apart to determine how much used time and ready time has been accumulated by the virtual machines during the time between snapshots.

Once you have ready time data for each virtual machine, you can estimate how much of the observed response time is ready time. If the shortfalls in meeting response time targets for the applications appear largely due to the ready time, you can take steps to address the excessive ready time.

You can take various steps to reduce ready time for a virtual machine. One option is moving the virtual machine to another server using VMotion. Reducing the load from this virtual machine and others is another. You can reduce load from particular virtual machines by turning off unnecessary daemons, processes, and services in the guest operating system. Another option is adjusting CPU shares or CPU minimums in ESX Server to give more resources to virtual machines that must have short response times.

Recent changes to the ESX Server scheduler may help lower the amount of ready time observed and make scheduling more equitable and timely in general. We recommend upgrading to ESX Server 2.5.3, which has a partial set of the changes made, or to ESX Server 3.0 to take full advantage of the recent improvements.

VMware, Inc. 3145 Porter Drive Palo Alto, CA 94304 www.vmware.com

Revision 20061101 Item: ESX-ENG-Q406-331