

TO V OR NOT TO V: A PRACTICAL GUIDE TO VIRTUALIZATION

Gene Fernando
Professional Services
BMC Software, Inc
Email: eugene_fernando@bmc.com

Virtualization has become so widespread that industry pundits have called it a megatrend. Many forms of Virtualization are available today, each with its own set of potential gains. Of course, with every new technology comes a new set of challenges. This paper explores the benefits of Virtualization and discusses the difficulties in measuring the results in the real world. The inter-play between hardware and software is examined by covering two popular implementations of Virtualization: Intel's Hyper-threading technology and VMWare's ESX server software. A case study is included to present a practical approach for measuring and forecasting growth for virtual servers.

*To be or not to be, that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles,
And by opposing end them?"*

-- From Hamlet (III, i, 56-68)

Introduction

The use of virtualization techniques, long a mainstay of the mainframe world, has become widespread in both the Unix and Windows environments. The abstraction of applications and Operating Systems from the actual physical hardware is an ongoing process pushed along by the rapid pace of technological advances in IT. Many forms of Virtualization are available today, each with its own set of potential gains. Of course, with every new technology comes a new set of problems. The question, as the Bard so elegantly poses, is whether to attack these problems head on, or just sit this one out and wait for the next wave of technological change.

Let us first start with definitions. Virtualization is a broad term that encompasses many products on the market today. Loosely defined, it is a means of partitioning the resources of a computer into multiple units, each able to act independently of each other. It can be implemented at different levels: Processor, Memory, Storage. At the processor level, these units may be called Virtual Machines (VM). These VM's are designed to present a complete abstraction of the physical system environment, thereby allowing each system image to behave as if it were running on its own physical system. This paper focuses on two popular Virtualization techniques: Intel's Hyper-threading Processor Architecture and VMWare's ESX server software.

History of VM

Virtualization, and the use of Virtual Machines, has been around for quite some time. The abstraction of

applications and operating systems from the actual physical hardware is an ongoing process pushed along by many factors. One of the developers of IBM's VM/370 operating system stated that VM was created in order to prolong application longevity and assure the smooth operation of old and new programs in the same physical machine. One of the primary design goals was to isolate each virtual machine such that multiple applications and/or users would not interfere with each other [Creasey]. Program isolation would ease the transition from one hardware platform to the next, a very useful feature for a hardware vendor!

Memory management was one of the first areas addressed by these virtualization techniques. The ability to run a program using relative addressing, instead of referring to a physical address in memory, allowed multiple address spaces to run in the same machine as if each one owned the whole processor. This reduces the actual physical memory requirements since multiple memory address spaces can share the same physical memory. This level of virtualization is characterized by mainframe operating systems such as zOS.

As virtualization techniques became more sophisticated, vendors began to offer products that allowed multiple operating systems to run on the same physical system. The hardware layer is further abstracted such that each VM guest thinks that it is running in its own environment. Software implementation of this concept is exemplified by IBM's VM operating system. It allowed multiple OS (DOS, MVS, CMS) to run on the same physical processor, sharing CPU, I/O, and memory resources. The use of LPARs blends hardware and software to implement the same concept. As the idea gained market acceptance, vendors such as IBM, Amdahl, and Hitachi offered competing variants. Today, virtualization technology is enjoying rapid growth in the Windows, Unix, and Linux environments. These implementations range from hardware-based offerings from chip manufacturers to software-based offerings from various vendors.

Hardware-Based Virtualization

Virtualization is best known as a software based tool that allows the creation of virtual machines. This allows one physical system to host many system images. However, this software technology would not be possible without developments at the hardware level that allow processors to efficiently service these VM's. Thus, although technologies such as Intel's Hyper-threading are not usually thought of as Virtualization, it is presented here in order to start the story at the very beginning.

At the hardware level, processor throughput is enhanced using multi-threading techniques. Unix vendors, such as Sun, use chip multi-threading (CMT) processors (UltraSparc IV) to enhance thread level parallelism. Windows vendors, such as Intel, enhanced the multi-threading capability of their CPU's even further by introducing Hyper-threading technology. Hyper-threading technology makes a single physical processor appear as two logical processors to the OS. This capability is available with both the Xeon and Pentium 4 family of processors.

Hyper-threading has an interesting effect on CPU metrics. A hyper-threaded system will report double the number of processors to the Windows OS. Thus, a dual processor machine may potentially show a utilization of 400%. Furthermore, the CPU utilization of specific processes may sum up to greater than the reported total system utilization. To understand this phenomenon, let us take a closer look at the micro-architecture of the Intel Xeon hyper-threading processor. Fig.1 shows the major components of this processor. Note that the Rapid Execution Engine is the only place

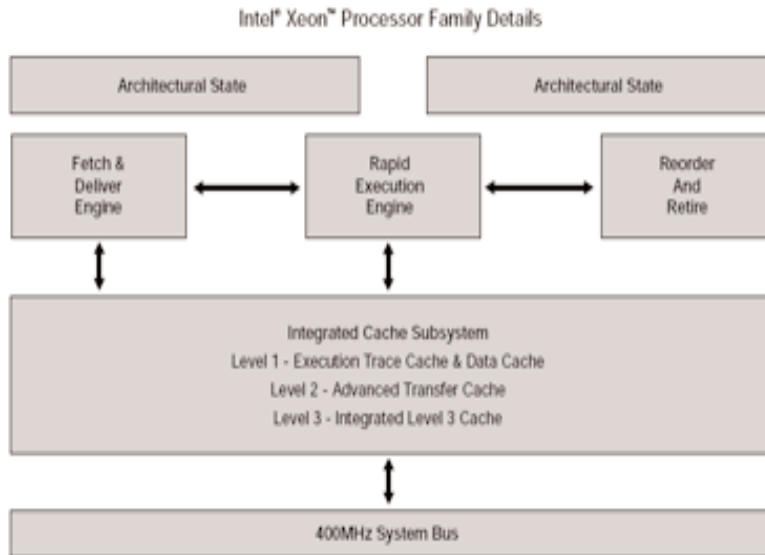


Figure 1. Micro-architecture components of an Intel Xeon Processor Source : [Intel1]

where instruction execution takes place. Like traditional processors, it can only process one set of instructions at a time. However, the key difference is that it

can store two architectural states. This allows the processor to switch between instructions from different threads or logical processors. By examining the components in this processor's micro-architecture, one gains an understanding of the interplay between instruction execution resources and the instruction preparation areas. The components in the Intel chip have analogs in most modern processor chip families.

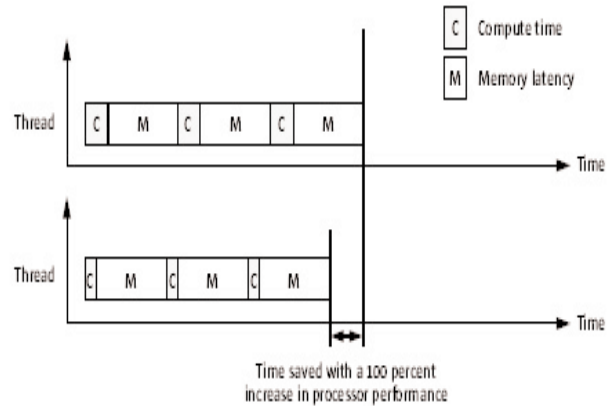


Figure 2 Single-threaded execution

From the Windows OS point of view, CPU time starts when the process enters the CPU. Since the CPU can handle multiple threads, it is entirely possible for multiple threads to be in the processor simultaneously using various components. Each of these threads is accumulating CPU active time.

Now we can watch the effects of parallel execution on throughput. Each thread has an execution component and a preparatory component, denoted in Fig. 2 as C or M. The illustration shows only memory, but this component also includes instruction pre-fetch, cache memory access time, etc. Since the processor is executing one thread at a time, the execution unit must wait until all the preparation steps are finished before it can go on. The situation is exacerbated by the inherent slowness of memory relative to processor clock speed. Since the preparation component is much larger than the execution component, increasing CPU speed has a smaller effect on throughput than one would initially assume. The old adage applies: "All CPU's wait at the same speed".

A processor using multi-threading technology is able to take better advantage of CPU speed. As illustrated in Fig 3, the ability to process multiple threads allows the execution unit to process work while preparatory work is being performed for other threads. The end result is higher throughput and better utilization of the execution unit. This processor would be able to take advantage of

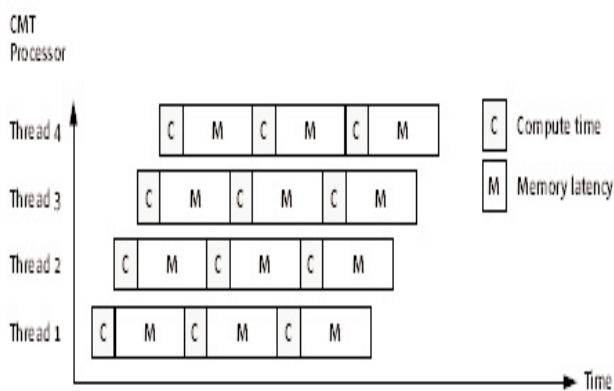


Figure 3 Multi-threaded execution

higher clock rates. The use of hyper-threading in CPUs presents the Capacity Planner with the following challenge: since each thread is accumulating CPU active time when it is in the processing unit, multi-threading will result in higher total processor active time – sometimes greater than actual elapsed time! This becomes readily apparent when the total CPU time of all running processes is compared to the physical CPU utilization. In many architectures, the number of processors presented to the OS is greater than the number of physical processors. For example, when Hyper-threading is turned on at the BIOS level for a server with two Intel processors, the Windows operating system will think it has four processors. It will dispatch work accordingly.

Answering what-if questions is the raison d'être for most capacity planners. In order to be worth their salt, they must be able to predict the impact of workload growth or system upgrades on system performance and resource utilization. Virtualized systems present difficult challenges to this process because there is a greater amount of volatility to an application's behavior as workload levels change. Further complicating the picture, applications take advantage of parallel processing with varying degrees of success. Slight changes in system software can have huge impacts on throughput. For example, moving from Linux version 2.4 to 2.5 results in almost a doubling of throughput because the latter version is designed to use parallel processing more effectively.

To illustrate these points, let's look at a benchmark run by Dr. Vianney at IBM [Vianney]. This benchmark used

Table 1. Effects of Hyper-Threading on chat throughput [Vianney]

Number of chat rooms	2419s-noht	2419s-ht	Speed-up
20	164071	202809	24.00%
30	151530	184803	22.00%
40	140301	171187	22.00%
50	123842	158543	28.00%
Geometric Mean	144167	178589	24.00%

Note: Data is the number of messages sent by client: higher is better.

a chat room simulation and compared the results when run in a processor with and without hyper-threading. The OS used was Linux 2.4. Table 1 shows that there is a consistent improvement between test cases on a non-hyper-threaded system versus a system with hyper-threading turned on. This is the good news. The bad news is that the amount of improvement is not constant, nor does it follow a discernible trend.

The test was repeated using Linux 2.5. (Table 2) This version of Linux contained significant improvements for use in Hyper-threaded systems. Like the previous test, significant improvements were observed with hyper-threading turned on. The speed-up was higher than in the Linux 2.4 case, but again, the increased throughput followed no consistent trend.

Table 2. Effects of Hyper-Threading on Linux kernel 2.5.32 [Vianney]

chat workload			
Number of chat rooms	2532s-noht	2532s-ht	Speed-up
20	137792	207788	51.00%
30	138832	195765	41.00%
40	144454	231509	47.00%
50	137745	191834	39.00%
Geometric Mean	139678	202034	45.00%

Lest we leave the reader with the wrong impression, it must be pointed out that the preceding tests used an application that was multi-threaded and multi-user. The results using single-threaded applications and kernel function calls were mixed. Results ranged from significant improvements to significant degradation. [Vianney]

As the previous sections have shown, the use of Hyper-threading technology brings a host of advantages and disadvantages. As with any technology, the question becomes: where is it best used? Hyper-threading brings improvements in performance at a very reasonable cost. Implementing the technology requires the use of servers containing Intel Xeon or Pentium 4 processors. Once this requirement is met, the only action required is to physically enable Hyper-threading. This is usually a switch setting on the BIOS. There are no software changes required, and users will not see any difference in the use of their applications. Of course, in order to take full advantage of Hyper-threading facilities, OS and application programmers may want to optimize the design of their software! However, legacy applications should still work, albeit with varying degrees of performance improvement. A scan of industry literature, performance tests, and Intel articles, leads one to expect a 15-30% improvement in throughput. [Parikh]

The disadvantages of Hyper-threading revolve around the predictability of performance and the challenges it poses to the capacity planner as a result. The preced-

ing sections outlined the ability of this technology to process multiple streams of programs and double the number of processors available for dispatch. It also showed examples of performance tests that highlighted the variability in throughput improvements. How does the Capacity Planner plan for this uncertain environment? Let's look at a real world example of Hyper-threading in action.

Case Study

Problem

Users are complaining about response time on their IE browser during daytime hours. The server does not seem to be fully utilized. No response time or transaction metrics are available, but the following information was provided to the capacity planner:

The CPU is configured as a standard

Production server (Siebel01). A listing of the server's configuration was provided using statistics from Windows metrics (Table 3). MFR1 is a similarly configured server which will be used for comparative purposes. Siebel01's CPU utilization (fig. 4) shows that the server is busy, but does not exceed 85% during peak hour. One would assume that there would be additional CPU resources available.

Could the workload be constrained by other resources, such as I/O or memory? Fig 5 shows that CPU is the major component of response time for the workload in question: IE browser (labeled NT-INET). This suggests that the solution to the problem lies in the CPU.

Table 3. Server Configuration

Computer	OS	Identified Model	System Model	Motherboard Model	CPU Model	Clock Rate Mhz	Max. Processors
MFR1	Windows2003 5.2.3790	DELL 2650@3200	Dell Computer Corporation PowerEdge 2650	Dell Computer Corporation 0D4921	GenuineIntel Intel(R) Xeon(TM) CPU 3.20GHz	3182	4
Siebel01	Windows2000 5.0.2195	DELL 2650@3200	Dell Computer Corporation PowerEdge 2650	Dell Computer Corporation 0D5995	GenuineIntel Intel(R) Xeon(TM) CPU 3.20GHz	3189	4

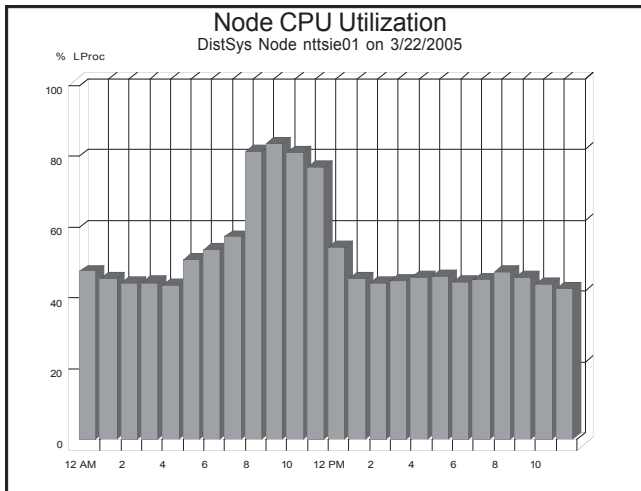


Figure 4. CPU Utilization for Siebel01

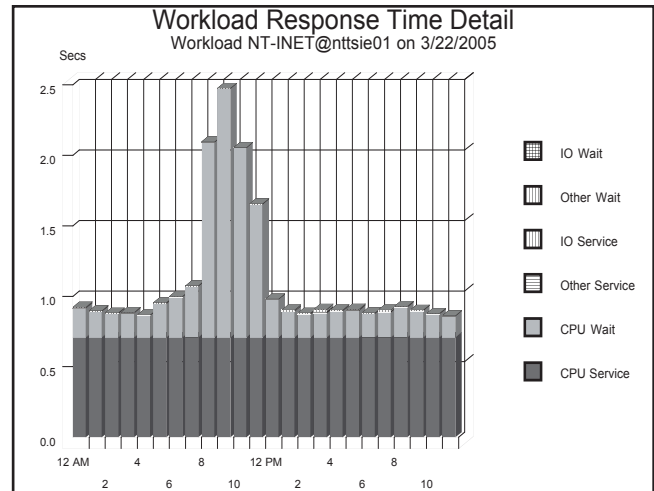


Figure 5. Components of Response time for the Internet Explorer Workload

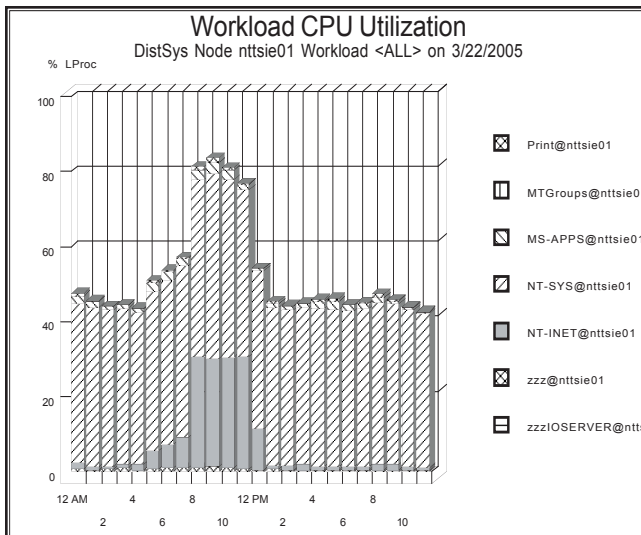


Figure 6. CPU Utilization of Workload in Siebel01

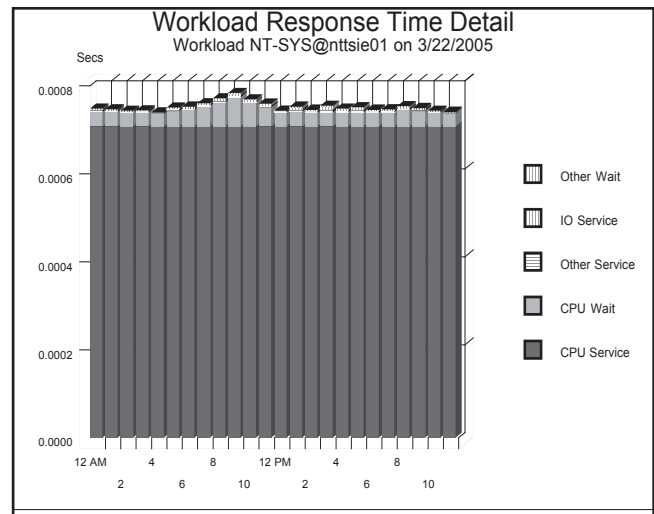


Figure 7. Components of Response time for NT-SYS

Could other workloads be contending for CPU resources with the IE browser? The next two graphs certainly suggests this possibility. There is a large NT System workload that is present most hours of the day (fig 6). Since this server does quite a bit of file server functions, this would be expected. The NT-SYS workload executes at high priority, so one would expect that it would get the CPU resources it requires. The response time graph shows that this is indeed the case: response time shows a very small CPU wait component (fig 7).

However, the key question still remains: why is the IE browser workload experiencing CPU wait when there seems to be CPU resources available ?

Analysis

Further analysis of system statistics (Table 4) still does not show cause for concern, although one starts to question why Q Length is higher than one would

Table 4 Server CPU Utilization statistics

Computer	CPU Model	Vendor	OS	Q Length	CPU Util	Out Of
					%	%
MFR1	2650@3200	DELL	Windows	2.75	150.99	400
MFR2	2650@3200	DELL	Windows	1.67	103.34	400
Siebel01	2650@3200	DELL	Windows	4.32	323.62	400

Table 5. Detailed Server CPU Utilization Report

Computer	Processor	Physical ID	CPU Util	Out Of	User Time	Priv Time	DPC Time	Interrupt Time	Idle Time	Ctx Switches	System Calls
			%	%	%	%	%	%	%	1/sec	1/sec
MFR1	[Total rows: 4]		150.99	400	126.48	24.51	1.88	1.91	249.01	3346.42	11806.63
	Processor 0	0	37.41	100	30.27	7.14	1.18	0.54	62.59		
	Processor 1	3	37.44	100	32.16	5.28	0.21	0.33	62.56		
	Processor 2	0	37.71	100	31.52	6.19	0.39	0.65	62.28		
	Processor 3	3	38.42	100	32.54	5.89	0.1	0.38	61.57		
Siebel01	[Total rows: 4]		323.62	400	62.85	260.76	0.62	0.46	76.38	255831.84	113365.45
	Processor 0	0	82.94	100	6.98	75.96	0.21	0.08	17.06		
	Processor 1	3	79.21	100	16.55	62.65	0.13	0.13	20.79		
	Processor 2	0	83.5	100	20.87	62.63	0.2	0.17	16.5		
	Processor 3	3	77.96	100	18.45	59.51	0.09	0.09	22.03		

expect. It is not until the Capacity Planner looks at the Physical Processor report that the problem becomes apparent. Looking closely at the Physical ID column, it becomes clear that there are only two physical processors on Siebel01 since there are only two unique ID's. The system has Hyper-threading turned on and is reporting 4 logical processors to the OS! Instead of CPU usage of 323% out of 400%, or 4 CPU's, it is really executing on two. This has huge implications on the ability of the server to handle large number of transactions! The MFR1 server is configured in a similar manner.

However, since the server is less busy, there is a much smaller quantity of context switches and system calls.

The ability of Siebel01 to accomplish 323% CPU Utilization on a 2 processor machine would seem to imply that it is getting at least some benefit out of hyper-threading. The high rate of system calls and context switches, along with the large amount of CPU Wait would imply that CPU is getting scarce. The question is: when did the CPU run out? Without end user response time and transaction measurements, this question is difficult to answer. However, by looking at the synthetic response time graph (fig 5), one can see that the "knee-of-the-curve" seems to occur at 7:00AM or 1:00 PM.

In summary, the degradation in response time can be explained once the shortage of CPU resources is uncovered. The workload is CPU-constrained and there is a large quantity of higher priority work executing in the system.

Proposed Solution

Now that the issue is known, the problem can be resolved by adding the appropriate amount of resources. Since this workload is important to the company (hopefully the users are not just surfing the net), and the high priority NT-SYS workload cannot be moved or reduced, the capacity planner is left with the option of adding more CPU resources. The problem is

how much and when. Management has forecast a 10% growth per month. Using this growth as the main driving force, an analytical model was used to choose the correct server size and forecast the server's useful life.

Table 6. Baseline Model CPU Report

Computer	Operating System	CPU Model	Vendor	CPU Utilization %	%Utilization out of	Queue Length	Throughput trans/hr
Siebel01	Windows	2650@3200	DELL	323.62	400	5.83	9950706.74

The first step in modeling is to choose the proper interval to model. In this case, a model was built using the peak hour of day (March 22 @ 9:00AM). The results of the baseline model is then calibrated by comparing the results of the model with the actual measurements for the chosen time interval. Since the workload is CPU bound, more attention is focused on CPU metrics. Actual measurements from Table 4 are comparable to results from the model listed in Table 6. The model shows that approximately half of the browser workload's (NT_INET) response time is due to CPU Wait (Table 7).

Table 7 Components of Response Time (Model).

Workload	Total Response Time	CPU Service Time	CPU Wait Time	IO Service Time	IO Wait Time	Other Service Time	Other Wait Time
	sec	sec	sec	sec	sec	sec	sec
MS-APPS@Siebel01	0.03	0.02	0.01	0	0	0	0
NT-INET@Siebel01	1.36	0.71	0.65	0	0	0	0
NT-SYS@Siebel01	1.24	0.71	0.53	0	0	0	0

The next step in the modeling process is to upgrade the processor to reduce the CPU bottleneck. An upgrade to a Dell 6600 4-way Xeon processor @3 GHz seems to be sufficient. (Table 8) Note that the Dell 6600 is actually slower than the 2650. However, it does allow a maximum of 4 processors, whereas the 2650 can only contain 2 CPU's. With Hyper-threading turned on, the model shows a CPU utilization of 326 % out of a possible 800. Again, 4 physical processors are being presented to the OS as 8 logical CPU's.

The processor upgrade eliminates

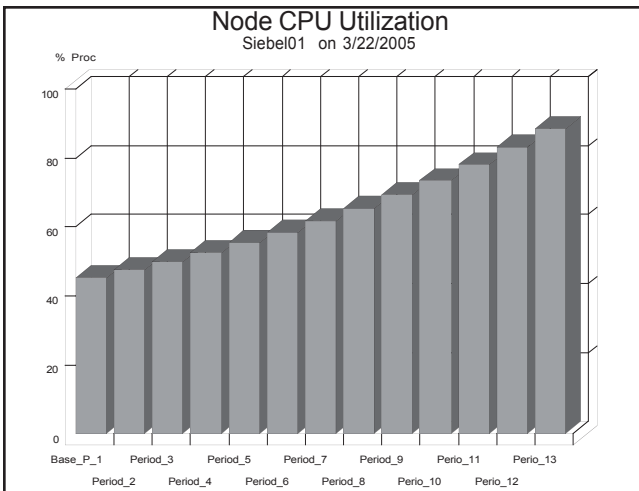


Figure 8. Best-case scenario - CPU forecast

the CPU bottleneck for all workloads running on the system. (Table 9) The end user should see a response time improvement of over 40%!

Growth Forecasts

The analytical model shows that the CPU upgrade will address user complaints during peak processing hours. Next, the capacity planner must understand how long this resource will continue to meet user demand. This is the tricky part!

Management forecasts call for a growth rate of 10% per month. Applying this growth to the current model results in the following forecast. It appears that the upgrade will be sufficient to meet user demand for 10 months if users can live with a 15% response time degradation at the end of the period. (Fig 8 and 9) Note that the upgrade gave them a

Table 8. CPU Report for Proposed Upgrade

Computer	Operating System	CPU Model	Vendor	CPU Utilization	%Utilization out of	Queue Length	Throughput
				%			trans/hr
Siebel01	Windows	6600@3000	DELL	326.99	800	3.28	297243.48

Table 9. Components of Response Time (Proposed Upgrade)

Workload	Total Response Time	CPU Service Time	CPU Wait Time	IO Service Time	IO Wait Time	Other Service Time	Other Wait Time
	sec	sec	sec	sec	sec	sec	sec
MS-APPS@Siebel01	0.02	0.02	0	0	0	0	0
NT-INET@Siebel01	0.72	0.72	0	0	0	0	0
NT-SYS@Siebel01	0.72	0.72	0	0	0	0	0

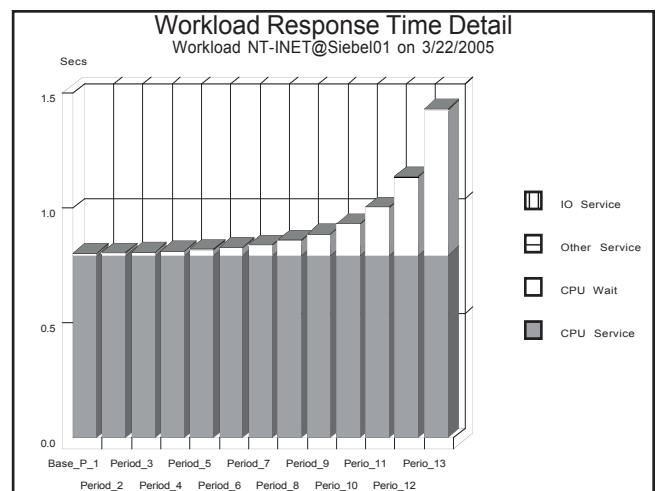


Figure 9. Best-case scenario - Response time forecast

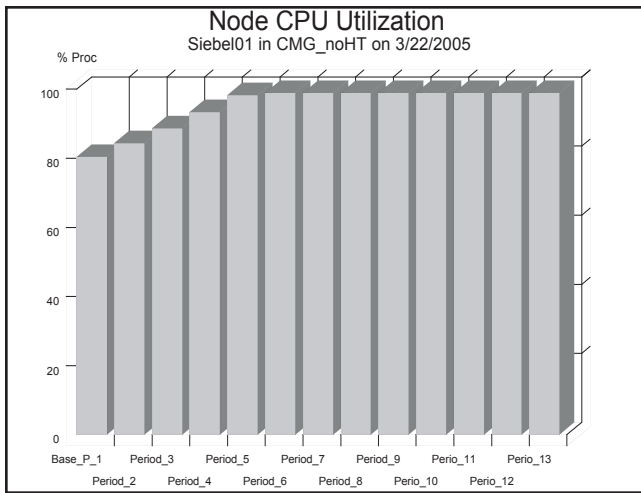


Figure 10. Worst-case scenario - CPU forecast

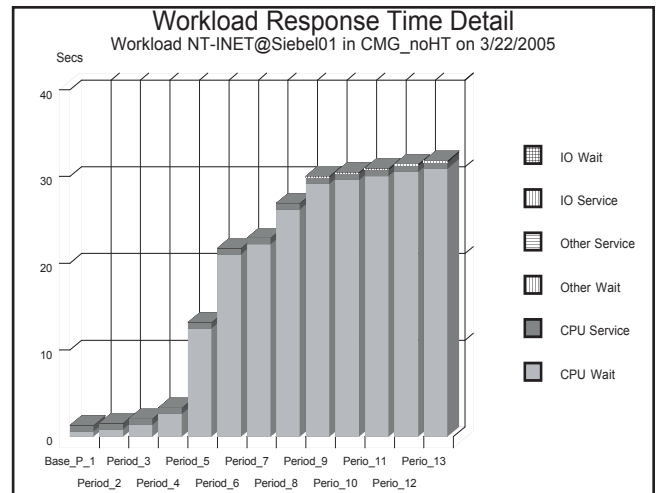


Figure 11. Worst-case scenario - Response time forecast

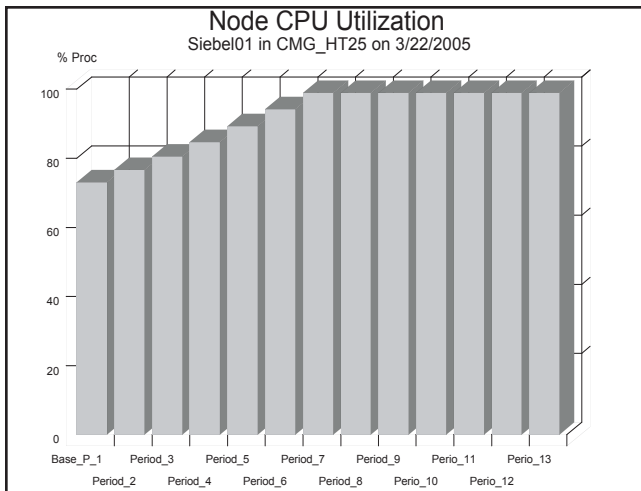


Figure 12. Most-likely scenario - CPU forecast

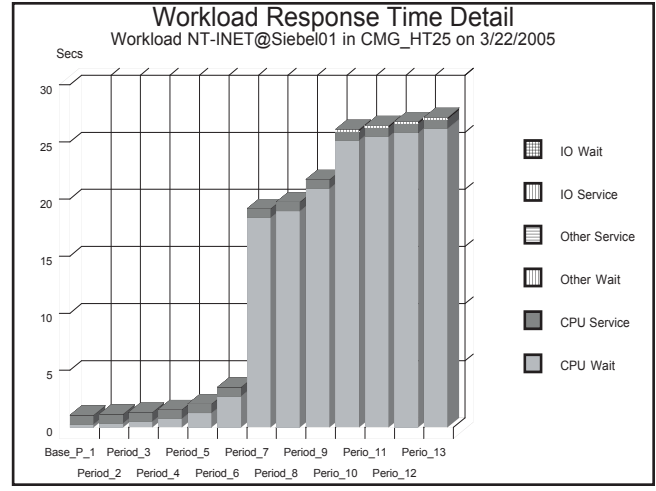


Figure 13. Most-likely scenario - Response time forecast

40% improvement, so this seems to be a reasonable situation. However, this model assumes that Hyper-threading will continue to deliver maximum benefits. In other words, the model is working on the assumption that 4 physical processors will be able to do the work of 8. What if the application can only realize partial or no benefit at all?

The next scenario models the worst-case, where Hyper-threading offers no improvement. In this scenario, the workload is allowed to execute on 4 processors with no hyper-threading benefits. This scenario calls for a processor upgrade after 30% growth. (Fig 10 & 11)

The last modeling iteration illustrates the most likely scenario. This model shows hyper-threading effectiveness of approximately 25%. The results call for an upgrade after 40% growth. (Fig 12 & 13)

At this point, it would be worthwhile to consolidate the results of the modeling runs to show the differences between the different assumptions. (Fig 14 & 15)

Showing the best-case and worst-case scenarios adds an important dimension to the forecast by documenting the potential risks in the forecast. The CPU Utilization graph illustrates the ability of hyper-threaded processors to sustain higher transaction rates. In the best case scenario, CPU Utilization does not reach 100% over the planning period. However, the most-likely scenario shows that hyper-threading will only add a few months to the life of the server. This is borne out in the Response Time chart (Fig. 15). If the limit for response time is set to 2 seconds, Hyper-threading will extend the life of the server for only 2.5 months. The importance of taking response times into account is illustrated in this example. Response times degrade at least 1 period before maximum CPU utilization is reached.

Recommendations

The case study illustrates the importance of the following:

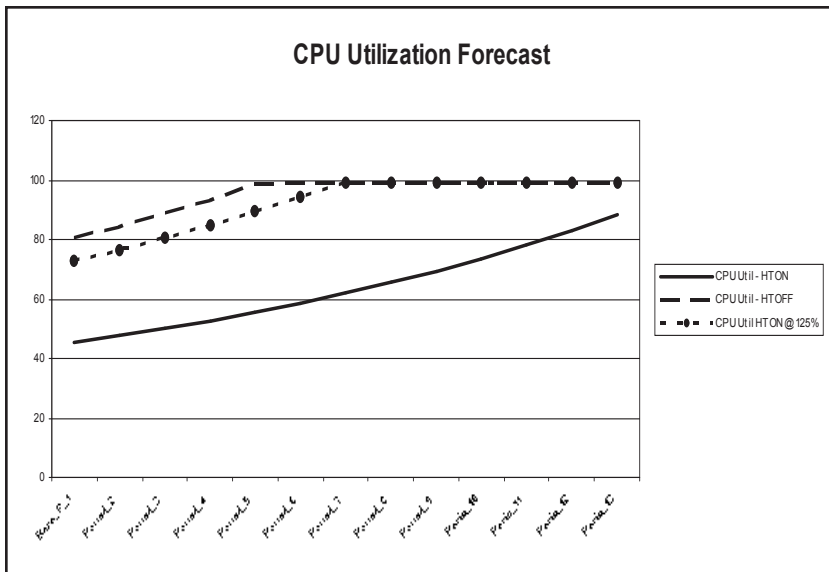


Figure 14. CPU Forecast with Sensitivity Analysis

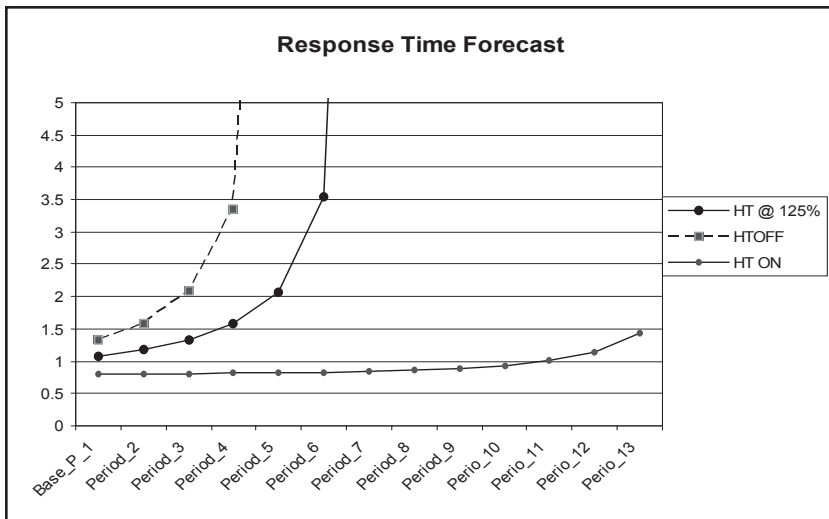


Figure 15. Response Time Forecast with Sensitivity Analysis

1. As internal measures of performance become less clear, the use of external measurements become more important. Traditional metrics, such as CPU service time and CPU Utilization, are ambiguous in a hyper-threaded system. In order to understand the quality of service provided, one must measure end user response time and business transaction counts.

2. The Capacity Planner must have an accurate source of information about the physical configuration of servers. ITIL best practices calls for the creation of a Capacity Database which contains information useful for the capacity planner and maintained automatically through interfaces with Change Management processes.

3. More than ever, the available capacity on a server depends on the application's ability to take advantage of hardware facilities. Due to parallel execution and the vagaries of OS met-

rics, it may be possible to deliver more than 100% CPU utilization! Application profiles, which document an application's use of hardware resources, becomes of paramount importance. These profiles must be reviewed with every major change in OS or application code.

4. Resource forecasts must take into account the variability inherent in the performance of logical processors and VM's. The use of sensitivity analysis reminds managers of this variability and highlights the best-case / worst - case scenario.

Software-Based Virtualization

Hyper-threading improves the throughput of an Intel processor by presenting the OS with double the number of logical processors. This allows the OS to dispatch work across more threads, thereby reducing the amount of idle time in the CPU. Software-based Virtualization builds on this concept by creating entire logical systems, complete with logical CPU's, memory and I/O devices. Virtualization at this level provides not only utilization improvements, but also the potential to improve availability and serviceability.

In order to understand the structure of Virtualization software, it is helpful to understand its niche in the marketplace. Virtualization is used to improve Total Cost of Ownership (TCO) in Windows and Linux environments. In so doing, Virtualization has been used in the areas of: availability management, load balancing, scalability, capacity on demand, parallel processing and simplified systems management. Server Consolidation is

the area where Virtualization enjoys widespread usage and acceptance. The ability to isolate VM's from each other allows consolidation of mixed workload systems without fear of unexpected or harmful interactions. Since each VM has its own complete virtual system environment, there is no need to share resources such as DLL's, registers, and memory structures.

Availability management is another popular area for Virtualization. It is fairly expensive to keep dedicated backup servers configured and online. However, with the magic of VM, multiple servers can exist on the same physical machine. Fail-over and hot-backup configurations can now be available at reasonable cost. Development and test environments are another variation of this concept. Instead of having dedicated test servers, multiple virtual system images can be stored in a physical server for use as needed.

Of course, Virtualization does have costs and disad-

vantages. There are new costs for VM software and the physical resources required to run the environment. Additional management complexity is also introduced because all the original OS images are still in place, and there are new VM OS administrative tasks.

tion, it is necessary to understand which processes are used by an application, and how much CPU, memory and disk are consumed by these processes. These metrics are harder to come by in a virtualized environment for two reasons. The first issue has already been discussed in the previous section: Hyper-threading introduces an uncertainty as to the real capacity of the system. The second issue revolves around the timekeeping mechanism used by the OS. In hosted VM environments, such as VMWare, guest machines usually provide inaccurate processor active times. In fact, any metric which uses the system clock becomes suspect. This of course includes all utilization metrics. To understand the reason for this inaccuracy, it is necessary to delve into interrupt timer mechanisms.

Every computer system has a timing mechanism. In fact, there are usually multiple timers available to handle the many requirements for some form of timekeeping. Time and date functions are kept by the CMOS TOD (Time of Day) clock, while timing functions are provided by timers located in other parts of the hardware. The OS uses these timers to provide a multitude of services, ranging from playing music to calculating CPU utilization. With so many different timers, it is fair to assume that every OS would differ in their use. However, one key assumption made by most operating systems is that time marches on at a constant rate. Using this assumption, a timer is set to generate an interrupt at a set frequency. For Windows, this frequency can be anywhere from 50 to 1000 times per second. This constant arrival rate of interrupts can be used to time the duration of events, like processor active time and process CPU consumption. For example, CPU active time is simply a Performance counter that is incremented in value every time an interrupt occurs and the CPU is busy. CPU Utilization is calculated using the formula:

$$\text{CPU Utilization} = \frac{\{\text{number of interrupts} / \text{freq rate}\}}{\text{time interval}}$$

This elegantly simple mechanism works until the VMM swaps out the guest OS and the flow of time is suspended!

VMWare can affect the flow of time in a guest machine three ways:

1. An interrupt may occur when the VM guest is swapped out. If the system is not too busy, the VM guest will be swapped in to take the interrupt. If not, the guest will miss the interrupt, and a chunk of time will be lost.
2. In a highly utilized system, dropped interrupts may occur so often that the guest falls far behind.

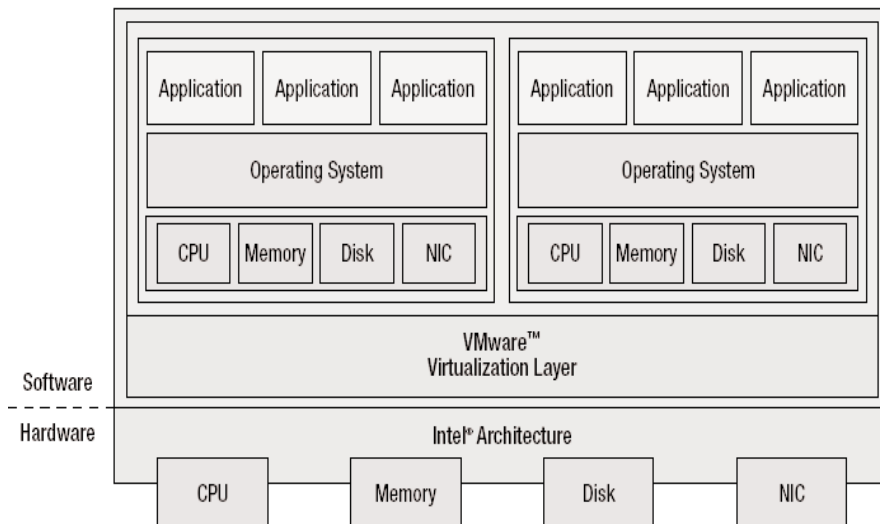


Figure 16. VMWare Pictorial (from VMWare3)

In addition, there are peculiarities involved in metrics coming from a VM hosted system. These will be covered in a later section.

VMWARE ESX Technical Overview

It would be useful at this point to focus on a popular implementation of Virtualization in order to study the characteristics of the breed. VMWare exemplifies the vendors that participate in this niche. Their ESX Server provides the management interface between the guest machine (VM) and physical system resources. This allows the abstraction of physical resources for each logical partition. Each virtual system environment uses standard operating systems and application software for individual tasks (Fig. 16).

The VMware™ virtualization layer, located between the hardware and software on a system, allows the creation of virtual machines equivalent to standard Intel® x86 Architecture. The virtualization layer allows for multiplexing of virtual machines on a single physical system with low overhead impact to system performance. Each virtual machine utilizes the virtual CPU, virtual memory, and virtual I/O devices provided by the virtualization layer. [VMWARE3]

Measuring VMWare systems

Capacity Planners are often asked to provide resource consumption for specific applications. This is especially important in servers where multiple applications are running. In order to measure the resource consump-

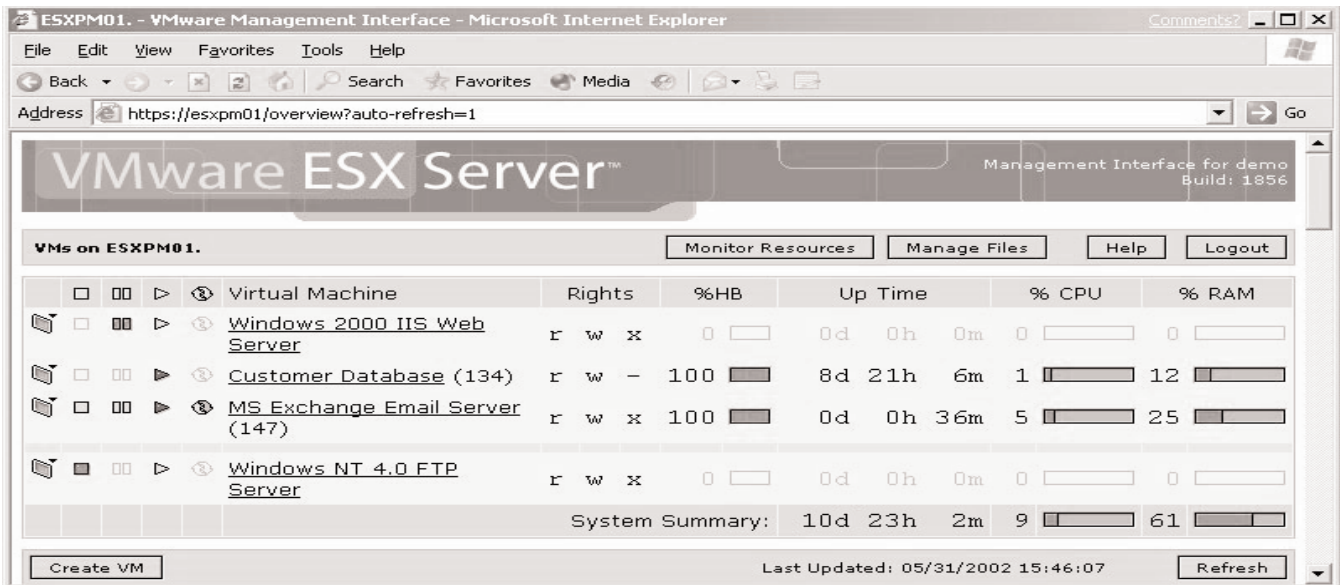


Figure 17. VMM Console Screen (from VMWare3)

In cases like this, VMM will increase the interrupt rate. Time speeds up!

3. VMM tries to sync up with the system TOD clock. This process usually takes place at one minute intervals. If the VM guest is far behind, pending interrupts will be cancelled, and the timer is reset.

Given the vagaries of timekeeping within a VM guest, it becomes clear that the most accurate measurements must come from an external source. Fortunately, the VMM kernel provides some of the key metrics. Reports from the VMkernel (Fig. 17) provide information on the actual amount of CPU and memory given to each VM guest. However, this level of data provides high level resource utilization only. For example, it does not keep track of the resource utilization for individual processes. In order to determine the CPU utilization of a workload, one must first determine the CPU utilization of a guest machine (from the VM kernel), compare that to the utilization from the guest's perspective, look at the reported CPU consumption for each process, assume that the ratio of skew is the same (a BIG assumption) and apply the ratio to obtain the estimated process utilization.

Scheduling and Availability Management are the other areas impacted by clocking issues. Some hardware implementations of VM allow the sharing of processors between guest machines. It may also allow VM's to operate only within certain time windows: i.e., 9 AM to 5PM. Care must be taken to ensure that jobs are scheduled to run during times when the domain is actually active.

There are other factors which must be taken into account when the Capacity Planner attempts to determine the health and capacity of VMWare systems:

1. The allocation of CPU resources to VM guests is

determined by many factors. Guests are allocated CPU shares, which control the over-all amount of CPU given to the partition. However, these allocations are modified by parameters such as CPU affinity, min/max allocations, etc. For example, it is common in multi-processor systems to lock the VMM usage to CPU 0 and prevent all other guests from accessing this CPU. This allows the VMkernel to access CPU whenever it needs it, thereby assuring that system services are not bogged down by user workloads.

2. The servicing of timer interrupts require a context switch for each OS guest. Some applications, like Apple's QuickTime, will raise the timer frequency to 1000 Hz. In a VMWare server with 20+ partitions, this can result in a lot of context switches!

3. VMWare has its own memory management mechanism which is independent from its guest OS. These mechanisms, which include ballooning and swapping, are used in addition to the normal paging and swapping mechanism in the guest OS. In systems where Real memory is over-committed, the Capacity Planner must carefully observe the VMM swap rate.

4. VMWare uses a SCSI device driver to represent I/O devices in each guest OS. This may be completely different from the topology of the actual physical devices. Although I/O counts are not affected by the timing issues discussed in the previous section, there may still be a difference between logical and physical I/O. Thus, I/O tuning may need to be done at multiple levels.

Useful Metrics and Graphs for VMWare Systems

After absorbing all the issues discussed in the previous section, the Capacity Planner may be asking what metrics should be closely monitored in a VMWare system.

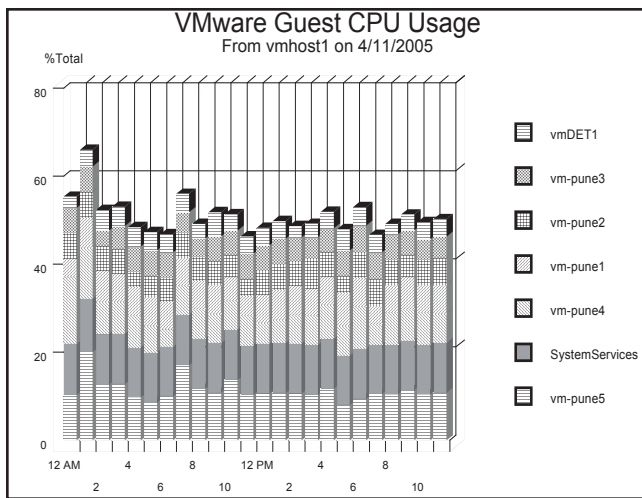


Figure 18. Aggregate CPU Utilization

Some of these key metrics are presented in the next few graphs.

Server Consolidation is a popular area for Virtualization, and the CPU graph presented in Fig. 18 is a good example of why. Many, if not the majority, of Windows servers use less than 15% of its CPU resources. Fig. 18 shows the consolidation of 6 servers into one physical system. Consolidation ratios of 20:1 are common in the industry. Note that VMWare overhead, labeled SystemServices, is shown as another graph item. This reminds viewers that the use of VMWare does consume CPU resources and has a capacity impact.

Physical I/O activity is important to monitor from a consolidated perspective. It is important to know how much I/O a VM guest is performing and the aggregated total I/O of all guest systems (Fig 19). Response times of each physical device can then be compared to the over-all activity rate.

Fig. 20 presents the amount of memory used by each VM guest. As the aggregate value approaches the amount of Real memory installed on the server, it becomes necessary to watch ballooning and swapping activity. An increase in the amount of context switches may be a warning sign for over-committed memory.

As more system images are placed into a consolidated server, it becomes very important to monitor the number and type of shares allocated to each image and the usage of these shares.

Observations and Recommendations

As we conclude our discussion of Virtualization, it is important to summarize these points:

1. Measure resource utilization from both the VMM kernel and the VM guest. If there is no VMM, look at both logical and physical processor utilization. Remember that hyper-threaded systems will report twice the number of CPU's to the OS. As such, it becomes critical to have an up-to-date hardware

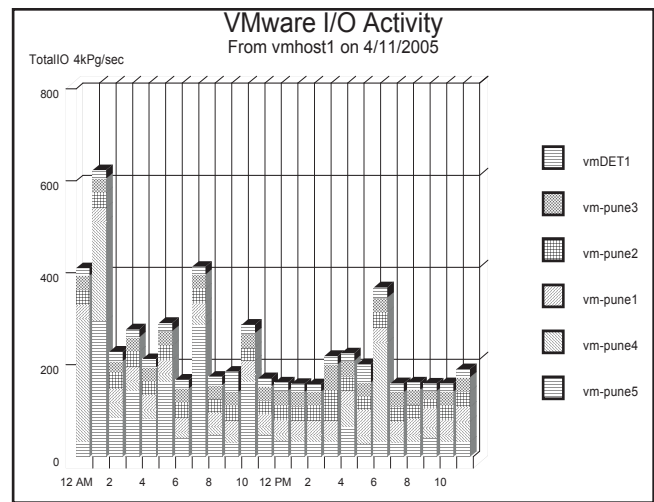


Figure 19. Total I/O for each VM Guest

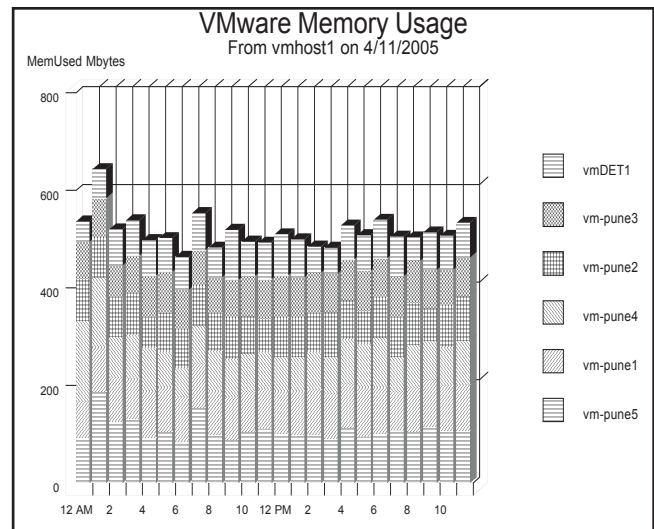


Figure 20. Total Memory Usage for each VM Guest

profile. One must know with great certainty what type of hardware is being used by an application.

2. Due to clocking issues, utilization measurements from within logical systems are suspect. More than ever, the total capacity of a system is dependent on the application's workload characteristics.

3. The total amount of throughput available from a physical processor depends on the application's ability to take advantage of virtualization and hyper-threading. The maximum throughput of a system may be less or more than 100%. Thus, it becomes very important to profile your major applications to determine capacity requirements. These profiles must be reviewed with every major change in application code or OS version.

4. Measure application response time as close to the user's perspective as possible and understand what is included and what is not.

5. Resource forecasts must take into account the variability inherent in the performance of logical processors and VMs. One should make a point of including sensitivity analysis for important forecasts.

Perhaps the most important lesson to come out of all this discussion is the importance of comparing measurements from within virtual systems with an external source. Ideally, this external source would measure events at the user's perspective. Figure 21 illustrates this ideal scenario. The Performance model is built from information gathered from three primary sources. Collected system metrics represent data collected from within the virtual guest, from the VM kernel, and external application logs. Business metrics represent data that maps internal system metrics with business transactions. This would imply that key transactions from each business application are profiled, measured, and logged. Finally, the third data source comes from external measurements of transaction volumes and response time. The accuracy and utility of Performance Models have always depended on the data used to create them. This is even more true in the Virtual world.

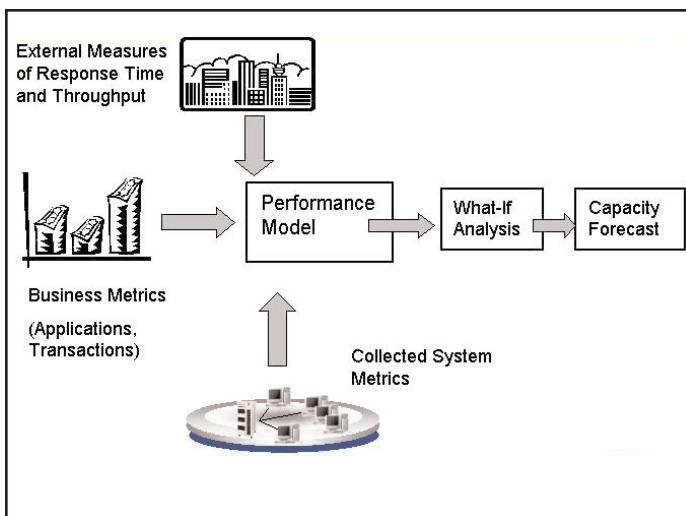


Figure 21. Ideal Monitoring environment for Virtual Systems

Summary

This paper visited a few islands in the vast sea that is Virtualization. The popularity and widespread acceptance of products from various vendors suggests that Capacity Planners will be dealing with these concepts for years to come. In recent press releases, Intel announced features in its latest chip architectures which are intended to facilitate the implementation of Virtualization. This is encouraging news because the tighter blending of hardware and software may address some of the issues discussed in this paper. But then again, I am known as an incurable optimist !

At the start of the paper, the question was posed: should one address Virtualization technology today, or

simply wait for the next technology wave. The answer seems to be that Virtualization has been with us for awhile; it is here to stay; and the forecast is for a more complex environment, not a simpler one. So our recommendation is: Batten down the hatches and go bravely into the storm !

References

[Borozan] John Borozan, Microsoft Windows-Based Servers and Intel Hyper-Threading Technology, Microsoft Windows Technical Article, April 2002, <http://www.microsoft.com/windows2000/docs/hyper-threading.doc>

[Binstock] Binstock, Andrew Intel Virtualization Technology: A Primer Intel Software Network

[Creasey] Creasy, R. J., The Origin of the VM/370 Time-sharing System, IBM J. RES. DEVELOP. VOL. 2 SEPTEMBER 1981, <http://www.research.ibm.com/journal/rd/255/ibmrd2505M.pdf>

[Fernando] Fernando, Gene Challenges of Capacity Planning in a Virtual Environment, CMG Measure-IT, April 2005

[Friedman] Mark Friedman. Hyper-threading - Two for the price of one? , Measure IT, March 1, 2003 http://www.cmg.org/measureit/issues/mit01/m_1_1.html

[Intel1] Hyper-Threading Technology on the Intel Xeon Processor Family for Servers http://www.intel.com/business/bss/products/hyper-threading/server/ht_server.pdf

[Intel2] Hyper-Threading Technology Architecture and Microarchitecture http://www.intel.com/technology/itj/2002/volume06issue01/art01_hyper/p01_abstract.htm

[Marr] Deborah Marr, Frank Binns, David Hill, Glenn Hinton, David Koufaty, J. Miller, Michael Upton, "Hyper-Threading Technology Architecture and Micro architecture," Intel Technology Journal (Hyper-Threading Technology), Volume 06 Issue 01, February 14, 2002

[Parikh] Parikh, Sunish and Thomas E. Martinez, Intel Corp, Dual Processors, Hyper-threading Technology, and Multi-core Systems, <http://www.intel.com/cd/ids/developer/asmo-na/eng/200677.htm>

[Smith] Smith, Roger Intel Xeon Processors and Itanium 2 Processors – Picking the Right "Tool for the Job" Intel Software Network

[Sun] Sun Microsystems Sun Fire Enterprise Servers and the UltraSPARC IV Processor http://www.sun.com/servers/highend/whitepapers/Sun_Fire_Enterprise_Servers_Performance.pdf

[Vianney] Duc Vianney, Hyper-Threading speeds Linux, IBM DeveloperWorks, <http://www-106.ibm.com/developerworks/linux/library/l-htl/>

[VMWare1], http://www.vmware.com/pdf/esx2_best_practices.pdf, Best Practices for VMware ESX Server 2, VMWare Whitepaper

[VMWare2], http://www.vmware.com/pdf/esx2_performance_implications.pdf

[VMWare3], http://www.vmware.com/pdf/Solution_Blueprint.pdf, Server Virtualization Solution, VMWare Whitepaper

[Yiping] Yiping Ding, Performance Implications of Hyper-Threading http://www.bmc.com/offers/performance/whitepapers/docs/2003/Perform_Impl_of_Hyperthreading.pdf

Trademarks

Intel Pentium and Xeon are registered trademarks of Intel Corporation

Linus is a registered trademark of Linus Torvalds

Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

MVS, VM, and zOS are registered trademarks of IBM Corporation

VMWare is a registered trademark of VMWare, Inc

All other marks and names mentioned herein may be trademarks of their respective companies.