SERVER CONSOLIDATION

Perceived Performance and Virtual Operating Systems

Timothy Mangan
TMurgent Technologies
April 1, 2005

Table of Contents

Abstract

Many businesses are using, or evaluating, virtual Operating Systems in their server centers. Whether used for server consolidation, redundancy, or flexibility, the benefits are too great to ignore. And yet little is known about the performance aspects of these virtual environments.

The majority of consolidation projects today are focusing on replacing a number of existing underutilized systems with a more modern server using virtualization. The "conventional wisdom" in the field is to keep the number of virtual machines per server low and incorporate only servers that have low resource utilization.

This paper looks at the two leading "virtual machine[1]" architectures, from Microsoft and VMware. In the paper, tests are described and results published that provide insight into how these systems perform under scale.

Experience in the related field of Terminal Servers provides guidance in steps that can be taken that will allow an enterprise to use more resource intense virtual machines in their deployments. Tests results are also included in this paper that make use of existing 3rd party tools that are worthy of consideration in virtual environments.

Introduction

Virtual machines are typically considered for one or more of three reasons; server consolidation, redundancy (ease of hardware system replacement), and flexibility (the ability to roll out new servers when demand warrants).

In a virtual machine deployment, software is added to a host server that allows it to run one or more complete guest operating systems on top of the host operating system. These guest operating systems act as complete systems (either desktops or servers) that run abstracted from the underlying hardware.

When used for server consolidation, the benefits include cost savings in staffing, hardware, software, and facilities (space, and/or power)[1]. With the reduced budgets that most enterprises have been living with since 2001, many are continuing to maintain servers beyond their life expectancy. These servers are running on outdated hardware that is expensive to maintain and for which replacement parts are more scarce or impossible to find. An example might be a 666mhz uni-processor server. The typical server consolidation project consists of taking a handful of these systems and deploying them on a more modern multiprocessor server. By using virtual operating systems, the server software can be moved in entirety and run without risk of integration with software from other servers.

[1] In this paper the term "virtual machine" is used, without capitalization, as a generic term. References to the Microsoft product Virtual Server are appropriately capitalized.

Virtual machine technology is also used for the purpose of redundancy. While backup strategies, including complete server imaging (such as Norton Ghost and others) can be effectively used when well understood, these solutions require identical hardware on the new server. By using virtual machines, the backup image of the guest OS is a simple file to be backed up and restored on any hardware platform thanks to the hardware abstraction. Features such as VMotion (from VMware) extend automation of this redundancy concept.

Virtual machine deployments also allow for flexibility needed by some companies. Perhaps an enterprise needs more web servers at the end of the year and more productivity servers during the rest of the year.

And it is impossible to talk about virtual machine deployments without mentioning the trend toward deploying the hardware as blades as well. The combination of using blades and virtual machines has become very popular, especially with server consolidation.

The leading providers of server virtualization software today are VMware[2] and Microsoft. VMware is the industry leader, with Microsoft playing catch-up by buying a company with a competing product[3] and "beefing it up".

VMware offers two versions, GSX and ESX. The ESX product is more robust, however requires its own operating system for the host (their own brew of a Linux kernel), whereas the GSX product will run on either Microsoft or Linux commercial offerings.

Microsoft offers two versions as well, called Virtual PC and Virtual Server. As the names imply, the former is for use on desktop PCs and the latter for servers. Although results of testing are not shown in this paper, tests performed in the lab indicate that either Microsoft product can run virtualized servers with equivalent performance. The main difference between the two Microsoft versions come is in the console interface. The Virtual PC console is rudimentary.

Server Consolidation Projects

Currently, the "conventional wisdom" on consolidation projects is to focus on migrating existing, vastly underutilized, systems. Typical of these are file or web servers that are now running on single CPU systems with clock rates under 1ghz. Usually, candidates for consolidation are using less than 20% of the CPU even on these outdated systems. In the time since these systems were deployed, server performance has grown significantly. To illustrate, typical TPC-C benchmarks (which standardize a method to measure the number of transactions-per-minute) were in the 10,000 TPM rage in 1996. Current TPC measurements are in the hundreds of thousands range for typical servers, and over a million for the high end[3]. Consolidation rates for these kinds of underutilized servers

[2] VMware was acquired by EMC Corporation in 2004. As of this publishing it continues to operate as a separate subsidiary under its own name.
[3] Microsoft Corporation acquired this software from Connectix in 2003.

of up to 3-4 virtual machines per CPU (in the new server) are reported. Given the improvements in computing performance over the time span, more should be possible.

It is reasonable to expect that with success on these early projects, businesses will more aggressively approach server consolidation; incorporating more resource intensive servers if they can prove that they can maintain the same level of responsiveness and stability in their test labs and pilot rollouts.

These more aggressive deployments will face some of the same resource contention issues that are faced today in larger Terminal Server deployments. Many of the proven solutions to resource issues in the Terminal Server field should prove useful in these virtual machine deployments as well.

Virtual machine Architecture

A virtual machine deployment consists of software that is placed on a server with a host operating system which allows multiple, independent, and complete, operating systems (virtual machines) to run on the same hardware at the same time. Each virtual machine thinks it is running on dedicated hardware and is unaware that the hardware it thinks it sees is an abstraction, nor is it aware of the host OS, or other virtual machines co-resident. By tradition, the software providing this abstraction is called a "*virtual machine monitor*"[4], although this paper will use a more appropriate term "*virtual abstraction layer*" (as far more than monitoring is involved).

The virtual abstraction layer supplies an interface at the bios level. It presents the virtual (guest) OS with virtual hardware that looks like a minimal and common hardware system. The virtual hardware would have a single CPU, standard hard disk(s), mouse, keyboard, com ports, and a NIC. The virtual hardware presented was selected by the vendors because it is easily recognized and supported by all the major operating systems that might be used in the virtual machine, and due to efficiency. It is reported, for example, that in some cases using a virtual SCSI disk in the guest OS improves performance over a virtual IDE disk – even when the underlying physical disk is IDE.

A goal of these virtual machine architectures is to minimize "instruction emulation" to reduce overhead. Most instructions carried out within the virtual machine guest OS are directly executed on the underlying hardware, even if intercepted by an additional virtual bios layer. The remaining instructions are executed by the virtual abstraction layer (primarily hardware operations that require synchronization with other virtual machines needs), or emulated on real hardware when required. An excellent description of this process and other historical techniques at virtualization is given in reference [1]. It is for these reasons that one cannot run a virtual 64-bit operating system on a 32-bit processor.

[4] This stems from the IBM use of the term, and is still in use in architectures such as Xen.

The virtual machine architectures of both VMware[5] and Microsoft Virtual Server are summarized in Figure 1 below.

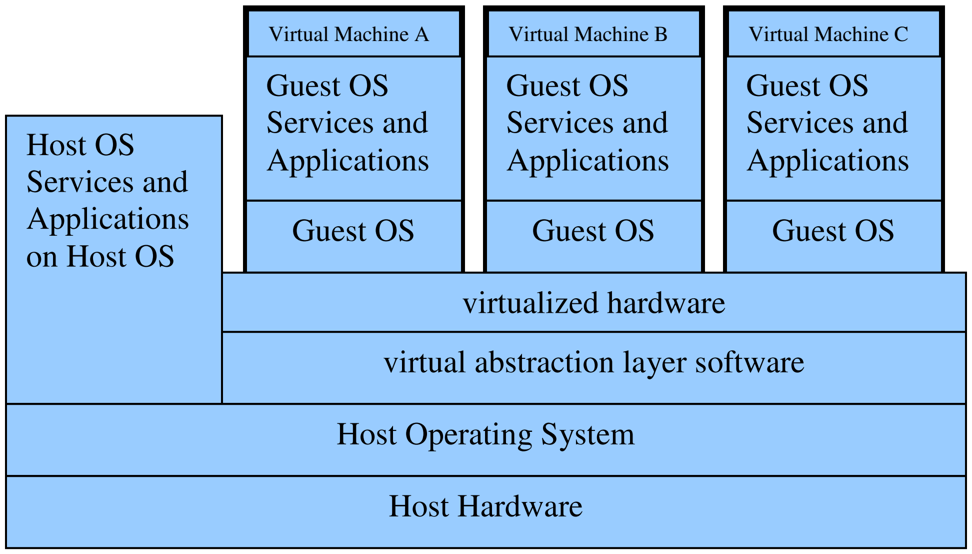

Virtual Machine A
Virtual Machine B
Virtual Machine C
Guest OS Services and Applications
Guest OS Services and Applications
Guest OS Services and Applications
Host OS Services and Applications on Host OS
Guest OS
Guest OS
Guest OS
virtualized hardware
virtual abstraction layer software
Host Operating System
Host Hardware

Figure 1 - Architecture of a virtual machine

This form of virtualization is referred to as full-virtualization. Each virtual machine is implemented as a separate thread of the virtual abstraction layer software process on the host operating system. Because any given thread in the host OS can only run in one processor at a time, the Guest OS is presented with a virtual machine that looks like a single processor system. The host hardware may (and usually is) a SMP (symmetric multi-processor) system. Much as any given thread might run in different processors at different times, this virtual machine thread may run in different physical processors on the host hardware at different times also. There is no need to peg certain virtual machines to certain processors (called setting processor affinity), and it is possible (even reasonable) to have more virtual machines than processors. In addition, no more concern need be given to hardware technologies, such as *HyperThreading,* than would be given any other application on the host OS.

In practice, the testing performed for this paper indicates that the ratio of virtual machines to processors is dependent only on the sum of the resource requirements (primarily CPU and memory) of the virtual machines.

[5] This diagram is appropriate for the GSX server. The diagram normally depicted for VMware ESX server usually leaves out the host OS. This is no more than a marketing pitch, as a specialized version of a Linux kernel is used in the ESX solution. This paper contends that the diagram in Figure 1 is appropriate for the GSX as well.

Final notes regarding virtualization architecture:

- Full-virtualization is not the only option. See the section on Additional Alternatives regarding para-virtualization.
- Two of major CPU manufacturers, Intel and AMD have each announced plans to produce chip versions that would allow virtual machines to run at the hardware level. It was the chip support by Intel that made virtual memory operating systems efficient through the use of a special register – similar well thought out support for virtual machines within the CPU might produce similar results. While potentially <u>very</u> interesting, these plans are potential futures and it is not clear that they will work without special support by the host operating system either; these plans should not be cause for an enterprise to take a "wait and see" attitude toward virtual machines. .

Virtual Machine Performance Testing

The term "Perceived Performance" was first described by Tim Mangan in his paper "*Perceived Performance" [3]* in 2003. In that paper, Perceived Performance, differentiated from Computational Performance, is described as "*a methodology where one analyzes the system with a goal of improving user productivity by focusing on issues that affect the performance of the system as perceived by the users using it*".

One of the key new benchmarks used in analyzing server performance is the "Perceived Performance Profile". The technique of making measurements that describe the performance of a system as viewed by a system user was It was in a 2004 follow-up paper sponsored by triCerat and also called "*Perceived Performance" [5]*, that a method of visualizing the performance, called Perceived Performance Profile, was first explained.

In this paper the perceived performance profile will be used to demonstrate performance test results. These tests serve to help us understand server performance of virtual machines. Some notes about the test setups are in order:

- Test results in this paper were created from test data on a server that is typical of those used for servers today. All the tests presented in this paper were performed on hardware consisting of hardware that included dual 2.6Ghz Xeon processors with HyperThreading enabled (essentially a quad processor system), 4GB of RAM, and local SCSI disks[6]. The hardware was running Microsoft Windows Server 2003 for the host operating system.

- To virtualize the guest operating systems, either Microsoft Virtual Server 1.1 EE or VMware GSX 3.1.0 was used (as indicated on the graphs). In the figures that follow in the paper, eight virtual machines (using a mixture of operating systems) were running. Each virtual machine was configured with a minimal but sufficient RAM so that paging inside the guest OS was minimized. For all but the last two test cases shown, seven of the virtual machines were running but essentially idle. The loading of the eighth virtual machine was varied and performance of that virtual machine was measured. The eighth virtual machine, where performance measurements were made, was a Windows 2000 server.

- Tests were conducted in a controlled lab at TMurgent Technologies. Domain, Name, and licensing servers were all external but local resources. Where terminal services were utilized, local profiles used.

- The CPU priority performance product used in some tests (where noted) was a pre-release of Simplify Resources 4.0 from triCerat[7].

[6] Although SAN storage is proving more and more popular in server deployments, testing in this paper uses local, non-RAID, storage for more consistent test results.
[7] Funding to support this paper came from triCerat.

The Perceived Performance Profile on a Native Server

The Perceived Performance Profile (which will herein be referred to as **P3**) shows the amount and variability of delay felt by a user to typical operations that they do. These operations themselves take little CPU resources, but are affected by everything else going on in the server at that time. In the case of virtual machines, the **P3** can be affected by operations in any of the virtual machines or host server itself. Figure 3, shows the **P3** for a typical server, such as would be used in a Web or Terminal Server deployment, without any virtualization in use (see test notes in previous section).

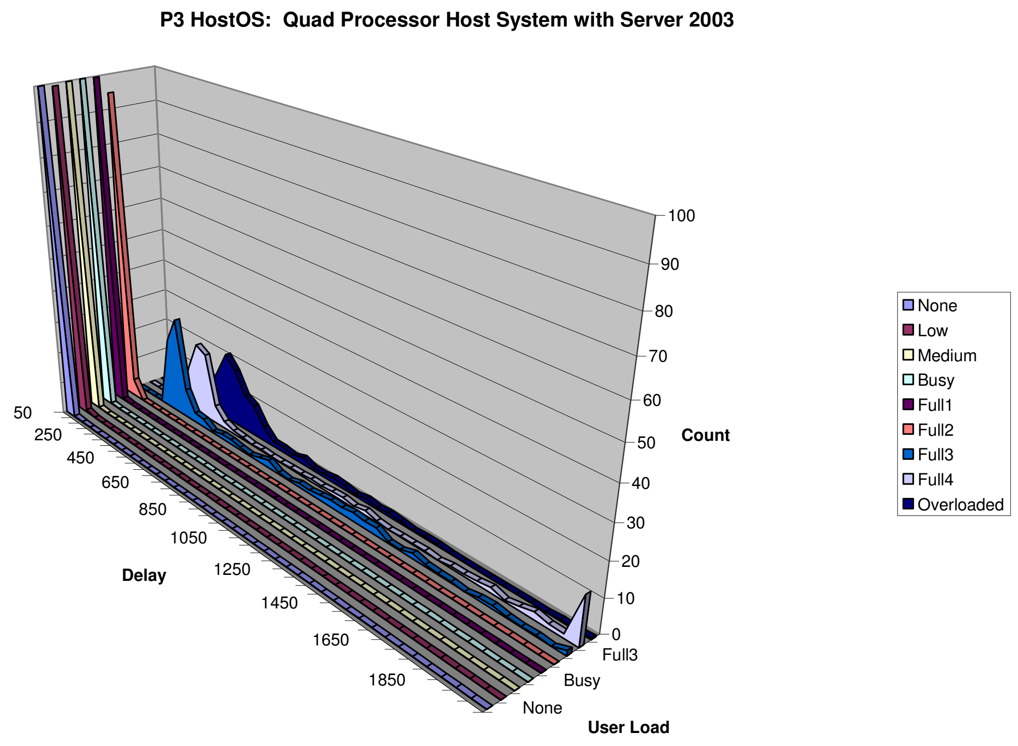


Figure 2 - P3 for typical server

Each colored row in the **P3** indicates the likelihood that the user will feel delays of a given duration. For example, when user loading on the server is low, the user experiences delay of under fifty ms nearly 100% of the time. As server loading is increased, the user experiences increased delays averaging one-half to three-quarters of a second. This graph provides visual support to the "common wisdom" that you keep your servers running at no more than 50% of capacity (the salmon colored row).

Even more import than the average delay, according to this research [3], is the variability that occurs in delay. Consistent visual response to user actions is the most important factor to user productivity. This delay variability is clearly visible in the **P3** for the three last rows of the graph, where delays can occasionally reach two seconds!

Simple Virtual Machine Performance on a Lightly Loaded System

Of interest is to show the effect of moving users from a real server onto a virtualized server. Figure 3 and Figure 4 present **P3**s for a virtual machine from each vendor. In the Figure 3 case we used Microsoft Virtual Server to host the guest OS; in Figure 4 VMware's GSX Server software was used to host the guest OS. In both cases, the CPU load was placed inside a virtual guest operating system and p Performance measurements were made against user sessions within the guest operating system.

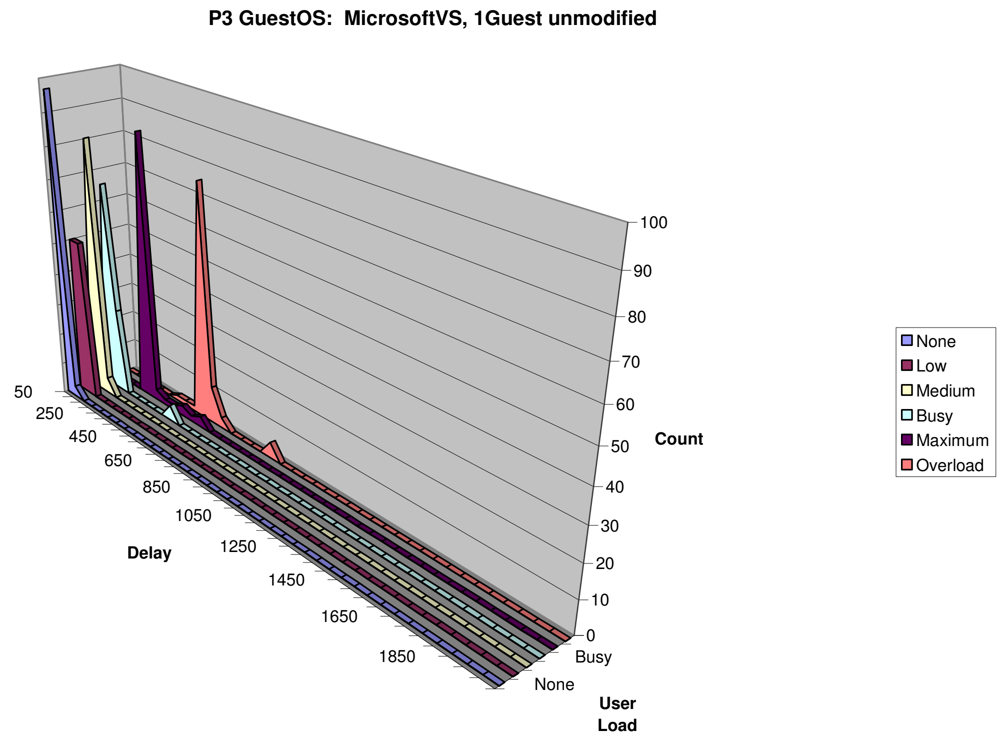


Figure 3 - P3 for Single Virtual Machine (Microsoft Virtual Server)

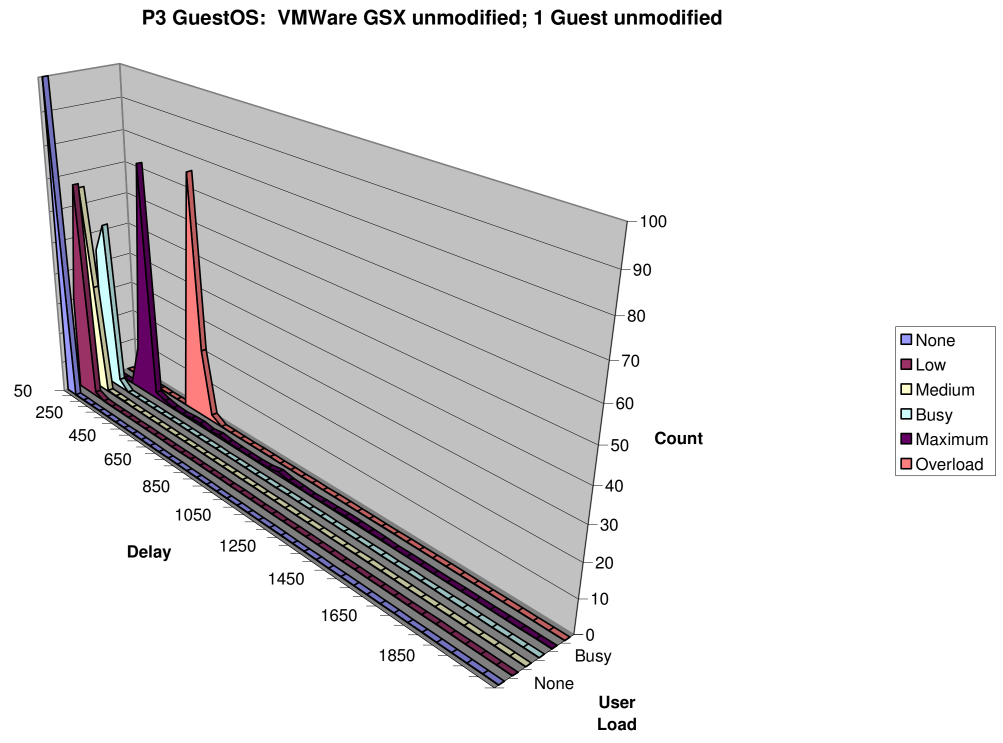


Figure 4 - P3 for Single Virtual Machine (VMware GSX Server)

These **P3**s show what happens to system performance as load is added to the guest OS server. The results are similar for each virtual machine host.

These results show that a user experiences similar average performance degradation to that of non virtualized server. We do see a decrease in delay variability, however, that is due to the remainder of the host system being under-utilized.

Result

Guest OS performance degrades as load on the virtual machine increases. Degradation starts at a lower loading level than a non virtual machine.

Because the virtual machine host limits the Guest OS to a single processor, the definition of "busy" means that the guest OS is only using ¼ of the total CPU on our 4 processor system. The implementations by both Microsoft and VMware GSX do not restrict the guest to the same processor each time, but only a single processor can run on behalf of a given virtual machine at a time[8]. Later in this paper results are presented showing multiple guest OSs being loaded up with users.

[8] In a VMware ESX server guest operating systems are given visibility to more than one processor. Perceived Performance results on the ESX server will need to be the subject of a future paper.

A better comparison might be to run these same two tests on a uni-processor system. Funding limitations prevented setting up such a test for this paper, however, the expectation is that under such a test, both the average and variability of delay would increase, possibly significantly, in the guest operating system case.

Before presenting additional results, there are some other observations should be passed along from the test experience. These observations are presented without backing data.

- Generally, virtual machines use about 5% CPU overhead.
- The Windows Task Manager, within the guest OS, is an unreliable indicator of CPU usage. Pauses in the virtual machine (due to contention by host OS processes or other virtual machines) are charged to whatever is running in the virtual OS at the time of the pause as CPU cycles consumed. It was necessary to develop other techniques to determine the actual CPU usage.
- Although some improvement is seen in delay variability for the GSX hosted OS in Figure 4 versus the Microsoft case, this was found to be unique to the under-loaded host server scenario that is *not* typical for what customers implement when consolidating.
- There are some non-intuitive results observed which brings into question the effectiveness of the "File Cache" of the Microsoft operating system. It may be possible that the effect of "double buffering" the cache actually improves typical operation. This has not been directly observed, nor has any test been proposed to prove or disprove it. It is mentioned here for possible future research.

Virtual Machine Performance with Resource Management

It is common, especially with Terminal Servers, to improve system performance of the native operating system by using add-on resource management products, such as triCerat's *Simplify Resources.*

But do these products work in the virtual machine? And where do you deploy them, in the host OS or server. In the next set of tests, light is shed on what happens when you add in these products, either within the guest OS or in the host OS.

Resource Management in the Virtual Guest OS

In Figure 5, we present the **P3** of the same test setup used for Figure 3, but with a resource management product added into the guest OS of the eighth server.

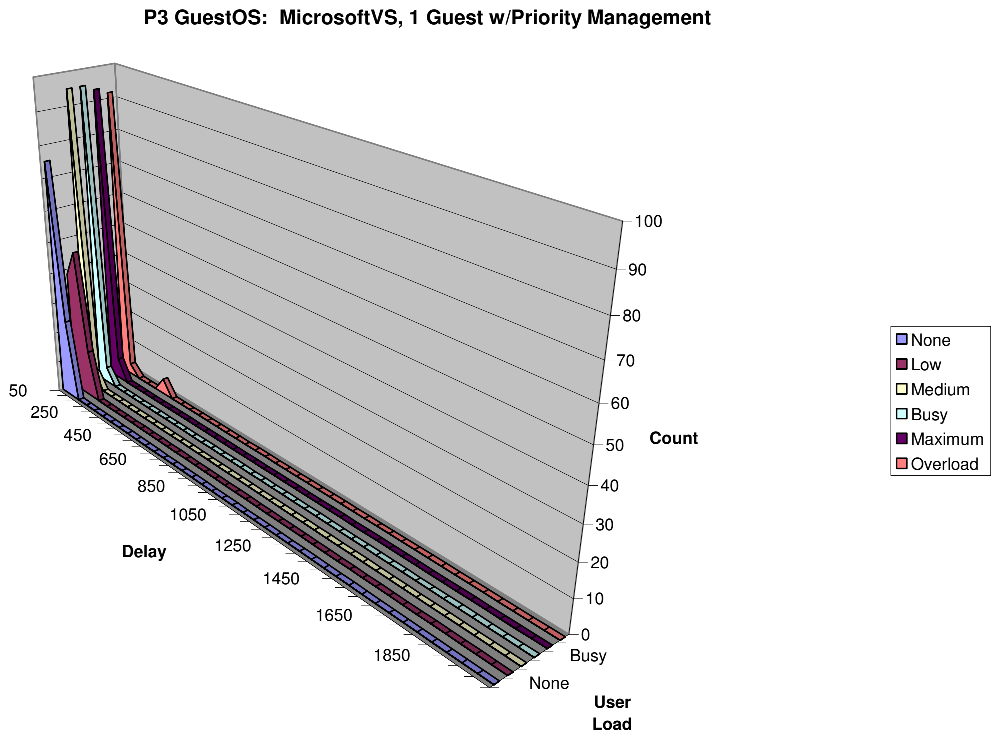


Figure 5 - P3 Single Virtual Machine with CPU management in Virtual OS

These results show that introducing resource management product to the virtual operating system improves the **P3**, pretty much eliminating the impact of virtual machine load. These results are mostly consistent with the results seen in the non-virtual environment.

Result

Virtual server performance is significantly improved with resource management tools added to the virtual OS.

Although test results are not included in the paper, it should be noted that the advantage of adding resource management software to perform prioritization in a single guest OS aids only the guest OS and not the host or other guest OSs[9]. For best results such software would be deployed in each guest OS.

Resource Management in the Host OS

What happens when you add such resource management software to the host OS instead of the virtual OS? It is important to note that documentation from both of the major vendors caution strongly against using add-on resource management products on the host

[9] Some Resource Management products support both CPU Prioritization and CPU Limiting. When CPU limiting is used inside a guest OS, the benefits are felt by users of the host OS or other guest OSs sharing the same hardware.

OS. The concern here is that all CPU cycles used by virtual machines show up in the host OS under different threads of the virtual abstraction layer process. If the resource management product interferes with the virtual abstraction layer processes because of the high level of resources consumed, virtual machine performance will degrade. In addition, as we shall see later, the virtual abstraction layer itself performs load balancing functions between virtual machines. At this time it is unwise for a third party product to interfere with the virtual abstraction layer processes.

However, the host server consists of much more than just the virtual abstraction layer processes. Even if no users ever log into the host OS, there are a variety of other occasionally resource consuming processes on the server, possibly including a Web Server (for virtual machine management), DNS, Licensing, Active Directory, licensing, and file services. Properly configured, resource management software on the host OS should improve the virtual machine performance.

The results of such a test are shown in Figure 6 below.

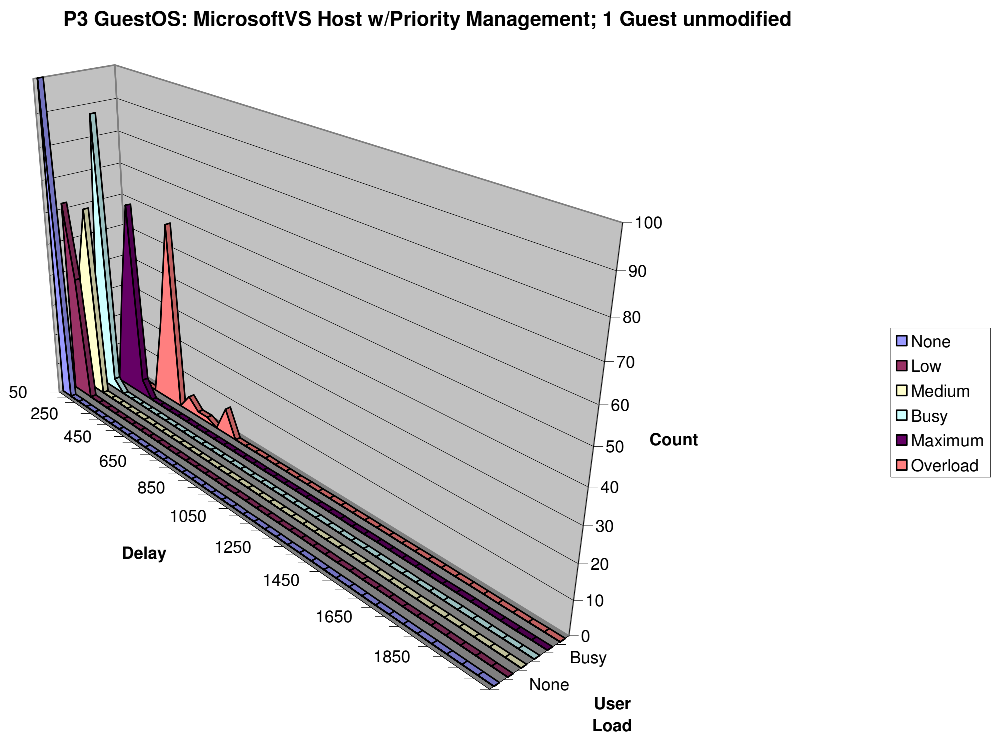


Figure 6 - Adding Resource Management to the Virtual OS

While the results show a slight improvement as the virtual machine load increases (about 100-200ms improvement in the overloaded case), the improvement is modest. The reader is cautioned that this is on a lightly loaded system, and the amount of improvement will rapidly increase as the overall server load increases. This is especially true if host OS processes become active, such as a console logon or placing the Active Directory Server on the host OS.

<u>Result</u>

Virtual Server performance is slightly improved with resource management tools added to the host OS. The amount of improvement increases with overall system load.

Resource Management in the Both OSs

In Figure 7, the results of adding resource management to both the host and virtual OS are presented. As expected, this scenario presents the best results of an optimized system.

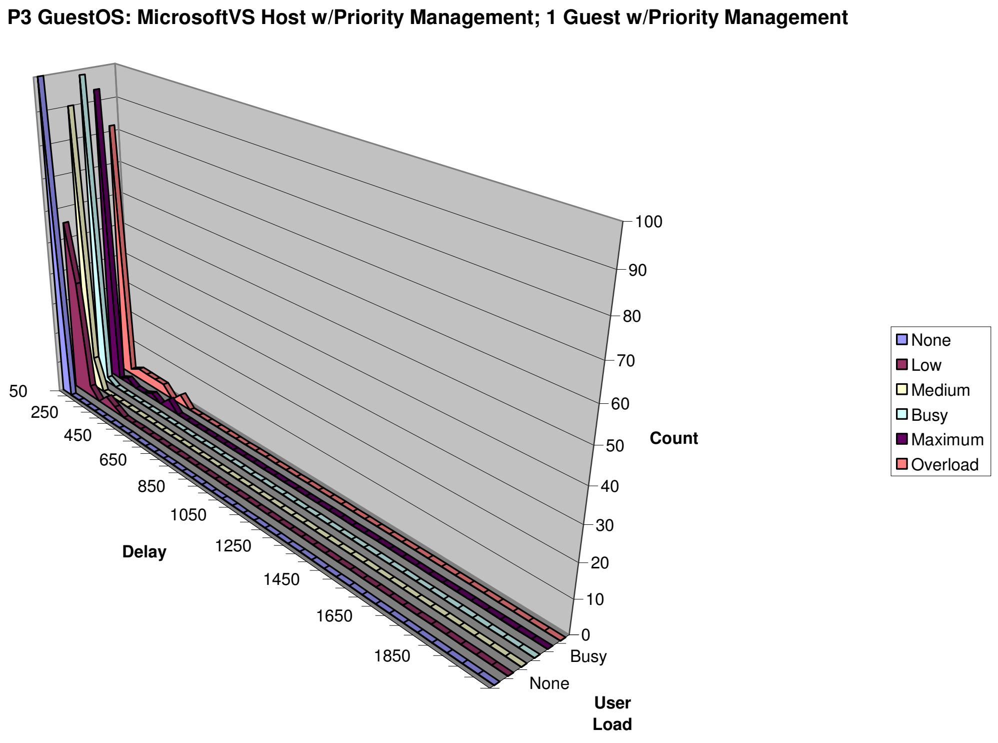


Figure 7 - Adding Resource Management to Host and Virtual OSes

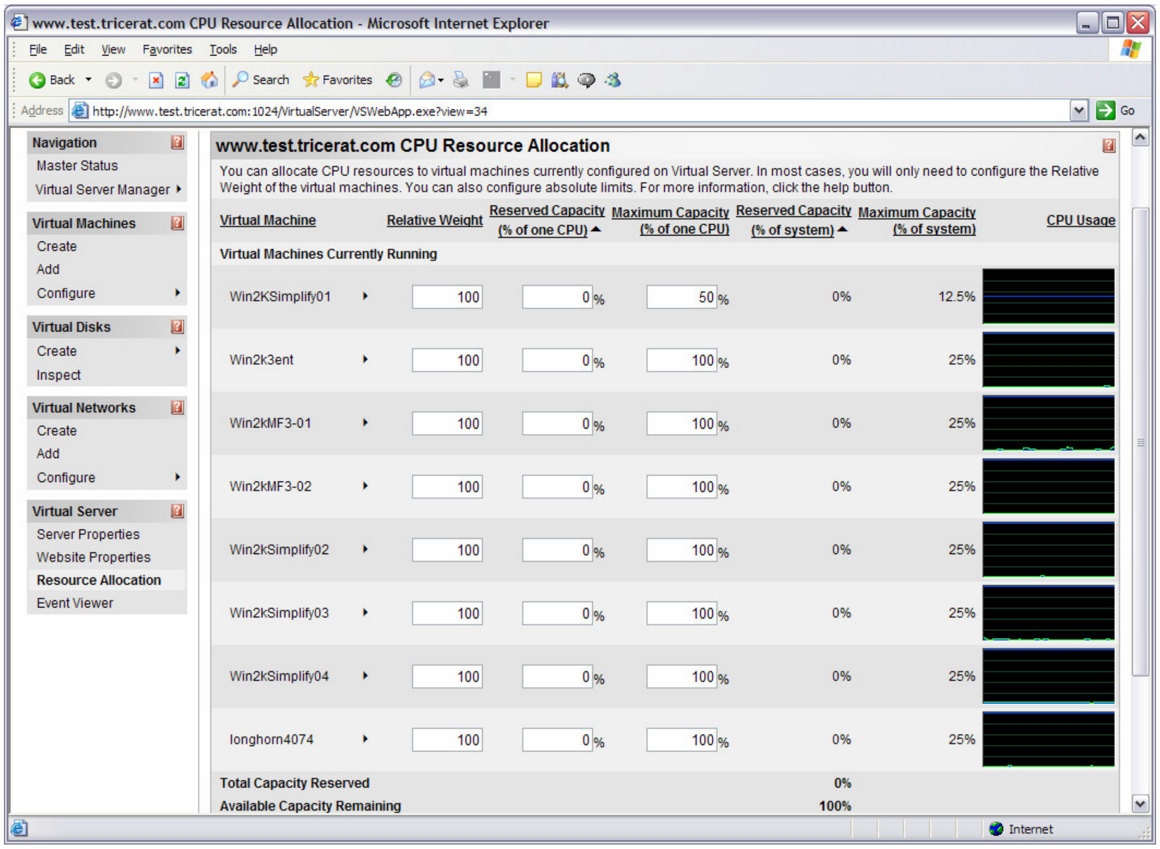
<u>Result</u>

Virtual Server performance is best optimized with resource management tools added to the host and virtual operating systems.

Enabling limits on Virtual Machine Instances

As mentioned earlier, the virtual machine can and does impose limitations on resource consumption when faced with limits. These limits can occur because of actual contention between virtual machines, or due to static constraints imposed by an administrator.

Under Microsoft Virtual Server, there is a configuration constraint in the form of a maximum CPU utilization for each guest OS[10]. Figure 8 shows the setup of such a static constraint on the server Win2kSimplify01. In the example that follows this virtual machine to use no more than 50% of a single CPU.



www.test.tricerat.com CPU Resource Allocation

You can allocate CPU resources to virtual machines currently configured on Virtual Server. In most cases, you will only need to configure the Relative Weight of the virtual machines. You can also configure absolute limits. For more information, click the help button.

Virtual Machine	Relative Weight	Reserved Capacity (% of one CPU)	Maximum Capacity (% of one CPU)	Reserved Capacity (% of system)	Maximum Capacity (% of system)	CPU Usage
Virtual Machines Currently Running						
Win2KSimplify01	100	0 %	50 %	0%	12.5%	
Win2k3ent	100	0 %	100 %	0%	25%	
Win2KMF3-01	100	0 %	100 %	0%	25%	
Win2KMF3-02	100	0 %	100 %	0%	25%	
Win2kSimplify02	100	0 %	100 %	0%	25%	
Win2kSimplify03	100	0 %	100 %	0%	25%	
Win2kSimplify04	100	0 %	100 %	0%	25%	
longhorn4074	100	0 %	100 %	0%	25%	
Total Capacity Reserved				**0%**		
Available Capacity Remaining				**100%**		

Figure 8 - Using CPU Constraints on Microsoft Virtual Server

As a side note, our investigation showed that this 50% constraint is imposed on the effective CPU utilization of the virtual machine. The effective utilization includes both the CPU used by the virtual machine, plus the overhead imposed in virtualization. This is typically about 5%. Thus a virtual machine that is using 20% of a CPU looks like 25% for the purpose of the constraint[11].

[10] The VMware GSX server does not have a similar constraint feature. As the results show, this is a good thing.

[11] Note that the task manager within the guest OS reports nearly 100% CPU utilization in this situation. These measurements are taken on the host OS.

50% Static Constraint on an Unmodified Virtual Machine

Performance tests within the constrained virtual guest OS were performed. Once again, this test is run with other virtual machines effectively idle. The test allows illumination on the effectiveness of the load balancing software within the virtual abstraction layer itself. The results of this test are shown in Figure 9.

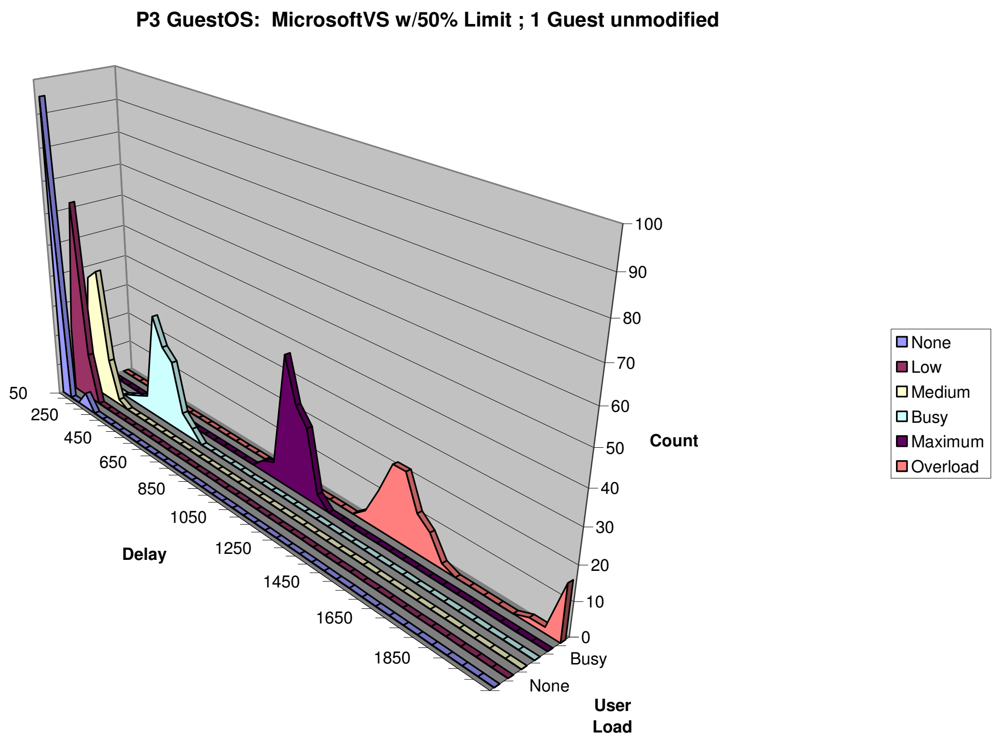


Figure 9 - 50% Constraint Imposed

As the **P3** shows, the performance experienced by users of the virtual machine drops considerably when load balancing occurs unless the virtual machine is very lightly loaded. Notice that even under “Medium” loading of the virtual machine – in which software is tuned to consume about half of the CPU available to the guest OS - the effect is felt.

Analysis shows that the virtual abstraction layer, having little insight into what is occurring within the virtual OS, is ill prepared to optimize using such a constraint. This is the reason that “conventional wisdom” is to use only lightly loaded virtual machines/servers and avoid these constraints. Being a static constraint, the example in Figure 9 serves as a “worst case scenario” for the effect of dynamic constraints that occur in loaded, multi guest OS situations.

Result

Poorly implemented virtual machine tools to load balance between servers cause very poor performance on a system.

50% Static Constraint with Resource Management tools

In the test results shown in Figure 10, we see that adding resource management help within the Virtual OS to the statically constrained system helps significantly.

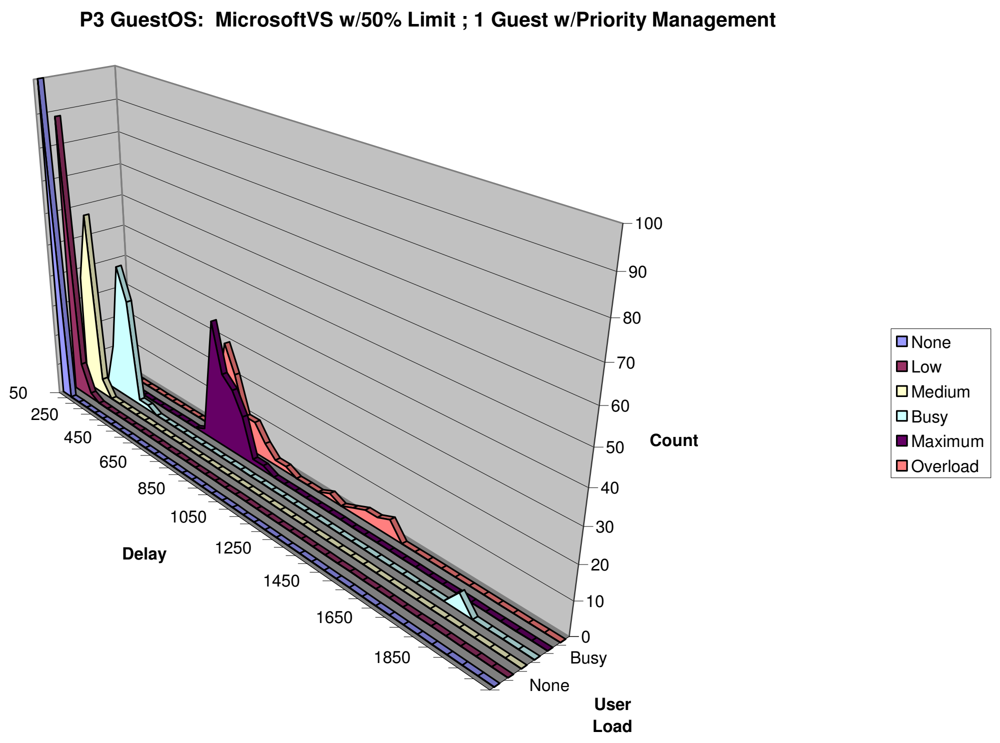


Figure 10 – 50% Constraint with SR Resource Management

While not necessarily an ideal **P3**, the results make the system considerably more palatable. It is considered that a one second delay is about the limit of user endurance for occasional delays.

Of course, the picture painted here in this example is a worst case scenario (designed to illustrate how the virtual abstraction layer software works) that should not occur on reasonably provisioned systems, even when loading is high.

In the real world, use of statically configured constraints such as this are not recommended. Allowing the virtual abstraction layer to dynamically arbitrate between contentious systems is the preferred method[12]. Thus, such a constraint on one virtual machine would only occur when the sum of the requested loads exceeds 100% of the overall CPUs. Even if the overall system runs at an average of 75% of capacity, load balancing limitations would only occur occasionally, and would not likely be long lasting.

By adding resource management software to the host OS and virtual machines, we can improve performance of a given loading scenario. Alternatively, we can also increase the loading of a server while maintaining **P3** by adding in resource management software.

Result

Adding Resource Management in the Guest OS alone significantly helps, but cannot cure, poor performance due to strict load balancing performed at the Host OS level.

Performance of Multiple Guest OSs with Heavy Loads

The static constraints imposed by the limited virtual OS tools do a poor job of mimicking the environment of customers consolidating multiple virtual machines. In the final two tests, the 50% constraint was removed and, using multiple virtual machines, a requested load ranging from 80% to 100% of what could be performed by the hardware was used. Again, we will show results with and without added resource management software.

Dynamic Contention without Resource Management Software

Figure 11, presents the results of **P3** without resource management software added. In scenario shown, the overall system CPU climbs from just over 80% to 100% as the user load of this virtual machine increases.

[12] VMware does support some less drastic host-side controls that behave better than the CPU limitation shown in this paper. Although not presented here, testing shows controls are better than nothing on the host OS, but because they do not affect other host software processes best results would be achieved by including a properly configured add-on management package to the host OS.

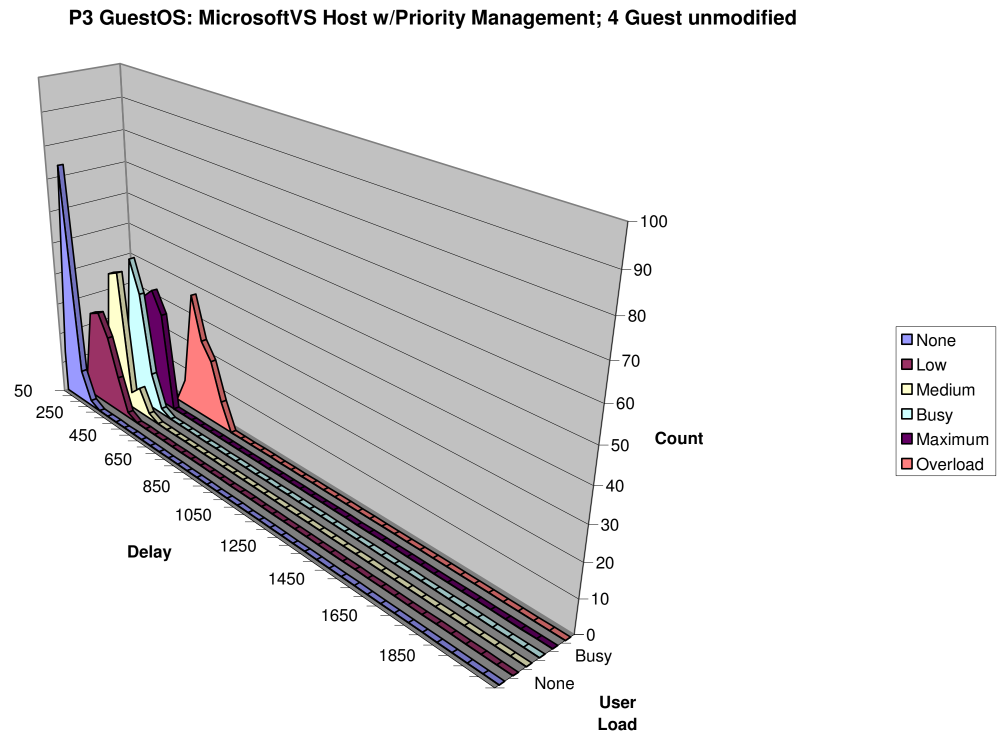


Figure 11 - P3 under Contention on Unmodified System

As expected, **P3** shows performance degradation as we reach the limit. From Figure 9 we know that if we added more loading on additional virtual machines, the performance drop-off would be more dramatic. Still, worse performance was expected in this test than the results we obtained.

After a bit of investigation, the conclusion is that this is a natural occurrence of dynamic contention. Static contention is typically implemented using timer, simply turning the virtual machine on and off regardless of what it might or might not be doing. Dynamic contention can take advantage of natural gaps that occur (for example waiting for a file to be read in) and is better able to balance the system.

Result

Dynamic Contention is preferred over Static Contention.

Comparing Figure 11 with Figure 2, it can also be seen that the point of significant performance degradation occurs at a higher overall CPU[13] when we are using virtual machines. This also, is due to this dynamic contention. However, it does not follow that one can improve the performance of a server by adding in virtualization. It must be

[13] Measured CPU, not necessarily "useful work".

remembered that these tests are not designed to present a fixed "useful workload", but to present a known CPU load. The overall "useful work" done by a native system running at 80% CPU is far less than multiple virtual machines running at a combined rate of 80% due to overhead.

Dynamic Contention with Resource Management Software

In the final test, results are presented for the case of adding resource management software to both the host and virtual OS in the dynamic contention scenario.

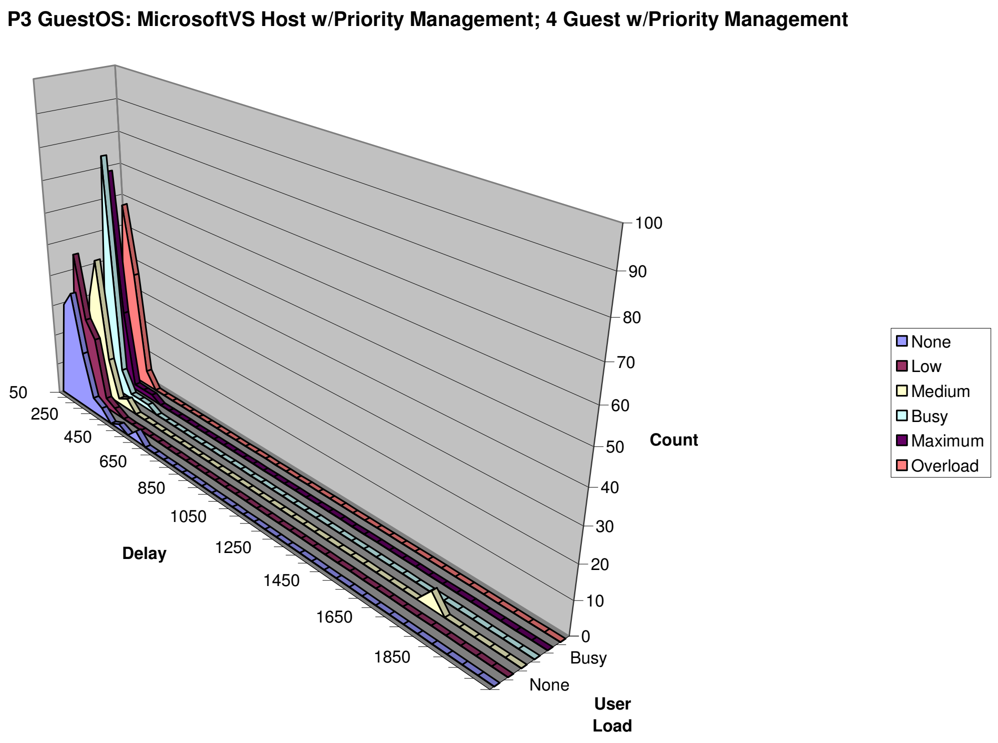


Figure 12 – under Contention with Resource Management Added

Figure 12 shows the **P3** results of this test. These results are not only very impressive, but the fully loaded case compares quite well with results of a fully loaded non-virtual system we have performed in the past.

<u>Result</u>
With proper add-on tools, Virtual Server performance can nearly rival that of real server performance.

Additional Alternatives

Virtualization is nothing new, and one should asses other uses of virtualization and other approaches to computing in the computer industry to aid in understanding this use. This section provides a sample of various other virtualization forms used in computing. It is not an exhaustive list.

Virtual Machine Monitor

Research in the 1960s lead IBM Corporation to release the first commercial use of virtual machines in 1972. VM allowed mainframe customers to run multiple operating system instances on top of the VMM base. Very popular in the 1970s, this approach fell out of favor when IBM introduced MVS, which finally allowed for multiple users (isolation), plus the falling hardware prices that spelled the doom of the mainframe. Much of today's operating system virtualization is based upon the same basic concepts, yet little can be found publicly from the Web. Yet some interesting nuggets can be found. This paper[6] from 1970 includes a description of how the mammoth 256K bytes of memory could be broken up into 64 4K pages!. Another interesting paper[7], written by VMware co-founder Mendel Rosenblum, ties this work in with VMware today.

Emulation

Emulation involves the interpretation of machine instructions on one hardware platform so as to emulate a different hardware platform. Examples include x86 DOS emulators on both Apple and Unix boxes popular in the 1980s and 1990s. Another example is how 16-bit software is emulated on 32-bit processors in the Windows OS via nvtdm today.

Instruction by instruction emulation is slower than virtualization. In today's virtualization, hardware is virtualized through device drivers or through API filtering. The underlying computer hardware architecture and machine instructions remain intact.

"Real Time" OSs

Developers of "embedded" systems, such as most intelligent communications gear of the 1980's and 1990's used operating systems termed "real-time" operating systems. Often, the non-hardware specific portions of these systems were developed and debugged by hosting the real-time OS within a lightweight thread of a UNIX operating system.

This involved what looks like today's OS virtualization (without the fancy management packages) where device drivers where essentially stubbed out or connected to a user console so that a human could "emulate" the device. While effective for their purposes, performance was not typically a consideration.

Terminal Server and Citrix

The Microsoft Terminal Server[14], especially popular in use with Citrix, provide a form of virtualization as well. In this case it is the user that is virtualized, or at least the

[14] Microsoft Terminal Server is made possible by technology licensed from Citrix. This underpinning virtualizes users into "sessions". Citrix also directly sells a popular add-on called Metaframe that enhances the user experience and provides management capabilities.

input/output devices associated with the user. Also called Thin-Client Computing, this allows an inexpensive and remote terminal to provide keyboard, mouse, and monitor to run software on a central server. This solution also carves up the server operating system into (mostly) independent sessions so that many users may run applications on the server at the same time. The goal in this environment is to reduce the cost of remote hardware and software (and maintenance) via centralization.

X
The X-windows system[8] under various UNIX platforms provide a similar, yet different, method to remote users. Under X, however, an application executes instructions on both sides of the remote interface. The goal under X was to remove some unnecessary processing on the central server and push it out to the remote client.

Softricity
Rather than virtualize an entire OS, Softricity SoftGrid[9] virtualizes the application. Where virtual operating systems do their filtering at the bottom of the (guest) OS, SoftGrid does the filtering at the interface layer between the application and the OS. This provides complete application isolation without the overhead of an additional OS.

Where OS virtualization allows the running of different guest operating systems on a single hardware platform, SoftGrid allows the running of different versions of software applications to run on the same operating system. In a heterogeneous environment (e.g. no Linux or Unix) virtualization at the application layer allows substantially more performance and scalability than virtual operating systems can ever deliver.

XEN
Xen[10] started as a project out of the Systems Research Group of the University of Cambridge (UK). Xen is an open source OS virtualization solution with the potential for greater performance than the architectures in use in the products tested as part of this paper. Those products leave the guest operating system fairly intact, using filters and virtualized device drivers. This is referred to as *para-virtualization* as opposed to *full-virtualization.* Xen requires access to the source code of the guest OS in order to rewrite portions. This allows for a more complete and streamlined integration.

Indeed, one of our biggest complaints in OS virtualization is that the guest OS does its job by putting up artificial barriers. These barriers create the ideal virtual environment so that applications cannot tell it is not running on real hardware, but also prevent the sharing of important information and control that is necessary to achieve performance and scalability. An example of how this limits performance is when a low priority task running in one virtual machine competes on equal footing with a high priority task in a different virtual machine. An example limiting scalability would be the inability to pool “free memory”.

XEN will soon be included in the standard Linux (2.6) distribution. While interesting from a technology point of view, the requirement of source code changes to the OS kernel will be quite a limiting factor (support by Microsoft notwithstanding). Given that the VMware ESX server essentially runs on a Linux base, the true value of Zen may be to provide VMware a boost in their commercial efforts (especially if Xen 3.0 supports guest operating systems that cannot be source modified).

GRID Computing

Grid computing is yet another way to get a days work done. Under Grid computing, multiple computers cooperate to perform work for a variety of users. To be successful, this requires a massive amount of software to coordinate and manage the activities. Applications, too, must be (re)written to take advantage of the grid. IBM [14] and others[11][12][13] are pouring many resources into the Grid initiative. Early success will come from large scale problems, [15] such as are solved using Super Computers today. Eventually the benefits can, and should trickle down.

.NET

Microsoft .NET may be many things, and part of it (combined with web technology) includes the exploitation of the independence of component location and execution. Since the debut of .NET, this key feature of the architecture has taken a back seat and may soon be completely forgotten. But maybe not!

Conclusions

Several conclusions can be made from the research and testing performed for the development of this paper.

Virtual operating systems are a viable solution today to consolidate numerous underperforming servers. By including additional resource management software on these virtual machines, both at the host OS level and inside the virtual machines, consolidated servers can be provisioned with more loading than the current "conventional wisdom" dictates.

The virtual abstraction layer load balancing algorithms in place today are rudimentary, at best. Static virtual machine limitations are counter-productive and should be avoided at all costs.

Performance Monitoring tools running within the guest OS that rely on counters obtained for the virtual OS are misleading. Examples of this include the Windows Task Manager, the Performance Monitor, as well as PDH and WMI based tools. Only tools that make user based measurements (such as we use to create the profiles presented in this paper) are effective to use within the guest OS.

Ultimately, operating system virtualization creates barriers that shield the host from knowledge of the level of importance of resource use within a guest or between guests. This prevents the host from truly applying the kind of optimizations that will best improve user's perceptions of system performance. Even the VMware ESX today ultimately has a Linux based host OS with limited visibility into the guest OS. Additional advances in the performance of large scale virtual machines should be obtained with advances in this area.

References

1. *CIO Update: Server Consolidation Can Offer a Range of Benefits,* Gartner, March 2004.
2. *An Introduction to Server Consolidation Technologies*, A Leostream White Paper; http://www.leostream.com/pdf/LeostreamServerConsolidationWhitePaper10.pdf
3. Transaction Processing Performance Council; http://www.tpc.org/tpcc/results
4. *Perceived Performance, Tuning a System For What Really Matters*, Tim Mangan, TMurgent Technologies; http://www.tmurgent.com/images/PerceivedPerformace.pdf
5. *Perceived Performance,* triCerat Inc; Privately Distributed.
6. *A Virtual Machine Time-Sharing System,* Meyer and Seawright IBM, 1970; http://www.research.ibm.com/journal/sj/093/ibmsj0903D.pdf
7. *Virtual Machine Monitors: The Original Execution Environment,* IBM Research; http://www.research.ibm.com/vee04/Rosenblum.pdf
8. *X Windows System,* X.Org Foundation; http://www.x.org/
9. *Application Virtualization: The Core of SoftGrid,* Softricity Inc; http://www.softricity.com/products/what.asp
10. *The Xen Virtual Machine Monitor,* University of Cambridge Computer Laboratory; http://www.cl.cam.ac.uk/Research/SRG/netos/xen/ (This link provides access to a number of white papers and information regarding Xen).
11. *Grid.org; http://www.grid.org/home.htm*
12. *Global Grid Forum;* http://www.ggf.org/
13. *IEEE Technical Committee on Scalable Computing,* IEEE; http://www.ieeetcsc.org/
14. *What is Grid Computing?,* IBM Corporation; http://www-1.ibm.com/grid/about_grid/what_is.shtml
15. *The Anatomy of the Grid,* Foster, Kesselman, and Tuecke, The Globus Alliance; http://www.globus.org/research/papers/anatomy.pdf