

How Many Guests Can You Serve? - On the Number of Partitions

Yiping Ding Ethan Bolker*
BMC Software

Virtualization is still today the hot topic it's been for several years. It has entered production environments both for server consolidation and for the rollout of new applications. Several implementations are available to choose from. In this paper we introduce a statistical model that helps you answer one of the first questions you must ask: how many virtual systems can comfortably run in a particular virtual environment, with known physical processor, memory and disk resources?

1. Introduction

As server power becomes cheaper while costs increase for other resources like footprint, electricity and, most important, management, it is natural to look to centralization as a cost effective solution. But server consolidation is more than just moving applications onto fewer more powerful servers. Different applications may require different operating systems. An unreliable application may affect others with which it shares a server. Here virtualization comes to the rescue – with it you can support and isolate heterogeneous environments on a single hardware platform.

To take advantage of virtualization you begin by creating two or more virtual subsystems or guests in your physical server, each of which behaves as if it were a separate physical system. A natural first question is: how many guests can a particular hardware configuration and virtualization platform support, even before you begin to think about the resources required to run applications in those guests?¹ We begin to answer this question. We introduce a simple model for a virtualized system and then analyze

¹ The analogous question for real guests might be “Figure out how many will fit around your dining room table even before you begin to think about how much they will eat.”

* Ethan Bolker is also Professor of Computer Science at the University of Massachusetts, Boston.

potential factors that limit the number of guests. We discuss the tradeoffs between the overhead introduced by the partitioning and the efficiency gained through virtualization.

We consider transaction processing workloads, for which the response time is commonly the best measure of service. We provide a straightforward analysis of the CPU contribution to response time in a virtual system. For other resources (memory, disk utilization) we discuss standard performance metrics at a lower level, with the understanding that there are tools to compute the impact of those metrics on transaction response time. In our discussion, we will often use VMware as an example, although the model and methods discussed apply to other virtualization architectures as well.

2. A Simple Model for a Virtualized System

In this section, we introduce a simple model for virtualization. The model helps us understand the linkage between physical and logical systems and identify the factors that limit the number of guests that can be supported.

Figure 1 is a rough picture of a standard computer's architecture. Figure 2 illustrates a typical virtualized system with three guests (logical partitions) running on a single physical system, which may itself consist of multiple physical servers. The virtualization manager

between the guests and the physical servers maps resource requests from the guests to the physical system. If we compare the architecture of each guest to the architecture of a traditional computer, shown in Figure 1, we can see that they are conceptually identical. Moreover, the architecture below the virtualization manager is identical as well. Thus we can view the virtualization manager as a new “operating system” and the guests as applications running on that operating system. The virtualization manager meets the primary goal of an operating system: to make the computer system *convenient* to use. Employing the computer hardware *efficiently* is a secondary goal, but one that must be addressed when planning for application performance.

Figures 1 and 2 remind us that whether the server is virtualized or not, all computing eventually takes place in hardware somewhere. A virtualized system cannot deliver performance beyond what the physical hardware is capable of providing after subtracting the overhead required to manage the virtualization.

Table 1 contains a list of major components that may be either virtual or physical. The capacity or capability of each physical component may be specified by the hardware manufacturer or measured by some accepted methodology. The way those logical components are partitioned among the guests is set by the Virtualization Manager, either with default values or values configurable by the user.

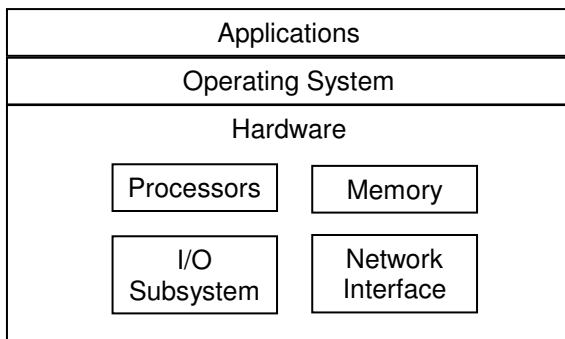


Figure 1. A traditional computer system.

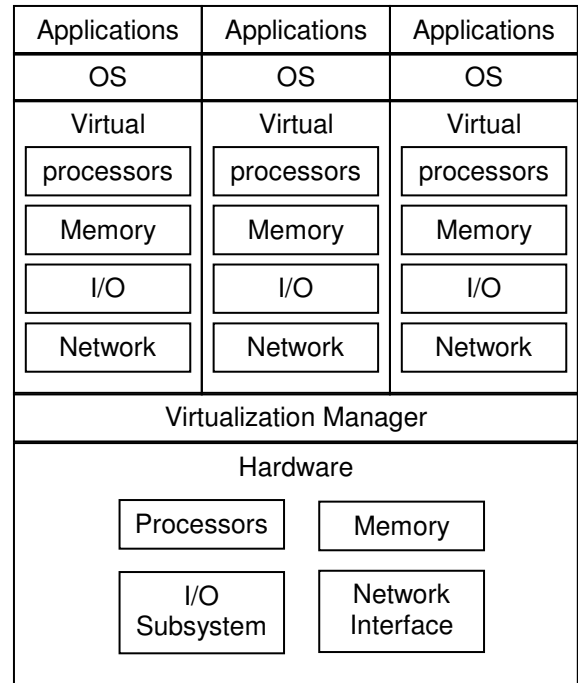


Figure 2. A virtualized system with 3 guests. Each guest runs its own operating system, which may be different from the others. The Virtualization Manager schedules access to the real physical resources to support each guest. The Hardware need not be a single system; it might be a cluster of servers managed by the Virtualization Manager.

The first step in performance management is to measure what’s measurable and interpret the results. In a virtualized system it is crucial to understand how performance metrics collected for a logical component relate to measurements at the physical hardware level or at the Virtualization Manager level. Studies have shown that the measurements taken at the logical component level may differ significantly from corresponding measurements at the Virtualization Manager level [BD]. In this paper, we assume that all measurements for the performance metrics defined in Table 1 are collected at the Virtualization Manager level.

Suppose a virtualized system has n partitions or guests, G_1, G_2, \dots, G_n , running on a physical system whose four hardware components are Processor P , Memory M , I/O subsystem D , and Network N . A Virtualization Manager, V , maps

Hardware Components	Hardware Specifications	Related Performance Metrics
Processors	CPU Performance Rating	* Utilization * Response Time
Memory	Memory Size	Paging Rate
I/O Subsystem	* Disk Space * Access Time * Bandwidth	* % Used * Delay * Utilization
Network	* Bandwidth	* Utilization

Table 1. Hardware components and their related performance metrics.

resource requests by logical partitions to the physical components. Table 2 defines the notation we will use for performance metrics at the logical and physical levels. For consistency, we assume that the Virtualization Manager collects all measurements for both physical and virtual components.

Component performance metrics	For physical system	For logical (Virtual) partition i
Processor Utilization	U	* U_i * u_i
Virtualization Overhead	$U_o(n)$	$f \times u_i + c/n$
Memory Size	M	M_i
Paging Rate	P	P_i
Disk Space	D	D_i
Disk Bandwidth	S	S_i
Network Bandwidth	B	B_i
Network Latency	L	L_i

Table 2. A set of performance metrics for logical partitions and physical system.

We assume that the measurements are all consistent² so that we always have

² Not always true, but that's a subject for another time.

$$U = \sum_{i=1}^n U_i . \quad (2.1)$$

But virtualization isn't free. The real total utilization U includes overhead, which we assume has been charged to the guests. Thus each guest's utilization U_i may be written

$$U_i = u_i + f \times u_i + c/n , \quad (2.2)$$

where u_i is the utilization the manager charges for real application work inside the guest, and $f \times u_i + c/n$ is virtualization overhead. The first term, $f \times u_i$, is proportional to the guest's useful work, the second, c/n , is the guest's share of the cost c of running the virtualization manager, which must be paid even when the guest is idle. Thus when there are n guests we have

$$U = \sum_{i=1}^n U_i = (1+f) \sum_{i=1}^n u_i + c . \quad (2.3)$$

The total overhead is

$$U_o(n) = f \times \sum_{i=1}^n u_i + c . \quad (2.4)$$

Since we plan to model a transaction processing workload, we will specify service level objectives in terms of transaction response time r for a nominal transaction with average service time of one second. We will consider two ways to do that

- A bound R_{avg} for the average transaction response time. For example, we might require that average to be less than 3 seconds.
- A bound T for the fraction of the transactions whose response time may exceed R_{max} . For example, we might require that 90% of all transactions finish in less than 3 seconds. (See Figure 5.)

Our model will show, statistically, the breakeven point beyond which adding more guests leads to performance penalties that make us miss our

service level objective and thus may outweigh the benefits of virtualization.

Traditionally, the CPU response time for a simple transaction processing application on a system dedicated to that application is modeled by an M/M/1 queue (transaction arrivals are a Poisson process and service times are exponentially distributed). If transactions arrive at a rate of λ per second and each requires an average of s seconds of computation the average response time will be $s/(1-u)$, where $u = \lambda s$ is the CPU utilization. In particular, when utilization is about 80%, response time is about 5 times service time. For our nominal one second transactions the arrival rate in transactions per second is numerically the same as the utilization.

Suppose we have installed n identical guests on a system n times as powerful as the servers on which each was running separately, and arranged for each to have access to $1/n^{\text{th}}$ of the processor. Then each will behave just as it did before it was virtualized. We may have made the system easier to administer, but we haven't improved performance.

Fortunately, it's unlikely that all the guests will be equally busy at every moment. If the system is configured so that resources are allocated on the basis of need then the random variations of each guest's needs offer an opportunity.

3. The Benefit of Virtualization

In this section we analyze a model that shows statistically how virtualizing a set of servers may have performance benefits: the sum can be better than the sum of the parts. The extent of the benefit depends on the virtualization overhead.

Let's assume that there are n separate identical physical servers with the same processing power or performance rating and the same average utilization $\bar{U}_i = U$, $i = 1, 2, \dots, n$. We will model the virtual system as an M/M/n queue[J]. That captures our assumption that when one guest is

idle another guest may use its virtual CPU. (The virtualization engine may not see things that way, but our model can in any case.) Then standard queueing theory shows that the average transaction response time will be better in the virtual system than in the individual ones.

Before proceeding to discuss the effect of virtualization overhead, we make explicit an assumption that has been implicit until now: all resources are available to all guests at any time. It is precisely that assumption that allows a momentarily very busy guest to take advantage of free resources because other guests are momentarily idle. If you configure the virtualized system with caps on the fractions of resources available to each guest this can no longer happen. There may still be other benefits to virtualization, but this is not one of them.

Even that benefit diminishes when we take virtualization overhead into account. Figure 3 shows the effect on average response time of virtualizing a four server system. We assume $c = 0$ and show what happens for various values of f (0, 20%, 30%, and 40%). Even at an overhead rate of 20%, the virtualized system does better than the individual system at all loads up to $u = 0.72$. At a 30% overhead rate the virtualized system is better for load up to about $u = 0.62$. In other words, when each standalone server has low average utilization, it is worthwhile to pay some small overhead to consolidate and virtualize those standalone servers.

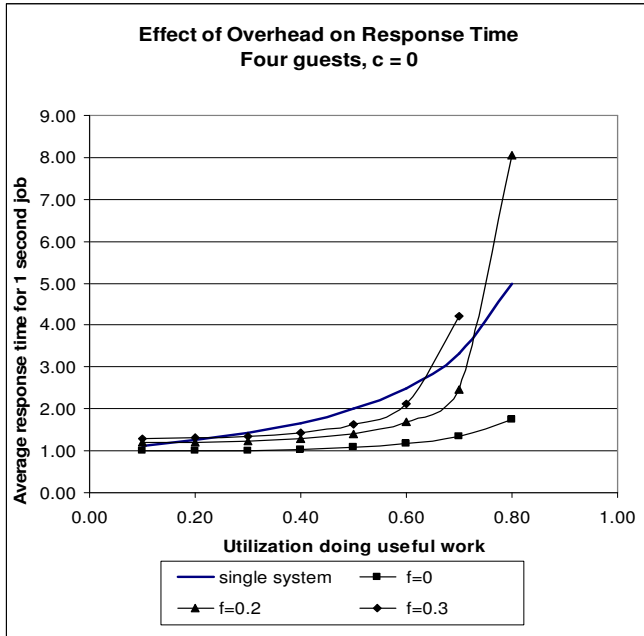


Figure 3. Comparing the performance of a single system with that of virtualized system with 4 guests, the virtualized system performs better at low utilization, worse at high. The crossover point depends on the virtualization overhead.

Figure 4 is the analogue of Figure 3 when the service level objective is a bound on the probability of a long response time. It shows even more strikingly the benefit of virtualization: the advantage of the virtualized system over the single system persists at higher utilizations. When the average job service time is 1 second, the minimum response time is also 1 second. If we set the response time SLO = 2 seconds, at an overhead rate of 20%, the virtualized system does better than the individual system at all loads up to $u = 0.75$. At a 30% overhead rate the virtualized system is better for load up to about $u = 0.65$. Figure 5 shows that when the response time SLO is 3 seconds, the virtualized system does even better: the crossover points occur later. Henceforth we will discuss just this second kind of service level objective.

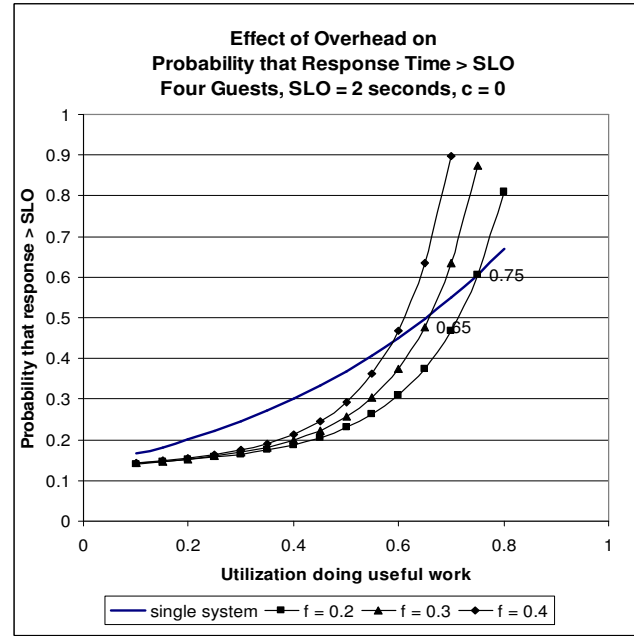


Figure 4. Comparing the performance of a single system with that of a virtualized system with 4 guests. The virtualized system performs better at low utilization, worse at high. The crossover point depends on the virtualization overhead, the response time Service Level Objective (see Figure 5), and the number of partitions (see Figure 6).

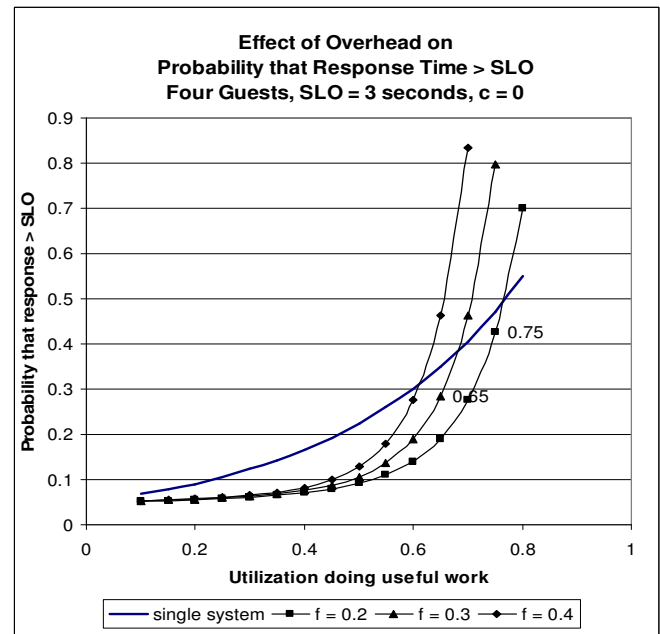


Figure 5. The virtualized system performs even better when the response time SLO is changed from 2 seconds (as in Figure 4) to 3 seconds.

Figure 5 shows that even with 30% virtualization overhead, when the utilization doing useful work

is less than 50%, more than 90% of all transactions finish in less than 3 seconds. For the single system that occurs only for utilizations less than about 25%.

Figure 6 shows that at relatively low virtualization overhead consolidating many separate servers can significantly decrease the probability of long transaction response times.

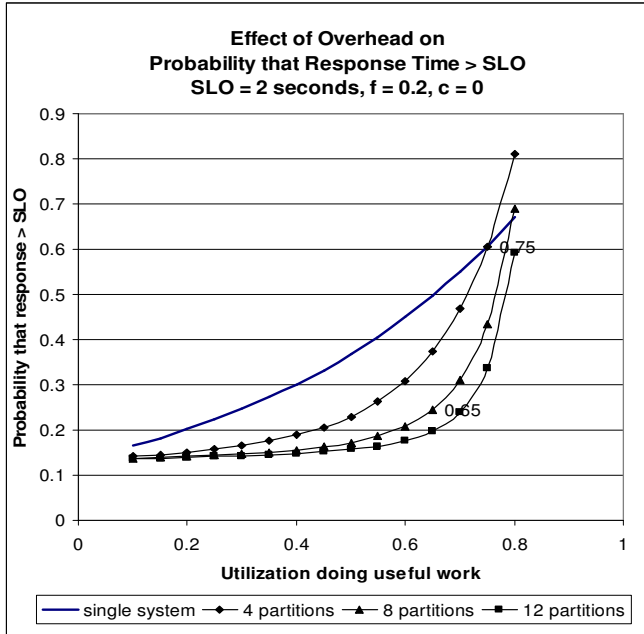


Figure 6. Comparing the performance of a single system with that of virtualized system when virtualization overhead is relatively small. The virtualized system performs significantly better as number of partitions increases.

Figure 7. shows that even at high virtualization overhead ($f=0.5$, $c=0.5$) if the server handling the consolidated load is powerful enough to manage a large number of partitions it can absorb that overhead and still ensure that service level objectives are met.

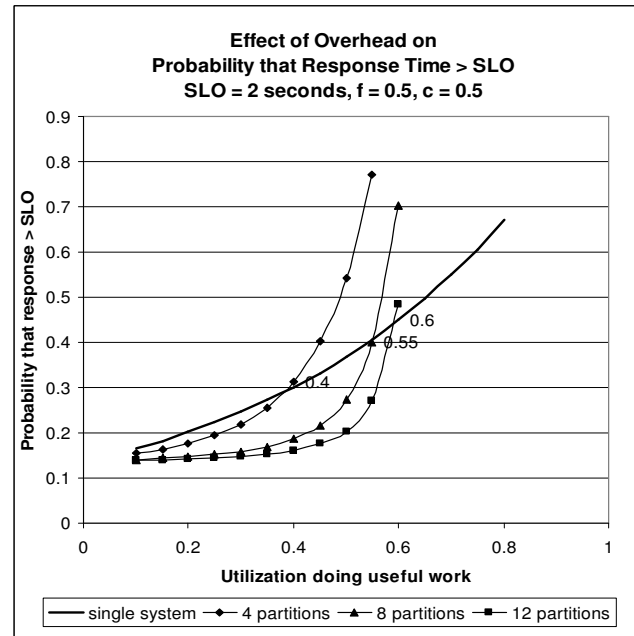


Figure 7. The impact of overhead on the probability that response time > SLO when the fixed overhead ($c = 0.5$) as well as utilization dependent overhead ($f = 0.5$) are relatively large.

4. Other Resource Constraints

In this section we consider other constraints that may limit the number of guests: Memory, Disk, and Network.

4.1. Memory

Since each guest runs its own operating system, each needs enough memory to operate properly. That memory requirement varies from OS to OS. For example, Microsoft Windows XP Professional with SP2 requires at least 128 megabytes (MB) of RAM while SUSE Linux Enterprise Server 9 needs 512 MB to 3 GB of RAM or at least 256 MB of RAM per CPU. Thus if the virtualized system has 10 guests, the operating systems alone will need 1.28 GB of RAM for Windows XP and 2.56 GB of RAM Linux. To that one must add the memory requirements for applications running in each guest. Because the physical memory is shared among the guests as virtual memory allocated by the manager, the total memory required for the entire

virtualized system may be lower than the sum of the memory sizes required for all partitions when those partitions are standalone servers, since the guests might not each need their full complement simultaneously. Under reasonable assumptions we can calculate the amount of physical memory needed in the virtual system. The cost of insufficient memory is increased paging (P), since from an applications point of view all memory is virtual. Users can ask for as much virtual memory as they want. In this section we will discuss the probability that at any time there is insufficient physical memory. Then we will discuss the impact on transaction response time.

Suppose each of the n identical guests needs M_i MB of RAM, where M_1, M_2, \dots, M_n are mutually independent, identically distributed exponential random variables with an average memory requirement of \bar{M} MB. For each guest, the probability of requiring more than \bar{M} MB of memory is

$$P(M_i > \bar{M}) = e^{-1} = 37\%. \quad (4.1)$$

It follows that the probability that the total instantaneous memory requirement for the virtual system, $M = M_1 + M_2 + \dots + M_n$, exceeds $n \times \bar{M}$ MB is:

$$P(M > n\bar{M}) = e^{-n} \sum_{i=0}^{n-1} (n)^i / i!.$$

For example, when $n = 8$,

$$P(M > 8\bar{M}) = e^{-8} \sum_{i=0}^7 (8)^i / i! = 45\%. \quad (4.2)$$

It is interesting to note that, from (4.1) and (4.2), there is a high probability that the n guests will need more than n times the average memory required by a single guest. However, a virtualized system with many guests is less likely to require much more memory beyond its average needs,

$n \times \bar{M}$ since the more guests the less likely it is that many will simultaneously require more than their average.

If we define x as a multiple (>1) of the average memory size, then we will have

$$P(M > xn\bar{M}) < P(M_i > x\bar{M})$$

for x beyond a certain value.

Figure 8 shows that when $x > 1.125$,

$$P(M > x8\bar{M}) < P(M_i > x\bar{M}),$$

for a virtualized system with 8 partitions.

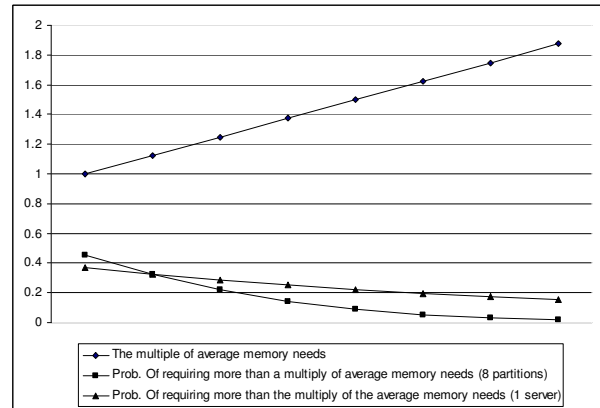


Figure 8. With 8 partitions, the physical memory is utilized more efficiently.

When physical memory runs out, paging increases. When paging increases, so does transaction response time. So we need to understand the connection between overcommitted memory and paging rate. Empirical evidence collected over the years suggests that the relationship between paging rate P_i and memory size M_i for guest i can be approximated as [DA]:

$$P_i = C_i M_i^{-\alpha_i}, \quad (4.3)$$

where C_i is a paging coefficient that converts memory constraint to pages per second and $\alpha_i > 1$ is the paging exponent. Figure 9

shows an example when $C_i = 100$ and $\alpha_i = 1.1$. Although for different applications the values of the paging coefficient and the paging exponent will be different, the paging curve will always resemble Figure 8.

Each guest can define its own paging rate target P_i . With that target and application paging behavior parameters C_i and α_i , guest i needs

$$M_i = \alpha_i \sqrt{\frac{C_i}{P_i}}$$

MB of physical memory. Since the virtualization manager often chooses to partition the physical memory statically in order to achieve performance isolation, the total memory requirement for the virtual system is therefore,

$$\sum_{i=1}^n \alpha_i \sqrt{\frac{C_i}{P_i}}$$

If the size of the memory for the virtual system is given, say M , then the number of partitions, n , that can be supported must satisfy:

$$\sum_{i=1}^n \alpha_i \sqrt{\frac{C_i}{P_i}} + M_0 < M,$$

where M_0 is the extra reserved memory set aside by the virtualization manager so that when a particular guest needs more memory than its share, the reserved memory can be used based on certain memory management policies. In this case, the virtualization manager views each guest an application and memory modeling techniques can be applied at this level as well. The extra memory can also be used as swap space. In the case of VMware, M_0 includes "Shared Common," "Virtualization," "Service Console," and "Free."

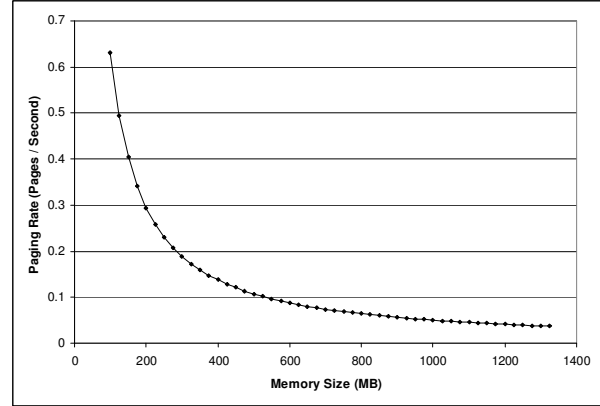


Figure 9. The relationship between memory size and paging rate for fixed demand for virtual memory.

4.2 Disk

In a virtualized system, memory is shared among running guests in order to manage volatile information. Disks store information with longer persistence. Disks in a virtualized system may be virtual for the OS and application code, or attached directly to individual guests via the network for data access. We will discuss just the virtual disk storage here.

Each guest will need enough disk space for its operating system and applications. OS and applications vendors usually provide guidelines or recommendations about the amount of space required. For instance, Microsoft specifies that Windows XP Professional with SP2 requires 1.5 GB of available hard disk space. SUSE Linux Enterprise Server 9 requires 4 GB of hard disk space, while RedHat recommends minimum of 900 MB for a workstation and 1.7 GB for a server.

The total physical disk space D must exceed the total of the disk space needed for each guest:

$$\sum_{i=1}^n D_i < D.$$

Recall that when calculating disk requirements the availability of space is only the first part of the story. We must also consider potential disk utilization. Multiple partitions increase the chance of concurrent I/O requests to the same physical device. We must make sure that I/O throughput is well balanced among all physical disks and each disk bandwidth S can handle the load from multiple guests (a fraction of throughput S_i from multiple guest i). For a given I/O subsystem SLO, how many partitions that the subsystem can support depends on characteristics of applications or workloads (whether or not they are I/O intensive) and where data is stored. A paper like this can't provide generic solutions. You have to run your applications and collect and analyze real data.

4.3. Network

Most applications these days require communication among computers. Often I/O requests that look local to the application actually go through a SAN, competing with other network traffic as well as with other systems trying to access the same physical disk.

Figure 10 shows the Virtual Ethernet Switch that VMware ESX server software uses to virtualize the “network experience.”

Although you can configure a guest with network access speed from 10 to 1000 mbps, the actual speed will depend on the actual speed of the physical adapter. (This is analogous to the situation with virtual memory, which can be requested at will but provided only by underlying physical memory.) Tools exist with which you can model a Virtual Ethernet Switch.

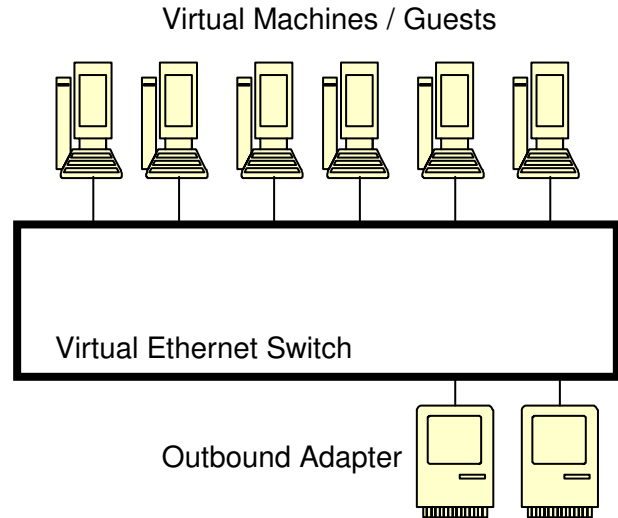


Figure 10 VMware ESX Virtual Ethernet Switch.

The number of guests a virtual system can support depends on how many *total* packets the guests will send or receive in a given time interval and on the network and application service level objectives. In this case, the number of partitions is not a determining factor, although statistically a large number of partitions are more likely to generate more network traffic. Here too the randomness we postulated in our discussion of CPU needs may provide some relief: it's unlikely that all the guests will need lots of network access simultaneously.

Assume that there are n partitions in a virtualized system and partition i generates K_i packets per second. Suppose there are m network adapters with b mbps bandwidth each and the Incoming (outgoing) packet rate through adapter j is I_j (O_j). Clearly, the sum of all in and out packet rates cannot exceed the Virtual Ethernet Switch bandwidth and the total bandwidth of the network adapters.

For the Virtual Ethernet Switch, the following inequality must hold:

$$\sum_{i=1}^n K_i + \sum_{j=1}^m I_j < B\rho/k,$$

where B is the bandwidth of the Virtual Ethernet switch, k is the average packet size, and ρ is the network utilization service level objective. Note that we only count the packets generated by the guests; not the packets received by the guests. That is because other guests must either send the packets that were received or they are inbound packets through the network adapters. Those inbound packets are counted by the second term of the summation in the inequality above.

For the network adaptor, the following inequality must hold:

$$(I_j + O_j)k < b.$$

From the network adapter's point of view, the number of partitions is not a direct factor. If all the guests just talk to each other, then the Virtual Ethernet Switch could be saturated while the network adapters are idle. On the other hand, if network adaptors are saturated, then the Virtual Ethernet Switch will not be idle. The switch is likely to be very busy if the network adaptors and the switch have compatible bandwidths.

5. Summary

In this paper, we have looked at scalability issues related to the number of partitions that a given system can support. We examined the factors that affect that number from several perspectives: processors, memory, disk, and network. It is clear that many traditional performance modeling techniques and tools can help answer scalability and capacity planning questions, as long as we clearly understand what is virtual, what is real, and how to map one to the other. We also show that virtualization may have inherent statistical advantages. With the same load to capacity ratio, virtualized systems are usually more efficient and do better in terms of performance, as long as the overhead of virtualization does not overtake the benefit.

We demonstrated this for processors and memory. Virtualization is a useful concept and a proven technology. Now it is up to hardware and software vendors to provide implementations that are easy to use and incur low overhead.

References

[BD] Ethan Bolker, Yiping Ding, "Virtual performance won't do: Capacity planning for virtual systems," CMG Proceedings 2005. pp 39-49.

[DA] Yiping Ding, Subhash C. Agrawal, "Tuning Memory Pool Sizes in AS/400," CMG Proceedings 1994. pp1302-1313.

[IV] High-Volume Web Site and VMware team, "VMware ESX Server 2: Performance and Scalability Evaluation," January 27, 2004

http://www-128.ibm.com/developerworks/websphere/library/techarticles/hipods/esx_evaluation.html

[OS] Operating System, http://en.wikipedia.org/wiki/Operating_system

[J] Raj Jain, "The Art of Computer Systems Performance Analysis – Techniques for Experimental Design, Measurement, Simulation, and Modeling." John Wiley & Sons, Inc. 1991.

[SU] SUSE Linux Enterprise Server 9 System requirements:

<http://www.novell.com/products/linuxenterpriseserver/sysreqs.html>

[MS] Windows XP Professional System Requirements,

<http://www.microsoft.com/windowsxp/pro/evaluation/sysreqs.mspx>

Appendix A

The response time cumulative distribution function for an M/M/m queue:

$$P(R \leq r) = \begin{cases} 1 - e^{-r\mu} - \frac{\rho_m}{1 - m + m\rho} [e^{-m\mu(1-\rho)r} - e^{-r\mu}] & \rho \neq \frac{m-1}{m} \\ 1 - e^{-r\mu} - r\rho_m\mu e^{-r\mu}, & \rho = \frac{m-1}{m} \end{cases} \quad r > 0$$

where $\rho = \lambda / (m\mu)$, $\rho_m = P(\geq m \text{ jobs}) = \frac{(m\rho)^m}{m!(1-\rho)} p_0$, and the probability of zero jobs in the system p_0 is

$$p_0 = \left[1 + \frac{(m\rho)^m}{m!(1-\rho)} + \sum_{i=1}^{m-1} \frac{(m\rho)^i}{i!} \right]^{-1}.$$

This formula comes from [J]; we have taken the liberty of correcting a typographical error by introducing the square braces.