A Simalytic Approach to Modeling Virtualized Environments

Dr. Tim R. Norton
Simalytic Solutions, LLC

CMG 2008 Paper 8199

System Virtualization allows multiple O/S images to execute on a single physical host computer. Measuring the host resource usage is straightforward, and the necessary tools are included with most virtualization environments. Complexities introduced by the different virtualization techniques create problems with measurements within the guests that impact the ability to precisely plan capacity for applications. This paper proposes a more holistic approach, using the Simalytic Modeling technique, to planning capacity needs by understanding the effect of resource usage on application performance.

1. <u>Introduction</u>

Two key aspects of capacity planning are the demand for available resources and the effective completion of work that fulfills a business need. Although some batch-orientated applications still exist, the trend is to design applications that address the business need using interactive transactions. For these applications, the appropriate measurement of demand is the arrival rate of the application workload, usually in transactions per second. Measurement of resource availability requires understanding how the transactions use each resource. The time each transaction uses a resource is called the service time. The more often transactions use the resource, (the higher the arrival rate) and/or the longer it takes with each use, the more of the available resource will be used. This relationship, called the Utilization Law, is simply the arrival rate times the service time. The result is the utilization of the resource expressed as percentage with 0% being not used at all and 100% being totally used. As resource utilization increases, each transaction must wait longer for its turn because other transactions are ahead of it in the "queue" for the resource. The response time of a transaction is the sum of its service time plus its waiting time (the sum of the service times of all the transactions ahead of it) and is expressed as an average over some measurement interval. Capacity planning uses models to project the usage of all resources into the future to see the impact on transaction response times. The accuracy of these models depends on the accuracy of the transaction measurements.

Virtualization is about increasing the parallelization within the host system to increase the amount of productive business-related work that is done. This is accomplished by increasing the usage of resources, but not to the point that the business work slows down. However, virtualization is pervasive, and it is a very broad topic. In this discussion, the term "virtualization" is limited to system virtualization, which is a control program on a host computer running one or more guest operating system images as if they were on independent systems. The control program in a virtualized environment, called either a VMM (Virtual Machine Monitor) or a hypervisor[1], can accurately measure the resource usage by each guest or VM (Virtual machine). For measurement of individual transactions, however, we must continue to rely on the guest operating system, which unfortunately, often does not understand that it is running in a virtualized environment. From a Capacity Planning point-of-view, this presents problems with virtualized environments that are related to the quality of resource measurements, the granularity of resource measurements and guest interactions with the virtualization environment.

This paper proposes an approach to modeling transactional applications running in a virtualized environment that focuses on the business impact rather than attempting to resolve the guest measurement problem.

2. <u>The Problems</u>

These problems were introduced by the author as part of a larger discussion about how many of the aspects of virtualization impact the capacity planning process. (Norton, 2007) The subsections presented here restate some of those problems, with minor changes, that are focused on the current topic.

2.1 Accuracy of Measurements

Any capacity analysis relies on measurements of resources, both usage and potential. Virtualization at any level tends to generalize these measurements because the point of the virtualization is to abstract the underlying resources. Problems arise when the entity collecting the measurements, be it the operating system, an application or a performance measurement utility, doesn't understand that the measurements are of the generalized resource instead of the underlying actual resource. A measurement technique must make assumptions about what is being measured in order to create a practical implementation. However, these assumptions can cause significant problems when the resources are virtual-

[1] Hypervisor usually implies a hardware implementation and VMM a purely software implementation but, for this paper, VMM will be used for all virtualization control programs regardless of implementation.

ized. For example, many operating systems measure the time a process uses the processor by recording the time from the system clock when the process is dispatched and again when the state of the process is saved so another can be dispatched. The difference between the two times is how long the process ran for that dispatch event and the accumulation of those differences over the life of the process is the total time it used the processor. This is a perfectly reasonable approach because the operating system has total control over which processes run on which processors. A process cannot start or stop running without operation system involvement. When the operating system is running as a guest in a virtualized environment, then this measurement depends on how the system clocks are virtualized. If the guest operating system uses the actual system clock, then any time that a different guest operating system was running will be accounted to the running processes. The guest operating system is unaware of the fact that it lost the use of the physical processors for a while and greatly overstates the amount of time some processes used the processor. If the guest operating system uses a virtualized system clock, then how it is virtualized becomes a significant issue. Many operating systems update the system clock using a timer interrupt but virtualization can cause the interrupts to be delayed. When this happens the virtualized system clock can advance in non-uniform increments causing some processes to appear to use more processor time while others appear to use less.

This accuracy problem applies to the potential, or capacity, of a resource as well as the use of it. The most common assumption is that a resource can be completely used (i.e., 100% utilization) under ideal conditions. However, the nature of the virtualization of the resource can make that not only impossible but also make it impossible to tell what the maximum utilization really is. For example, the way the virtualization control program dispatches guests can affect transaction response times, without any change to the service time, by adding more queue time (when other VMs have control of the resources). Measurements in the guest can see this as elongated processor wait time and as exacerbated processor contention from other workloads.

Because of the accuracy of measurements problem, the capacity planner has a choice between two unpleasant options: using erroneous measurements or doing without measurements. Neither of these options is particularly useful, and it is not readily apparent which one is the better choice. What complicates understanding of this problem is that the magnitude of the inaccuracies varies significantly across platforms. The IBM mainframe environment is much more mature and has resolved many of these problems, but the Windows and Unix environments are much more problematic because of the number of hardware and software vendors involved. The long-term solution lies in operating systems and other measurement utilities using standardized virtualization-specific features in new processors, systems designers implementing those processors and the buying public's willingness to pay for them.

The impact of this problem on capacity planning is that application and resource measurements are less reliable so any projections must include compensation for the increased variability, which usually means including additional capacity.

2.2 Virtualization Implementation

The key concept with system level virtualization is that the underlying resources are shared in a way that increases parallelism. In other words, two or more systems appear to be using the same physical hardware at the same time. This very complex topic cannot be covered adequately here. Michael Salsburg, et al. provide some insight into those complexities:

> For example, what is the basic overhead of running a hypervisor on which the OS images dwell? How does this overhead change as a function of the number of virtual machines and physical CPUs? Can we accurately predict the effects of queueing both at the physical and virtual CPU levels? What is the impact of I/O activity? How about the impact of allocating specific quanta of CPU cycles to each machine? How is performance affected by the selection of a specific virtual technology?
> (Salsburg 2006)

The trick to successful virtualization, regardless of the techniques used, is to maximize the use of resources without negatively impacting application performance. Virtualization raises questions about the overhead of the hypervisor (virtualization control software), clock synchronization and granularity, processor dispatch granularity (does the hypervisor dispatch processors individually or does a guest operating system wait until there are as many physical processors available as defined logic processor units for that guest), and the impact of interrupt delays on the guest operating systems. All of these, and other, topics act as hidden consumers of resources because they are usually not seen and measured by the guest operating systems, and the hypervisor does not provide detailed enough measurements to understand their impact at the workload and process level.

The impact of this problem on capacity planning is that the environment becomes much more complex and much more dynamic. Planning at the resource level becomes impossible for two reasons. First, the resource usage measurements are imprecise and unreliable at best and may even be incorrect in some cases. Second, the drivers for resource consumption are no longer just the application business drivers but everything running on all of the other guest operating systems sharing the same physical resources. The lowest priority work in a guest with a large share of the physical resources can easily run before, or even instead of, the highest priority work in another guest.

2.3 Workload Characterization

Workload characterization used to be a relatively straightforward matter of assigning processes or users or transactions or whatever to workload groups. But now, as more and more resources are shared in ever increasingly complex ways, those assignments are not so simple. Virtualization at many different levels makes it almost impossible to assign the use of a resource to a single application workload. The standard apportionment techniques (Norton 2004) for approximating how much usage of a resource should be attributed to an application are no longer adequate because they rely of either precise measurements or a consistent ratio of usage over time. Precise measurements are lost for all of the reasons already discussed, and the very nature of virtualization is to allocate resources as needed, certainly not in the same ratio from one time interval to the next.

2.4 Building Models

While capacity planning involves much more than building models, the ability to accurately represent a system or application with an abstract model is still a key tool in the planning process. The commercial and Open Source modeling tools available today are all quite capable of modeling complex virtualized environments. The questions are: "What is the expected granularity and precision of the results?" and "What is the required effort to get those results?"

2.4.1 Service Time Calculations

Models use abstraction to represent the time something takes at each stage of a process. Each stage is a server or service center, and the time is the corresponding service time. Because the service center is an abstraction of a more complex process, the service time is also an abstraction. Different types of models use a variety of techniques to achieve a sufficient level of abstraction to make the model practical to solve and yet have a sufficient level of detail to give the results meaning. There is almost always more complexity at the next level down. It is theoretically possible to build a model of an entire environment, from the behavior of the application to the way the network passes data to the operating system services to the management of cache to the pipeline of the microprocessor to the speculative execution of the underlying micro-op instructions. However, such a model would most likely take forever to build and somewhat longer to run. The success of a model lays in the ability to cost effectively approximate the behavior at each service center while producing results in enough detail to allow for meaningful predictions.

How does virtualization affect this abstraction of service time? It may not be measurable within the needed precision for the desired results. Some of the concerns are:

- Accuracy of measurements: If the guest operating system doesn't know it's running in a virtual environment, then any rate metric (utilization, I/Os per second, interrupts per second, etc.) may be incorrect.
- Virtualization overhead: How much of the host's resources are lost to enable virtualization? Some virtualization techniques must emulate or translate guest instructions (especially privileged instructions), which means executing hundreds or even thousands of actual instructions. The VMM must manage and dispatch the guests, manage physical memory and map real interrupts to virtualized interrupts.
- Virtualization delays: The very nature of a shared environment means that virtualization enables one guest to use host resources when another is waiting, but it also means that the resources will likely not be available the instant a guest is ready to use them. The most common of these delays are VM dispatch delay, (the time from a guest being ready to use the processor and when the VMM assigns it to a physical processor) and interrupt delays (the time between the real physical interrupt and when the virtual interrupt is presented to the guest). Increasing the accuracy of measuring these delays will usually increase the virtualization overhead.
- Interference from other VMs: Most VMMs allow some control over the prioritization of guests (which one is dispatched first), but they do not connect that prioritization to the operating system process prioritization (with the exception of IBM mainframe virtualization which is starting to allow WorkLoad Manager to adjust VM weights). The impact is that very low priority work in one VM can interferer with high priority work in another VM.

The truly confounding dilemma is the inverse relationship between many of the problems. Some conditions cause overstatement of service time to increase as service center utilization increases and others cause it to increase as service center utilization decreases. Increasing accounting reduces guest responsiveness. Reducing delays increases overhead. Reducing interference increases management overhead.

2.5 Uniformity

Many of the assumptions made when building a model are about how work is distributed to the service centers. For example, the processors in an SMP system (tightly-coupled processors) can be modeled as a single server where the service time is adjusted for the number of processors and the interprocessor communication (Menascé 1994, p. 263-4). Underlying this technique is the assumption that the application can actually use all of the processors. Virtualization can change the validity of this assumption by masking the actual use of the real resources. Those changes can vary dynamically as systems load varies, both in a VM and for the host as a whole. The ability to spread a given workload across all available similar resources may not always be a good assumption. Losing the ability to use such simplifying assumptions will increase the time to both build and to run a model.

3. A Proposed Solution

There are no easy solutions for these problems but dealing with less than perfect modeling measurement data is not new to the capacity planner. In the past, it was the norm for operating systems to account for some processor time incorrectly, and this is again the case with virtualization. What we need is a way to use known good measurements in an application model. This paper proposes doing that with the Simalytic Modeling Technique by using what can be accurately measured (transaction arrivals, transaction response time and utilizations from the VMM) to create a profile of the application's performance on each virtual system. Simalytic Modeling addresses the problems with performance modeling of virtual environments by reducing the reliance on guest operating system measurements. It also addresses the problems with predictive modeling of virtual environments by effectively using other techniques, such as analytic modeling tools and load test tools, which are difficult to use in these environments.

Simalytic Modeling is a hybrid modeling technique that uses load dependent servers to simplify the representation of complex resources in a model. (Norton 2001) It is not a product but rather a technique to combine modeling tools. Almost any of the available tools can be used in this unique way to address the problems modeling multi-tier applications. As originally developed, Simalytic Modeling uses an analytic modeling technique to profile the response time of a transaction workload at each node in a multi-tier environment. A simulation model is then created to describe the transaction flow across the tiers, using the node level response time profile to create a load dependent service center for each node. The complexities of the service centers are abstracted by using a service time that is somehow dependent on the arrival rate of each transaction workload at that service center. The details of how a load dependent service center is constructed have been discussed in other works (Norton 1997a, Norton 1997b) and will not be presented here.

This proposed extension for virtualization modifies the load dependent service center creation in a Simalytic Model by including information from the VMM about both the VM utilization and the host utilization. The concept is to vary the service time of each service center in a virtual environment by the VM utilization to represent the interference from work in the same guest, by the host utilization to represent the interference of work in other VMs and by the arrival rate of the application workload to represent its service demand. The objective is to build a comprehensive model of all transaction workloads in the virtual environment to explore the affects on response times as arrivals increase.

3.1 Performance Modeling

One measure of operating system accounting accuracy is the capture ratio, which is the sum of the time for all processes over an interval divided by the system busy time over the same interval (usually calculated as the interval time minus the idle time). In a virtual environment it is relatively easy to check the system busy time using a similar technique (guest operating system busy time divided by the VMM busy time for that guest). This virtual guest capture ratio can be helpful in determining the overall effectiveness of the guest operating system utilization measurement. but it doesn't address the measurements of individual processes inside the guest operating system. The fundamental assumption behind the capture ratio metric is that the operating system is doing some work that just can't be accounted to a process, such as the early processing in an interrupt (before the operating system knows which process it's for). In a virtualization environment there is no easy way to know when additional "non-work" time (when the host is running a different VM) is being added to the processes, including the idle process.

Two different modeling scenarios need to be discussed.

1. Single Application of Interest – Model only the transaction workload of interest and treat all other workloads as static interference. This is similar to modeling the same workload in a stand-alone environment except that some of the interference (the other VMs) is not subject to the process prioritization of the guest operating system. This type of model can be effective when the other workloads are relatively static, but it rapidly becomes problematic when the other workloads are dynamic.
2. Total Environment – Model all transaction workloads that share the same virtual environment. This produces a much more complex model with many more model executions. Creating and running models this complex is seldom a practical solution because of the time and effort required. A useful compromise may be to model the largest two or three applications and treat the rest as static interference. This can still lead to an explosion in the number of model sceneries to find a balance when the host cannot support a reasonable arrival rate for all workloads simultaneously. Identifying the effect of the maximum arrival rate of all of the workloads is not useful because the point of virtualization is to interleave the utilization peaks and valleys of different workloads.

Two different VM scenarios need to be discussed.

1. Single Workload Guest – the VM is only processing the application's transactions. The service time will include some non-transaction "overhead" but

the effect of this decreases as transaction arrivals increase.

2. Multiple Workload Guest – the VM processes other significant workloads in addition to the transactions of the application of interest. Here the service time of each workload must be adjusted in some way that minimizes the influence from the other workloads.

This proposal is best illustrated by an example of a hypothetical three-tiered application as shown in Figure 1. Assume that the application has a web component, an application server component and a database component, all of which are implemented in a virtual environment along with several other workloads (some transactional and some not).

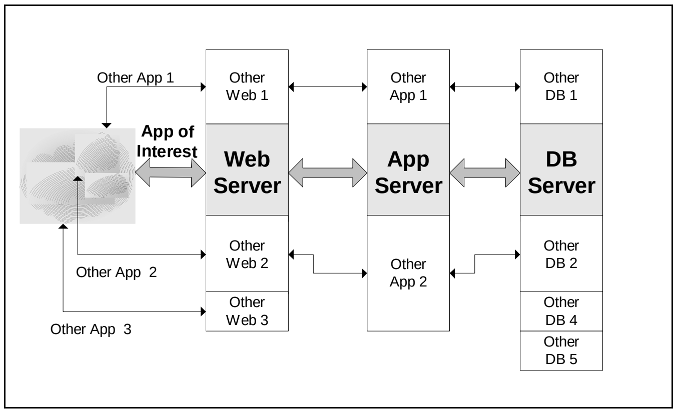

Figure 1 Hypothetical Application Environment

The easiest scenario to model is the Single Application of Interest in the Single Workload Guest. The only virtualization issue is to account for the interference from the other VMs and the VMM overhead. This could be done by adjusting the service times for the Application of Interest (AoI) service centers (Web Server, App Server and DB Server in Figure 1) so that the measured response times match the model response times as the utilization of the other VMs increases. Because the effect of the other VMs is to reduce the capacity of the AoI VMs, they can be treated as overhead. How (and if) the VMM prioritizes VM dispatch determines how much of the utilization of these other VMs takes capacity away from the AoI VMs.

The next easiest scenario to model is the Single Application of Interest in the Multiple Workload Guest. The same virtualization issues apply as above, but we also have to account for the interference from the other workloads in the AoI VM. There are well known techniques for modeling this combination in non-virtualization environments, but they rely on good measurements from the operating system, which may not be the case in a virtualization environment. Simalytic Modeling addresses the lack of good measurements because the load dependent service center profiles are built from measured transaction response times at each node instead of from the guest operating system measurements. For example, if increases in the arrival rate for Other App 1 in Figure 1 do not noticeably increase the host utilization or significant change the AoI response time, then it may be possible to discount the effect of that workload on the AoI at higher than measured arrival rates. On the other hand, if the increases in the arrivals of Other App 1 cause a large increase in host utilization and significant changes in the AoI response time, then the creation of the load dependent service center profile for the AoI must include the Other App 1 arrival rate as a factor. How this is done will vary with each model implementation, but it is relatively easy to validate from on-going measurements.

The hardest scenario to model is the Total Environment with Multiple Workload Guests. There are many virtualization issues to account for, and the dynamic nature of all of the interactions is difficult to measure. In addition, because virtualization is seen as a “cost savings” technique, there is often political pressure to increase the number of VMs on each host, which increases the complexity of the environment to be modeled. A model that accurately represents this environment would have to include details for all of the other workloads in the load dependent service center of every workload. Even with just the small number of VMs shown in Figure 1, this is not practical, let alone in a virtualization environment where each host has dozens of VMs.

Most of the work in building this type of model is not in actually building the model, but in determining just what needs to be in the model. The first step is to analyze each workload to understand its work distribution over time (peak hour of the day, peak day of the week, annual seasonality, etc.) and determine the most likely arrival rates. Using the maximum arrival rate of each workload is not useful unless those maximums occur at the same time, which would mean that the applications were not the best choices to put on the same virtualization host in the first place. The second step is to understand how each workload affects each of the others by correlating the utilization measurements of each VM from the VMM with the response times for all of the applications of interest.

There are, of course, many more scenarios than these three, but these show most of the main issues. The objective is to reduce the model requirements to the one or two applications of interest that drive the overall capacity of the host. The way in which the other applications use the resources at non-peak times can be discounted (unless they are likely to grow and become the dominate application on the host or unless there are performance bottlenecks that affect response times).

3.1.1 Model Validation

Once a Simalytic Model has been created for a virtualization environment, it can be validated against actual production measurements. Again, the known good measurements are used (application response time and VM utilizations from the VM) as comparisons to model results. A variety of intervals can be selected from the actual production measurements and those arrival rates used in the

Simalytic Model. If the model results do not match the actual response times within the desired margin of error, then the load dependent service center profiles would need to be adjusted until they do. Once all of the test cases have been validated, then the model can be used to explore other scenarios.

3.2 Predicting the Future

Probably the most significant problem with modeling applications in virtualization environments is the unreliability of service time measurements from the guest operating system. If the service time isn't correct, then most of the modeling techniques do not work correctly. Even when the guest operating system has an understanding of virtualization and measures the service time correctly, the effects of outside forces (other applications in other VMs) must be taken into account. Simalytic Modeling provides a technique to account for these forces by adjusting the load dependent service center profiles based on observed measurements, such as load test results, or theoretical results, such as those from other modeling tools.

3.2.1 Using load testing

Simalytic Modeling reduces the number of test scenarios needed. Some cases can be eliminated because the workloads do not affect the overall capacity requirements (they are too small during the host peaks) or they do not affect the AoI response time (possibly because of a lower VM dispatch priority). The remaining load test cases can be targeted to creating the load dependent service center profiles rather than trying to test all of the different combinations of arrival rates for all of the different applications. The greatest value to load testing is to see actual measurements for higher than expected arrival rates. If the load tests show response times below the "knee of the curve" (that point in the arrival rate response time curve when queue time suddenly increases, and the response time becomes significantly unacceptable), then creating the load dependent service center profiles is relatively straightforward.

3.2.2 Using Modeling Tools

Other modeling tools can be used to collect and analyze the AoI in a VM if the guest operating system measurements can be relied upon. The easiest way to do that is to run the AoI VM as the only active VM on the host. This can be done even in production environments when the applications are implemented with multiple host servers in each tier (Figure 1 shows a single host server for each tier) because the other VMs can be quiesed on one of the hosts. The load balancers would route that traffic to the VMs on the other hosts in that tier so that the AoI VM was getting all of the resources but still be in the virtualization environment. With most virtualization implementations, this minimizes the impact of virtualization errors in the guest operating system measurements to the point that they are usable by most modeling tools. Once these modeling tools have good measurements, they can generate good predictive models of the AoI and therefore good load dependent service center profiles for a Simalytic Model.

4. <u>Conclusion</u>

Virtualization is the current hot solution but it creates a number of other problems. Given the complexities, uncertainties and variabilities discussed here, it is obvious that planning resources is becoming harder and less precise. Planners must go back to the basics of what capacity planning should be about anyway, understanding the needs of the business, and think in terms of identifying and reducing bottlenecks that prevent applications from achieving the business objectives. Efficient use of resources isn't the issue because it's almost impossible to understand. Planning should take a holistic view that balances costs, technical, financial and human, against the overall benefits with the understanding that there will always be localized inefficiencies that inhibit achieving global optimization. Simalytic Modeling provides a technique that allows using available tools in combination, while reducing the complexity of the overall environment to be modeled in order to explore both performance and capacity constraints on specific applications in virtualization environments.

5. <u>References</u>

Menascé, D., V. Almeida, and L. Dowdy. 1994. *Capacity Planning and Performance Modeling: from mainframes to client-server systems*. Englewood Cliffs, New Jersey: Prentice Hall.

Norton, Tim R., 1997a. *Simalytic Hybrid Modeling: Planning the Capacity of Client/Server Applications*, in 15th IMACS World Congress 1997 on Scientific Computation Proceedings, Modelling and Applied Mathematics conference, August 24-29, 1997 Berlin, Germany

Norton, Tim R., 1997b. *Simalytic Modeling: A Hybrid Technique for Client/Server Capacity Planning*, 1997 Summer Computer Simulation Conference (SCSC '97) Proceedings, July 13-17, 1997 Arlington, VA

Norton, Tim R., 2001. The Simalytic Modeling Technique chapter in *Performance Engineering, State of the Art and Current Trends*. Springer-Verlag Berlin/Heidelberg 3-540-42145-9.

Norton, Tim R., 2007. *The Myth of Precision Planning:Understanding Capacity in an Age of Virtual Parallelism*. In Computer Measurement Group Proceedings: San Diego, CA: CMG, Inc.

Salsburg, Michael, Peter Karnazes, and Bill Maimone, 2006. *It May be Virtual … but the Overhead is Not*. In Computer Measurement Group Proceedings: Reno, NV: CMG, Inc