



**The Association of System
Performance Professionals**

The **Computer Measurement Group**, commonly called **CMG**, is a not for profit, worldwide organization of data processing professionals committed to the measurement and management of computer systems. CMG members are primarily concerned with performance evaluation of existing systems to maximize performance (eg. response time, throughput, etc.) and with capacity management where planned enhancements to existing systems or the design of new systems are evaluated to find the necessary resources required to provide adequate performance at a reasonable cost.

This paper was originally published in the Proceedings of the Computer Measurement Group's 2009 International Conference.

For more information on CMG please visit <http://www.cmq.org>

Copyright 2009 by The Computer Measurement Group, Inc. All Rights Reserved

Published by The Computer Measurement Group, Inc., a non-profit Illinois membership corporation. Permission to reprint in whole or in any part may be granted for educational and scientific purposes upon written application to the Editor, CMG Headquarters, 151 Fries Mill Road, Suite 104, Turnersville, NJ 08012. Permission is hereby granted to CMG members to reproduce this publication in whole or in part solely for internal distribution with the member's organization provided the copyright notice above is set forth in full text on the title page of each item reproduced. The ideas and concepts set forth in this publication are solely those of the respective authors, and not of CMG, and CMG does not endorse, guarantee or otherwise certify any such ideas or concepts in any application or usage. Printed in the United States of America.

Modeling Virtualized Environments in Simalytic Models by Computing Missing Service Demand Parameters

Dr. Tim R. Norton
Simalytic Solutions, LLC
CMG 2009 Paper 9103 Session 601

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. However, complexities introduced by the different virtualization techniques create problems with measurements within the guests, resulting in missing model parameters. This paper shows how to use the Menascé technique that computes missing parameters with the Simalytic Modeling technique to predict application performance more effectively in virtualization environments.

1. Introduction

Two key aspects of predicting resource usage 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. The time each transaction visits a resource is the service time and the total time each transaction uses a resource for all visits is its service demand. The more transactions that use the resource, (the higher the arrival rate) and/or the longer they take with each use, the more of the available resource will be used, as expressed by the Utilization Law (simply the arrival rate times the service demand) (Menascé 1994, p. 134). 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 demand for all resources plus its waiting time (the sum of all of the service times of all the transactions ahead of it) and is expressed as an average over some measurement interval. Models are used to project the usage of the 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 of slowing down the business work. Virtualization is pervasive across all level of the computing architecture but this discussion is focused on 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, can accurately measure the resource usage by each guest or VM (Virtual Machine). For measurement of individual transactions, however, we must rely on the guest operating system, which unfortunately, often is not aware that it is running in a virtualized environment. 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. Since these measurements are not to be trusted, they are in effect, missing.

This paper is an extension of prior work (Norton, 2008) that 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, the missing service demand parameters are computed from measurements known to be good in virtualized environments. The next section is an overview of the problems encountered with measurements from operating systems running in virtual machines. The following section reviews the Simalytic Modeling technique and explains how to calculate the missing model parameters using the techniques developed by Daniel A. Menascé at George Mason University (Menascé, 2008).

2. The Problems

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, focused on the current topic.

2.1 Accuracy of Measurements

Virtualization at any level tends to generalize measurements because the point of the virtualization is to ab-

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.

stract 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 virtualized. For example, the guest operating system may add the time that the other VMs were dispatched to a running process because it doesn't know what else to do with the missing time.

2.1.1 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. Michael Salsburg, et al., provided some insight into those complexities (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, the impact of interrupt delays on the guest operating systems and 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). All of these are not usually 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.

2.1.2 Workload Characterization

Workload characterization used to be a relatively straightforward matter of assigning processes, users, 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 on either precise measurements or a consistent ratio of usage over time. Precise measurements are lost for all of the reasons already discussed. The very nature of virtualization is to allocate resources as needed, certainly not in the same ratio from one time interval to the next. The ability to calculate the missing model parameters as described by Menascé (Menascé, 2008) greatly improves the accuracy of workload characterization.

2.1.3 Service Demand 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 demand. Because the service center is an abstraction of a more complex process, the service demand 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 demand? It may not be measurable within the needed precision for the desired results. 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. 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 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, such as when a guest is ready to use a logical processor but the VMM has not assigned it to a physical processor. 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. The impact of this is that very low priority work in one VM can interfere with high priority work in another VM.

2.1.4 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 demand is adjusted for the number of processors and the interprocessor communication (Menascé 1994, pp. 263-264). Virtualization can change the validity of this assumption by masking the actual use of the real resources.

3. A Simalytic Modeling Solution

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 demand 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, 1997b). The general idea is to create a function, referred to as a Simalytic Function, which returns the appropriate response time for each arrival to be used as the service demand for the service center in the simulation model. There are two major difficulties when creating a Simalytic Function. One is how to properly map the discrete arrivals to an average arrival rate. The other is how to account for increased internal queuing in the load dependent service center as the arrival rate increases. Modeling a virtual environment does not change how the average arrival rate is calculated but it does significantly change the service center calculation because of the missing service demand parameters for each workload.

There are no easy solutions for the problems caused by virtualization discussed above but dealing with less than perfect modeling measurement data is not new to anyone building a performance model. In the past, it was the norm for operating systems to account for some resource usage 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 the combination of the Menascé technique for calculating the missing parameters with the Simalytic Modeling Technique for modeling multi-tier applications. By using what can be accurately measured (transaction arrivals, transaction response time and utilizations from the VMM), the application's performance can be characterized for each virtual system in the multi-tier environment. This combination addresses the problems with performance modeling virtual environments by reducing the reliance on guest operating system measurements. It also addresses the problems with predictive modeling for virtual environments by effectively using other techniques, such as load test tools, which are difficult to use when some of all or the systems are virtualized.

This combined technique modifies the load dependent service center creation in a Simalytic Model by including information from the VMM about both the VM

| Class Service Demands | | | | | | |
|-----------------------|----------------|-------|----------------|-------|----------------|-------|
| | Actual | | Estimate | | Computed | |
| | Service Demand | | Service Demand | | Service Demand | |
| | 1 | 2 | 1 | 2 | 1 | 2 |
| CPU | 0.030 | 0.045 | 0.038 | 0.038 | 0.035 | 0.037 |
| Disk 1 | 0.025 | 0.038 | 0.032 | 0.032 | 0.021 | 0.047 |
| Disk 2 | 0.050 | 0.045 | 0.048 | 0.048 | 0.048 | 0.044 |

Table 1. Actual vs. Computed Service Demands

utilizations and the host utilization in a way that allows a more dynamic Simalytic Function. The concept is to vary the service demand of each workload at each service center in a virtual environment so that it is consistent with the total host utilization, with the total VM utilization and with the workload response time; while at the same time accounting for the interference from other work in the same guest as the application being modeled. The VMM on the host accurately measures how each VM uses each device. Device utilization by each of the other VMs can be used to infer the interference from other workloads running in them. The objective is to build a comprehensive model of all transaction workloads in the virtual environment to explore the affects on end-to-end response times as arrivals increase.

3.1 Performance Modeling Example

Virtual environments usually support a large number of applications and are therefore significantly more complex than environments where each application runs on dedicated servers. Figure 1 shows a hypothetical three-tier application environment that is typical of many virtualization environments.

To provide a manageable example, we will consider the simple case of modeling processor and disk utilization where there are two applications sharing the same VMs at each tier. This technique can easily be extended to include other resources, such as network devices. The other VMs on each host are then viewed as interference limiting the available resources to the applications of interest (depending on the priority and scheduling techniques of the VMM). Because the process to create a Simalytic Function is inde-

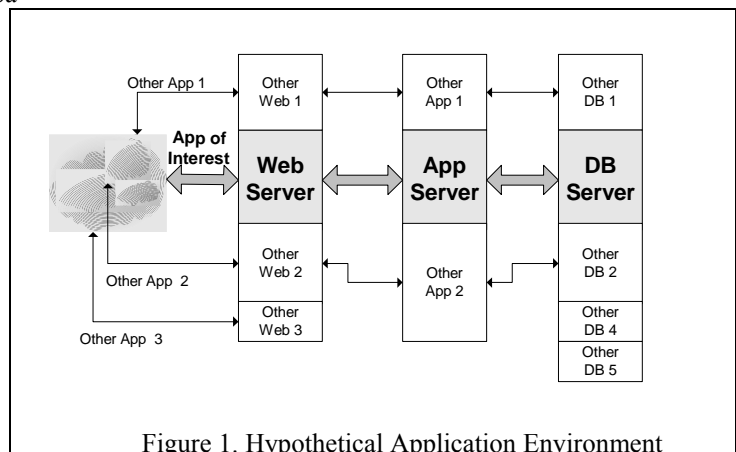


Figure 1. Hypothetical Application Environment

pendent of the specific tier, we will focus only on a single generic virtual host for this example.

The objective of this example is to show that the service demand for multiple workloads can be computed from measurements outside of the guest VM. It models only the transaction workloads of interest and treats all other workloads in the other VMs 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 because of the large number of scenarios that must be modeled. Most modeling efforts in this situation will use a worst-case (high resource utilization on the other VMs) and best-case (low resource utilization on the other VMs) approach and then develop additional scenarios as needed. In each scenario, the other VMs can then be represented as interference that reduces the total utilization available to the modeled VM based on the relative priorities between VMs. For example, if the modeled VM were only allowed to actually use 60% of the total resource then the queuing formulae would be adjusted to show 0.6 as the resource capacity instead of 1.0. The interference from other VMs can be adjusted dynamically as the model execution proceeds by creating a Simalytic Function that changes how much of each resource is available based on VMM measurements of resource usage by the other VMs. This technique can also be used to account for guest operating system and VMM overhead because both will always be higher priority work than the applications. The simplest case is where the VMM allocates processor resources based on relative weights. There are other, and much more complex, VMM policies, which makes how this interference is accounted for very situation dependent. The focus of this paper is to show that all of the missing parameters can be computed from VMM measurements. The interference from the other VMs will not be shown in the example since the adjustments to the formulae can easily be made once the VMM policy is defined. For this example, assume that there are two transaction workloads using the same VM at one tier of the virtualized environment. The workload service demand values used the example in (Menascé, 2008, Table 1) are shown in Table 1 as the “Actual” values. Using these workload and service demand values as if they were actual measurements allows us to create a table of measurements as shown in Table 2 (generated using standard utilization and queuing formulae), which will be used as input values for the model as if they were the actual measured values. The values in Table 3 were created the same way as those in Table 2 but they will be used as if they were measurements from the applications taken after the model was created and will be used for validation of the computed missing values.

$$T_r = \sum_{i=1}^K \frac{D_{i,r}}{1 - \sum_{v=1}^R \lambda_v \times D_{i,v}} \quad (1)$$

(Menascé, 2008, (1) p. 242)

The approach presented in (Menascé, 2008) is to start with the basic open multiclass QN model (1) where:

- R: number of classes of the QN model,
- K: number of devices in the QN model,
- λ_r : arrival rate (in transactions per second [TPS]) of class r requests,
- $D_{i,r}$: service demand (in seconds) for class r requests at device i , and
- T_r : average response time of class r requests measured at the real system.

Because there is no unique solution when more than one service demand is unknown, computing the missing values requires solving a non-linear constraint problem (Menascé, 2008, (13) pp. 244-245) shown here as (2). Each term inside the parentheses in (2) must equal zero for each class r when the solution to the problem satisfies (1). The

Minimize

$$\sum_{r=1}^R \left(T_r - \sum_{i=1}^K \frac{D_{i,r}}{1 - \sum_{v=1}^R \lambda_v \times D_{i,v}} \right)^2 \quad (2)$$

Subject to

$$\begin{aligned} D_{i,r} &\geq 0 \quad \forall i, r \\ \sum_{r=1}^R \lambda_r D_{i,r} &< 1 \quad \forall i. \end{aligned}$$

(Menascé, 2008, (13) p. 244)

first constraint means that the service demand cannot be negative. The second constraint means that the utilization of any device must be less than one (Menascé, 2008, pp. 244-245). Please refer to (Menascé, 2008) for specific details. This constraint will be adjusted to reflect the maximum available utilization for the device within the VM being modeled. Any of the many available packages could be used to solve this problem with these givens (Menascé, 2008, pp. 244-245):

- Given the response times T_r for all classes $r = 1, \dots, R$. Let us call these response time goals.
- Given a subset S of the service demand values $D_{i,r}$, for $i = 1, \dots, K$ and $r = 1, \dots, R$.
- Given the arrival rates λ_r for all classes $r = 1, \dots, R$.

The objective is to:

- Find a set of values for the service demands not included in S such that the resulting response times computed using Equation (1) are the same as the response time goals (Menascé, 2008, pp. 244-245).

Notice that in this application of Menascé’s technique the subset of S of known service demands is null (they are all missing) so the objective is to find all service demands. Because all of the service demands values are missing, many more equations are required. Each row in Table 2 represents a pair of equations like (1), one for each class. Microsoft Excel Solver was used to solve for the computed service demands in this example by finding service demand values that satisfy all of the equations represented in Table 2. This iterative solution method starts with an initial solution as a matrix of estimated service demands (Menascé, 2008, pp. 244-245). The VMM provides good measurements for the utilization of each device so the initial estimate of each service demand is easily calculated using the Utilization Law ($D_i = U_i / \lambda$). Using the lowest arrival rate reduces the queue time in this initial estimate. The initial estimate matrix for this example, 0.0375, .0.315 and 0.0475 for the three devices, CPU, Disk 1 and Disk 2, respectively, was calculated from the first row of Table 2. Table 1 shows all three service demands for both classes; the actual service demands (measurements normally taken from the operating system but missing in many virtual environments), the estimated service demands (the initial guess for the equation solver) and the computed service demands (the equation solver results).

3.1.1 Model Validation

Once service demands have been created for a virtualization environment, they can be validated against actual production measurements. Again, the known good measurements are used (application response time and VM utilizations from the VM) for comparison to model results. A variety of intervals can be selected from the actual production measurements and those arrival rates used with the computed service demands. 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. Table 4 shows the response times and device utilizations calculated with Equation (1) but using the computed service demands. These are compared to Table 3 to see how well model results using the computed service demands match the results using the actual service demands. Figure 2 shows that these match well enough to be very useful in predicting application response times for loads greater than those actually measured.

| Arrival Rate | | Response Times | | Device Utilization | | |
|--------------|---------|----------------|---------|--------------------|--------|--------|
| Class 1 | Class 2 | Class 1 | Class 2 | CPU | Disk 1 | Disk 2 |
| 0.1 | 0.1 | 0.11 | 0.13 | 0.0075 | 0.0063 | 0.0095 |
| 0.5 | 1.0 | 0.11 | 0.14 | 0.0600 | 0.0505 | 0.0700 |
| 1.0 | 1.5 | 0.12 | 0.14 | 0.0975 | 0.0820 | 0.1175 |
| 1.5 | 2.0 | 0.12 | 0.15 | 0.1350 | 0.1135 | 0.1650 |
| 2.0 | 2.5 | 0.13 | 0.16 | 0.1725 | 0.1450 | 0.2125 |
| 2.5 | 3.0 | 0.14 | 0.16 | 0.2100 | 0.1765 | 0.2600 |
| 3.0 | 3.5 | 0.14 | 0.17 | 0.2475 | 0.2080 | 0.3075 |
| 3.5 | 4.0 | 0.15 | 0.18 | 0.2850 | 0.2395 | 0.3550 |
| 4.0 | 4.5 | 0.16 | 0.19 | 0.3225 | 0.2710 | 0.4025 |
| 4.5 | 5.0 | 0.17 | 0.21 | 0.3600 | 0.3025 | 0.4500 |
| 5.0 | 5.5 | 0.19 | 0.22 | 0.3975 | 0.3340 | 0.4975 |
| 5.5 | 6.0 | 0.20 | 0.24 | 0.4350 | 0.3655 | 0.5450 |
| 6.0 | 6.5 | 0.22 | 0.26 | 0.4725 | 0.3970 | 0.5925 |
| 6.5 | 7.0 | 0.24 | 0.28 | 0.5100 | 0.4285 | 0.6400 |
| 7.0 | 7.5 | 0.27 | 0.31 | 0.5475 | 0.4600 | 0.6875 |
| 7.5 | 8.0 | 0.31 | 0.35 | 0.5850 | 0.4915 | 0.7350 |

Table 2. Model Input Class Response Times and Device Utilizations, highlighted shows example form (Menascé 2008)

3.1.2 Practical Application

Even with just two applications of interest, this is a complex model, which will require many model executions to explore all of the scenarios. Creating and running models that are more 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

| Arrival Rate | | Response Times | | Device Utilization | | |
|--------------|---------|----------------|---------|--------------------|--------|--------|
| Class 1 | Class 2 | Class 1 | Class 2 | 4 | 4.5 | 5 |
| 8.0 | 8.5 | 0.36 | 0.41 | 0.6225 | 0.5230 | 0.7825 |
| 8.5 | 9.0 | 0.44 | 0.48 | 0.6600 | 0.5545 | 0.8300 |
| 9.0 | 9.5 | 0.57 | 0.61 | 0.6975 | 0.5860 | 0.8775 |
| 9.1 | 9.6 | 0.61 | 0.64 | 0.7050 | 0.5923 | 0.8870 |
| 9.2 | 9.7 | 0.65 | 0.69 | 0.7125 | 0.5986 | 0.8965 |
| 9.3 | 9.8 | 0.70 | 0.74 | 0.7200 | 0.6049 | 0.9060 |
| 9.4 | 9.9 | 0.77 | 0.80 | 0.7275 | 0.6112 | 0.9155 |
| 9.5 | 10.0 | 0.85 | 0.87 | 0.7350 | 0.6175 | 0.9250 |
| 9.6 | 10.1 | 0.95 | 0.96 | 0.7425 | 0.6238 | 0.9345 |
| 9.7 | 10.2 | 1.08 | 1.09 | 0.7500 | 0.6301 | 0.9440 |
| 9.8 | 10.3 | 1.27 | 1.26 | 0.7575 | 0.6364 | 0.9535 |
| 9.9 | 10.4 | 1.55 | 1.51 | 0.7650 | 0.6427 | 0.9630 |
| 10.0 | 10.5 | 2.02 | 1.94 | 0.7725 | 0.6490 | 0.9725 |
| 10.1 | 10.6 | 2.99 | 2.81 | 0.7800 | 0.6553 | 0.9820 |
| 10.2 | 10.7 | 6.10 | 5.62 | 0.7875 | 0.6616 | 0.9915 |

Table 3. Measured Class Response Times and Device Utilizations

and valleys of different workloads.

Because the VM processes overhead work in addition to the transactions of the applications of interest, the service demand of each workload must be adjusted for the that interference. This is reasonably straightforward if we can make two assumptions. First, that system work (operating system and VM) is always a higher priority than application work and therefore has the effect of reducing the maximum possible utilization in the calculations. This is addressed with a simple adjustment to the queuing formulae. Second, that any other work is at a lower priority and will always be preempted by the applications being modeled.

Any work not meeting these two assumptions would have to be modeled as another workload class, which requires a measured arrival rate (adding a closed QN component to this model is possible but beyond the scope of this paper).

3.2 Load Testing

Load testing (driving increasing numbers of synthetic transactions through the application infrastructure) is often used to understand how both the application software and the infrastructure hardware perform under higher loads than have been actually measured in the production environment. Virtualization greatly increases any load test effort because of the large number of test scenarios needed to understand the interaction between applications and VMs. Simalytic Modeling reduces the number of test scenarios needed because some cases can be eliminated as the workloads do not affect the overall capacity requirements (they are too small during the host peaks) or they do not affect application response time (possibly because of a lower VM dispatch priority). The remaining load test cases can be targeted to creating the Simalytic Functions for the load dependent service centers 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 and to show when response times exceed 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). Using this combined technique further reduces the number of scenarios by changing the load test objective from measuring all possible combinations of application loads to validating the results of application models that use the computed service demands.

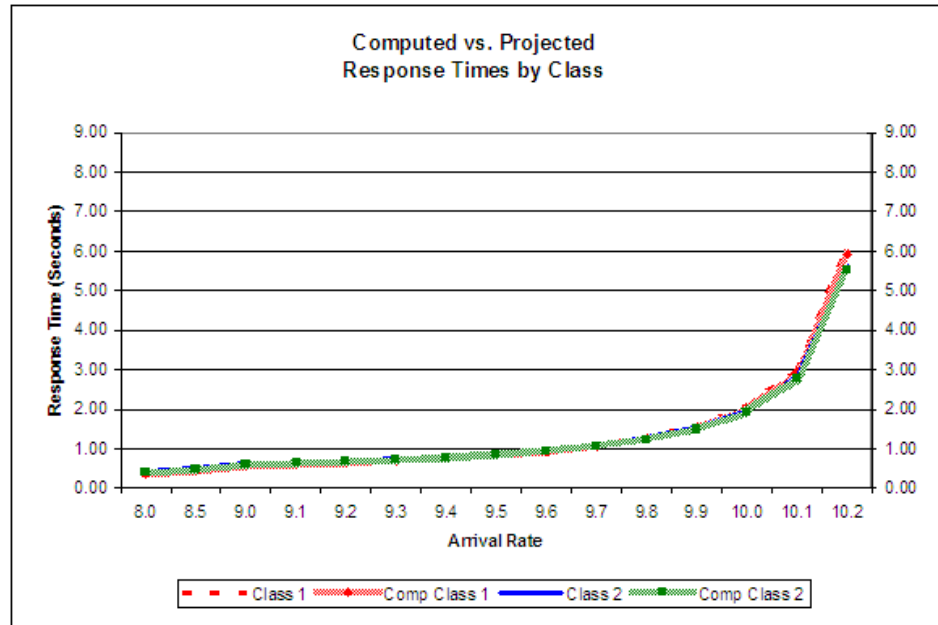


Figure 2. Validation of Computed Values

3.3 Future Research

By using this combined technique, it may be possible to compute all missing service demands for all workload classes in all VMs on a VMM host. This could allow all applications to be modeled together dynamically, with proper adjustments for the static VMM, VM and operating system overheads, and for the policy based limits. Such a model could treat the VMM host as a pool of resources and continually recompute the service demands in a manor similar to what Menascé presented as “Online Estimation of Service Demand” (Menascé, 2008, pp. 245-246), which starts with initial estimates and computes new estimates as new inputs are received. Using this technique, a Simalytic Function could refine the service demands during the simulation model warm-up period and produce higher quality results for analysis. Correlating the transaction arrival traces

| Arrival Rate | | Response Times | | Device Utilization | | |
|--------------|---------|----------------|--------------|--------------------|--------|--------|
| Class 1 | Class 2 | Comp Class 1 | Comp Class 2 | CPU | Disk 1 | Disk 2 |
| 8.0 | 8.5 | 0.36 | 0.40 | 0.6225 | 0.5230 | 0.7825 |
| 8.5 | 9.0 | 0.44 | 0.47 | 0.6600 | 0.5545 | 0.8300 |
| 9.0 | 9.5 | 0.56 | 0.60 | 0.6975 | 0.5860 | 0.8775 |
| 9.1 | 9.6 | 0.60 | 0.63 | 0.7050 | 0.5923 | 0.8870 |
| 9.2 | 9.7 | 0.64 | 0.67 | 0.7125 | 0.5986 | 0.8965 |
| 9.3 | 9.8 | 0.69 | 0.72 | 0.7200 | 0.6049 | 0.9060 |
| 9.4 | 9.9 | 0.76 | 0.78 | 0.7275 | 0.6112 | 0.9155 |
| 9.5 | 10.0 | 0.83 | 0.85 | 0.7350 | 0.6175 | 0.9250 |
| 9.6 | 10.1 | 0.93 | 0.94 | 0.7425 | 0.6238 | 0.9345 |
| 9.7 | 10.2 | 1.06 | 1.07 | 0.7500 | 0.6301 | 0.9440 |
| 9.8 | 10.3 | 1.25 | 1.23 | 0.7575 | 0.6364 | 0.9535 |
| 9.9 | 10.4 | 1.52 | 1.49 | 0.7650 | 0.6427 | 0.9630 |
| 10.0 | 10.5 | 1.98 | 1.91 | 0.7725 | 0.6490 | 0.9725 |
| 10.1 | 10.6 | 2.91 | 2.77 | 0.7800 | 0.6553 | 0.9820 |
| 10.2 | 10.7 | 5.93 | 5.54 | 0.7875 | 0.6616 | 0.9915 |

Table 4. Computed Response Times and Device Utilizations

with the VMM actual measurements is an area for future research.

4. Conclusion

Probably the most significant problem with modeling applications in virtualization environments is the unreliability of service demand measurements from the guest operating system. If the service demand 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 demand 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 Simalytic Function for each load dependent service center based on observed measurements or load test results. Using the Menascé technique to compute missing service demand parameters from known good measurements allows for the creation of much more accurate and useful models.

5. References

- Menascé, D. (2008) Computing Missing Service Demand Parameters for Performance Models. Proceedings of the Thirty-fourth International Computer Measurement Group Conference, December 7-12, Las Vegas, NV, USA, CMG, Inc., pp. 241-247.
- Menascé, D., Almeida, V. and Dowdy, L. (1994) Capacity Planning and Performance Modeling: from mainframes to client-server systems, Prentice Hall, Englewood Cliffs, New Jersey.
- Norton, T. R. (2008) A Simalytic Approach to Modeling Virtualized Environments, in Proceedings of the Thirty-fourth International Computer Measurement Group Conference, December 7-12, Las Vegas, NV, CMG, Inc., pp. 261-266.
- Norton, T. R. (2007) The Myth of Precision Planning: Understanding Capacity in an Age of Virtual Parallelism, in Proceedings of the Thirty-third International Computer Measurement Group Conference, December 2-7, San Diego, CA, CMG, Inc., pp. 471-482.
- Norton, T. R. (2001) The Simalytic Modeling Technique chapter in Performance Engineering, State of the Art and Current Trends. Springer-Verlag, 3-540-42145-9, Berlin/Heidelberg, Germany, pp. 223-238.
- Norton, T. R. (1997a) Simalytic Hybrid Modeling: Planning the Capacity of Client/Server Applications, in Sydow A. (Ed.) Proceedings of the 15th IMACS World Congress 1997 on Scientific Computation, Modelling and Applied Mathematics conference, Volume 6, Application in Modelling and Simulation, August 24-29, Berlin, Germany, Wissenschaft & Technik Verlag, pp. 583-588.
- Norton, T. R. (1997b) Simalytic Modeling: A Hybrid Technique for Client/Server Capacity Planning, in Obaidat, M. S. and Illgen, J. (Eds.) Proceedings of the Summer Computer Simulation Conference (SCSC '97), July 13-17, Arlington, VA, USA, Society for Computer Simulation International, pp. 172-177
- Salsburg, M., Karnazes, P., and Maimone, B. (2006) It May be Virtual ... but the Overhead is Not, in Proceedings of the Thirty-second International Computer Measurement Group Conference, December 3-8, Reno, NV, USA, CMG, Inc., pp. 675-686.