Simalytic Hybrid Modeling: Planning the Capacity of Client/Server Applications

by
Tim R. Norton
Doctoral Candidate
Department of Computer Science
Colorado Technical University

Submission Draft for the 15th IMACS World Congress 1997

Abstract

The designs of computer applications are changing from single system designs to cross platform client/server designs utilizing the features of different types of computers, operating systems and networks. Modeling provides the necessary tools and techniques for planning the capacity of large computer installations using multiple systems by constructing an enterprise view of the applications with an understanding of each of these areas and the inter-relationships between them.

*The "Simalytic" (**Sim**ulation/An**alytic**) Modeling Technique[1] is a hybrid technique for planning the computer system capacity requirements of complex multiple-platform computer applications by modeling at an enterprise level. This technique uses a general purpose simulation tool as an underlying framework and an analytic tool to represent individual nodes when predicting capacity requirements for an application across an enterprise. It combines both platform-centric tools (limited features but detailed platform information) and general purpose tools (rich low level features) to address today's large heterogeneous enterprises. This methodology takes advantage of features in the different techniques (simulation vs. analytic queuing theory) as well as features in the different tools (platform-centric vs. general purpose) by defining the interface between the simulation framework and queuing theory node models.*

1. Introduction and General Background

Applications that once would have been implemented as batch systems on a single computer are now multi-platform on-line transaction processing client/server systems with GUI (graphical user interface) front-ends on PWS's (programmable work-stations) attached to departmental servers and mainframe repositories. These new applications utilize the features and services of different types of computers (mainframe, mid-range, desktop) running different operating systems connected by a variety of communication network techniques (Hatheson 1995; Wilson 1994).

As applications move into this new client/server world, how do we select the right systems at each level and, once selected, how do we insure those systems are the right size? If any one of them is too small the whole application will fail. If any are too big, the cost of running the application may exceed the revenue it generates. Neither is a very attractive situation.

The objective of capacity planning is to find the successful middle ground. Today, planning the capacity of large computer installations with multiple systems requires an understanding of not only the operating systems, the platforms, the clients, the servers, the networks, the transaction systems, etc., but more importantly, the applications and the relationships between them. Once those relationships are defined and understood, the application's performance can be assessed against the business ob-

[1] *Simalytic™, Simalytic Modeling ™, Simalytic Modeling Technique ™* and *Simalytic Enterprise Modeling ™* are trademarked by Tim R. Norton
All other trademarked names and terms are the property of their respective owners.

jectives and goals. Projected business volumes are then modeled to predict the capacity required to meet those goals at future volumes.

There are many modeling tools and techniques that address both performance and capacity for each of the systems in today's multi-platform environment (Pooley 1995; Smith 1995). The Simalytic Modeling technique provides a bridge across these existing tools to allow the construction of an enterprise level application model that takes advantage of models and tools already in place for planning the capacity of each system.

The major topics covered in this paper are:

- Section 1, *Introduction and General Background*, is a brief overview of modeling techniques related to capacity planning. An expanded discussion of the topic can be found in (Norton 1996).
- Section 2, *Simalytic Modeling Methodology*, is a description of the methodology to be used, which includes the mathematical foundation behind the Simalytic Modeling technique, a discussion of its application to address the issues from Section 1 and some preliminary results.
- Section 3, *Conclusion*, is a discusses the implications of the results, the application of the technique and areas of further research.

1.1 Capacity Planning

The capacity of a system can be measured many different ways, depending on the business the system supports. Generally, the way a system is measured centers around the performance of one or more of the applications. The system "has enough capacity" if everything is getting done when it is needed. This may sound like a simplistic statement, but the key to understanding the capacity of a system is the definition of the performance objectives. (Domanski 1995 13; Rosenberg and Friedman 1984; Wicks 1989; Wilson 1994). Therefore, capacity planning is making decisions about the resource requirements of a given computer system based on the forecasting of future application performance using the goals and expectations of the business. What do we have to buy and when do we have to buy it to make sure that the applications that run the business perform at the level required to insure the business succeeds?

1.2 Transaction Based Applications

Although there are still many important batch applications, this discussion will center around transaction based applications for several reasons. First, the multi-platform client/server systems are generally focused toward small real-time units of work such as transactions rather than the large long-running units of work associated with batch. Second, large batch applications generally have long execution times and points-in-time when all work must be completed. It just doesn't matter when a third of the paychecks are printed; they *all* must be ready when the time comes to distribute them. Third, batch workloads have a much lower arrival rate, often once a day or less, and tend to be serialized. Fourth, transaction based applications are much more sensitive to the demands of momentary peak loads where batch based applications are more sensitive to scheduling issues and interference from higher priority workloads. Therefore, the traditional single system view will continue to provide adequate performance and capacity planning for batch workloads where transaction workloads require a more enterprise view to understand the relationships between responsiveness as a whole and the individual platform resource requirements.

What is a transaction? Transaction processing systems, often referred to as OLTP (On-Line Transaction Processing), allow the end-user to enter a relative small independent unit of work into the system and receive some information as a response in near real-time. Transactions, which can be defined from different points-of-view, include entering an order at a terminal (business transaction), an SQL command (database transaction) or some keystrokes followed by a carriage-return (interactive transaction). In one sense, each keystroke a user types in a text editor is a transaction because a small unit of work (the keystroke) is sent to the sever, acted upon and information is returned to the user (the keystroke is echoed). A transaction might be defined as messages received from or sent to 3270 terminals (which were really early PWS's) by an OLTP system such as CICS or IMS or as logical units of work marked by "commit" commands by database systems such as Oracle and Sybase (BGS 1996).

The concept of a transaction is important because it is meaningful from the end-user's point-of-view. Transactions can be counted to establish load (e.g. arrival rate) and measured to establish performance (e.g. response time). The responsiveness of the transactions associated with an application deter-

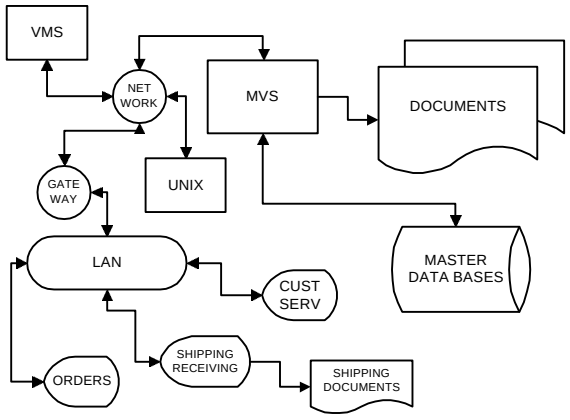


Figure 1 The "Client/Server" System

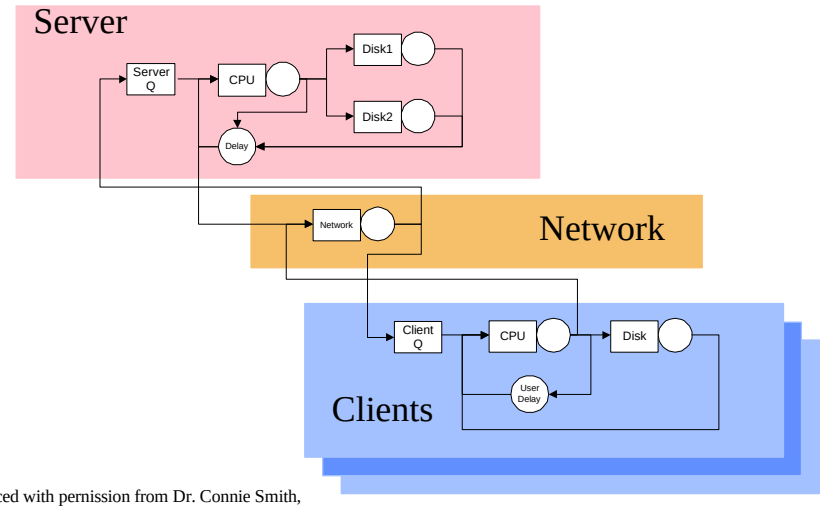
mine if that application meets the needs of the business. A customer service representative can answer more inquiries more effectively when the requested information is presented in one or two seconds than if it is presented in ten or twenty minutes. Modeling techniques can then be used to predict application responsiveness at higher transaction rates.

1.2.1 Client/Server Environment

Figure 1 The "Client/Server" System shows a hypothetical client/server environment to illustrate the problem of modeling application performance. Which techniques and products are chosen doesn't matter, but what is important is understanding that any of the client applications on any of the PWS's can, and will, send transactions to several of the legacy applications to provide the end-user a screen of complete and interrelated information. For example, the Order Entry user may type in the name of an existing customer and get not only their address but any pending or past orders and the status of their account. This may provide better service, but it also causes transactions to be sent to each of the other systems.

Capacity planning in a client/server environment is much harder than in a single computer environment. In the example above, if the Shipping workload outgrows the Unix system, it can impact the responsiveness of the Order Entry transactions. In addition, growth in the Order Entry workload will now impact the Unix system, but only if the orders are from existing customers.

Modeling in this environment is truly a challenge. Each of the systems require a different knowledge base and expertise (Gunther 1995; Hatheson 1995). None of the systems can be modeled independently because the transaction arrival rate for one system may be dependent on the response times of the others. *Figure 2 An Enterprise Model* shows a very simplistic model for each of the major areas of a client/server application and, although it only shows a single server, the interdependence is evident. The responsiveness of one part of the model (server, client or network) will have an impact on the other two. In a more realistic application the software on the client PWS could issue transactions to several servers (request everything about customer #123) or it might issue them in series and have to wait for one response before sending the next (request Jones' customer number; then request everything about that number). While the former situation will cause the instantaneous peaks to synchronize on all of the servers; the latter will slow everything down as one of the servers becomes overloaded and its response times increase. "While it is important to be able to model specific UNIX or NT hardware, the problem we face is modeling the environment that has a diverse collection of hardware, operating system, database management system, and network hardware."



Reproduced with pernission from Dr. Connie Smith, Performance Engineering Services.

Figure 2 An Enterprise Model

(Domanski 1995).

1.3 Modeling Capacity Projections

The use of models to assess and predict the performance of computer systems is not new. Capacity planning has always relied to some extent on modeling because of the need to predict future requirements. A capacity planner can analyze the workloads and make predictions based on experience or simple trending. Models can be constructed to understand how an application functions without any intention to predict future performance. The area of interest here is the intersection of the two fields; models used to predict capacity requirements based on performance expectations. Each of the models in *Figure 2* is well defined, but when taken as a whole, the complexity rises quickly as additional servers and clients are included in the model.

1.3.1 Approaches to Capacity Planning Modeling

The approaches to capacity planning range from the application of rules-of-thumb to full scale benchmarks of the application or system (Brunetto 1984; Gilmore 1980; Hanna 1988; Mills 1991). *Figure 3 Capacity Planning Approaches* shows the relationship between these approaches. (This figure is not to scale; no meaning should be inferred about the absolute level of either axis, only that the relative position of an item implies some amount more than an item to the left or below.) Business Analysis (Rules-Of-Thumb) and Trends (Linear Projections) rely on historical analysis and the assumption that future performances is a direct extension of past performance. Benchmarks can be the most accurate because they actually implement the applications, but at the greatest cost. Modeling is the middle ground between the high cost and effort of benchmarks and the low prediction ability of trends.

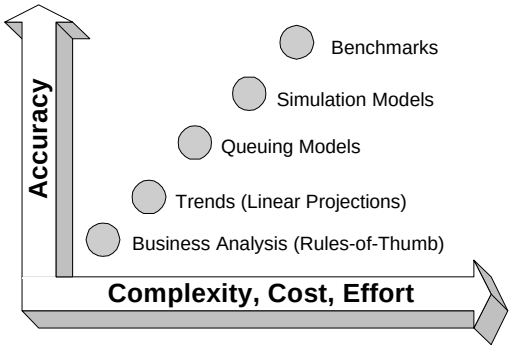


Figure 3 Capacity Planning Approaches

1.3.2 Response Time Modeling

The key to the capacity planning methodology discussed so far is the ability to predict the performance of a future workload given a desired system configuration. As applications move more towards being transaction based, the definition of application performance becomes centered around transaction response time. For this reason, modeling the transaction response time of a application is crucial to the ability to predict the future performance of that application.

There are two basic modeling techniques used for computer performance modeling: simulation and analytic queuing theory (Kobayashi 1981; Menascé, Almeida, and Dowdy 1994). Either of these techniques will build a model that represents the major components of the computer system to be modeled. A third technique, hybrid modeling, is the combination of both simulation and analytic techniques in a single model (Kobayashi 1981).

1.4 Modeling Tools

In addition to the choice between analytic and simulation tools, the capacity planner or performance analyst has the choice between platform-centric and general purpose tools. The basic difference between these two groups is the problem set the tools were designed to address.

1.4.1 Platform-Centric

Platform-centric means that the tool contains detailed information about the platform, but does not allow more than one platform to be modeled at a time. A platform-centric tool would include information about the number and type of processors for each model of a system build by a given vendor (e.g. an IBM 9021-982 has eight processors @ 60 MIPS each). Platform-centric models are generally easier to build because they are made of "building blocks" already defined to the tools and the relationships between them are fully understood by the model. However, these tools cannot be used to model an environment not built into the tool. For example, a tool with the above "building blocks" could not be used to model Unix running on an HP system. Although many platform-centric tools allow the user to define new servers with new performance characteristics, they generally do not provide large libraries of device and system definitions dramatically different from the supported platform. Platform-centric tools are *generally*

implemented using analytic or queuing theory modeling techniques and process performance and configuration data collected from existing running systems.

1.4.2 General Purpose

General purpose means that the tool contains the features to allow the user to model anything they would like to build, but with little or no "built-in" understanding of any given platform. These tools are used to model more than just the hardware, including such things as the design of an application, traffic flows and communications networks. Using the example above, the model builder would have to understand the design of the IBM 9021-982, how the eight processors communicate, and what is involved in completing a unit of work within the operating system. The modeler using a general purpose tool would be required to either build a submodel to simulate the underlying architecture or to determine a delay value to use whenever the event happened. Simulation of the architecture is much more accurate but also much more difficult and time consuming, and may require information the model builder does not have. Although many general purpose tools provide libraries of sub-models for a variety of systems and devices, they generally do not provide the required level of granularity, being either too general or too detailed for the situation. Building the relationships between the submodels is generally part of the overall model construction and may require an in-depth understanding of all of the submodels used. General purpose tools are *generally* implemented using simulation modeling techniques.

1.5 Hybrid Modeling

A hybrid modeling technique as a combination of both simulation and analytic queuing theory. The reasons for using a hybrid technique for capacity planning modeling are fundamentally the same as for other areas that have turned to hybrid techniques: accuracy improvement and cost reduction. By merging the best features of the two techniques, along with the use of existing modeling tools, the capacity planner can produce a more accurate model with less effort and lower processing requirements.

2. Simalytic Modeling Methodology

"Simalytic" (**Sim**ulation/An**alytic**) Modeling is a hybrid modeling technique. The methodology uses a general purpose simulation modeling tool as a underlying framework and an analytic modeling tool (or the results thereof) to represent the individual nodes or systems. The problem addressed by Simalytic Modeling is at the intersection of several areas: capacity planning, modeling (both simulation and queuing theory), client/server transaction processing systems, and commercial tools (both general purpose and platform-centric). The goal of a Simalytic Model is to predict the capacity requirements of an application executing on heterogeneous computer systems by creating an enterprise level application model.

There are two key differences between the existing modeling tools and the Simalytic Modeling technique. The first is the ability to use the results from not only a different tool, but a different modeling technique altogether, as a submodel within an enterprise model. The second is the ability to use the results from tools or techniques already being used to model individual nodes in the system. These differences reduce the time and effort to build an enterprise level model by using the results from commercially available platform-centric tools or existing detailed application models.

2.1 Methodology

Simalytic Modeling brings together existing performance models and application information. One of the problems with queuing theory is the reliance on averages, such as average response time, average service time and average arrival rate. These models are generally more efficient to execute than simulation models, but, because of the use of averages that normalize all activity for each server in the model, they are often less accurate. The goal of Simalytic Modeling is to model the application over longer periods of time to understand the application dynamics without increasing the error due to greater variation in the data items used for the above averages. When using commercial queuing theory tools, it is generally understood that shorter intervals[2] usually produce better model results because there is less variation in the measurement data. The node model can produce very accurate response time predictions when built with a queuing theory tool using a short data collection interval to minimize the variability in the data.

[2] In this context, 'interval' refers to the time period for which measurement data was collected to be used in building a model. Interval selection is the analysis of all available measurement data to determine the interval that is most representative of the application situation to be modeled.

Exponential interarrival time distribution (also referred to as Poisson arrival distributions is used because it is generally considered to most closely the actual arrival patterns in transactions systems (Buzen and Potier 1977). An additional advantage is that the arrival rate at a server from multiple sources is the sum of the arrival rates generated by each source and that the arrival rate at one of multiple servers from a single source is the original rate times the probability that server will be selected, which allows the consolidation of servers. Exponential service time distributions are also useful because of the Markov or "memoryless" property (the service delivered to one customer does not predict the service that will be delivered to the next) (Allen 1978; Allen 1990, p. 255; Kobayashi 1981, p. 103-5; Pooch and Wall 1993, p. 342-3). Trace-driven simulations are of more use in capacity planning and performance modeling because they remove one major issue in model construction; transaction arrival distribution. The increase in arrival rate to represent the growth of a workload is based on the current arrival rate and distribution.

Regardless of the underlying technique used in the simulation tool, there are two important characteristics of these tools that make them well suited to be used as the framework in Simalytic Modeling. The first is the ability to maintain the identity of each transaction, and its associated attributes, throughout the entire model and have the model react to these attributes. The second is the ability to preserve the specifics of the interarrival times between individual transactions. Even though Poisson distribution is accepted as generally the most representative in models of these types of systems, other distributions can be used if analysis of the trace data shows another distribution to be more appropriate.

Simalytic Modeling is based on a hybrid technique that allows the models to use the best features of each tool. The concept of submodels allows each part to be represented by a different technique. Submodels are supported by most of the commercially available modeling tools, but with varying abilities to utilize completely different techniques in the submodel. Submodels are a key concept because they allow some part of the model to be replaced with a different model as long as it provides appropriate functionality and results; similar to the FESC (flow-equivalent service center) decomposition technique discussed in (Menascé, Almeida, and Dowdy 1994).

2.1.1 Methodology Assumptions

Simalytic Modeling is not a technique for collecting data or measuring systems or applications. There are several underlying assumptions that must be true before the Simalytic Modeling technique can be used:

- Each workload to be modeled must be consistently defined across all of the models used.
- The application details must be understood at the enterprise level, which includes transaction arrival distributions.
- A valid model (proven to produce accurate predictions) must exist for each system or node to be included in the application enterprise model.
- The simulation tool selected for the enterprise model framework must support submodels, must be able to invoke external functions and must support the modeling of individual transactions.

2.1.2 Methodology Process

Once the model builder has all of the fundamental information, she can construct an enterprise level model. The simplest way to do this is to construct a very high level simulation model of the enterprise where each system is a single server capable of some amount of parallelism defined by the architecture of each system and of the application. Then, instead of using a pre-defined service time, each server would use a transformation function that maps the transaction arrival rates to service times. In the enterprise model the service time and the response time for each server will be the same because the queue time is accounted for in the response time data for the server. Each node in the simulation model must allow enough parallelism to avoid queuing to enter the node.

Continuing with the same example, some number of the Order Entry transactions would be routed to the Shipping server. Assume it has been determined that Shipping can provide a response time of one second when arrivals are less than three per minute and a response time of two seconds when arrivals are more than three per minute. When the Order Entry transaction rate increases such that more than three per minute are sent to Shipping, the response time will jump from one to two seconds. This is an overly simple example, but it illustrates the point. The increased service time at Shipping will cause

the overall response time for those transactions to increase, which will be seen as a longer average response time or reduced through-put for the application.

Figure 4 Simple Application Model shows a diagram of this model. The response time is measured from Arrivals to Departures, either through the Shipping node or around it. If there is a limit on the number of transactions that can be active in the Order Entry system at any given time, then there could be some queuing to get into the system. This example shows how the Simalytic Model connects what is happening in the application on the different servers. If the Order Entry system is modeled by itself, the workload representing the long (Shipping) transactions would not reflect the increased response time due to the load at Shipping. Because of the additional application information in the Simalytic Model, it could adjust the service time in the Order Entry server by replacing the measured wait time component of the response time with the projected delay from the Shipping server.

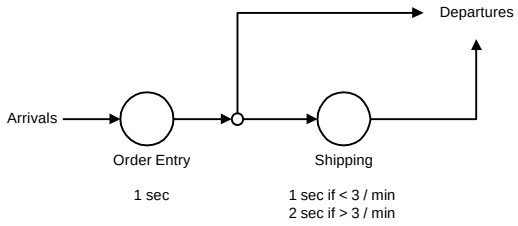


Figure 4 Simple Application Model

The question is "how does the Shipping server know what arrival rate to use for a single transaction?" The arrival rate must be calculated for each transaction based on how long it has been since the prior transaction (the transaction interarrival time). If the interarrival time is less than 20 seconds, then the arrival rate to account for that interarrival time would have to be greater than three per minute. If the interarrival time is longer than 20 seconds, then the arrival rate would have to be less than three per minute. Therefore, knowing the interarrival time between each pair of transactions, the model can calculate the pseudo-arrival rate at each server. As shown in *Figure 5 Shipping Transaction Arrivals*, when the transaction arrivals are close together the response time is high and when the arrivals are further apart the response time is low.

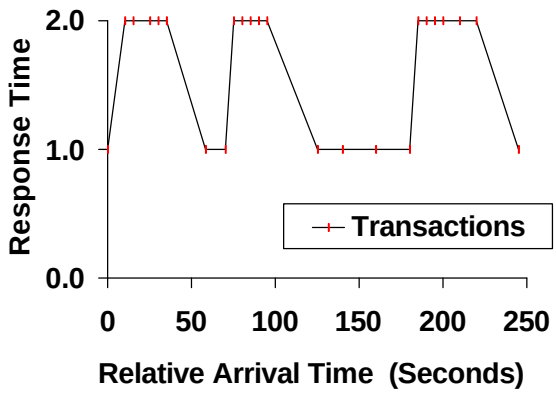


Figure 5 Shipping Transaction Arrivals

The next step is to analyze the model using the business objectives. Assume that the manager of the Order Entry department has requested a model to determine when the Order Entry system will need to be upgraded in order to maintain the required response time of less than 1.7 seconds. The arrival rate is assumed to have a constant increase over the next 18 months (the scope of the analysis) and the percent of the Order Entry transactions must also query the Shipping system is assumed to be 30%. The response time goal for the Shipping system is less

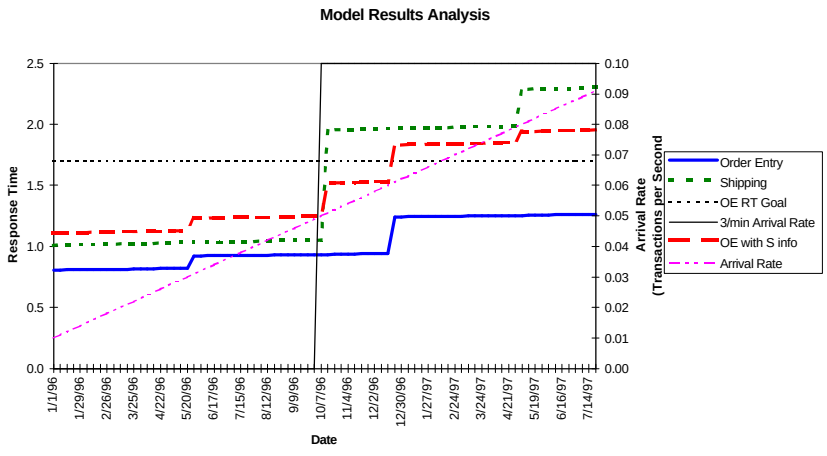


Figure 6 Model Results Analysis

than 10 Seconds (because these transactions generally do not involve a waiting customer) and this response time is acceptable. The objectives of the analysis are to answer two questions: "When does the Order Entry system fail to meet the business response time goal?" and "What will fix the problem?".

Figure 6 Model Results Analysis shows the hypothetical results of this example model. When each of the systems are analyzed independently, neither of the response times ever approach the business goal of 10 seconds for the Shipping transactions and 1.7 seconds for the Order Entry transactions. However, when the relationship between Shipping and Order Entry is added to the chart in the form of results from a Simalytic Model, the revised Order Entry response times show that system will need to be upgraded by year end, well within the scope of the analysis. In addition, the 'fix' to the problem is to upgrade the Shipping system, which never exceeds its response time goal. The Simalytic Model allows the analyst to see the impact of relationships that, although known, may not be full appreciated.

2.2 Foundation

The queuing theory mathematical formula for the average response time (T) of transactions in a simple single-server analytic model is shown in *Equation 1 Analytic Response Time Formula* from (Menascé, Almeida, and Dowdy 1994, p. 108) where S is the average time spent at the server and l is the average arrival rate of transactions. A detailed description of this formula and its application can also be found in (Buzen 1984).

Equation 1 Analytic Response Time Formula

$$T = \frac{S}{1 - l\ S}$$

The mathematical formula to estimate the average response time (T) of transactions in a simple single-server simulation model is shown in *Equation 2 Simulation Response Time Formula* from (Menascé, Almeida, and Dowdy 1994, p. 108) where T_i is the response time of the i^{th} transaction and n_t is the total number of transactions that visited the server during the simulation.

Equation 2 Simulation Response Time Formula

$$T = \frac{\sum_{i=1}^{n_t} T_i}{n_t}$$

The Simalytic Modeling technique combines these two mathematical formulae into the Simalytic Modeling formula shown in *Equation 3 Simalytic Response Time Formula*. As the framework for a Simalytic model is a simulation model, we start with the simulation response time formula shown in *Equation 2*. The server time from each iteration (T_i) is replaced with a transformation function f (λ_i), where i is the iteration and λ is the arrival rate per second calculated from the interarrival time by dividing the number of simulation clock ticks per second (b) by the difference in the simulation clock value for the current iteration and the prior iteration (c_i - c_{i-1}) as shown in *Equation 3*.

The transformation function f is based on the results of the analytic response time formula similar to the one in *Equation 1*, either directly or indirectly, and represents whatever is required to return the correct response time for each given arrival rate. It may be a known and understood formula, as in the case of *Equation 1*, or it may be the results derived from proprietary algorithms implemented in a commercial tool. Directly means that the simulation modeling tool would invoke a submodel to calculate and return the response time based on λ. Indirectly means that the analytic formula has been invoked at some other time for some subset of the expected values for λ and the response times placed in a look-up table. The simulation model then invokes a submodel that returns the response time for the closest λ found in the table.

Equation 3 Simalytic Response Time Formula

$$T = \frac{\sum_{i=1}^{n_t} f(l_i)}{n_t}$$

where l = arrivals per second as:

$$l_i = \frac{b}{c_i - c_{i-1}}$$

where c = simulation clock value

and b = simulation clock ticks per second

Figure 7 Interarrival Time Example shows an example of how the arrival rate can be calculated from the interarrival times between transactions. The i^{th} transaction (t_i) arrives at the clock value of c_j where j is the clock counter, but as it is the clock value for the i^{th} transaction, it is also referred to as c_i. The arrival of the prior transaction (t_{i-1}) is at c_{i-1} (or c_{j-3}). Therefore, the interarrival time for t_i is ir_i, which starts at the prior arrival (a_{i-1}) and is calculated as the

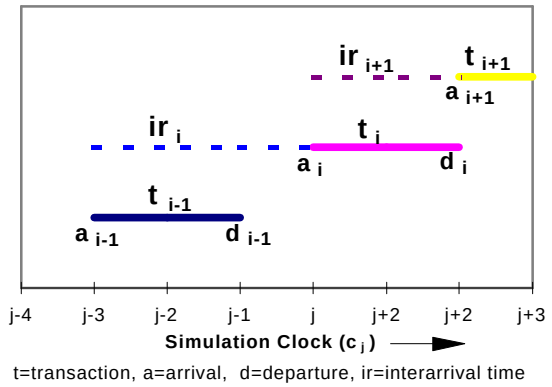


Figure 7 Interarrival Time Example

clock value at a_i ($c_i \equiv c_j$) minus the clock value at a_{i-1} ($c_{i-1} \equiv c_{j-3}$) which is three. If there are six clock ticks per second, then 6/3 = 2, or an arrival rate of two transactions per second. The same process is used to calculate the pseudo-arrival rate for t_{i+1}. The clock value at a_{i+1} ($c_{i+1} \equiv c_{j+2}$) minus the clock value at a_i ($c_i \equiv c_j$) which is two. At six clock ticks per second 6/2 = 3, or an arrival rate of three transactions per second.

Once the arrival rate is calculated, the transformation function (f) is called with the rate for that transaction (λ_i), two in the first case and three in the second.. The function will approximate the results of the queuing theory formula *Equation 1 Analytic Response Time Formula* to a greater or lesser degree, depending on the complexity designed into f.

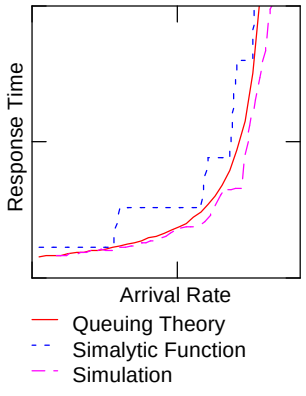


Figure 8 Simalytic Function

Figure 8 Simalytic Function shows a stylized view of the relationship between this function (the short broken line) and the results of *Equation 1* (the solid line) and the results a of series of simulations at different arrival rates (the long broken line).

The calculation of the arrival rate for the queuing theory submodel is a key principle to Simalytic Modeling because it provides the bridge between the two types of models (simulation and queuing theory). One of the assumptions for Simalytic Modeling (see section *2.1.1 Methodology Assumptions*) discusses the importance of using only results from validated node models in the Simalytic Model. The ability to use the pseudo-arrival rate calculated from the transaction interarrival times between each pair of transactions is valid because the queuing theory models for each node has already be proven to produce acceptable predictions using those arrival rates and the function used for the server service time is a transformation function between the two valid models. The arrival rate calculated from the interarrival time is really a way of obtaining the arrival rate that would cause the given interarrival time in a steady state. Using that arrival rate will already be validated as part of the construction of the node level model.

Even assuming a best case situation where the interarrival times of the actual transactions are uniform (no variation), there would still be variation in the response times due to variation in the workload (different transactions, different calculations, different logical path, etc.) and due to variations in the device service times (CPU cache and pipeline, disk cache and rotation, interference form higher priority workloads, etc.). When using any queuing theory model, it must be accepted that the predicted results represent an average based on the average of each of the parameters over the data collection interval. This is similar to the FESC (flow-equivalent service center) decomposition technique discussed in (Menascé, Almeida, and Dowdy 1994), used for solving complex queuing theory models. The system to be modeled is divided into parts that can each be analyzed and solved independently. Each part is then replaced with a single server that is representative of the flow through that part of the system. One of the main points in each of the discussions (Menascé, Almeida, and Dowdy 1994, p. 163-7 and 236-9) is that the FESC must be solved independently and must maintain the 'flow' of the overall model (i.e. the behavior of each FESC must be almost indistinguishable from the subsystem it replaces). The subsystem model is solved to obtain

the subsystem throughputs as a function of multiprogramming level (for closed models) or the subsystem response times as a function of arrival rate (for open models). The FESC is then substituted for the subsystem in the overall model using the function to describe the subsystem's behavior. (Menascé, Almeida, and Dowdy 1994, p. 236) cites (Chandy and Sauer 1978) that the flow-equivalent method yields exact results when applied to closed single class product form models and cites (Cortois 1975) that little error is introduced if the transition rate within the submodel is much greater than the interaction rate between the submodel and the overall model (which will necessarily be the case when the submodel represents an entire independent system or node).

Equation 3 only calculates the response time for a single workload on a single server. The response times for additional workloads and servers would be calculated the same way. The average system response time could then be calculated by adding the response times together based on the probability of each workload visiting each server. As can be seen by this simple example, the calculations will quickly grow out of hand. To avoid this, the Simalytic Modeling Technique uses existing simulation and queuing theory tools together to implement a Simalytic Model. In addition, the simulation tools allow transactions to be assigned attributes that can contain application design information not available in the measurement data.

2.3 Preliminary Results

The preliminary results were produced using MathCAD to implement the both the queuing theory formula and the simulation results. These graphs were generated using an exponential distribution of service times around the average shown because this more closely matches the M/M/1 queuing model. The QRT function is an implementation of *Equation 1 Analytic Response Time Formula*. The SrvRTime function is an implementation of *Equation 3 Simalytic Response Time Formula*. The CurvFt function is the result of fitting a curve to the average results of actual simulations for each arrival rate. The simulation builds an array of response times at 50 different arrival rates between .1 and (1/service time)+.1. The average for each different arrival rate is then plotted and a curve fit to the points. Graphs are shown for two different service times; graphs at other service times produced similar results.

The graph in *Figure 9* shows the correlation between the three techniques at a service time of .1 seconds. The step function in the response times for the Simalytic function is larger than desired because of the simplistic nature of the MathCAD implementation. It can be minimized to the extent desired by increasing the number of steps.

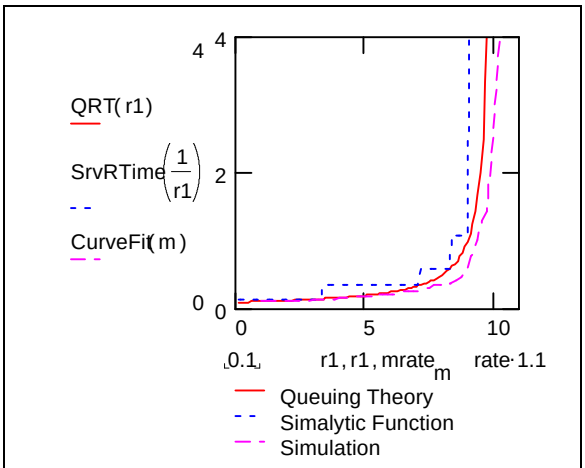


Figure 9 Response Time Comparison for Service Time of .1 Seconds

The graph in *Figure 10* shows the correlation between the three techniques at a service time of .2 seconds. Again the three track nicely. The little dip in the simulation line comes and goes with each recalculation of the random number generator for the distributions of service times.

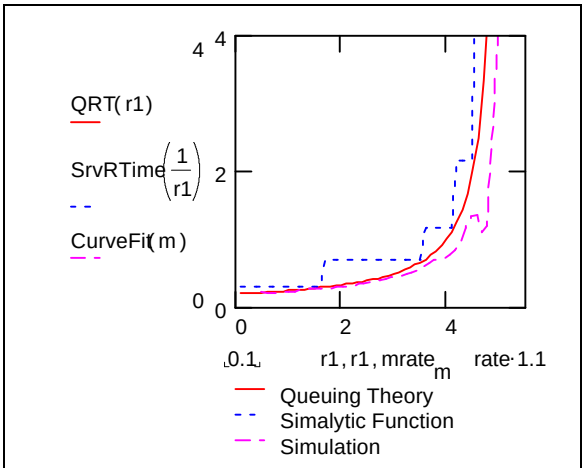


Figure 10 Response Time Comparison for Service Time of .2 Seconds

3. Conclusion

Capacity planning for single-platform computer systems has developed over the years into a well

disciplined field, but only if the input parameters and goals are well defined. Predicting the resource requirements in a client/server environment is possible, but again, only if the input parameters and goals are well defined. Applications designed to exploit a client/server architecture greatly increase the complexity of both the computer system configurations and the applications themselves. Predicting the responsiveness of those more complex applications requires a more complex modeling methodology.

Modeling an application at the enterprise level requires an understanding of the applications and measurements of the transaction response times. Different modeling techniques (simulation, analytic queuing theory or hybrid) and different modeling tools (platform-centric or general purpose) can be used to predict transaction response times for individual systems or nodes. But none of these can be used alone to produce a detailed enterprise level model at a reasonable development cost. The expense, time and effort to plan the required future capacity of a system must be substantially less than the cost that is being avoided. It has been possible to devote a great deal of time, effort and money to capacity planning in the mainframe arena because the equipment costs to be avoided were so very large. The lower equipment costs and scarcity of experienced planners in the distributed environments have often made the cost to be avoided less than the cost of planning. Unfortunately, both of these situations are changing as mainframe equipment costs spiral down and client/server complexities push the enterprise costs up. The cost to be avoided may still be too small to justify the effort, but the key is finding where the real problem is. Adding equipment that fixes several non-problems quickly changes the equation in favor of effective capacity planning.

Simalytic Modeling can be used to take advantage of existing application and system models to reduce the time and effort to produce detailed enterprise level models. Using this technique will both improve the understanding of the application as well as identify which systems require more detailed analysis and which systems will continue to meet the business needs without additional equipment. The implementation of the technique using any of the many existing tools not only protects the investment an organization has made in tool acquisition and training, but it also will reduce the time and effort to produce a model that will predict the impact of business growth on the entire enterprise.

3.1.1 Future Research

The author is currently developing a modeling tool that will be built on a simulation framework and implement the results of a queuing theory model as a Simalytic Modeling function. This tool will allow someone interested in predicting the performance of an application based on the anticipated growth in business transactions to construct a simulation model of the application using the results of some other technique to control the service times at each node.

{NOTE TO REVIEWER: This tool should be at a demonstrable stage when this paper is presented. Additional results will also be included in the camera ready paper. The background paper (Norton 1996) will be published in December 1996. It and other related work (both published and unpublished are available at the author's web site: http:\\www.simalytic.com.}

4. Acknowledgments

The author would like to thank Dr. Jeff Buzen and Mr. Rick Lebsack for their interest and in-depth critiques of the early versions. A special thanks is also expressed to Dr. John Zingg, Dissertation Committee Chair, for his insight and assistance.

5. Bibliography

Allen, Arnold O. 1978. Probability, Statistics and Queueing Theory With Computer Science Applications. New York, NY: Academic Press.

Allen, Arnold O. 1990. Probability, Statistics and Queueing Theory With Computer Science Applications. San Diego, CA: Academic Press.

BGS. 1996. UNIX Client Server Sizing and Performance: BGS Systems, Inc. White Paper.

Brunetto, Anthony F. 1984. Benchmarking Decisions - A Management Tutorial. CMG Proceedings,: 755-761: Computer Measurement Group.

Buzen, Jeffrey P. 1984. A Simple Model of Transaction Processing. CMG Proceedings,: 835-839: Computer Measurement Group.

Buzen, J.P. and D. Potier. 1977. Accuracy Of Exponential Assumptions In Closed Queueing Models. CMG Proceedings: Computer Measurement Group.

Chandy, K. and E. Sauer. 1978. Approximate methods for analyzing queuing network models of computing systems. ACM Computing Surveys 10 (3).

Cortois, P. 1975. Decomposability, instabilities and saturation in multiprogramming systems. Communications of the ACM 18 (7).

Domanski, Bernard. 1995. Capacity Management for Client-Server Architectures. Enterprise Systems Journal 10 (11): p78(6).

Gilmore, Martha R. 1980. Capacity Planning Methodologies. CMG Proceedings: Computer Measurement Group.

Gunther, Neil J. 1995. Performance Analysis and Capacity Planning for Datacenter Parallelism. CMG Proceedings: Computer Measurement Group.

Hanna, Carolyn. 1988. A Production Control Model Of On-Line Systems A Capacity Planning Overview. CMG Proceedings,: 592-598: Computer Measurement Group.

Hatheson, Amanda. 1995. Two Unix Client/Server Capacity Planning Case Studies. CMG British Proceedings: Computer Measurement Group.

Kobayashi, Hisashi. 1981. Modeling and Analysis: An Introduction to System Performance Evaluation Methodology. The Systems Programming Series. Reading, MA: Addison-Wesley Publishing Company.

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.

Mills, Terry L. 1991. Capacity Planning For Managers: an Implementation Architecture. CMG Transactions: Computer Measurement Group.

Norton, Tim R. 1996. Simalytic Enterprise Modeling: The Best of Both Worlds. CMG Proceedings,: 1-12: Computer Measurement Group.

Pooch, Udo W. and James A. Wall. 1993. Discrete Event Simulation: A Practical Approach. Edited by Udo W. Pooch. Computer Engineering Series. Baca Raton: CRC Press.

Pooley, Rob. 1995. Performance Analysis Tools in Europe. Informationstechnik und Technische Informatik 37 : 10-16.

Rosenberg, Jerry L. and Ellen M. Friedman. 1984. Capacity Planning in a Decentralized Environment. CMG Proceedings,: 422-424: Computer Measurement Group.

Smith, Connie U. 1995. The Evolution of Performance Analysis Tools. Informationstechnik und Technische Informatik 37 : 17-20.

Wicks, Ray. 1989. Balanced Systems. CMG Transactions: Computer Measurement Group.

Wilson, Gregory L. 1994. Capacity planning in a high-growth organization. CMG Proceedings. Orlando, FL: Computer Measurement Group.