Functional Requirements for Client/Server Performance Modeling: An Implementation using Discrete Event Simulation

Joseph J. Martinka

Hewlett Packard Laboratories
Hewlett-Packard Company, Palo Alto, CA 94304-1126

Abstract

Design, management, and capacity planning of application performance becomes more demanding as the client/server paradigm is used in commercial enterprise applications. The ability to model distributed applications is crucial, and the modeling capability has to be available to larger populations of application designers and capacity planners. This paper motivates the need for modeling capabilities for these applications, and specifies the functional requirements of performance modeling of commercial OLTP client/server applications based on Remote Procedure Calls (RPC). These requirements include the ability to model transactions consisting of multiple, nested, synchronous and concurrent RPC's, as well as techniques to make such model solutions tractable. We show how we employed modeling techniques using discrete event simulation. Several examples of distributed application models are presented as results of our research to validate our approach and evaluate the modeling's tractability for typical problems. Suggestions for future work concludes the paper.

1. Introduction

Designing and building distributed applications is a risky assignment without a modeling environment. The techniques used in the past to design and manage monolithic applications are inadequate in the new wave of open distributed systems. A system model is no longer optional. This paper highlights a critical need for performance modeling in the area of client/server application design and management. It defines the functionality required in these models.

After discussing the motivation for simulation models and related research in section 2, a compilation of client/server model requirements follows in section 3. Section 4 describes our modelling approach and technology. Validations of our prototype model using RPC instrumentation is summarized in section 5, while in section 6, an example of a medium-sized modelling experiment to demonstrate scalability is shown. Section 7 concludes with observations for future work.

2. Motivation

The use of system modeling tools is essential to understand the trade-offs in application design, capacity configuration, and scalability of a client/server application. Modeling is the most cost-efficient way to project the performance of an new application beyond the geographical limits of a test network. Failing to understand the effects of operational network latencies and compute times on the application before deployment risks expensive failures or delays of a poorly-designed conversion. Performance engineering in all phases of an application's life cycle is crucial for lower-risk, cost-effective solutions [23].

Traditionally, the **computer system vendor** used modeling expertise to design, test, evaluate and produce computer boxes as an integrated system. These computers were developed around a design center based on "typical" monolithic customer applications. These efforts focused on the need for the hardware design to balance the customer's typical application use of the system's internal compute resources and communication paths (e.g. system buses, caches, and I/O channels). Sophisticated simulations and models were usually necessary to create price and performance competitive systems to solve customers' needs.

The customer then designed and built an application residing on a single compute node. Rules of thumb or straight-forward node analysis based on prototype tests could often project the system size needed for release to production. The processing power of the computer and number of disk drives were the primary hardware capacity planning decisions. These methods often carried over into management of these applications.

The advent of client/server applications has changed the situation from the scenario above. The overall system design role that computer vendors played in the past is no longer solely the vendor's responsibility. In the brave, new world of distributed computing, the distributed application designer is compelled to enter the world of system design and configuration. The designer who builds a distributed application faces many of the same design issues that vendors confront when designing a computer node. This analysis is conducted in a larger context, and without the tools or experience that computer vendors have accumulated.

The analogy is striking. The enterprise backbone or WAN network of a distributed system functions for the application as the single computer node's back-plane memory bus does for a node. The application's use of LAN networks are analogous to the single computer node's I/O channels. The application design decisions about where to partition and locate the compute and the storage requirements of the application, the use of caching, and the latency of network communications have their analogy in the memory controller, I/O controller, and memory bus design of a computer node. Today, these computer nodes in a distributed system are merely part of the solution, a collection of resources which the customer intends to harness together into a larger system. The enterprise risks disaster when configuring balanced system solutions without a modeling methodology addressing the design and management issues inherent in such a task.

Commercial modeling environments for client/server applications are not ready for general use. Thus, there is an unmet need to support large-scale client/server applications. However, without modeling, designing effective applications and quantifying tradeoffs in the design is akin more to astrology than engineering.

2.1 Related Research

Advocates of performance modeling early in the design cycle support the notion of decompositional techniques, especially applicable with the increasing role that middleware components play in newer applications. Vetland [22] generates a multi-level representation of an application based on its (reusable) devolved constituent demands on a system resource. A series of operations using linking complexity matrices yields resource parameterizations for a Stochastic Petri Net (SPN) or other model. Hillston's work [10] creates a compositional approach to performance modeling based on a stochastic algebra called PEPA. A series of equivalence relations using weak isomorphism model simplifications yields more tractable solutions of the underlying Markov model state space. Libraries of components in either effort can make performance analysis accessible to the non-expert.

Andleigh and Gretzinger [1] address how abstraction models can be used to structure a distributed database design. A hierarchial series of design models is described comprising a system design methodology. This technique requires evaluation of design choices through transaction, structural and behavioral models. For cost effective implementations, these evaluations should be completed before coding takes place, and simulations of the application are indispensable in the technology assessment phase. The simulation evaluates choices in the behavioral and structural models, a part of their overall object model. Franken and Haverkort [6] describe a performance model-based "Performability Manager" that uses a SPN solution to analyze the performance components of a QoS specification in a distributed environment, guaranteeing user-requested QoS, including reliability. They show how it can be used in an ANSAware-based environment for performability management, but recognize the complexity involved in the mapping of SPN components to alternative configurations.

A simulation approach to understand distributed database scheduling deadlines (maximum desired response times) was presented in [21] but lacked the client/server functionality described here. Its parameterization required a (too) detailed understanding of the application data locality not available to us in the more general case. Several efforts to characterize client-server distributed applications using SPN's have been reported [6][11][12][22] with the attendant state explosion problem which limits model scalability. Petriu describes an approximate Mean Value Analysis (MVA) approach called Stochastic Rendezvous Networks which has better solution times than SPN's [16]. This impressive work is extended to multi-threaded clients more recently in [17] but still assumes exponential client and server service times. The results in the paper were evaluating abstraction and analytic assumptions comparing to discrete and SPN simulations, but using the same service time distribution assumptions, not experimental results from actual systems.

Many researchers and practitioners have recognized the critical need for modeling and instrumentation integration[11][13][20]. Others note the current lack of these capabilities in distributed client/server systems [4][18][22]. The instrumentation in distributed systems to support these models and tools is in its infancy or not available [22].

3. Client-Server Model Requirements

Several factors compound the performance challenge when modeling a distributed application versus a monolithic application. Foremost is the injection of LAN or WAN network latencies into the application delay paths, which directly affects the user-perceived Quality of Service (QoS).

Secondly, system boundaries for the model extend past the conventional analysis for a single server queueing model where the focus is node-based performance [14]. For the solution to be tractable, the performance model of a distributed system must raise its abstraction level, limited only by the set of design questions of the model [4]. This abstraction level is higher than many prevalent modeling tools, techniques and parameterizations.

3.1 Design Questions for Modeling

Design questions of a new or existing distributed application can be addressed by a general modeling approach to distributed systems. Some of the questions befuddle

today's designers of even simple distributed systems. In general: What design parameters affect good performance (i.e. achieve the service agreement or response time targets) for this distributed application? In particular, some critical design questions are:

- How does geographical dispersion of compute nodes or dataset locations, and the attendant network latencies affect response times and throughput?
- How does the speed or capacity of any of the participating node's CPU have on response time and throughput?
- What are the software bottlenecks due to congestion or RPC nesting?
- How do interfaces and dependencies on legacy applications (and their behavior) impact the design and performance of the distributed application?
- What is the effect of adding other applications or users to the environment of the modeled application?
- What is the cost for different levels of security: authentication, authorization, data integrity, or encryption?
- What is the effect of using distributed or replicated services? (e.g., database, binding, security)
- What is the effect of upgrading low bandwidth networks on response times and throughputs?
- How susceptible is performance to network utilization, bandwidth, packet loss, congestion, and distance?
- How does asynchronous overhead related to replicas and distributed update algorithms increase with workload, and impact scalability of the application?

3.2 Functional Specifications for Modelling

The framework presented in [6] will be used to organize the C/S applications performance model's functionality requirements that a successful model allows.

Task specification

- identifies user-level tasks that need to complete to accomplish useful work
- determines the rate at which these tasks are introduced
- permits stochastically generated arrival rate of tasks to the system, from specific nodes or a set of nodes
- identifies how applications are invoked, distinguished by sequential or concurrent order to complete a task.

Application specification

- supports RPC's with deterministic (constant) or with a specified load-dependent service times
- contains one or more concurrent threads of execution which can generate RPC's to distributed nodes or other processes,
- generate sets of RPC's which execute concurrently. These sets can in turn be executed serially (the next set starts only after the slowest RPC of the previous set has completed) or concurrently with other sets.
- capability to send an RPC to the same node used by the client (local RPC). RPC's that are local to the compute node do not access the network.
- allow RPC's to invoke another server. This **nesting** of RPC's within applications is often a critical part of most applications, creating software congestion.
- multi-class workloads, transactions and applications must be supported in the same model
- allow transactions which create asynchronous RPC's or other applications. This requirement is due to the implications of replication and record-keeping. Overhead work may be asynchronously completed (some messaging paradigms can be modeled with this requirement). Overhead is a part of a workload which determines scalability of an application.

System Node Specification

- specifies the instruction processing speed of the node,
- allows a capability for multi-processors at a compute node including the specification of non-linear scalability. This permits lightly loaded MP's to execute threads at uni-processor speeds, but degrade more quickly as the load builds.
- memory utilization shows characteristics such as working set size and disk statistics to indicate device utilizations and delays
- CPU degradation from other non-modeled applications is accommodated in resource contention delay

Network Specification

- provides a workable performance abstraction for various network types which accommodate a parameterized delay based on distance between notes for speed-of-light based LAN, MAN and WAN delays
- provides a capability to model multi-packet transfer size buffers and an abstraction of multi-packet windowing flow control in the network delay model
- model protocols applicable to legacy applications (peer to peer and messaging protocols)
- retry and sanity check operations for long-lived RPC's.

3.3 Simulation Environment for the Tool

The mechanics of performing a series of simulation runs should be automated to the extent possible by the tool for scenario and sensitivity exploration of the experimental space. The tool should offer built-in assistance to the modeler in making and understanding experimental design of simulation runs [2][3]. Minimally, eliminate manual reconstructing and recompiling of the model for each run for GUI-based graphic models. The simulation engine should sense and terminate a run on monotonically increasing queue lengths due to resource saturation as well as automatic detection of specified confidence interval of key simulation statistics [13]. Incorporate hybrid modeling techniques in larger models [17] (e.g., building MVA elements which extend the other sub-models), or with modeled or real distributed agents contributing to a hierarchial solution [9].

4. Modeling Methodology

Based on the complexity of distributed systems and the uncertainty in treating them analytically, we opted to begin this research using a general discrete simulation engine. We used SES/Workbench from Scientific and Engineering

Software (SES), Inc. [19]. Our prototype model described in this paper is based on 1993 source from SES's C/S Composer [5], but heavily modified to meet most of the functional requirements described in Section 3. The actual enhancements made to the source and model creation details are in [15] and briefly described here. These enhancements focused primarily on the requirements for flexible topology specifications, nested and asynchronous RPC's, automatic model termination, and a network abstraction with realistic latencies.

The simulation modeling software is based on a resource-centered event which has favorable simulation run-time advantages compared to an RPC-centered event principally since there are fewer queued events in the latter. In addition, this choice lends significant flexibility to the specification of the model itself. The model binaries need not be recompiled when changing the modeling topology or parameters. A large variety of distributed applications and topologies including all those described in this paper are specified using ASCII input files. This method allows the modeler to easily specify the number and type of compute nodes, application transactions, users, networks, routers and routing technology. It also activates a variable degree of statistical reporting functions based on the modelers needs. Scripts were used to create this input file, or a GUI to simplify interactive modifications of the model through this configuration file.

(editor's note: this section can be expanded if reviewers identify cuts elsewhere.)

5. Model Validation Experiments

Our group had a contemporaneous effort with other industry partners to specify a heterogenous distributed instrumentation architecture integrated into the Distributed Computing Environment (DCE) [7]. A client/server performance metric collection system [8] supported the modelling experiments described below.

5.1 Phone Database Application

The sample DCE application called *Phone DB* supplied with HP's DCE product is a client/server in-memory database application implementing many of the features of the DCE infrastructure. We modeled *Phone DB* as a multiclass client/server application running two kinds of transactions characterized by their server's CPU consumption. Parameterization of the model was made using response times and RPC counts provided by the prototype RPC instrumentation. Data was collected for each class of transaction in isolation at a low throughput rate (to minimize queueing and contention) so that response time could be substituted as CPU service times.

In Figure 1, the average of the two classes of transaction response time is plotted with increasing load showing that

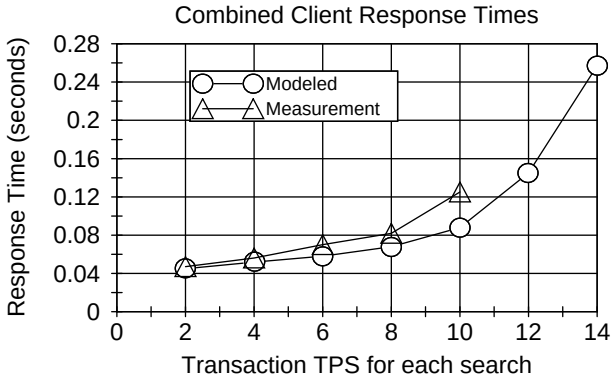

Figure 1 Phone Database Model Validation

the modeled results corresponds within 10% of the instrumented application below the knee of the curve which starts at 10 TPS.

5.2 OLTP Monitor Application

In a more challenging environment, we modeled an OLTP transaction using Encina, a DCE-based middleware product offering transactional semantics. Encina ships with a sample application called *telshop* which was instrumented in a similar fashion as the *Phone DB* described earlier. We discovered its behavior using event-tracing[1], and modeled it using simulation. In Figure 2, we plot the modeled and measured results of a transaction consisting of five read queries to an inventory database for a range of workload levels. The agreement with the measured results was satisfactory.

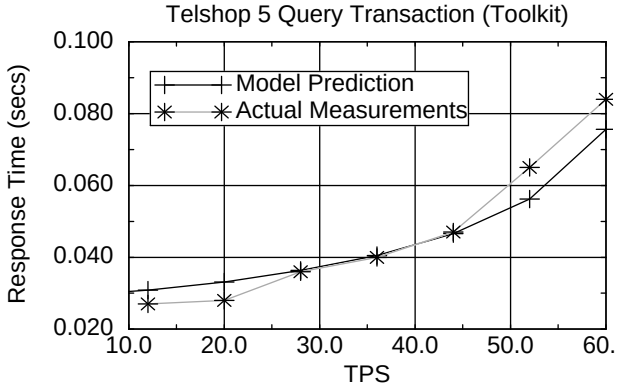

Figure 2 Encina's Telshop Read Task

Simulation of update transactions shown in Figure 3, the initial model did not validate the measurements of a batch of entry (update) transactions at higher transaction rates. Investigation revealed that the Encina log server process which ensures the durability of the transaction was implemented with a load dependent service time. At a threshold arrival rate, RPC's are held for 300 ms in the expectation that some logging requests can be combined, thereby con-

1. Event tracing at the RPC interface is provided by the instrumented IDL compiler (I2DL) in HP's DCE product offering.

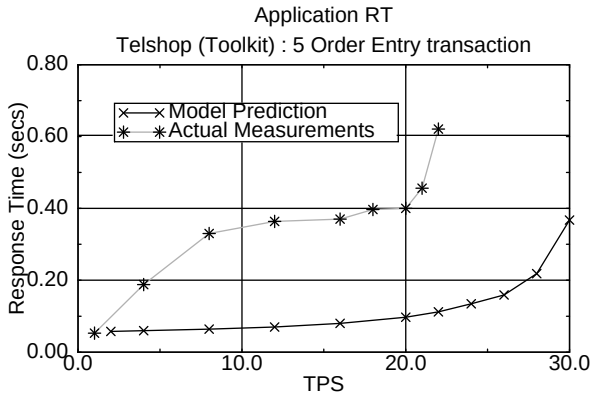

Figure 3 Encina's Telshop Update Task

solidating disk I/O's and amortizing the CPU time. This experience brought us face to face with a probable behavior in tuned servers: load-dependant service times. Once the simulation was changed to model this behavior, the results yielded a satisfactory match compared to measured response times (omitted due to space limitations). The lesson is that models must be validated with real systems, not simply with other models to uncover abstraction pitfalls such as this.

6. Model Scalability Experiment

An experiment was conducted to exercise most of the modeling functionality discussed in section 3. Our goals were to demonstrate the capabilities of the simulation for sensitivity analysis in the application design phase, and to show how nested, asynchronous RPC's are essential to be integrated into the model functionality.

6.1 R.P.C. Corporation System Description

A hypothetical corporation's distributed application is used as an example to illustrate some distributed application design tradeoffs. Raincoat Ponchos Company (R.P.C.) contemplates the design of a distributed Order Processing system. This system is being designed as a client-server distributed system across the enterprise.

R.P.C. is an international enterprise which has a corporate headquarters and four regional offices, each with five branch offices. User workstations used for sales activities in every branch office are connected through LANs to the branch office computer. The legacy corporate information data center is in New York City. New York, Dallas and San Francisco comprise the three domestic sales regions. Each region has several branch offices, one co-located, the others at more remote locations. The international regional sales office is in Singapore, with its branches in Hong Kong and Japan.

Later sections discuss five representative branch offices. While each branch is executing a similar transaction load, they have dissimilar performance characteristics due to network latencies. These branches are bold ovals in Figure 4.

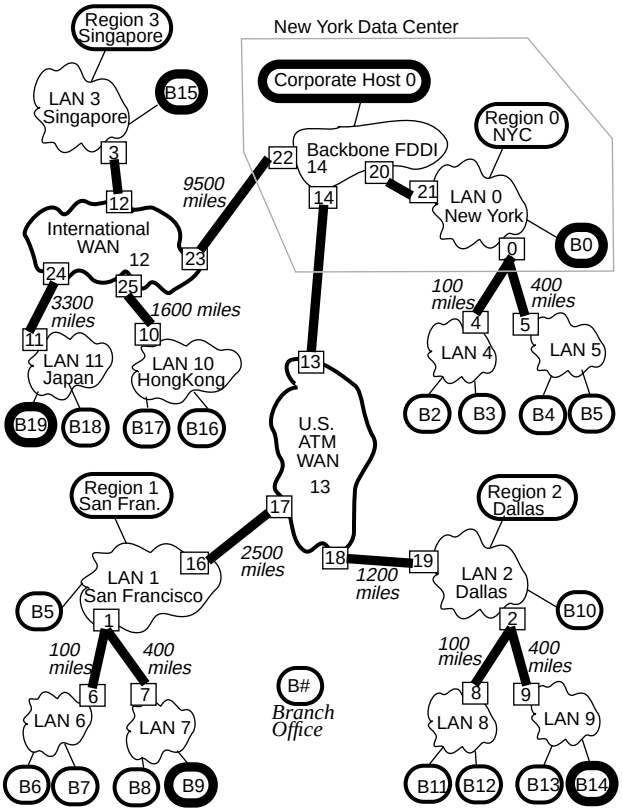

Figure 4 R.P.C. Network/Node Topology

Network Topology

We depict R.P.C.'s computing and networking topology in Figure 4. The network is modeled only to a level of detail which exposes the principal performance concerns of the modeling goals. If the distance between networks is significant, it is expressed in miles on the dark thick line connecting routers. A FDDI fiber backbone connects the corporate server with other corporation computers and the networking ports. The U.S. domestic locations are connected internally using LANs (ethernet), and are inter-connected via Asynchronous Transfer Mode (ATM) public networks. The international nodes are inter-connected by slower WAN technology (e.g., X.25).

When a router is handling a packet to another router with a distance specification, a delay is imposed which relates to the distance traveled. The modeled network throughput for the different varieties of network types used by R.P.C. is listed in the following table. Modeled queuing disciplines are included.

A routing map was built for the model which describes the network route that messages from each branch must traverse to communicate with all regional compute centers,

as well as with the corporate computer center.

Network type	BW in MB/sec	Modeled queueing discipline
ethernet LAN	1.25	resource sharing
domestic ATM	18.7	first come first served
local FDDI LAN	12.5	first come first served
international WAN	0.0009	infinite server

Computer node assumptions

The simulation model accommodates the probability that compute nodes in distributed systems such as R.P.C. will have multiprocessors. The compute power for each computing node assumed in this model is shown below:

location	node CPU MIPS	MP count
corporate	90	4
regional	40	2
branch	30	1

The computer processing costs for operating on these datasets are measured or estimated in CPU instructions. Disparate compute nodes with an known MIPS rating which act as clients and servers can be substituted in the model to determine capacity and scalability.

6.2 Application Workload Characteristics

The Order Processing application uses datasets which are shared and sometimes replicated. These datasets are planned to reside at selected compute nodes in the company. They are implemented in database technologies using a common transaction management interface (e.g., X/Open's XA). These datasets for the application are listed below and represent a first-cut partitioning and location of data. In the initial design, there are four distinct datasets, of which three are replicated.

Dataset Description	Replica?	Initial Location
Product description & Inventory		corp
Branch & Sales Rep totals	yes	corp
Order Tracking		region
Customer Info & Accounts	yes	region
Product Description &Inventory	yes	region
Customer Info & Accounts		branch
Branch & Sales Rep totals		branch

Transaction descriptions and dataset operations

A two-class workload was constructed where two transaction types were modeled: **Order View** and **Order Entry**. Other transactions such as ship order and cancel order are considered derivatives of the above transactions.

Order View transaction is a read-only transaction. The dataset operations are:

1. Read customer account information
2. Read product information
3. Read order tracking information

Order Entry is the update transaction. The required dataset operations are:

1. Read customer account information
2. Read product information
3. Insert order in Tracking database
4. Enter order in customer account
5. Debit customer account
6. Enter order in the branch and sales rep totals
7. Update the branch and sales rep totals replicas
8. Update the Product information
9. Update the Product information replica

Some of these operations that involve a read and a write to the same dataset are considered to be part of a single data packet exchange to a remote system in this model. All updates to these datasets are controlled by a two-phase commit protocol by participating nodes.

The network traffic for each of the datasets assumes that any data to be transferred as part of the RPC can be carried within a single 1500 byte Internet Protocol (IP) packet.

Workload constraints

Transactions originate at individual desktop workstations, however, the modelled load is initiated at the branch computer. It is assumed that the workstation transmission times have known and negligible delays to the branch node.

The branch office is the focus of the most interesting order processing transaction behavior. It is assumed that the average rate of **Order View** and **Order Entry** transactions will be equal, although other load mixes are possible. Similarly, branch offices present equal transaction demands to the system.

Each branch office computer will generate both types of transactions with an exponentially distributed inter-arrival time. While no bottlenecks are present, the throughput for each branch will be roughly equivalent. The response times for each transaction will serve to differentiate one branch from the other and provide a figure of merit for database partitioning decisions.

6.3 Model Results

We present some model results of the initial design choices for R.P.C.'s application in this section.

Configuration Application Performance

The simulation model was run for a range of branch

office transaction rates from 2 to 18 transactions per second (TPS), the higher level is beyond saturation of some system resource[1]. Transactions consisted of similar rates of the two transaction types. The transactions rates are expressed in terms of the number of transactions originating at any one particular branch office. This load growth could be generated by more client workstations or by increased traffic from existing workstations.

The selected five branch offices of R.P.C. Corp. to provide an interesting spectrum of network delays. In Figure 5, the response times for these branches are plotted. Be

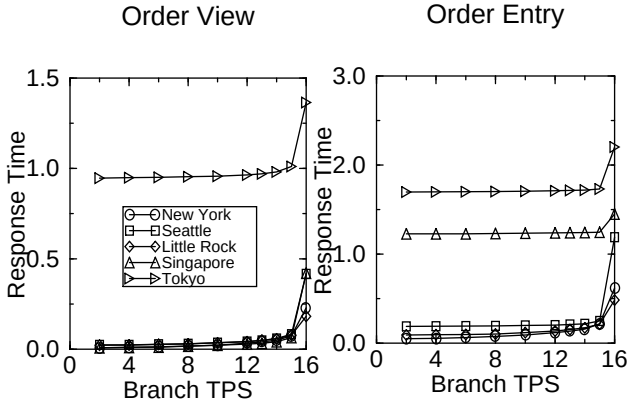

Figure 5 Selected Branch Response Times

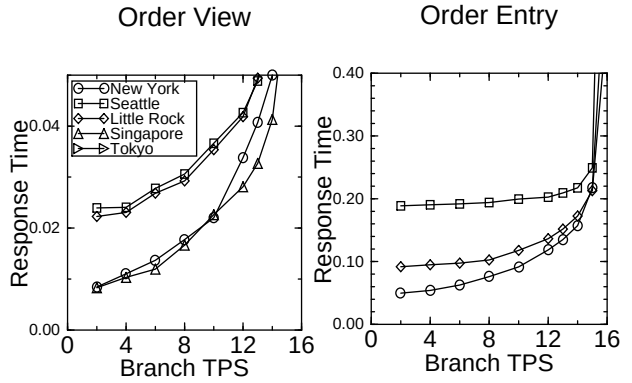

Figure 6 Expanded Branch Response Times

careful to note the response time axis differences between the two transactions in all of these graphs.

The Order Viewing transaction shows that four out of the five branch offices presented have under 30 ms response times. Recall that for a **Order View** transaction, only the regional CPU and the branch CPU participate in the transaction. Thus the Tokyo region has response times of nearly one second since it is the farthest from its regional office in Singapore.

1. 20 branch offices generate 40 to 360 TPS. Do not confuse this TPS to a different fruit from the Transaction Processing Council (TPC)

The **Order Entry** transaction shows the effect of including RPC's to the corporate CPU. Not only are the response times a great deal larger, now the Singapore branch reflects the additional network delay experienced by communications to the corporate CPU. The individual branch behaviors in Figure 6 show the detail of branches with lower response times. Branch offices in cities remote from their regional office have response times which are twice as long as the branch offices which are in located the same city as their region.

In the **Order View** transactions the branches that are further away from their regional CPU (Little Rock, Seattle) have an extra 15 millisecond delay than the branch office CPUs co-located with their regional CPU (Singapore and New York). These response time relationships increase by an order of magnitude for the **Order Entry** transaction since Singapore has a longer delay to get to the corporate CPU in New York, than does the Seattle office.

Compute Bottleneck

The "knee" of the response time curve is often due to a resource bottleneck. In this scenario, the regional CPU node becomes saturated when each of its five branch offices are generating 15 to 16 TPS. In Figure 7 we plot the modeled CPU utilizations for the branch, region and the corporate CPU nodes.

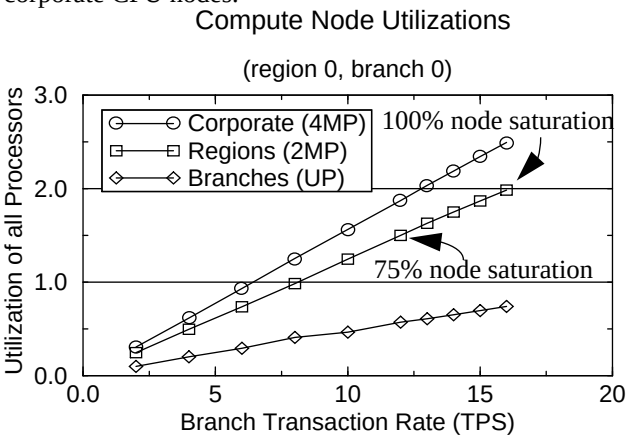

Figure 7 CPU Utilizations

Recall from section 6.1 that the regional and corporate CPUs were multi-processor (MP) machines, whereas the branch CPU was a uni-processor (UP). The two-way regional CPU reaches saturation above 15 TPS and is the primary reason for the knee in the response time curves of Figure 5 and Figure 6. The branch CPU is not too far behind however. At nearly 75% utilization of the regional CPU, it is beginning to contribute to the large queueing delays of the transactions at 12 TPS per branch. The corporate CPU is sized so that it is 40% idle, or available for other application activities.

6.4 Model Runtime Costs

The simulation runs were driven by simple script programs. Several model scenarios at different workload factors could be started and run to conclusion during off-hours.

We were somewhat pleased in the efficiency of the model to generate a series of results enabling design tradeoffs and sensitivity analysis. This model was run on a 99 MHz HP Series 735 workstation under HP-UX 9.03. For every TPS in the graphs shown earlier, there is one order and one entry transaction. This model has 16 RPC's per TPS. The simulation runs in batches to calculate and terminate when a 90% confidence interval for user response times is reached. Simulation run time performance is listed below includes a warm-up batch.

TPS	Transactions	RPC's	CPU time	Sim time
4	16,100	257,000	6 min	60 sec
8	32,000	512,000	16.3 min	60 sec
12	47,000	752,000	40 min	110 sec

The CPU cost was 1.4 to 3.2 msec per simulated RPC (including network and both client/server compute nodes), a range which is satisfactory for design analysis. To succeed at our eventual goals of adaptive distributed systems during operations, heuristic or analytical techniques will be necessary.

6.5 Other Model Scenarios

Alternate design scenarios were modeled to demonstrate flexibility and provide quantitative alternatives for design options:

- updating of all replica datasets is made asynchronous to the update transaction (occurs in the background).
- alternate placement of datasets where some datasets are moved to the branch node, and new replicas are created. The **Order View** transaction can complete with local procedure calls, but the **Order Entry** transaction becomes more complex.
- updating of all replica datasets is made asynchronous to the transaction.
- the instructions and network traffic to update replicas are considered as a third, asynchronous transaction which proceeds at the same rate as the **Order Entry** transaction but not necessarily completing as the Entry transaction finishes.

7. Conclusions

Modeling of distributed client/server applications is a critical factor in their design, deployment, and later scalability. The modeling technologies needed in this effort are not generally available, and not ready for broad distribution to application designers and planners. This paper highlights the functionality needs for client/server models and describes design questions to be addressed. A prototype simulation model implemented many of the requirements listed, and its use was demonstrated in several real and hypothetical examples.

There is much research and practical work to do in the user interface to the modeling engine, methodology to decompose the activities of a middle-ware dependant application, automatic model parameterization, and further validation of a simulation approach to modeling client/server applications. Strides are needed not only in the models which meet many of the specifications described in this paper, but also in the modeling methodology and integration with distributed instrumentation as it makes its appearance in the middle-ware infrastructure.

Acknowledgments

I am indebted to my colleagues Rich Friedrich, Steve Saunders and Tracy Sienknecht for their support.

References

[1] Prabhat Andleigh and Michael Gretzinger, *Distributed Object Oriented Data Systems Design*, Prentice Hall, Englewood Cliffs, NJ, 1993?.

[2] Robert Berry, *Experimental Design and Computer Performance Analysis*, Proc. of Computer Measurement Group (CMG) '92, Reno, Nevada, 1992, pp 1100-1110.

[3] George E.P. Box, William G. Hunter, J. Stuart Hunter, *Statistics for Experimenters*, John Wiley & Sons, New York, 1978.

[4] Peter Dauphin, et al., *ZM4/Simple: A General Approach to Performance Measurement and Evaluation of Distributed Systems*, Readings in Distributed Computing Systems, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp 288-309.

[5] *C/S Composer User's Guide Release 1.1*, Scientific and Engineering Software, Inc., Austin Texas, 1994.

[6] Leonard Franken and Boudewign Haverkort, *The Performability Manager*, IEEE Network, Jan/Feb 1994, pp 24-32.

[7] Richard Friedrich, *The Requirements for the Performance Instrumentation of the DCE RPC and CDS Services*, Open Software Foundation DCE Request for Comment (OSF DCE-RFC 32.0), June 1993.

[8] Richard Friedrich, Joe Martinka, Tracy Sienknecht and Steve Saunders, *Integration of Performance Measurement and Modeling for Open Distributed Processing*, Proceedings of the International Conference on Open Distributed Processing (ICODP '95), February 1995 , pp 341-352.

[9] D. Gaïti, *Intelligent Distributed Systems: New Trends*, Proceedings of the 4th Workshop on Future Trends of Distributed Computing Systems, Lisbon, Portugal, Sept 1993, pp 106-111.

[10] Jane Hillston, *A Compositional Approach to Performance Modeling*, PhD Thesis, University of Edinburgh, 1984.

[11] Peter Hughes and Dominique Potier, *The Integrated Modelling Support Environment*, Esprit 89 Conference Proceedings, Document IMSE R-1.2-4, STC plc and Thomson CSF, 1989.

[12] Oliver Ibe, Hoon Choi, and Kishor Trivedi, *Performance Evaluation of Client Server Systems*, IEEE Transactions on

Parallel and Distributed Systems, Nov 1993, pp 1217-1229.

[13] Raj Jain, *The Art of Computer Systems Performance Analysis*, John Wiley, 1991.

[14] Edward Lazowska, et al. *Quantitative System Performance - Computer System Analysis Using Network Models*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

[15] Joseph Martinka, *A Performance Model of a Client-Server OLTP System Using SES/Workbench*, Proceedings of the Fourth Annual SES Users Group, SES, Inc., Austin, Texas, April 1994.

[16] D.C. Petriu and C.M. Woodside, *Approximate MVA from Markov Model of Software Client/Server Systems*, Proceedings of 3rd IEEE Symposium on Parallel and Distributed Processing, pp. 322-329, 1991.

[17] Dorina C. Petriu, et al, *Analytic Performance Estimation of Client-Server Systems with Multi-Threaded Clients*, MASCOTS '94: Modeling, Analysis, and Simulation Int'l Workshop, pp 96-100.

[18] Jerome A. Rolia, *Distributed Application Performance Metrics and Management*, Proc. from 2nd International Conference on Open Distributed Processing '93, Elsevier Science B.V. (North Holland), pp 235-246?.

[19] *SES/workbench Reference Manual*, Scientific and Engineering Software, Inc., Austin, Texas.

[20] Connie Smith, *Performance Engineering of Software Systems*, Addison-Wesley, 1990.

[21] Özgür Ulusoy, Geneva Belford, *A Simulation Model for Distributed Real-Time Database Systems*, Proc. of 25th Annual Simulation Symposium, Orlando, Florida, April 6-9, 1992, pp 232-240.

[22] Vidar Vetland, *Measurement-Based Composite Computational Work Modelling of Software*, PhD Thesis, Norwegian Institute of Technology, The University of Trondhem, 1993.

[23] Eyal Zimran, David Butchart, *Performance Engineering Through the Product Life Cycle*, COMPEURO '93 Computers in Design, Manufacturing, and Production, pp 344-349.