Requirements for Client/Server Performance Modeling

Joseph J. Martinka

Hewlett-Packard Laboratories
Palo Alto, California
martinka@hpl.hp.com

Abstract

Design, performance management, and capacity planning of client/server applications in the commercial enterprise depends on the ability to model these distributed applications at design time as well as during normal operations. This paper specifies the functional requirements of performance modeling of the class of applications based on Remote Procedure Calls (RPC). These specifications include the need to model transactions consisting of multiple, nested, synchronous and concurrent RPC's using tractable techniques. Examples of distributed application models are presented based on discrete event simulations. Suggestions for future work are included.

1. Introduction

Designing and operating 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 era 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.

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.

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 [11] 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 [4] creates a compositional approach to performance modeling based on a stochastic process algebra. Franken and Haverkort [2] 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. There is other work as well with similar goals.

A simulation approach to understand distributed database scheduling deadlines (maximum desired response times) was presented in [10] but lacked the client/server generality described here. Petriu describes an approximate Mean Value Analysis (MVA) approach called Stochastic Rendezvous Networks which has better solution times than SPN's. It was extended to multi-threaded clients [7], but still assumes exponential client and server service times.

Many researchers and practitioners have recognized the critical need for modeling and instrumentation integration. The instrumentation in distributed systems to support these models and tools is in its infancy or not available. 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.

2. Client-Server Model Requirements

Several factors compound the performance challenge when modeling a distributed application over 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. 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. This abstraction level is higher than many prevalent modeling tools, techniques and parameterizations.

2.1 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?

Functional Specifications for Modelling

The framework presented in [2] 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

- provide 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
- provide 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.

2.2 Simulation Environment for the Tool

The mechanics of performing a series of simulation runs should be automated to the extent possible by the tool. This includes assistance to the modeler in making and understanding experimental design of simulation runs for scenario and sensitivity exploration of the experimental space[1]. Minimally, the environment should eliminate manual reconstructing and recompiling of the model for each run for GUI-specified models. The simulation engine should sense and terminate a run when a problem like monotonically increasing queue lengths is detected due to resource saturation, similar to termination on automatic detection of specified confidence interval of key simulation statistics. Incorporate hybrid modeling techniques in larger models [7] (e.g., building MVA elements which extend the other submodels), or with modeled or real distributed agents contributing to a hierarchial solution.

3. Modeling Experiments

Based on the complexity of distributed systems and the uncertainty in treating them analytically, we began this research using a general discrete simulation engine. The remaining sections of this paper are brief, but are presented more fully in [6], the extended version of this paper.

A client/server performance metric collection system [3] destined as an OSF standard for DCE supported the modelling experiments. **W**e modeled an OLTP transaction using Encina, a DCE-based middleware product offering transactional semantics. Encina ships with a sample application called *telshop* which we instrumented. We discovered its behavior using event-tracing, and modeled it using a simulation engine SES/Workbench from Scientific and Engineering Software (SES), Inc. [9]. A detailed description of the simulation source and model creation are found in [5]. In Figure 1, 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.

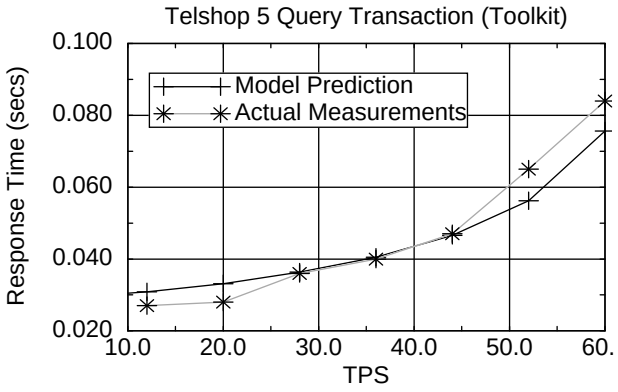

Figure 1 Encina's Telshop Read Task

4. Model Scalability Experiment

An modeling experiment was conducted to exercise most of the modeling functionality discussed in section 3. Our goals were, in part, 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.

A hypothetical corporation's distributed application is used as an example to illustrate some distributed application design tradeoffs. It is an international enterprise with a corporate headquarters and four regional offices. Each region has 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. New York, Dallas and San Francisco comprise the three U.S. 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.

The WAN and LAN network is modeled only to a level of detail which exposes the principal performance concerns of the modeling goals. 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.

The computer processing costs for operating on these datasets are measured or estimated in CPU instructions. Disparate compute nodes with known MIPS ratings acting as clients and servers are used in the model to determine capacity and scalability.

The Order Processing application uses seven datasets which are shared and sometimes replicated. 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

A two-class workload was constructed where two transaction types were modeled. All updates to these datasets are controlled by a two-phase commit protocol by participating nodes. The branch office is the focus of the most interesting order processing transaction behavior. 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.

4.1 Model Results

We present an example of simulation results of the initial design choices for the application. 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. 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 branch office.

The response times for five branch offices are shown in Figure 2. The Order Viewing transaction shows that four out of the five branch offices presented have under 30 ms response times. In an **Order View** transaction, only the

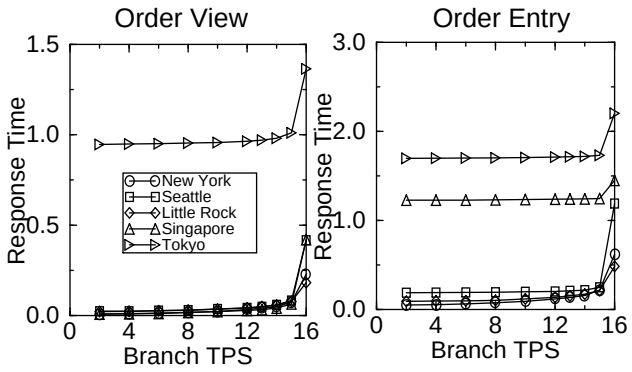

Figure 2 Selected Branch Response Times

regional CPU and the branch CPU participate in the transaction. The Tokyo region has response times of one second since it is the farthest from its regional office in Singapore.

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. 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.

The model yields utilizations of networks and computers, queueing for software bottlenecks and other concurrency estimates. Alternate design scenarios were modeled to demonstrate flexibility and provide quantitative alternatives for design options.

4.2 Model Runtime Costs

The simulation runs were driven by simple script programs. This model was run on a 99 MHz HP Series 735 workstation under HP-UX 9.03. 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.

5. 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. Advances 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.

References

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

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

[3] 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.

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

[5] 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.

[6] Joseph Martinka, *Functional Requirements for Client/Server Performance Modeling: An Implementation using Discrete Event Simulation*, HP Laboratories Technical Report, HPL-95-92, July 1995.

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

[8] 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.

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

[10] 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.

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