Performance Prediction of Client-Server Systems by high-level Abstraction Models

Alexander Pastsyak
Author Affiliation line 1
Author Affiliation line 2
email:
Alexander.Pastsyak@siemens.com

Yana Rebrova
Author Affiliation line 1
Author Affiliation line 2
email:
Yana.Rebrova@siemens.com

Vladimir Okulevich
Author Affiliation line 1
Author Affiliation line 2
email:
Vladimir.Okulevich@siemens.com

Abstract

Performance modeling and simulation offer powerful approaches to predict performance characteristics of software applications and in particular web applications. At the same time the usage of modeling techniques in real projects is still limited due to comprehensive structure of applications and impossibility to know internal structure of application's components needed to create a precise model. In our work we build and evaluate the high level abstraction models of Content Management System "Joomla" to estimate their ability to provide reliable predictions for different performance characteristics. To build the models we consider three formalisms: Layered Queuing Networks, Performance Evaluation Process Algebra and Queuing Petri Nets. Predictions obtained from models simulation have been compared with measurements of the real system by HP Load Runner toolkit. Results show that overall throughput of the system can be predicted with high level of accuracy while prediction of response time by the same models is less reliable.

Keywords: *Software performance simulation, performance modeling, web based systems*

1. Introduction

Performance simulation of client-server systems has been a topic of interest for software engineering for a long time. A lot of formalisms and supporting methods have been developed to describe this kind of systems and obtain performance characteristics from related models. Most important of them are: Queuing Networks, Stochastic Process Algebras and Stochastic Petri Nets. All of these techniques have been applied to client-server applications and in particular to web applications, that showed their powerful ability to predict system behavior from the performance point of view. However the software modeling still has not become a daily practice in product life-cycle due to the complexity of model creation. It is caused by comprehensive structure of real-life applications and "black-box" nature of the application's components. To investigate model ability to predict system properties without detail information on internal structure we selected three different modern performance description formalisms to describe Content Management System "Joomla" [13]. During our study we built three different performance models of this system deployed in several configurations of distributed environment. These models have been calibrated and analyzed to obtain performance predictions, which have been compared with experimental results received by HP Load Runner package [7].

2. Related Work

Black box performance modeling approaches are most promising ones for analysis of real-life computer systems. Successful applications of such approaches for performance modeling of self-tuning controllers for web servers, storage systems and parallel applications are shown in [3, 8, 10]. These papers are focused on the techniques to understand parameter space of black-box system. On the other hand we consider simple performance models with high level of abstraction and estimate their capabilities to predict performance characteristic of real-life web application.

3. Problem Statement

The goal of our study is to analyze formalisms which allow estimating performance characteristics for the system with unknown internal structure. In particular we are interested in the following questions:

- How to map different system architecture entities to the model elements (build the models)?
- What is the method to calibrate the models?
- Are models capable to make reliable predictions for throughput characteristic?
- What kind of predictions can be obtained from the models for response time characteristic?

Our investigations of these issues are based on analysis of the experimental test system which has been modeled with help of selected formalisms.

4. Performance Modeling Formalisms

For our study we chose three formalisms by one from every group of modeling approaches: Queuing Networks (QN), Stochastic Process Algebras (SPA) and Stochastic Petri Nets (SPN). During this selection we were considering the following criteria:

- actuality: approach should be widely used for different performance investigations and being actively supported/developed;
- availability of tools: approach should be supported by easy-to-work tools.

Finally we decided to select Layered Queuing Networks (LQN) and LQNSolver tool [14] from QN. From SPA we chose Performance Evaluation Process Algebra (PEPA) and PEPA Workbench [15]. Also we opt for Queuing Petri Nets (QPN) and QPN Modeling Environment (QPME) [17] from SPN.

4.1 Layered Queuing Networks

Layered Queuing Networks (LQN) model [4, 5] is a canonical form for extended queuing networks with a layered structure. The layered structure arises from servers at one level sending requests to servers at lower levels as a consequence of request from a higher level. LQN is applied to any extended queuing network with multiple resource possession, in which multiple resources are held in a nested fashion. Resources are released in the reverse order of their acquisition and the resource order is consistent across the system. So, higher layer resources are acquired earlier and released later than in lower layers.

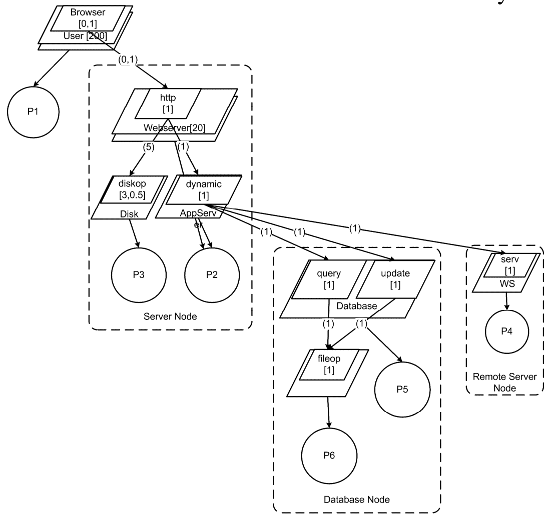

Figure 1. Example of LQN Model

Figure 1 illustrates the LQN notation with an example of a web server that has a connection to database and serves both static and dynamic pages. In LQN software resources are called tasks. The tasks have queues and provide classes of service which are called entries. In Figure 1 tasks are shown as parallelograms, containing nested parallelograms to describe entries. Processor resources are shown as circles attached to tasks which are using them. Stacked icons represent tasks or processors with multiplicity forming a multiserver. The multiserver may represent either a multi-threaded task, or a collection of identical users, or a symmetric multiprocessor with a common scheduler. Multiplicity is shown on the diagram with a label in brackets. For example, there are 20 copies of the task *WebServer* on Figure 1.

Entries have directed arcs to other entries at lower layers to represent service requests (requests may go directly through the layers). A request from one entry to another may return a reply to the original entry (a synchronous request) indicated on Figure 1 by solid arrows with closed arrowheads. For example, task *AppServer* send a request to task *Database* which then issue a request to task *FileServer*. While task *FileServer* is serving the request, tasks *Database* and *AppServer* are blocked. Alternatively a request may be forwarded to another entry for later reply, or may return no reply (an asynchronous request).

4.2 Performance Evaluation Process Algebra

In Performance Evaluation Process Algebra (PEPA) [6], system is considered as a set of components which carries out activities either individually or in cooperation with other components. Each activity is characterized by an action type α and a duration r which is exponentially distributed. This is written as a pair (α, r). Duration may be any positive real number or it may be unspecified. Activity is called shared if several components synchronize over it. The distinguished symbol **T** is used to indicate that the rate is not specified by the component. Such component is said to be passive with respect to this action type. The rate of the shared activity is defined by cooperation with another component.

PEPA provides a set of combinators which allow building expressions to define behavior of components via activities. These combinators are presented below:

- **Prefix (α, r).P:** Prefix is the basic mechanism by which behavior of components is constructed. This combinator implies that after the component has made activity (α, r) it behaves as component P;

- **Choice P1+P2:** This combinator represents a competition between components. The system may behave either as component P1 or as P2. The probability to choose one of these components is defined by the rate of their first activity;
- **Cooperation P1<L>P2:** This describes the synchronization of components P1 and P2 over the activities in the cooperation set L. The components may proceed independently with activities whose types do not belong to this set. In cooperation, the rate of a shared activity is defined as the rate of the slowest component;
- **Hiding P/L:** This component behaves like P except that any activities of types within the set L are hidden, i.e. such an activity exhibits the unknown type τ and the activity can be presented as an internal delay by the component. Such activity cannot be carried out in cooperation with any other component.
- **Constant A def = P:** Constants are components whose meaning is given by a defining statement: A def = P gives the constant A the behavior of the component P. This is how we assign names to components.

4.3 Queuing Petri Net

The main idea behind the Queuing Petri Net (QPN) modeling paradigm [1] is to add queuing and timing aspects to places of modeling formalism Colored Generalized Stochastic Petri Net (CGSPN) (subset of SPN) to provide means for direct representation of queuing disciplines. A place of CGSPN with an integrated queue is called a queuing place and consists of two components: the queue and a depository. This is depicted on Figure 2.

The behavior of the net is the following: tokens fired into a queuing place by any of its input transitions, are inserted into the queue according to the queue's scheduling strategy. After completion of its service, a token is immediately moved to the depository, where it becomes available for output transitions of the place. This type of queuing place is called *timed queuing place*. Also QPN introduces *immediate queuing places* (ordinary places) which allow pure scheduling aspects to be described. Tokens in ordinary places can be viewed as being served immediately. Scheduling in such places has higher priority than both scheduling/service in timed queuing places and firing of timed transitions. The rest of the net behaves like a normal CGSPN.

As it is shown in [2, 9] QPN has greater expressive modeling power than QN, extended QN and SPN. This formalism provides possibilities to model simultaneous resource possession, blocking synchronization and scheduling strategies.

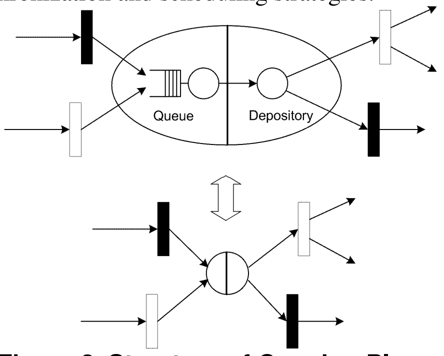

Figure 2. Structure of Queuing Place

5. Architecture of the Test System

The following criteria have been considered during hardware and software selection for this study:

- hardware is to be widely available, represent different classes of systems and provide enough power to run appropriate applications;
- software should be a common and popular web application with comprehensive internal structure, which hardly can be modeled in details.

For our test system we have chosen Content Managing System Joomla [13] deployed in the distributed environment with load balancer which is topologically placed in front of two web servers. [Figure 3]. Joomla is popular and convenient CMS used for many web sites. Being based on PHP and MySql it can be easily installed and configured in most of the environments.

Selected hardware and installed software are introduced in Table 1. All these hardware has been connected through 100Mbit Ethernet.

Table 1. Hardware/Software for Test System

Title	Hardware	Software
Web Load Balancer	2.16Ghz Core2Duo PC	Apache Web Server v2.2 [12] with enabled modules *mod_proxy* and *mod_proxy_balancer*
Web Server 1	3.5Ghz Celeron PC	Apache Web Server v2.2 with enabled module *mod_fast_cgi*, PHP v5.2.6 [16] compiled with *fast_cgi* support; CMS Joomla, v1.5.3
Web Server 2	2.2 Ghz Core2Duo PC	
DB Server	2.2 Ghz Core2Duo PC	MySQL v 5.1

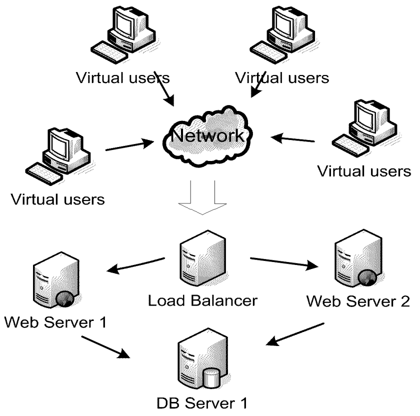

Figure 3. Test System Infrastructure

Three different configurations have been analyzed in our study:

- Configuration 1: all client requests are processed by WebServer 1
- Configuration 2: all client requests are processed by Web Server 2
- Configuration 3: all client requests are sent to Web Load Balancer, which forwards them to Web Server 1 and Web Server 2.

To verify model predictions we have measured throughput and response time characteristics of the system in all these configurations by HP Load Runner [7]. This tool is widely used for load-testing of web applications.

6. Models of the System

To define the performance models for our system we separated the following entities, which have been later mapped to the appropriate model elements:

- Clients: independent tasks which send incoming requests to the Load Balancer. We used so-called closed workload where client can send a next request only when server has completed the previous one.
- Load Balancer: server which forwards requests to one of the Web Servers. The time for forwarding is negligible.
- WebServer 1 and WebServer 2: servers which run the Joomla installation and perform main part of request processing. Times to process single request by these servers are T_{WS1} and T_{WS2} respectively. For convenience in our models we use inverse values called service rates: $Rate_{WS1}=T_{WS1}^{-1}$ and $Rate_{WS2}= T_{WS2}^{-1}$.
- Database: server which runs MySql installation. Service rate for this server is $Rate_{DB}$.

The rates have been defined through a normalization procedure described later. In the next sections we'll show how these entities are mapped to model elements.

6.1 LQN Model

In LQN model every entity derived above from the system architecture is mapped directly to a task element. Since tasks are running on processors and contain entries we have to define two more kinds of elements to complete the model. Graphical notation of the resulting model is shown on Figure 3. Every task is mapped to processor (circle) and contains entry which consumes processor time. Web Server 2 is shown with two tasks to reflect its Core2Duo CPU.

Time demands for WebServer 1, WebServer 2 and Database entries are defined as $Rate_{WS1}^{-1}$, $Rate_{WS2}^{-1}$, $Rate_{DB}^{-1}$ respectively. The demands for Clients and Load Balancer entries are negligible.

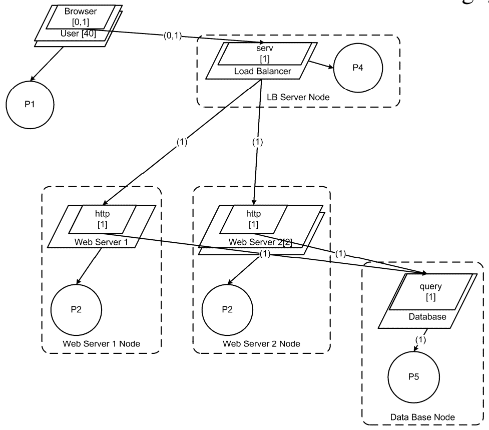

Figure 3. LQN Model of Test System

6.2 PEPA Model

In case of PEPA model system entities are mapped to components. Every component runs infinitely and cooperates with other ones through synchronizing actions. Also in PEPA model there is no need to implement Load Balancer directly – in can be realized implicitly through choice cooperator of the client component. Full PEPA model is listed below.

```
// Rate for intermediate actions
r=10000;
// Ratio of the requests to the WS_2.
d=0.7;
// CLIENT_0 - client, also reflects the presence of Load
balancer
CLIENT_0=(move,r).CLIENT_0_0;
CLIENT_0_0 =(move_2_0,d*r).CLIENT_0+
    (move_1_3,(1-d)*r).CLIENT_0;
// Core2Duo Web Server 2 with two cores
WS_2_0=(move_2_0,T).WS_2_0_0;
WS_2_0_0=(move_1_1,r).WS_2_0
+(move_1_2,r).WS_2_0;
T_1 =     (move_1_1,T).(comp_1, RateWS2).
(move_db, T).T_1;
T_2 =     (move_1_2,T).(comp_2, RateWS2).
(move_db, T).T_2;
// Web Server 1
WS_1 =    (move_1_3, T).(comp_3, RateWS1).
(move_db, T).WS_1;
// Database
DB_0=(move_db, RateDB).DB_0
CLIENT_0[40]<move_2_0,move_1_3>     ((WS_2_0     <move_1_1,
move_1_2> (T_1 <> T_2 ))<> WS_1) <move_db> DB_0
```

6.3 QPN Model

Elements of QPN formalism provide an easy way to represent architecture entities [9]:

- Web Server 1, Web Server 2, DB Server are mapped to queuing places with Processor-Sharing (PS) scheduling strategy. The service rates of places correspond to time demands of servers;
- Clients are mapped to queuing place with Infinite Server (IS) scheduling strategy which is used to represent clients sending requests to the system. The service time of this place corresponds to the average load;
- Thread pools of Web Servers are mapped to an ordinary place.

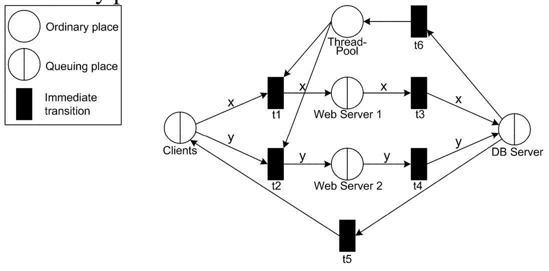

Figure 4. QPN Model of Test System

Two types of tokens (token colors) are used in the model for distinguishing requests between web servers: *x* and *y*. In QPN model Load Balancer was not implemented directly as it was done in PEPA. This implementation was realized implicitly through choice of immediate transition from a client request.

7. Models Calibration

Before starting models analysis we need to define $Rate_{WS1}$, $Rate_{WS2}$ and $Rate_{DB}$ parameters in our models, i.e. calibrate our models. To accomplish it we performed a measurement of the transaction response time of the Web Server 2 with only one virtual user. Received response time of overall CPU load was about 0.6 second with average 35% percentage. Taken into account that this server has two cores and both of them equally participate in handling requests (due to used module mod_fast_cgi which starts several instances of PHP) the time to handle the request by one core is 0.6*0.35*2=0.42 sec. This is Web Server CPU depended part of the request processing and the estimation of the service rate of another CPU can be received by normalizing of this time to CPU frequency. For Web Server 1 we received: 0.42/3.5*2.2 = 0.264 sec. Among the Web Server CPU depended part of the request processing there is another part, which takes 0.6-0.42=0.18 sec. and it's treated as Database processing time. The rates of the models are defined as follows:

- $Rate_{WS1}$: 0.42^{-1} =2.38 sec-1
- $Rate_{WS2}$: 0.26^{-1} =3.78 sec-1
- $Rate_{DB}$: 0.18^{-1} =5.55 sec-1.

These values have been used in model analysis.

8. Models Analysis

During model analysis we made performance predictions of the system behavior under different external workload by varying number of clients simultaneously sending requests to the system from 1 to 40. Every model has been analyzed with its corresponding technique.

For LQN model we used *spex* utility, which is included in LQNS package. This tool reads parameterized LQN model, replaces parameter names with parameter values and then solves the model using *lqns* solver. *Spex* gathers measures of interest and creates text report with obtained data.

During analysis of PEPA model we used *experimentation* feature built-in in the PEPA Workbench. It provides the same functionality for PEPA models as *spex* for LQN models. Also has graphical user interface and is able to build graphs from gathered data.

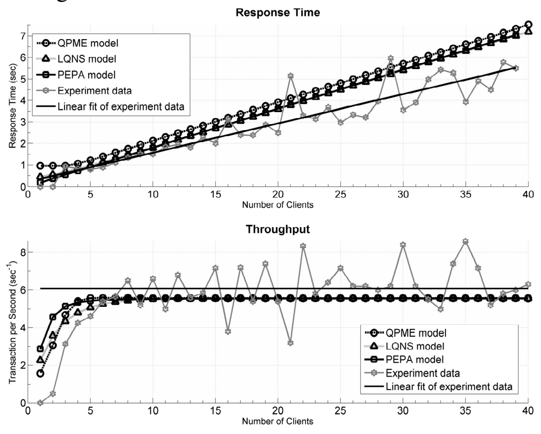

Figure 5. Metrics for Configuration 3

Analysis of QPN model has been done by *SimQPN* simulator of QPME package executed as a standalone Java application. QPME doesn't have any possibilities to simulate parameterized QPN models. That's why we developed a Perl script to simulate different workload by replacing parameters of QPN model in a loop and extract essential data from *SimQPN* output for further analysis.

Simulation results and experimental data for infrastructure 3 with Load Balancer are presented on Figure 5 and relative errors for all analyzed configurations are shown on Figure 6. It can be seen that our models are able to predict throughput (transaction per second) of the system with higher precision with errors less then 10%. While the predictions for response time are less precise and their errors are below 30%. The most possible reason of these errors is an influence of an unrevealed internal system structure. So it would be promising to analyze the effect of hidden queue buffers and request handle threads to model predictions.

It has been revealed that the fastest utility for model simulation is *lqns*. From another point of view QPME tool and PEPA Workbench provide convenient GUI editors for model creation.

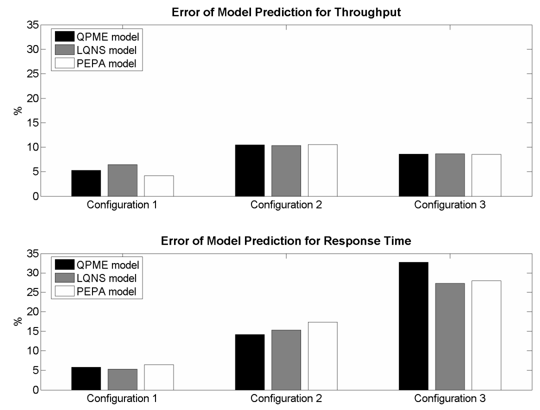

Figure 6. Relative Errors

9. Conclusion and Future Work

During this study we have created performance models of the content management system "Joomla" using three different software performance description formalisms. Models have been calibrated using measurements of the test system with single-user workload. The comparison of the model simulation results with experimental data obtained by HP Load Runner has shown that all of the applied techniques are able to predict system behavior without detailed knowledge about the internal system structure. Difference from model predictions and experimental results lay in the acceptable area: for throughput it's less then 10%, for response time – less then 30%. Such results make possible to use model predictions during early performance analysis of infrastructure for distributed business applications.

The investigation of errors caused by hidden structure of system components and methods to estimate them is the subject for further work.

10. References

[1] Bause F., "Queuing Petri Nets – a formalism for the combined qualitative and quantitative analysis of systems", *5th International Workshop on Petri Nets and Performance Models,* Toulouse(France), pp 14-23, 1993.

[2] Bause F., P. Buchholz, and P. Kemper. „Integrating Software and Hardware Performance Models Using Hierarchical Queuing Petri Nets". *In Proceedings of the 9. ITG / GI - Fachtagung Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen, (MMB'97)*, Freiberg (Germany), 1997.

[3] Benjamin C. Lee, David M. Brooks, Bronis R. de Supinski, Martin Schulz, Karan Singh, Sally A. McKee, "Methods of Inference and Learning for Performance Modeling of Parallel Applications", *PPoPP'07,* San Jose, California, USA ,March 14–17, 2007.

[4] Franks G., S. Majumdar, J. Neilson, D. Petriu, J. Rolia, and M. Woodside. "Performance analysis of distributed server systems", *6th Int. Conf. on Software Quality (6ICSQ)*,pages 15-26, Ottawa, Canada, Oct. 1996.

[5] Gilmore S. and Tribastone M., "Evaluating the Scalability of a Web Service-Based Distributed e-Learning and Course Management System", *WS-FM 2006, LNCS 4184,* pp. 214–226, 2006.

[6] Hillston J., "A Compositional Approach to Performance Modeling" *Cambridge University Press, 1996.*

[7] HP Load Runner, "HP essential knowledge series: an introduction to load testing for web applications", *White paper*, 2007.

[8] Karlsson M., M.Covell, "Dynamic Black-Box Performance Model Estimation for Self-Tuning Regulators", *Proceedings of the Second International Conference on Autonomic Computing (ICAC'05)*

[9] Kounev S., "Performance Engineering of Distributed Component-Based Systems – Benchmarking, Modeling and Performance Prediction", Shaker Verlag, 2005.

[10] Omari T., Franks G., Woodside M., Pan A., "Solving Layered Queueing Networks of Large Client-Server Systems with Symmetric Replication", *WOSP'05,* July 12-14, 2005

[11] Yin L., S.Uttamchandani, R.Katz, "An Empirical Exploration of Black-Box Performance Models for Storage System", *14th IEEE International Symposium on Modeling, Analysis, and Simulation*, 2006

[12] Apache HTTP Server, http://httpd.apache.org/

[13] CMS Joomla, http://www.joomla.org/

[14] Layered Queuing Network Solver software package, http://www.sce.carleton.ca/rads/lqns/

[15] PEPA Workbench, http://www.dcs.ed.ac.uk/pepa/

[16] PHP, http://www.php.net/

[17] Queuing Petri net Modeling Environment, http:// sdq.ipd.uka.de/people/samuel_kounev/projects/QPME