Simulating and Modeling Secure Web Applications

Ramon Nou, Jordi Guitart, and Jordi Torres

Barcelona Supercomputing Center(BSC)
Computer Architecture Department
Technical University of Catalonia
C/Jordi Girona 1-3, Campus Nord UPC
E-08034 Barcelona, Spain
{rnou, jguitart, torres}@ac.upc.edu

Abstract. Characterizing application servers performance becomes hard work when the system is unavailable or when a great amount of time and resources are required to generate the results. In this paper we propose the modeling and simulation of complex systems, such as application servers, in order to alleviate this limitation. Using simulations, and specifically coarse-grain simulations as we propose here, allows us to overcome the necessity of using the real system while taking only 1/10 of the time than that of the real system to generate the results. Our simulation proposal can be used to obtain server performance measurements, to evaluate server behavior with different configuration parameters or to evaluate the impact of incorporating additional mechanisms to the servers to improve their performance without the necessity of using the real system.

1 Introduction

We can view an Application Server based on the J2EE platform as a complex system, several things happen at the same time, and there are many levels involved; from threads to TCP/IP, including cryptographic and security issues. At the same time, all information that is confidential or has market value must be carefully protected when transmitted over the open Internet. Security between network nodes over the Internet is traditionally provided using HTTPS[1].

The increasing load that e-commerce sites must support and the widespread diffusion of dynamic web content and SSL increase the performance demand on application servers that host these sites. At the same time, the workload of Internet applications is known to vary over time, often in an unpredictable fashion, including flash crowds that cannot be processed. These factors sometimes lead to these servers getting overloaded. In e-commerce applications, which are heavily based on the use of security, such server behavior could translate to sizable revenue losses. These systems are known to be very complex to characterize. In our research, we need to spend four hours per test (including the start and stop of the server, the server warm up time, and several executions corresponding to several number of clients and different server parameters), but this time is

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part I, LNCS 3991, pp. 84–91, 2006.

real time and exclusive, so resources are difficult to get. How could we test some hypotheses or ideas without using these resources? We propose to resolve this by estimating the performance measurements using some kind of performance model. With this approach we can get an approximation of the results using less resources and time.

Different approaches have been proposed in publications on performance analysis and prediction models for this type of e-business systems. Most of them exploit analytical models where analysis is based on Markov Chain Theory [2]. Queuing Networks and Petri Nets are among the most popular modeling formalisms that have been used. But due the complexity of todays e-business systems, the analytical methods for solving can not be used. Only by simplifying the system we can obtain manageable models. On the other hand, these systems cannot model, in an easy way, timeouts behavior. The simulation model is an abstract representation of the system elements and their interactions, and an alternative to analytical mathematical models. The main advantage of simulations is that it overcomes the limitation of complex theoretical models, while the methodological approach to simulation models and the analysis of simulation results is supported by statistical principles developed in the last 30 years. There are several works that simulate systems in order to extract information about them [3, 4], but in general, the number of proposals including modeling application servers and problems like the ones we are facing are scarce.

We are able to get results that would take a whole day on a real system (testing some parameter values), in less than an hour using only a desktop computer. This gives us the possibility to test several QoS policies in real time without having it implemented and running on a real machine.

The rest of the paper is organized as follows: Section 2 introduces the analyzed system, an application server with SSL security. Section 3 explains simulation environment and tools used. On Section 4 we describe the experimental environment. Inside Section 5 we explain all the blocks that build the simulated model. Section 6 compares simulation results with experimental results, and shows some interesting findings that can be obtained from simulated models. Some other experiments with simulation can be found on Report [5]. An extended version of this paper with more results can be found on Report [6].

2 Secure Dynamic Web Applications

The two components we are using on our systems are Dynamic web applications and SSL security. Dynamic web applications are multi-tiered applications, normally using a database. The client receives a HTML page with information gathered and computed from the database by the application server. Communication between client and server is protected using the SSL protocol ([7]). SSL increases the computation time necessary to serve a connection, due to the use of cryptography. The impact of SSL on server performance has been evaluated in [8]. It concludes that saturation of a server with SSL security provokes the degradation of the throughput. Further information about an admission control

using SSL connections and its impact can be found in [9]. More information about the impact of using SSL on server performance can be found in [8].

3 Simulation Proposal

We are using a simulation tool and a performance analysis framework to generate the simulation.

Simulations are usually much more computationally intensive than analytic models. On the other hand, simulation models can be made as accurate as desired and focus over some specific target. To the best of our knowledge, there are not any simulation packages that are specifically oriented for these types of systems. We decided to use Objective Modular Network Testbed in C++ (OMNet++) [10]. OMNet++ offers us a way to build our modules with a programming language, so we can test our changes and fine-tune server model in a fast way.

In order to obtain the computation times of the different services, which will be used to construct the simulations model we propose using a performance analysis framework developed in our research center. This framework, which consists of an instrumentation tool called Java Instrumentation Suite (JIS [11]) and a visualization and analysis tool called Paraver [12], allows a fine-grain analysis of dynamic web applications. Further information about the implementation and its use for the analysis of dynamic web applications can be found in [11].

4 Benchmark and Platform

We simulate an e-business multi-tiered system composed of an application server, a database server and some distributed clients performing requests to the application server. This kind of environment has been frequently modeled [13], but we have added several new components to the simulation, which have not been considered before. First, a SSL protocol between the clients and the application server, and lastly, timeouts, which are harder to get on an analytical model.

We use Tomcat v5.0.19 [14] as the application server. In addition, in order to prevent server overload in secure environments, [9] we added to the Tomcat server a session-oriented adaptive mechanism that performs admission control based on SSL connections differentiation. Further details can be read on [9]. The client is using Httperf [15] to generate the requests of the workload from RUBiS benchmark [16] (RUBiS attempts to produce a workload similar to eBay auction site). The workload is divided into sessions, where every session issues several requests (burst and not-burst) with a thinktime between them.

5 Model Description

The system was modeled using five black boxes 1: A client that simulates Httperf and RUBiS (several others can be used, to represent other kind of scenarios), an

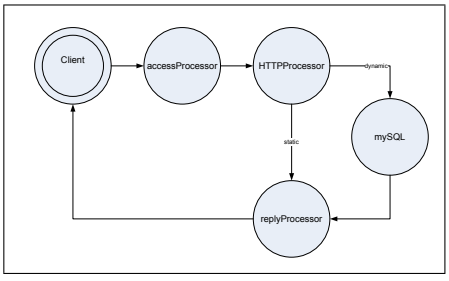


Fig. 1. Simulation modules

accessProcessor that simulates operating system backlog queue of connections and allow the setting up of admission policies. HTTPProcessor manages connections using SSL handshake and reusing SSL. HTTPProcessor also processes requests and sends to the MySQL database as needed. To conclude a replyProcessor gets the reply from MySQL or the HTTPProcessor and sends it to the client.

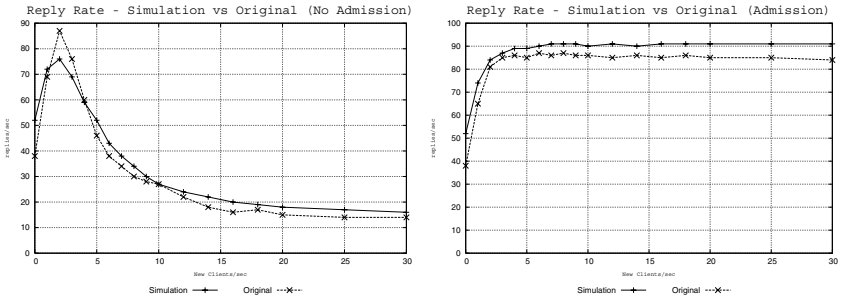
We do not want to build a system with a great level of detail; we do not include threads in our model even though we modeled a simplified HTTP 1.1 scheme. Our objective was to achieve an approximation of the real system values, without using a large processing time. This has some handicaps, because we cannot simulate or retrieve data for the response time (for example); we had not modeled all the components that affect it on a per-request level. We seek throughput data only so our model is enough for us, response time modeling will give more detail to cache-like mechanisms. We will show more specific details for some of these blocks. The code from the simulation framework can be found on [17] for more detail.

accessProcessor. Has a queue that sends requests to HTTPProcessor (and performs admission control). From here it is easy to control limitation policies on HTTPProcessor so we can quickly test several policies by just changing the code. Inside this block we had modeled typical Linux buffers behavior(backlog).

HTTPProcessor. Uses Processor Sharing scheduling. We are using a two pass processing mechanism: when a request arrives we process the connection (SSL process and request service) then we process static data as much as is needed. Next we see if the request needs further processing and we send it to the database block. These two phases were created to simulate the behavior of Httperf timeouts. Requests from this block can go to mySQL or to the client.

6 Evaluation

Here we include the validation of our approach when comparing the results obtained with real life systems. We also present some experiments showing the usefulness of our approach to evaluate system behavior when a real system is not available.



(a) original (b) admission

Fig. 2. Throughput comparison between original Tomcat and with admission control in the real system versus when simulated

6.1 Model Validation

Comparison with the original Tomcat. Fig. 2.a shows the Tomcat throughput as a function of the number of new clients per second initiating a session with the server. When the number of clients needed to overload the server has been achieved, the server throughput degrades until approximately 20% of the maximum achievable throughput while the number of clients increases. The cause of this great performance degradation on server overload has been analyzed in [8]. When the server is overloaded, it cannot handle the incoming requests before the client timeouts expire; In this case, clients with expired timeouts are discarded and new ones are initiated, provoking the arrival of a great amount of new client connections that need the negotiation of a full SSL handshake, causing degradation of the server performance. In our simulation, as shown in Figure 2.a, the graph is a similar shape (with approximate values).

Comparison with Tomcat with Admission Control. Considering the described behavior, it makes sense to apply an admission control mechanism in order to improve server performance. Figure 2.b shows the Tomcat throughput as a function of the number of new clients per second initiating a session with the server. Notice that the server throughput increases linearly with respect to the input load (the server scales) until a determined number of clients hit the server. At this point , the throughput achieves its maximum value. Until this point, the server with admission control behaves in the same way as the original server. However, when the number of clients that would overload the server has been achieved, the admission control mechanism can avoid the throughput degradation, maintaining it in the maximum achievable throughput; in this case, the simulation is also able to achieve almost the same results than the real system, as shown in Figure 2.b.

6.2 Simulation Results

In the previous subsection, we have demonstrated that the simulation with a simple model, using a low level of detail, adapts well to the real system.

In addition, this kind of coarse-grain simulation allows us to run tests over hardware or systems that are not available and still obtain performance predictions. Its low complexity allows us to simulate long experiments without a high computational cost. Nevertheless, if it is necessary, this simulation model can be extended in any detail level in order to obtain other types of information.

In order to illustrate the possibilities that our simulation approach offers, in this section we numerate some experiments that cannot be performed in the real system (because it is not available or the execution time needed to run them is extremely high). In this way, we demonstrate that simulations can be very helpful to answer a great number of questions; confirming or discarding hypotheses and providing hints about server behavior with different system configurations.

Scalability of the application. Although our model is very simple, it can describe with great success the real model pattern. If we need to know the behavior of our application with a system with more CPUs, we can do it. However these kind of results with our simulation should be restricted to a certain number of CPUs. To enhance this number, we should increase complexity of the model to enable a more detailed behavior of contention between processors (for example to model thread behavior inside HTTPProcessors). With the same environment we can use simulation to provide us with a way to test several QoS admission policies, and how they will scale on the real system without using real resources. As an example we do not have machines available with more than 4 CPUs, so simulation fills this gap. Simulation is used to determine if a proposal is worth testing on a real system or not.

Extracting extra data from simulations. We can adapt the model to help us to extract other kind of data also. We can accomplish tests that would cost a lot of resources, while simulation also gives us more flexibility to add variables and count things like the maximum number of opened sessions on an instant. With this variable we could see how admission control affects the number of opened sessions. These kinds of statistical results can be achieved with ease by adding little code to the model.

Testing hypothesis involving system changes. Simulations were used to test how different algorithms, which change the order of requests, can improve server performance. The order should be changed on the backlog queue, so it requires several Linux Kernel modifications. Testing on a real system would involve rebooting machines (server is multi-user), modifying kernel internals, debugging and several other time consuming chores. With a simulation we were able to test these changes with a low cost (in time and resources). After changing the order of the backlog queue we obtained the results that we expected. Now we can start moving resources to implement our hypothesis on a real system, with a lower probability of failure.

7 Conclusions

In this paper we have presented a method to simulate an existing system and shown how we can obtain some experimental data without the need to use much resources or time. In this work, we used only data from a single CPU system run, and from its sample workload. Our model uses OMNET++ to the system behavior with all the detail we need, and predicts the shape of the real system graph. Although we do not model the system with deep enough detail to predict real numbers, we found that the results are very similar and good enough to make some predictions. In our model, every time we added a new feature, the results became more accurate, and we had achieved a great approximation on Figure 2.b.

We added to the normal simulations of application web servers some handicaps like timeouts and SSL, to show how it affects the server, and how it shapes the real life graphics. For more specific results we will need a more detailed model (i.e. OS contention with more than one CPU).

Using simulations gives us more flexibility and capability to extract data and run some tests that could be unaffordable with real resources. We can measure what throughput we could achieve (or at least an approximation) with the addition of more processors, threads or with a change of the number of clients. We did not need the real system for this, only a simulation. We can slightly modify the model to help us to decide what proposal of admission control is the best in a first approach. For example we can quickly analyze the number of rejected sessions(rejected sessions leads to lost revenue).

Simulations can be useful to test changes such as changing the backlog from FIFO to LIFO [18] and see how it improves performance. To conclude, coarse-grain simulations could give us basic guidelines to help us to test some changes in our systems and there are a great number of questions that can be answered by them. Thanks to the simulations we have obtained results such as those on 6.2. In these we can analyze the behavior of the system when we have more CPUs. Thanks to the validation we have seen that the obtained results come closer to the real ones. This can help us to evaluate these policies in a fast and quite reliable way and give us some hints to know if a proposal is good or bad.

Future work will include introducing simulations as a predictive tool inside an application server.

Acknowledgment

This work is supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2004- 07739-C02-01. Thanks to Lídia Montero from Dept. of Statistics and Operations Research, UPC for her help. For additional information please visit the Barcelona eDragon Research Group web site [17].

References

1. Rescorla, E.: HTTP over TLS. RFC 2818 (2000)
2. Bolch, G., Greiner, S., Meer, H.D., Trivedi, K.S.: Queueing networks and markov chains. Modelling and Performance Evaluation with Computer Science Applications. Wiley, New York (1998)
3. Stewart, C., Shen, K.: Performance modeling and system management for multi-component online services. NSDI (2005)
4. Uragonkar, B., G.Pacifi, Shenoy, P., Spreitzer, M., Tantawi, A.: An analytical model for multi-tier internet services and its applications. SIGMETRICS'05, Alberta, Canada (2005)
5. Nou, R.: Sorting backlog queue, impact over tomcat. Research Report UPC-DAC-RR-CAP-2005-30 (2005)
6. Nou, R., Guitart, J., Torres, J.: Simulating and modeling secure e-business applications. Research Report UPC-DAC-RR-2005-31 (2005)
7. Dierks, T., Allen, C.: The tls protocol, version 1.0. RFC 2246 (1999)
8. Guitart, J., Beltran, V., Carrera, D., Torres, J., Ayguadé, E.: Characterizing secure dynamic web applications scalability. IPDPS'05, Denver, Colorado, USA (2005)
9. Guitart, J., Beltran, V., Carrera, D., Torres, J., Ayguadé, E.: Session-based adaptative overload control for secure dynamic web application. ICPP-05, Oslo, Norway (2005)
10. WebPage: Omnet++. http://www.omnetpp.org (2005)
11. Carrera, D., Guitart, J., Torres, J., Ayguadé, E., Labarta, J.: Complete instrumentation requirements for performance analysis of web based technologies. ISPASS'03, pp. 166-176, Austin, Texas, USA (2003)
12. WebPage: Paraver. http://www.cepba.upc.es/paraver (2005)
13. Menacé, D.A., Almeida, V.A.F., Dowdy, L.W.: Performance by Design. Pearson Education (2004)
14. WebPage: Jakarta tomcat servlet container. http://jakarta.apache.org/tomcat (2005)
15. Mosberger, D., Jin, T.: httperf: A tool for measuring web server performance. Workshop on Internet Server Performance (WISP'98) (in conjunction with SIGMETRICS'98), pp. 59-67. Madison, Wisconsin, USA (1998)
16. Amza, C., Cecchet, E., Chanda, A., Cox, A., Elnikety, S., Gil, R., Marguerite, J., Rajamani, K., Zwaenepoel, W.: Specification and implementation of dynamic web site benchmarks. WWC-5, Austin, Texas, USA. (2002)
17. WebPage: Barcelona edragon research group. http://www.bsc.es/eDragon (2005)
18. N.Singhmar, Mathur, V., Apte, V., D.Manjunath: A combined lifo-priority scheme for overload control of e-commerce web servers. International Infrastructure Survivability Workshop, Lisbon, Portugal (2004)