COSIM(HARDWARE-SOFTWARE COSIMULATOR): JAVABEANS-BASED SIMULATION TOOL FOR WEB APPLICATIONS

Kangsun Lee , Jaeho Jung and Youngsuk Hwang

Dept. of Computer Engineering
MyongJi University
San 38-2 Namdong, YongIn
Kyunggido, Korea 449-728, Korea

ABSTRACT

CoSim (Hardware and Software Co-Simulator) is a JavaBeans-based simulation tool for validating systems architecture and estimating performance of web applications. CoSim has four components: Modeler, Translator, Engine and Scenario. Users start from Modeler to describe systems architecture in UML(Unified Modeling Language) deployment diagram, and then specify hardware & software performance parameters such as execution delay, network topology, and frame size. All information specified on Modeler are sent to Translator, and then automatically converted to Java programs. Scenario is responsible to run the Java program and produce results in text reports and graphs. Developers can reduce development time and cost by validating systems architecture of web applications before the actual deployment.

1 INTRODUCTION

As electronic business expands all around the world, surges in volumes of e-business transactions, delays, and outages occur as systems and networks become overloaded. [1] Surveys and studies indicate that slow downloading time is one of the most often cited reasons that an online customer leaves a site and looks for another vendor's website instead. [2] A good web application requires not only good contents but also satisfactory performance, for example, speed and scalability. Web application analysts and designers should answer the following typical questions to guarantee performance criteria:

- What is the average response time of an e-commerce function during the busiest hour?
- What would be the online store performance if the number of customer sessions doubles during the peak hour?
- What components of the site should be upgraded to be scaled up? Database servers? Web servers? Application servers? Network link bandwidth?

Although these questions are typical in web application development, most development process handles these performance issues only during testing, or worse after deployment. Therefore, failures to meet performance criteria will require redesign and implementation process.

Most performance simulation tools lack explicit support for evaluating systems architecture of web applications. Unique characteristics and typical what-if questions of web applications are not easily expressed in the tools. Quantitative approaches for evaluating web applications are easily locatable on many references[1], however, there are not many supporting software tools specifically designed for evaluating web applications.

In this paper, we present CoSim (Hardware and Software Co Simulator), a design-time performance prediction tool for web applications. CoSim predicts performance in the early cycle of development: analysis and design. CoSim starts from the deployment diagrams of UML(Unified Modeling Language) which show the configuration of run-time processing elements and the software processes living on them[4]. CoSim extends the UML deployment diagram in the sense that performance input data are specified to each component of the run-time architecture, for examples, component execution time and network type. These information are automatically translated to Java codes, and simulated on CoSim. Performance estimation results are demonstrated in summary reports and graphs, which allow the architecture team to estimate the system performance and discuss issues such as scalability and maintainability of web applications being built.

This paper is organized as follows: In Section2, we introduce CoSim architecture. A complete process from the model specification to estimation results on ComSim is illustrated in Section 3 through an on-line investment company example. We conclude in Section 4 with future works to achieve.

2 COSIM ARCHITECTURE

ComSim is composed of Modeler, Translator, Engine and Scenario as shown in Figure 1. Section 2.1 – 2.3 discuss each of the components.

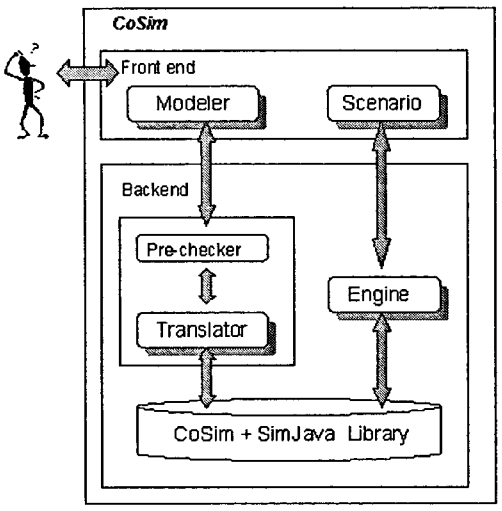

Figure 1. CoSim Architecture

2.1 Modeler

Modeler provides GUI(Graphical User Interface) to model and simulate a system architecture. A user creates a system architecture in a deployment diagram of UML. A deployment diagram shows the physical relationships among software and hardware components in the delivered system [12]. A node is a run-time resource, such as a computer, device, or memory. Physical relationships among the computational units are represented as links.

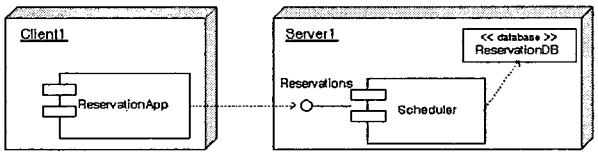

Figure 2. UML deployment diagram: An example

Figure 2 shows a deployment diagram for an online reservation system. The diagram shows the kinds of nodes in the system and the kinds of component they hold. A node is shown as a cube symbol.

The UML deployment diagram is good to show the run-time architecture of web applications. However, the diagram itself is not an executable simulation model ; we can not get the performance estimation data from the diagram.

Below is the part of the information we need in order to estimate performance from the diagram.

- Node related data – component complexity, component execution time (estimated value), hardware information (processor type, memory size, hard disk capacity, etc.)
- Link related data – network type, band width, etc.
- Simulation objectives – server utilization, client response time
- General information on simulation – schedule information, simulation time, number of iterations, confidence interval

Users draw an extended deployment diagram on Modeler, where they specify the run-time architecture of the web application along with the performance data to simulate the architecture.

Modeler is implemented in JavaBeans. JavaBeans component is a component model of Java.[8] A JavaBean is a reusable software component that can interactively be modified and combined with other components. We define common and generally accepted interfaces for Modeler. Modeler components can be used in any development tool supporting the JavaBeans standard, therefore increases reuse opportunities.

2.2 Translator

Pre-checker examines whether the correct Java codes can be constructed from the information gathered in Modeler. If enough, Translator generates Java codes equivalent to the extended deployment diagram specified in Modeler.

Translator generates Java codes by the following three steps:

- Model information analysis step – Translator analyzes, classifies, and saves model information in an intermediate language form defined in between the Modeler and Translator. Pre-Checker helps this step.
- Simulation algorithm generation step – Translator generates simulation structure in Java by using SimJava library. Translator constructs classes for the nodes in the deployment diagram, and implements these classes with threads in Java. The scheduling structure (the order of simulations) between nodes are determined by analyzing input/output relationships between the nodes and links. Links are implemented by the interactions between the threads.
- Simulation objective generation step – CoSim provides response time, utilization, throughput, and transmission time as the performance measurements. Translator has a set of methods to calculate the measurements, and records which methods to use for each of the nodes.

SimJava is a process-based discrete event simulation package for Java. [6] It is a collection of entities each of which runs in its own thread. Since the entities are connected together by ports, they can communicate with each other by sending and receiving event objects. Each

node has independence and concurrency, therefore, simulation can be more realistic. CoSim extends SimJava library to construct the simulation codes. More information on SimJava can be found in Reference [6].

2.3 Engine & Scenario

Engine runs Java codes generated from Translator with the specified simulation time. Engine uses CoSim run time library(SimJava Library, and other utility library) to simulate the codes, and then produces simulation results. Simulation results are summarized by Scenario in a report and graphs.

3 ONLINE INVESTMENT COMPANY

In this section, we illustrate how performance estimation is proceeded on CoSim by an example.
Consider an online investment company that is planning a site to support a financial investment transaction application. The site architecture has three layers. The first one consists of a Web server accessed by Internet clients through a browser. The second layer is an application server that receives transaction requests generated at the Web server. To carry out the transaction, the application server communicates with the database server and requests specific database information. The Web server provides static HTML documents to customers, collects data, and transfers them to the application. The application server gets requests from the Web server, parses them, and activates the processes that carry out the requested e-business function(e.g., Login, User Information, Show Portfolio, and Sell Stocks). A set of input data for the financial site are given in Table 1.

Table 1. Input Data for the Financial Site

Server	Processor Demand	I/O Demand
Web (WS)	19 msec	12 msec
Application(AS)	40 msec	48 msec
Databse(DB)	35 msec	56 msec

Assume that we want to answer the following five questions:
1) What is the average client response time?
2) What is the utilization for the three servers?
3) What would be the average client response time if the server is replaced by the two times faster one?
4) What would be the average client response time if the number of customer sessions doubles during the peak hour?
5) How would be the average response time increased as a function of the business transaction arrival rate?

Figure 3 shows how the system architecture is specified on CoSim. Users draw the extended deployment diagram on Modeler, and then specifies various performance input data for each of the nodes and links using specification windows. Figure 4 shows two specification windows: Node Specification Window and Link Specification Window. Users specify delay time(processor demand + I/O demand) and frame size in the node specification window. Information on the hardware (process type, memory size, and hard disk capacity) is specified in the hardware specification tab, while information on components (component complexity, and exceptions) is specified in the component specification tab. Information on the links is specified in the link specified information, including network delay time, and hardware data (network type and network bandwidth).

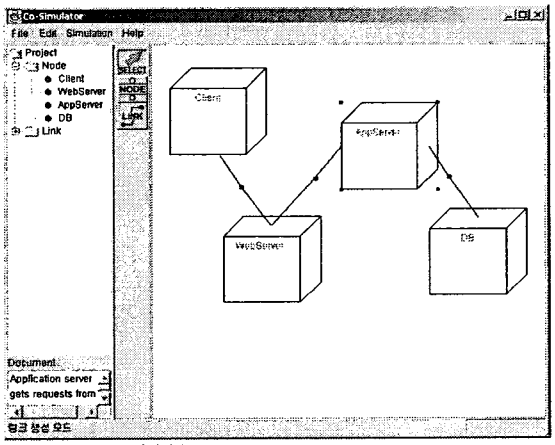

Figure 3. Example: Deployment Diagram on Modeler

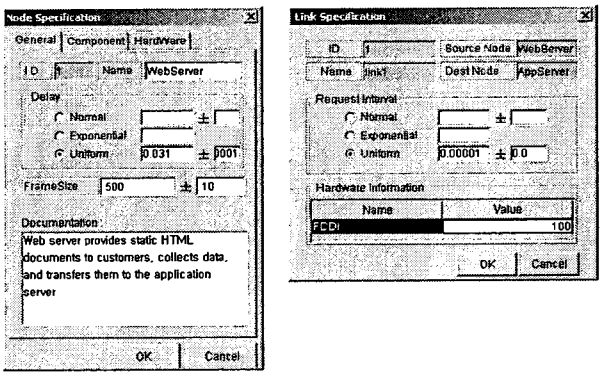

Figure 4. Example: Performance Input Data (Table 1) Specification

Simulation orders are specified with the help of Simulation Wizard as shown in Figure 5. The simulation wizard helps

a user correctly specify 1) the simulation orders between the nodes, 2) simulation objectives of nodes, and 3) simulation time. In this example, a transaction is instantiated by the Client node, WebServer, and AppServer followed by DB. Simulation objectives may be specified by the typical simulation queries as shown in Figure 6.

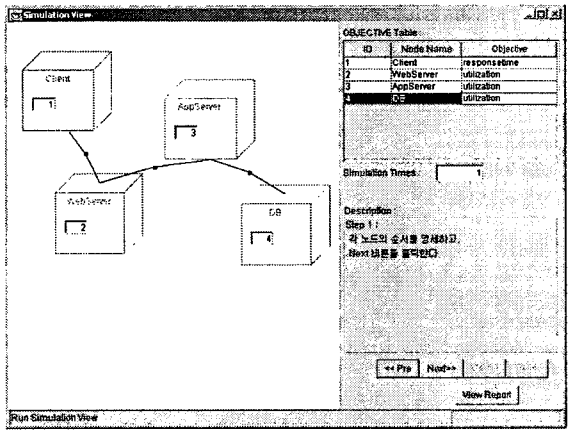

Figure 5. Example: Simulation Wizard

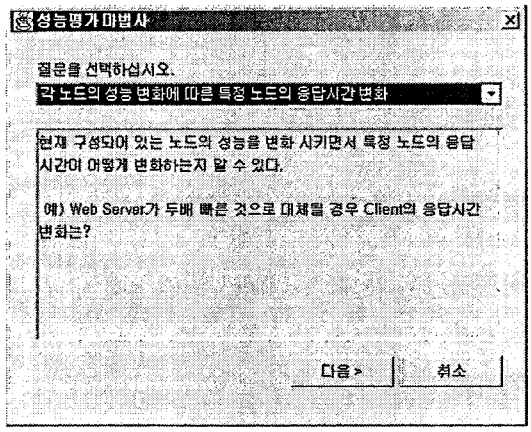

Figure 6. Query-Driven Simulation: Example

Translator generates Java codes after successfully finishing Simulation Wizard. The generated Java codes are executed on Engine using the run time library of CoSim.

Simulation results are summarized as follows:

1) What is the average client response time?: 0.207 sec
2) What is the utilization for the three servers?
   WS: 27.8%, AS: 79%, DB: 79%
3) What would be the average client response time if the server is replaced by the two times faster one?
   New client response time: 0.191 sec
4) What would be the average client response time if the number of customer sessions doubles during the peak hour?
   New client response time: 1.813 sec
5) How would be the average response time increased as a function of the business transaction arrival rate? : As summarized in Figure 7, the average client response time is drastically increased as the client inter-arrival time becomes 11 sec.

Scenario visualizes the simulation results as shown in Figure 8.

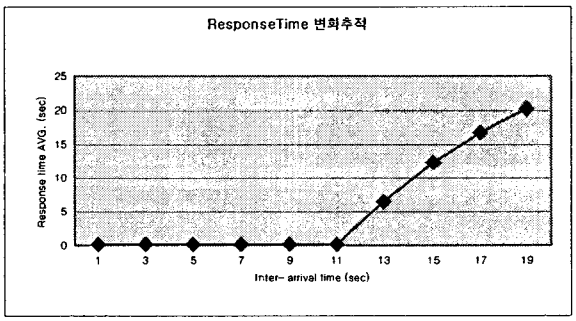

Figure 7. Response time of the Financial Site

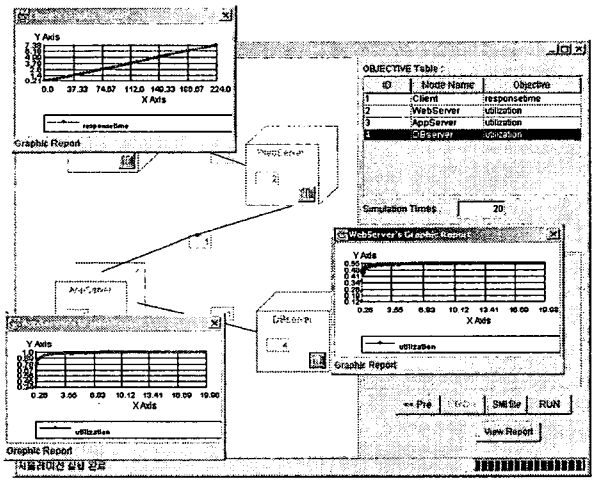

Figure 8. Example: Scenario

4 CONCLUSIONS AND FUTURE WORKS

In this paper, we introduced CoSim, a design-time performance estimation tool for web applications. CoSim generates Java codes which are equivalent to the extended deployment diagram, and then simulates the codes to get the performance data. The system architecture of a web application is validated based on the performance

estimation results provided by CoSim in the earlier stage of development. Failures to meet performance criteria are detected before testing and actual deployment. Therefore, development costs and time can be reduced by minimizing redesign and re-implementation risks.

We illustrated the four important elements of the CoSim: Modeler, Translator, Engine and Scenario. Though the four elements have well-defined interfaces to be reused in the future, we are planning to componentize each of the elements to enable plug-and-play binary reuse. Also, we want to achieve better reuse chances by incorporating DSSA (Domain-Specific Software Architecture) concepts to CoSim. DSSA is based on the concept of an accepted generic software architecture for the target domain. The existence of a domain-specific architecture and conformant component base will dictate a significantly different approach to software application development. [9] We would like to continue out efforts to select specific domains and create a set of architecture on CoSim whose performance are proven to be satisfied. Then, developers will not wait until detailed design or implementation to search for reuse opportunities; instead, he/she will be driven by the architecture whose components and performance have been already verified. For this purpose, we would like to create CoSim domain wizards and reuse library where proven architectures can be created and stored.

REFERENCES

[1] Daniel A Menasce and Virgilio A. F. Almeda, 2000, *Scaling for E-Business*, Prentice Hall

[2] GNV, "GVU's WWW User Surveys", `http://www.gvu.gatech.edu`

[3] Svend Frolind and Pankaj Garg, *Design-Time Simulation of a Large-Scaled, Distributed Object System*, ACM Transactions on Modeling and Computer Simulation, vol 8., no 4. October 1998, pp 374-400

[4] James Runbaugh, Ivar Jacobson & Grady Booch, *The Unified Modeling Language Reference Manual*, Addison-Wesley

[5] Jim Conallen, 1999, *Building Web Applications with UML*, Addison-Wesley

[6] SimJava Library, *Writing simulation using the library*, `http://www.dcs.ed.ac.uk/home/hase/simjava`

[7] Craig Larman, 1998, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, Prentice Hall

[8] Sun Microsystems: *JavaBeans 1.01 API Specification*, 2001, `http://java.sun.com/products/javabeans/glasgow`

[9] R. Taylor, W. Tracz, and L. Coglianese, 1995, *Software Development Using Domain-Specific Software Architectures: CDRL A011A curriculum Module in the SEI style*, SIGSOFT Software Engineering Notes, vol. 20, pp 27-37

[10] Enterprise JavaBeans™ technology, `http://java.sun.com/products/ejb`

[11] Vlada Matena and Beth Stearns, 2000, APPLYING ENTERPRISE JAVABEANS: Component-Based Development For The J2EE Platform, Addison-Wesley Pub Co (Sd)

AUTHOR BIOGRAPHIES

KANGSUN LEE an Assistant Professor of Computer Science and Engineering at MyongJi University in South Korea. She received her BS and MS in Computer Science from the Ewha Women's University, Seoul, South Korea, in 1992 and 1994, respectively, and Ph.D. in Computer and Information Science and Engineering from the University of Florida in 1998. Her research interests are in computer modeling methodology for complex systems. Her email address is <ksl@mju.ac.kr>.

JAEHO JUNG a graduate student of Computer Science and Engineering at MyongJi University in South Korea. He received his BS degree in Computer Engineering from MyongJi University, South Korea, in 2000. His research interests are in computer modeling methodology for complex systems. His e-mail address is <jjho@mju.ac.kr>

YOUNGSUK HWANG a graduate student of Computer Engineering at MyongJi University in South Korea. He received his BS degree in Computer Engineering from MyongJi University, South Korea, in 2000. His research interests include computer modeling methodology for complex systems. His email address is <yshwang@mju.ac.kr>.