Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

WEB-BASED PERFORMANCE VISUALIZATION OF DISTRIBUTED DISCRETE EVENT SIMULATION

Adel S. Elmaghraby
Sherif A. Elfayoumy
Irfan S. Karachiwala
James H. Graham
Ahmed Z. Emam
AlaaEldin M. Sleem

Speed School, Computer Engineering and Computer Science Dept.
University of Louisville
Louisville, KY 40292, U.S.A.

ABSTRACT

This paper reports on an effort to adapt an existing distributed simulation visualization system to become Web accessible. The system was originally developed for performance visualization and experimentation with parameters affecting PDES systems using the Time Warp protocols. This paper presents a model for converting legacy PDES systems to be Web accessible, and discusses the initial results from the conversion effort on this specific application. After finishing this work, we will be able to collect a wealth of data through the Web for future data mining, and to create an intelligent agent for performance tuning of Time Warp applications.

1 INTRODUCTION

The introduction of the World Wide Web has created a universal access interface that transcends geographic boundaries and has created an illusion of machine independence and interoperability for many applications. Traditionally, parallel and distributed simulations have been targeted to special purpose parallel machines or specialized clusters of workstations at a single location. Access to such systems typically requires the use of X-terminals or other types of special purpose interfaces. In many instances remote access is desirable and a Web-based interface would remove many barriers of interoperability, and thus greatly expand access to such simulations.

The need to visualize and experiment with performance parameters of parallel and distributed simulations, rather than just with final simulation results, is of importance for research and for practical performance tuning. In earlier work, a visualization tool based on X-Windows interfaces was developed and tested for usability (Karachiwala 1998). As a follow to this work, this current investigation focuses on development of Web-based access to the visualization package. The long-range goal of the research is to build a database of performance information for various parameter choices that could be analyzed using data-mining techniques.

In evaluating approaches to create Web-based simulations, one can immediately identify two potential methodologies:

1) newly developed Web-aware distributed simulations (typically Java-based), and
2) legacy systems that have been modified to become Web-accessible.

For a legacy system that has evolved over years of experimentation, it is probably best to start by Web-enabling the system or making it Web-accessible. Software engineering of legacy systems has provided insights into how to proceed with this approach. The analogy of our work with encapsulation of legacy systems for discrete-event applications is helpful in understanding our approach.

The following two sections provide an overview of performance visualization issues and approaches and a discussion of the performance parameters for Time Warp distributed simulation. This is followed by a section describing the software architecture used to allow Web access to an existing distributed simulation (Karachiwala 1998). The final two sections discuss a prototype access based upon this architecture, followed by conclusions and future directions for this work.

2 PERFORMANCE VISUALIZATION

Visualization is a tool that helps in understanding the behavior of complex programs and can be used to analyze system performance. Performance visualization of parallel and distributed systems in its simplest form is a method for output data representation. Performance visualization tools can be used for monitoring or for predicting the system performance. There are several tools developed for specific parallel applications. Rover, Heath, and Malony (1995) provide a high level abstract model for performance visualization based on display of performance information. Harden et al., (1995) provide a multi-computer performance monitoring system using a combination of hardware and software. Paragraph is an animation tool used to trace the dynamic behavior of the program (Heath, and Etheridge 1991), and Paradyn is a tool for measuring performance of a large-scale parallel system (Mller et al. 1995).

P^3 T is a performance estimator tool that achieves high estimation accuracy (Fahringer 1995). Avtar is a virtual data environment (Reed et al. 1995) that allows users to explore parallel performance data and modify application and system parameters to see how performance is affected. Lilith Lights (Evensky, Gentile, and Wyckoff 1998) is a visualization tool for monitoring and debugging code execution in a parallel and distributed computing system. Visputer (Zhang, and Marwaha 1995) is a parallel program visualization tool that provides the ability to graphically design programs and visualize their execution.

In the area of parallel simulation, TANGO is a tool developed by Das, Fujimoto and Stasko (1992) to animate the execution of Time Warp optimistic simulation. TANGO helps to visualize simulation behavior and monitor its performance. Visualizing the simulation behavior is achieved by animating the progress of logical processes and permitting users to pause the simulation and move logical processes in any direction. Xtracker is a Motif based visualization tool developed by Bellenot and Duty (1995). It visualizes the execution of parallel simulations. It supports optimistic simulations that ignore the overheads. PVaniM-GTW is a graphically visualization tool developed by Carothers et al. (1997), which supports Time Warp simulation performance by understanding the degradation of performance. Another Motif based visualization tool, developed by Karachiwala (1998) provides a real time animated visualization of performance parameters of a distributed Time Warp simulation. A total of thirty-eight users individually tested and evaluated this visualization system. On average, each user took approximately thirty minutes to reduce performance problems caused due to improper load balancing, and approximately fifteen minutes to determine a satisfactory checkpoint interval as shown in Figure 1. It is clear from

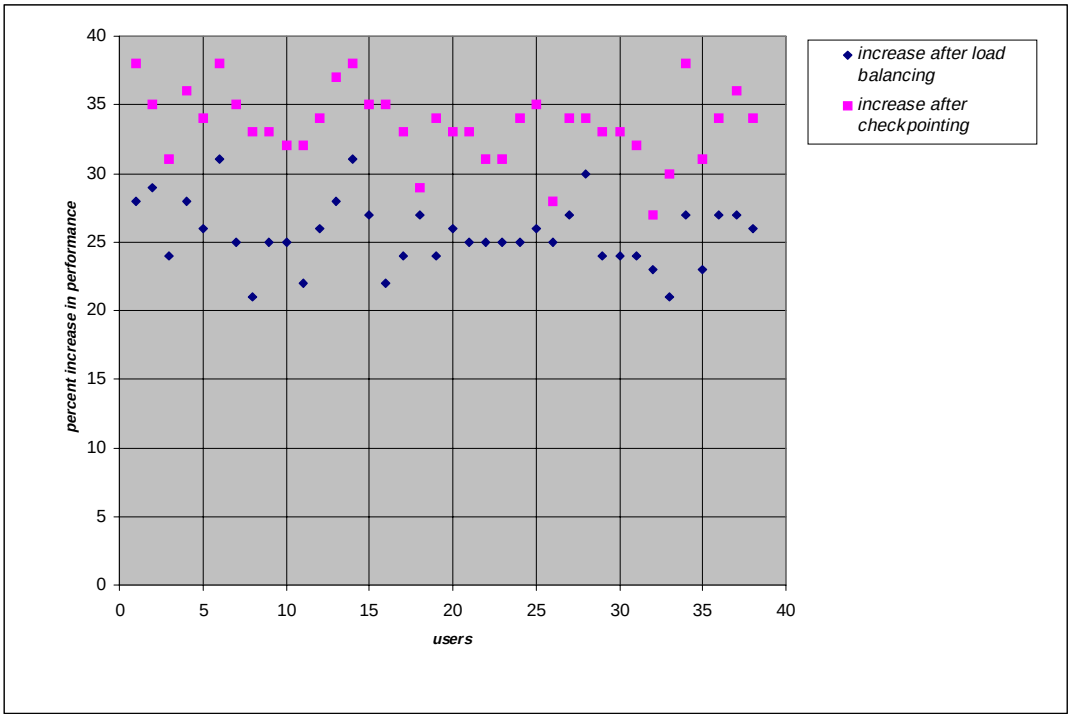

Figure 1: Performance Improvement, from (Karachiwala 1998)

this experiment that visualizing performance parameters enabled users to significantly improve the performance of a distributed Time Warp simulation.

In summary, there are several models for performance visualization, but these models are application dependent. A good visualization model must satisfy the following: provide suitable level of information based on the user level, provide easy understanding and prediction of the execution behavior, be interactive and scalable, and provide information from several different performance perspectives. An approach for achieving these goals is discussed in the next section.

3 PERFORMANCE PARAMETERS FOR TIME WARP SIMULATION

Execution time is a good performance measure of most computer applications; although, in parallel and distributed applications there are often many performance-related parameters involved. Time Warp based simulations have more performance parameters than most other distributed systems, and in addition, measuring these performance parameters is also different. For example, the utilization of computing processors, in most systems, is measured as the ratio between the processors busy time to idle time. This is not exactly the case with Time Warp based systems, where processors can be busy doing *non-productive* or *false* computations. So the real measure of utilization should be based on the amount of time spent doing *correct* work.

Performance-related parameters in distributed Time Warp simulations can be projected in more than one dimension, such as hardware, operating system, communication, simulation kernel and algorithms, and workload. Visualizing these parameters, as the time progresses, will give good insight to system behavior and provide guidance for the change of startup settings to enhance system performance.

A database of different output values, startup settings, and performance parameter values for different runs by different users will be built next; so far we are logging these values for later use. Having Web access to the system will enrich this database with experiments for people with different needs and backgrounds, so the system can get benefit of both good and bad settings and results. In the following sections, the performance parameters are introduced and discussed. We focus on workload, simulation kernel and synchronization algorithms, and communication dimensions, which are the performance parameters of particular significance in Time Warp based systems. Although performance parameters in the hardware and operating system dimensions are truly important, they are not within the scope of this prototype and so are not included in this discussion.

3.1 Workload Dimension

Processor utilization is the first parameter in this dimension and it is defined as the amount of time spent doing correct work. Unbalanced utilization is introduced to the system by either improper load balancing, or a poor communication topology. Furthermore, false utilization, or the amount of time spent doing and undoing incorrect work is a good measure of the amount of under-utilized potential. This parameter requires visualizing the utilization summary of different processor states (busy, false, and idle) as well as the percentage of processing time that was allocated by every processor.

Load balancing is the second parameter in this dimension. If the simulation load is not well balanced, some processors may become overly aggressive and thus decrease the system performance. Any processor that is not loaded sufficiently tends to optimistically process events in the future far ahead of other processors. Therefore, the more it processes future events, the more optimistic it gets, and the more likely that a false event is processed. The false events schedule new false events on other processors. Furthermore, the original false events are eventually rolled back, and their scheduled false messages have to be annihilated. Thus because the slow processors experience more delays and communication overhead, they occasionally cause an avalanche effect of false events which decreases the overall system performance. The amount of optimism can be controlled by blocking the overly optimistic processors, redistributing the load, or choosing a more effective scheduling strategy. Visualizing the amount of optimism or aggressiveness of each processor will help identify the proper action to be made.

3.2 Simulation Kernel and Synchronization Algorithms

To allow rollback, each processor must save its state. State saving overhead directly affects the memory demands on processors and thus affect the performance. There is a tradeoff between the state saving and rollback. Increasing the checkpoint interval reduces the frequency of states saving and consequently the memory consumption and processing time are reduced. However, upon rollback, a process may have to reconstruct a state that was not saved earlier by re-processing events that it has already processed.

Synchronization also affects the memory consumption. Time spent measuring the Global Virtual Time (GVT) and reclaiming the memory (fossil collection) are the time penalties introduced by synchronization. Small synchronization intervals decrease memory consumption and improve algorithm performance at the cost of increased synchronization overheads. These issues require visualizing the cost of synchronization and state saving

overheads, as well as memory consumption and utilization on individual processors.

3.3 Communication Dimension

Communication delays cause another performance bottlenecks in parallel simulation running on networked computers. Processes should be assigned to processors such that they are clustered in groups to avoid excessive external communication. Communication bottlenecks are introduced by either poor topologies, uneven loads, or network hardware limitations. In practice assigning processes involves a tradeoff between good communication topologies and even loads. The solution is to find the balance that is most beneficial to the simulation. Communication issues require visualizing the total network communication traffic as a function of time as well as the intersections among processors and processes through space and time.

4 A SOFTWARE MODEL FOR WEB ACCESS

As stated previously there are two approaches to run applications from the World Wide Web. The first approach is to develop the application to be Web aware using Web development techniques such as: HTML, Java, and CGI tools. The second approach transforms an already existing legacy application, which was not designed for the Web, into a Web-accessible application.

Any suggested model for the distributed legacy systems should consider the following list of problems that have to be overcome safely and efficiently. Running a distributed application from the Internet creates technical difficulties because the Web-user always has a limited environment. Usually, relaxing these limits results in possible system security risks. The third problem involves concurrency; having the system available on the Internet may result in having more than one user trying to run the system concurrently. This greatly affects the system behavior and performance measurements.

4.1 Web-enabling Software Model

The layered model, shown in Figure 2, consists of the following layers: *web browser, authentication layer, environment setup layer, listening daemons and application launcher layer, legacy application layer, and operating system layer*. Each of these layers will be briefly discussed in remainder of this section.

The Web browser layer is the interface layer where users can interact with the application from the Web, and the authentication layer authenticates the user and validates the login name and password. The environment setup layer is responsible for setting up all the required environment to control the concurrency, create run profile, and talk to the

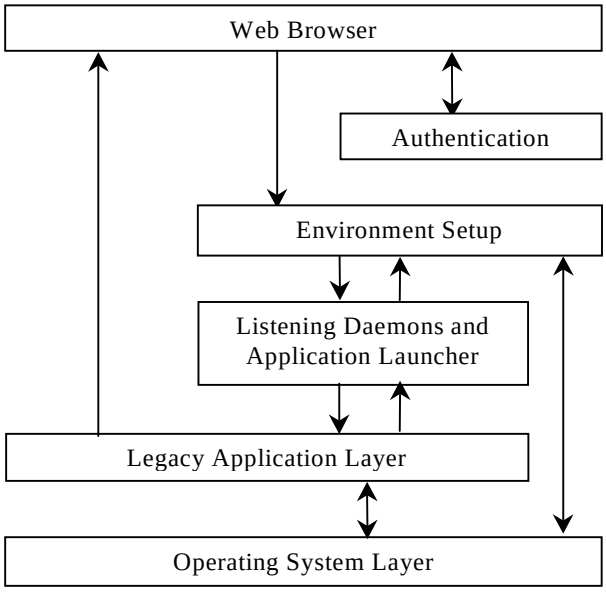

Figure 2: Web-enabling Sofware Model

listening daemons layer. This layer tackles the security hazard problem gently such that the user does not interfere with the legacy application layer directly. Also, it would block new users from using the system and prevent the current user from running more than one instance. This layer receives the user request and helps creating the startup settings. The user request along with the startup settings will be sent to the listening daemons layer.

The listening daemons and application launcher layer exists in all the system machines up and running all the time. Because it is light-weight, it does not affect the system overall performance. This layer receives the user request and the startup settings from the environment setup layer. Once it receives the user request and the startup settings it launches the Distributed Time Warp Simulation. Actually, only the machines specified on the startup settings will launch the simulation program. After the simulation program ends, this layer is responsible for restoring the system settings and getting the system available for other runs by the same user or other users. Finally, the legacy application layer is the application itself.

4.2 Design Limitations of Layered Model

Using the World Wide Web as an interface to run applications that were not developed to be executed from the Web may add some restrictions and limitation to the accessibility. Some of these limitations are related to formatting the output and manipulating the input data. In this work, the Web-enabled version has somewhat different output than the original system. Although the Web-user gets only summarized statistics of the system performance, the detailed output is logged for future reference.

5 WEB-BASED SIMULATION PROTOTYPE

A prototype was developed using the software model introduced above. The user must enter a valid login name and password to be able to use the system. This is passed to the authentication layer. After being authenticated, the environment setup layer establishes the required environment for running the application. The Web user is identified as a regular system user, and thus inherits more privileges than what is normally assigned by the Web server. Once the user gets into the system the system becomes unavailable to other concurrent users. The user is then ready to define the startup settings identifying the machines, the GVT server, processors loading, synchronization and check-point intervals as shown in Figure 3.

Once the user completes this startup step, flags are set on. All the system machines are running, with daemons listening to these flags. Only the machines identified on the startup settings will launch the simulation program with the assigned workload. While the simulation is running, the performance-related parameter values are directed to a shared file system for logging. By comparison, in the original system all the outputs are directed to a centralized visualization server for real time visualization (Graham, Karachiwala, and Elmaghraby 1998).

After the simulation is completed, a summary report is sent to the user. This report includes the number of rollbacks experienced by each machine, and the overall execution time. More detailed performance and parameter values are saved in files that may be accessed by the user. Eventually this data will be incorporated into a master database of simulation performance information as part of our project to develop data mining techniques for learning simulation tuning parameters.

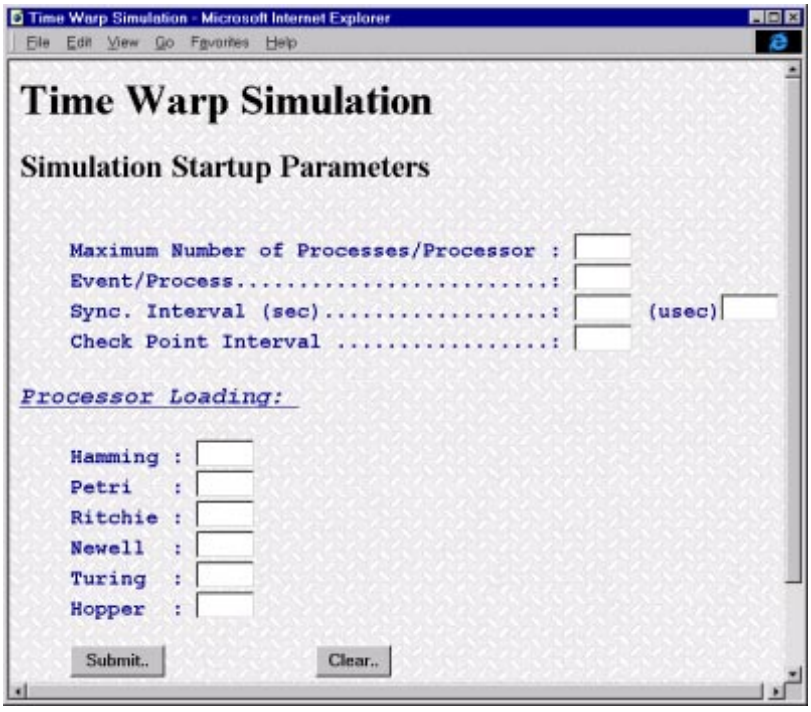

Figure 3: Selection Screen for Simulation Startup Parameters

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have developed and demonstrated a software model that can be used for transforming legacy simulation systems into Web-accessible applications. Problems unique to enabling distributed simulation systems include concurrency, system security, data formatting, and interface compatibility, and are significantly more challenging than enabling of standalone applications. Future work will involve the construction of a performance information database, and investigation of intelligent agents for assisting users in performance tuning activities.

REFERENCES

Bellenot, S., and L. Duty. 1995. Xtracker, A Graphical Tool for Parallel Simulations. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation*, 191-194.

Carothers, C.D., B. Topol, R. Fujimoto, J. T. Stasko and V. Sunderam. 1997. Visualizing Parallel Simulations in

Network Computing Environments: A Case Study. In *Proceedings of the 1997 Winter Simulation Conference*, 110-117.

Das, S.R., J.T. Stasko, and R. Fujimoto. 1992. Animating the Execution of Time Warp Programs. *Proceedings of the 6th Workshop on Parallel and Distributed Simulation*, 195-196.

Evensky, D., A. Gentile, and P. Wyckoff. 1998. A Visualization Tool For Parallel and Distributed Computing Using The Lilith Framework. In *Proceedings of SIGMETRICS Symposium on Parallel and Distributed Tools*, 150.

Fahringer, T. 1995. Estimating and Optimizing Performance for Parallel Programs. *Computer* 28(11):47-56.

Graham, J., I. Karachiwala, A. S. Elmaghraby. 1998. Evaluation of Prototype Visualization for Distributed Simulation. In *Proceedings of 1998 Winter Simulation Conference*, 1469-1477.

Harden, J.C., D.S. Reese, M. B. Evans, S. Kadambi, G. J. Henley, C.E. Hudnall, and C. Alexander. 1995. In Search of Standards-Based Approach to Hybrid Performance Monitoring. In *Proceedings of Parallel and Distributed Technology*, 61-71.

Heath, M.T., and J.A. Etheridge. 1991. Visualizing the Performance of Parallel Programs. *IEEE Software* 8:29-39.

Karachiwala, I. 1998. Visualization the Performance of Distributed Discrete-Event Simulation. Ph.D. Dissertation, University of Louisville, Louisville, Kentucky.

Mller, B.P., M.D. Callaghan, J. M. Cargille, J. K. Hollingsworth. 1995. The Paradyn Parallel Performance Measurement Tool. *Computer* 28(11):37-45.

Reed, D.A., K.A. Shield, W. H. Scullin, L. F. Travera, and C. L. Elford, 1995. Virtual Reality and Parallel System Performance Analysis. *Computer* 28(11):57-67.

Rover, D.,T. Heath, and A.D. Malony. 1995. Parallel Performance Visualization: from Practice to Theory. In *Proceedings of Parallel and Distributed Technology*, 44-60.

Soliman, H., and A.S. Elmaghraby. 1995. Performance Study of A Parallel-Event Simulator. In *Proceedings of ISCA Int'l Conference*, 478-481.

Zhang, K. and G. Marwaha, 1995. Visputer–A graphical Visualization tool for Parallel Programming. *The Computer Journal* 38(8):658-669.

AUTHOR BIOGRAPHIES

ADEL ELMAGHRABY is a professor of Computer Science and Engineering and the Director of the Multimedia Research Lab. He has also held appointments at the Software Engineering Institute - Carnegie-Mellon University, and the University of Wisconsin-Madison. His research contributions and consulting spans the areas of Intelligent Multimedia Systems, Neural Networks, PDCS, Visualization, and Simulation. He is a well-published author, a public speaker, member of editorial boards, and technical reviewer. He has been recognized for his achievements by several professional organizations including a Golden Membership Award by the IEEE Computer Society.

SHERIF ELFAYOUMY is a research assistant and a Ph.D. candidate in computer science and engineering at the University of Louisville. He earned his B.S. degree in electrical engineering from Zagazig University, Egypt, in 1993, and the M.S. degree in computer science from University of Louisville in 1998.

IRFAN KARACHIWALA received his B.S. degree from the University of North Carolina at Greensboro in 1991, and the M.S. and Ph.D. degrees from the University of Louisville in 1994 and 1998 respectively. He is a member of the Association for Computer Machinery and his research interests include parallel and distributed processing, optimistic discrete-event simulation, machine learning, and decision support systems.

JAMES GRAHAM is the Henry Vogt Professor of Computer Science and Engineering at the University of Louisville. He earned the B.S. degree in electrical engineering from the Rose-Hulman Institute of Technology in 1972, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University in 1978 and 1980, respectively. He has worked as a product design engineer for General Motors Corporation, and as a faculty member at Purdue, Rensselaer Polytechnic Institute, and the University of Louisville. He has served as a consultant to a number of companies, including the General Electric Company, and GTE, Inc. He is the editor of two books, and the author or coauthor of over 150 technical publications. He has supervised nine Ph.D. dissertations, and over 30 master's theses. He is a senior member of the Institute of Electrical and Electronic Engineers (IEEE), a member of the Association for Computing Machinery (ACM), and a member of the American Association for Artificial Intelligence (AAAI).

AHMED EAMAM is a research assistant in the Logistic and Distribution Institute, at the University of Louisville. He earned his B.S. degree in Computer Science from Ain Shams University, Egypt, in 1987, and the M.S. degree in computer science from Menoufya University, Egypt, in 1994.

ALAAELDIN SLEEM is a teaching assistant and a Ph.D. student in computer science and engineering at the University of Louisville. He earned his B.S. degree in electrical engineering from Cairo University, Egypt, in 1985, and the M.B.A degree in information technology from Maastricht School of Management, the Netherlands in 1996.