Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

WEB-BASED SIMULATION EXPERIMENTS

Enver Yücesan
Technology Mgmt Area
INSEAD
77305 Fontainebleau Cedex,
FRANCE

Chun Hung Chen
Dept. of Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104, U.S.A

Insup Lee
Dept. of Computer Science
University of Pennsylvania
Philadelphia, PA 19104, U.S.A.

ABSTRACT

The use of the World Wide Web and Java-based mobile code provides new opportunities for distributed simulation. First, the infrastructure provided by the Internet eliminates the need for multiprocessor hardware, making it feasible to distribute simulation models over different hardware platforms through the Internet. Second, "Internet-aware" mobile Java code makes the applications fully portable and reusable. These enabling technologies are already exploited for distributed simulation modeling. This paper reports on the preliminary results of an on-going effort to construct a parallel discrete event simulation support system to distribute simulation experiments over the Internet with a view on simulation optimization. A research prototype for ranking and selection problems is described. The overall project goals are discussed.

1 INTRODUCTION

Man-made systems such as manufacturing and service facilities, computer networks, and transportation systems are typically complex systems that are expensive to build and operate. The design of such systems, typically referred to as "discrete event systems" (DES), therefore require careful performance analysis. In a DES, the system changes state in an asynchronous fashion at (typically) random points in time. For example, a job might be submitted to a computer system, requesting a certain type of service. The system then reacts to this event in order to satisfy the request. While the design concerns may focus on system capacity, i.e., the number of jobs that can be handled in a fixed time period, operational problems may be concerned with the dynamic handling of incoming jobs so as to minimize average response time. Given the inherent complexity of such systems, computer simulation is used for modeling and performance assessment of DES. Computer simulation is a flexible modeling tool in that simulation models can accommodate arbitrary levels of system detail. Simulation is also an attractive analysis tool since simulation experiments are typically less costly than other experimental approaches.

Increased levels of detail in simulation models (e.g., explicit representation of an increasing number of resources such as people, machines, tools, and jobs in manufacturing models), considerably higher levels of activity in the modeled system (e.g., millions of messages to be handled by a telecommunications network), and significantly improved levels of reliability of modeled systems (e.g., a highly reliable ticket reservation computer) are straining the effectiveness of simulation experiments.

Various schemes have been proposed to enhance the effectiveness of simulation experiments. On the modeling side, alternative modeling approaches have been introduced to reduce the inherent complexity of models either by implicitly representing most system entities (Yücesan and Schruben 1993) or by reducing the size of the state space of the underlying model (Kang and Lee 1994). On the analysis side, various approaches have been introduced to improve the statistical efficiency of simulation experiments (Wilson 1984, Glynn and Iglehart 1989, and Chen et al. 1996). Finally, parallel execution algorithms have been developed for run time speed up. There are several approaches to exploiting parallelism in discrete event simulation:

- *Dedicated execution.* This is the approach where dedicated functional units execute specific sequential simulation functions such as random number generation, future events list management, and statistics collection (Comfort 1984).
- *Hierarchical decomposition.* This is the approach where the simulation model is decomposed in a hierarchical fashion so as to allow an event consisting of several sub-events to be processed concurrently (Concepcion 1989).
- *Parallel replication.* This is the approach where several replications of a sequential simulation are executed independently on different processors (Heidelberger 1988).

- *Parallel execution.* This is the execution of a simulation model on a parallel computer by decomposing the simulation model implementation into a set of concurrently executing processes (Misra 1986).

Dedicated execution and hierarchical decomposition exploit the cost advantage of microcomputers and high-bandwidth lines by partitioning the simulation problem and executing the parts in parallel. There are, however, inherent drawbacks in each of the approaches. In dedicated execution, the speedup tends to be very limited, as the management of the future events list becomes the bottleneck. The performance improvement in hierarchical decomposition is largely model-dependent. For example, some queueing network models have been successfully simulated in a distributed fashion by simulating each queueing node on a separate processor. For other models, however, decomposing the model for parallel execution is not trivial. Som and Sargent (1989) propose a conceptual algorithm for hierarchical decomposition. Lin (1992) introduces algorithms for critical path analysis aimed at uncovering inherent parallelism in simulation models.

In parallel replication, each processor must contain sufficient memory to hold the entire simulation model. This is becoming less of a constraint with the rapid advances in hardware. Nevertheless, one important problem remains: if one desires to distribute the experiment, where the results of one replication are used to determine which run to execute next, the execution of the overall experiment becomes once again sequential.

The application of parallel discrete event simulation (PDES) has therefore been limited. Until recently, parallel computers could be found only in research laboratories or large universities. Furthermore, system software to support large-scale distributed simulations remains scarce. Under these circumstances, the simulation community has largely been reluctant to explore the potential gains offered by this technology. In fact, this lack of acceptance by the simulation community has recently triggered a heated discussion about the future of PDES (Fujimoto 1993).

We view the Internet and the web-based technologies as key enablers for conducting large-scale distributed simulation experiments. Our research is aimed at developing a scalable software architecture that would support PDES. This paper describes the research prototype of such a software architecture. This architecture will ultimately be used to investigate the impact of the Internet and web-based technologies on PDES with a particular focus on the design of distributed simulation experiments and simulation optimization. The paper is organized as follows: Section 2 describes the implementation platform. Section 3 provides an example of parallel replication schemes for ranking and selection in which computational effort is allocated dynamically to competing designs in an intelligent fashion (Chen et al 1996). Section 4 briefly describes the on-going work in this domain and the overall goals of the research initiative.

2 IMPLEMENTATION PLATFORM

There exist several Java-based distributed simulation platforms (Page et al. 1997). These platforms, however, focus almost exclusively on distributed modeling or parallel execution of a single simulation replication rather than full-scale experimentation and optimization.

Infrastructure for Distributed Enterprise Simulation, IDES, (Nicol et al. 1997) is a parallel simulation framework for large-scale enterprise simulations. More specifically, IDES is a simulation tool for analyzing complex system models to discover the collective behavior of the system through the interaction of detailed individual sub-model simulations. Such an experiment is typically referred to as *enterprise simulation*. Enterprise simulation necessitates the interoperability of models constructed under different formalisms such as discrete event models and differential equations. Since our focus is on the simulation of DES, the functionality of the proposed software system need not be as far reaching as that of IDES. However, IDES's capability of coordinating the execution of independent sub-models serves as a crucial benchmark for the coordination of distributed simulation experiments.

Multimodeling Object-Oriented Simulation Environment, MOOSE, (Fishwick 1997) is a modeling environment that supports the development of models under various formalisms and the sharing of those models through a web-based environment. MOOSE focuses on object-oriented modeling and promotes the possibility of integrating those models seamlessly with others found in model repositories throughout the Internet. In developing our software architecture, we follow the model management protocols and web interfacing approaches deployed by MOOSE.

We develop a PDES Support System to coordinate the execution of distributed simulation experiments, as illustrated in Figure 1. The research prototype of the PDES Support System possesses the following capabilities:

@ *Site i:* The site, a personal computer or a workstation, is equipped with a web browser which supports the Internet communication protocols (TCP/IP). The communication infrastructure is crucial in managing the messages during the parallel execution of a simulation replication.

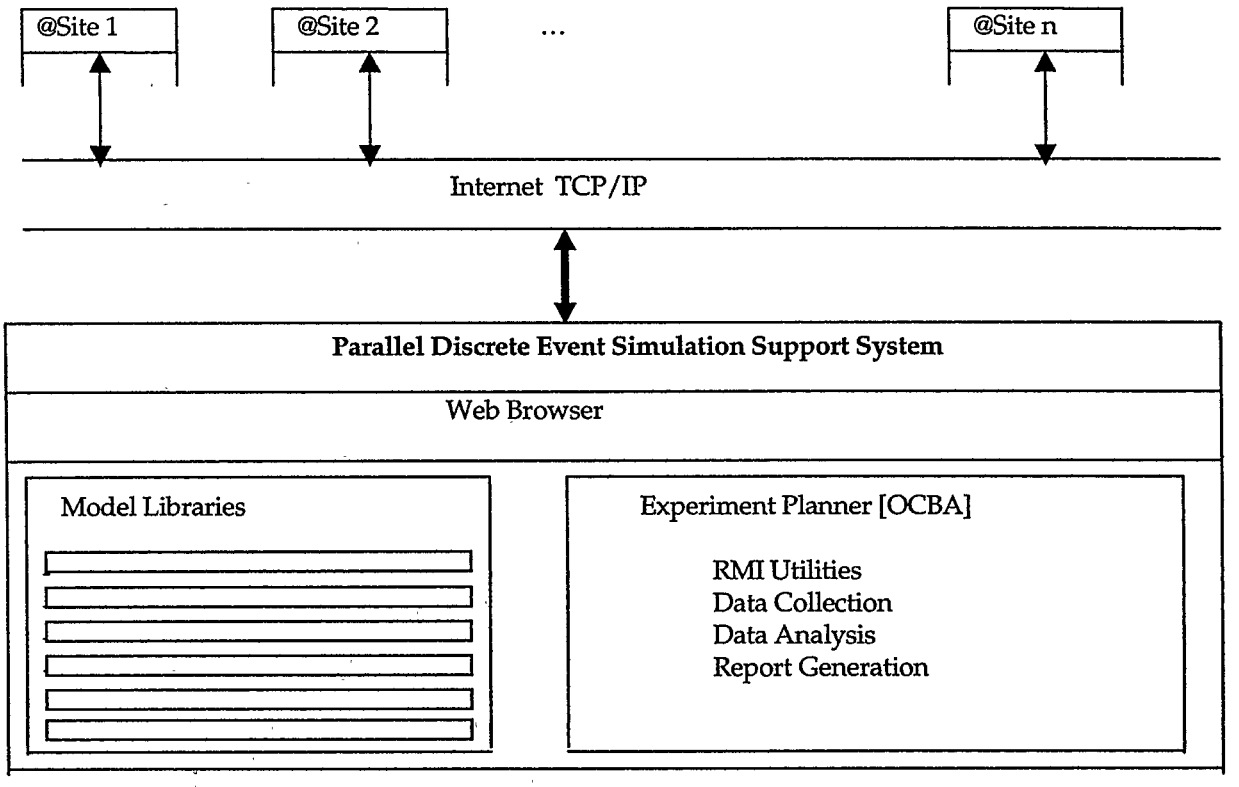

Figure 1: PDES Support System to Coordinate Distributed Experiments

The PDES Support System: This is a central platform on the host machine that plans parallel simulation experiments and dynamically manages their execution. Via the Internet, the system has access to a (large) number of sites, which can be dynamically selected to execute a parallel replication as part of an experimental design. Further technical details on the Experiment Planner (OCBA) are provided in Section 3.

The system also has data collection and analysis capabilities. Output data from each individual processor are collected and analyzed to decide what other replications to run and where to execute the additional replications. In this setting, scalability implies the dynamic deployment of any number of processors as required by the experimental plan.

Finally, the system contains a reporting mechanism which compiles and presents the results of the parallel experiment. The reports will ultimately be in a multimedia format, consisting of tabular and graphical summaries as well as brief animations of the underlying simulation model to depict the dynamic behavior of the system.

We should point out that both the World Wide Web and the Java-based technologies already offer a set of capabilities to facilitate web-based simulation. Key features of web technology include (Ferscha and Richter 1997):

- Transparency of network heterogeneity: Interoperability of different networks is achieved through well-defined, standardized protocols such as HTTP and CGI. The HTTP protocol defines a uniform information transport mechanism, while CGI establishes the interface between a web server and arbitrary programs executing on the same machine.

- Transparency of operating system heterogeneity: The Java virtual machine, a platform-neutral architecture definition, provides a uniform processing environment due to its integration into standard web browsers.

- Transparency of user interface heterogeneity: Along with Java, a class library for user interface programming now exists to support standardized concepts for graphical interfaces. Dynamic retrieval of classes from the net, run-time linking, significantly enriches user interfaces.

The dynamic web environment raises questions on such crucial issues as network congestion, communication reliability, and network security. The immediate objective of our work is to establish the feasibility of Internet-based approaches to distribute simulation experiments with an emphasis on stochastic optimization. Our first concern is therefore capability rather than run-time performance. Issues of portability, maintainability, and conformance to standards are crucial in demonstrating the feasibility of PDES over the Internet. Our second concern is execution

speed, which will be addressed both by designing intelligent distribution algorithms and by taking advantage of emerging techniques aimed at speeding up communication over the Internet.

3 A RESEARCH PROTOTYPE

In addition to distributing model execution and experimentation, we use the parallel environment in distributing simulation optimization algorithms such as rapid learning (Cassandras 1993) and optimal computing budget allocation (Chen et al. 1996). These optimization algorithms have thus far been tested on sequential computing environments, while significant speed up can be obtained through parallel execution.

This paper focuses on *optimal computing budget allocation* (OCBA), which is a scheme aimed at improving the efficiency of ranking and selection procedures in simulation. Chen et al. (1996) formulate the procedure of selecting the best design as an optimization problem, which optimally allocates a computing budget and processors to the designs under evaluation. Denote SQ as the overall simulation quality. Examples of SQ include the mean squared error of the performance measure or the probability of correctly selecting the true best design. Also denote by N_i the number of simulation replications for design i and by k the total number of alternative designs. If the simulation experiment is executed sequentially on a single processor, the budget allocation problem can be expressed as:

$$\max_{N_1,\cdots,N_k} SQ$$

$$\text{s.t. } N_1 + N_2 + \cdots + N_k = B,$$

where B is the given computing budget. Chen et al. (1996) present a solution technique to this budget allocation problem. The underlying ideas are as follows. Intuitively, some inferior designs can be discarded at an early stage of the simulation experiment. As the experiments proceed, other designs can be further ignored when higher simulation accuracy for the remaining designs is obtained. This procedure is repeated until a desired probability of correct selection is achieved. In doing so, less computational effort is spent on simulating inferior designs and, hence, the overall simulation time is reduced. Ideally, we want to optimally choose the number of simulation replications for all designs to minimize the total simulation cost, while obtaining the desired correct selection probability. The question is therefore equivalent to optimally decide which designs will receive additional computing budget for continuing the simulation further or to find an optimal way to reach an optimal design.

Preliminary tests indicate that the proposed technique is significantly faster than the traditional two-stage procedures (Dudewicz and Dalal 1975 and Rinott 1977) for a 10-design G/G/1 queueing system. Furthermore, this technique provides an optimal way to reach an optimal design.

Our research prototype extends the idea of optimal budget allocation to multiple-processor environments. The budget allocation technique can effectively manage limited computing resources, including a limited number of computers and/or computation time. This technique is extended to the multiple-processor environment in the following way: For a single-processor environment, the improvement in the probability of correct selection if design i is simulated is evaluated for all i's. Then the design with maximum estimated correct selection probability improvement is chosen to perform the simulation. A quick approach for the case of having m processors is that we select m design alternatives for further simulation each time. To maximize actual correct selection probability, we select those m designs which have top-m estimated improvement of correct selection probability.

Example: For illustration, a two-station tandem queueing network is considered. The customers arrive to the first station with an interarrival time distributed uniformly. The design alternatives are concerned with the selection of servers for the two stations. Since the performance measure of interest is the average sojourn time, high service rates are desirable. We consider twelve designs, where design #12 provides the fastest service, hence, minimizes the average sojourn time. The user interface of a representative run is shown in Figure 2.

The current implementation, which focuses on feasibility, distributes the execution of the individual threads over a group of available servers. Initially, each design alternative is run for a fixed number of replications. Each replication is executed as an individual thread. Once the execution of a thread is completed, the simulation output is returned to the Experiment Planner in the PDES Support System to update the approximate probability of correct selection (APCS) using a Bayesian framework. This probability is then used to allocate further computation budget (i.e., additional number of replications) to competing design alternatives. Once the additional number of replications is determined, the new threads are assigned to available servers for further execution. The procedure is repeated until the APCS achieves the desired value.

For further gains in efficiency, we are currently moving toward a scheme of predicting the m designs that have the greatest combined improvement of APCS. This approach moves beyond finding the "top m" designs, as we would be analyzing their impact as a group, rather than their individual contribution.

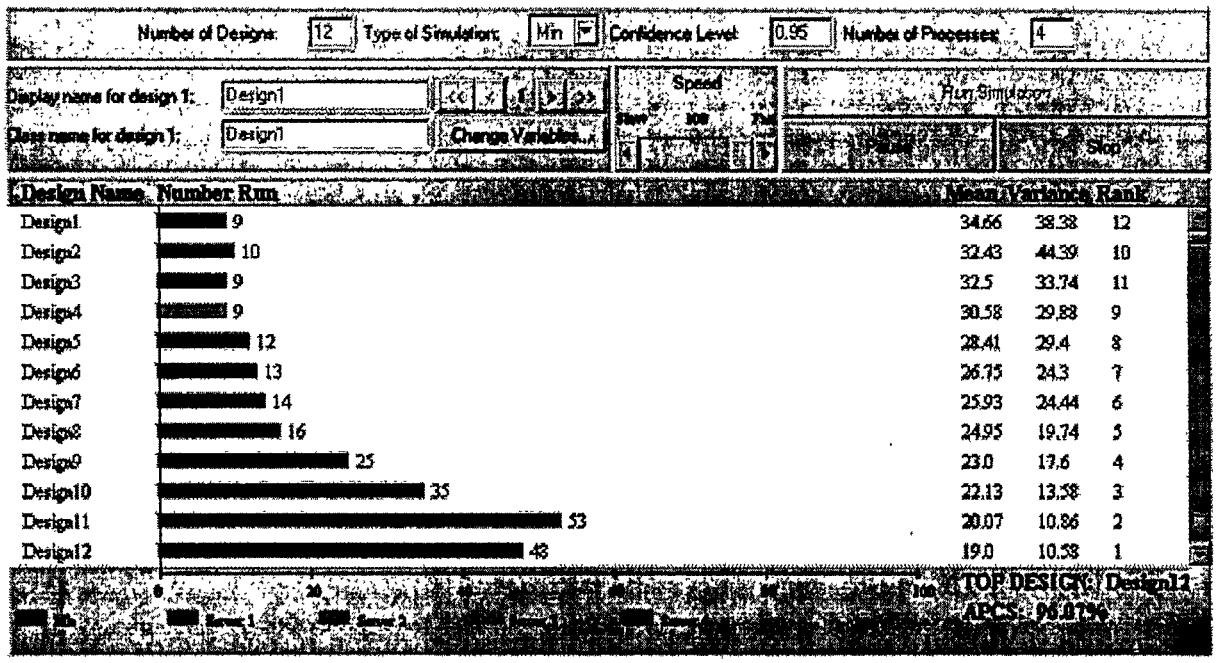

Figure 2: OCBA Implementation for Distributed Ranking and Selection

4 CONCLUDING REMARKS

This paper reports on the preliminary results of an on-going effort to construct a PDES Support System to distribute simulation experiments over the Internet with a view on simulation optimization. A research prototype for a ranking and selection problem is described.

We believe that the use of the World Wide Web and Java-based mobile code as the enabling technologies will provide a breakthrough for distributed simulation. First, the infrastructure provided by the Internet eliminates the need for multiprocessor hardware. It is already feasible to distribute simulation models over different hardware platforms through the Internet. Second, "Internet-aware" mobile Java code makes the applications fully portable and reusable.

The ultimate goal of this research initiative is to develop a scalable software architecture that would support parallel experimentation with discrete event simulation models. This architecture will then be used to explore the impact of the Internet and web-based technologies on parallel discrete event simulation with a particular focus on the design of distributed simulation experiments and simulation optimization.

ACKNOWLEDGMENTS

The authors thank Tagar C. Olson and Sy Damle for their invaluable programming assistance.

REFERENCES

Cassandras, C.G. 1993. *Discrete Event Systems: Modeling and Performance Analysis.* Aksen Associates, Inc.

Chen, C. H., H.C. Chen, and L. Dai. 1996. A gradient approach of smartly allocating computing budget for discrete event simulation, In *Proceedings of the Winter Simulation Conference.* 398-405.

Comfort, J.C. 1984. The simulation of a master-slave event set processor. *Simulation.* 42: 117-124.

Concepcion, A.I. 1989. A hierarchical computer architecture for distributed simulation. *IEEE Transactions on Computing.* C-38: 311-319.

Dudewicz, E. J. and S. R. Dalal. 1975. Allocation of observations in ranking and selection with unequal variances. *Sankhya.* B-37: 28-78.

Ferscha, A. and M. Richter. 1997. Java based conservative distributed simulation. In *Proceedings of the Winter Simulation Conference.* 381-388.

Fishwick, P.A. 1997. Web-based simulation. In *Proceedings of the Winter Simulation Conference.* 100-102.

Fujimoto, R.M. 1993. Parallel discrete event simulation: will the field survive? *ORSA Journal on Computing.* 5: 218-230.

Glynn, P.W. and D.L. Iglehart. 1989. Importance sampling for stochastic simulations, *Management Science.* 35: 1367-1392.

Heidelberger, P. 1988. Discrete event simulations and parallel processing: statistical properties. *SIAM Journal Scientific and Statistical Computing.* 9: 1114-1132.

Kang, I. and I. Lee. 1994. State minimization for concurrent system analysis based on state space exploration. In *Proceedings of the Conference on Computer Assurance.*

Lin, Y.-B. 1992. Parallelism analyzers for parallel discrete event simulation. *ACM Transactions on Modeling and Computer Simulation.* 2: 239-264.

Misra, J. 1986. Distributed discrete-event simulation. *Computing Surveys.* 18: 39-65.

Nicol, D.M., M.M. Johnson, and A.S. Yoshimura. 1997. The IDES framework: a case study in development of a parallel discrete event simulation system. In *Proceedings of the Winter Simulation Conference.* 93-99.

Page, E.H., R.L. Moose, Jr., and S.P. Griffin. 1997. Web-based simulation in SimJava using remote method invocation. In *Proceedings of the Winter Simulation Conference.* 468-474.

Rinott, Y. 1977. On two-stage selection procedures and related probability inequalities. *Communications in Statistics.* A-7: 799-811.

Som, T.K. and R.G. Sargent. 1989. A formal development of event graphs as an aid to structured and efficient simulation programs. *ORSA Journal on Computing.* 1: 107-125.

Wilson, J.R. 1984. Variance reduction techniques for digital simulation. *American Journal of Mathematical and Management Sciences.* 4: 277-312.

Yücesan, E. and L. Schruben. 1993. Modeling paradigms for discrete event simulations. *Operations Research Letters.* 13: 265-276.

AUTHOR BIOGRAPHIES

ENVER YÜCESAN is an Associate Professor of Operations Research at the Technology Management Area at INSEAD in Fontainebleau, France. He holds a BSIE degree from Purdue University, and an MS and a Ph.D. both in OR, from Cornell University. The work described in this paper has been initiated while he was visiting the Department of Systems Engineering at the University of Pennsylvania.

CHUN-HUNG CHEN is an Assistant Professor of Systems Engineering at the University of Pennsylvania, Philadelphia, PA. He received his Ph.D. degree in Simulation and Decision from Harvard University in 1994. His research interests cover a wide range of areas in Monte Carlo simulation, optimal control, stochastic decision processes, ordinal optimization, perturbation analysis, and their applications to manufacturing systems. Dr. Chen won the 1994 Harvard University Eliahu I. Jury Award for the best thesis in the field of control. He is also one of the recipients of the 1992 MasPar Parallel Computer Challenge Award.

INSUP LEE is a Professor at the Department of Computer and Information Science at the University of Pennsylvania, Philadelphia, PA. He received a B.S. degree in mathematics from the University of North Carolina, Chapel Hill, in 1977, and a Ph.D. degree in computer science from the University of Wisconsin, Madison, in 1983. His research interests cover a range of issues in the areas of distributed systems and real-time computing, including operating systems, formal methods, programming languages, and software engineering tools.