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

EVALUATION OF A PROTOTYPE VISUALIZATION FOR DISTRIBUTED SIMULATIONS

James H. Graham
Irfan S. Karachiwala
Adel S. Elmaghraby

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

ABSTRACT

Monitoring and improving the performance of complex distributed simulations can be challenging. Without proper software tools, the process of performance tuning could be complicated and tedious. This paper presents a new approach for visualizing the performance of distributed simulations. In particular, a formal methodology for constructing a visualization is presented, which includes a formal model for visualization and a systematic approach for identifying performance issues. This paper also presents results of a controlled evaluation of a prototype visualization created using the new methodology.

1 INTRODUCTION

Although the primary reason for using parallel simulations is their increased performance potential, this potential is difficult to achieve in practice. In order to achieve significant performance gains, users must analyze and tune different parameters of their simulation. However, due to the volume and complexity of the resulting performance data, understanding the performance of parallel simulations could be rather difficult. There is, thus, a need for performance visualization tools which can make sense of the large volume of data, and present the information in an intelligent way in order to help guide the user in finding performance bottlenecks, and thus improve performance.

Many visualizations that have been built so far simply extract a subset of the data generated by the simulation, aggregate it to reduce the amount of data, and display it using some sort of visual representations. It is difficult for the user to understand how this data relates to performance issues, and even more difficult to determine what factors in the simulation have limited performance. In essence, most visualizations, which have been created, simply display the progress of a simulation, and do not explain why the simulation behaves the way it does. We believe, that creating effective visualizations which are capable of extracting key performance information, requires a more systematic, formal approach that begins with identifying the performance issues at hand, determining effective visual representations for them, extracting and transforming raw data, and displaying them through multiple simulation perspectives in a structured manner.

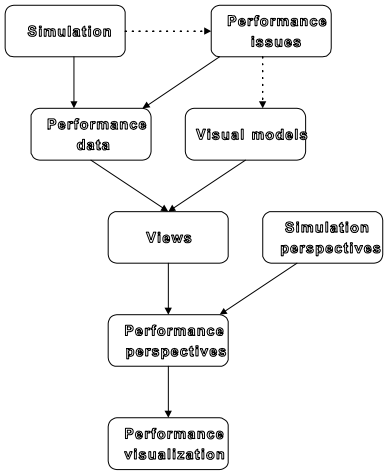
In section two, we discuss a model for creating effective visualizations. In section three, we discuss a prototype performance visualization system for a distributed discrete-event simulation using the Time Warp (Jefferson and Sowizral 1985) protocol on a network of UNIX workstations. Section four presents an evaluation of this visualization prototype, and section five presents conclusions, and future directions.

2 MODEL FOR PERFORMANCE VISUALIZATION

The underlying theme of the model is that visualizations should not be built from raw simulation data alone. To create an effective visualization, one must first understand the issues which could limit the performance of the simulation, and it is these issues that must be visualized. In the context of creating effective visualizations three important question need to be asked: what needs to be displayed? (i.e. what information is relevant to gain insights into performance analysis of a simulation?); how does it need to be displayed? (i.e. how can this information be displayed in a manner which makes sense to the user?); and how can it help in isolating performance problems?

Figure 1 displays an overview of the model for performance visualization. It suggests starting with analyzing the simulation under study, and identifying the issues which can affect its performance. Once these performance issues have been identified, effective visual models must be found that can represent each performance issue clearly.

Only after that, should the appropriate data be extracted from the simulation. In many cases, this data is not readily available from the simulation, and requires additional instrumentation, or intelligent transformations of the raw performance data. The performance issues could then be displayed using views, which are representation of the performance issues using visual models such as space time diagrams, x-y scatter plots, graphs etc. Once multiple views are generated, this information is presented through some context which users can relate to. This is done by separating these views into performance perspectives. A set of perspectives, alongwith some means for letting a user switch between perspectives, go together in making a good visualization.



2.1 Basic Definitions

Distributed simulation involves modeling a real world system on a network of computers. The system being modeled consists of entities and relationships. An entity has attributes, constraints, and transition functions.

Definition 1: A distributed simulation, DS, is defined as:

$DS = \{P, H, M, S_0, \delta, \tau, D\}$ where

- $P =$ the parameters of the simulation
- $H =$ the underlying hardware and interconnections
- $M =$ the mapping function that determines how processes are assigned to the hardware
- $S_0 =$ the initial state
- $\delta = S \rightarrow S$: the state transition function
- $\tau =$ the time advance function
- $D =$ the output data available from the simulation

A view is a representation of a subset of the output data using visual abstraction models. The set A of typical visual models include space-time diagrams, time-series display, histograms, Gnatt charts, Kivat diagrams, bar graphs, scatter-plots, matrix displays, and are found in tools such as ParaGraph (Heath and Etheridge 1991), Paradyn (Miller et al. 1995) and Pablo (Reed et al. 1992).

Definition 2: A View, V, is defined as:

$V = \{D, I, S, T_D, O, A\}$ where

- $D =$ the data available from the simulation
- $I =$ the performance issue to be monitored
- $S =$ the selection criteria for the data
- $T_D = \wp(D) \times I \rightarrow O$: the transformation function that gathers and transforms a subset of the data info a representable form
- $O =$ the range set of resulting performance metrics
- $A =$ the display choice or the visual model used

Definition 3: A Perspective, P, is defined as a set of related views. That is, $P \in \wp(V)$ where $\wp$ denotes the power set operator.

A visualization is an interface which displays one or more perspectives at a time with the ability to switch between perspectives.

Definition 4: A Visualization, W, is defined as:

$W = \{Q, C\}$ where

- Q : $\{P_i \mid P_i \in \wp(V)\}$, where P_i is the perspectives chosen to be displayed
- C : $Q \rightarrow \{0, 1\}$, a mechanism which lets the user select between different perspectives

3 PROTOTYPE VISUALIZATION

3.1 The Time Warp Simulation

The system simulated is a discrete event simulation of a hypothetical closed network of '*n*' airports (nodes). Each node is initially populated with a fixed number of planes ('*k*') waiting to be serviced and take off. Planes arrive at a node, are serviced (service time), take off, and are scheduled to arrive at another airport based on a shifted exponential distribution (inter-arrival time). Thus, the number of planes in the system always remains the same. Each node has a fixed number of inputs (f_{in}) and outputs (f_{out}). Logical processes were used to represent nodes and distributed among the processors responsible for the simulation. Planes were modeled as messages that were passed between processes. All communication between processors (message passing) is done using TCP/IP sockets. Processes that belong to the same processor communicate using local memory. Synchronization is performed by an independent processor which computes GVT values using Samadi's (1985) message acknowledgment algorithm using a fixed synchronization interval.

3.2 Creating the Visualization

The model presented in Section 2 dictates that we follow specific guidelines in order to create a successful visualization. The first step is determining what needs to be visualized (i.e., what characterizes performance of the simulation). Once these issues are identified, the next step is to determine how they are to be visualized.

An important issue which limits performance of most parallel programs is poor utilization. In parallel computing, utilization is measured by the amount of busy time or concurrency on the processors of the system. Utilization for these systems is simply the ratio of processor busy verses idle times. Clearly, this is not true of optimistic simulations as most of the otherwise unused cycles of processors are spent optimistically processing events in the hope that they might be true. A novel, and more effective, approach is to use the amount of time spent doing correct work as a measure of effective processor utilization. An imbalance in the effective utilization of processors is caused 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 potential that is under-utilized.

Another major problem with optimistic simulations is that if the simulation load on processors is not well-balanced or the scheduling policy is not well-chosen, some processes may tend to get overly aggressive. This aggressive behavior tends to significantly decrease the overall performance of Time Warp. The amount of optimism could be controlled by either blocking processes which get overly optimistic, redistributing some load from slower processors to overly optimistic ones, or choosing a more efficient scheduling strategy.

Communication delays cause yet other performance bottleneck in parallel simulations running on network computing environments. Processes must be assigned to processors such that they are clustered in groups which avoid excessive external communication. Unfortunately, because of the nature of most parallel simulations, this is either not feasible, or introduces uneven loads on processors. Communication bottlenecks are caused due to poor topologies, uneven loads, or simply due to hardware (network) limitations.

State saving and synchronization overheads are yet other problems which limit the performance potential of optimistic simulations. Both these issues deal with memory utilization on processors, and cause additional overheads.

Another performance issue is that of rollback overhead. Rollback costs include all the overhead work that is not associated with the normal forward progress of the simulation. This includes the time taken for processing false events and of undoing them. Excessive rollbacks are caused due to uneven loads, poor topologies, or badly chosen scheduling policies.

Other performance issues include the amount of concurrency in the system, and the geometry of computation. Concurrency is mostly affected by the scheduling policies, and could be visualized using concurrency profiles of processors states. Insights into the geometry of computations could be obtained by visualizing the fan-in and fan-out of communication messages on processors.

3.3 The Resulting Performance Visualization

The resulting visualization is comprised of three distinct components: the visualization server, the performance displays and a static tool to modify simulation parameters.

3.3.1 The Visualization Server

The visualization server was build to collect, analyze, and transform the required performance data from all the processes involved in the simulation as well as the GVT server. Each processor communicates with the server using TCP/IP sockets. Performance data is collected from the processors, transformed such that intelligent information can be extracted, and displayed using the performance views. The server is also responsible for both starting/stopping the simulation as well as controlling the multiple performance displays. Additionally, the server also provides a complete online help facility.

3.3.2 The Performance Displays

The resulting views obtained from section 3.2 were grouped together into related categories such as utilization, communication, memory, etc. to create performance displays. These displays are solely responsible for the intelligent presentation of the transformed data. Though illustrations cannot convey the dynamic nature of these display, these are nevertheless shown as snapshots.

3.3.2.1 The Processor Utilization Display

The processor utilization display (Figure 2), presents both physical and application level information. It shows how effectively processors are being utilized and how evenly the

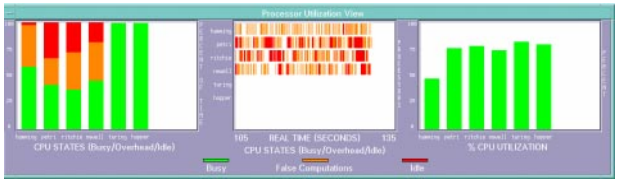


Figure 2: The Processor Utilization Display

computational work is distributed among them. It is comprised of the following three views:

Utilization Summary - This view (left view in figure 2) shows the percent of time each processor spends in each state (Busy, False {overhead}, and Idle). Busy time is the time taken for work that is eventually committed divided by the sampling interval, while overhead is the amount of time taken for doing and undoing erroneous work divided by the sampling interval. The color scheme is borrowed from traffic signals: green for busy, orange for overhead, and red for idle. Typically, the idle times and overheads of processors should be kept to a minimum. An imbalance in overheads/idle times implies that the simulation load is not well balanced among the processors.

Concurrency Profile (Utilization History) - This view (middle view in figure 2) uses a Gantt chart to depict the activity of individual processors. The color of each vertical bar indicates the busy/overhead/idle status of the corresponding processor as a function of time. An advantage of this view is that it shows the fine detail which can sometimes be missed in the utilization summary.

CPU Utilization- This view (right view in figure 2) displays the percentage of CPU time that is allocated to the simulation. Clearly, the larger this value, the more CPU time consumed by the simulation process. This view has the ability to account for both the simulation load as well as external loads on processors.

3.3.2.2 The Communication Utilization Display

These views provide information pertaining to interprocess and interprocessor communication. These displays are helpful in determining communication frequency, volume, overall pattern, and whether there is congestion in the message queues. Typically, processes that communicate frequently should be clustered together and assigned to the same processor to reduce the amount of communication traffic between processors.

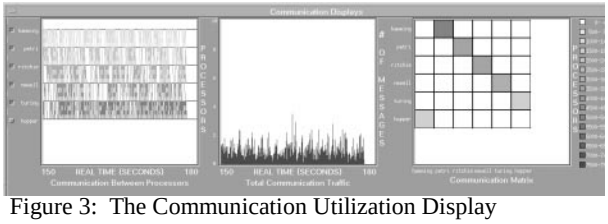


Figure 3: The Communication Utilization Display

Communication Between Processors - This view (left view in figure 3) uses a space-time diagram to depict communication interactions among processors through space and time. Messages between processors are depicted using slanted lines between the sending and receiving processor activity lines, indicating the times at which each message was sent and received. These sending and receiving times are from logical process to logical process (not just the physical transmission time), and hence the slopes of the resulting lines gives a good indication of the delay between a message being produced on one processor and received on the other. Individual processors or any combination of them could be selected at a time to display communication characteristics for that set.

Total Communication Traffic - This view (middle view in figure 3) displays the total amount of interprocessor communication traffic in the network over time, and is used along with the Communication Matrix view to reduce overall external communication.

Communication Matrix - This is a dual view which gives insight into both the communication patterns between the active hosts as well as processes involved in the simulation (right view in figure 3). Views can be switched by simply clicking anywhere within the window. These views can be used to help redistribute processes in order to keep the external communication to a minimum. Each square represents communication volume between two processes/ processors. Processes are grouped by the processors they reside on. Darker shades indicate larger volumes.

3.3.2.3 Synchronization and Memory Utilization

This display is mainly used to adjust the checkpoint and synchronization intervals. Memory and overhead information for both these issues is presented here. It is comprised of the following views:

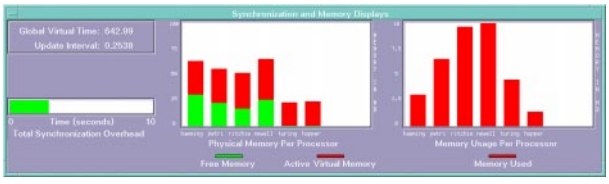


Figure 4: Synchronization and Memory Utilization

Synchronization Interval - This view (top left view in figure 4) uses simple textual widgets to indicate the current known value for GVT and the synchronization interval.

Synchronization Overhead - This view (middle left view in figure 4) indicates the total amount of overhead for both computing GVT as well as fossil collection.

State Saving Overhead - This view (bottom left view in figure 4) indicates the total amount of overhead for allocating memory, checkpointing, and coasting forward.

Physical Memory per Processor - This view (middle view in figure 4) is used to determine the amount of memory that is available on processors. The vertical

green bars represent the amount of free RAM, while the vertical red bars indicate the amount of virtual RAM currently utilized.

Memory Usage per Processor - This is the total amount of memory that is currently being utilized by every processor in the simulation (right view in figure 4). This includes memory used for state saving, maintaining input queues and negative copies of messages, as well as additional memories used by objects, variables etc.

3.3.2.4 Simulation Perspective

This display is used to evaluate overall simulation performance. Performance information relates not just to the speed to computation, but to the efficiency and effectiveness of the simulation in utilizing the many shared resources and services within the distributed environment. The display is comprised of the following two views:

Simulation Effectiveness - This view (left window in figure 5) displays the overall effectiveness of the simulation using simple textual values. Simulation progress rate, event rates and effective processor

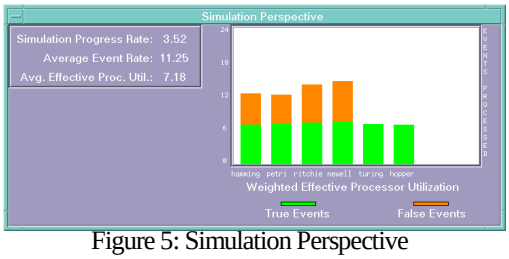


Figure 5: Simulation Perspective

utilization rates are displayed here.

Simulation Potential - This view (right window in figure 5) is used to determine the work potential of processors. Each vertical bar represents the average capacity for work for every process on a single processor (i.e. the average number of events each process is capable of processing per second). The green portion represents the average amount of work that was actually committed (realized potential), while the orange portion represents the average amount of work that was erroneous (unrealized potential).

3.3.2.5 Rollback Perspective

The views presented in this perspective, display both the amount of optimism in processors as well as the number of false events processes due to their aggressiveness.

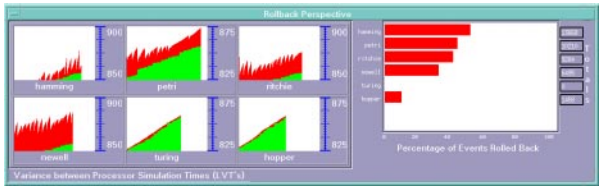


Figure 6: Rollback Perspective

Amount of Optimism (Variance between processor LVT's) - This view displays the amount of optimism in processors, and the average rate of processing events (left view in figure 6). Green areas indicate the currently known value of GVT, while the red areas indicate the timestamp of the last processed event (LVT). The greater this difference, the greater the amount of optimism on the processor. Additionally, the slope indicates event rates. The greater the slope, the slower the rate of processing events.

Percentage of Events Rolled Back - This view indicates the percentage of events processed which are rolled back (right view in figure 6). The depth of individual rollbacks could be obtained from the previous view.

3.3.3 The Simulation Parameter Modification Tool

The third component of the visualization is a static tool which allows users to setup and modify simulation parameters. The tool was initially built to provide users with easy interface for static load balancing. Users are able to set up the simulation by using the tools drag and drop interface to assign host processors responsible for running the simulation protocol, the visualization, and the GVT server. Loads and communication topologies can be re-balanced by simply moving processes from one processor to another. In addition, it allows users to modify many simulation parameters such as the checkpoint and synchronization intervals.

4 EVALUATION OF VISUALIZATION

In order to evaluate the effectiveness and ability of the visualization prototype to improve performance, a carefully controlled experimental study was conducted The visualization was evaluated by thirty-eight users with varied levels of familiarity with the concepts of parallel processing and distributed simulation. Users were asked to use the visualization tool to improve the performance of a Time Warp based optimistic simulation. Results were collected and statistically analyzed.

4.1 User Testing and Evaluation

In order to keep the case study simple, users were asked to identify and tune parameters associated with just two bottlenecks. The issues chosen were rollback overheads

and state saving overheads. For rollback overheads, users were asked to study the aggressiveness of processors and reduce the excessive aggressiveness of individual processors by re-distributing loads. For state saving overheads, users were simply asked to find a good checkpoint interval which decreased memory usage but did not significantly increase the cost of coast-forward overhead. In all cases, the simulation was run for 1000 units of simulation time.

4.1.1 The Case Study

The system simulated was the one described in Section 3.1. A network of 7 heterogenous HP-UX workstations interconnected by a 10Mbit/sec ethernet were chosen for this simulation. One 9000/735 (at 99MHz w/ 256Mb RAM), four 9000/715's (at 80MHz w/ 128Mb RAM), and two 9000/712's (at 60MHz w/ 32Mb RAM). Six of the machines were assigned to the simulation while one 715 was assigned to the visualization as well as to the GVT server. The system was assumed to have 48 nodes which were distributed evenly amongst the processors.

4.1.2 Protocol for User Evaluation

The users chosen to test and evaluate the visualization were mostly graduate students in the Engineering Math and Computer Science department at the University of Louisville. All were currently taking a graduate level course in discrete-event simulation, and were representative of the skill levels of the computer professionals who might actually need a visualization system while working with a distributed discrete event simulation in their professional practice. The user evaluations were conducted in five phases: introduction, familiarization, testing, evaluation, and statistical analysis.

Phase 1: (Introductory phase) - This phase was used to introduce users to the main ideas in parallel simulation, Time Warp, and its potential performance problems. This was done using a series of three lectures, and additional reading materials, which included a paper on "Parallel Discrete Event Simulation" by Fujimoto (1990), as well as a handout which briefly introduced the concepts of distributed simulation and Time Warp, and also explained the case study and the physical hardware used for the visualization. The handout also contained an explanation of some of the performance bottlenecks and tunable parameters in Time Warp, as well as the initial parameters used in the case study.

Phase 2: (Familiarization phase) - This phase was conducted individually with each user just before his/her evaluation. Users were given a hands-on introduction to the performance visualization with an emphasis on the kinds of performance bottlenecks viewed through each view. The static simulation-tuning tool and the online help facility were also introduced.

Phase 3: (Testing Phase) - In this phase, users were asked to improve upon the simulation performance by identifying performance faults and tweaking parameters for the two performance issues mentioned above. The only help users were allowed to receive was through the performance help system built into the visualization.

Phase 4: (Evaluation Phase) - In this phase, users were asked to fill out a brief questionnaire of about 18 questions on a scale of 0-10. The questions were grouped into categories such that they would allow analysis of the visualization's ability to improve performance, its effectiveness, and its ease of use. The goal here was to get information into how users perceived the visualization.

Phase 5: (Statistical Analysis Phase) - During this phase, the effectiveness of the performance visualization tool was determined by analyzing the performance improvement data from phase 3 and questionnaires obtained from phase 4. These results are presented in the next section.

4.2 An Analysis of the Results

A total of thirty eight users individually tested and evaluated the visualization. The entire evaluation process lasted about two and a half weeks. On the average, each user took approximately 30 minutes to reduce performance problems caused due to excessive aggressiveness and improper load balancing, and approximately 15 minutes to determine a satisfactory checkpoint interval.

4.2.1 Performance Improvement Using Visualization

The questionnaires were first analyzed to determine the visualizations ability to provide a significant increase in performance. Users were asked to record the execution times for the simulation (for 1000 units of simulation time) both before, and after, reducing the two performance problems.

The average execution time for users before tuning the simulation was 232.8 seconds. After using the visualization to modify the load and checkpoint interval, the average simulation run times were decreased to 171.8 and 155.1 seconds respectively. Standard deviations for the resulting run times were 5.6 and 5.9 seconds respectively. Figure 7 displays the speedup achieved by users both before and after tuning simulation parameters. An average increase in speedup of about 1.62 was achieved (since users were allowed to use at most six processors, the maximum possible speedup is six).

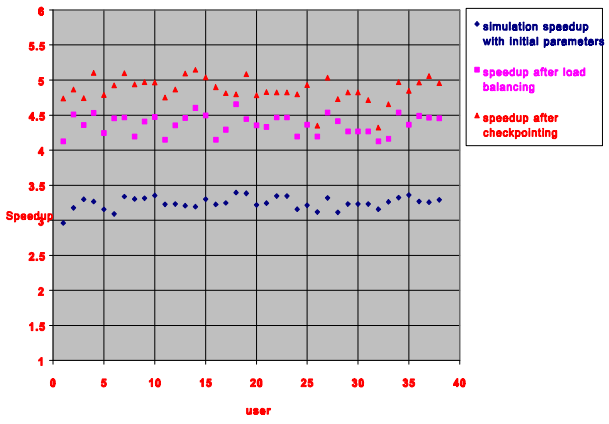


Figure 7: Simulation Speedup for Users Before and After Modifying Simulation Parameters

One additional issue studied was the rate of performance increase shown by users. In order to study this issue, intermediate run times were automatically recorded every time a user modified parameters and re-ran the simulation. Figure 8 shows the average intermediate run rates for users while decreasing performance problems. It is clear from the figure that though users modified parameters and re-ran the simulation a few times before they were satisfied with their performance increase, the visualization was very effective in helping users identify bottlenecks and quickly achieved reasonable performance gains.

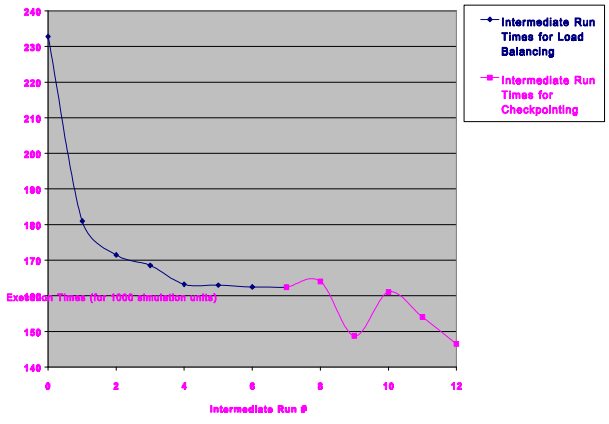


Figure 8: Average intermediate run times while decreasing performance problems

4.2.2 Subjective Evaluation of Visualization Features

As explained previously, each user filled out an evaluation questionnaire at the end of the experiment. This form, contained six questions regarding features of the visualization and four questions regarding user perceptions of the effectiveness of the visualization.

The features evaluated were performance displays, layouts, views, controls, colors, and the setup tool. One question required a yes/no answer, and asked if the performance displays were helpful in contributing towards a performance increase. All thirty-eight users answered in the positive.

Histograms for the responses to the questions regarding features are given in Figures 9 through 13. One question asked whether the performance displays and their layouts were easy to understand. About 90% of the users indicated that they were relatively easy. The average response was 1.7 with a standard deviation of 1.2. Another question asked if it was easy to switch between performance views. The majority of users (about 95%) gave high ratings. The next question asked if the controls used to set up the performance displays and start the simulation were difficult to use. Only one user indicated that the controls were hard.

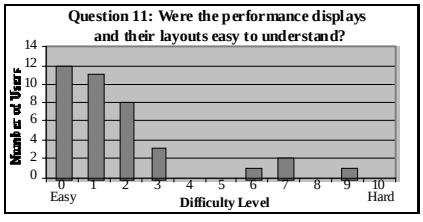


Figure 9: User Evaluations of Performance Displays

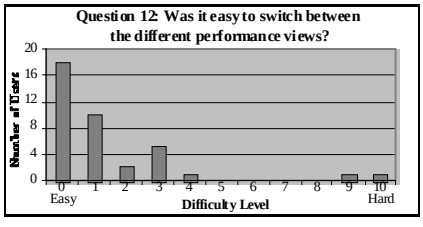


Figure 10: User Evaluations of Performance Views

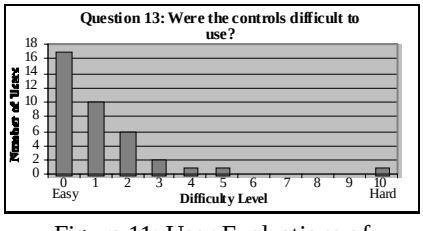


Figure 11: User Evaluations of Difficulty of Controls

Another question asked if the colors in the performance displays were effective. All thirty-eight users indicated responses of near-excellent. The last question asked if the static simulation parameter modification tool made it

easy to modify and tune simulation parameters. Once again, about 90% of the users gave very positive responses.

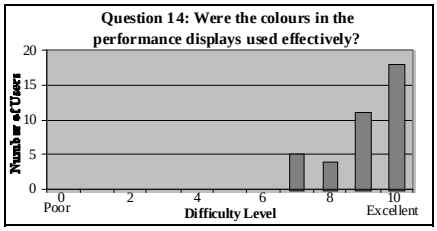


Figure 12: User Evaluations of Use of Colors

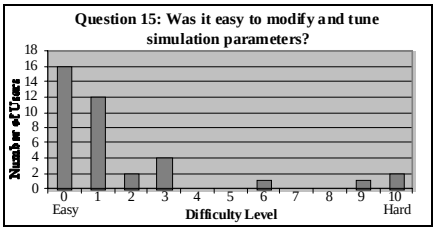


Figure 13: User Evaluations of Setup Tool

Four additional questions regarded user perception about the visualization and its overall effectiveness. One question asked users if they felt that their performance increase was significant enough to justify the use of the performance visualization. All thirty-eight users responded positively. Histograms for the responses to the other three questions are given in Figures 14 through 16.

One question asked users to rate the visualizations ability to help increase performance. About two-thirds of the users rated the visualizations ability as near excellent. The average response was 8.6 with a standard deviation of 1.4. Another question asked users to rate the ability of the visualization to identify performance bottlenecks. About 90 percent of the responses were near excellent. The last question asked users to rate the level of competence that they thought was required for the visualization. The average result was 2.1 with a standard deviation of 1.8, indicating that users felt that the visualization required a nominal level of familiarity with distributed simulation.

4.3 Summary of Evaluation Results

The testing and evaluations done in the previous sections provided valuable information about the effectiveness of the visualization prototype. Using the visualization, users were able to significantly improve upon the performance of a distributed discrete event simulation. The prototype allowed users to quickly locate and isolate performance bottlenecks, make effective changes in system parameters,

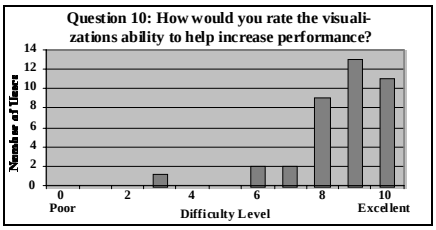


Figure 14: User Perceptions of Ability to Help Increase Performance

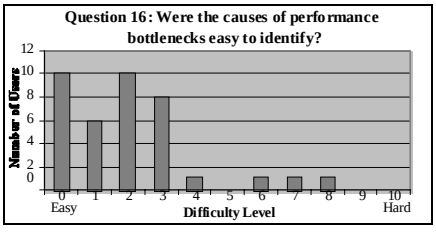


Figure 15: User Perceptions of Ability to Identify Performance Bottlenecks

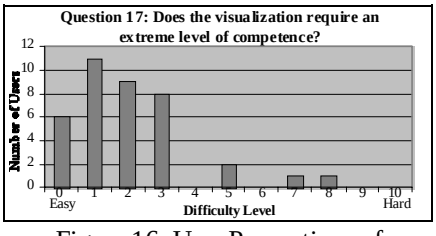


Figure 16: User Perceptions of Competence Level Required

and use available computing resources more effectively as evident by the speedup results.

Finally, it can be concluded that the performance visualization has significant potential as a teaching tool to illustrate how different parallel simulation methods execute as well as to study the effects of different performance parameters.

5 CONCLUSIONS AND DIRECTIONS

Visualizing the progress of a distributed simulation may be easy, but visualizing its performance is not. However, not only is visualizing performance hard, but the performance data needed may not be easily available from the simulation. In this paper, we have developed and presented a structured model for visualizing the performance of a distributed simulation. Our approach stresses the importance of visualizing performance issues rather than visualizing raw data or simulation results. Using this approach, performance bottlenecks can be easily identified, which may aid in tuning the simulation and achieving increased performance. The experimental

evaluation validated the general capabilities of the visualization approach.

The next logical step in enhancing the visualization is to add "intelligent" support to assist the user in utilizing the features of the visualization, understanding the information in the views, and taking action to enhance performance. We envisage a combination of rule-based and model-based approaches being used to achieve this enhancement.

REFERENCES

Jefferson, D., and H. Sowizral. 1985. Fast Concurrent Simulation Using the Time Warp Mechanism. *Distributed Simulation* 15 (2): 63-69.

Heath, M. T., and J. A. Etheridge. 1991. Visualizing the Performance of Parallel Programs. *IEEE Software*. 8(5): 29-39.

Miller, B. P., M. D. Callaghan, J. M. Cargille, J. K. Hollingsworth, R. B. Irvin, K. L. Karavanic, K. Kunchithapadam and T. Newhall. 1995. The Paradyn Parallel Performance Measurement Tool. *Computer*. 28 (11): 37-45.

Reed, D. A., R. A. Aydt, T. M. Madhyastha, R. J. Noe, K. A. Shields and B. W. Schawartz. 1992. The Pablo Performance Analysis Environment. Department of Computer Science, University of Illinois.

Samadi, B. Distributed Simulation Algorithms and Performance Analysis. 1985. Ph.D. Dissertation, University of California, Los Angeles.

Fujimoto, R. M. 1990. Parallel Discrete Event Simulation. Communications of the ACM., 33 (10): 30-53.

AUTHOR BIOGRAPHIES

JAMES H. GRAHAM is currently the Henry Vogt Professor of Computer Science and Engineering at the University of Louisville. He has previously served as a faculty member at Rensselaer Polytechnic Institute, and as a product engineer with General Motors Corporation. He received his B.S. degree from Rose-Hulman Institute of Technology, and the M.S. and Ph.D. degrees from Purdue University. He has organized two IEEE workshops on special computer architectures for robotics and automation, being the program chair for the 1997 International Conference on Intelligent Systems. He is the editor of the books Computer Architectures for Robotics and Automation and Safety, Reliability and Human Factors in Robotic Systems.

IRFAN S. 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 Computing Machinery and his research interests include parallel and distributed processing, optimistic discrete-event simulation, machine learning and decision support systems.

ADEL S. ELMAGHRABY is a Professor of Computer Science and Engineering at the University of Louisville and is the Director of the Multimedia Research Lab. He received his Ph.D. from the University of Wisconsin-Madison and has held appointments as a visiting scientist at the Software Engineering Institute at Carnegie-Mellon University, as a Fulbright Scholar at Qatar University, as a visiting professor at the American University in Cairo and as a visiting researcher at the University of Wisconsin. He is the founding editor of the ACM/IEEE Joint Simulation Newsletter and is an associate editor of SCS SIMULATION. He is a member of several conference steering committees and program committees and was program co-chair for the ISCA 1995 International PDCS Conference.