Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

SCHEDULING TIME WARP PROCESSES USING ADAPTIVE CONTROL TECHNIQUES *

Avinash C. Palaniswamy
Motorola
Schaumberg, IL 60196
aap008@email.mot.com

Philip A. Wilsey
Center for Digital Systems Engineering
Cincinnati, Ohio 45221-0030
phil.wilsey@uc.edu

ABSTRACT

Optimistic techniques using the Time Warp mechanism has shown great promise in speeding up Parallel Discrete Event Simulations. However, Time Warp has been plagued by problems such as excessive rollbacks, memory usage, and wasted lookahead computation. In particular, excessive rollbacks can result in a deterioration of the advancement of the simulation. Consequently, techniques to properly control the optimism in Time Warp are needed to alleviate non-productive lookahead. This paper presents a Logical Process (LP) scheduling algorithm based on concepts from adaptive control theory. In particular, we develop a performance index called *useful work*. The *useful work* parameter represents the amount of productive work done by the process and it is used by our scheduling algorithm to aid in LP scheduling decisions. The scheduling algorithm presented in this paper is compared with the widely used smallest timestamp first scheduling algorithm to show its usefulness in a Time Warp simulation.

1 INTRODUCTION

The time warp mechanism is a method for synchronizing a parallel discrete event driven simulator. Under time warp, a physical system is decomposed into a set of Logical Processes (LPs) that operate as a set of asynchronously communicating discrete event processes. Each LP maintains a local simulation time (called the *Local Virtual Time*, or LVT) and communicates with other LPs through timestamped messages. All LPs execute optimistically, without regard to the state of other LPs and may occasionally be required to rollback to process a late arriving event (called a straggler event).

Clearly, time warp simulation can be successful only if the cost and the number of rollbacks are kept to a minimum. If the cost and the number of rollbacks are minimized in all processes in the simulation, the throughput of the simulator is maximized. A number of techniques can be employed to reduce the cost and number of rollbacks. For example, rollback costs can be reduced with hardware assist (Fujimoto et al., 1992) or with improved algorithms such as lazy cancellation (West, 1988). Likewise, the number of rollbacks can be reduced (i) with efficient process to processor bindings (Davoren, 1989), (ii) by restricting lookahead distance (Matsumoto and Taki, 1993; Palaniswamy and Wilsey, 1993a), or with time warp specific scheduling algorithms (Lin and Lazowska, 1991; Reiher et al., 1989).

This paper addresses the problem of building a time warp specific scheduling algorithm. In particular, we examine the construction of a process scheduler that attempts to favor LPs whose past history of behavior indicates higher productivity than other LPs. The chief difficulty in building an effective scheduling technique is the derivation of a measure that accurately reflects LPs whose behavior is "more productive" than others. In the simple case, one might mistakenly believe that simple scheduling to favor the LP with the lowest LVT value would be a suitable technique. Likewise favoring LPs with lower rollback activity does not necessarily produce the best schedule. Thus, a detailed analysis of time warp must be conducted to derive a measure for "productivity" that can be used in making, among other things (Palaniswamy, 1994), scheduling decisions.

This paper presents a new concept called *useful work*, which parameterizes the work done and overhead incurred for each time warp process in a simulation. The time warp simulation is viewed as a dynamic system of communicating processes, and simple adaptive control techniques are applied to maxi-

*This work is partially supported by the Defense Advanced Research Projects Agency under order number 7056, monitored by the Federal Bureau of Investigation under contract numbers J-FBI-89-094 and J-FBI-93-116.

mize throughput. In particular, we design the time warp simulation as an open loop adaptive control system based on the *Model Reference Adaptive System* (Kokotovic, 1991) model. Using the Index of Performance *useful work*, LPs are scheduled to maximize the throughput of each process and also, overall system performance.

The rest of the paper is organized as follows: Section 2 presents some background work on other techniques to stabilize time warp simulators. Section 3 presents some background work in adaptive control techniques. Section 4 introduces the concept of *useful work* and describes time warp as a dynamic system of processes that can be adaptively controlled. Section 5 presents the new scheduling algorithm. In addition, a brief description of the smallest timestamp first scheduling algorithm is presented. Section 6 presents some performance results using the new scheduling policy. Finally, Section 7 provides some concluding remarks.

2 BACKGROUND WORK

Bounded Time Warp (Turner and Xu, 1992) and *window based throttling* (Reiher et al., 1989) restrict optimistic processing by allowing each logical process to execute within a bounded window. The window is static in size and is globally the same for all processes in the simulation. While this technique constrains the undesirable lookahead computation, it also limits the optimism of processes that were achieving good lookahead. This optimization is difficult to use because different simulation models (within the same application domain) may require different window sizes, for optimal performance. Furthermore, an incorrectly sized bounding window can dramatically decrease performance. At present, no known techniques exist for effectively determining a static size for the bounding window.

Adaptive Time Warp (Ball and Hoyt, 1990) introduces conservatism into the system by allowing a process which experiences a large number of rollbacks to block for a particular time duration called *BW*. BW is varied depending on the time spent in blocked and faulted states (duration of wasted lookahead computation). However, determining a faulted curve which fits the simulation is a complex problem. Unless a perfect fit is obtained, the bound obtained will not maximize speedup. The assumption that the faulted curve will be same for all processes does not always hold and, therefore the initial value of BW would be invalid.

Penalty Based Throttling (Reiher et al., 1989) schedules processes which receive the least number of antimessages. Processes are penalized based on the antimessage count, and are basically skipped by the scheduler until the penalty reduces to zero. The major disadvantage of this technique is that the processor is not utilized when all processes are penalized in the system, thereby wasting CPU cycles. Performance results from such techniques has discouraged further use of this technique.

A combination of smallest timestamp first scheduling and load balancing to reduce the number of rollbacks has been presented by Glazer and Tropper (Glazer and Tropper, 1993). Load balancing is accomplished by varying time slices to processes based on the parameter *simulation advance rate* of each process. This is very similar to other approaches (Matsumoto and Taki, 1993; Palaniswamy and Wilsey, 1993a), but differs by the parameter used for perturbing the time slices.

Adaptive Bounded Time Windows (Palaniswamy and Wilsey, 1993a) uses the concept of *useful work* and dynamically sizes windows to maximize speedup. This technique solves all the problems associated with the other windowing techniques. *Adaptive Time-Ceiling* (Matsumoto and Taki, 1993) is based on a similar concept, although the window sizes are chosen from a set of discrete values. The controlling parameter is very simple and does not contain global information such as antimessage count, positive message count, etc.

Breathing Time Warp (Steinman, 1993) is a combination of breathing time buckets and time warp. This technique is based on the principle that events closer to GVT have a lesser probability of being rolled back. Thus N_1 events close to GVT are executed optimistically, and N_2 events after that point are executed using breathing time buckets. However, the mechanism provides no means to determine the values of N_1 and N_2, or dynamically altering their values to minimize execution time.

Hybrid approaches have been developed that add optimism to conservative algorithms. SRADS with local rollback (Dickens and Jr., 1990) and speculative computing (Mehl, 1991) optimistically process unsafe events locally; however, the results are not transmitted to other processes in the system. Thus, rollbacks are confined to local processes, and such approaches are termed as *aggressive but lacking risk*. Breathing Time Buckets (Steinman, 1991) is similar to the SRADS protocol with local rollbacks, but has the advantage of dynamically varying the conservative time windows based on the concept of global and local event horizons. Unfortunately, this technique performs poorly under small lookahead conditions. Another drawback of this technique is that all messages have to be flushed before calculating the global event horizon, *i.e.*, no message can be in transit.

Bounded lag (Lubachevsky, 1978) uses the minimum *distance* between logical processes as a basis for deciding safe events and requires the programmer to provide apriori lower bounds between causally connected events. Filtered Rollback relaxes the conservative time window to allow causality violation and recovers from errors by using optimistic rollback techniques. However, there is no provision to determine the *distance values* between processes to vary optimism to produce maximum speedup.

The smallest timestamp first (Lin and Lazowska, 1991) scheduling policy has been shown to have the capability to produce a time warp simulation which is conservative optimal. However, in a general purpose multiprocessing environment with processes partitioned across processors, the selected process can be rolled back by a straggler. Such cases can deteriorate the simulation. This is explained in detail in a later section. Thus, there is a need to derive a simple heuristic which schedules processes based on its rollback behavior and productive work.

The following can be concluded from the above discussion:

- The windowing approaches to limit optimism are promising, but there is a need to provide a mechanism to initialize and dynamically size the windows to maximize speedup. Thus rather than limit optimism, it has to be *controlled.*
- Adding optimism to conservative methods requires, explicit information from the user about *distances between processes* in some cases, and in general there is no provision to determine the optimal optimistic and conservative window sizes. These techniques usually fail for simulations with small lookahead since they are *risk-free.*
- Scheduling processes such that the LVT advances in a fairly constant manner has not been studied in detail. Thus, there is a need to devise, study, and compare scheduling techniques for time warp simulation.

The solution to maximize speedup is to *control optimism*, *i.e.*, to limit the optimism dynamically as the simulation proceeds. The following questions arise:

1. Do we control the optimism based on global or local information, or a combination of both?
2. What is the overhead of the control mechanism?

The solution here is to speed up or slow down individual processes (by intelligent scheduling) based on its own performance globally and locally. Implementing such a solution would require integration of simple control system techniques. Thus, a brief introduction to control systems techniques is necessary to elaborate on the solution.

3 CONTROL THEORY

Control theory is concerned with modifying the behavior of dynamical systems so as to achieve desired goals (Kokotovic, 1991). These goals include maintaining outputs at constant levels, assuring that the overall system track specified trajectories, or more generally the overall system optimizes a specified criteria. The goal is achieved by computing a suitable control input based on the observed outputs of the system. The fundamental process involved in *controlling* a dynamical system includes:

1. mathematical modeling of the system,
2. identification of the system based on experimental data, and
3. processing outputs and using them to synthesize control inputs to achieve desired behavior.

Adaptive control provides potential solutions to problems of the following general form. *The system to be controlled is normally exposed to a time-varying environment, in the form of a system with changing parameters, input signals, disturbances with time-varying statistical characteristics, or changing performance objectives.* However, adaptive control is not easy for the following reasons:

- Most systems have transfer functions which are impossible to determine.
- Parameter estimation is difficult.
- The adaptive element should be as simple as possible, so as not to affect the performance of the system.

To overcome these problems, the following adaptive system called *Model Reference Adaptive System (MRAS)* has been proposed (Kokotovic, 1991).

Model Reference Adaptive Systems (MRAS): An MRAS open loop model is shown in Figure 1. In such systems, an index of performance (IP) is identified, which indicates the systems instantaneous or short-term average performance quality. A control loop called the adaptive control routine or the adaptive element is then set up to optimize the IP automatically by adjusting control parameters.

The Adaptive Element: The adaptive element functions in 3 distinct steps, namely:

1. **Identification**, which is defined as the process by which the system is characterized, or by which the IP value is measured.

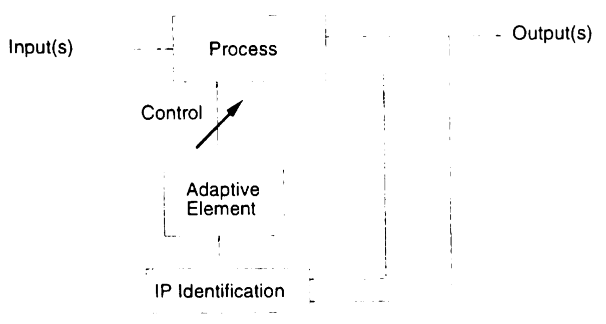

Figure 1: Model Reference Adaptive System

2. **Decision**, is the process by which the IP measurements are used to decide how system performance relates to the desired optimum.
3. **Modification**, is the process by which the system (control or plant) parameters are changed towards the optimum setting, as dictated by the identification and decision process.

The identification process is system dependent and is usually a heuristic IP value which closely represents the performance of the system. The decision and modification process is based on one of the following gradient methods (Eveleight, 1967): (i) steepest ascent, (ii) steepest descent, and (iii) fixed-increment ascent or descent techniques.

4 TIME WARP AS A SYSTEM OF PROCESSES

Consider a time warp simulation as a system of processes as shown in the block diagram of Figure 2. The complete time warp system is considered as a system of processes interconnected as determined by the communication structure of the simulation. The time warp simulation would be optimal if the output at each block is optimal. The output at each block can be made optimal by adaptively controlling the process using the MRAS model.

A fixed parameter system will not result in an optimal system as was observed from earlier discussions. Thus, an adaptive mechanism as shown in Figure 1 is required to continuously vary control parameters to provide optimal simulation. Since there is no way of determining the transfer function of the system, an IP which represents the progress of the system is required. The final system for each time warp process is configured as shown in Figure 1. The difficult stage in the design of the adaptive control is deciding on an IP for each time warp process. The next subsection deals with the selection of an appropriate Index of Performance (IP), which characterizes the performance of a time warp process.

4.1 The Concept of Useful Work

Ideally, for time warp to be optimal, all processes should execute without being rolled back. However, practically processes do get rolled back. To minimize execution time, the following conditions should exist in the parallel environment, namely:

- The number of rollbacks in each process should be minimal.
- The number of antimessages sent by each process should be minimal.
- The number of transactions undone should be reduced, so as to maximize lookahead computation.
- The number of transactions committed in a given time interval should be maximum, so as to increase the throughput of the system.

A lumped parameter (for each process) which reflects a combination of the above mentioned conditions would be very useful in determining the actual amount of optimism utilized by the processes. A simple quantity which reflects the progress of the process would be the number of events committed per second, *i.e.*, the throughput of the system. In this paper, the parameter is termed as the measure of *useful work* (W_i).

We formally define an initial measure of useful work by an LP as the ratio of the total number of events committed (E_c) every GVT cycle to the CPU time used by the process to commit those E_c events (δ_{cpu}) and is termed as the transition value of useful work (W_t):

$$W_t = \frac{E_c}{\delta_{cpu}} \qquad (1)$$

However, this simplistic measure does not reflect the actual throughput of the process for the following reasons:

- It is not necessary for all events to have the same execution times, *i.e.*, events can have varying execution times.
- Events can occur at varying frequencies, *i.e.*, more than one event can occur at the same LVT. This implies that there may be instances at which many events occur, and instances where a few events occur.

This implies that the ratio of the number of events committed per second as the measure of *useful work* is not correct for a real Parallel Discrete Event Simulation (PDES) environment. However, if all events have equal execution times and occur at a constant

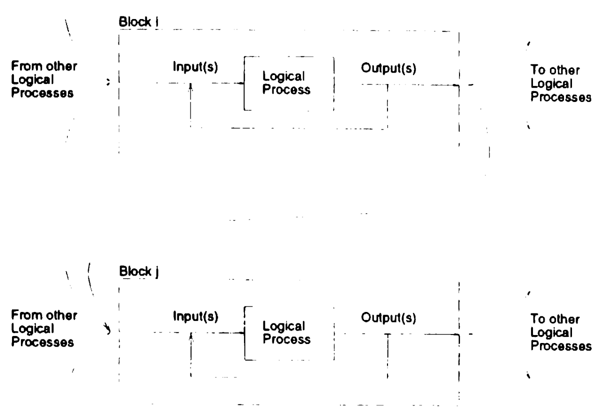
Block i
From other Logical Processes
Input(s)
Logical Process
Output(s)
To other Logical Processes
Block j
From other Logical Processes
Input(s)
Logical Process
Output(s)
To other Logical Processes

Figure 2: Time Warp Control System

frequency, then for such PDES environments Equation 1 holds as the measure of useful work. The next subsection presents a complete definition for a useful work parameter taking into account all possible overhead scenarios.

4.2 Performance Index: Useful Work

Assume that the useful work is calculated every GVT cycle, and all measurements are taken in this interval called the *measurement cycle*. Smaller the measurement cycle, closer will the system tend to be balanced. However, small measurement cycles would result in excessive time spent in calculating W_i rather than in useful computing.

Definitions:

- Let E_t represent the total number of events processed in the measurement cycle.
- Let E_c represent the number of events committed in the measurement cycle.
- Let E_u represent the number of events undone in the measurement cycle.
- Let M_n represent the number of antimessages (negative) sent in the measurement cycle.
- Let M_p represent the number of positive messages sent in the measurement cycle.
- Let α be the ratio of undone events to the total number of events executed.

$$\alpha = \frac{E_u}{E_t} \tag{2}$$

  Ideally we would like α to be equal to zero, *i.e.*, E_u is zero.
- Let γ be the average rollback length in the measurement cycle.
- Let N_r be the total number of rollbacks in the measurement cycle.

Assertions: The following assertions form the basis of the useful work model to be developed later.

1. A small value of α would result in more work done by the process, therefore:

$$W_i \propto \frac{1}{\alpha} \tag{3}$$

2. Every antimessage sent results in a performance deterioration, and therefore a decrease in the amount of useful work done, therefore:

$$W_i \propto \frac{1}{M_n} \tag{4}$$

3. Every rollback results in the process wasting time in restoring the current state and recomputation of events, therefore:

$$W_i \propto \frac{1}{N_r \gamma} \tag{5}$$

Model: Consider a logical process i in the time warp synchronized simulation. For the entire system of processes to execute in the shortest time possible, each process should execute in a minimum time. Therefore, each process should not waste excessive processing bandwidth on synchronization overheads (rollback, antimessages, state-saving, etc). From the assertions, we define a new quantity for the useful work W_i:

$$W_i = \frac{k}{\alpha\, M_n\, N_r\, \gamma} \tag{6}$$

where k is the constant of proportionality. Substituting for α, we obtain:

$$W_i = \frac{k\, E_t}{E_u\, M_n\, N_r\, \gamma} \tag{7}$$

A new parameter called *useful work*, which represents the productive work done, the overhead incurred and the forward progress for each time warp process has been presented. Two modifications needed to time warp have been successfully implemented and tested. The two implementations based on the concept of *useful work* are: (i) Adaptive Periodic state saving (Palaniswamy and Wilsey, 1993b), and (ii) Adaptive Bounded Time windows, which was proposed to solve the problem of optimal BTW (Palaniswamy and Wilsey, 1993a).

The final modification required for time warp is the need for a process scheduler. A parameterized process scheduler based on the concept of *useful work* has been proposed, implemented and tested in this paper. A detailed description of the proposed design is presented in the next section.

5 LP SCHEDULING

Consider a time warp simulation of n processes $(P_1, P_2, ...P_n)$, partitioned across m machines (processors). The collection of processes $(P_j ... P_k)$ on machine m_i is termed a *cluster* C_i. A formal statement of the scheduling problem in a time warp simulation is as follows:

For each cluster C_i schedule process P_e so as to satisfy the following criteria:

Criteria 1: The scheduler should be simple and not produce probing effects in the simulation. □

Criteria 2: The process to be scheduled should be chosen such that the overall execution time is reduced. □

Criteria 3: All processes should be given a fair chance to execute, otherwise deadlock due to starvation may occur. □

Given these criteria, we review two LP scheduling strategies, namely: *Smallest Timestamp First* and *Useful Work*.

5.1 Smallest Timestamp First Scheduling

The process with the smallest timestamp is always scheduled to execute (Lin and Lazowska, 1991). This technique results in good performance in homogeneous environments with global knowledge of LP advancement. However, in a distributed network or multicomputer environment the most efficient processing may not reside with the slowest advancing LP. This is especially true when the critical path of simulation time advancement migrates among a large region of the LPs and when LPs off the critical path can perform useful lookahead (especially when that lookahead advances a region of the simulation to another region of the critical path). In addition, the following phenomenon may also deteriorate overall system performance when smallest timestamp first scheduling is employed:

Phenomena 1: *Straggler messages* destined to the scheduled process may cause a performance deterioration. Bad partitioning and varying communication latencies could cause such a condition to frequently occur. In such cases, there is again a violation of Criteria 2. □

Phenomena 2: Processes with their *local clock at infinity* may have (true or false) messages waiting to be processed that could cause the process to rollback to the smallest timestamp in the system. Thus, smallest timestamp first scheduling does not satisfy Criteria 2. □

Put more simply, there are instances of simulation behaviors where LVT advancement is not the sole measure of productivity. Other factors such as rollback activity and frequency of antimessage generation are also contributors to measures of productivity.

5.2 Useful Work Scheduling

To accommodate all of the above criteria, a heuristic approach to select the appropriate process has to be devised. This scheduling algorithm, presented below, is based on the hypothesis that *a process with the largest value of* useful work *should be given priority for scheduling, since this would result in the overall productivity of the system of processes increasing.* This implies that the scheduling priorities for the processes are based on their local parameter of the Index of Performance, namely *useful work*. This algorithm will be particularly useful when there are a lot of *true messages* in transit, a condition where the scheduling decision based on the smallest timestamp first policy would not be optimal.

The scheduling algorithm is presented in Figure 3. An array of *useful work* values for all processes controlled by the scheduler is maintained. The process with the largest value of *useful work* is scheduled by the scheduler. Once this process has executed, it is placed at the end of the queue, and the process with the next highest *useful work* is scheduled. This is done in order to provide fair scheduling policy that

```
/*****************************************
  Let n be the number of processes to be
  scheduled.

  Maintain an array of length n sorted in
  decreasing order of useful work.

  Update and Sort the array after every
  GVT computation.
*****************************************/

begin Scheduler

  1. Schedule the process with the largest
     value of useful work.
  2. After execution move process to the
     tail end of the array.
  3. Schedule process with next highest
     value of useful work.
  4. Continue until simulation ends.

end Scheduler
```

Figure 3: Scheduling Algorithm using IP useful work

prevents deadlock (thus satisfying criteria 1 and 3). New values for *useful work* are calculated after every revision in GVT.

6 EXPERIMENTAL RESULTS

Both algorithms have been implemented for empirical performance comparison. In particular, an experimental platform of PDES for simulating digital system descriptions described using the hardware description language VHDL is used. The simulator runs on a 4 node SMP SPARC-1000 running SOLARIS 2.3. The benchmarks used are:

1. a 8 process Parallel to Serial Converter (PTSC),
2. a 20 process Parallel Multiplier (PMULT),
3. a 20 process Arithmetic Logic Unit (ALU), and
4. a 32 process Floating Point Arithmetic Unit (FPA).

Since processes use a large amount of system resources, time warp processes were implemented as threads, greatly reducing the size and execution time of the simulation.

The scheduler for each processor is configured as shown in Figure 4. The scheduler selects the thread to be executed from the pool of threads based on the scheduling policy (Each thread being a process to be executed). The present implementation schedules all threads for equal amounts of time on the processor.

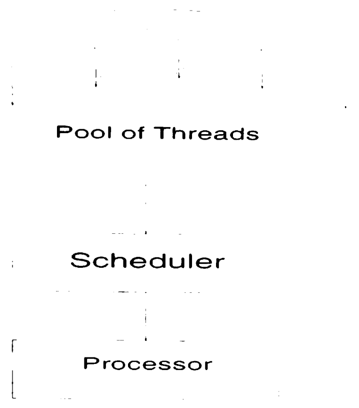

Figure 4: Time Warp Process Scheduler

Test	Policy	# Rollbacks	Time (Secs)
PTSC	LTF	385	16.24
	UW	395	16.37
PMULT	LTF	449	185.46
	UW	441	184.724
ALU	LTF	251	19.68
	UW	245	15.14
FPA	LTF	2784	35.42
	UW	2546	34.66

Table 1: Performance Results

All policies were executed with similar load conditions existing on the machine. The goal of the study was to compare the performance of the adaptive scheduling strategy with the smallest timestamp first scheduling policy.

Comparative Results for the benchmarks are presented in Table 1. From the table, it can be observed that the scheduling policy using the index of performance *useful work* performs as well or outperforms the smallest timestamp first scheduling policy. In general, the total number of rollbacks are decreased (2% to 10% reduction) using the new scheduling policy. Thus, initial experimental results have shown that the scheduling policy using *useful work* has promise and outperforms smallest timestamp by 20% for ALU. However, further testing with significantly more complex models are necessary to further validate the approach.

7 CONCLUSIONS

A new approach to visualizing time warp simulations as a dynamic system of processes that can be adaptively controlled has been presented. An index of performance (IP) called *useful work* has been derived. A new scheduling algorithm, based on the value of *useful* *work* of each process in the simulation has been proposed. The scheduling policy was implemented and compared against the smallest timestamp first algorithms. Initial results have been encouraging, and the new scheduling policy using model reference adaptive techniques has outperformed or performed as well as the smallest timestamp first policy.

The drawbacks of LVT scheduling have been overcome in the adaptive scheduling policy. In particular, the presented algorithm is based on model reference adaptive control techniques, with the following characteristics: (i) simplicity of implementation (ii) dynamic ordering of the round robin queue based on the simulation, and (iii) priority based on the productivity of each process in the simulation.

The new scheduling algorithm has shown promise to be effective at improving performance of time warp simulations. However, more empirical results are necessary to concretely conclude its usefulness.

REFERENCES

Ball, D. and Hoyt, S. (1990). The adaptive time-warp concurrency control algorithm. In *Distributed Simulation*, pages 174–177. SCS.

Davoren, M. (1989). *A Structural Approach to the Mapping Problem in Parallel Discrete Event Logic Simulations*. PhD thesis, Dept of Computer Science, University of Edinburgh, Edinburgh UK.

Dickens, P. M. and Jr., P. F. R. (1990). Srads with local rollback. In *Distributed Simulation*, pages 161–164. SCS.

Eveleight, V. W. (1967). *Adaptive Control and optimization techniques*. McGraw-Hill, New York, NY.

Fujimoto, R. M., Tsai, J., and Gopalakrishnan, G. C. (1992). Design and evaluation of the rollback chip: Special purpose hardware for time warp. *IEEE Trans. on Computers*, 41(1):68–82.

Glazer, D. W. and Tropper, C. (March 1993). On process migration and load balancing in time warp. *IEEE Transactions on Parallel and Distributed Systems*, 4(3):318–327.

Kokotovic, P. (1991). *Foundations of Adaptive Control*. Springer–Verlag, Berlin, Heidelberg.

Lin, Y.-B. and Lazowska, E. D. (1991). Processor scheduling for time warp parallel simulation. In *Advances in Parallel and Distributed Simulation*, pages 11–14. SCS.

Lubachevsky, B. D. (1978). Efficient distributed event driven simulations. *Communications of the ACM*, 32(1):63–72.

Matsumoto, Y. and Taki, K. (1993). Adaptive time-ceiling for efficient parallel discrete event simulation. In *Object-Oriented Simulation Conference*, pages 101–106. SCS.

Mehl, H. (1991). Speed-up of conservative distributed discrete event simulation methods by speculative computing. In *5th Workshop on Parallel and Distributed Simulation*, pages 163–166. SCS.

Palaniswamy, A. (1994). *Dynamic Parameter Adjustment to Speed Time Warp Simulation*. PhD thesis, Dept of ECE, University of Cincinnati.

Palaniswamy, A. and Wilsey, P. A. (1993a). Adaptive bounded time windows in an optimistically synchronized simulator. In *Great Lakes VLSI Conference*, pages 114–118.

Palaniswamy, A. and Wilsey, P. A. (1993b). Adaptive checkpoint intervals in an optimistically synchronized parallel digital system simulator. In *VLSI 93*, pages 353–362.

Reiher, P. L., Wieland, F., and Jefferson, D. R. (1989). Limitation of optimism in the Time Warp Operating System. In *Winter Simulation Conference*, pages 765–770. SCS.

Steinman, J. S. (1991). Speedes: A unified approach to parallel simulation. In *6th Workshop on Parallel and Distributed Simulation*, pages 75–84. SCS.

Steinman, J. S. (1993). Breathing time warp. In *7th Workshop on Parallel and Distributed Simulation*, pages 109–118. SCS.

Turner, S. J. and Xu, M. Q. (1992). Performance evaluation of the bounded time warp algorithm. In *6th Workshop on Parallel and Distributed Simulation*, pages 117–126. SCS.

West, D. (1988). Optimizing time warp: Lazy rollback and lazy re-evaluation. Master's thesis, University of Calgary, Calgary, Alberta.

AUTHOR BIOGRAPHIES

AVINASH C. PALANISWAMY completed his Ph.D. in Electrical and Computer Engineering from the University of Cincinnati in May, 1994. His research interests include, parallel synchronization algorithms, high speed parallel simulation of digital systems, CAD for board level simulation, Hardware/Software Co-Simulation, and acceleration of VHDL simulation.

PHILIP A. WILSEY is an assistant professor at the University of Cincinnati. His primary research interests are computer architecture, parallel processing, hardware description languages, MIMD on SIMD, and CAD.