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

COMBINING OPTIMISM LIMITING SCHEMES IN TIME WARP BASED PARALLEL SIMULATIONS

Kevin Jones and Samir R. Das

Division of Computer Science
The University of Texas at San Antonio
San Antonio, TX 78213-0667, U.S.A.

ABSTRACT

The Time Warp protocol is considered to be an effective synchronization mechanism for parallel discrete event simulation (PDES). However, it is widely recognized that it suffers over-optimistic behavior on the part of the simulation processes that may be very harmful for performance. In current literature, two techniques have been used to counteract this problem — (i) throttling of over-optimistic processes and (ii) scheduling or load balancing. However, study of these techniques has been primarily done in isolation. We demonstrate using a parameterized simulation model of Time Warp that an appropriate combination of throttling and global scheduling using LP migration can be very beneficial for performance compared to any one of these schemes acting in isolation. This study forms the basis of the design of more powerful control schemes that use a combination of multiple techniques.

1 INTRODUCTION

The Time Warp protocol (Jefferson, 1985) for parallel discrete event simulation (PDES) has the unique ability to transparently synchronize the executions of simulation events without requiring any significant model-specific information. Experimental studies demonstrate that Time Warp is also able to extract significantly more parallelism from the application simulation model compared to more traditional conservative mechanisms (Fujimoto, 1990a). Though otherwise promising, Time Warp is prone to inefficient execution in many situations because of over-optimistic behavior (Das and Fujimoto, 1997a). This behavior occurs when some logical processes (LPs) operate at a much larger simulation time than the others. Over-optimism may cause long and/or cascaded rollbacks (Lubachevsky, Shwartz, and Weiss, 1991) and the Time Warp system may spend a considerable amount of time in rolling back incorrect computation. Over-optimism may lead to other performance problems as well. For example, over-optimistic LPs may consume memory resources at an uncontrollable rate, making it impossible to complete the simulation with a finite amount of memory. Even if sufficient memory can be provided, memory management overheads may dominate (Das and Fujimoto, 1997a, Das and Fujimoto, 1997b). In the description that follows, we assume a general familiarity on the part of the reader about PDES (Fujimoto, 1990a) and specifically the Time Warp protocol (Jefferson, 1985).

Several mechanisms have been investigated in the literature to alleviate the problem of over-optimism in Time Warp. They can be broadly classified into two categories: *throttling* and *scheduling*. In throttling mechanisms, the optimism is controlled by preventing over-optimistic LPs from proceeding "too far ahead" in the simulation time. The basic idea here is to improve the *temporal locality* of the Time Warp computations (Jefferson, 1985). Temporal locality is used here in simulation time sense, i.e., the temporal locality principle applies if most events that are processed concurrently have close simulation timestamps. This reduces both the possibility of rollbacks and resource usage. Throttling mechanisms are not "work-conserving," as LPs may be blocked while CPU cycles are wasted. Too much throttling may also be harmful for performance as too few events are admitted for processing. This makes it mandatory that throttling must be controlled appropriately, by possibly an adaptive technique (Das, 1996a).

In scheduling, on the other hand, the question is deciding when and where (i.e., on which processor) an event with a given timestamp should be processed. A simple and widely followed scheduling discipline is to statically map LPs to processors and then schedule LPs mapped onto the same processor on an earliest LVT first basis. LVT or Local Virtual Time of an LP is the minimum timestamp of all unprocessed events on

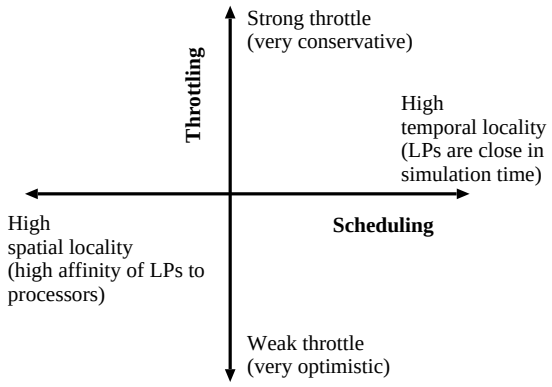

Figure 1: Throttling Techniques and Scheduling/Load Balancing Techniques Can be Applied Together.

that LP. Similarly, LVT of a processor is the minimum timestamp of all unprocessed events of all LPs currently mapped onto that processor. (Note that "virtual time" and "simulation time" are used synonymously in parallel simulation literature.) The *scheduling quantum* can be one event, i.e., rescheduling occurs after processing each event. The quantum may be higher if scheduling overhead is deemed expensive. However, many variations of this simple scheduling scheme are possible. For example, on shared memory systems a global scheduling discipline can be adopted (Ahmed, Rönngren, and Ayani, 1994), where all LPs go into a central pool. Each processor schedules on itself any currently unscheduled LP in the pool with the lowest LVT. Scheduling quanta here play a very important role, as scheduling overheads may be significant, dominated by the central pool bottleneck as well as the lack of *spatial locality*. Poor spatial locality comes from the lack of affinity of LPs to processors and gives rise to overhead costs in terms of cache coherence traffic.

Several schemes in distributed memory systems (including workstation cluster platforms) also work with similar philosophy, i.e., "trading spatial locality to gain temporal locality." The goal here is to achieve good temporal locality via migration of LPs with low LVT to processors with high LVT. When an explicit LP migration is involved, the term *load balancing* is commonly used in literature rather than *scheduling.* However, we will continue to use the term *scheduling* for any processor resource allocation decision irrespective of the specific mechanism or architectural platform. Note that all such scheduling mechanisms are "work conserving," i.e., CPU resources are never wasted as long as there are unprocessed events. Also, unlike throttling, scheduling may affect spatial locality. Depending on the underlying architecture, loss of spatial locality can significantly impact performance.

In current literature, throttling and scheduling schemes have been primarily treated in isolation, even though they work towards the common goal, *viz.*, improving temporal locality in Time Warp programs. Our thesis is that a combination of both schemes can offer a significant performance advantage compared to any single scheme alone. This is based on the general observation that throttling wastes processor cycles, when all LPs mapped onto a processor are blocked. These cycles can be utilized if allocated to LP(s) that are slow in simulation time. However, this can only be cost-effective when the migration of LP(s) from another processor is not expensive compared to savings obtained from utilization of processor cycles. Thus, depending on costs associated with (i) rollback and (ii) loss of spatial locality, a combination of these otherwise orthogonal schemes may prove beneficial (see Figure 1). Based on this observation, we take a holistic view of the scheduling paradigm and with a parameterized simulation model of a Time Warp system investigate the scenarios where combination of throttling and scheduling can be beneficial.

The rest of this paper is organized as follows. In the next section, we briefly review existing throttling and scheduling schemes for Time Warp. In Section 3, we describe a simulation model of Time Warp used to evaluate the performance of throttling and scheduling. We present the performance benchmarking results in Section 4 and conclusions in Section 5.

2 RELATED WORK

In this section we briefly review the existing literature on throttling and scheduling in connection with Time Warp.

2.1 Throttling

Quite a few mechanisms have been developed that use some form of optimism control to improve the performance of Time Warp. Important examples include (i) limiting all event computations within a simulated time window (Sokol and Stucky, 1990, Turner and Xu, 1992) above the global virtual time (GVT), GVT being the minimum timestamp of all the unprocessed events in Time Warp (Jefferson, 1985), (ii) rolling back all processes to GVT (or close to GVT) at stochastically selected intervals in real time (Madisetti, Hardaker, and Fujimoto, 1993), (iii) not sending messages unless they are guaranteed to be correct, thereby eliminating the need for anti-messages (Dickens and Reynolds, Jr., 1990, Steinman, 1992) (this is also known as *risk-free* computation), (iv) bounding the total amount of memory that can be allocated to the Time Warp system using memory management protocols like cancelback (Das and Fujimoto, 1997b,

Das and Fujimoto, 1997a), (v) limiting the number of events each LP may execute beyond GVT (Steinman, 1993).

All these mechanisms have been shown to perform better than the "purely optimistic" Time Warp for certain simulation models. In general, it is believed that an appropriate throttling of Time Warp execution (i.e., blocking one or more LPs even if they have unprocessed events in their future event list) has a strong potential for improved performance. However, the throttle must be applied with caution. As observed in (Srinivasan and Reynolds, 1995a), both purely optimistic Time Warp and Time Warp with an adaptive throttle can arbitrarily outperform each other under specific circumstances. This makes it imperative to study the appropriateness and amount of throttle required for the best possible performance.

Thus, several schemes study the so-called *adaptive* mechanisms that dynamically change certain control parameters to influence the degree of throttling. Notable examples are described in following. An adaptive concurrency control scheme proposed by (Ball and Hoyt, 1990) adjusts a blocking window according to the minimum of a function that describes the CPU cycles lost due to blocking and due to rollback recovery. An adaptive memory management protocol proposed by (Das and Fujimoto, 1997a) takes an indirect approach and controls the memory allocation to adjust the throttle. (Ferscha and Lüthi, 1995) use a probabilistic cost model to evaluate the tradeoff between optimistically processing and conservatively blocking the simulation engine. Their model computes an optimal real time delay interval to minimize the rollback overhead. (Ferscha, 1995) suggests probabilistic throttling of logical processes based on time-series forecasting. (Hamnes and Tripathi, 1994) propose a local adaptive protocol, where real time blocking windows are computed based on inter-arrival times (both real and virtual) of messages in each input channel. (Srinivasan and Reynolds, 1995b) promote the use of near perfect state information (NPSI) (such as, a good estimate of GVT) to compute the *error potential* for logical processes and to throttle the processes with higher error potential. (Tay, Teo, and Kong, 1997) design an adaptive throttle that controls the scheduling quanta of the LPs on a processor based on their virtual time progress.

2.2 Scheduling

Comparatively fewer papers deal with the issue of scheduling and load balancing. In (Ahmed, Rñnigren, and Ayani, 1994) the global scheduling discipline is studied similar to that described in Section 1. The central scheduling queue is implemented as a concurrent priority queue. Experimental results demonstrate significant performance advantages over static mapping. However, the authors note that the loss of spatial locality and bottleneck at the central queue can be a limiting factor for large scale implementations, beyond 8 or 16 processors.

Several papers have studied some form of dynamic load balancing for Time Warp programs. In one of the early works, (Reiher and Jefferson, 1990) propose a metric called *effective utilization*, which is defined as the fraction of time a processor is computing events that are not rolled back (Reiher and Jefferson, 1990). Based on this metric they propose a strategy that migrates LPs from processors with high utilization to processors with low utilization. They also propose splitting of LPs in *phases* to reduce LP migration costs. (Glazer and Tropper, 1993) propose allocating virtual time-slices to LPs, based on their observed rate of advancing the simulation clock. (Burdorf and Marti, 1993) propose an approach to balance the LVTs of LPs by moving "slow" (in simulation time) LPs to processors with "fast" LPs. They observe that this approach balances the workload in presence of external computations competing for the same processors. (SchlagenHaft, Ruhwandl, Sporrer, and Bauer, 1995) present an approach of load balancing based on *virtual time progress*, which reflects how fast a simulation process continues in virtual time. (Carothers and Fujimoto, 1996) propose a similar dynamic load management policy for background execution of Time Warp programs on workstation clusters. Their work also includes a processor allocation policy where the the set of processors allocated for simulation can be changed dynamically. Finally, (Avril and Tropper, 1996) consider a clustered approach where a more conventional definition of *load* is used. Load here is defined in terms of the total number of events processed. The load balancing algorithm takes into account the fine granularity of logic simulation problems, for which the technique is targeted, as well as interprocessor communication overheads. A triggering mechanism based on simulation throughput is also described.

The central contribution of our work is the consideration of throttling and global scheduling in the same framework and a study of the tradeoffs. Throttling is implemented using a simple time window based mechanism. LP migration is used for global scheduling. Using a simulation model of Time Warp, we demonstrate the performance advantage of this "combined" approach for a range of overheads affecting rollback and LP migration costs.

3 A TIME WARP MODEL WITH COMBINED THROTTLING AND LP MIGRATION

A simulation model of Time Warp is designed using the process-oriented simulation tool CSIM (Schwetman, 1994). Though an operational Time Warp system could be used, a

simulation model is preferred as it provides the flexibility to use various overhead costs that model a range of architectural platforms. In the simulated Time Warp system, each processor is modeled by a CSIM process. An LP is modeled as an event list, ordered by timestamps and a current state vector. The complete state is checkpointed every event. The system starts with a random mapping of LPs onto processors, with each processor getting equal number of LPs. Each processor performs a local scheduling of the LPs mapped onto itself after every event using the earliest LVT first discipline. The Global Virtual Time (GVT) (Jefferson, 1985) is evaluated after processing every event and subsequently any accumulated fossils are collected.

A time window based scheme (Das, 1996b) is used to implement throttling. If the width of the time window is T_w, the LPs with LVT beyond (GVT + T_w) are not scheduled for execution. Note that the edge of the window always moves forward as GVT is updated. The value T_w is fixed and is an experimental parameter.

LPs can migrate between processors based on periodic remapping decisions. Currently, LPs are remapped onto processors such that (i) the N slowest LPs in simulation time are mapped onto different processors, where N is the number of processors, and (ii) each processor has the same number of LPs. This ensures that after each remapping the slowest LPs get processor cycles to make progress. The remapping is done at regular intervals, the interval being an experimental parameter. After each remapping, the actual LP migrations are performed asynchronously after all pending operations (such as rollbacks) on the migrating LP are completed.

The following are the only costs assumed in the Time Warp simulation:

- The processing time of an event (T_e).
- Overhead cost for sending a message or antimessage (T_s) and cost for an event cancellation (T_c).
- Overhead cost for migrating a single LP to a remote processor (T_m).

All other costs are ignored. To prevent an explosion of number of experiments, T_s is assumed to be equal to T_c. For a meaningful characterization of various overheads, the experiments are described in terms of two parameters, r_1 and r_2, as follows:

$$r_1 = \frac{T_s}{T_e}, r_2 = \frac{T_m}{T_s}.$$

The parameters r_1 and r_2 model the communication/rollback related cost and process migration related cost respectively, normalized to event granularity. All time values above reflect mean values, in case a stochastic distribution is used in the simulation to choose an actual value.

4 PERFORMANCE ANALYSIS

Here we describe the Benchmark model used for our experiments and an analysis of the performance results.

4.1 Benchmark Model

A synthetic benchmark application is constructed for performance evaluation. The benchmark is intentionally made highly unbalanced and dynamically changing to bring out the worst case performance in Time Warp. This not only presents a challenging test case, but also makes optimism limiting schemes worthwhile. The benchmark uses the popular PHOLD model (Fujimoto, 1990b) with hotspot traffic. The hotspot targets move dynamically. The PHOLD model is similar to a closed queueing network model with infinite number of servers in each node. Each queueing node is modeled as an LP. A fixed number of jobs (called *message population*) move around from queue to queue. The queueing network is completely connected. The routing probabilities are non-uniform and dynamically changing to create moving hotspots.

In the experimental results that follow, 32 processors and 512 LPs have been used. 32 out of the 512 LPs are designated to be hotspots and a fixed fraction of all messages are targeted to the hotspot LPs. In the experiments, this fraction is 40%. The message population is 8192. The service times in a queueing node are exponentially distributed with mean 1.0. Each 1.0 time unit in simulation time, the hotspots migrate to a different set of LPs. The new set of hotspot LPs, however, is not completely disjoint from the previous set; it has a 50% overlap with the previous set. Thus hotspot movements are somewhat gradual.

Event processing times are exponentially distributed with mean 1.0 time unit thus giving a sequential execution speed of 1 event per time unit. This moving hotspot model brings out the worst in Time Warp execution. Without any throttling and migration the execution on 32 processors is much slower than sequential with more than 99% events rolling back.

4.2 Experimental Results

The experiments are done with different values of the ratio, r_1/r_2. r_1 is always chosen to be 1, i.e., T_s = T_e. This is not unreasonable as simulation events are typically very fine grain on state-of-the-art processors, making event computation time and message sending time comparable even in shared memory systems. r_2 is varied in the

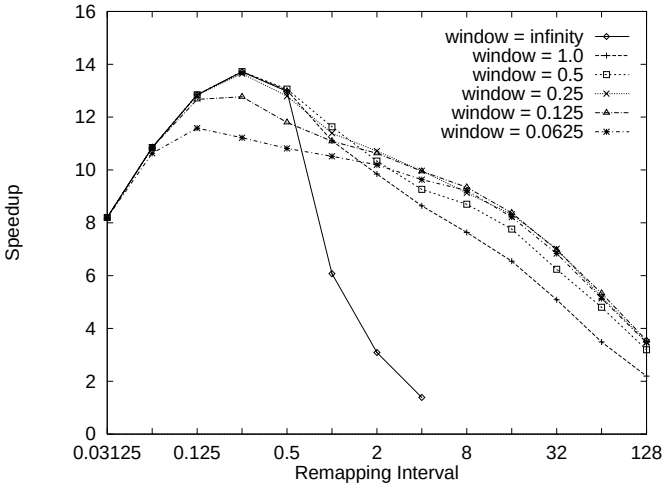

Figure 2: Speedups for Varying Remapping Interval and Time Window size. Cost Parameters: $r_1 = 1, r_2 = 1$.

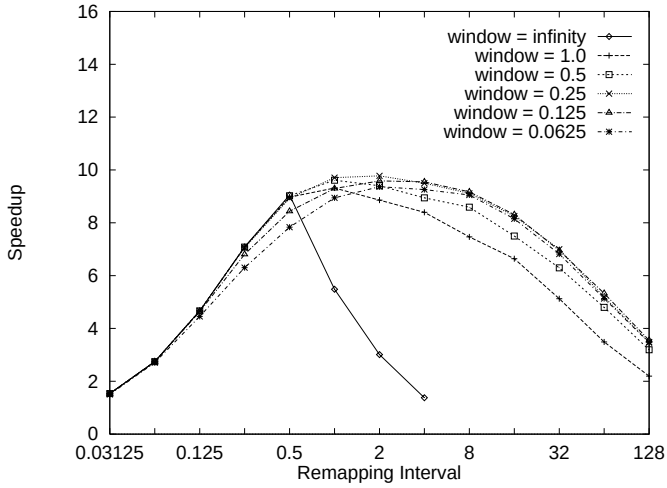

Figure 3: Speedups for Varying Remapping Interval and Time Window size. Cost Parameters: $r_1 = 1, r_2 = 10$.

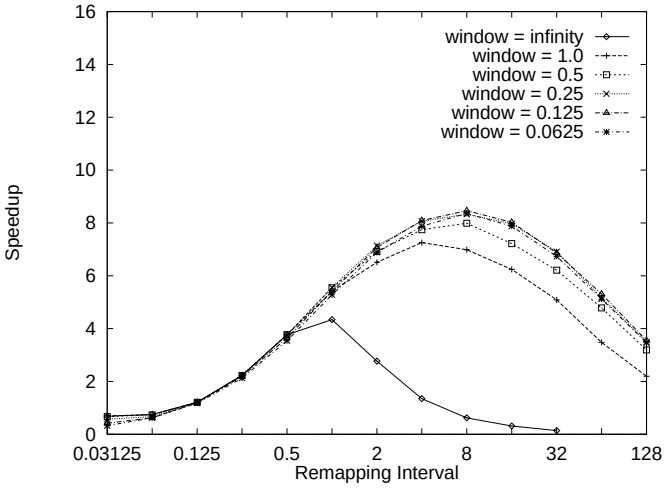

Figure 4: Speedups for Varying Remapping Interval and Time Window Size. Cost Parameters: $r_1 = 1, r_2 = 50$.

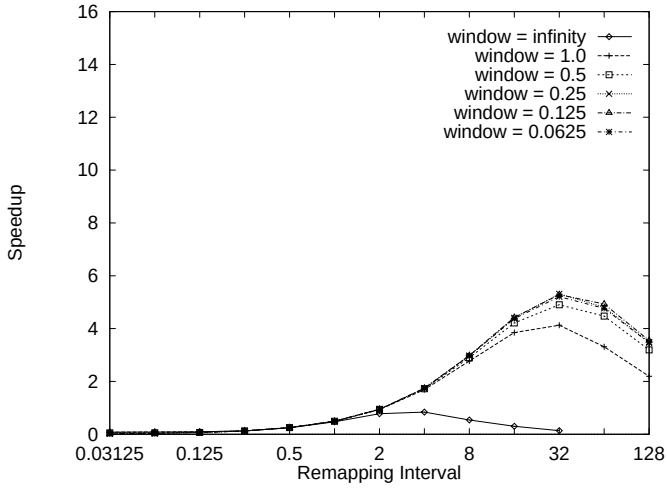

Figure 5: Speedups for Varying Remapping Interval and Time Window Size. Cost Parameters: $r_1 = 1, r_2 = 1000$.

range 1 to 1000, thus simulating a range of LP migration costs from very cheap (same as sending a single message, $r_2 = 1$) to very expensive (1000 times more expensive than sending a single message, $r_2 = 1000$). This models a range of architectural platforms, for example, from shared memory to network of workstations, where LP migrations will incur different costs. Figures 2 to 5 present speedup data for (i) different values of r_2, (ii) different sizes of time window and (iii) different remapping intervals. Remapping interval is expressed in terms of GVT progress in simulation time units between subsequent remapping. The simulation run length has been chosen to be 128 time units, which corresponds to more than a million "committed" events in Time Warp. Thus, in any of the performance figures the points corresponding to remapping interval = 128, actually correspondto no remapping.

It is noted that for a given cost of LP migration and a given size of time window, the speedup data peaks at a certain remapping interval. For higher remapping intervals, rollbacks are observed to be progressively more frequent. On the other hand, lower remapping intervals result in higher cost due to migrations. Thus, the peak of each single line in the plots represents the most cost-effective balance between temporal and spatial locality. Also, notice that throttling (i.e., time window size less than infinity) starts producing benefits as migration cost increases (i.e., as r_2 goes up). For $r_2 = 1$, throttling has no performance advantage (Figure 2). For $r_2 = 10$,

Table 1: Comparison of the Optimal Speedup With Speedup Achievable With Throttling or Migration Alone.

Value of r_2	Optimal speedup	Best speedup with migration alone	Best speedup with throttling alone
1	13.7	13.7	3.6
10	9.8	9.1	3.6
50	8.5	4.3	3.6
1000	5.3	0.8	3.6

throttling starts showing some advantage. Note that the peak performance occurs for time window = 0.25 in Figure 3. As r_2 is increased further, throttling produces very significant advantage (Figures 4 and 5). In particular, the optimal width of the time window goes down a little with increasing r_2 (increasing migration cost).

Our strongest observation, however, is that throttling or remapping alone can achieve only a fraction of the optimal performance that can be obtained when both are employed together. In Figures 2 to 5, the solid line corresponds to time window = infinity, i.e., no throttling. Also, in each figure, the points on the right hand extreme (at remapping frequency = 128) correspond to no remapping or process migration. Table 1 summarizes the observations on these points. Note that unless migration is very inexpensive, there is always a significant benefit from the use of combined throttling and migration techniques.

5 CONCLUSIONS

The over-optimistic behavior in Time Warp is the source of many performance ills, including high rollback related costs and memory management overheads. The study presented here is the first step in using a combination of two disparate techniques in controlling the over-optimistic behavior. It is demonstrated using a simple parameterized simulation model that the right combination of global LP scheduling and throttling provides performance benefits not attainable with a single scheme used in isolation. The benefits, however, do depend on various overheads. In particular, if the LP migration overhead is small (for example, in a shared memory system), the benefit is little. The benefit progressively increases as migration overhead becomes more significant (which will be the case in distributed memory systems and network of workstations) with throttling making more and more impact on performance.

Future study will include more comprehensive performance evaluation of the combined technique and investigation of adaptive techniques for dynamically choosing appropriate LP remapping intervals and time window sizes. More sophisticated scheduling disciplines will also be considered. We expect to benefit from the wealth of literature dealing with adaptive techniques (Das, 1996a). They, however, all are unidimensional and are concerned with controlling a single parameter. Another future step will be implementation of the combined technique on an operational Time Warp system.

ACKNOWLEDGMENTS

This work is partially supported by AFOSR grant no. F49260-96-1-0472, Texas Advanced Technology Program grant no. 010115-248b and NSF CAREER award no. ASC-9733836.

REFERENCES

Ahmed, H., R. Rönngren, and R. Ayani 1994. Impact of event scheduling on performance of Time Warp parallel simulations. In *Proceedings of the 27th Annual Hawaii Intl. Conf. on System Sciences*.

Avril, H. and C. Tropper 1996. The dynamic load balancing of clustered time warp for logic simulation. In *Proceedings of the 10th Workshop on Parallel and Distributed Simulation (PADS)*, pp. 20–27.

Ball, D. and S. Hoyt 1990. The adaptive Time-Warp concurrency control algorithm. *Proceedings of the SCS Multiconference on Distributed Simulation 22*(1), 174–177.

Burdorf, C. and J. Marti 1993. Load balancing strategies for Time Warp on multi-user workstations. *The Computer Journal 36*(2), 168–176.

Carothers, C. and R. M. Fujimoto 1996. Background execution of time warp programs. In *Proceedings of the 10th Workshop on Parallel and Distributed Simulation (PADS)*, pp. 12–19.

Das, S. R. 1996a. Adaptive protocols for parallel discrete event simulation. In *Proceedings of the 1996 Winter Simulation Conference*, pp. 186–193.

Das, S. R. 1996b. Estimating the cost of throttled execution in Time Warp. In *Proceedings of the 10th Workshop on Parallel and Distributed Simulation*, pp. 186–189.

Das, S. R. and R. M. Fujimoto 1997a. Adaptive memory management and optimism control in Time Warp. *ACM Transactions on Modeling and Computer Simulation 7*(2), 239–271.

Das, S. R. and R. M. Fujimoto 1997b. An empirical evaluation of performance-memory trade-offs in Time Warp. *IEEE Transactions on Parallel and Distributed Systems 8*(2), 210–224.

Dickens, P. M. and P. F. Reynolds, Jr. 1990. SRADS with local rollback. *Proceedings of the SCS Multiconference on Distributed Simulation 22*(1), 161–164.

Ferscha, A. 1995. Probabilistic adaptive direct optimism control in Time Warp. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation*, pp. 120–129.

Ferscha, A. and J. Lüthi 1995. Estimating rollback overhead for optimism control in Time Warp. In *Proceedings of the 28th Annual Simulation Symposium*.

Fujimoto, R. M. 1990a. Parallel discrete event simulation. *Communications of the ACM 33*(10), 30–53.

Fujimoto, R. M. 1990b. Performance of Time Warp under synthetic workloads. *Proceedings of the SCS Multiconference on Distributed Simulation 22*(1), 23–28.

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

Hamnes, D. O. and A. Tripathi 1994. Evaluation of a local adaptive protocol for distributed discrete event simulation. In *Proceedings of the 1994 International Conference on Parallel Processing*, pp. III:127–134.

Jefferson, D. R. 1985. Virtual time. *ACM Transactions on Programming Languages and Systems 7*(3), 404–425.

Lubachevsky, B. D., A. Shwartz, and A. Weiss 1991. An analysis of rollback-based simulation. *ACM Transaction on Modeling and Computer Simulation 1*(2), 154–193.

Madisetti, V. K., D. A. Hardaker, and R. M. Fujimoto 1993. The MIMDIX operating system for parallel simulation and supercomputing. *Journal of Parallel and Distributed Computing 18*(4), 473–483.

Reiher, P. L. and D. R. Jefferson 1990. Virtual time based dynamic load management in the Time Warp Operating System. *Proceedings of the SCS Multiconference on Distributed Simulation 22*(1), 103–111.

SchlagenHaft, R., M. Ruhwandl, C. Sporrer, and H. Bauer 1995. Dynamic load balancing of a multicluster simulation on a network of workstations. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS)*, pp. 175–180.

Schwetman, H. 1994. CSIM17: A simulation model building toolkit. In *1994 Winter Simulation Conference Proceedings*, pp. 464–470.

Sokol, L. M. and B. K. Stucky 1990. MTW: experimental results for a constrained optimistic scheduling paradigm. *Proceedings of the SCS Multiconference on Distributed Simulation 22*(1), 169–173.

Srinivasan, S. and P. F. Reynolds 1995a. Adaptive algorithms vs. Time Warp: An analytical comparison. In *1995 Winter Simulation Proceedings*, pp. 666–673.

Srinivasan, S. and P. F. Reynolds 1995b. NPSI adaptive synchronization algorithms for PDES. In *1995 Winter Simulation Proceedings*, pp. 658–665.

Steinman, J. S. 1992. SPEEDES: A unified approach to parallel simulation. In *6th Workshop on Parallel and Distributed Simulation*, pp. 75–84.

Steinman, J. S. 1993. Breathing Time Warp. In *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, pp. 109–118.

Tay, S. C., T. M. Teo, and S. T. Kong 1997. Speculative parallel simulation with an adaptive throttle scheme. In *Proceedings of the 11th Workshop on Parallel and Distributed Simulation (PADS)*, pp. 116–123.

Turner, S. J. and M. Q. Xu 1992. Performance evaluation of the bounded Time Warp algorithm. *Proceedings of the SCS Multiconference on Parallel and Distributed Simulation 24*(3), 117–126.

AUTHOR BIOGRAPHIES

KEVIN JONES is a Ph.D. student in the Division of Computer Science at The University of Texas at San Antonio. He received his B.S. degree from The University of Texas at San Antonio in 1993 and M.S. degree from The University of Texas at Austin in 1995, both in computer science. His research interests are in parallel discrete event simulation.

SAMIR R. DAS is an assistant professor in the Division of Computer Science at The University of Texas at San Antonio. He received his B.E. degree in electronics and telecommunication engineering from Jadavpur University, Calcutta, in 1986; M.E. degree in computer science and engineering from the Indian Institute of Science, Bangalore, in 1988; and Ph.D. degree in computer science from the Georgia Institute of Technology, Atlanta, in 1994. His current research interests include parallel simulation, mobile/wireless networking and performance evaluation. Dr. Das served on the program committees of several annual ACM/IEEE/SCS workshops on Parallel and Distributed Simulation (PADS). He received the U.S. National Science Foundation's Faculty Early CAREER award in 1998. He is a member of the IEEE, IEEE Computer Society and ACM.