Extensions to Time Warp Parallel Simulation for Spatial Decomposed Applications

Benno J. Overeinder Peter M. A. Sloot
Department of Computer Science, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract

In recent years, the use of discrete event simulation to solve problems from natural sciences has become more common as the dynamic time evolution of the real-world system is naturally incorporated in the discrete event system model. The parallel simulation of these discrete event systems puts some extra requirements on the parallel synchronization schemes such as Time Warp. The large scientific problems require efficient memory management—both time and space efficient—and parallelism control to achieve satisfactory performance.

1 Introduction

There is an increasing interest in the application of discrete event simulation to solve problems from natural sciences. In particular, problems with heterogeneous spatial and temporal behavior are, in general, most exactly mapped to asynchronous models [Bersini and Detours, 1994; Lumer and Nicolis, 1994]. The interest in discrete event simulation now is motivated by the ability of this protocol to capture the asynchronous behavior that is a qualifying characteristic of these models. Besides the aspect of asynchronicity, a general tendency is the construction of more realistic models resulting in more complex and larger simulations, which requires vast amounts of execution time. One fundamental method to reduce the execution time of large discrete event simulations is the exploitation of parallelism inherent to this class of simulations [Livny, 1985; Overeinder and Sloot, 1995]. Most research in Parallel Discrete Event Simulation (PDES) is focussed on protocol design; and although there are encouraging advances, none of the protocols devised thus far have been shown to perform efficiently for different applications.

We are specifically interested in the application of PDES methods, in particular the Time Warp method [Jefferson, 1985], to dynamic complex systems that can be modeled with Asynchronous Cellular Automata (ACA) [Overeinder and Sloot, 1993]. Typical ACA models are for example continuous-time Ising spin systems or spatial-decomposed population dynamics models. In the next sections, we will describe some new insights and extensions to the original Time Warp method in order to apply the PDES method successfully to ACA models. In Section 4, we present the results of a parallel Ising spin simulation using the Time Warp execution environment. Section 5 concludes with a discussion and some future work.

2 Background

Discrete event systems are characterized by a limited number of events occuring at irregular time intervals. Efficient simulation of such systems require concurrent execution of events at different points in simulation time. The absence of a global clock necessitates sophisticated synchronization algorithms to ensure that cause-and-effect relationships are correctly reproduced by the simulator. Parallel discrete event simulation is essentially concerned with the correct ordering, or scheduling, of the asynchronous execution of events over distributed or parallel systems. There are basically two methods to impose the correct temporal order of the asynchronous event execution: conservative and optimistic methods.

First, the conservative approach proposed by Chandy and Misra [1979] strictly imposes the correct temporal order of events. Second, the optimistic approach, introduced by Jefferson [1985], uses a detection and recovery mechanism: whenever the incorrect temporal order of events is detected a rollback mechanism is invoked to recover. Although both approaches have their specific application area, optimistic methods offer the greatest potential as a general-purpose simulation mechanism.

The most well-known optimistic method is the Time Warp simulation mechanism, which is based on the concept of virtual time. Virtual time describes how different distributed objects interact in time, and can therefore be used to serve as a basis for distributed simulation. The Time Warp mechanism implements virtual time and adheres to the temporal coordinate system imposed on a distributed simulation.

In optimistic simulation, the parallel simulation processes execute events and proceed in local simulated time as long as they have any input at all. A consequence of the optimistic execution of events is that the local clock or Local Virtual Time (LVT) of a process may get ahead of its neighbors' LVTs, and it may receive an event message from a neighbor with timestamp smaller than its LVT, that is, in the past of the simulation process. The event causing the causality error is called a *straggler*. If we allow causality errors happen, we must provide a mechanism to recover from these errors in order to guarantee a causally correct parallel simulation. Recovery is accomplished by undoing the effects of all events that have been processed prematurely by the process receiving the straggler. The net effect of the recovery procedure is that the simulation process rolls back in simulated time.

The premature execution of an event results in two things that have to be rolled back: (i) the state of the simulation process and (ii) the event messages sent to other processes. The rollback of the state is accomplished by periodically saving the process state and restoring an old state vector on rollback: the simulation process sets its current state to the last state vector saved with simulated time earlier than the timestamp of the straggler. Recovering from premature sent messages is accomplished by sending an *anti-message* that annihilates the original when it reaches its destination. The messages that are sent while the process is propagating forward in simulated time, and hence correspond with simulation events, are called *positive messages*.

A direct consequence of the rollback mechanism is that more anti-messages may be sent to other processes recursively, and allows all effects of erroneous computation to be eventually canceled. As the smallest unprocessed event in the simulation is always safe to process, it can be shown that this mechanism always makes progress under some mild constraints.

In optimistic simulation the notion of global progress in simulated time is administered by the Global Virtual Time (GVT). The GVT is the minimum of the LVTs for all the processes and the timestamps of all messages (including anti-messages) sent but unprocessed. No event with timestamp smaller than the GVT will ever be rolled back, so storage used by such event (i.e., saved state vector and event message) can be discarded. Also, irrevocable operations such as I/O cannot be committed before the GVT sweeps past the simulation time at which the operation occurred. The process of reclaiming memory and committing irrevocable operations is referred to as *fossil collection*.

3 Optimistic Parallel Discrete Event Simulation and Ising Spin Systems

The Asynchronous Cellular Automata (ACA), like the "classical" Cellular Automata (CA), is a set of dynamic systems where space and variables are discrete. The synchronous CA evolves in discrete time $t = 1, 2, \ldots$. The state of a cell at $t+1$ is determined by the state of the cell and its neighbors at t and may explicitly depend on t and the result of a random experiment. The ACA evolves, unlike the CA, not in discrete, but in continuous time. In the ACA model, the state changes at different cells occur asynchronously at unpredictable random times, and thus is a discrete event system. Zeigler [1982] also postulated that discrete event ACA models might be a more natural and adequate representation of the universe at the level of basic physics.

A specific example of an ACA is the continuous-time Ising spin model. Glauber [1963] introduced continuous-time probabilistic dynamics for an Ising system to represent the time evolution of the physical system. The Ising spin model with continuous-time probabilistic dynamics cannot be solved by Monte Carlo simulation, since time has no explicit implication on the evolution of the system in this execution model. To capture the asynchronous continuous-time dynamics correctly, the most efficient underlying execution model is discrete event simulation. In this respect, the Ising spin model is an ideal application to study the effectiveness of PDES methods, as the dynamic behavior of the Ising spin model is essentially determined by one parameter, namely the temperature of the system. The temperature determines the ratio of communication to computation in the Ising spin model, and gives a well-defined parameter to evaluate the Time Warp method with a realistic application.

The Ising model is a popular model of a system of interacting variables in statistical physics. To introduce the Ising model, consider a lattice containing N sites and assume that each lattice site i has associated with it a number s_i, where $s_i = +1$ for an "up" spin and $s_i = -1$ for a "down" spin. The total energy of the Ising model is given by

$$E = -J \sum_{i,j=\mathrm{nn}(i)}^{N} s_i s_j - \mu_0 H \sum_{i=1}^{N} s_i \,,$$

where $s_i = \pm 1$, J is the measure of the strength of the interaction between the spins, and the first sum is over all pairs of spins that are nearest neighbors. The second term is the energy of interaction of the magnetic moments, μ_0, associated with the spins with an external magnetic field, H. In our discussion, the external magnetic field H is zero.

Figure 1: The interaction energy between nearest neighbor spins in the absence of an external magnetic field.

If $J > 0$, then the states $\uparrow\uparrow$ and $\downarrow\downarrow$ are energetically favored in comparison to the states $\uparrow\downarrow$ and $\downarrow\uparrow$ (see Fig. 1). Hence for $J > 0$, we expect that the state of the lowest total energy is ferromagnetic, i.e., the spins all point to the same direction. If $J < 0$, the states $\uparrow\downarrow$ and $\downarrow\uparrow$ are favored and the state of the lowest energy is expected to be antiferromagnetic, i.e., alternate spins are aligned.

At random times, a spin is granted a chance to change the state, a so-called spin flip. The attempted state change arrivals for a particular spin form a Poisson process. The Poisson arrival processes for different spins are independent, however, the arrival rate is the same for each spin. The attempted spin flip, or trial, is realized by calculating the energy difference ΔE between the new configuration and the old configuration. The spin flip is accepted with a probability P given by

$$P = \begin{cases} 1, & \text{if } \Delta E \leq 0, \\ \exp(-\Delta E/kT), & \text{otherwise,} \end{cases}$$

given temperature T and Boltzmann's constant k. For convenience we measure energy in units of k and take $J = 1$, so that T is effectively unitless.

The resulting continuous-time Ising spin model is parallelized by spatial decomposition. The Ising spin lattice is partitioned into sub-lattices, and the sub-lattices are mapped onto the parallel processors. To minimize the communication between sub-lattices, local copies of the neighbor boundaries are stored locally (see Fig. 2). By maintaining local copies of neighbor boundaries, spin values are only communicated when they are actually changed, rather than when they are only referenced. A spin flip along the boundary is communicated to the neighbors by an event message. The causal order of the event messages, and thus the spin updates, are guaranteed by the Time Warp mechanism.

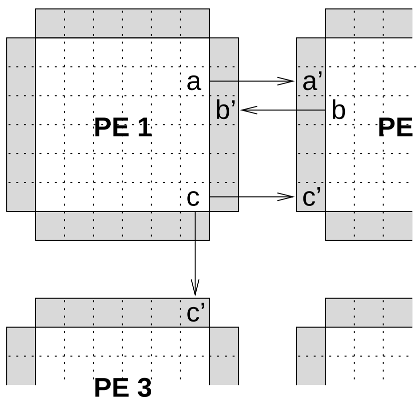

Figure 2: Spatial decomposition of the Ising spin lattice. The grey areas are local copies of neighbor boundary strips. For example, processor PE 2 has a local copy of spin "a" owned by processor PE 1. Processors PE 2 and PE 3 both own a copy of spin "c". The arrows in the figure indicate the event messages sent upon a spin flip.

Asynchronous Cellular Automata, and thus also the Ising spin model, put some additional requirements on the original formulation of the Time Warp method. For example, the Time Warp method, as all optimistic PDES methods, must save its state vector each time an event is executed. The state vector of an spatial decomposed ACA can be arbitrarily large, that is, all the cells in the sub-lattice are part of the state vector. For efficient memory management, we incorporate incremental state saving in the Time Warp method [Overeinder and Sloot, 1993]. Incremental state saving stores not the full state vector, but saves only the changes to the state vector due to the execution of an event, which is only a small fraction of the full state. Besides efficient memory management, incremental state saving also reduces the time overhead related to the memory copy.

With incremental state saving, no full copy of a state vector at a certain simulation time exists in the simulation execution environment. Instead, upon a rollback of a series of events, the state vector is reconstructed by processing the event–partial state collection in reverse order. Although incremental state saving requires less state saving time and memory, there is an increased cost of state reconstruction. In general, the number of rolled back events is a fraction of the number of events executed during forward simulation. The fraction of rolled back events and the time overhead difference between state saving is an order of 10 bytes versus an order of 10^6 bytes, therefore incremental state saving is favorable in spatial decomposed ACA applications.

Alternatively, Lubachevsky [1987] presented a conservative parallel simulation method to simulate ACA. Our perspective and experimental framework essentially differs from his work, as we apply ACA to optimistic methods to study the application parameters that influence the dynamic execution behavior of the Time Warp method.

4 Results

To validate the efficacy of the proposed extensions to the original Time Warp method, we have designed and implemented a parallel discrete event execution environment called the Amsterdam Parallel Simulation System (APSIS). APSIS is by design a portable simulation environment, which runs currently on a number of Unix platforms, such as Solaris, Linux, and BSD/OS, and with various communication libraries, like PVM, MPI, and the lightweight Communication Kernel [Overeinder *et al.*, 1995]. The experiments with the Ising spin model were performed on the Distributed ASCI Supercomputer (DAS). The DAS consists of four wide-area distributed clusters of total 200 Pentium Pro nodes. ATM is used to realize the wide-area interconnection between the clusters, while the Pentium Pro nodes within a cluster are connected with Myrinet system area network technology.

A series of experiments were executed to get insight into the efficiency and scalability behavior of Time Warp, for different problem sizes and Ising spin parameter settings. In the first series of experiments, we study the *absolute efficiency* of the parallel Ising spin simulation compared to the best-known sequential Ising spin simulation for different temperatures, problem sizes, and event granularity (that is, the amount of work per event). The sequential continuous-time Ising spin simulation is basically a Monte Carlo simulation extended with a Poisson arrival process to incorporate time evolution into the model. The Monte Carlo simulation execution mechanism is a lightweight process compared to sequential discrete event simulation execution mechanism. With Monte Carlo simulation there is nearly no overhead involved in the execution of the spin flip trials. A random spin is selected and a trail is executed. With discrete event simulation, a trial is an event that must be scheduled for future execution, that is, inserted into the event list (in general a priority queue). Later, if the scheduled trial is the next pending event, the event is dequeued and the trial is executed. Parallel discrete event simulation includes, besides the event list management overhead, also the state saving and rollback overhead as described in the previous section. The absolute efficiency figures include all these extra overhead costs compared to the sequential Monte Carlo simulation.

The absolute efficiency figures Fig. 3 and Fig. 4 indicate that the parallel performance increases with the problem size and event granularity. The event granularity determines the PDES protocol overhead ratio, apart from synchronization errors. The temperature T for the Ising spin system determines the computation/communication ratio: as the temperature increases, the behavior of the system becomes more dynamic and hence more communication occurs between the nodes. In particular, the efficiency for the relative small lattice size of 32×32 is sensitive to the temperature T, as with small lattice sizes relatively more changes occur along the boundaries. See for example the absolute efficiency figures for the 32×32 (Fig. 3) and 128×128 (Fig. 4) lattices with temperatures $T = 2.0$ and $T = 3.0$. While the figures for $T = 2.0$ are almost equal for both lattices, the figures for $T = 3.0$ differ. Due to the higher temperature of the spin system, more trials are accepted. The fraction of successful trials that must be communicated to the neighbors is proportional to the ratio of boundary lattice points and the total number of lattice points, which is approximate $4/L$, where $N = L \times L$. For lattice size 32×32 the ratio is 0.125, while for lattice size 128×128 the ratio is approximate 0.031.

The event granularity (amount of work per event or in this discussion per trial) is expressed as the amount of extra computational work in terms of a sinus and exponential evaluation. The results for event granularity 0 are for the basic Ising spin system. The results for increasing event granularities give an indication how a similar problem scales as the amount of computational work to evaluate a state change increases.

In Fig. 5 the ratio between the committed events and the total number of processed events is depicted. Committed events are the definite events after the GVT sweeps past their simulation time, see Section 3. The number of processed events is the total number of committed and rolled back events. The ratio of committed events is thus an indication how effective the Time Warp simulation mechanism is to synchronize the parallel simulation processes. From the figure we can see that the ratio of committed events ranges between 0.8 and 0.9, for respectively high and low temperatures. Although the ratios of committed events, and thus the efficacy of Time Warp, do not dramatically differ for the different temperatures, the way it is accomplished does. Figure 6 shows the average rollback length for the different temperatures. Whereas the ratios of committed

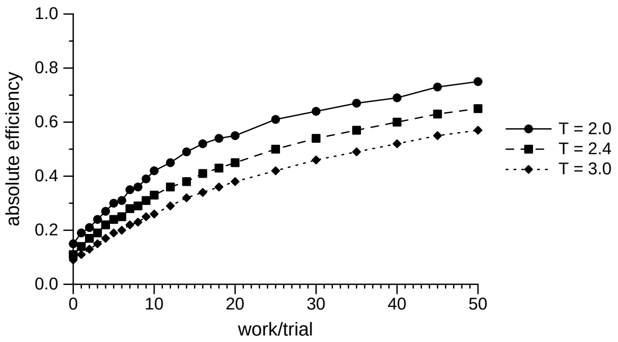

Figure 3: Absolute efficiency versus event granularity (work/trial) for parallel Ising spin simulation of 32×32 spins on 4 processors.

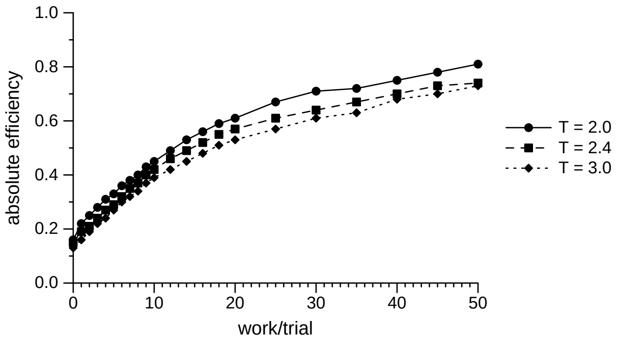

Figure 4: Absolute efficiency versus event granularity (work/trial) for parallel Ising spin simulation of 128×128 spins on 4 processors.

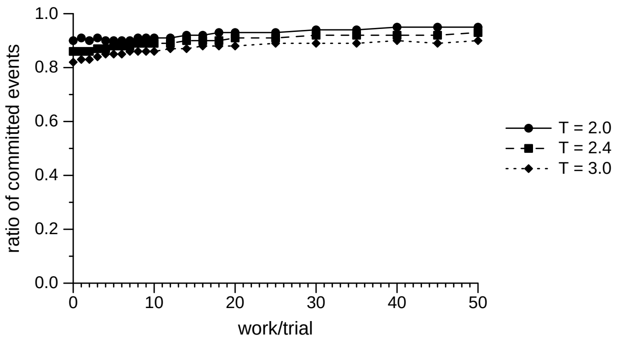

Figure 5: Ratio of committed events versus event granularity (work/trial) for parallel Ising spin simulation of 128×128 spins on 4 processors.

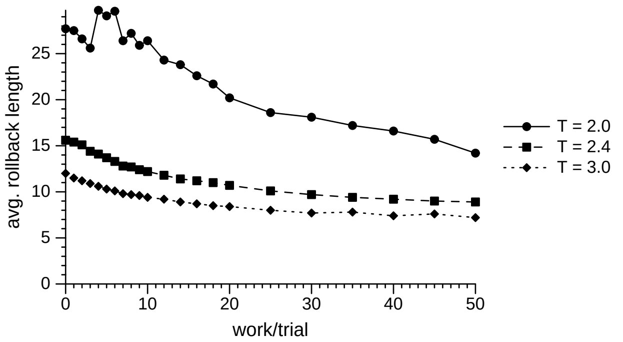

Figure 6: Average rollback length versus event granularity (work/trial) for parallel Ising spin simulation of 128×128 spins on 4 processors.

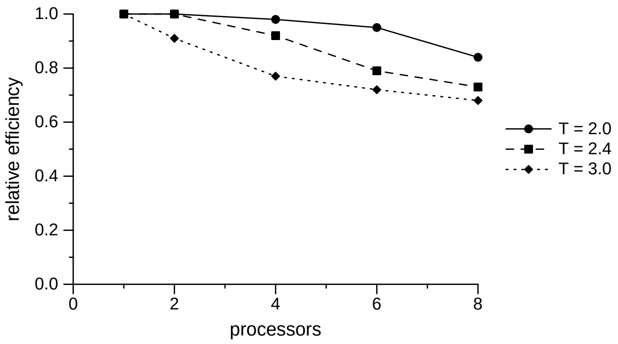

Figure 8: Relative efficiency of parallel Ising spin for lattice size 128×128.

events are almost identical, the average rollback length for $T = 2.0$ is almost twice as large as the average rollback length of $T = 2.4$ and $T = 3.0$. With low temperatures, the fraction of successful trials is small and hence the frequency of synchronization between the neighbors is low. Consequently, the infrequent synchronization makes it likely that the time retardation between the simulation processes is relatively large and one of the processes must rollback this distance in simulated time.

The typical variation of the $T = 2.0$ curve for event granularity 4–10 in Fig. 6 is discussed at the end of this section.

To determine the *relative efficiency* and *scalability* of the parallel Ising spin implementation, the execution time of the parallel simulation on one processor is compared with the execution time on different number of processors. Figure 7 shows the relation between execution time and the number of processors for a fixed problem size. Together with the results from Fig. 8, we can see that the parallel Ising spin for $T = 2.0$ scales almost linearly up to 6 processors, but eventually drops to a relative efficiency of 0.83 for 8 processors. For temperature $T = 3.0$ the relative efficiency decreases gradually to 0.68 for 8 processors. The decreasing efficiency is mainly due to the increased costs to synchronize the parallel processes. With the increase of the number of processors, the time period necessary to synchronize the parallel simulation processes also increases.

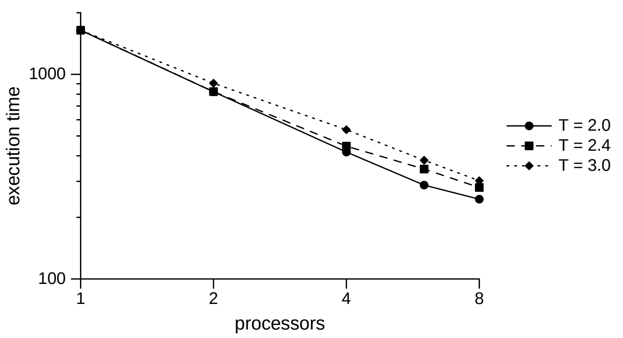

Figure 7: Log-log plot of the scalability of parallel Ising spin for lattice size 128×128.

The influence of the number of processors on the synchronization of the parallel simulation processes is further investigated. With parallel executions up to six processors, the influence of the increase in temperature is relatively small, but with eight or more processors, the Time Warp method requires more time to synchronize the computation over the processors. The increased overhead of the Time Warp method is directly apparent from the number of rolled back events, which increases up to 40%–60%.

From the three parameters determining the parallel performance—the event granularity, the computation/communication ratio, and the number of rollbacks—the number of rollbacks seems the most predominant. The detailed execution trace shown in Fig. 9 and Fig. 10 exposes that during periods of synchronization, it takes up to ten seconds to resynchronize the parallel computation before any progress can be made (each GVT update takes 0.05 seconds in this execution). The event rate is the number of events that are committed per second, and in this respect a measure for progress. During normal operation, the simulation reaches a event rate of 19 000 events per second. In Fig. 9 we can identify four serious glitches in the event rate, around GVT update period 400, 950, 1150, and 1750. In these periods, the event rate drops to 10% of the steady state performance (about 2000 events per second). In particular the period centered around 1750 takes about 10 seconds to resynchronize (see Fig. 10) and weight heavy upon the parallel performance.

The periods of resynchronization are a typical example of thrashing, where most of the time is spent on simulation rollback instead of forward simulation. While one simulation process rolls back, another process advances in simulation time. When the rollback is completed, the simulation process restarts with event execution and as a result sends event messages to neighboring processes. These event messages arrive in the simulation past of the neighboring processes, and trigger a rollback, etc., etc., until the simulation processes are in synchrony. The thrashing behavior is a combination of a number of factors: number of processors, lattice size, event granularity, and temperature (synchronization frequency). This behavior is also appar-

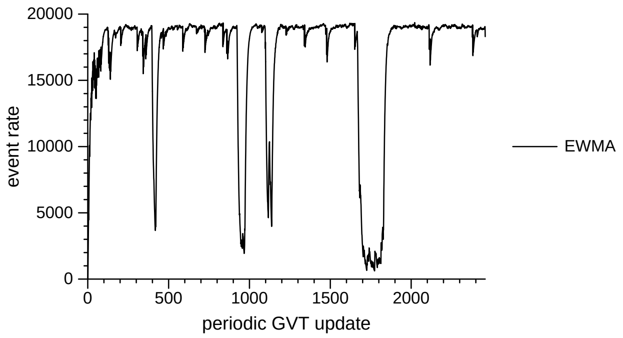

Figure 9: Simulation progress (event rate) during execution. Parallel Ising spin with lattice size 256×256 on 6 processors. The curve is smoothed by taking the exponential weighted moving average (EWMA), as the EWMA follows the dynamic behavior accurately and can be efficiently computed.

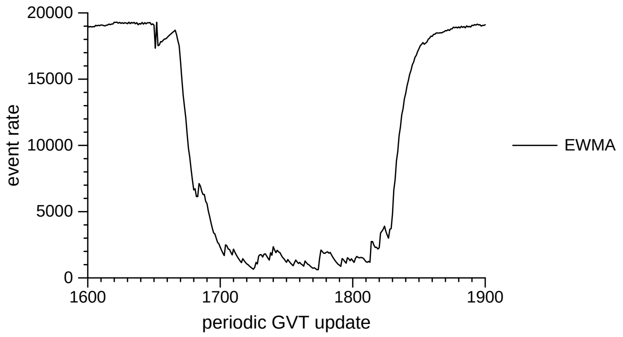

Figure 10: Simulation progress (event rate) during execution (detail of Fig. 9 between 1600 and 1900).

ent from Fig. 6 for temperature $T = 2.0$. Although it not significantly influences the parallel performance, the sudden increase in average rollback length for event granularity range 4–10 is an indication of instability.

To shorten these periods of resynchronization, the optimism of the protocol must be throttled, that is, the simulation execution mechanism should not execute events that lie in the remote future as it is likely that these events have to be rolled back eventually. In effect, the progress of the individual simulation process should be bound to a limited simulation time window. In this way, the processes are forced to synchronize with each other in a short time frame, after which the simulation can continue as before. The effect of a simulation time window can be clearly seen in Fig. 11. For simulation time window 3000, the absolute efficiency drops abruptly for the event granularity range 4–10, after which the absolute efficiency figure slowly recovers (see also Fig. 6). If the optimism is throttled with a simulation time window of 2000, the absolute efficiency starts below the results for the time window 3000 in the event granularity range 0–3. However, in the 4–25 range the throttled simulation with time window 2000 performs superior to the time window 3000 simulation. Afterwards, their results are almost identical. This observation motivates an adaptive approach that recognizes the periods of resynchronization and adjusts the simulation time window dynamically, such that the efficiency is improved significantly.

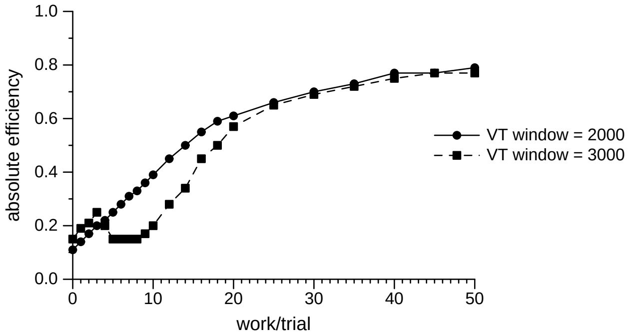

Figure 11: Absolute efficiency for different Virtual Time windows. Parallel Ising spin with lattice size 256×256 and $T = 2.0$ on 6 processors.

5 Conclusions

The application of optimistic parallel discrete event simulation methods such as Time Warp to asynchronous cellular automata is in potential a viable approach to parallelize the simulation. However, two essential extensions to the Time Warp method have to be included: incremental state saving and optimism control (throttling). The results show that given a fast communication network such as Myrinet, Time Warp is viable alternative for parallel simulation. In particular, low communication latencies are essential to achieve performance, as the event messages are small.

In order to design and implement effective optimism control, we will further study the different phases in the dynamic behavior of Time Warp. The formulation of simple though applicable metrics to control the amount of optimism in the Time Warp method determines the success of the mechanism.

References

H. Bersini and V. Detours. Asynchrony induces stability in cellular automata based models. In *Proceedings of the IVth Conference on Artificial Life*, pages 382–387, Cambridge, MA, July 1994.

K. M. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, September 1979.

R. J. Glauber. Time-dependent statistics of the Ising model. *Journal of Mathematical Physics*, 4(2):294–307, February 1963.

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

M. Livny. A study of parallelism in distributed simulation. In *Proceedings of the 1985 SCS Multiconference on Distributed Simulation*, pages 94–98, San Diego, CA, January 1985.

B. D. Lubachevsky. Efficient parallel simulation of asynchronous cellular arrays. *Complex Systems*, 1(6):1099–1123, December 1987.

E. D. Lumer and G. Nicolis. Synchronous versus asynchronous dynamics in spatially distributed systems. *Physica D*, 71:440–452, 1994.

B. J. Overeinder and P. M. A. Sloot. Application of Time Warp to parallel simulations with asynchronous cellular automata. In *Proceedings of the 1993 European Simulation Symposium*, pages 397–402, Delft, The Netherlands, October 1993.

B. J. Overeinder and P. M. A. Sloot. Parallel performance evaluation through critical path analysis. In *High-Performance Computing and Networking (HPCN Europe '95)*, number 919 in LNCS, pages 634–639. Springer-Verlag, May 1995.

B. J. Overeinder, J. J. J. Vesseur, F. v/d Linden, and P. M. A. Sloot. A communication kernel for parallel programming support on a massively parallel processor system. In *Proceedings of the Workshop on Parallel Programming and Computation (ZEUS'95) and the 4th Nordic Transputer Conference (NTUG'95)*, pages 259–266, Linkøping, Sweden, May 1995.

B. P. Zeigler. Discrete event models for cell space simulation. *International Journal of Theoretical Physics*, 21(6/7):573–588, 1982.

Biography

Benno Overeinder received his Master's degree (cum laude) in Computer Science at the University of Amsterdam. He is currently a senior researcher at the department of Computer Science. His research focus is parallel simulation of discrete event systems. He has published various papers on parallel discrete event simulation, optimistic parallel simulation methods, parallel performance analysis, and run-time support systems for parallel and cluster machines. His research interests are methods and paradigms in parallel simulation, parallel program environments, and load balancing.

He can be reached via email: `bjo@wins.uva.nl`, or URL: `http://wins.uva.nl/~bjo/`

Peter Sloot received a Master's degree in Chemical Physics and Theoretical Physics at the University of Amsterdam. His PhD work was carried out in collaboration with the Dutch Cancer Institute and the University of Amsterdam. Currently he is a professor of Computational Physics at the University of Amsterdam. His main interest is in the modeling and simulation of complex dynamical systems. He is the project leader of the Parallel Scientific Computing and Simulation group at the UvA (`http://www.wins.uva.nl/research/pscs/`), co-founder of the Computational Science Center Amsterdam (`http://www.beta.uva.nl/institutes/-csa/`) and a consultant in a number of national and international high performance computing initiatives.