NPSI adaptive synchronization algorithms for PDES

Sudhir Srinivasan
Paul F. Reynolds, Jr.

Computer Science Report No. CS-94-44
Issued Nov. 8, 1994; revised April 6, 1995

NPSI adaptive synchronization algorithms for PDES

Sudhir Srinivasan Paul F. Reynolds, Jr.
Department of Computer Science
Olsson Hall, University of Virginia
Charlottesville, VA 22903.
Phone: (804) 982-2200 Fax: (804) 982-2214
Email: {ss7a|pfr}@uvacs.cs.Virginia.EDU

Abstract

Research in parallel discrete event simulation indicates that neither purely conservative nor purely optimistic synchronization algorithms will perform well consistently. We survey several new approaches that attempt to improve performance by limiting optimistic execution. In most of these, the criterion for limiting optimism is static or based on local information, which conflicts with the dynamic nature of discrete event simulations. We contend that an adaptive approach based on low cost near-perfect system state information is the most likely to yield a consistently efficient synchronization algorithm. We suggest a framework by which NPSI (near-perfect state information) adaptive protocols could be designed and describe the first such protocol - Elastic Time Algorithm. We present performance results from an implementation of this algorithm which show that adaptive protocols based on the use of NPSI are promising. In particular, we show that NPSI adaptive protocols have the capacity to be more efficient than Time Warp in both time *and* space. We identify major issues in the design and usability of NPSI protocols and discuss ongoing research.

Keywords: Modeling methodology, parallel simulation, adaptive protocols, limited optimism, Time Warp, Elastic Time Algorithm, reduction networks.

Submitted to: *1995 Winter Simulation Conference*

1 Introduction

The state-of-the-art in parallel discrete event simulation (PDES) is captured well in the recent survey by Fujimoto [Fuji90]. Although this survey is dated eleven years since the inception of PDES, it is dedicated entirely to the discussion of synchronization mechanisms for PDES. *The synchronization problem has remained the central challenge in PDES.* A large number of protocols have been proposed to solve the problem and many have reported good performance. However, a general solution which can be applied to a wide range of simulations still eludes the community. A more recent survey [FuNi92] confirms that this state has not changed. We present initial results of research into a novel and unique class of protocols - those in which LP's adapt their behavior dynamically to changes in the simulation using low-cost near-perfect system state information. We refer to these protocols as *NPSI (Near-Perfect State Information) adaptive protocols*. Given the inherently dynamic nature of simulations [NiRe90], we believe NPSI adaptive protocols offer the best hope of finding a consistently efficient, general protocol.

We assume familiarity with the common approach to PDES [Fuji90], namely the partitioning of a simulation into logical processes (LP's). Each LP is itself a sequential discrete event simulator which can schedule events at other LP's using timestamped messages. The LP's must execute events, whether generated internally or scheduled by other LP's, without violating causality constraints (effectively). Typically, this is the responsibility of a *protocol*. The nine design variables in [Reyn88] define the design space for protocols. An *aggressive* protocol is one which executes events without the guarantee of freedom from causality errors. A protocol has *risk* if it propagates messages based on aggressive or inaccurate computation. Based on these two variables, a *conservative* protocol is non-aggressive and without risk, while an *optimistic* protocol is aggressive and with risk. These two categories form the end points of a spectrum of protocols with limited optimism (aggressiveness and risk).

Adaptiveness for PDES protocols is defined in [Reyn88] as the capability of a protocol to modify the bindings of one or more of its design variables during the simulation. Several protocols have been proposed that limit optimism, but most of them are not adaptive by this definition. In order to make dynamic decisions based on system state, processes must be provided with near-perfect state information at low cost. Gathering such information by exchanging messages through the communication network of a multi-processor as the simulation proceeds is infeasible due to the relatively high cost of such communication. This has been the major obstacle in the study of NPSI adaptive protocols.

To overcome this obstacle, we assume an asynchronous dynamic feedback system which provides each LP with a near-perfect snapshot of the system state (in a reduced form) at very low cost. This is done for three reasons: (i) we believe NPSI adaptive protocols have significant potential and thus warrant a detailed study, (ii) a feasible implementation for the feedback system is a high-speed reduction network, and (iii) an implementation exists for such a reduction network [RePS92]. As we shall see later, our first NPSI adaptive protocol shows significant improvements over pure Time Warp [Jeff85], in terms of both time and space.

2 Previous work

We categorize protocols that limit optimism based on the criterion for limiting optimism:

- *Window based*: Only those events within a (common or independent) window are executed aggressively. Similarly only those messages within a (possibly different) window are sent out. This ensures that all of the LP's remain close to each other in logical time. Uncontrolled echoing and cascading rollbacks cannot occur. Examples are [SoBW88, LuWS89, ReJe89, McAf90, TuXu92, Dick93, Stei93].
- *Space based*: Here, the boundaries for limiting optmism are spatial rather than temporal. In general, the processors are divided into clusters which use Time Warp internally. Interaction among clusters follows some non-aggressive protocol. Examples are [Gima89, RaAT93]. An interesting special case is when each cluster contains exactly one LP; then we have a risk-free system [DiRe90, Mehl91, Stei91, Bell93].
- *Penalty based*: This approach assumes that the recent past is a good predictor of the near future. Based on their recent behavior, some LP's are penalized (and consequently block) while others are favored (and consequently continue). Examples are [ReJe89, BaHo90]. In [Madi93], the penalty is based on the difference between an LP's logical clock and *estimates* of the logical clocks of other LP's.
- *Knowledge based:* The basic idea is to contain the propagation of incorrect computation as soon as it is determined that the computation is incorrect. When an LP suffers a primary rollback, any aggressive processing it has done and sent to other LP's is potentially incorrect. So it

broadcasts a message to all "affected" LP's informing them that their computation is potentially incorrect. These LP's then limit their optimistic processing. Examples are [MaWM88, PrSu91].

- *Probabilistic*: A special process periodically makes a probabilistic decision to send out synchronization messages to all LP's, forcing them to synchronize.

Most of these protocols have demonstrated better performance than pure Time Warp under specific tests. However, we expect that all of them will have limited performance in the general case due to one or more of the following: (i) the criterion for limiting optimism (window size, cluster size, penalty thresholds, probabilities, etc.) is predetermined (ii) the decision to limit optimism is based solely on local history (iii) the LP's are loosely synchronous (i.e. not asynchronous).

Recently, three protocols have been proposed which we categorize as *state based* protocols. They differ from those listed above in two significant ways: first, they are truly adaptive in that the LP's continually adjust their optimism, and second, adaptive decisions are based on state information which, although available locally, is directly affected by the actions of other LP's. Thus, these protocols are similar to NPSI protocols. The first two of these [HaTr94, FeTr94] are similar to each other in that they both utilize channel information to decide when and for how long LP's should wait. Intuitively, it may be argued that our approach has an advantage over these because LP's in NPSI protocols can receive information from all of their predecessors whereas channel protocols provide information only about immediate predecessors. The benefit comes from the fact that in a channel protocol, information has to percolate through the predecessors of an LP before it reaches that LP. During this time, the LP may have moved farther ahead than it should have. Moreover, the protocols in [HaTr94, FeTr94] cannot determine if a predecessor is rolling back until they receive a null- or anti-message conveying that information. NPSI protocols are able to determine this information earlier. The third protocol in this class [DaFu94] uses memory consumption as the basis for limiting optimism. The protocol limits the memory consumption of LP's adaptively and consequently, also limits their optimism. It is based on a memory management protocol such as Cancelback or Artificial Rollback. We take the opposite approach in that NPSI adaptive protocols directly limit the optimism of LP's and consequently limit their memory consumption. Our approach has the potential for eliminating the need for costly memory management schemes completely.

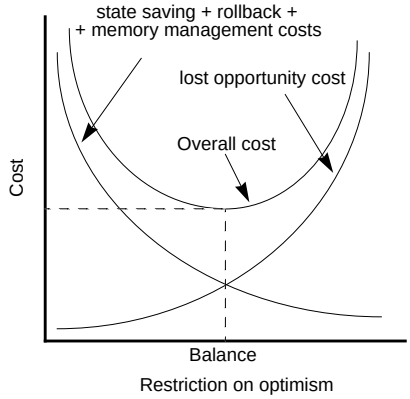


Figure 1 : Trade-off introduced by limiting optimism

3 Effect of limiting optimism

Aggressive protocols incur three direct costs: state saving cost, rollback cost and memory management cost. Limiting aggressiveness and risk introduces a fourth cost: *lost opportunity cost*, characterized by the potential loss in performance when an LP stops executing events or sending messages even though it is safe for it to continue. In order to obtain good performance, protocols must attempt to minimize:

state saving cost + rollback cost +
memory management cost + lost opportunity cost

While limiting optimism tends to decrease the first three of these, it also tends to increase the fourth, leading to the typical trade-off shown in Figure 1. This trade-off must be balanced properly in order to obtain the best possible performance. To do so, protocols must attempt to distinguish incorrect computations from correct ones and limit the propagation of the former while allowing the latter to progress.

PDES's are known to be very dynamic in nature, i.e. the locality of events in the system changes as the simulation progresses. Typically, this is due to the fact that simulated systems have some form of information flow which is translated in the simulator into a causal chain of events among LP's. Since the propagation of such chains is based on probabilistic decisions and input parameters of the simulation, it is impossible to determine the flows a priori (except in special cases). This suggests that in order to balance the trade-off above, a protocol must also be dynamic, adapting its behavior in response to observed changes in the system.

4 Near-perfect state information

We believe the two key requirements for a protocol to be consistently efficient are that it is dynamic and that it uses feedback from the simulation to adapt. Ideally, these requirements should be met by providing LP's with perfect state information, such that any relevant change in system state is visible instantaneously. However, this is impossible to achieve due to various latencies in computing distributed snapshots. Therefore, a good approximation of perfect state information is required.

We take the direct approach by assuming the existence of an asynchronous dynamic *feedback system* which operates asynchronously with respect to the LP's and provides them with near-perfect information at almost no cost. Clearly, the feedback system cannot operate over the native communication system of the multi-processor executing the PDES. One solution is to use a high-speed reduction network. A global reduction network is one in which binary, associative operations (minimum, summation, etc.) are used to reduce state information so that, say, the minimum simulation clock value can be computed across all LP's. Our experience with the design, construction and testing of a global reduction network [RePS92] suggests that a production version of such a network can operate at very high speeds (less than 20 nanoseconds per stage in the tree). These low latencies, combined with its pipelined, tree structure make such a network scalable up to thousands of processors. We have used this reduction network for the perfromance analysis described later.

5 NPSI adaptive protocols

NPSI adaptive protocols are optimistic protocols (such as Time Warp) in which the aggressiveness and risk are controlled dynamically using near-perfect state information. There are two phases in the design of NPSI adaptive protocols:

- identifying the information on which the decision to limit optimism is to be based;
- designing the mechanism that translates this information into control over an LP's optimism;

Clearly, there are numerous choices for each of these. In order to facilitate independent study of each, we uncouple them by introducing a quantity called *error potential* (EP_i) associated with each LP_i. The value of EP_i is used to control LP_i's optimism. The framework we propose is shown in Figure 2. The NPSI adaptive protocol keeps each EP_i up-to-date as the simulation progresses, by evaluating M_1 at high frequency using state information it receives from the feedback system.

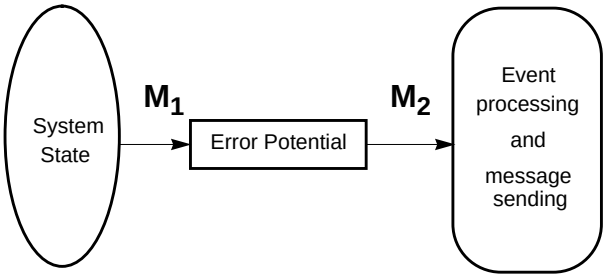


Figure 2 : General framework for adaptive protocols

Similarly, M_2 reflects new values of EP_i in the event execution and communication rates dynamically. Different protocols may be devised by changing M_1 and M_2. The goal of our research is to design mappings M_1 and M_2 such that their combination forms an adaptive protocol that performs well consistently.

To achieve optimal performance, LP's must be able to identify computations that will be rendered incorrect in the future and limit their propagation. Obviously, this requires the ability to predict the future which is difficult at best. In the framework described above, the error potential is a way of labeling computation as *potentially incorrect*. EP_i is simply a number which indicates the likelihood of LP_i's computation becoming incorrect in the near future: the higher the value of EP_i, the higher this likelihood. The key to consistently good performance is to devise an M_1 which will predict the nature of the LP's computation (i.e. whether that computation will be rolled back or not) accurately most of the time. An inaccurate M_1 can produce a low value of EP when the computation is erroneous, resulting in higher rollback costs, or a high value of EP when the computation is correct, resulting in higher lost opportunity cost.

M_2 must be such that higher values of EP_i result in lesser aggressiveness and risk at LP_i. A simple scheme is to establish a *threshold* such that if the value of EP exceeds this threshold, event execution and communications are suspended until EP falls below the threshold again. A more sophisticated scheme would reduce the event execution and communication rates gradually as EP increases. This deceleration can be achieved by inserting delays at appropriate points. M_2 would then be a function that maps EP to a wall-clock time delay.

6 Elastic Time Algorithm

We describe the specifics of the *elastic time algorithm* (ETA), the first NPSI adaptive protocol. ETA has been implemented and the results of preliminary performance analysis are presented here. The protocol can be specified completely by describing the two mappings M_1 and M_2.

6.1 M_1: computing EP_i

M_1 is the following function:

$$M_1 : EP_i = \text{logical clock}_i - GVT$$

where GVT (global virtual time [Jeff85]) is defined as the minimum of the logical clocks of all LP's and the timestamps of any messages in transit. Thus, ETA is based on near-perfect values of GVT being available to the LP's. We use the reduction network to provide accurate GVT at high frequency and low cost [SrRe93].

The rationale behind this M_1 is that if an LP is far ahead of others, it is likely to be rolled back soon and therefore should be slowed down. The algorithm's name derives from the following analogy: one may imagine an LP and its predecessors as pins moving along the logical time line with an elastic band around them. The farther an LP moves away from the rest, the slower its progress due to the restraining pull of the elastic band. When the LP farthest behind moves forward, the restraint on the LP farthest ahead is reduced so that it may quicken its pace again. As the load locality changes among LP's, this scheme adapts by restraining those LP's far ahead in logical time.

6.2 M_2: controlling optimism

Given a value of EP_i computed by M_1, we use the following function to scale it to a delay value, δ_i:

$$\delta_i = s \cdot \frac{EP_i}{MaxEP_i}$$

where $MaxEP_i$ is the maximum value of EP_i observed thus far and *s* is a scaling factor (we defer discussion on *s* to Section 7). The event processing loop of a Time Warp LP is modified as shown in Figure 3 to incorporate adaptive delaying. Some interesting features of this M_2 must be noted:

i) The blocked state of the LP is not "opaque" in that while in this blocked state, the LP observes its input channels for messages that may cause it to rollback. If such a message arrives, the waiting is aborted and rollback is initiated. Also, the LP may perform useful work in the blocked state such as converting messages that it receives (that do *not* cause a rollback) into future events.

ii) EP_i and δ_i are updated in each iteration of the loop in the blocked state.

iii) The waiting scheme is not memoryless. The wait timer is started only once at the beginning of the wait period. As LP_i goes through successive iterations of the wait-loop, its wait time increases whereas δ_i decreases (because its logical clock is constant and GVT is monotonically increasing). Thus, LP_i interprets each new value of δ_i as an estimate of the amount of time it should have waited since the start of the waiting period. When this value becomes smaller than the time it has actually waited, LP_i exits the wait-loop.

iv) This M_2 provides direct control over an LP's aggressiveness only. The LP's risk is controlled only to the extent that while in a blocked state, the LP does not send out messages. Clearly, it is possible (and perhaps desirable) to have a separate mechanism to limit risk.

```
While there are events to be processed
   Update logical clock and process event
   Start wait timer
   do
       Receive messages and exit from loop if
           there is a message that will cause
           rollback or if the message has
           timestamp = GVT
       Update GVT, EP_i and δ
       Read wait timer
   while wait timer value < δ
   Rollback if necessary
   Process messages
   Save state
   Collect fossils
endwhile
```

Figure 3 - Event processing loop with adaptive waiting

Putting ETA in perspective with previous protocols, we observe that it is a state based protocol that is similar to window based protocols with two significant differences:

- It is completely asynchronous - there are no barrier synchronizations to negotiate windows
- Each LP's logical time window may be considered infinite but the event execution rate drops rapidly as the LP moves farther away from the base of the window

6.3 Performance analysis

We present the results of performance tests on ETA. We describe the testing environment, followed by test cases, metrics used and results.

Hardware

The hardware consists of a cluster of four Sparc 2 workstations connected by Ethernet and a *parallel reduction network* (PRN) that we have designed and built [RePS92]. Each workstation communicates with its own

auxiliary processor (AP) through dual-ported memory (DPRAM). The four AP's are connected to the PRN which computes and disseminates globally reduced values at very high speeds. The AP's are responsible for sending values into the PRN and reading its global output. Dedicated processors have been used to offload this task from the workstations so as to keep interference with the simulation to a minimum. The workstations communicate relevant state changes to the AP's through the DPRAM. The AP's process this information and send it to the PRN. Also, the AP's read the globally reduced output of the PRN and propagate relevant parts of it to the workstation.

Software

The two primary software components are the Time Warp LP's and the GVT computation algorithms executing on the workstations and the AP's respectively. The LP's use aggressive cancellation and do not support event pre-emption; they include the M_1 and M_2 for ETA. The GVT computation algorithm for the AP's is described in [SrRe93].

Test cases

We employ a parameterized synthetic workload generator to create our test cases with the important parameters being event execution time, average timestamp increment, state saving cost, communication topology and distributions, number of local events per message, and state saving and fossil collection frequencies. Time consuming actions such as event executions and state saving are simulated by busy loops; all other mechanisms such as rollback, restoring state, sending antimessages and fossil collection and data structures such as saved state list and antimessage list are implemented in detail. A synthetic workload generator is used instead of actual applications because it allows us to mimic those applications without the excessive time and effort required to implement each of them. Our workload generator is very similar to the PHOLD model [DaFu94].

We tested ETA on several workloads and observed that it outperformed Time Warp on all of them. We present results for the following:

a) **Workload 1** consists of four LP's with the T_1 communication topology shown in Figure 4 (a torus). The number on an arc is the probability that a generated message is sent along that arc. The workload is self-initiating (each event schedules the next local event [Nico91]) and the probability of an LP sending a message after an event is 0.2. Messages may cause rollbacks, but do not schedule events.

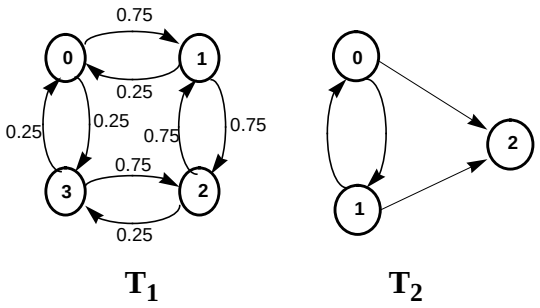


Figure 4 - Communication topologies for test workloads

b) **Workload 2** also uses topology T_1 but is message initiating (messages cause events to be scheduled). Each message schedules a job event. Execution of a job event creates an output event, which generates a message. Every LP has 25 events initially. Thus, this workload resembles a closed queueing network with density 25.

c) **Workload 3** is an implementation of the *echoing* example described in [LuWS89]. LP_0 and LP_1 execute in self-initiating mode, sending messages to LP_2 after each event. In addition, there is a single message-initiating event that is at LP_0 initially. Upon processing this event, LP_0 sends a message to LP_1. This causes LP_1 to schedule an event, possibly after rolling back. After processing this event, LP_1 sends a message back to LP_0 and the cycle repeats. The idea is that this "causal" message rolls back its receiver and while the receiver is rolling back, the sender continues to move ahead, sending messages to LP_2. Since the cost of rollback is proportional to the depth of rollback and the event execution time is comparable to the cost of rollback, LP_0 and LP_1 spend increasing amounts of time rolling back and the progress of the simulation diminishes rapidly.

For all three workloads, the mean event execution time is 100 μs, the mean state saving time is 25 μs and state saving and fossil collection are performed after each event.

Metrics

One of the important aspects of ETA is the scaling factor, *s* in M_2. This factor is required to translate the value of EP_i which is in logical time units into a delay in real-time. The range of EP_i is dependent on the logical time increments exhibited by the LP's and the rate at which they execute events, send messages, etc. Since these factors differ considerably across applications, it is

expected that the value of *s* that maximizes performance will be different for each application (and may in fact vary during a single simulation). Thus, *s* is a good candidate for the choice of independent variable in the performance tests. The problem of determining a good value of *s* dynamically is discussed in Section 7.

The most important metric, of course, is completion time. Yet another metric is rollback time, which is the time an LP spends rolling back (including state restoration and sending of antimessages). Since limiting of aggressiveness is expected to reduce rollback time, ideally to zero, this metric seems to be a good indicator of how close the actual performance is to the goal.

Results

Graphs 1, 2 and 3 show the variation of completion time and rollback time with the scaling factor *s* for the three workloads respectively. When s=0, δ=0 and ETA is essentially identical to Time Warp. As *s* increases, the aggressiveness and risk of the LP's decrease. The completion time in graph 1 has the expected parabolic shape based on the trade-off shown in Figure 1. The absence of such a parabolic shape in graphs 2 and 3 is due to the nature of the workloads. There is very little concurrency in workload 2 owing to the small event execution times, the high latency of Ethernet and high connectivity of topology T_1. A simple critical path analysis of workload 3 shows that it is also inherently sequential. Generally, the waiting period at each LP increases with *s*, approaching sequential execution. However, a threshold is reached such that further increase in *s* does not increase waiting due to the following: consider LP_i waiting to execute an event. When the last event with timestamp less than LP_i's logical clock has been executed and all messages with timestamps less than LP_i's logical clock have been received, GVT will equal LP_i's logical clock causing EP_i (and δ_i) to drop to zero and LP_i to come out of waiting in the next iteration. This analysis demonstrates that ETA has the capability to approach sequential execution but not become arbitrarily slower than it during phases in a simulation where there is so little concurrency that parallel execution is detrimental.

The enormous reduction in completion time in graph 3 demonstrates that ETA can avoid unstable situations such as echoing. The instability is manifest in the large variations in both curves in graph 3 for small values of *s*. It is important to note that in all three graphs, the rollback time is close to zero when completion time is minimized. This suggests that the reduction achieved by ETA is close to the maximum possible.

Memory considerations

In the general case, memory consumption appears to be a serious problem with Time Warp. Excessive memory consumption is usually due to so-called *runaway processes* - LP's that execute events faster than other LP's so that (i) they have a large number of processed events as yet uncommitted, and (ii) they schedule a large number of unprocessed events at other LP's. Several memory management schemes have been proposed to reclaim memory from future events (since this memory cannot be reclaimed by fossil collection) [Lin92]. We expect that NPSI adaptive protocols will eliminate the need for these schemes for two reasons. First, any approach that limits the aggressiveness of LP's inherently reduces memory requirements by not permitting runaway processes to get too far ahead. Second, an adaptive, memory-based flow control scheme [DaFu94] can be integrated naturally with NPSI protocols by including information about the memory availability of down-stream LP's (perhaps immediate successors only) in the mapping M_1. In this way, an LP could slow down when any of its successors is at the risk of running out of memory.

ETA does not include information about successors' memory usage in its M_1. In spite of this, we observed significant savings in memory requirements due to the limited aggressiveness. For the three test workloads, we measured the average and maximum size of the saved-state list (in terms of the number of entries in the state list) as an indicator of the LP's processed but uncommitted memory requirements. The maximum size is important because the workstation must have sufficient memory to store that much state even if it is a rare occurrence.

Graphs 4, 5 and 6 illustrate the substantial savings in memory consumption (despite the fact that only state-saving space is being considered here). In the two stable workloads (graphs 4 and 5), the savings in maximum state list size is noteworthy. In the echoing workload (graph 6), it is interesting to observe that the average state list size is very high. This is because for small values of *s*, the state lists grow unboundedly due to instability.

Given the definition of ETA, an LP that is only a source of messages (i.e. that has no predecessors) will never have its optimism throttled. This does not pose a problem when the workload is homogenous. However, if the source is faster than its successors, it becomes a runaway process which will not be limited by ETA. Thus, it is important that the memory-based flow control described earlier which solves this problem, be included in ETA. We expect to report on this in the near future.

7 Further issues

The results described in Section 6 are very encouraging. They demonstrate clearly the viability of NPSI adaptive protocols. Combined with the fact that we have not yet encountered a workload for which ETA performs worse than pure Time Warp (albeit with the proper selection of *s*), these results strengthen our conviction that NPSI adaptive protocols can be universally efficient. However, some important issues must be resolved before any definitive conclusions can be drawn. In the following, we discuss these issues in some detail. Most of these are being explored currently.

Tunable parameters

One of the conclusions that must be drawn from the results presented in Section 6.3 is that the scaling factor *s* must be chosen properly for the protocol to perform well. Too small a value will not reduce rollback costs sufficiently while too large a value will introduce lost opportunity cost. The "best" value of *s* depends on the particular application and hence cannot be determined a priori in general. The user may be able to provide some hints about this value; on the other hand, there is no theory to aid the user. We refer to parameters like *s* as *tunable parameters*.

While *s* is an artifact of the particular M_2 chosen for ETA, there are reasons to believe that such tunable parameters will exist in any adaptive scheme:

- By definition, adaptive schemes are expected to react to changes in the system. If the scheme is a general one (i.e. one that will be used with a wide range of applications), it must necessarily determine the context of the particular application in order to interpret the state information it obtains. This process seems to entail tunable parameters such as sampling intervals, size of history, filter constants and autoscaling factors.
- A problem similar to adaptive synchronization of PDES's is adaptive load balancing/sharing in parallel systems. It is well known that good load balancing requires good estimation of tunable parameters such as thresholds and maximum number of transfer-hops.

Not surprisingly, we observe that all existing adaptive protocols admit tunable parameters of some sort (the time-window-bound in [TuXu92], N_1 and N_2 in [Stei93], cluster size in [RaAT93], blocking window size in [BaHo90], the scaling factor c_i in [HaTr94] and M_f and M_p in [DaFu94]). Clearly, there is a need to invent a scheme by which adaptive protocols may tune the values of such parameters dynamically. The Adaptive Time Warp algorithm [BaHo90] computes its blocking period based on a particular logistic response function. Thus its efficacy depends on how accurately this function reflects reality. Moreover, it is an entirely local optimization. The Local Adaptive Protocol [HaTr94] also uses a local optimization strategy in which the parameter is tuned based on observed rate of progress of virtual time per channel. This has the drawback that it cannot distinguish between a decrease in porgress rate due to excessive rollbacks from one that occurs inherently in the application. The scheme of [DaFu94] tunes two parameters, M_f and M_p dynamically based on an analysis of the flow of memory buffers in a Time Warp simulation. This scheme is very specific to the nature of the tunable parameters.

The general solution to this problem (and the one we adopt) is to monitor the progress of the simulation in order to determine how well the current values for tunable parameters are performing. The key is to identify metrics that can be obtained by each LP during the simulation and that reflect the progress of the computation accurately. Once again, an analogy may be drawn with load balancing where a good load index is essential [FeZh87].

In parallel simulation, as in many parallel computations, local (per processor) optmization does not imply global optmization (in fact it may be detrimental). Consequently, a purely local metric may not be advisable. On the other hand, if a single global metric is used, an LP may not be able to determine if a change in the measured value occurred due to some action that it performed (or that it should have performed). Therefore, we propose a combination of two metrics:

- R_C - the *maximum* rate of commitment of events over all LP's. We believe this global metric to be a better indicator of global progress than the rate of advance of GVT because the base of reference for R_C is the same for all applications unlike that for GVT. By taking the maximum commitment rate over all LP's we take into account the case where events are distributed unevenly over the logical time line at different LP's.
- P_{RB} - the percentage of events rolled back at an LP. This local metric combines the number and depth of rollbacks and, thus, appears to be a good indicator of whether an LP is a cause for slow progress.

R_C indicates the overall progress of the simulation while P_{RB} indicates to an LP whether it is a potential cause for any observed poor global performance. Table 1 summarizes the actions an LP takes under various

situations. In terms of our tunable parameter *s*, optimism is increased (decreased) by decreasing (increasing) *s*. R_C, which is a global metric, can be computed for NPSI protocols accurately at very low cost to the simulation using the feedback system. We expect to have performance results for a scheme that uses these metrics in the near future.

In [PaWi93], the authors suggest the use of similar metrics. Similar to R_C, they define W_i as the rate of commitment of events at LP_i in one GVT cycle. Thus, W_i is a local metric. While [PaWi93] argue for an indicator of rollback behavior, they do not suggest one - the algorithm they propose uses W_i only.

R_C	P_{RB}	Action
Low	Low	Increase aggressiveness/risk rapidly
Low	High	Decrease aggressiveness/risk rapidly
High	Low	Increase aggressiveness/risk gradually
High	High	Decrease aggressiveness/risk gradually

Table 1 - Actions for dynamic tuning of parameters

Larger systems

For the purpose of experimentation with implementations of NPSI adaptive protocols, the prototype reduction network restricts us to four processors. To establish the effectiveness of NPSI protocols, they must be tested on larger systems as well. We have constructed a detailed simulation of the hardware and software test environment and verified it against data from the experiments on the prototype hardware. Further, we have assembled a suite of workloads ranging from those where Time Warp is expected to perform well to those considered as “stress” cases for Time Warp. Extensive testing of ETA and its variants is underway.

Enhancing M1

In the context of our framework for NPSI adaptive protocols, we note that the higher the accuracy of M_1 in identifying potentially incorrect computations, the less the reliance of M_2 on tunable parameters like *s*. For example, in the ideal case, if M_1 is sophisticated enough to always identify any computation exactly as either correct or not, then M_2 is simply a binary function that stops event execution (and message sending) when M_1 flags an error. Accordingly, we are focusing on identifying other kinds of information that may be computed by M_1 and would increase the accuracy of the error potential. The goal is to minimize the dependence of M_2 on tunable parameters. Of course, care must be taken not to introduce such parameters into M_1 in the process.

8 Summary

We have introduced a new class of synchronization protocols called NPSI (near-perfect state information) adaptive protocols. These differ from previous approaches to adaptiveness in that they base their adaptive decisions on near-perfect information about the state of relevant parts of the entire system. In [RaSZ89], it has been shown that a load sharing policy that assumes perfect state information at zero cost offers the best solution. Correspondingly, we believe that NPSI adaptive protocols will provide a general, efficient solution to the synchronization problem of PDES.

A framework has been suggested for the design of NPSI adaptive protocols. Based on this framework, the Elastic Time Algorithm has been designed and implemented. For this implementation, near-perfect state information is computed and disseminated through a high-speed reduction network. The protocol has been tested with several workloads, the results from three of which have been presented here. From these results, it is evident that NPSI adaptive protocols can outperform pure Time Warp in both time and space. Some issues must be resolved before any conclusive statements can be made about the relative performance and usability of NPSI adaptive protocols. These include: designing a scheme for the LP’s to tune any parameters of the protocol automatically; testing on larger systems; designing more NPSI protocols and comparison with other adaptive protocols. Ongoing research into some of these issues has been described.

Acknowledgments

We are grateful to Bronis de Supinski for proof reading the paper and for his insightful suggestions. We thank the members of the Smart Interconnection Networks group at the University of Virginia for their suggestsions. This work was supported by Mystech, Inc. (Academic Affiliates Program).

References

[BaHo90] Ball, D. and Hoyt, S., "The adaptive Time-Warp concurrency control algorithm", *Proceedings of the SCS Multiconference on Distributed Simulation*, January 1990, 174-177.

[Bell93] Bellenot, S., "Performance of a riskfree Time Warp operating system", *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, May 1993, 155-158.

[DaFu94] Das, S. and Fujimoto, R.M., "An adaptive memory management protocol for Time Warp parallel simulation", *Proceedings of SIGMETRICS '94*, May 1994, 201-210.

[Dick93] Dickens, P.M., "Analysis of an aggressive global windowing algorithm", Ph.D. thesis, Department of Computer Science, University of Virginia, May 1993.

[DiRe90] Dickens, P.M. and Reynolds, P.F., Jr., "SRADS with local rollback", *Proceedings of the 1990 SCS Multiconference on Distributed Simulation*, January 1990, 161-164.

[FeZh87] Ferrari, D. and Zhou, S., "An empirical investigation of load indices for load balancing applications", Report number CSD-87-353, Computer Science Division, University of California at Berkeley, 1987.

[FeTr94] Ferscha, A. and Tripathi, S.K., "Parallel and distributed simulation of discrete event systems", University of Maryland at College Park Technical Report number CS-TR-3336, August 1994.

[Fuji90] Fujimoto, R.M., "Parallel discrete event simulation", *CACM*, Vol. 33, No. 10, October 1990, 30-53.

[FuNi92] Fujimoto, R.M. and Nicol, D.M., "State of the art in parallel simulation", *Proceedings of the 1992 Winter Simulation Conference*, December 1992, 246-254.

[Gima89] Gimarc, R.L., "Distributed simulation using hierarchical rollback", *Proceedings of the 1989 Winter Simulation Conference*, December 1989, 621-629.

[HaTr94] Hamnes, D.O. and Tripathi, A., "Evaluation of a local adaptive protocol for distributed discrete event simulation", *Proceedings of the 1994 International Conference on Parallel Processing*, August 1994, Vol. III, 127-134.

[Jeff85] Jefferson, D.R., "Virtual time", *ACM Transactions on Programming Languages and Systems*, Vol. 7, No. 3, July 1985, 404-425.

[Lin92] Lin. Y-B., "Memory management algorithms for optimistic parallel simulation", *Proceedings of the 6th Workshop on Parallel and Distributed Simulation*, January 1992, 43-52.

[LuWS89] Lubachevsky, B., Weiss, A. and Shwartz, A., "Rollback sometimes works . . . if filtered", *Proceedings of the 1989 Winter Simulation Conference*, December 1989, 630-639.

[Madi93] Madisetti, V.K., "Randomized algorithms for self-synchronization", Private communication, 1993.

[MaWM88]Madisetti, V.K., Walrand, J. and Messerschmitt, D., "WOLF: A rollback algorithm for optimistic distributed simulation systems", *Proceedings of the 1988 Winter Simulation Conferences*, December 1988, 296-305.

[McAf90] McAffer, J., "A unified distributed simulation system", *Proceedings of the 1990 Winter Simulation Conference*, December 1990, 415-422.

[Mehl91] Mehl, H., "Speed-up of conservative distributed discrete event simulation methods by speculative computing", *Proceedings of the 5th Workshop on Parallel and Distributed Simulation*, January 1991, 163-166.

[Nico91] Nicol, D.M., "Performance bounds on parallel self-initiating discrete-event simulations", *ACM Transactions on Modeling and Computer Simulations*, Vol. 1, No. 1, 1991, pp 24-50.

[NiRe90] Nicol, D.M. and Reynolds, P.F., Jr., "Optimal dynamic remapping of parallel computations", *IEEE Trans. on Computers*, Vol. 39, No. 2, February 1990.

[PaWi93] Palaniswamy, A.C. and Wilsley, P.A., "Adaptive bounded time window in an optimistically synchronized simulator", *Proceedings of the third Great Lakes symposium on VLSI*, 1993, 114-118.

[PrSu91] Prakash, A. and Subramanian, R., "Filter: An algorithm for reducing cascaded rollbacks in optimistic distributed simulations", *Proceedings. of the 24th Annual Simulation Symposium*, April 1991, 123-132.

[RaAT93] Rajaei, H., Ayani, R. and Thorelli, L-E., "The Local Time Warp approach to parallel simulation", *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, May 1993, 119-126.

[RaSZ89] Ramamritham, K., Stankovic, J.A. and Zhao, W., "Ditributed scheduling of tasks with deadlines and resource requirements", *IEEE Transactions on Computers*, Vol. 38, No. 8, August 1989, 1110-1123.

[ReJe89] Reiher, P.L. and Jefferson, D., “Limitation of optimism in the Time Warp operating system”, *Proceedings of the 1989 Winter Simulation Conference*, December 1989, 765-770.

[Reyn88] Reynolds, P.F., Jr., “A spectrum of options for parallel simulation”, *Proceedings of the 1988 Winter Simulation Conference*, December 1988, 325-332.

[RePS92] Reynolds, P.F., Jr., Pancerella, C.M. and Srinivasan, S., “Making parallel simulations go fast”, *Proceedings of the 1992 Winter Simulation Conference*, December 1992, 646-656.

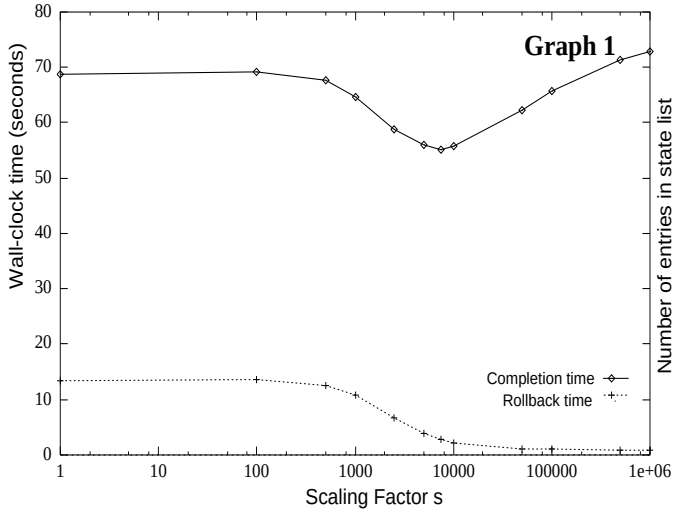
[SoBW88] Sokol, L.M., Briscoe, D.P. and Wieland, A.P., “MTW: a strategy for scheduling discrete events for concurrent execution”, *Proceedings of the SCS Multiconference on Distributed Simulation*, 1988, 34-42.

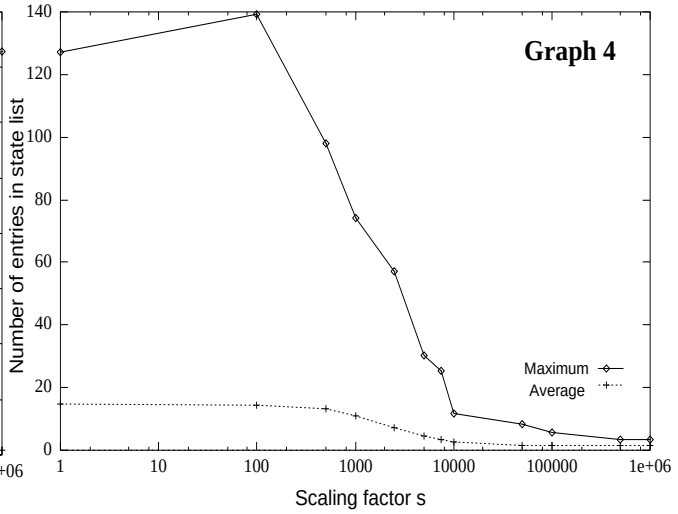
[SrRe93] Srinivasan, S. and Reynolds, P.F., Jr., “Non-interfering GVT computation via asynchronous global reductions”, *Proceedings of the 1993 Winter Simulation Conference*, December 1993, 740-749.

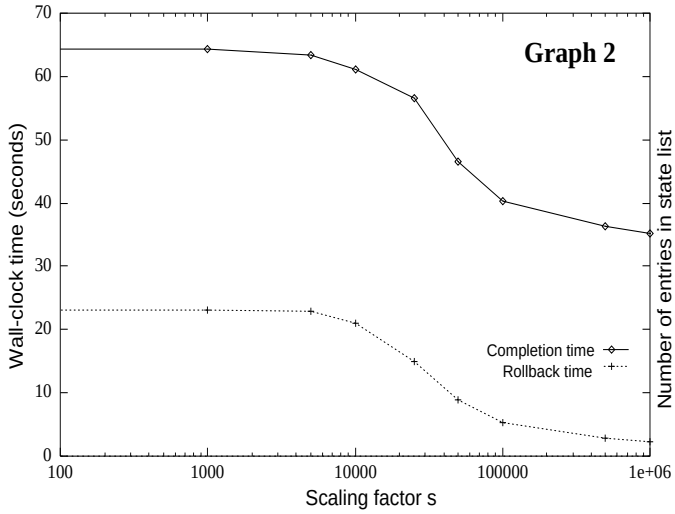
[Stei91] Steinman, J.S., “Interactive SPEEDES”, *Proceedings of the 24th Annual Simulation Symposium*, 1991, 149-158.

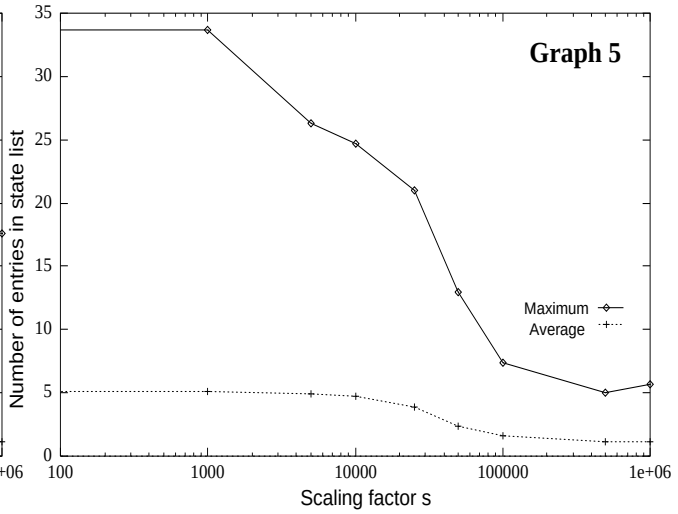
[Stei93] Steinman, J.S., “Breathing Time Warp”, *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, May 1993, 109-118.

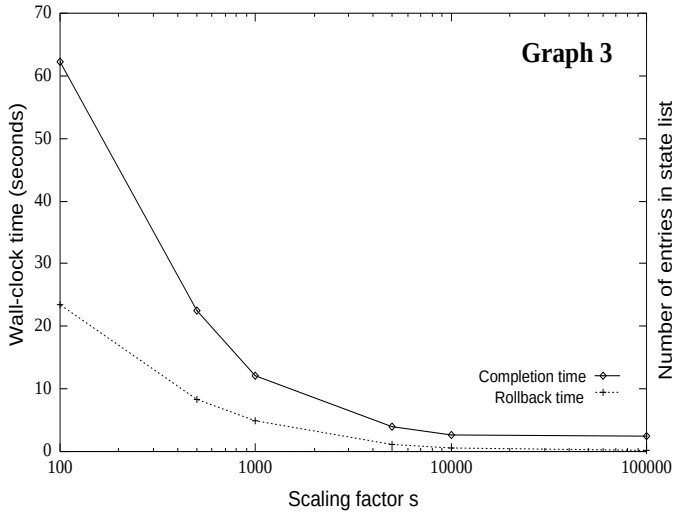
[TuXu92] Turner, S.J. and Xu, M.Q., “Performance evaluation of the Bounded Time Warp algorithm”, *Proceedings of the 6th Workshop on Parallel and Distributed Simulation*, January 1992, 117-126.

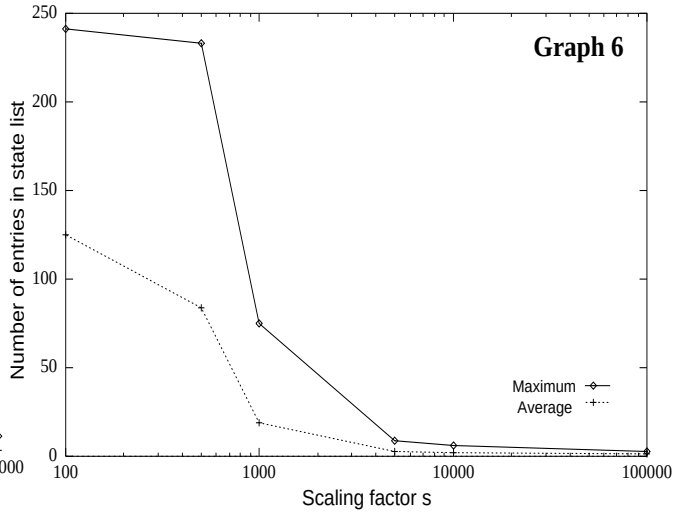

Graph 1
Wall-clock time (seconds)
Scaling Factor s
Completion time
Rollback time


Graph 4
Number of entries in state list
Scaling factor s
Maximum
Average


Graph 2
Wall-clock time (seconds)
Scaling factor s
Completion time
Rollback time


Graph 5
Number of entries in state list
Scaling factor s
Maximum
Average


Graph 3
Wall-clock time (seconds)
Scaling factor s
Completion time
Rollback time


Graph 6
Number of entries in state list
Scaling factor s
Maximum
Average