ELSEVIER

Information Processing Letters 83 (2002) 243–249

Information Processing Letters

www.elsevier.com/locate/ipl

A restriction of the elastic time algorithm

Francesco Quaglia

Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy

Received 8 January 2001; received in revised form 12 December 2001

Communicated by A.A. Bertossi

Keywords: Parallel discrete event simulation; Parallel processing; Virtual time synchronization; Performance optimization

1. Introduction

A parallel discrete event simulator consists of a set of logical processes (LPs), each one modeling a specific portion of the simulated system, which are essentially discrete event simulators having their own simulation clock, their own state variables and their own event list [3]. The execution of a simulation event at an LP usually modifies the LP state and possibly schedules new events to be executed at later simulation time. The scheduling of events among distinct LPs takes place through the exchange of messages carrying the content and the simulation time, namely *timestamp*, of the event.

The central problem of parallel discrete event simulation is *synchronization*, which must ensure the sequence of event executions at any LP satisfies some correctness criterion, typically *timestamp ordering*. To solve the synchronization problem many algorithms have been proposed, differing from each other by the strategy for simulating the events. In the conservative strategy an LP executes an event only after determining the event execution will not result in any timestamp order violation. In the optimistic strategy, the LPs execute events aggressively without the guarantee of no timestamp order violation. When real violations are detected, a rollback mechanism is used to recover the state of the simulation application to a correct value. As compared to the conservative strategy, the optimistic one allows the exploitation of parallelism anytime it is possible for violations to occur but they do not.

Some optimistic algorithms, like classical Time Warp [7], adopt no control at all on the optimism of event execution. Some others adopt controlled optimism (see, for example, [1,2,8–10,13,14]) in order to keep low the amount of rollback while still exploiting parallelism internal to the application. A classification of these algorithms based on controlling criteria has been recently presented by Srinivasan and Reynolds [11].

The work in [11] introduces also a new class of algorithms with controlled optimism, namely the *near perfect state information* (NPSI) algorithms [11]. In these algorithms state information is used to compute an *error potential* (EP) that constitutes the basis for optimism control. Specifically, the control mechanism consists of delaying the event execution (i.e., limiting aggressiveness in the event execution) based on the EP value. In addition, the authors instantiate an NPSI algorithm, called the *elastic time algorithm* (ETA), in which the EP of an LP is computed using state infor-

E-mail address: quaglia@dis.uniroma1.it (F. Quaglia).

mation related to all the predecessors of the LP in the communication graph of the simulation application.

In this paper we show by discussion how, depending on proper dynamics of the application, the EP calculation underlying ETA might result excessively conservative in that the computed value for EP might result over-sized. We then propose a restricted version of ETA, namely *restricted elastic time* (RETA), in which the EP value is computed more optimistically using state information associated with a restricted set of predecessors of the LP.

Results of an empirical study on a stress case simulation model demonstrate the potential of RETA in reducing the completion time of the parallel simulation application.

The remainder of the paper is structured as follows. In Section 2 we provide details on the NPSI approach and on ETA. In Sections 3 and 4 we discuss the EP calculation associated with ETA and point out approaches for a more optimistic calculation. Section 5 presents RETA. The results of the empirical study are reported in Section 6.

2. The NPSI approach and ETA

The NPSI approach can be easily described through Fig. 1. A function M_1 computes the EP value of the LP using as input state information collected through an adequate feedback system. Once computed EP, a function M_2 uses this value to control optimism by determining the delay in the event execution.

The feedback system should provide up to date state information and the function M_1 should be evaluated frequently. This allows NPSI algorithms to be adaptive in that the EP value changes dynamically depending on current system conditions, therefore the delay in the event execution changes dynamically as well.

In ETA, the function M_1 computes the EP value associated with an LP by using state information related to all the LPs belonging to the predecessor set of that LP. We recall that the predecessor set of an LP is determined by the communication graph defined by the LPs in a way that a directed edge exists from LP_j to LP_i if LP_j can schedule simulation events for LP_i. In particular, the predecessor set of LP_i includes any LP_k such that there exists a directed path from LP_k to LP_i in the communication graph. We denote the predecessor set of LP_i as $PS(LP_i)$.

Denoting with:

σ_i	the simulation clock (logical clock) of LP_i;
ν_i	the minimum unreceived message time of LP_i (that is the smallest timestamp of events scheduled by LP_i but not yet incorporated in the event lists of the recipient LPs);
η_i	the next event time of LP_i (this value is equal to σ_i while LP_i executes an event);
α_i	the minimum between η_i and ν_i (i.e., $\alpha_i = \min(\eta_i, \nu_i)$);
α'_i	the following value: $\min_{LP_k \in PS(LP_i)}(\alpha_k)$

the function M_1 of ETA computes the EP_i value of LP_i as follows:

$$M_1\colon\ EP_i = \max\bigl(\eta_i - \alpha'_i, 0\bigr). \tag{1}$$

α'_i is referred to as the *minimum future time* associated with LP_i. It indicates the minimum simulation time from which events not yet executed, or currently being executed, can affect LP_i. The responsibility to calculate α'_i and to provide it to M_1 pertains to the feedback system implementing an adequate reduction model. Implementations of reduction models can be found in [5] for the case of a shared memory architecture and in [12] for the case of a distributed memory architecture. For both ETA and the restricted version we propose, namely RETA, we have used a solution similar to the one in [12]. This solution has been designed for the case of a distributed memory system with a high speed Myrinet switch, which is exactly

$$\begin{tikzcd}
\begin{array}{c}\text{state}\\ \text{information}\end{array} \arrow[r] & M_1 \arrow[r] & \text{EP} \arrow[r] & M_2 \arrow[r] & \text{delay}
\end{tikzcd}$$

Fig. 1. Scheme for NPSI algorithms.

the type of architecture we have used for the empirical study.

The function M_2 of ETA, which computes the delay δ_i in the execution of the next event of LP_i, is defined as follows:

$$M_2\colon\ \delta_i = s \times EP_i, \tag{2}$$

where s is a scaling factor. The authors pointed out that s is a global static parameter whose value must be determined in an application specific manner. The staticity of s comes out from that the adaptiveness of ETA derives primarily from the dynamic recalculation of the error potential EP. They also pointed out how an adequate value for s can be selected automatically during the simulation execution by monitoring the impact of its variation on the rate of commitment of events [11].

3. Comments on the EP calculation of ETA

To understand our perspective on the EP calculation of ETA produced by M_1, we start by introducing an EP calculation function relying on a global measure of the simulation time that we denote as *global minimum future time* (*GMFT*). Using the notations introduced in Section 2 and denoting as X the set of all the LPs of the simulation, *GMFT* is computed as follows:

$$GMFT = \min_{LP_k \in X}(\alpha_k) \tag{3}$$

(recall α_k is the minimum timestamp value of events not yet executed by LP_k, or currently being executed by LP_k, or produced by LP_k and carried by messages still in transit). Note that the value of *GMFT* could be different from the minimum future time α'_i associated with LP_i unless $PS(LP_i) = X$.

We can now envisage the following EP calculation function, namely M_1^{GMFT}, relying on *GMFT*:

$$M_1^{GMFT}\colon\ EP_i = \max(\eta_i - GMFT, 0). \tag{4}$$

As the reader can check, the function M_1 can be thought as derived from M_1^{GMFT} by restricting the domain of application of the min function. Specifically, M_1^{GMFT} relies on *GMFT* which is computed applying the min function over the α_k values associated with all the LPs belonging to X (recall X contains all the LPs of the simulation). Instead, M_1 relies on α'_i which is computed applying the min function only over the α_k values associated with the LPs belonging to $PS(LP_i) \subseteq X$ (recall $PS(LP_i)$ contains only the LPs that are predecessors of LP_i).

By construction, we get $\alpha'_i \geqslant GMFT$, therefore the EP_i value computed by M_1^{GMFT} is an upper bound on the EP_i value computed by M_1. Consequently, we can think of M_1^{GMFT} as a more conservative EP calculation function, compared to M_1.

Both M_1^{GMFT} and M_1 compute the EP value taking into account the immediate future (i.e., timestamps of events not yet executed or currently being executed), which, as shown in [11], leads to effective optimism control. Therefore we could now wonder which is the real justification for using the less conservative function M_1. The justification is that if LP_j does not belong to the predecessor set of LP_i, then there is no directed path from LP_j to LP_i in the communication graph, so the simulation progress at LP_j cannot affect the simulation progress at LP_i. Hence, there is no need to include state information associated with LP_j in the calculation of EP_i. In other words, the function M_1^{GMFT} might over-size the EP value because it does not take into account the topological structure of the communication graph.

The problem we can rise now is: "could the function M_1 reveal still excessively conservative?". In other words, beyond the topological structure of the communication graph, do other features proper of the application exist such that if we use them we can get a less conservative function distinct from M_1 which defines a more refined EP calculation?

The motivation for previous question is that, although the topological structure of the communication graph identifies the elements belonging to $PS(LP_i)$ as the set of LPs whose immediate future might theoretically affect LP_i, there exists the possibility that an LP belonging to $PS(LP_i)$ only marginally affects the progress of LP_i in practice. This can be due exactly to a variety of application specific features. Considering for example branching, there exists the possibility that the directed path between LP_j and LP_i is extremely unlikely to favor that an event at LP_j ultimately cause an event to be scheduled at LP_i, even if there exists the possibility for this to occur. Furthermore, even though no branching exists along the path between LP_j and LP_i, there is still the possibility that the next event of LP_j will never ultimately cause an event to be scheduled at LP_i; this might be due to the proper nature of the simulation application.

This points out how the function M_1, although less conservative than M_1^{GMFT}, could still over-size the EP value depending on application features distinct from the topological structure of the communication graph. In other words, we can think of M_1 as a function which computes the upper bound value of EP with respect to the immediate future and the topological structure of the communication graph. This bound reveals tight (or not) depending on other application specific features. Therefore M_1 reveals excessively conservative (or not) depending on these features.

4. More optimism in the EP calculation

The ideal approach to solve the problem raised in previous section would be to explicitly take into account application features distinct from the topological structure of the communication graph for optimism control purposes. This might render the EP calculation less conservative in some circumstances and better tailored to the specific application; we call this approach as *opaque*. However, this solution suffers from the following main problems:

- The feedback system that manipulates state information could have to be re-designed and re-implemented each time a new application must be supported. This is because it must be instantiated according to a number of application features whose nature can vary from an application to another. For the case of feedback system implemented with general purpose commercial hardware, this might originate enormous design/programming efforts. For the case of feedback system implemented with special purpose hardware, we might also incur additional hardware costs.
- In some case the global impact of application specific features on synchronization requirements could be practically impossible to identify.

As opposed to the opaque solution, we can envisage a *transparent* approach aiming at rendering the M_1 function less conservative without explicitly considering application features distinct from the topological structure of the communication graph. This approach is structured as follows. Let us consider the predecessor set $PS(LP_i)$ of LP_i, and define a subset $S(LP_i) \subseteq PS(LP_i)$. Denote as α_i'' the following:

$$\alpha_i'' = \min_{LP_k \in S(LP_i)} (\alpha_k). \tag{5}$$

We call α_i'' the *restricted minimum future time* as it is analogous to the the minimum future time α_i' associated with LP_i except for that it is computed over a subset (therefore a restricted set) of the elements in $PS(LP_i)$. By using α_i'' we introduce the following EP calculation function:

$$M_1'\colon\ EP_i = \max\bigl(\eta_i - \alpha_i'', 0\bigr). \tag{6}$$

This function has structure similar to the function M_1 of ETA. It uses α_i'' instead of α_i' for computing EP_i. By construction $\alpha_i'' \geqslant \alpha_i'$ (due to the fact that $S(LP_i) \subseteq PS(LP_i)$), hence the EP_i value computed by M_1 is an upper bound on the EP_i value computed by the function M_1'. Therefore, using the minimum restricted future time in the calculation of EP leads to a less conservative solution as compared to M_1. This is exactly what we were looking for. Overall, in the transparent approach we try to compensate the potentially too conservative EP calculation of M_1 without explicitly considering application specific features distinct from the topological structure of the communication graph. The RETA algorithm that we propose in the next section is based exactly on this transparent approach.

5. Restricted ETA

The key point of the transparent approach is to identify a subset $S(LP_i)$ of $PS(LP_i)$ such that the EP calculation, although less conservative as compared to the one performed by the function M_1 associated with ETA, still allows effective optimism control.

In our perspective the selection of the elements $LP_j \in PS(LP_i)$ to be included in the set $S(LP_i)$ should be based on the distance between LP_j and LP_i in the communication graph, evaluated in terms of directed edges to be crossed starting from LP_j to reach LP_i. More precisely, we define the distance between LP_j and LP_i, denoted as $D(LP_j, LP_i)$, as the *minimum* number of edges to be crossed starting from LP_j to reach LP_i. By convention, if there is no directed path between LP_j and LP_i in the communication graph, then $D(LP_j, LP_i) = \infty$. Trivially, $1 \leqslant D(LP_j, LP_i) \leqslant \infty$.

In the RETA algorithm that we propose, the value α''_i of the restricted minimum future time is computed applying the min function over the set $S(LP_i)$ which is originated from $PS(LP_i)$ by discarding all the LPs whose distance from LP_i is larger than one. In other words, the set $S(LP_i)$ contains only the immediate predecessors of LP_i. Formally:

$$S(LP_i) = \big\{LP_j \mid D(LP_j, LP_i) = 1\big\}. \tag{7}$$

Therefore, RETA adopts the function M'_1 in (6) with α''_i computed over the set $S(LP_i)$ defined in (7).

For what concerns the final computation of the event execution delay, RETA adopts the same function M_2 adopted in ETA (see expression (2)), therefore the EP_i value computed using the restricted minimum future time is translated into delay through a simple scaling factor.

As respect to effectiveness, computing the restricted minimum future time based only on state information associated with the immediate predecessors of any LP originates a final behavior such that the simulation progress of LP_i remains still controlled based on state information associated with all the LPs belonging to $PS(LP_i)$, which is the basic philosophy underlying ETA. Such a final behavior derives from the fact that the progress of the immediate predecessors of LP_i is, in its turn, controlled based on state information associated with their immediate predecessors and so on. Therefore, LPs belonging to $PS(LP_i)$, which are not immediate predecessors of LP_i, still have a form of control on the progress of LP_i.

6. Performance analysis

In this section we report the results of an empirical analysis to compare ETA and RETA. As a testing environment we have used a cluster of PCs Pentium II 300 MHz (128 MB RAM) with LINUX, interconnected by a high speed Myrinet switch. To calculate α'_i and α''_i we have used an implementation of a distributed reduction model consisting of a communication module to disseminate state information among the hosts and a module to calculate the reduction function on the basis of information collected through the communication module. Actually, the implementation is similar to the one presented in [12].

In the simulation software we have used, memory space for new entries into the event lists of the LPs is allocated dynamically. The same dynamical approach has been used for the entries of the stack of checkpointed state vectors. Rollback is non-preemptive and adopts aggressive antimessage sending [6]. Global Virtual Time calculation and "fossil collection" are executed periodically. Also, in the implementations of ETA and RETA, the blocking state due to event execution delay is opaque in that the LP is allowed to receive messages while in the blocking state. Waiting is aborted in case of receipt of a message/antimessage that causes rollback.

The simulation model we have used for the empirical study is derived as a particular instance of the PHOLD model [4]. We recall PHOLD is widely used as a standard benchmark in the parallel discrete event simulation literature. It consists of a fixed number of LPs and of a constant number of events, namely *jobs*, circulating among the LPs. Routing of jobs and timestamp increments are taken from some stochastic distributions.

The configuration of PHOLD we have selected has a bi-directional ring topology in which each LP has a left and a right neighbor LP, and each job is equally likely to be forwarded to one of the two neighbor LPs. The bi-directional ring topology has been selected for the following two reasons:

(i) For model size (number of LPs) larger than three, $PS(LP_i)$ does not coincide with the set $S(LP_i)$ defined in (7), thus α'_i does not coincide with α''_i and the two functions M_1 and M'_1 lead ETA and RETA to perform different optimism control decisions (recall that if $S(LP_i)$ coincides with $PS(LP_i)$, then RETA boils down to ETA).

(ii) For model size larger than two, there is non-null probability that an event, namely a job arrival, at any LP will never ultimately cause an event to be scheduled at another LP within a finite amount of simulation time. (In other words, a job might move in the ring without ever reaching a given LP within a finite amount of simulation time.) Therefore, as pointed out in Section 3, more optimism

Table 1
Event rate vs. model size (standard deviation and confidence intervals are reported within brackets)

Synchronization	Model size		
algorithm	4	5	6
ETA	9370 (78, ±64)	10,803 (83, ±69)	11,512 (101, ±84)
RETA	9506 (93, ±77)	11,572 (124, ±103)	12,834 (119, ±99)

in the EP calculation, as compared to the calculation underlying ETA, might be desirable. This feature allows us to test the two algorithms on a stress case simulation model having the capability to highlight the advantages of RETA.

We have fixed the event granularity, i.e., the event processing time, at about 200 microseconds and checkpointing to support rollback is performed after the execution of each event. Since this application is actually stateless, checkpointing consists only of copying the simulation clock and the seeds for the random number generation. Also, the timestamp increment is selected from an exponential distribution with mean 10 simulation time units and the job population has been fixed at 5 jobs per LP (a similar workload has been adopted for some instances of the PHOLD model used in [11] for studying the performance of ETA). In the experiments we have mapped each LP onto a distinct machine and we have varied the model size (number of LPs) from four to six.[1] As pointed out before, model size less than or equal to three is useless since it would lead RETA to boil down to ETA.

Finally, to ensure fairness in the comparison, the communication module implementing part of the reduction model has been set in order to perform the same communication operations independently of the reduction function that must be supported. Otherwise the reduction function for α_i' could require higher communication overhead as compared to the reduction function for α_i'' (recall that the calculation of α_i' at the machine hosting LP_i requires information associated with all the predecessors of LP_i, while the calculation of α_i'' requires only information associated with the immediate predecessors).

As a performance parameter, we have selected the *event rate*, i.e., the number of committed events per second. This parameter indicates how fast is the simulation execution with a given synchronization algorithm, therefore it is representative of the achieved performance. For this parameter, we report the average value, computed over ten runs, the standard deviation over the ten samples and the confidence interval computed at the 95% confidence level on the basis of the t-Student distribution. All the runs were done with different seeds for the random number generation and at least 1×10^6 committed events were simulated in each run. For both ETA and RETA we report the value of the event rate obtained in correspondence to the “best suited” value of s, i.e., the value yielding the best performance.[2]

The results are reported in Table 1. They point out that RETA provides performance gain that increases with the model size. This is an expected behavior since increasing the model size exalts features in points (i) and (ii), which were expected to highlight differences between ETA and RETA and also advantages of RETA. Note that the gain provided for the case of maximal model size is in the order of 11%, which represents a relevant performance improvement when considering it is achieved over an optimized synchronization algorithm.

As a final point, tailoring the communication module associated with the implementation of the reduction model in order to transmit the minimal amount of information required to calculate α_i'' should be likely

[1] One LP per machine leads to the highest degree of parallelism in the simulation model execution, which could originate excessive amount of rollback if synchronization algorithms with no controlled optimism are adopted. Therefore, this choice allows us to test ETA and RETA in a situation in which controlled optimism is mandatory in practice. Also, the maximal model size we have considered is six since this is the number of available machines in the cluster.

[2] In the experiments we have varied manually the value of s between 0 and 200 microseconds per simulation time unit, with step of 10 at each increment. For both ETA and RETA the characteristic peak in performance has been noted for values of s within that interval.

to yield further performance improvements due to possible reductions of the communication overhead. This would also allow higher scalability of RETA.

References

[1] A. Ferscha, Probabilistic adaptive direct optimism control in time warp, in: Proc. 9th Workshop on Parallel and Distributed Simulation (PADS'95), June 1995, pp. 120–129.

[2] A. Ferscha, J. Luthi, Estimating rollback overhead for optimism control in time warp, in: Proc. 28th Annual Simulation Symposium, April 1995, pp. 2–12.

[3] R.M. Fujimoto, Parallel discrete event simulation, Comm. ACM 33 (10) (1990) 30–53.

[4] R.M. Fujimoto, Performance of time warp under synthetic workloads, Proc. Multiconf. Distributed Simulation 22 (1) (1990).

[5] R.M. Fujimoto, M. Hybinette, Computing global virtual time in shared-memory multiprocessors, ACM Trans. Modeling Comput. Simulation 7 (4) (1990).

[6] A. Gafni, Space management and cancellation mechanisms for time warp, Tech. Rept. TR-85-341, University of Southern California, Los Angeles, CA.

[7] D. Jefferson, Virtual time, ACM Trans. Programming Languages Systems 7 (3) (1985) 404–425.

[8] B. Lubachevsky, A. Weiss, A. Shwartz, Rollback sometimes works ... if filtered, in: Proc. 1989 Winter Simulation Conference, December 1989, pp. 630–639.

[9] V.K. Madisetti, D.A. Hardaker, R.M. Fujimoto, The MIMDIX environment for parallel simulation, J. Parallel Distributed Comput. 18 (1993) 473–483.

[10] L.M. Sokol, D.P. Briscoe, P.A. Wieland, MTW: A strategy for scheduling discrete events for concurrent execution, in: Proc. SCS Multiconference on Distributed Simulation, July 1988, pp. 34–42.

[11] S. Srinivasan, P.F. Reynolds Jr., Elastic time, ACM Trans. Modeling Comput. Simulation 8 (2) (1998) 103–139.

[12] S. Srinivasan, M.J. Lyell, P.F. Reynolds Jr., J. Wehrwein, Implementation of reductions in support of PDES on a network of workstations, in: Proc. 12th Workshop on Parallel and Distributed Simulation (PADS'98), May 1998, pp. 116–123.

[13] J. Steinman, Breathing time warp, in: Proc. 7th Workshop on Parallel and Distributed Simulation (PADS'93), May 1993, pp. 109–118.

[14] S.J. Turner, M.Q. Xu, Performance evaluation of the bounded time warp algorithm, in: Proc. 6th Workshop on Parallel and Distributed Simulation (PADS'92), January 1992, pp. 117–126.