ICOT Technical Report: TR-0798

TR 0798

Adaptive Time-Ceiling for Efficient Parrallel Discrete Event Simulation

by
Y. Matsumoto & K. Taki

August, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

ADAPTIVE TIME-CEILING FOR EFFICIENT PARALLEL DISCRETE EVENT SIMULATION

Yukinori Matsumoto and Kazuo Taki

Institute for New Generation Computer Technology
1-4-28 Mita, Minato-ku,
Tokyo 108, Japan

ABSTRACT

This paper presents the Adaptive Time-Ceiling (ATC) mechanism which improves the efficiency of parallel discrete event simulation based on the Time Warp mechanism. The Time Ceiling, a temporary upper-bound of time to be simulated, suppresses speculation errors which require recovery overheads. ATC is a mechanism which advances the Time-Ceiling position adaptively at runtime, according to the current simulator performance and the error frequency. Thus, ATC is expected to enhance the efficiency of the Time Warp mechanism for a wide range of problems. We applied ATC to a parallel logic simulator on a distributed memory machine. Measurements were taken on a 256-processor machine using ISCAS'89 benchmarks. The performance was 37 % higher than that without ATC on average, and was 111 % higher at the best case.

1 INTRODUCTION

In discrete event simulation, real systems to be simulated are usually modeled as collections of *objects* which communicate with others via messages. This object-oriented methodology realizes natural and faithful modeling of target systems.

Currently, most debates in this field are concentrated on parallel discrete event simulation. For the efficient execution of parallel discrete event simulation, an efficient mechanism for keeping time correctness is essential. The Time Warp mechanism[Jefferson 1985] is an efficient mechanism capable of extracting high parallelism from target problems with speculative computation. The speculative computation, however, sometimes results in errors. These errors require recovery operations called *rollback* which decrease the performance of simulators. Therefore, suppressing the speculation errors is important for enhancing the efficiency of the Time Warp mechanism.

The Moving Time Window (MTW) [Sokol 1988] and the Bounding Window (BW) [Briner 1991] have been proposed to reduce speculation errors. In both approaches, a *window* is used over the simulated time space. The window suspends the simulation of events with extremely large time-stamps and thus can reduce the speculation errors.

However, both approaches have shortcomings. In MTW, the window is slid to the future whenever events with the smallest time are simulated. In other words, the global virtual time (GVT) must be calculated very frequently to advance the window. As GVT update operations require global communication such as global synchronization, MTW is efficiently applicable only to small-scale shared memory machines. Frequent global communication may cause significant overhead on highly parallel machines with distributed memory.

On the other hand, in BW, a new window is created next to the previous window, only when the events within the previous window have been completely processed. Therefore, BW does not cause frequent GVT updating. However, many processors may idle at the end of each window.

Furthermore, both approaches leave an essential problem unsolved, that is, how to estimate the optimum size of the window before simulation is carried out. The problem is quite difficult because the optimum size of the window depends highly on the target problems.

We are aiming at enhancing the efficiency of the Time Warp mechanism for large-scale multiprocessors, most of which have distributed memory. In this paper, we present the Adaptive Time-Ceiling (ATC), a practical mechanism for reducing speculation errors without depending on target problems. Although the Time-Ceiling is equivalent to the upper bound of the window in MTW or BW, the ATC makes up for their shortcomings. First, the height of the Time-Ceiling, which corresponds to the window size in MTW or BW, *varies* according to the current simulator performance and the current frequency of speculation error occurrence. Second, updating the Time-Ceiling can be performed before the completion of event execution below the previous Time-Ceiling. Thus, the idling problem is not as serious as in BW, while the updating frequency is much less than that in MTW (approximately equal to that in BW).

We have built a parallel logic simulator on the PIM machine[Nakashima 1992, Taki 1992], a large-scale MIMD multiprocessor system with distributed memory. We have measured the performance of the simulator in order to ascertain the effectiveness of ATC.

This paper is organized as follows: Section 2 outlines the Time Warp mechanism. ATC is detailed in Section 3 and the method of deciding the height of the Time-Ceiling is described. In Section 4, a parallel logic simulator, a typical application of parallel discrete event simulation, and our parallel machine PIM are overviewed. Section 5 reports the results of measuring the effectiveness of ATC on the simulator. Our conclusion is given in Section 6.

2 THE TIME WARP MECHANISM

Discrete event simulation can be modeled so that several objects change their states by exchanging messages with each other. A message has information on an event whose occurrence time is stamped on the message (time-stamp). In discrete event simulation, messages must be evaluated by their destination object in time-stamp order. Otherwise, the results of simulation will be incorrect.

The Time Warp mechanism [Jefferson 1985] is a mechanism which keeps the time-correctness of discrete event simulation in an asynchronous manner. In the Time Warp mechanism, each object basically assumes that messages arrive chronologically. Whenever messages arrive in time stamp order, the objects simply evaluate these (this is speculative computation) while recording the history of the messages and states.

However, when a message arrives at an object out of time-stamp order (a speculation error is detected), the object rewinds its history and makes adjustments as if the message had arrived in the correct order. This procedure is called rollback, its frequent occurrence often decreases simulator performance.

As the number of objects in actual applications are much more than the number of processors, plural objects are assigned to each processor. Thus appropriate event scheduling within each processor is useful for suppressing rollback frequency.

Global Virtual Time (GVT) should sometimes be updated. GVT is a global lower bound of the time to be simulated. As rollback never happens before GVT, the history area before GVT can be released and reused. Details are presented in [Jefferson 1985].

3 ADAPTIVE TIME-CEILING (ATC)

In a naively-implemented Time Warp mechanism, a message is evaluated as soon as it arrives at its destination object, even if its time-stamp is extremely large. Such a message, however, will probably be rewound later. Therefore, a mechanism suppressing the evaluation of messages with extremely large time-stamps should reduce the rewound messages or, in other words, the speculation errors.

In this section, we describe the Adaptive Time-Ceiling (ATC), which is a new mechanism applicable to a wide range of problems for reducing speculation errors.

3.1 Suppression of Speculation Errors with Time-Ceiling

The Time-Ceiling is equivalent to the upper bound of the window in BW. The Time-Ceiling is set above the GVT and is occasionally updated as GVT is renewed. Here, we call the interval between the GVT and the Time-Ceiling the *height* of the Time-Ceiling.

Since the Time-Ceiling gives the upper bound of the time to be simulated, it should contribute to reducing the speculation errors. Figure 1 shows several events waiting in a scheduling queue to be simulated. Here, the GVT obtained is 60 and the Time-Ceiling height is 40. In this example, events with time-stamps 70 and 90 can be simulated at appropriate moments, whereas the rest of the events must continue waiting until the Time-Ceiling is updated into the simulated future.

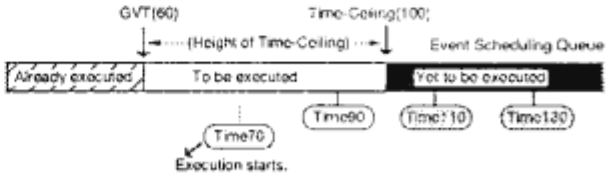

Figure 1: The role of the Time-Ceiling

3.2 Adaptive Decision of Time-Ceiling Height

Although the Time-Ceiling is expected to reduce speculation errors, it is quite difficult to know the optimum Time-Ceiling height before simulation starts because the optimum height depends on the target problem. If the height is too low, many processors might idle, although the speculation errors would be greatly reduced. Besides, frequent updates of the Time-Ceiling may cause large overheads. Conversely, if the Time-Ceiling jumps to the distant future, the speculation errors will hardly be reduced at all.

Figure 2 is a rough sketch of the relationship between the Time-Ceiling height and simulator performance. Curve A corresponds to the case where speculation errors are very few even without Time-Ceiling, or the rollback cost can be ignored. For this case, there is no need to introduce the Time-Ceiling or, if introduced, the height should be large enough, as indicated by a downward pointing arrow. Conversely, Curve B corresponds to the case where many speculation errors occur, or the rollback cost is high. For this case,

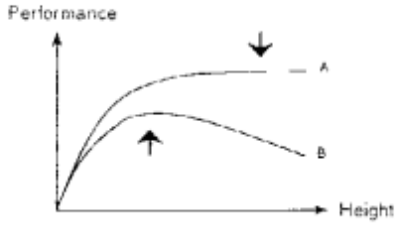

Figure 2: Time-Ceiling height vs. performance

the appropriate height should be at the point where there is an upward pointing arrow.

ATC is a mechanism which adjusts the Time-Ceiling height at runtime. The current performance of the simulator and the frequency of speculation error occurrence (the ratio of rewound messages to non-rewound messages) give information for adaptive changes. A strategy for the adjustment of the Time-Ceiling height is shown below. Here, we assume that the Time-Ceiling height can vary within a set of discrete values*. An example of the table is shown in Figure 3. Also, we assume that the initial Time-Ceiling height is set to the minimum value.

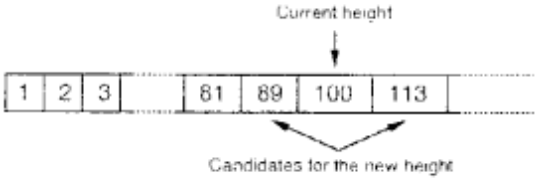

Figure 3: Table of Time-Ceiling heights (an example)

The algorithm is:

IF $R_{err} > Threshold$ **THEN**
The Time-Ceiling height is moved to the smaller side.

ELSE IF $K \geq 0$ **THEN**
The Time-Ceiling height is moved to the larger side.

ELSE IF $K < 0$ **THEN**
The Time-Ceiling height is moved to the smaller side.

where R_{err} is the current frequency of speculation error occurrence. Here, we define the ratio of rewound messages to non rewound messages as the frequency of speculation error occurrence. K is the gradient of the least-squares approximation line for the last N samples (Figure 4). Unfortunately, several experiments may be needed to find the appropriate N and $Threshold$. However, the parameters are less sensitive, and will be more general than the window size in BW or MTW.

*These values should be given in an exponential manner.

Figure 4: Adaptive decision of the Time-Ceiling height

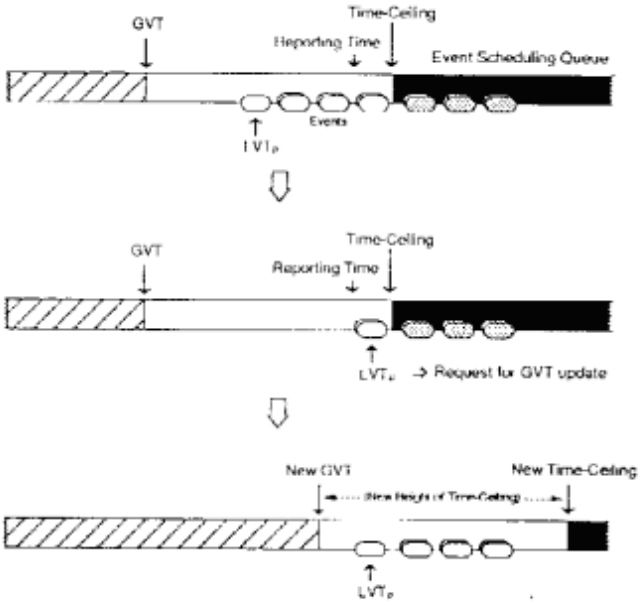

Figure 5: Process of the Time-Ceiling update

3.3 Timing of Time-Ceiling Update

There still remains a problem of when the Time-Ceiling should be updated. If the Time-Ceiling is updated after the completion of all events below the previous Time-Ceiling, like BW, many processors will inevitably be idle. To avoid this problem, the update operation should precede the completion of previous events.

Our strategy of starting the update operation is as follows. Here, we assume that each processor has a scheduler which manages all events simulated in the processor. Also, we assume that a GVT controller exists in the simulator to update GVT. A scheduler manages the local time of each processor (LVT_p), which is defined as the minimum of the local times of objects belonging to the processor.

When LVT_p first reaches the reporting time, which is below the current Time-Ceiling, the scheduler sends a request message to the GVT controller. On receiving request messages from all schedulers, the GVT controller updates GVT. Then the Time-Ceiling can be updated to be the sum of the

new GVT and the new height**. Meanwhile, the events between the reporting time and the previous GVT can be simulated. Figure 5 illustrates the update process of the Time-Ceiling†.

4 LOGIC SIMULATOR, LANGUAGE AND MACHINE

Parallel logic simulation is a typical application of parallel discrete event simulation. Here, each gate is modeled as an object, and a change of signal corresponds to an event.

We built a parallel logic simulator[Matsumoto 1992] on the PIM machine to evaluate the effectiveness of ATC in improving simulator performance.

4.1 Overview of the Logic Simulator

A rough specification of our simulator is shown below.

Target Circuits
Both combinatorial and sequential circuits, where all components must be described at gate level.

Signal values
3 values (Hi, Lo, X)

Delay
Nominal delay model (different delays can be assigned to each gate.)

In our logic simulator, target circuits are partitioned automatically in the preprocessing stage. Once the subcircuits are assigned to processors, they cannot migrate to other processors (static load distribution).

With respect to scheduling within a processor, event scheduling is realized by controlling the message delivery to destination gates. Each processor has a message scheduler, which transmits the message with the smallest time-stamp to its destination first.

4.2 KL1 Language

The simulator is written in concurrent logic language KL1[Ueda 1990]. KL1 has the following characteristics.

First, KL1 has an aspect of an object-oriented language. Objects' behavior against messages received (e.g. sending new messages or changing their states) can be naturally described with KL1. Second, KL1 realizes implicit data flow synchronization. This means that KL1 never compels programmers to describe synchronization explicitly at a primitive level. Third, the KL1 language assumes a system-wide (global) name space even on distributed memory machines. The programmer only adds a "pragma" to specify object allocation to a certain processor, as shown below.

$$\ldots, B@node(PE), \ldots$$

where B is a "goal" of KL1, which represents an object. These features provide us with a very good environment for object-oriented programming of parallel applications.

4.3 Parallel Inference Machine

The Parallel Inference Machine (PIM)[Nakashima 1992, Taki 1992] is the prototype hardware system of the Japanese Fifth Generation Computer Systems project. Programs written in KL1 can be executed efficiently on PIM. Five PIM models have been developed. Our simulator is evaluated on PIM/m.

PIM/m consists of 256 processors that are connected with others in a 2-dimensional mesh. Each processor has 80MB of local memory, a 1KB instruction cache and a 4KB data cache. A processing unit is a 40 bit (32 bits for data and 8 bits for a tag) pipelined microprocessor and the machine cycle is 65 nsec. The connection bandwidth between processors is 3.85MB/second.

Since PIM/m is a loosely-coupled distributed memory machine, the cost of data access to other processors is rather high compared to the cost within a processor. However, the difference between these costs is much less than that in systems where workstations are connected by a local area network.

5 MEASUREMENT RESULTS

We experimented with six sequential circuits of ISCAS'89 benchmarks.

Table 1 shows the size of the circuits.

Circuits	No. of gates
s38584	27,965
s38417	31,995
s35932	26,433
s15850	13,354
s13207	11,965
s9234	6,965

Table 1: Size of the circuits

With respect to the parameters, *Threshold*, the limit of speculation error frequency, was 0.5, and N, the number of samples for least-square approximation, was 9.

Figure 6 shows the performance vs. the number of processors. In the figure, the solid lines show the simulator performance for each circuit using the ATC, whereas dotted lines indicate performance without ATC.

Sometimes performance was fairly good even without ATC, and for these cases, ATC decreased performance a little. In other cases, ATC enhanced performance. We can see

**Of course, this triggering sometimes causes premature operation. In this case, the Time-Ceiling position should be kept and request messages are sent again at the same reporting time.

†Another possible strategy is to trigger the update operation when the number of events in the scheduling queue becomes smaller than the given threshold.

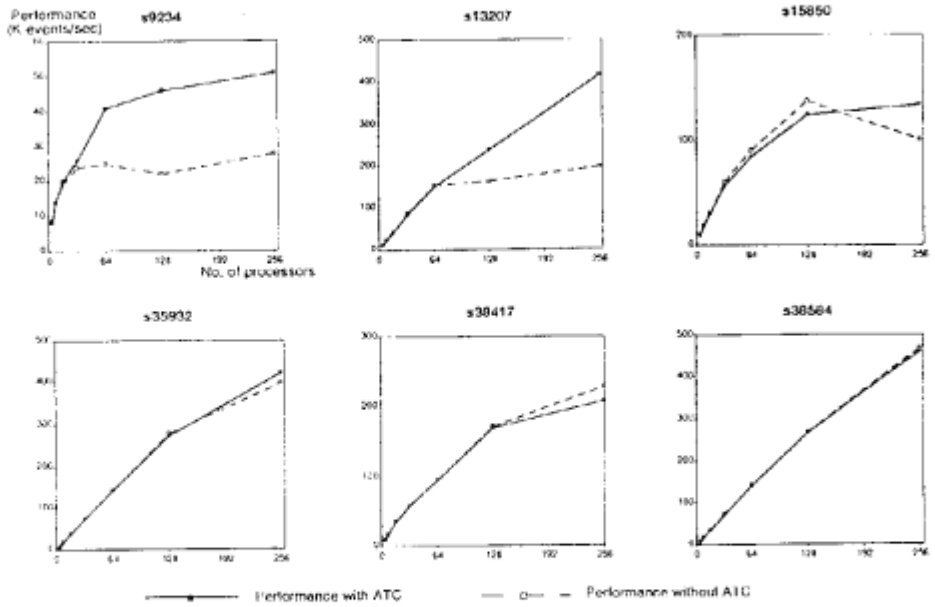

Figure 6: Effect of ATC on performance

that the more processors that are available, the higher the effectiveness of ATC. In fact, in our experiments, ATC showed the greatest effectiveness, 37% improvement in performance on average, when using 256 processors. This is quite natural because the frequency of speculation error occurrence grows as the number of processors increases.

Table 2 compares the frequency of speculation error occurrence for 256 processors. Apparently, the frequency was greatly reduced by ATC in all cases.

These results show that the ATC overhead is not so large, and ATC is, on average, effective not only for reducing speculation errors but also for obtaining near-optimum performance.

Circuits	$R_{err}^{\dagger}$ without ATC	$R_{err}^{\dagger}$ with ATC
s38584	0.299	0.262
s38417	1.027	0.349
s35932	0.496	0.318
s15850	2.345	0.377
s13207	0.648	0.090
s9234	3.921	0.285

$^{\dagger}R_{err}$ is the frequency of speculation error occurrence.

Table 2: Effect of the ATC in reducing speculation errors (for 256 processors)

The transition of the Time-Ceiling height is shown in Figure 7. In order to see the validity of the obtained values, we executed further simulation where the Time-Ceiling height is kept a constant value. The relationship between the height and performance is shown beside the transition graph. These graphs show that the height increased quickly and then moved around the optimum height.

However, sometimes the height is quite unstable. One of the reasons for this might be that the performance varies with simulated time. For such cases, parameter N, the number of samples for least-square approximation, should be larger.

6 CONCLUSION

The Adaptive Time-Ceiling (ATC), a practical mechanism for reducing speculation errors without depending on target problems, is presented. ATC is expected to enhance the efficiency of the Time Warp mechanism.

The ATC has advantages over similar approaches such as the Moving Time Window (MTW) and the Bounding Window (BW). First, the ATC can automatically find the optimum Time Ceiling height at a very low cost. Thus, ATC makes the Time Warp mechanism applicable to a wide range of problems. Second, the Time-Ceiling is updated less frequently than the Time Window is in MTW. This means that the time-consumption for updating GVT is much smaller than MTW, and so ATC is much more suitable for distributed memory machines. Third, the risk of processor idling is much reduced compared to BW because ATC updates GVT before completion of event simulation below the current Time-Ceiling.

We evaluated the effectiveness of the ATC by applying it

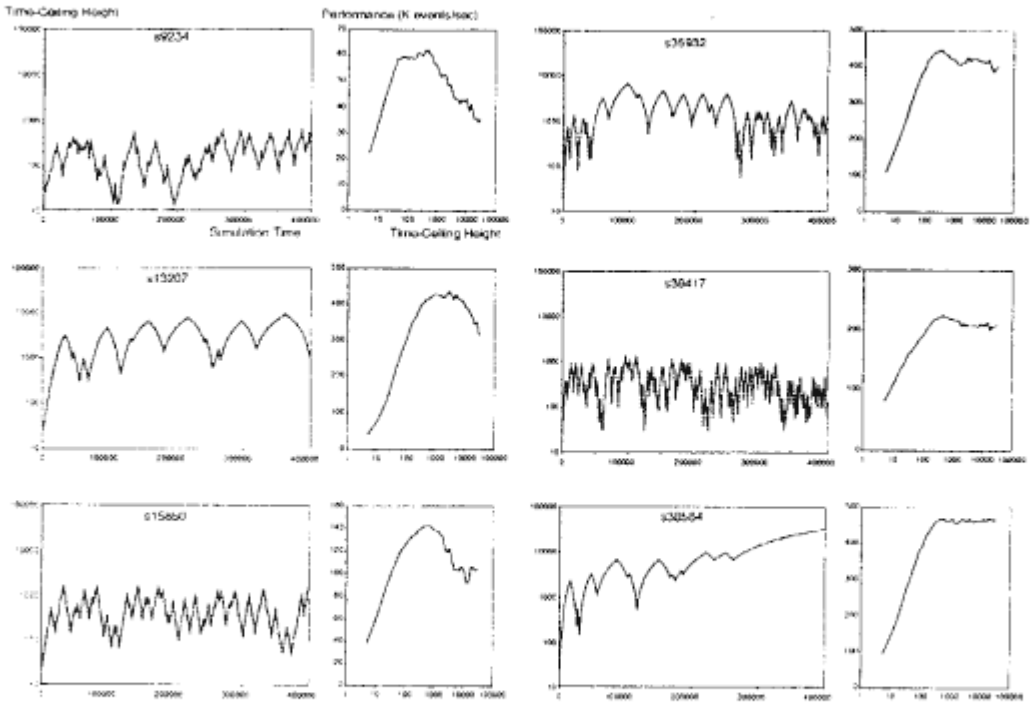

Figure 7: Transition of the Time-Ceiling height and the height vs. performance

to parallel logic simulation. Our experiments revealed that ATC enhanced simulator performance, especially when using many processors. When using 256 processors, the performance was improved, on average, by 37 % over that without ATC. Although the overheads of ATC decreased performance sometimes, this decrease is so slight that it can be ignored.

One of the remaining problems is the appropriate selection of a few parameters such as the limit frequency of speculation errors and the sampling number N, although its sensitivity is much lower than that in methods where the size of the time window must be specified. The height fluctuation problem should also be solved.

References

[Briner 1991] J. V. Briner *et al.* 1991. "Parallel Mixed-level Simulation using Virtual Time." *CAD accelerators*, North Holland, pp. 273–285.

[Fujimoto 1990] R. M. Fujimoto. 1990. "Parallel Discrete Event Simulation." *Communications of the ACM*, Vol.33, No.10, pp. 30–53.

[Jefferson 1985] D. R. Jefferson. 1985. "Virtual Time." *ACM Trans. on Programming Languages and Systems*, Vol.7, No.3, pp. 404–425.

[Matsumoto 1992] Y. Matsumoto and K. Taki. 1992. "Parallel Logic Simulation on a Distributed Memory Machine." In *Proc. 1992 European Conf. on Design Automation*, pp. 76–80.

[Misra 1986] J. Misra. 1986. "Distributed Discrete-Event Simulation." *ACM Computing Surveys*, Vol 18, No.1, pp. 39–64.

[Nakashima 1992] H. Nakashima *et al.* 1992. "Architecture and Implementation of PIM/m." In *Proc. Int. Conf. on Fifth Generation Computer Systems*, pp. 425–435.

[Sokol 1988] L.M. Sokol *et al.* 1988. "MTW: A strategy for scheduling discrete simulation events for concurrent execution." *SCS Multiconference on Distributed Simulation*, pp. 34–42.

[Taki 1992] K. Taki. 1992. "Parallel Inference Machine PIM." In *Proc. Int. Conf. on Fifth Generation Computer Systems*, pp. 50–72.

[Ueda 1990] K. Ueda and T. Chikayama. 1990. "Design of the Kernel Language for the Parallel Inference Machine." *The Computer Journal*, Vol.33, No.6, pp. 494–500.