Proc. Summer Computer Simulation Conference, San Diego, CA, July 2002

Searching for Parallelism in Discrete Event Simulation

Gilbert Chen and Boleslaw K. Szymanski
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street, Troy, NY 12180, U.S.A.

Keywords: PDES, lookahead, lookback, impact time, supercriticality

Abstract

We discuss synchronization protocols for Parallel Discrete Event Simulations. We start with traditional conservative protocols based on lookahead and optimistic ones employing rollbacks. Next, we observe that a logical process may be able to change the simulation past locally (without involving other logical processes). This ability is named lookback, and is a basis for a new class of PDES synchronization protocols, named lookback-based protocols. Another notion, impact time, is introduced to identify the maximum amount of lookback. We also show that lookback is always larger than or equal to lookahead, and lookback-based protocols can circumvent the execution time limit imposed by the cumulative execution time of events on the critical path on traditional conservative protocols and optimistic protocols with no optimization.

INTRODUCTION

Simulation is the technology that uses special devices, in most cases digital computers, to replicate the behavior of the system under investigation. A simulation can, if constructed correctly, represent the real system with a high degree of fidelity. The dynamic characteristics of the real system can then be easily inferred from the results produced by the simulation.

Discrete event simulation is a special class of simulations where variables representing physical states are discretized and change only at a countable number of points in the simulation time. An event represents a change in the state variables. The simulation time at which the event occurs is called the timestamp of the event. Many real systems can be abstracted in this way, making discrete event simulation a useful technique in practice.

The goal of Parallel Discrete Event Simulation (PDES) is to run discrete event simulation on multiple processors, in order to simulate large-scale complex systems as fast as possible. Because of the assumption that event granularity is so small that little benefit can be obtained by parallelizing single events, the only way of achieving substantial parallel efficiency is to execute events concurrently on different processors. However, events executed in parallel cannot affect each other, hence comes the fundamental problem of PDES [1]: how do we know that events do not affect each other without actually performing the simulation?

The research of PDES is largely centered on the solution of this fundamental problem. Parallel simulation results match sequential execution results if the causality between events is preserved during the parallel execution. Traditional PDES approaches thus focus on synchronization mechanisms that would not violate the causality constraint. We realized, however, the preserved causality is merely a sufficient, not necessary, condition to ensure matching of the results of parallel and sequential simulations. With relaxed causality constraint, more parallelism becomes available.

In this paper, we first give a brief overview of current PDES research. We then present a new class of synchronization protocols that differ significantly from classical PDES protocols. We will also discuss some fundamental issues in the PDES literature, such as lookahead, supercriticality, as well as several new concepts, such as lookback, impact time, which may bring new insights into the PDES research.

CONSERVATIVE AND OPTIMISTIC

The advent of PDES was marked by the invention of conservative protocols, the first of which was the null message protocol, also known as the Chandy/Misra/Bryant protocol [2,3], developed in 1979. Conservative protocols require each logical process to broadcast to its neighbors, in the form of null messages, a lower bound on the timestamp of events it will send to other logical processes, or Earliest Output Time (EOT). By listening to the null messages from all neighbors, each logical process can determine the lowest timestamp of any events it will receive, or Earliest Input Time (EIT). If the EIT is larger than the timestamp of the earliest event in its local event list, the logical process is certain that this earliest event can be processed without violating the causality constraint. Otherwise, the logical process has to block until the earliest local event is safe to be processed.

In 1985, Jefferson proposed a new synchronization protocol, called Time Warp [4]. In the Time Warp and other optimistic protocols, a logical process is allowed to aggressively process local events, and to send to other logical processes new messages generated by the event execution. However, when an event arrives with a timestamp smaller than the local simulation time, which is called a *straggler*, a causality error is triggered. All processed local events later than the straggler must be undone, and anti-messages must be sent to other logical processes to cancel messages sent during the execution of these events. Ironically, although they are called optimistic, these protocols are rather pessimistic, because they assume that every operation is unsafe and subject to a rollback, and therefore they have to save every change made to the state variables in order to be able to recover from the erroneous event execution. The Global Virtual Time (GVT) gives a lower bound on the timestamp of the earliest unprocessed event in the simulation. Any event processed with simulation time earlier than the GVT is committed, in the sense that it will never be rolled back. For such events, the logical process can reclaim the memory used to store the associated state (or state changes if incremental state saving is used).

Research on PDES has been largely dominated by the studies of conservative and optimistic protocols and comparison of their performance. Unfortunately, both types of protocols have their strengths and weaknesses. Efficiency of conservative protocols is limited by the amount of lookahead, which does not exist in many simulation models. Additionally, null messages required to collaboratively advance the simulation clock in conservative protocols often incur significant overhead. As a result, parallelized execution may be even slower than the sequential one. On the other hand, optimistic protocols do not depend on lookahead and null messages. However, state saving usually requires storing and accessing large amounts of memory. This negatively impacts the speed of execution because of the relatively slow improvement in the memory access speed within the current VLSI technology. The handling of anti-messages complicates the simulation model development. Furthermore, as Nicol and Liu pointed out [5], optimistic models may exhibit unexpected behavior caused by inconsistent messages resulting from rollback inconsistencies and stale states.

RELAXED CAUSALITY CONSTRAINT

The causality constraint dictates that events must be processed in the timestamp order, because otherwise causality errors might occur, which may lead to incorrect simulation results. Both conservative and optimistic protocols obey the causality constraint. Conservative protocols strictly avoid causality errors, while optimistic protocols take a detection and recovery approach.

However, causality errors may not always violate simulation correctness. Events can be processed in any order, as long as the simulation results are not affected. The key to ensure the correctness is to use a different event handler to process events, an event handler that is aware of the out-of-timestamp status and able to repair the causality error.

One example that has always been used to illustrate the importance of preserving the causality constraint is the missile and tank example. A missile is fired, and the tank is hit and explodes. One would argue that if the causality constraint is violated, an observer may see the explosion first, then the fire. However, if the observer is clever enough to remember events, he would not be surprised to see the explosion without missile firing, and can construct the complete scenario once the fire event is received.

When a logical process is free to process any event, a natural choice is to select the earliest local event whose timestamp becomes the current simulation time. What would happen if later there are events received from other logical processes? Events with a timestamp larger than the current simulation time can be simply put into the local event list for later processing, while events with a timestamp smaller than the current simulation time, or stragglers, are better to be processed as soon as possible.

This requires that logical processes must be capable of processing stragglers correctly, in other words, to change the past. Optimistic protocols, in fact, are able to process stragglers, if the rollback and reprocessing of affected events are viewed as part of the straggler handler. However, to avoid anti-messages, it must be guaranteed that the logical process alone is able to handler stragglers, and we define this ability to change the past *locally* as *lookback* [6].

LOOKBACK

Formally, if at simulation time T, a logical process can process stragglers with timestamp down to *T-L*, without sending anti-messages, this logical process is said to contain a lookback of *L*. The window *[T-L,T]* is called the lookback window, and the lower end of the window, *T-L*, is referred to as the *virtual lookback time* (VLT). The procedure used to process stragglers that fall in the lookback window is named the *lookback procedure*.

Lookback-based synchronization protocols were originally aimed at reducing the rollback frequency in optimistic simulation. Stragglers with a timestamp larger than the virtual lookback time can be processed by the lookback procedure, thus the rollback and recovery procedure can be less frequently invoked.

It was found out later that lookback-based protocols can completely eliminate rollbacks. If logical processes choose to advance the simulation clock cautiously such that all stragglers received later would have a timestamp larger than the virtual lookback time, the lookback procedure alone is sufficient to handler all stragglers, rendering the rollback and recovery procedure totally unnecessary. Denoting as LBTS the lower bound on the timestamp of any events that will be received later, the constraint that the LBTS must always be larger than or equal to the virtual lookback time is referred to as the *lookback constraint*.

In order to adhere to the lookback constraint, logical processes, before processing an event, must determine if the execution of the event would violate the lookback constraint. This requires that logical processes be able to pre-compute the virtual lookback time without executing the event. The pre-computation may be very fast. For example, in an FCFS (First-Come-First-Served) server, when a packet is about to leave the server, the virtual lookback time after the packet leave is equal to the arrival time of the same packet, because the departure event cannot be affected by any other packet arriving later. In this case, the pre-computation of the virtual lookback time may need just a few memory references.

The following gives the main algorithm of the lookback-based protocols where rollbacks never occur. By comparing the timestamp of the earliest local event *e* with the current simulation time, it can be determined if *e* is a normal event or a straggler. Either the normal event handler or the lookback procedure can then be chosen to process event *e*.

```
while (the termination condition is not met)
    find the earliest local event e
    pre-compute VLT based on e
    if VLT > LBTS
        process e
    else
        update LBTS
    end if
end while
```

Depending on the policy adopted to estimate LBTS, several variants can be derived. LB-GVT uses GVT as the estimation of LBTS, while LB-EIT approximates LBTS with EIT. Because the computation of GVT does not take into account the interconnections between logical processes, it is a less accurate estimation of the LBTS, but it is also less sensitive to the topology. In contrast, the LB-EIT protocol requires that each logical process maintain its own EIT based on EOTs received from neighbors, thus helping to improve the accuracy of the estimation. We showed that LB-GVT is deadlock free, but LB-EIT is prone to deadlock, when a cycle of logical processes is formed [6]. Every logical process in the cycle assumes the earliest event will come from others. Hence, for the LB-EIT protocol we have developed an algorithm for deadlock detection and recovery that does not rely on the existence of lookahead.

LOCAL ROLLBACK

The aforementioned lookback procedure attempts to process stragglers directly. For some simulation models, the direct lookback procedure may be quite simple. In an FCFS server, for instance, a straggler packet can be simply inserted into the proper place in the waiting list, instead of being appended to the tail of the waiting list. However, it might be very difficult to implement the direct lookback procedure, especially if there are more than two types of events in a logical process, because in such cases the number of combinations of event arrivals increases exponentially.

A *universal lookback procedure* can be used when the direct lookback procedure is too complicated to implement. The universal lookback procedure works almost the same way as optimistic protocols. Upon arrival of a straggler, the universal lookback procedure rolls back all processed events with a timestamp larger than the timestamp of the straggler, in decreasing timestamp order (no anti-messages are necessary during this step). Then, it processes the straggler, and finally re-executes the events that have been rolled back, in increasing timestamp order.

It is interesting that lookback-based protocol with the universal lookback procedure fall into the class of synchronization protocols known as local rollback [7], the idea of which is to avoid anti-messages but to allow events to be aggressively processed. Two such protocols previously known, SRADS with local rollback [7] and Breathing Time Window [8] require logical processes to work collaboratively, by exchanging event information, to avoid sending any erroneous message. In lookback-based protocols, however, each logical process alone can determine whether a message can be sent out (by allowing or disallowing execution of the corresponding event).

IMPACT TIME

It is evident now that lookback enables a new class of synchronization protocols for PDES. The subsequent question is, how do we know whether or not lookback exists, and, if it does, how large it is?

Clearly, since lookback is defined as the ability to change the past locally, it is not unreasonable to believe that any change made to local state variables can be aggressively repaired by some means, possibly by the universal lookback procedure. Because shared variables are usually excluded in the logical process paradigm, the lookback would be infinite if no messages have been sent out during the event execution. If the event execution produces messages, the lookback may not necessarily be zero. It actually depends on a property of the message being sent out.

We define the *impact time* of a message as the lower bound on the timestamp of any stragglers that can change or cancel the message. If the event execution produces exactly one message, the impact time of the message gives the largest lookback available. If a straggler comes with a timestamp smaller than the impact time of the message, this straggler cannot be processed by the lookback procedure, because it may change the message, thus requiring an anti-message to be sent to cancel out the old message, which is strictly prohibited by the lookback-based protocols. If the event execution produces more than one message, the maximum amount of available lookback is equal to the maximum of the impact times of all messages.

Besides serving as an indication of how much lookback is available, the notion of impact time has another practical use. In optimistic simulation, if a straggler does not affect the message produced by an event later than the straggler, it is always beneficial not to deliver a corresponding anti-message. Lazy cancellation [9] is one of the well-known techniques that can achieve this goal. Upon receiving a straggler, it does not immediately cancel out messages affected by the straggler. Instead it rolls back affected events, processes the straggler, and re-executes affected events. In the process, it compares the old messages with the new ones produced by the re-execution of events. If they are the same, anti-messages are suppressed. Otherwise, anti-messages are still required.

The main problem of the lazy cancellation is that it delays the sending of the anti-messages. The delay causes incorrect messages to be more likely processed by other logical processes. Also, the comparison of messages incurs some overhead.

Impact time provides a better method to avoid unnecessary anti-messages. When a straggler comes, messages whose impact time is larger than the timestamp of the straggler need not be cancelled, because by the definition of impact time it is impossible for the straggler to affect those messages. Therefore, only a timestamp comparison is required for anti-message avoidance, and there is no delay in the propagation of anti-messages.

MESSAGE STRENGTH

What if the impact time of a message is equal to its timestamp, which means that there is no lookback? In such cases, there is still a chance we could suppress the sending of anti-messages. This depends on another property of a message, *strength*.

The strength of a message is defined to be the number of stragglers needed to affect this message. Since stragglers with a timestamp larger than the impact time of the message cannot affect the message by definition, here we only count the number of stragglers earlier than the impact time.

With the notion of message strength, a message may not need to be cancelled even though the straggler has a timestamp smaller than the impact time. Instead, the strength of the message is checked first. If it is already zero, an anti-message must be immediately sent out. Otherwise, the strength of the message is decremented, and the anti-message is unnecessary.

LOOKBACK VERSUS LOOKAHEAD

We have introduced the concept of lookback and presented lookback-based synchronization protocols that may use either direct lookback procedure or universal lookback procedure. A question regarding the usefulness of lookback-based protocols still remains. How frequently would lookback be found in real world simulation models?

This question does not seem to have a straight answer. However, we found that there is a connection between lookahead and lookback, which justifies the usefulness of lookback-based protocols.

Strangely enough, lookahead, informally known as the ability to predict the future, has no consistent definition. In Fujimoto's definition [10], a logical process is said to contain a lookahead of L at time T if it can schedule events with timestamp at least $T+L$. This definition does not consider the case in which previously scheduled messages may not have left the logical process. For instance, it is possible that the next outgoing message that is scheduled earlier contains timestamp $T+L/2$. The lookahead now is $L/2$, not L.

Another definition given by Jha and Bagrodia [11] defines lookahead as the differential between the EIT and the EOT. This definition does not consider local events. If the logical process is in the middle of processing a local event, the lookahead should be the differential between the timestamp of the local event being processed (which is the current simulation time) and the EOT. Therefore, a more accurate definition of lookahead at time T is the difference between the current simulation time and the EOT.

At the first glance, lookahead and lookback seem completely unrelated, because the former deals with the future, while the latter deals with the past. However, if the simulation clock is aggressively advanced, future will suddenly become past. The relationship between lookahead and lookback is therefore easy to understand.

We proved that lookback is always larger than or equal to lookahead [6]. When at time T there is a lookahead of L, we can aggressively advance the simulation clock to $T+L$, and process stragglers, if there are any. The property of lookahead guarantees that no message with a timestamp no smaller than $T+L$ can be affected by stragglers earlier than T, making the lookback at least L.

With careful examination, we can see that by lookahead a logical process actually makes two guarantees. One ensures the validity of the message, while the other promises a lower bound on the timestamp of messages that will be delivered later. Lookback removes the second guarantee, therefore it is no surprise that lookback is always more common than lookahead, and that lookback is exactly equal to lookahead when the timestamp of the sent message is also the lower bound on the timestamp of any future message, i.e., messages are delivered in timestamp order.

SUPERCRITICALITY

It has been generally accepted that conservative protocols, and even optimistic protocols without any optimization such as lazy cancellation (or impact time described above), cannot beat the cumulative execution time of events on the critical path [12]. In contrast, there are two cases in which lookback-based protocols can achieve supercriticality.

The supercriticality comes from independency, as observed by Gunter[13]. The property of lookback enables us to exploit such independency with little overhead. For instance, when an event is independent of its predecessor (the preceding event in the same logical process), it can be processed before its predecessor. The lookback constraint promises that the predecessor can always be correctly processed, even though it arrives later, otherwise the event would not have been executed. Even if event itself is affected by the predecessor, its messages are not. If the late arrival of the predecessor invalidates the event, the event has to be re-executed. However, the message produced during the first execution of the event is unaffected, which is guaranteed by the lookback constraint. The net effect is that the message is produced before the completion of the event that schedules the message.

Jefferson and Reiher characterized conservative protocols as mechanisms that never use “any form of event undoing, abortion, or rollback” [12]. According to this definition, lookback-based protocols with direct lookback procedure are conservative, in which every processed event is committed. The difference between them and the traditional conservative protocols is that the state is not committed.

Jefferson and Reiher proved that conservative protocols are bound by the critical times of events under the assumption that all correct conservative simulation mechanisms must use *elementary scheduling*, otherwise the simulation might be incorrect [12].

However, they consider neither mechanisms that ensure that stragglers can always be executed correctly, nor mechanisms other than guessing that allow a message to be sent before the completion of its antecedent. The lookback-based protocol guarantees that once an event is about to execute, all predecessors of this event, if there are any, can also be processed correctly. If it cannot make such a guarantee, the event processing has to be postponed. Also, event execution is no longer atomic in lookback-based protocols. The execution of an event may be composed of two parts, one by its normal event handler and the other by the lookback procedure of its predecessor (if it arrives later), and a message produced by the first execution is guaranteed to be correct, according to the notion of lookback. If it were not, the event would fail to be processed by its normal event handler and its execution would be delayed until the lookback constraint is satisfied.

CONCLUSION

We have presented in this paper synchronization protocols for Parallel Discrete Event Simulation and focused on the novel lookback-based protocols that are capable of archiving supercriticality. Lookback, the ability of change the simulation past locally, has been proven to be always larger than or equal to lookahead, the ability to predict the future. This is opposite to the real world, where changing the past is impossible while the future is more or less predictable.

The key idea behind lookback-based protocols is the relaxed causality constraint. The causality constraint, the sequencing constraint imposed on simulation by the physical cause-and-effect relationship, is no longer necessary if logical processes have more control over the simulation time, i.e., they are aware of causality errors and able to aggressively repair the damages by themselves. Adherence to the lookback constraint, which states that the virtual lookback time must always be no smaller than the LBTS, grants logical processes such capabilities.

Lookback-based protocols have been shown to be both theoretically and experimentally faster than lookahead-based conservative protocols [6]. The effectiveness of impact time and message strength in optimistic simulation has yet to be proven by extensive experiments.

ACKNOWLEDGEMENT

This work was partially supported by the NSF Grant KDI-9873139. The content of this paper does not necessarily reflect the position or policy of the U. S. Government – no official endorsement should be inferred or implied.

REFERENCES

[1] Fujimoto, R.M., *Parallel Discrete Event Simulation.* CACM, 1990: p. 30-53.

[2] Chandy, K.M. and J. Misra, *Distributed Simulation: A Case Study in Design and Verification of Distributed Programs.* IEEE Transactions on Software Engineering, 1979. **SE-5**: p. 440--452.

[3] Bryant, R.E., *Simulation of Packet Communications Architecture Computer Systems*. 1977, Technical Report MIT-LCS-TR-188, Massachusetts Institute of Technology.

[4] Jefferson, D.R., *Virtual Time.* ACM Transactions on Programming Languages and Systems, 1985. **7**(3): p. 404-425.

[5] Nicol, D. and X. Liu. *The Dark Side of Risk (What your mother never told you about Time Warp).* in *Proc. of the 1997 Workshop on Parallel and Distributed Simulation*. 1997. Austria. p. 188--195.

[6] Chen, G. and B.K. Szymanski, *Lookback: A New Way of Exploiting Parallelism in Discrete Event Simulation*, in *Proceedings of the 2002 Workshop on Parallel and Distributed Simulation.* 2002. Washington. p. 153-162

[7] Dickens, P. and P. Reynolds. *SRADS with Local Rollback*, in *Proc. of the SCS Multiconference on Distributed Simulation*. 1990. p. 161--164.

[8] Steinman, J. *Breathing Time Warp*, in *Proc. of the 7th Workshop on Parallel and Distributed Simulation*. 1993. San Diego. p. 109--118.

[9] Gafni, A. *Rollback Mechanisms for Optimistic Distributed Simulation*, in *Proc. of the SCS Multiconference on Distributed Simulation*. 1988.

[10] Fujimoto, R. *Performance Measurements of Distributed Simulation Strategies*, in *Proc. of the Distributed Simulation Conference*. 1988. p. 14--20.

[11] Jha, V. and R.L. Bagrodia, *Transparent Implementation of Conservative Algorithms in Parallel Simulation Languages*, in *Proc. of the 1993 Winter Simulation Conference*. 1993. p. 677-686.

[12] Jefferson, D. and P. Reiher. *Supercritical Speedup*, in *Proc. of the 24th Annual Simulation Symposium*. 1991. p. 159-168.

[13] Gunter, M.A. *Understanding Supercritical Speedup*, in *Proc. of the 1994 Workshop on Parallel and Distributed Simulation*. 1994. Edinburgh, Scotland. p. 81--87.