Four Types of Lookback

Gilbert Chen and Boleslaw K. Szymanski
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street,
Troy, NY 12180, U.S.A.
{cheng3,szymansk}@cs.rpi.edu

Abstract

We present a classification that groups lookback into four types: direct strong lookback, universal strong lookback, direct weak lookback, and universal weak lookback. They are defined in terms of absolute and dynamic impact times. We discuss relationships between lookback types by considering when rollbacks and/or anti-messages are avoided. From different types of lookback, we also derive three optimization techniques for optimistic simulation and point out their advantages over lazy cancellation. Finally, we show that all four types of lookback exist in the PCS network simulation and can be exploited by either lookback-based or optimistic protocols.

1. Introduction

Lookback is the ability to change the past locally [4], as opposed to lookahead, the ability to predict the future [6]. We have shown that lookback is more commonly observed than lookahead and enables a new class of synchronization protocols [4]. More importantly, the execution time of lookback-based protocols may be lower than the bound given by the critical times of events, which is an insurmountable limit for lookahead-based conservative protocols and even optimistic protocols without optimization [9].

From a historical point of view, lookback unifies two trends in PDES research. First, it follows an idea of out-of-timestamp order execution [10, 13, 16, 18], which has been pursued, yet achieved little success for general real-world simulations. Second, it is a perfect local rollback protocol. The well-known distinction between aggressiveness and risk predicted that there should be a new class of protocols that allow aggressiveness but not risk [15]. Two such protocols [5, 17] had been already developed, but the lookback-based protocol is the first asynchronous local roll-back protocol which controls the aggressiveness by maintaining an adequate amount of lookback.

The understanding and application of lookback are still in early stages. Although lookback is always greater than or equal to lookahead, they are exactly the same in the Closed Queuing Network (CQN) simulation presented in [4]. Hence the CQN simulation did not back the claim that, in practice, lookback-based protocols have a wider application range than lookahead-based protocols do. In this paper, we classify lookback into four types, and show that all four exist in the PCS (Personal Communication Service) network simulation. PCS network simulation has been widely studied by the PDES community [1, 8, 11, 12], yet it is difficult to simulate it efficiently using lookahead-based conservative protocols. In this paper, we demonstrate that lookback-based protocol is capable of efficient parallel PCS network simulations. Moreover, we demonstrate that different types of lookback can be exploited to improve also the performance of optimistic simulations in a similar way as lazy cancellation does.

2. Four Types of Lookback

The starting point of this paper is based on a close examination of the definition of lookback in [4]. A lookback of L was originally defined as the ability of a component at simulated time T to "execute correctly, without sending out anti-messages, any received event with timestamp between $T - L$ and T", and the lookback window is defined as the time interval $[T - L, T]$. There is some ambiguity in this definition with regard to the number of stragglers (events with timestamp smaller than the local simulated time) that can be processed correctly by the lookback procedure. The first straggler in the lookback window should always be processed without difficulty. However, the definition did not clearly specify whether the second, and subsequent stragglers, can also be correctly processed. Moreover, a restriction stronger than avoidance of anti-messages can also be

imposed on lookback window, namely forbidding a component to rollback any processed events.

Based on these choices, we can define four types of lookback (Figure 1), induced by two orthogonal classifications: direct versus universal, and strong versus weak. Direct lookback exists when any event within the lookback window can be processed without a rollback (directly). Any lookback without this restriction is called a universal one. A component with weak lookback is able to process only a limited number of events within the lookback window without anti-messages (or rollbacks for direct rollback). A component with strong lookback can process any number of stragglers that way.

Figure 1. Four Types of Lookback

By definition, weak lookback always contains the corresponding strong lookback. If we assume that the event history is not exposed to the lookback procedure, universal lookback always contains the corresponding direct lookback. Therefore, universal strong lookback always includes direct strong lookback, direct weak lookback always includes direct strong lookback, etc. These containing relations are depicted by arrows in Figure 1.

The relation between universal strong lookback and direct weak lookback bears some elaboration. It is not readily clear whether one encloses the other, but their definitions imply that they have at least an intersection, the direct strong lookback.

Let us consider whether rollbacks and anti-messages can occur during the lookback procedure with each type of lookback. For rollbacks, there are three possibilities: no rollbacks, some rollbacks, and always rollbacks, denoted by NR, SR, and AR, respectively. For anti-messages, there are only two possibilities, no anti-messages (NA) or some anti-messages (SA), because anti-messages are not always necessary, if no messages have been produced during the event execution. Of the six combinations of these possibilities, only five are feasible. The case of 'NR,SA' which stands for 'No Rollback and Some Anti-messages' is contradic-

tory, because an anti-message may trigger a rollback at the recipient (depending on its progress after the corresponding positive message was received). Let us denote these five cases as 1,2,3,4,5 respectively, as shown in Table 1.

Case	Rollbacks	Anti-messages
1	No Rollbacks	No Anti-messages
2	Some Rollbacks	No Anti-messages
3	Always Rollbacks	No Anti-messages
4	Some Rollbacks	Some Anti-messages
5	Always Rollbacks	Some Anti-messages

Table 1. All Five Possible Cases of Rollbacks and Anti-messages

It is now clear that with universal strong lookback, cases 1,2, and 3 can occur, while with direct weak lookback, cases 1,2, and 4 are possible. This infers that there is no containing relation between them. Direct strong lookback corresponds to case 1, while for universal weak lookback, the weakest form of the four types, all five cases could occur. Their relations are illustrated in Figure 2.
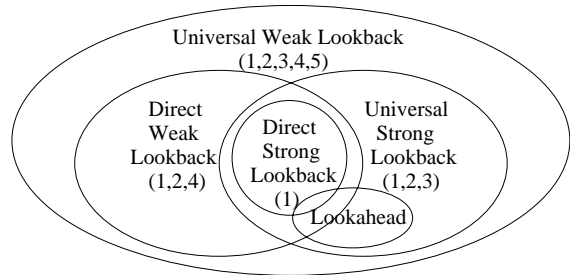
Figure 2. Relations of Four Types of Lookback and Lookahead

The lookback discussed in [4], which forms the basis for lookback-based protocols, is actually universal strong lookback. We can also deduce the position of lookahead in the picture. Since lookahead cannot be larger than the universal strong lookback, it must be strictly contained in the universal strong lookback. The claim that lookahead and direct strong lookback only intersect can be supported by three examples. First, in a server with an infinite capacity (all jobs start service as soon as they arrive), if the service time obeys an exponential distribution, the direct strong lookback is infinite while the lookahead is zero, so this is a case where direct strong lookback exists while lookahead does not. Second, if the service time is constant for any job, the direct strong lookback is still infinite but the lookahead is now equal to the service time, and both direct strong lookback and lookahead exist. Third, if departing jobs are dependent on each other, for instance, if each job contains

a field that records how many jobs are in the server at the time it arrives, the strong lookback becomes universal. This is because previously arriving jobs with timestamps larger than the straggler must be rolled back, if we assume that it is forbidden to directly modify the processed event list from the event handler of the arriving event. Therefore, this is an example where there is lookahead but no direct strong lookback.

Weak lookback does not support lookback-based protocols because the component has no knowledge of how many stragglers would come and therefore some of them may violate the lookback constraint. This can be demonstrated by looking at the range of the lookback function.

Theorem 1. $LB(e, T)$ *denotes the function that gives the value that the virtual lookback time would have if the event e were executed at simulated time T. For strong lookback, $LB(e, T) \leq ts(e)$ where $ts(e)$ denotes the timestamp of event e.*

Proof. Suppose that at simulated time T, e is a straggler and $LB(e, T) > ts(e)$. After e were processed, the virtual lookback time would have become equal to $LB(e, T)$. Now consider another straggler e'. If $ts(e') \geq ts(e)$, it is possible that $ts(e') < LB(e, T)$, which implies that e' could not be correctly processed by the lookback procedure, contradicting the definition of strong lookback. Therefore by contradiction $LB(e, T) \leq ts(e)$. $\square$

For weak lookback, $LB(e, T)$ might be larger than the timestamp of the straggler e. In such a case, the execution of the unprocessed event with the smallest timestamp in the entire simulation may increase the virtual lookback time to a value larger than the minimum timestamp of the remaining unprocessed events. Then, processing any of those events would violate the lookback constraint, so these events have to resort to the rollback and recovery procedure. Hence, weak lookback alone is not sufficient for lookback-based protocols.

3. Finding Lookback

How do we know whether or not lookback exists, and, if it does, to which type it belongs and how large it is?

To identify universal lookback, it is reasonable to assume that any change made to local state variables can be aggressively repaired by some means. Shared variables are usually excluded in PDES, so the universal lookback is infinite if no messages have been sent out during the event execution. Even if the event execution produces messages, the universal lookback may not necessarily be zero. It actually depends on a property, that we termed *impact time*, of the message being sent out.

We define the *impact time* of a message as the upper bound on the timestamp of any stragglers that can change or cancel this message. We further distinguish *absolute impact time* from *dynamic impact time*. The former cannot be changed by the arrival of a straggler, so any number of stragglers with a timestamp larger than the absolute impact time of a message cannot affect the messages. The latter is subject to change after a straggler has been processed, so only a limited number of stragglers can be correctly handled. One can easily deduce that the absolute impact time induces universal strong lookback while the dynamic impact time implies universal weak lookback. In both cases, if a straggler comes with a timestamp smaller than the impact time of the message, this straggler cannot be processed by the lookback procedure. The reason is that such a straggler may affect the message, thus requiring an anti-message to cancel it. This is strictly prohibited by the lookback-based protocols but allowed in optimistic protocols.

If the event execution produces exactly one message, the impact time of the message gives the largest possible lookback. If more messages are produced, the lookback is equal to the local simulated time minus the maximum of the impact times of all messages.

The distinction between direct and universal lookback depends on whether the direct access to the processed event list is allowed from the event handler of the straggler. If the direct access is not allowed or only a part of the processed event is exposed to the event handler, then it is not always possible to process stragglers directly. Direct lookback exists only when it is possible to do so. This is the assumption behind Figure 2. On the other hand, if the direct access is granted, then the component can take aggressive actions to repair all processed events. The net result is as if all events arrived in timestamp order. In such a case, the direct and universal lookback are exactly the same.

4. Lookback-Based Optimization versus Lazy Cancellation

Based on the concept of different types of lookback, we can design three techniques to avoid unnecessary rollbacks and anti-messages in optimistic simulation. The first technique, based on direct lookback, can directly process a straggler without rolling back any other events, when a straggler is found to have a timestamp greater than the virtual lookback time,

The second and third techniques exploit universal lookback. If the straggler has a timestamp smaller than the virtual lookback time, the protocol has to resort to the rollback and recovery procedure. While rolling back processed events that are later than the straggler, the protocol can compare the impact times of the messages generated by processed events with the timestamp of the straggler. If the im-

pact time of a message is smaller than the timestamp of the straggler, it cannot be affected by the straggler, and therefore an anti-message is unnecessary. The only difference between the absolute and dynamic impact times is that the latter must be reset as each straggler is processed. As we discuss later, often the value set after one straggler is equal to the absolute impact time.

Lazy cancellation [7] is a well-known technique of avoiding sending unnecessary anti-messages when the previously delivered messages are not affected by a straggler. This technique first compares the new messages generated by the reprocessing of a rolled-back event and the old messages generated in the first execution of the same event. If they are the same, anti-messages are prevented. The importance of lazy cancellation is that it is one of the techniques that enable optimistic simulations to circumvent the execution time limit imposed by the critical times of events. In practice, it has been shown to be slightly, within 10%, faster than aggressive cancellation in some applications [14].

It is still unclear, however, in what applications lazy cancellation could perform better than aggressive cancellation. Fujimoto suggests that lazy cancellation automatically exploits lookahead, and hence simulations with good lookahead would be more amenable to lazy cancellation [6]. With the notion of lookback, we can claim that what is automatically exploited by lazy cancellation is actually lookback, or more specifically, all four types of lookback.

When a component contains a certain amount of lookback, it may be able to process at least some stragglers without sending anti-messages, or even without rollbacks, if the lookback is direct. Indeed, messages generated in the execution of events whose lookback window contains the straggler will not be affected by it. Hence, lazy cancellation will discover that these messages do not require anti-messages. Therefore, lazy cancellation benefits from the existence of lookback. This, however, does not invalidate the claim that lazy cancellation exploits lookahead, since lookahead is contained within universal strong lookback.

It is interesting to compare lazy cancellation and the optimization techniques based on lookback. Lazy cancellation avoids all unnecessary anti-messages. Lookback, on the other hand, might cancel the original message with a corresponding anti-message and then send the same message again. This happens when two different executions, one with the straggler and the other without, accidentally produce the same value.

We believe, however, that such accidental independence occurs rarely in real world. Most often, lazy cancellation and lookback-based optimization produce the same effect. In these cases, the latter would be a definite win, for it determines the necessity of anti-messages by a simple comparison of two floating point numbers. That is, lookback just needs to compare the timestamps of a straggler with the im-

pact time of a message (which is the virtual lookback time of the component, when the message is delivered). Lazy cancellation, however, determines the validity of a message only after the message is re-generated, thus it not only delays the propagation of anti-messages, if the message needs to be cancelled, but also incurs significant overhead, if the message is represented by complex or large data structures.

5. Exploiting Lookback in PCS Network Simulation

A PCS (Personal Communication Service) network is composed of a geographically distributed radio base stations. The portables, carried by users in the coverage area (or cell) of a base station, can use channels assigned to that station. The number of channels allocated to each cell may be smaller than the number of portables simultaneously active in the cell, so available channels are assigned in the order of requests. A channel can be occupied by a portable for a random call time and then released. A block occurs when a portable requests a channel while all channels in the cell have been allocated to other portables. When a portable moves from one cell to another during a phone call, a handoff is said to occur. In this case, the portable releases the occupied channel to the old cell and attempts to get a channel from the new cell.

The PCS network simulation actually represents a class of more general systems in which various mobile objects roaming around a spatially divided domain compete for limited number of resources in each location. Many real systems can be abstracted this way, such as spatially explicit problems and cellular automata, so the study of the PCS simulation may have important consequences for all these systems. We chose it also because the PCS network is hard to be efficiently simulated by traditional lookahead-based conservative protocols. This is because the time that a portable stays in a cell is usually drawn from an exponential distribution, which has no minimum value, thus resulting in zero lookahead.

We now show that all four types of lookback exist in PCS simulation.

5.1. Direct Strong Lookback

The requirement for the existence of direct strong lookback is that a component must be able to process stragglers without rollbacks while maintaining the lookback constraint.

In the PCS simulation, a straggler is always a new portable moving in from a neighboring component that has a smaller local simulated time. If this portable is not in the middle of a call, and if it does not make any new calls be-

fore the simulated time of the current cell, it obviously will not affect any other processed events; otherwise, it may.

Direct processing can be implemented in the following way (Figure 3). Each cell maintains two lists. One is the portable list and the other contains the processed event list. Each event contains four fields: the timestamp, the type of the event, the pointer to the portable in which the event occurred, and the number of the available channels before the event is processed. The event type could be either GET or RELEASE.
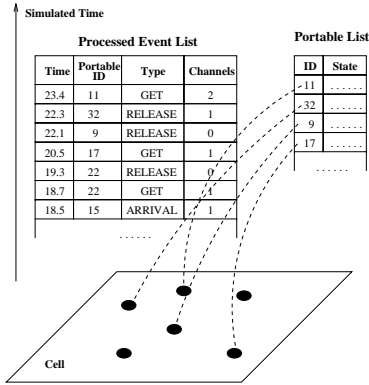

Figure 3. Data Structure for Exploiting Direct Weak Lookback in PCS Simulation

The last field is essential: it allows a reconstruction of the history of channel allocations and releases by scanning through the event list. To process a straggler, a new event corresponding to the straggler is inserted into the event list, and other events with a larger timestamp are modified, if necessary, one by one in increasing timestamp order. Notice that a change of the number of available channels may or may not invalidate the event. Only when this number changes from 0 to 1 or from 1 to 0, the event may become invalid. In such a case, all subsequent events occurring on the same portable must be removed from the event list, and the affected events must be reprocessed in timestamp order to generate correct subsequent events.

The question of determining the size of strong lookback still remains, because it is the strong lookback upon which lookback-based protocols depend. To answer this question, we have to look at the absolute impact time of outgoing messages. In the PCS network simulation, such a message is created for a portable that wants to leave the current cell, as shown in Figure 4.

The absolute impact time of such a message is the last time the portable attempted to get a channel, whether succeeded or not, because it was the latest point at which the portable required any information from the cell. Any activities happening after that point would have no affect on the outgoing messages. Notice that a RELEASE event cannot

be directly affected by a straggler. Therefore, the timestamp of the last GET event defines the strong lookback.
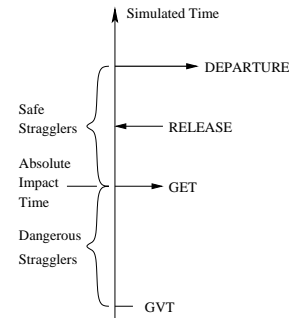
Figure 4. The Absolute Impact Time of a Departure Event in PCS Network Simulation

To apply the lookback-based protocol to the PCS simulation, a test is made, each time a portable wants to leave, to check whether or not the absolute impact time is smaller than the GVT (Global Virtual Time). If it is smaller, the portable can be safely delivered to other cells. Otherwise, the event must be suspended until this condition is met.

5.2. Universal Strong Lookback

The above algorithm that exploits direct strong lookback is quite complicated. Although the ability to reconstruct event history directly allows us to process stragglers at any time, to maintain this ability is a significant burden. It is possible to rewrite the algorithm using a rollback and recovery style, whose structure appears to be clearer and simpler. This is where universal strong lookback differs from direct strong lookback.

The processed event list is still required. However, the last field which stores the number of free channels in the event structure is no longer necessary. Instead, each cell keeps an integer which denotes the number of available channels at the current simulated time. When a straggler arrives, all events with a timestamp larger than that of the straggler are undone one by one in decreasing timestamp order. The purpose is to deduce the number of free channels at the time that the straggler is bound to occur. The straggler is processed based on this number and all rolled-back events are reprocessed in increasing timestamp order.

5.3. Direct Weak Lookback

To exploit the direct weak lookback in the PCS network simulation, we define the virtual lookback time of a cell. It is the last simulated time at which a portable released a channel at a time when there are no free channels. As

soon as a portable gets the last available channel, we set the weak lookback to zero (which means that the virtual lookback time is always equal to the local time of the cell afterwards). Theorem 2 proves that any straggler that falls into the lookback window cannot affect any processed events in the lookback window. Therefore, a portable arriving from other cells in the simulation past but within the lookback window can be granted a channel, without rolling back any processed events. To preserve the correctness, we must also set the lookback to zero after processing a straggler. Consequently, the existence of direct weak lookback enables us to reduce the rollback frequency for an optimistic PCS simulation.

Theorem 2. *If the lookback window begins with a RELEASE event that increases the number of free channels to 1 and ends with the first GET event that decreases the number of free channels to 0, any straggler arriving in the lookback window will not affect any processed event in the lookback window, except the ending GET event.*

Proof. Only GET events can be affected by a GET straggler. The condition for this happening is that the number of free channels before the GET event must be exactly 1. The only event that satisfies this condition in the lookback window is the ending GET event. The number of free channels before any other GET events must be at least 2, otherwise such an event would have been the ending GET event. The arrival of a GET straggler only decreases the number of free channels before these GET events by 1, so they can still obtain a channel and therefore remain unaffected.

In case of a RELEASE straggler, there is no event in the lookback window that can be affected. The RELEASE straggler only affects an unsuccessful GET event, that can occur only when the number of free channels is 0. However, at any point in the lookback window, the number of free channels is at least 1, because otherwise the GET event that makes this number 0 would have become the ending GET event. Therefore, a RELEASE straggler affects no events. ☐

Surprisingly, one can make use of the direct weak lookback in the lookback-based protocol that is based on universal strong lookback. The idea is almost the same as in optimistic simulations. When a straggler arrives, we first check to see if it falls into the weak lookback window. If it does, we simply process it by granting it a channel. Besides, we must set the weak lookback to zero and decrease the number of free channels by one. It is unnecessary to roll back any other events because they cannot be affected.

5.4. Universal Weak Lookback

In optimistic simulation, we can take advantage of the existence of direct weak lookback to reduce rollback fre-

quency, and the existence of universal strong lookback to avoid unnecessary anti-messages. In addition, we can exploit universal weak lookback with the notion of dynamic impact time to further reduce the number of unnecessary anti-messages.

We have known that the absolute impact time of an outgoing message in the PCS simulation is equal to the timestamp of the last GET event. The dynamic impact time of the message is at least the same as the absolute impact time. However, if, after the GET event, there is still at least one free channel, the dynamic impact time will be equal to the timestamp of the latest RELEASE (before the GET event) which increases the number of free channels from 0 to 1, as shown in Figure 5. The RELEASE event may belong to the same or a different portable. The numbers along the simulated time axis in the figure represent the number of free channels before and after each event.

The window from the RELEASE event to the GET event in the figure is part of the lookback window given by Theorem 2. As a result, a straggler with a timestamp smaller than the absolute impact time but larger than the dynamic impact time will not affect any events in the lookback window, except the ending GET event. The GET event shown in the Figure 5 is still in the lookback window starting from the RELEASE event, and it cannot be the ending GET event because after its execution there is still one free channel. Therefore, it cannot be affected by the straggler and the outgoing departure message remains unchanged.
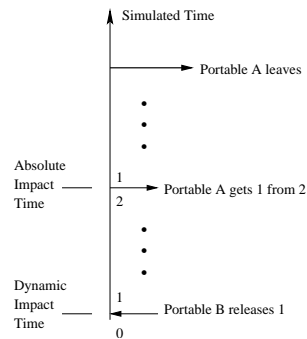
Figure 5. The Dynamic Impact Time of a Departure Event in PCS Network Simulation

A second straggler with a timestamp smaller than the absolute impact time but larger than the dynamic impact time cannot always be processed correctly without changing the departure message. As a result, we must set the dynamic impact time to absolute impact time after processing the first straggler. This is the reason that we call it 'dynamic'. It is also worth to note that the dynamic impact time is equal to the virtual lookback time given by the direct weak lookback at the simulated time when the last GET event is being

processed. This fact greatly simplifies the implementation.

6. Performance Evaluation

Our parallel simulation platform is built as an extension of COST, a component-oriented sequential simulator [3]. We have successfully implemented two synchronization protocols: one is a lookback-based protocol called LB-GVT, and the other is an optimistic protocol that utilizes Reverse Computation [2] called LBTW.

The PCS simulation that we executed contains 256 cells. Each cell is initially assigned 16 portables. The average call time and the average idle time are 36 seconds and 18 seconds, respectively. The average time a portable stays in a cell is by default 450 seconds. The actual call time, idle time, and residence time all obey exponential distributions. All experiments ran on a shared-memory computer with 4 Intel 500Mhz Pentium III processors. The last relevant parameter is the number of available channels in each cell. In each set of experiments, this number varied from 2 to 16.
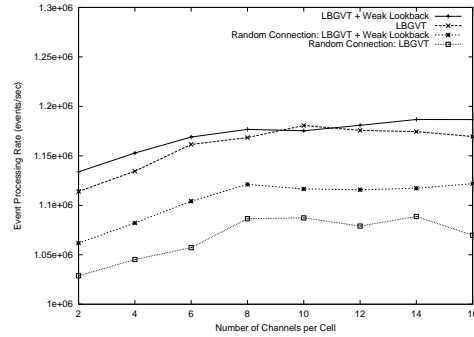
Figure 6 shows the performance of the LB-GVT protocols. Despite the fact that with universal strong lookback we may roll back processed events unaffected by a straggler, it performs better than a protocol with direct strong lookback, mainly due to a simpler design that incurs less overhead.
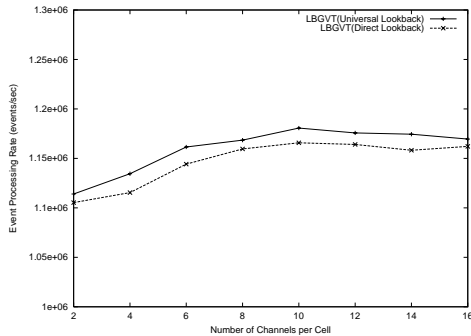
Figure 6. Performance of the LB-GVT protocol with Direct Strong Lookback and Universal Strong Lookback

As we pointed out earlier, direct weak lookback can help decrease the number of rollbacks in lookback-based protocols. Figure 7 empirically confirmed this claim. Weak lookback is able to slightly boost the performance in all but one case where each cell possesses 10 channels. In another set of experiments, we allow a portable, when it is departing, to move to a randomly chosen cell among all cells and not confined to neighboring cells. Such random connections naturally increase the rollback frequency. Consequently, event processing rates dropped, but the performance improvement resulting from the use of direct weak lookback became more apparent.

**Figure 7. Improve the LB-GVT Protocol by Exploiting Direct Weak Lookback**

Comparing Figure 7 and Figure 8, we can find that the performance of the PCS simulation with the LBTW protocol is better than with LB-GVT protocol. This is mainly because of the different implementations of the cell model. In PCS with LB-GVT we use two events to simulate a portable (one for call activities and the other for scheduling the departure event), while with LBTW, we use only one event.

We tested the three lookback-based optimization techniques and their combinations (random connection between cells was adopted, as describe earlier). Only direct weak lookback showed a very small improvement (Figure 8). However, when we changed the average residence time from 450 to 50, we obtained positive data (Figure 9). Both direct weak lookback and absolute impact time can improve the performance individually, and the combination of them is even better, obtaining an average improvement of about 3.7%. The inclusion of dynamic impact time seems to only slow down the simulation, due to the associated overhead. We also noticed from the data that the impact of direct weak lookback is less noticeable when there are few channels in each cell, but it continues to improve as more channels are available. This is no surprise since more free channels usually means more direct weak lookback.

7. Conclusion

We extended the theory of lookback by introducing its two orthogonal classifications, yielding four types of lookback. All of these types are useful in both lookback-based and optimistic protocols, as confirmed by the experiments conducted for the PCS simulations.

Although the performance improvement in the PCS simulation was rather modest, we believe that it was largely limited by the relatively small rollback frequency in simulations running on shared-memory computers. In all our ex-
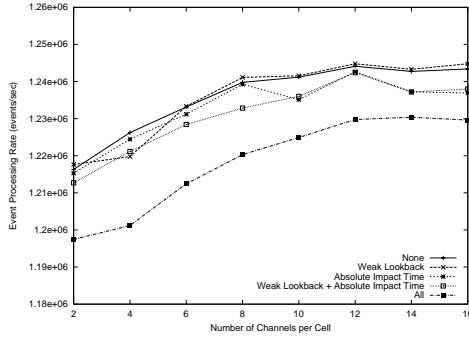
Figure 8. Optimistic PCS Simulation with Several Optimization Techniques (Average Residence Time = 450 seconds)
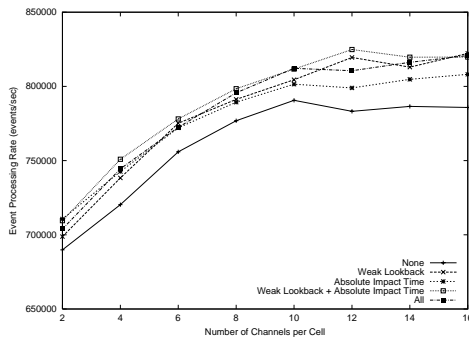
Figure 9. Optimistic PCS Simulation with Several Optimization Techniques (Average Residence Time = 50 seconds)

periments, the ratio of the number of rollbacks to the number of total events processed was less than 5%. We conjecture that lookback-based optimization techniques will yield more performance improvements for large simulations running on distributed memory parallel computers. There, messages transmission delay dominates the execution time and a considerable portion of inter-processor messages will be stragglers. Exploiting lookback to reduce rollbacks might be the only hope to obtain satisfactory speedups for simulations running on such machines.

References

[1] C. D. Carothers, R. M. Fujimoto, Y. B. Lin, et al. Distributed simulation of large-scale PCS networks. In *Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 2–6, 1994.

[2] C. D. Carothers, K. S. Perumall, and R. M. Fujimoto. Efficient optimistic parallel simulations using reverse computa-

tion. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation*, pages 126–135, 1999.

[3] G. Chen and B. K. Szymanski. COST: Component-oriented simulation toolkit. In *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[4] G. Chen and B. K. Szymanski. Lookback: A new way of exploiting parallelism in discrete event simulation. In *Proceedings of the 16th Workshop on Parallel and Distributed Simulation*, pages 153–162, 2002.

[5] P. Dickens and P. Reynolds. SRADS with local rollback. In *Proceedings of SCS Multiconference on Distributed Simulation*, pages 161–164, 1990.

[6] R. M. Fujimoto. Parallel discrete event simulation. *Communication of the ACM*, pages 30–53, October 1990.

[7] A. Gafni. Rollback mechanisms for optimistic distributed simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, pages 61–67, 1988.

[8] A. Greenberg, B. Lubachevsky, D. Nicol, and P. Wright. Efficient massively parallel simulation of dynamic channel assignment schemes for wireless cellular communication. In *Proceedings of the 8th Workshop on Parallel and Distributed Simulation*, pages 187–194, 1994.

[9] D. Jefferson and P. Reiher. Supercritical speedup. In *Proceedings of the 24th Annual Simulation Symposium*, pages 159–168, 1991.

[10] H. V. Leong and D. Agrawal. Semantics-based time warp protocols. Technical Report TRCS93-10, Department of Computer Science, University of California, Santa Barbara, 1993.

[11] Y.-B. Lin and P. A. Fishwick. Asynchronous parallel discrete event simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 26(4):397–412, 1996.

[12] B. A. Malloy and A. T. Montroy. A parallel distributed simulation of a large-scale PCS network: Keeping secrets. In C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, editors, *Proceedings of the 1995 Winter Simulation Conference*, pages 571–578, 1995.

[13] F. Quaglia and R. Baldoni. Exploiting intra-object dependencies in parallel simulation. *Information Processing Letters*, 70(3):119–125, 1999.

[14] P. L. Reiher, R. M. Fujimoto, S. Bellenot, and D. Jefferson. Cancellation strategies in optimistic execution systems. In *Proceedings of the SCS Multiconference on Distributed Simulation*, volume 22, pages 112–121, 1990.

[15] P. Reynolds, C. Weight, and J. Filder. Comparative analyses of parallel simulation protocols. In *Proceedings of the Winter Simulation Conference*, pages 671–679, 1989.

[16] L. M. Sokol, J. B. Weissman, and P. A. Mutchler. MTW: An empirical performance study. In *Proceedings of the 1991 Winter Simulation Conference*, pages 557–563, 1991.

[17] J. Steinman. Breathing time warp. In *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, pages 109–118, 1993.

[18] P. A. Wilsey, A. C. Palaniswamy, and S. Aji. Rollback relaxation: A technique for reducing rollback costs in an optimistically synchronized simulation. In *International Conference on Simulation and Hardware Description Languages*, pages 143–148, 1994.