An Empirical Performance Study of Connection Oriented Time Warp Parallel Simulation

Ali Al-Humaimidi and Hussam Ramadan
Information Systems Department, King Saud University, Saudi Arabia

Abstract: *Time warp is a well-known optimistic mechanism for parallel execution of simulation programs. Implementing time warp using a connection-oriented communication approach is proposed in the literature as a way to improve time warp performance because it allows for the use of more efficient event queue implementations. However, no empirical performance studies have been reported for connection-oriented time warp. In this paper, we present an enhanced version of the connection-oriented time warp algorithm along with its associated data structures. An empirical performance study of the connection-oriented time warp is conducted on a network of workstations using a standard synthetic benchmark simulation model. Experimental results show that this algorithm is capable of achieving better performance than that of traditional connectionless time warp for several performance measures.*

Keywords: *Parallel simulation, time warp, connection-oriented, connectionless.*

Received March 18, 2007; accepted December 13, 2007

1. Introduction

Parallel simulations attempt to speed up the execution of a simulation program by running it on a number of parallel processors in a multiprocessor machine or on a network of workstations. The simulation model needs to be partitioned in such a way that utilizes the parallel processing capabilities of the model execution platform. Specifically, the simulated system is usually modeled as a set of interacting Logical Processes (LPs) which are mapped to the different processors. Each logical process is a model of some component of the physical system, called a physical process. Logical processes interact with each other using time-stamped event messages which simulate interactions between physical processes in the real system. Each logical process has its own simulation clock and its own state variables representing a portion of the state of the corresponding physical process.

Parallel simulations synchronize the execution of simulation models on parallel processors using either a conservative approach or an optimistic approach. Conservative algorithms strictly avoid the possibility of incorrect sequencing of events by including strategies for determining which events are safe to process at each point in simulation time. Optimistic algorithms, on the other hand, use a detection and recovery approach that is based on detecting incorrect event execution order, and a rollback mechanism to recover from them [1].

Time warp, a well-known optimistic approach, is based on the idea of allowing LPs to execute optimistically until a causality error is detected. At such an instance, a rollback mechanism is initiated to recover from the incorrect computation. A causality error is detected whenever an LP receives an event message that contains a timestamp smaller than its local clock. Such a message is called a straggler message. Receiving a straggler message causes an LP to rollback in simulated time and re-executes previously-processed events whose timestamps are greater than that of the received straggler. Such an LP must cancel the effects of previously-sent erroneous messages by sending anti-messages. An anti-message is a control message that carries the identity of one of the previously-sent output messages and causes that message to be cancelled once they meet each other.

The basic time warp algorithm proposed by Jefferson *et al.* [2] assumes connectionless communication between the LPs of the simulation model. In this algorithm, an LP can send/receive messages to/from any other LP in the simulation without having to establish a connection with that LP. This leads to the creation of models with dynamic topologies where the communication links between LPs are not known prior to and may change during simulation execution. However, maintaining this feature requires expensive searches of each LP's input queue to insert newly-scheduled events during forward execution. In addition, each LP's input queue must also be searched again at rollback situations in order to cancel erroneous events. The basic time warp algorithm is shown in Figure 1.

Several studies have been made to improve the performance of time warp parallel simulations by improving the rollback and state-saving mechanisms, or by even simulation execution platforms [5]. However, Kalantery suggests [3] to improve time warp performance by changing the communication approach used in time warp from connectionless to connection-oriented. He proposes a connection-oriented implementation of time warp that uses modified and more efficient event queues, and assumes timestamp-ordered delivery of messages on each connection. The use of permanent communication channels between LPs and the use of the proposed event queues are shown to greatly reduce the event-set search requirements. This, in turn, is expected to improve the performance of time warp.

```
Each LP executes:

While (simulation termination criteria are not met) do
    IF (input-queue pointer points to NULL)
        Wait for new events to arrive
    Else
        Store received event in input queue.
        Store the state vector in the state queue.
        Execute the event.
        Store a negative copy of any outgoing.
        Message in the output queue.
        Advance the input-queue pointer.
    End else

To execute an anti-message:

    Reset input-queue pointer.
    Send the negative messages stored for all
    events between the old and new positions.
    erase the stored state vectors for those events.
    resume execution at new pointer location.
```

Figure 1. The Time Warp algorithm.

Time warp performance is highly affected by the efficiency of its event queue implementation. Researchers have empirically studied the effects of using fast priority queues to implement discrete-event simulation data sets on simulation performance. For example, a comparative study of a number of sequential and parallel priority queue implementations is reported in [6]. However, no empirical study has been reported to study the effect of changing the time warp communication pattern to be connection-oriented on time warp performance. In this paper, we report the results of an experimental study conducted to compare the performance of a connectionless and a modified version of connection-oriented time warp implementations using a synthetic simulation model consisting of a 4x4 torus topology running on a network of four workstations.

The remainder of this paper is organized as follows. In section 2, we present the requirements of implementing connection-oriented time warp. Section 3 describes the performance evaluation testbed and presents experimental results for both connection-oriented and connectionless time warp. Section 4 contains conclusions.

2. Implementing Connection-Oriented Time Warp

The time warp input queue data structure is important to performance. In connectionless time warp implementations, during forward execution, an LP's input queue must be searched to find the correct position to insert each newly received input event. Moreover, during rollback, the event set must be searched to find the event entry that must be deleted. Therefore, the connection-oriented approach to implementing time warp seeks to eliminate the need of searching input queues during the rollback and to reduce the search cost during forward execution.

Connection-oriented implementation of time warp was proposed by Kalantery [3]. It is based on the assumption that any communication link connecting any two physical processes of a simulated system is represented in the simulation model by a fixed logical channel connecting the corresponding LPs. This leads to forming static model topologies where communication channels are established before simulation execution, and cannot be changed until the end of execution. In addition, message delivery over any logical channel is assumed to be First-In-First-Out (FIFO). Based on these assumptions, a separate FIFO channel queue is associated with each channel to store messages arriving over that channel at the receiver side. With such channel queues, the event at the front of the channel queue has the smallest timestamp. When a new unprocessed event arrives on one of the LP's input channels, it is appended directly at the end of the FIFO input queue associated with that channel. With this arrangement, an LP selects for execution the event with the smallest timestamp among the front-entry events in the input queues of all incoming channels. This greatly reduces the event search space during event forward execution at each LP. The use of FIFO channel queues is depicted in Figure 2 for the case of an LP sending a sequence of three events to another LP in an increasing timestamp order.

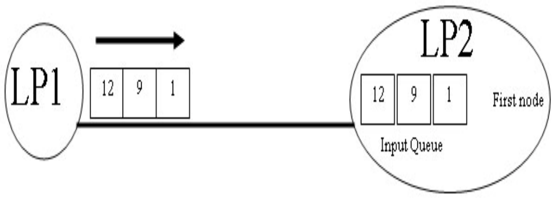

Figure 2. Ordered arrival of a sequence of three events sent by LP1 at LP2's channel queue.

Suppose that an event *e* at the front of one of LP2's input channel queues is selected for processing. In this case, event *e* is not deleted from its channel queue in order to allow for possible reprocessing in case a rollback occurs later. However, a first position pointer which points to the first unprocessed event in a channel queue is advanced forward by one step. This process is illustrated in Figure 3. Based on the above described ideas, a FIFO channel queue at any LP will consist of two sections [3]:

- Future section which contains unprocessed events.
- Past section which contains processed events.

Under connection-oriented time warp, the rollback mechanism becomes faster because, when an anti-message arrives over a channel, it finds its positive counterpart at the back of the channel queue. This eliminates the search cost for erroneous events during cancellation. An example for the direct cancellation of an erroneous event using an anti-message is shown in Figure 4. After deletion of the (processed) original message using the received anti-message, the first Position pointers of all input queues in the receiving LP are set to the input event whose timestamp is just before the timestamp of the anti-message. Moreover, the LP is restored to its earlier state which existed just before the time of the received anti-message. This way, input events are reprocessed in the same order in which they were previously processed before the occurrence of rollback. This arrangement avoids the creation of the combined input history suggested in [3] which provides the mechanism by which rolled back events can be reprocessed.

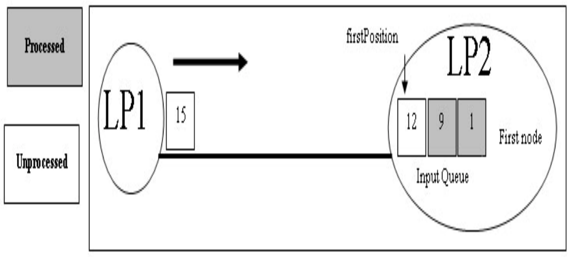

Figure 3. Advancing the first position pointer at LP2 during forward event execution.

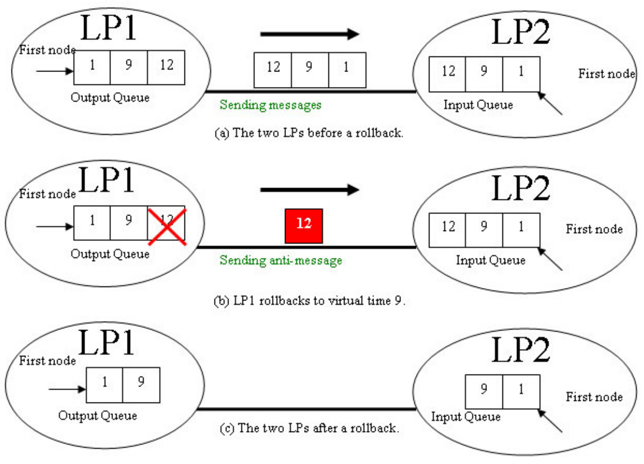
LP1
First node
1
9
12
Output Queue
12
9
1
Sending messages
LP2
Input Queue
First node
(a) The two LPs before a rollback.
Sending anti-message
(b) LP1 rollbacks to virtual time 9.
(c) The two LPs after a rollback.

Figure 4. The rollback mechanism in connection-oriented Time Warp.

With the above described connection-oriented approach for implementing time warp, we can also reduce the number of anti-messages sent over a channel. This can be achieved by sending a single anti-message to cancel its matching original message, as well as the group of erroneous messages which also need to be cancelled due to the occurrence of the rollback, and whose timestamps are greater than that of the anti-message. Figure 5 illustrates the case of a rollback occurrence at one LP, and the cancellation of a group of erroneous messages in another LP by only one anti-message.

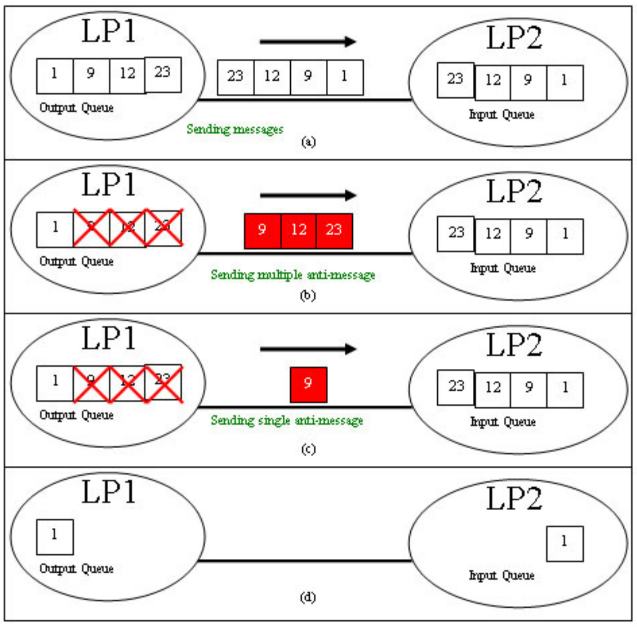

Figure 5. Example of group cancellation in connection-oriented Time Warp: (a) the two Lps before rollback, (b) LP1 sends multiple anti-messages with timestamp (9, 12, and 23) to LP2, (c) alternatively, lp1 sends a single anti-message with timestamp (9) to Lps, (d) the two Lps aft er a rollback either using a single anti-message or multiple anti-messages.

3. Performance Evaluation

Java-based implementations for both connection-oriented and connectionless time warp were developed on a network of four Pentium-4 workstations interconnected by a 100Mbit/sec Ethernet switch. Performance was tested using a synthetic workload benchmark consisting of a fixed number of LPs and a constant number of circulating event messages. Each of the four machines runs four LPs connected as a 2x2 torus forming an overall 8x8 torus topology as shown in Figure 6. A shifted exponential service function with a mean of 50 ms is used in our experiments. Each logical process selects the output link on which it forwards the next outgoing message according to a uniform distribution. Periodic state saving is performed after the execution of each event with a fixed time cost of 2 ms. a static event scheduler on each LP is implemented using a smallest-timestamp-first scheduling policy. The Global Virtual Time (GVT) algorithm used is based on Mattern's GVT algorithm [4].

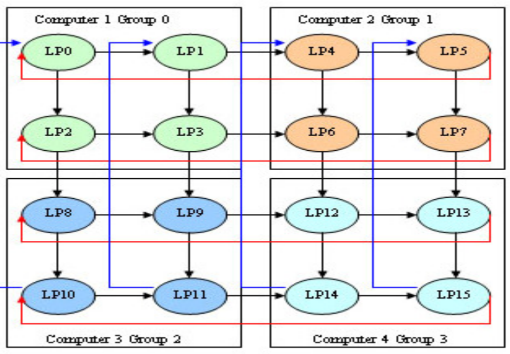

Figure 6. The 4x4 torus model topology.

We have selected the following performance measures for our experiments: the number of rollbacks, the number of anti-messages and the total simulation time. We conducted experiments to compare performance of the connection-oriented and connectionless time warp implementations as the initial message population per LP is varied from 5 to 20 messages per LP. Figure 7 shows the relationship between the number of rollbacks and the message population for the two implementations. It can be seen that, in the connection-oriented time warp implementation, the number of rollbacks is less than that in the case of the connectionless implementation. This can be attributed to the fast group cancellation of erroneous messages sent from one LP to another, which is a result of the more efficient implementation of event data sets in the case of the connection-oriented time warp implementation. This data set implementation greatly eliminates the possibility of occurrence of secondary rollbacks which is the main reason for the observed reduction in the number of rollbacks in the case of connection-oriented time warp. Moreover, we notice that the number of rollbacks in both implementations increases with the message population because, when we increase the initial number of messages in the input queue of each LP, the gap in virtual time between LPs decreases and, consequently, the overall number of rollbacks decreases.

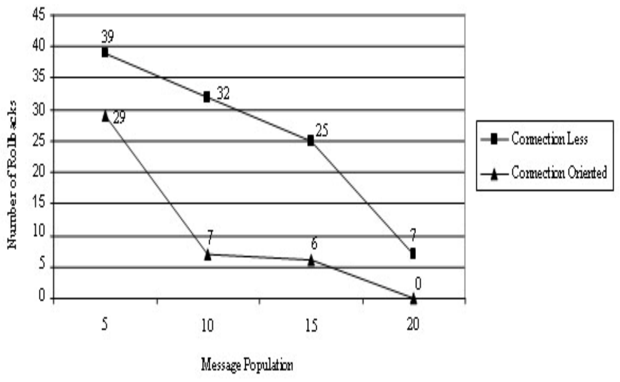

Figure 7. The number of rollbacks vs. message population for connection-oriented and connectionless time warp.

Figure 8 shows the relationship between the number of anti-messages and the message population for both time warp implementations. We note that there is a drop in the number of anti-messages in both cases because of the drop in the number of rollbacks with increasing message population previously observed in Figure 7. We also note that the number of anti-messages used by the connection-oriented implementation is considerably lower than that used by the connectionless implementation. This can also by explained by the group cancellation feature provided by the connection-oriented implementation.

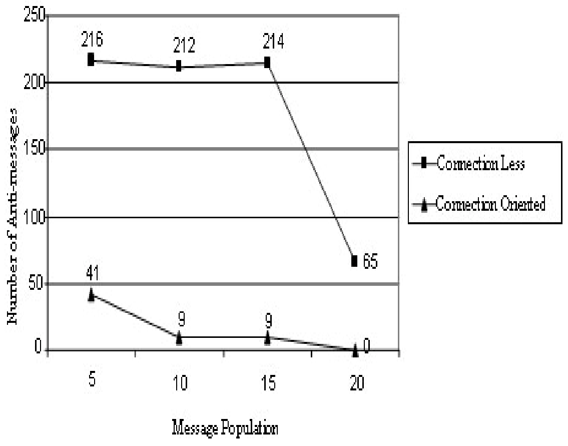

Figure 8. The number of anti-messages vs. message population for connection-oriented and connectionless Time Warp.

Finally, Figure 9 shows the relationship between the total simulation time and the initial message population for the two time warp implementations. It can be seen that the total simulation time for connection-oriented of rollbacks occurring in the connection-oriented case is less than the number of rollbacks in the connectionless case. This difference contributes time warp is less than that of connectionless time warp. This difference is logical given that the number significantly to reducing the number of anti-messages to be sent, which in turn results in reducing the total time, needed to complete the simulation.

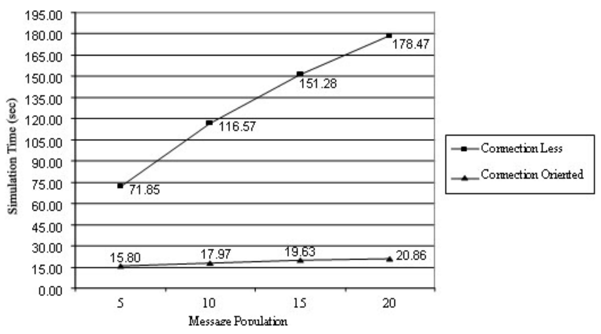

Figure 9. The total simulation time vs. message population for connection-oriented and connectionless time warp 4.

4. Conclusion

We have implemented an enhanced version of the connection-oriented time warp algorithm on a network of workstations using a standard benchmark simulation model. We have also conducted an experimental study to compare the performance of connection-oriented time warp with that of traditional connectionless Time Warp. Experimental results show that connection-oriented time warp algorithm is capable of achieving better performance than connectionless time warp for the used benchmark model for several performance measures.

Reference

[1] Fujmoto M., *Parallel and Distributed Simulation Systems*, John Wiley, New York, 2000.

[2] Jefferson D., Beckman B., Wieland F., and Blume L., "The Time Warp Operating System," *in Proceedings of the 11th Symposium on Operating System Principles*, pp. 77-93, United States, 1987.

[3] Kalantery N., "Time Warp Connection Oriented," *in Proceedings of the 18th Workshop on Parallel and Distributed Simulation*, pp. 71-77, United States, 2004.

[4] Mattern F., "Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation," *Computer Journal of Parallel and Distributed Computing*, vol. 8, no. 4, pp. 423-434, 1993.

[5] Perumalla S., "Parallel and Distributed Simulation: Traditional Techniques and Recent Advances," *in Proceedings of the 2006 Winter Simulation Conference*, pp. 84-95, California, 2006.

[6] Ronngren R. and Ayani R., "A Comparative Study of Parallel and Sequtial Priority Queue Algorithms," *Computer Journal ACM Transactions on Modeling and Computer Simulation*, vol. 7, no. 2, pp. 157-209, 1997.

Ali Al-Humaimidi received his BSc and MSc in information systems in 1998 and 2007, respectively from College of Computer and Information Sciences, King Saud University, Saudi Arabia.

Hussam Ramadan received his BSc in electrical engineering in 1988 from the George Washington University, USA, his MS degree in electrical engineering in 1991 from King Saud University, Saudi Arabia, and his PhD degree in computer science and engineering in 1995 from the University of Louisville, USA. Currently, he is the vice dean of the College of Computer and Information Sciences, King Saud University.