OFC: A Distributed Fossil-Collection Algorithm for Time-Warp*

Christopher H. Young, Nael B. Abu-Ghazaleh, and Philip A. Wilsey

Computer Architecture Design Laboratory
Dept. of ECECS, PO Box 210030, Cincinnati, OH 45221–0030
{cyoung,nabughaz,paw}@ececs.uc.edu

Abstract. In the Time-Warp synchronization model, the processes must occasionally interrupt execution in order to reclaim memory space used by state and event histories that are no longer needed (fossil-collection). Traditionally, fossil-collection techniques have required the processes to reach a consensus on the Global Virtual-Time (GVT) — the global progress time. Events with time-stamps less than GVT are guaranteed to have been processed correctly; their histories can be safely collected. This paper presents Optimistic Fossil-Collection (OFC), a new fossil-collection algorithm that is fully distributed. OFC uses a local decision function to estimate the fossilized portion of the histories (and optimistically collects them). Because a global property is estimated using local information only, an erroneous estimate is possible. Accordingly, OFC must also include a recovery mechanism to be feasible. An uncoordinated distributed checkpointing algorithm for Time-Warp that is domino-effect free and lightweight is used. We show that, in addition to eliminating the overhead for GVT estimation, OFC has several desireable memory-management properties.

1 Introduction

The Time-Warp paradigm is an optimistic distributed synchronization model that utilizes *Virtual Time* [6, 7]. In particular, Time-Warp is used to build optimistically synchronized parallel discrete-event simulators (PDES)[1]. Under this paradigm, the simulation model is partitioned across a collection of concurrent simulators called Logical Processes (LPs). Each LP maintains a Local Virtual Time (LVT) and communicates with other LPs by exchanging time-stamped event messages. Each LP operates independently; no explicit synchronization is enforced among the LPs. Instead, each LP enforces causal ordering on local events (by processing them in time-stamp order). Causality errors are tolerated temporarily across different LPs; a rollback mechanism is used upon detection of

* This work was partially supported by the Advanced Research Projects Agency and monitored by the Department of Justice under contract number J-FBI-93-116.

[1] While this paper focuses on Time-Warp in a PDES context, the techniques suggested herein generalize to any application of the Time-Warp model.

a causality error to restore a correct state. A causality error is detected when an arriving event message has a time-stamp less than LVT. Such events are called *straggler* events and their arrival forces the LP to halt processing and rolls back the LP to the earlier virtual time of the straggler. Thus, each LP must maintain state and event histories to enable recovery from straggler events.

The progress of the simulation is determined by the smallest time-stamp of an unprocessed event in the simulation (taking into account messages in transit). Since all the LPs have passed that time, messages with time-stamps earlier than that time cannot be generated (events may only schedule other events with a higher time-stamp). Hence, rollbacks to states before the global progress time are not possible. Accordingly, as progress is made by the simulation, some information in the state and event histories is no longer needed for rollback. These history elements are called *fossils* and the reclamation of the memory space occupied by these fossils is called *fossil collection*. Fossil collection is required to free up memory for new history items, and to enhance the memory locality properties of the simulation.

Traditionally, Time Warp simulators have implemented fossil collection by comparing history item time-stamps to the simulation's global progress time, called the *Global Virtual Time* (GVT). Histories with a time-stamp earlier than GVT can be safely fossil collected. The GVT estimation problem has been shown to be similar to the distributed snapshots problem [12] and requires the construction of a causally consistent state in a distributed system without a common global clock. In particular, GVT estimation algorithms require coordination among the distributed set of LPs to reach a consensus on GVT.

This paper presents a new model for fossil collection that does not require coordination among the LPs. This new model, called Optimistic Fossil-Collection (OFC), allows each of the LPs to estimate the fossilized portions of their history queues probabilistically. OFC eliminates the overheads associated with GVT calculation, and allows fossil-collection to be customized by each LP to best match its local behavior. However, as the decision functions use local information only, there is a nonzero probability of an occurance of a rollback to an erroneously fossil-collected state (an *OFC fault*). In order to overcome OFC faults, a recovery mechanism must be incorporated to restore the simulation to a consistent state. Thus, OFC consists of two primary mechanisms: (i) the local decision mechanism; and (ii) the recovery mechanism.

The remainder of this paper is organized as follows. Section 2 reviews Time-Warp memory management and GVT estimation algorithms. OFC is presented in more detail in Section 3. Section 4 examines the decision mechanism, while Section 5 discusses the facets of OFC's recovery mechanism. Section 6 presents our uncoordinated checkpointing algorithm for Time-Warp applications that is free from the domino-effect (traditionally, freedom from domino-effect is guaranteed only by coordinated checkpointing). In Section 7, OFC is emperically compared to traditional Time-Warp fossil collection techniques. Finally, Section 8 presents some concluding remarks.

2 Fossil Collection and GVT estimation

The Time-Warp synchronization paradigm implements causality by recovering from causal errors, rather than preventing them. When an LP discovers a causal violation (a message with a time-stamp *lower* than the LP's LVT is received), a rollback occurs. A rollback: (i) restores the latest state before the straggler message time from the state history queue, and (ii) cancels the messages that were sent out erroneously (due to optimistically executed events). Thus, rollbacks require that each LP store their state after every event is processed[2]. Since memory is bounded, Time Warp simulators must implement a fossil-collection mechanism to scavenge unneeded items from the state and event history queues.

While fossil-collection and GVT estimation are distinct operations, fossil-collection requires a GVT estimate in order to establish a marker, against which fossils can be identified. Fossil-collection occurs either as a "scavenge all fossils" operation [9] or a "scavenge one item" operation (*on-the-fly* fossil collection) [2]. In addition, GVT estimates can be maintained continuously [3, 12] or explicitly requested (usually when memory space is exhausted) [9]. Algorithms that continuously update GVT vary according to the level of aggressiveness in updating GVT. Less aggressive algorithms have a lower overhead but produce a relaxed estimate of GVT [12]. Aggressive algorithms maintain a close estimate of GVT but have a high overhead [3]. Despite the differences in obtaining the GVT estimate, fossil-collection algorithms have two common steps: (i) produce a GVT estimate; (ii) free up (all or a subset of) history items with a time-stamp lower than the estimate. A close estimate of GVT allows tight management of memory, increasing the memory locality properties and improving the range of optimism for models where memory space is constrained. However, tracking GVT aggressively increases the overhead of the GVT estimation algorithm and adversely affects performance.

3 Optimistic Fossil-Collection and Active History

Figure 1 presents a real-time snapshot of LPs in a Time-Warp simulation. Each axis represents a LP in the simulation; a solid square indicates the LP's current virtual time. Viewing the histories as a queue of entries ordered by simulation time-stamp, GVT[3] represents a time-line that separates fossils from *active histories*. Traditional Time-Warp identifies fossils by estimating GVT (using GVT estimation algorithms that calculate a virtual time that true GVT is guaranteed to have passed). Thus, the solid lines on the figure represent the history items that cannot be freed according to this model.

In practice, some items ahead of GVT may also never be needed for rollback purposes. The fact that such items exist is a by-product of the behavior that en-

[2] In order to minimize state saving overhead, states may be saved incrementally, or periodically — reducing the cost of state-saving, but requiring an overhead when a rollback occurs [5].

[3] Here GVT refers to the true simulation GVT; by the time GVT estimation algorithms reach a consensus on GVT, true GVT could have further advanced.

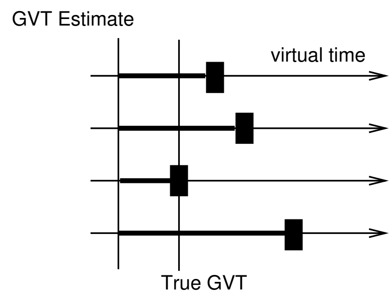

Fig. 1. LPs in a Time-Warp Simulation

ables optimistic synchronization in the first place; if none of the events executed optimistically are correct, then optimistic synchronization cannot succeed. Such events produce history items which will never be needed but are ahead of GVT. Unfortunately, these items cannot be identified at run-time without application information and global state information. Consider the extreme case where the LPs are completely independent — no rollbacks are necessary and the simulation completes correctly even if no state and event histories were maintained. GVT-based fossil-collection requires that each LP maintain history entries back to the current GVT estimate (even if their LVT is far ahead of GVT). Thus, GVT-based fossil-collection techniques maintain a conservative (larger than necessary) set of history information because: (i) the GVT-estimate lags the true GVT value; and (ii) true GVT is an absolute lower bound on fossilized entry times, but does not precisely mark the set of history items that are no longer needed.

Optimistic Fossil-Collection (OFC) is a fully distributed probabilistic fossil-collection technique for Time-Warp simulators. Under a basic OFC implementation, an LP: (i) implements a local decision function to estimate the portion of the history that is fossilized; and (ii) signals other LPs to recover if the LP detects an OFC fault (a rollback to a state that was optimistically collected). Thus, each LP makes its own fossil-collection decisions without the benefit of a GVT estimate. In addition to eliminating the overhead of GVT estimation, OFC allows each LP to customize its memory management to best suit its behavior. Because OFC is maintained optimistically, it can yield a tighter bound on memory than that produced by the GVT-based estimates. The decision function and recovery mechanism aspects of OFC are examined in more detail in the following two sections.

4 The Decision Function

The decision function is a critical component of the OFC technique. If the decision function is too aggressive in collecting fossils, OFC faults will occur frequently and the performance of the simulator drops. On the other hand, if it is too conservative, the memory consumed by history information will be large, causing inefficiencies due to loss of locality properties. It was shown [17], that OFC is safe and live for any decision function that increases its estimate after an OFC fault. Thus, there is greater freedom in implementing the decision function. In the remainder of this section, we identify two general classes of decision functions: one that predicts future rollback distances by statistical analysis of previous rollback behavior and one that converges heuristically to an active history estimate.

4.1 Statistically Bounding the Rollback Distance

This decision function requires that each LP sample event arrival times and create a statistical model of the rollback behavior. Note that, at any given point, the longest rollback distance from the current LVT defines the active history. The sampled rollback behavior is used to estimate a bound on the maximum rollback distance (the active history size). More precisely, an estimated bound X, can be obtained such that the probability of a future rollback of a distance larger than X (an OFC fault) is some pre-specified risk factor, α. Accordingly, history entries with a time-stamp smaller than $LVT - X$ can be optimistically reclaimed. If a smaller risk factor (α) is chosen, the predicted active-history size X, becomes larger, and OFC reclaims histories less aggressively.

The value of X can be expressed as a function of α if an underlying assumption on the rollback behavior is assumed. Consider the case where the length of a rollback is represented by a geometric distribution. Then the probability of a rollback of length l is given by $P(X = l) = p(1-p)^{l-1}$, where p is the probability of a rollback of length 1. A geometric distribution is a reasonable assumption since studies have shown that short rollbacks occur more frequently than long rollbacks [6]. The probability that a rollback of length X exceeds some distance l is given by $P(X > l) = (1-p)^{l+1}$. Thus, for a given p (obtained from sampling the rollback distances) and α, solve for l such that $P(X > l) < \alpha$ and $P(X > l-1) > \alpha$. This analysis can be adapted for other distributions too.

Note that it is not cost effective to verify that the sampled rollback distances conform to the presumed distribution at run time. The discrepancy between the distribution and the actual behavior of rollbacks adversely affects the risk factor. However, it is possible to use the Chebyshev inquality [14] to provide an upper bound on the X for a given α that is independent of the underlying distribution. The Chebyshev inquality provides an upper bound on the probability that a random variable will deviate a given distance from its mean μ. The bound is computed in terms of the variance σ^2 and holds regardless of the original distribution F (provided the variance and mean are finite). The Chebyshev inequality gives the probability of the LVT change exceeding l as

$$P\{|X-\mu| \geq l\} \leq \frac{\sigma^2}{l^2}.$$

The bound is a function of both the variance and the distance from the mean. Once an independent sample of the rollbacks has been gathered, a confidence interval for the mean can be determined via the central limit theorem [14]. Emperical results have shown that the statistical bounds are highly successful for the models that were studied [17].

4.2 Converging on the Active History Heuristically

The liveness proof for OFC required only that the decision function increase its estimate of the active history X when an OFC fault occurs [17]. Starting with an initial estimate of X, heuristic decision functions increase X to $f(X)$, where $f(X)$ is monotonic. An example of such a decision function is to simply double the current estimate of X when an OFC fault occurs. A conservative initial estimate of X will eventually converge on an upper limit for the active history size as it suffers OFC faults.

5 Recovering from OFC faults

Because the decision function estimates a non-local property using only local information, there is a nonzero probability that a history entry that was predicted to be fossilized (and was subsequently collected) will be needed in the future — an OFC fault occurs. When an OFC fault occurs, the simulation cannot overcome the causality error using rollback because the required history information is not available. In order for OFC to be feasible, a recovery mechanism that allows the simulation to return to a causally correct state must be implemented. With the additional level of recovery, OFC-based Time-Warp becomes a two level recovery system; it incorporates: (i) rollbacks to recover from erroneous optimism in computation, and (ii) a recovery mechanism to overcome erroneous optimism in fossil-collection.

The simplest possible recovery mechanism is to restart the simulation from the initial state; all the work invested in the simulation up to the OFC fault has to be discarded. Moreover, in this case guarantees on liveliness can only be provided if the memory is arbitrarily large [17]. In general, recovery can be implemented as a restart from a *globally consistent simulation state*. A globally consistent state is a state that may occur in a legal execution of the simulation [1, 11]. Note that the state of a distributed application consists not only of the local states of the processors, but also of the state of the communication channels between the processors. Thus, as the simulation is progressing, globally consistent checkpoints must be constructed infrequently. A full definition of global snapshot consistency is given by Chandy and Lamport [1].

Constructing globally consistent checkpoints is a well researched problem [4]. Uncoordinated checkpointing does not guarantee freedom from the domino-effect [13]; a globally consistent checkpoint cannot be guaranteed (other than the initial state). It is possible to prevent the domino-effect by performing *coordinated* checkpointing [4]; the processes save their local state, and coordinate in saving the state of the communication channels between them to insure that the local snapshots together form a "meaningful" global snapshot [1, 8, 12, 15].

Using coordinated checkpointing to protect from OFC faults presents the following dilemma. GVT algorithms have been shown to be an instance of coordinated distributed snapshot algorithms — What is the point of OFC if it requires coordinated checkpointing to eliminate the equivelant of coordinated checkpointing (GVT estimation)? The answer is: (i) if the decision function is accurate, checkpointing can be carried out infrequenty (since it is required if an OFC fault occurs but not under normal operation). In contrast GVT estimates are on the forward path of operation for traditional Time-Warp simulators and must be invoked frequently if a reasonably memory bound is desired; and (ii) in the next section, we present an uncoordinated distributed checkpointing algorithm for virtual time applications that is domino-effect free and lightweight.

6 Uncoordinated Distributed Checkpointing for Time-Warp

In this section, we present an algorithm for uncoordinated distributed checkpoints using virtual time (the viability of this approach was recognized previously [11]). There are two main advantages for this algorithm: (i) it produces domino-effect free consistent checkpoints without coordination; and (ii) the size of the checkpoint is small; instead of checkpointing all of the input, output, and state queues (as required by a consistent checkpoint in real time), only a minimal subset of the entries is required to be checkpointed for this algorithm. The algorithm is lazy, because it does not enforce consistency of the checkpoints as they are created; instead, the last consistent checkpoint is detected when a failure occurs. The algorithm is described in two parts: (i) *checkpointing*, and (ii) *recovery*.

Checkpointing: As each LP advances its simulation without an OFC fault, it checkpoints itself at pre-negotiated virtual times (similar to *Target Virtual Time* [16]). This feature is the key to the algorithms superiority to coordinated checkpointing algorithms; instead of coordinating *in real time* at a considerable overhead to realize a consistent state, static coordination in virtual time is established. The checkpointing steps are:

- LPs negotiate a simulation checkpoint interval, τ; this step could be carried out off-line, or infrequently.
- A checkpoint is taken independently by each LP as it reaches the preapproved checkpoint simulation time t; the checkpoint is taken at the last state *before* simulation time t. Note that if an LP rolls back past t, it will be checkpointed again when its simulation time reaches t again[4].
- At a checkpoint, each LP saves the state before t and all messages with a *sendtime* $< t$ and a *receivetime* $\geq t$.

[4] To minimize repeated checkpointing because of roll-backs, the checkpoint is taken when the checkpoint state is about to be fossil-collected.

Note that no communication occurs other than in the initial set-up step (the algorithm is uncoordinated), and that only one state entry and a subset of the messages in the output queue are saved at each checkpoint (the algorithm is lightweight).

Recovery: In the event of an OFC fault, the recovery mechanism is invoked. Because checkpointing is carried out without coordination, it is first necessary to determine which checkpoint should be used for recovery. The recovery algorithm composes a consistent checkpoint from the individual checkpoints by using the latest checkpoint that has been reached by all LPs (accounting for messages in transit)[5]. Determining the correct recovery checkpoint requires a consensus among the LPs to be reached, in a manner similar to GVT algorithms. The simulation is restarted by having each process discard state and event histories, reload the state information from the checkpoint, and resend the saved messages. Note that the messages in the network must be either drained before restart, or detected and discarded. For space considerations, we do not present proofs that the algorithm produces and detects consistent cuts [17]. Informally, the proof uses the following observations: (i) a set of states occuring at the same virtual time is legal if GVT has passed it (which is ensured by detecting the checkpoint that all LPs have passed) — it is the distributed equivelant of the state produced by sequential simulation; and (ii) the message behavior is preserved by regenerating only the messages that cross the restored snapshot (generated from its past and destined to its future).

Collecting the Checkpoints : Although the checkpoints are stored infrequently, there has to be a method to collect them to free up resources. There are several possibilities for collecting checkpoints, including: (i) invoke a lightweight GVT calculation when a certain number of checkpoints accumulates and free up the ones earlier than GVT; (ii) when an OFC fault occurs, a consistent checkpoint is detected and restored. At every LP, the checkpoints earlier than the one used can be collected (since a more recent checkpoint exists); (iii) a two level checkpointing scheme where checkpoints are maintained in memory. Occasionally, older checkpoints are flushed to disk (freeing up memory space). Although disk space is not infinite, this reduces the frequency at which the checkpoint space needs to be reclaimed.

7 Analysis

In this section, we compare the performance and memory properties of OFC to traditional fossil-collection techniques such as pGVT [3] and Mattern's [12] GVT algorithm. The pGVT algorithm maintains a tight bound on GVT by having the LPs report increments in their LVT to a statically elected LP (which then assumes the role of a GVT manager) periodically. In contrast, Mattern's algorithm

[5] Some LPs may have surged ahead with their computation and taken checkpoints that other LPs have not yet reached.

is a lightweight algorithm based on the Chandy-Lamport distributed termination detection algorithm [1]. Mattern's algorithm is less aggressive than pGVT; it has a lower overhead but produces a less accurate estimate of GVT. The three algorithms were implemented in the WARPED Time-Warp simulator [10]. The performance is reported for two simulation models (a RAID disk array model and the P-Hold benchmark) simulating across a network of workstations (4 processors were used for each simulation).

Algorithm	Model	Time (s)	Memory (bytes)	
			Avg.	Max
pGVT	RAID	1110.17	2133403	5400631
Mattern	RAID	1023.19	2180713	6540746
OFC	RAID	955.40	1864138	2013004
pGVT	PHOLD	593.26	1666736	5245655
Mattern	PHOLD	534.14	4118584	15731407
OFC	PHOLD	466.92	1558613	1616748

Table 1. Performance Characteristics of OFC, pGVT, and Mattern's Algorithm

Three simulator configurations were created: two corresponding to the two GVT-based garbage collectors, and one corresponding to an OFC-based garbage collector using the Chebyshev decision model with a low risk factor ($\alpha = 0.999$). For this risk factor, we expect the number of OFC faults to be small, and the memory bound to be conservative. Table 1 shows the performance results obtained for the two models. For both sets of experiments, OFC achieved better execution time and a tighter memory bound (average as well as maximum) than either pGVT or Mattern's algorithms. Recall that Mattern's algorithm is less aggressive than pGVT and, therefore, produces a less accurate memory bound but with a better execution time.

8 Conclusion

In Time-Warp simulations, each logical process (LP) maintains history queues in order to allow recovery from overly optimistic computation. As the global progress time of the simulation advances, older histories are no longer needed (fossilized), and the memory space they occupy must be reclaimed to allow newer histories to be stored. Traditional implementations of Time-Warp estimate the Global Virtual Time (GVT) in order to identify fossils. Estimating GVT involves a significant communication overhead that varies with the GVT estimation algorithm. In addition, because a global overhead is incurred whenever an individual LP requires fossil-collection, the technique is vulnerable to inefficiency from LPs that are memory constrained.

This paper presented Optimistic Fossil-Collection (OFC): a probabilistic distributed model for garbage collection on Time-Warp simulators. OFC eliminates the need for GVT estimates and allows the LPs to tailor their memory usage to their local rollback behavior. Each LP decides which history items are likely to be fossilized using a local decision function and proceeds to collect these items optimistically. A perfect decision function would converge on the *active history* for the LP. The active history is the minimum subset of history items that is sufficient to recover from any rollback that occurs in the future. Note that this limit is bounded by true GVT — a perfect decision mechanism will yield a memory bound superior to the true GVT bound (in fact, it is a minimal memory bound). Even though a perfect decision mechanism is impossible to construct at run-time, there is a potential for OFC to produce memory bounds superior to GVT-based algorithms, especially since the GVT estimate used by these algorithms lags true GVT.

Since history items are collected without global knowledge of the simulation state, there is a nonzero probability that an history item that was collected will be needed in the future — an OFC fault occurs. A recovery mechanism must be implemented to enable recovery from OFC faults. Thus, OFC consists of two, largely decoupled, facets: the decision mechanism and the recovery mechanism. The paper investigated models for both the decision and recovery aspects of OFC. We classified decision functions into statistical and heuristic based functions. The statistical decision functions presume a rollback distribution function (created by sampling local rollbacks) and predict an upper bound on the rollback distance with specified acceptable risk factor (confidence). Because the rollback behavior may not correspond to the presumed distribution, we also presented a limit based on the Chebyshev inequality — this limit is independent of the underlying rollback distribution. The heuristic functions adaptively converge to an active history estimate. Both of these approaches provide good bounds on the maximum rollback distance [17].

For the recovery aspect of OFC, a method for creating on-the-fly consistent checkpoints is required. We presented an algorithm for uncoordinated checkpointing of applications using virtual time. With OFC, Time-Warp simulators become a two-level recovery algorithm: (i) rollback to recover from erroneous optimism in computation, and (ii) the OFC recovery mechanism to overcome erroneous optimism in fossil-collection. We conducted an empirical comparison of OFC with two traditional GVT algorithms: pGVT [3] and Mattern's algorithm [12]. OFC executed faster than pGVT, while producing a tighter memory bound. Conversely, OFC produced a tighter memory bound than that achieved by Mattern's algorithm for the studied models. However, Mattern's algorithm execution time was better than OFC's on SMMP, but worse on RAID. We anticipate that refinement to our decision and recovery algorithms will further enhance the performance of OFC.

References

[1] Chandy, K. M., and Lamport, L. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems 3*, 1 (Feb. 1985), 63–75.

[2] Das, S., Fujimoto, R., Panesar, K., Allison, D., and Hybinette, M. GTW: a time warp system for shared memory multiprocessors. In *Proceedings of the 1994 Winter Simulation Conference* (Dec. 1994), J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, Eds., pp. 1332–1339.

[3] D'Souza, L. M., Fan, X., and Wilsey, P. A. pGVT: An algorithm for accurate GVT estimation. In *Proc. of the 8th Workshop on Parallel and Distributed Simulation (PADS 94)* (July 1994), Society for Computer Simulation, pp. 102–109.

[4] Elnozahy, E., Johnson, D., and Wang, Y. A survey of rollback-recovery protocols in message-passing systems. Tech. Rep. Tech. Rept. CMU-CS-96-181, School of Computer Science, Carnegie Mellon University, Oct. 1996.

[5] Fleischmann, J., and Wilsey, P. A. Comparative analysis of periodic state saving techniques in time warp simulators. In *Proc. of the 9th Workshop on Parallel and Distributed Simulation (PADS 95)* (June 1995), pp. 50–58.

[6] Fujimoto, R. Parallel discrete event simulation. *Communications of the ACM 33*, 10 (Oct. 1990), 30–53.

[7] Jefferson, D. Virtual time. *ACM Transactions on Programming Languages and Systems 7*, 3 (July 1985), 405–425.

[8] Lai, T., and Yang, J. On distributed snapshots. *Information Processing Letters 25* (May 1987), 153–158.

[9] Lin, Y.-B. Memory management algorithms for optimistic parallel simulation. In *6th Workshop on Parallel and Distributed Simulation* (Jan. 1992), Society for Computer Simulation, pp. 43–52.

[10] Martin, D. E., McBrayer, T. J., and Wilsey, P. A. warped: A time warp simulation kernel for analysis and application development. In *29th Hawaii International Conference on System Sciences (HICSS-29)* (Jan. 1996), H. El-Rewini and B. D. Shriver, Eds., vol. Volume I, pp. 383–386.

[11] Mattern, F. Virtual time and global states in distributed systems. In *Proc. Workshop on Parallel and Distributed Algorithms* (Oct. 1989), M. Cosnard et al, Ed., pp. 215–226.

[12] Mattern, F. Efficient algorithms for distributed snapshots and global virtual time approximation. *Journal of Parallel and Distributed Computing 18*, 4 (Aug. 1993), 423–434.

[13] Randell, B. System structure for software fault tolerance. *IEEE Trans. on Software Engineering SE-1*, 2 (June 1975), 220–232.

[14] Ross, S. M. *Introduction to Probability Models*, 4 ed. Academic Press, San Diego, CA, 1989.

[15] Spezialetti, M., and Kearns, P. Efficient distributed snapshots. In *Proc. IEEE International Conference on Distributed Computing Systems* (1986), pp. 382–388.

[16] Tomlinson, A. I., and Garg, V. K. An algorithm for minimally latent global virtual time. In *Proc of the 7th Workshop on Parallel and Distributed Simulation (PADS)* (July 1993), Society for Computer Simulation, pp. 35–42.

[17] Young, C. *Methods for Optimistic Reclamation of Fossils in Time Warp Simulation.* PhD thesis, University of Cincinnati, June 1997. (Ph.D. proposal).