THE IMPACT OF ADDING AGGRESSIVENESS TO A NON-AGGRESSIVE WINDOWING PROTOCOL

Phillip M. Dickens

Institute For Computer Applications
In Science and Engineering
NASA Langley Research Center
Hampton, Va. 23681, U.S.A

David M. Nicol

Department of Computer Science
The College of William and Mary
Williamsburg, Va 23187, U.S.A

Paul F. Reynolds, Jr.

Department of Computer Science
University of Virginia
Charlottesville, Va. 22903, U.S.A

John Mark Duva

Department of Applied Mathematics
University of Virginia
Charlottesville, Va. 22903, U.S.A

Abstract

In this paper we summarize the results of our theoretical investigation into the costs and benefits of extending the conservative simulation window established in a non-aggressive windowing algorithm. There are two primary costs incurred by the non-aggressive algorithm: the cost of global synchronization and the cost of blocking due to pessimistic synchronization constraints. As the conservative simulation window is extended processors are required to synchronize less often and parallelism is increased. However, the increased aggressiveness increases the costs associated with state saving and rollbacks. This is the fundamental trade-off we capture analytically.

1 Introduction

Most of the synchronization protocols developed for parallel discrete event simulation fall into two basic categories. Protocols that are non-aggressive (e.g. Chandy and Misra 1979, Chandy and Sherman 1989, Lubachevsky 1988, and Nicol 1993) do not allow a logical process (LP) to process an event with timestamp t if it is *possible* that it will receive another event with a timestamp less than t at some point in the future (when an LP processes an event with a timestamp less than an event already processed it is termed a *causality error*). Aggressive protocols (e.g. Time Warp, Jefferson 1985) allow an LP to process events in any timestamp order, and causality errors are corrected through a *rollback* mechanism.

Both approaches have inherent problems. Non-aggressive protocols tend to leave processors idle due to overly pessimistic synchronization constraints. Aggressive protocols are criticized for the high overhead costs associated with state saving and rollback, and because of the possibility of cascading rollbacks.

Recently, there has been much interest in the parallel simulation community in synchronization protocols which blend aspects of aggressive and non-aggressive techniques. The goal of this research is to maximize the advantages, and minimize the disadvantages, of each approach. Some researchers are investigating limiting the aggressiveness of Time Warp in order to reduce the costs of state saving and to eliminate cascading rollbacks (e.g. Madisetti *et al.* 1992, Turner and Xu, 1992 and Lubachevsky *et al.*, 1989). Other researchers are investigating adding aggressiveness to non-aggressive protocols in order to increase concurrency (e.g. Steinman 1992, and Dickens and Reynolds, 1990). To date, there have been few analytical studies which examine the costs and benefits of decreasing the aggressiveness of Time Warp or increasing the aggressiveness of non-aggressive protocols.

In this paper we summarize the results of our investigation into the adding of aggressive processing to a class of non-aggressive protocols termed *windowing protocols* (e.g. Nicol 1993, Chandy and Sherman 1989, Lubachevsky 1988). We chose a windowing algorithm because they are the only protocols for which important scalability results have been proven (Nicol 1993, Lubachevsky *et al.* 1989, Lubachevsky 1989a). We develop a model to show that adding limited aggressive processing offers the potential for significant improvement in performance while maintaining the important scalability features of the non-aggressive protocol.

We are not the first researchers to look at adding aggressiveness to a non-aggressive windowing protocol. Lubachevsky *et al.* (1989) did this in relation to the Bounded Lag Algorithm, and also showed that the aggressive version of the algorithm is scalable. The distinguishing feature of our work is in the analytic model. Our model provides a detailed study of the costs and benefits of adding aggressive processing to the non-aggressive protocol. Lubachevsky did not address this issue. The difference in the scalability results is that our model predicts the cost of aggressive processing *as a function of the number of LPs in the system.* Lubachevsky did not address the costs of his algorithm as a function of the number of LPs.

The protocol we propose to correct causality errors is essentially Time Warp with limited aggressiveness. As noted above, there are other researchers who have proposed and studied limiting the aggressiveness of Time Warp. Again the distinguishing feature of this work is in the development of the analytic model. We study the costs of aggressive processing *as a function of the level of aggressiveness.* This is unique among all of the studies of aggressive processing.

In this paper we present an overview of our modeling technique and give a summary of our results. The details of the model can be found in Dickens *et al.* (1993).

2 The Aggressive Windowing Protocol

In a windowing algorithm all concurrent simulation activity is constrained to be within some window of global simulation time. This simulation window, which we refer to as the *lookahead* window, is defined such that all events with timestamps falling within the window can be executed concurrently without any possibility of a causality error. The interested reader is directed to Nicol (1993) for the construction of one such lookahead window. Processing events with timestamps outside of the window may result in a causality error and is therefore not allowed. Windowing protocols generally proceed in three phases where the lookahead window is computed in the first phase, all events within the lookahead window are processed concurrently in the second phase, and all messages generated as a result of this processing are exchanged in the third phase. Each phase is separated by a barrier synchronization. There are two primary costs incurred by the non-aggressive windowing algorithm: the cost of global synchronization and the cost of blocking due to pessimistic synchronization constraints.

We seek to minimize these costs by extending the conservative simulation window defined by the algorithm. We term this extension to the lookahead window the *aggressive window* and allow an LP to process events within both windows. Assume the system is synchronized at logical time T where T is the current window floor. The non-aggressive windowing algorithm defines the lookahead window from logical time T to logical time $T + L$, where L is the width of the lookahead window. Our modified algorithm defines an aggressive simulation window from the upper bound of the lookahead window (logical time $T + L$) to logical time $T + L + A$.

Our modified windowing algorithm proceeds in two phases as follows. In the first phase the LPs cooperatively determine the lookahead and the aggressive windows. We discuss the best choice for the size of the aggressive window (based on the results of our model) in later sections. In the second phase of the modified algorithm all events with timestamps falling within the extended simulation window are processed. As can be seen, the amount of aggressiveness exhibited by the system is controlled by the size of the aggressive window.

We propose to use a simple state saving and rollback mechanism such as Time Warp to correct any causality errors that occur as a result of aggressive processing. We assume state is saved before the processing of any event within the aggressive window. If an LP receives an event with a timestamp t, and it has processed an event with a timestamp greater than t, it must perform a rollback. When this occurs the LP must restore the state that was saved immediately before logical time t, and all events with timestamps greater than t must be reprocessed. If the LP has sent any messages based on the processing of an event with timestamp greater than t, we assume the message was sent in error. In this case an *anti-message* is sent to cancel the message. An anti-message has the same content as the original message, and is sent to inform the receiving LP that the original message was sent in error. This corresponds to the *aggressive cancellation policy* (Reiher *et al.* 1990) in Time Warp.

3 Approach to Modeling

In both the aggressive and non-aggressive versions of the algorithm all of the LPs must wait until the *slowest* LP in the system completes it processing within the simulation window. In this section we describe our model to predict the performance of the slowest LP in the system for both approaches. We begin with a brief discussion of our model. Note that for the rest of this paper we use the terms *event* and *message* interchangeably.

Our model is closely related to the model developed by Nicol (1993), although he has not studied processing outside of the lookahead window. Our model is also closely related to the models developed by Akyilidiz *et al.* (1992) and Gupta *et al.* (1991), and uses the same assumptions. Akylidiz and Gupta however are investigating the behavior of Time Warp which does not limit aggressive processing as we are proposing.

We model our protocol as a collection of servers where *activities* occur. An activity (e.g. service given to a job at a queue) begins, ends, and upon its completion causes other activities. In our model each completion causes exactly one other activity somewhere in the system. We assume a completion causes an activity at a server that is picked at random, where each server is equally likely to be picked.

The delay in simulation time between when an activity begins and ends is called the *duration* of the activity. We assume each server chooses the duration of an activity from an independent, identically distributed exponential distribution with mean $1/\lambda$. We assume there are N servers, and one server per LP. Our model assumes a closed queueing system where the system is heavily loaded and the probability of a server being idle is very close to zero.

We define the *processing cost* of the given approach as the expected cost of processing through one unit of logical time. This cost includes the events that must be processed within the simulation window (i.e. the real work of the simulation), as well as the over-head costs associated with the particular approach. We model the over-head costs relative to the cost of processing a single event, and we assume each event takes approximately the same amount of real time to compute.

Given this brief introduction we describe the computation of the cost function for each approach. We derive the probability distribution for each random variable defined in the model in Dickens *et al.* (1993).

3.1 Costs of Non-Aggressive Processing

In the non-aggressive windowing algorithm the LPs concurrently process all of their events within the lookahead window. When an LP completes this processing it enters into a barrier synchronization waiting for the other LPs to similarly complete. Given our assumption that each event takes approximately the same amount of real time to compute, processing within the lookahead window will be dominated by the LP with the most messages to process.

After processing within the lookahead window the LPs enter into a global synchronization. We assume the system uses a traditional barrier synchronization such as the **gsync()** routine provided on an Intel iPSC2 hypercube. The cost of a barrier synchronization is $O(Log_2 P)$ given a system with P processors.

We define M_{LA} as the random variable representing the maximum number of events within the lookahead window taken over all of the LPs in the system. Let C_{LA} represent the cost (to the maximally loaded LP) of processing through one lookahead window. Then

$$C_{LA} = M_{LA} + c\, Log_2\, P. \tag{1}$$

The c term in Equation (1) is a factor used to express the cost of global synchronization relative to the cost of processing a single event. To compute the cost of processing through one unit of logical time we divide C_{LA} by L, the width of the lookahead window. We define $Cost_{NA}$ as the cost of processing through one unit of logical time for the non-aggressive approach.

$$Cost_{NA} = \frac{M_{LA} + c\, Log_2\, P}{L} \tag{2}$$

3.2 Costs of Aggressive Processing

Before developing our cost model for the aggressive approach it is necessary to define some terminology.

When an LP receives a message from another LP we term this message an *arrival message* (i.e. it arrives at the receiving LP). Due to the construction of the lookahead window it is guaranteed that no LP will receive an arrival message with a timestamp that falls within the aggressive window. However, an LP may receive an arrival message with a timestamp that falls within the aggressive window (or where the timestamp is greater than the upper bound of the aggressive window). Unless stated otherwise, whenever we refer to an arrival message we are referring to an arrival message with a timestamp within the aggressive window.

When an arrival message causes an anti-message to be produced it is termed a *first generation* anti-message. We define an Nth generation anti-message as one that is produced because of the receipt of an $Nth - 1$ generation anti-message. We define a *rollback chain* as a chain of anti-messages. The *depth* of the rollback chain is the maximum generation anti-message produced in the chain. We define the *maximum rollback chain* as the rollback chain which produces the highest generation anti-message observed in the system for a given iteration of the protocol.

Now that we have developed our terminology we continue with the development of the costs of aggressive processing.

3.3 Workload of the Dominant LP

We define a special LP which is used to track the occurrences in the system which dominate system performance. In order to ensure we track the processing costs of the *slowest* LP in the system, we define the workload of this special LP such that it is greater (or it is reasonably expected to be greater) than any LP in the system. We term this specially defined LP the *dominant LP*. We make a set of very pessimistic assumptions regarding the workload of the dominant LP, and we use this workload in our calculation of the cost of the aggressive approach.

First, we assume the dominant LP has the maximum number of events in the lookahead window taken over all LPs in the system. Recall we defined M_{LA} to represent this random variable. Similarly, we assume the dominant LP has the maximum number of events within the aggressive window taken over all LPs in the system. We define M_{Ag} as the random variable representing this cost.

Next, we assume the dominant LP receives the maximum number of arrival messages taken over all of the LPs in the system. This assumption places a very heavy workload on the dominant LP since each arrival message can invalidate (and thus force the reprocessing of) the events within the aggressive window as well as previously received arrival messages. Additionally, each time an event is reprocessed the state of the LP must be saved. We define M_{Ar} as the random variable representing the number of arrival messages received by the dominant LP, and we define Rep_{Ar} as the random variable representing the number of events that must be reprocessed due to the arrival messages.

Next, we consider the reprocessing caused by the receipt of anti-messages. It is important to note that the processing costs to a *given* LP that receives one of the anti-messages in a rollback chain may not be large, but the cost to the *system* is cumulative. That is, the entire system must remain blocked until *every* anti-message in a rollback chain is processed. In order to account for the cumulative effects of the maximum rollback chain we make the pessimistic assumption that the dominant LP receives an anti-message for *each generation anti-message* in the maximum rollback chain. This assumption also places a heavy workload on the dominant LP since the amount of reprocessing caused by an anti-message is a function of the number of messages processed aggressively. We define Rep_{An} as the random variable representing the amount of reprocessing caused by anti-messages in our system.

We define TM_{Ag} as the total number of messages processed by the dominant LP, including those messages reprocessed due to the receipt of arrival messages and anti-messages.

$$\begin{aligned} TM_{Ag} &= M_{LA} + M_{Ar} + M_{Ag} + \\ &\quad Rep_{Ar} + Rep_{An} \end{aligned} \tag{3}$$

Next, we consider the cost of saving state. Given our assumption that state is saved before every message that is processed aggressively, the dominant LP must save its state before the processing of all messages *except* those processed within the lookahead window. Let SS_T represent the total number of times the dominant LP saves its state. Then

$$SS_T = TM_{Ag} - M_{LA}. \tag{4}$$

Finally, we must consider the cost of global synchronization at the upper bound of the aggressive window. We note that due to unpredictable message exchange patterns we cannot use a traditional barrier synchronization, and our model therefore assumes an aggressive barrier synchronization such as that described by Nicol (1993a). As noted by Nicol, the cost of an aggressive barrier synchronization is on the order of two to three times that of a traditional barrier synchronization. Either Nicol (1993a), or Dickens *et al.* (1993), may be consulted for a thorough discussion of the issues associated with an aggressive barrier synchronization.

The cost of global synchronization for the aggressive algorithm is $2\,c\,Log_2\,P$. We define SS_c as the cost of saving state relative to the cost of processing a single message. We define C_{Ag} as the cost of processing through one simulation window for the dominant LP. Then

$$C_Ag = TM_{Ag} + SS_c\; SS_T + 2cLog_2 P. \tag{5}$$

We define $Cost_{Ag}$ as the cost of processing through one unit of logical time for the aggressive algorithm and give its value below.

$$Cost_Ag = \frac{C_{Ag}}{L + A} \tag{6}$$

4 Theoretical Results

In this section we compare the processing costs of the two approaches. All of our results assume a system with $N = 1024$ LPs and a mean service time of $1/\lambda = 1$. Given these parameters the expected width of the lookahead window is $E[L] = .0385$. In order to be consistent with our analytical model we assume there is one LP per processor, and thus we assume $P = 1024$. We define the expected improvement in processing costs as

$$EI = \frac{Cost_{NA}}{Cost_{Agg}} \tag{7}$$

Thus ratios greater than one indicate the superiority of the aggressive windowing algorithm.

We derive results for aggressive window sizes of $A =$ 100%, $A = 50\%$ and $A = 10\%$ of the mean of the service time distribution. We discuss this choice for A below. There are two independent variables which must be considered: the cost of global synchronization relative to the cost of processing a single message, and the cost of saving state relative to the cost of processing a single message. We present a family of curves where the state saving cost is varied on the x axis, and the expected improvement in processing costs is given on the y axis. State saving costs vary between zero and twice the cost of processing a single message. Each curve represents the improvement in processing cost given a cost of global synchronization that is 0, .25, .5, .75, 1.0 or 2.0 times the cost of processing a single message. We show the line where the expected processing costs for the two approaches are identical.

Before presenting the results of our analysis we briefly discuss the size of the aggressive window. The purpose of this analysis is to investigate the impact of *limited* aggressive processing on the performance of the non-aggressive windowing protocol. Since we are only considering limited aggressive processing we are able to make some approximations which significantly simplify our analysis (these approximations are discussed in detail in Dickens *et al.* 1993). These approximations begin to break down however if the size of the aggressive window becomes too large. Our approximations are very reasonable for aggressive window sizes up to the mean of the service time distribution, and for this reason we only consider aggressive window sizes between zero and 100% of the mean of the service time distributions. It is important to note that the impact of our approximations is to *underestimate* the improvement in performance that can be obtained due to the addition of limited aggressive processing.

In Figure 1 we plot the expected improvement given an aggressive window size of A = 100% of the mean service time. As can be seen, our model predicts significant improvement in performance for a very wide range of state saving and global synchronization costs. We find these results are very encouraging, particularly given our very pessimistic assumptions regarding the cost of aggressive processing.

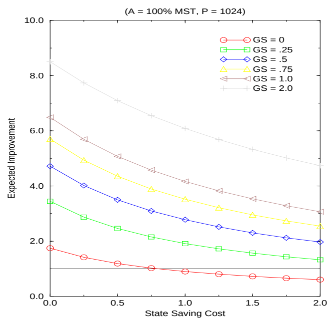

Figure 1: Theoretical Improvement, A = 100% MST

In Figure 2 we plot the expected improvement in performance given an aggressive window size that is 50% of the mean service time. Although the model still predicts significant improvements in performance, the predicted improvement is less than that for the larger aggressive window. This is because our model predicts, and simulation studies support, that if the costs of processing through an aggressive window of size $A1$ is Y, then the cost of processing through an aggressive window of size $A1/2$ is greater than $Y/2$. Thus the total cost of processing through two aggressive windows of length $A = 50\%$ of the mean service time is greater than the cost of processing through one aggressive window of length $A = 100\%$ of the mean service time.

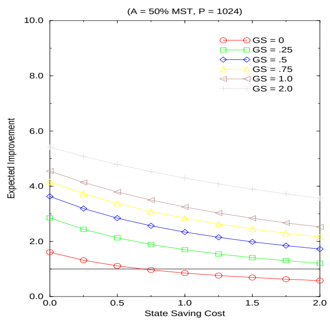

Figure 2: Theoretical Improvement, A = 50% MST

In Figure 3 we plot the expected improvement in performance given an aggressive window size of $A = 10\%$ of the mean service time. As can be seen, our model predicts only limited improvement in processing costs for this very small aggressive window. The reasons for this are the same as those discussed above.

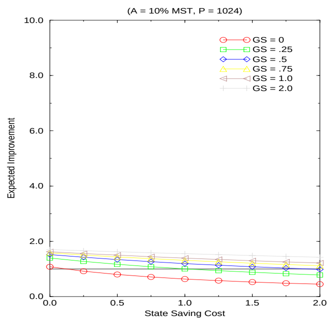

Figure 3: Theoretical Improvement, A = 10% MST

5 Simulation Studies

In this section we present a set of simulation studies in order to test the predictions of our model. All of the results (except where noted) assume $P = 1024$ processors, $N = 1024$ LPs and a mean service time of $1/\lambda = 1$. The system we simulated is a simple FCFS queueing network with one server per LP. Each data point represents the mean observed value taken over sixteen trials, where each trial consisted of one thousand iterations of the algorithm. Note there was very little observed difference between the trials.

In Figure 4 we show the number of messages processed by the LP which processed the maximum number of messages taken over all LPs in the system. This message count does not account for state saving or global synchronization costs, and thus is the number of messages captured by TM_{Ag} in Equation (3). As can be seen, our model does indeed give a very pessimistic view of the costs of aggressive processing.

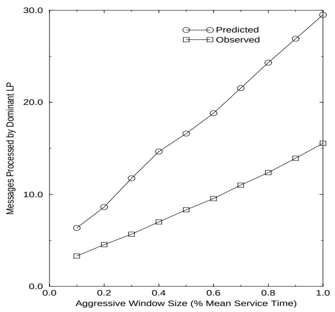

Figure 4: Number of Messages Processed by the Dominant LP

In Figure 5 we use these empirical results to compute the improvement in performance for an aggressive window size given $A = 100\%$ of the mean service time. In this graph we include the cost for state saving and global synchronization. As expected, the observed improvement in performance is better than that predicted by our model.

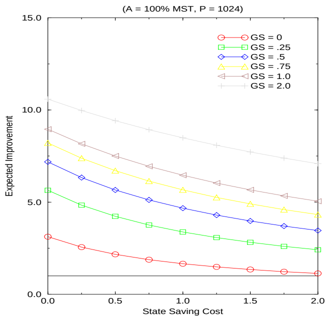

Figure 5: Observed Improvement, A = 100% MST

6 Scalability

We predict the cost of aggressive processing as the size of the problem and the size of the architecture are simultaneously increased. In order to compensate for the increasing cost of global synchronization as the number of processors is increased (i.e. keep the work load per processor constant), we assume there are $J = P\,Log_2 P$ LPs per proceeeor, and $N = J\,P$ LPs. We increase N from 512 to 65,536. The results are shown in Figure 6 (these results include the costs captured by TM_{Ag} in Equation (3)).

As can be seen, our model predicts that as we increase the number of LPs by a factor of 128, the number of messages processed by the dominant LP increases by approximately a factor of two. These are very encouraging results. In order to test these results we varied the number of LPs in our simulation from 512 to 4096 (the maximum number of LPs that can be handled in our simulation due to memory constraints). As can be seen, simulation results confirm the trend predicted by the model.

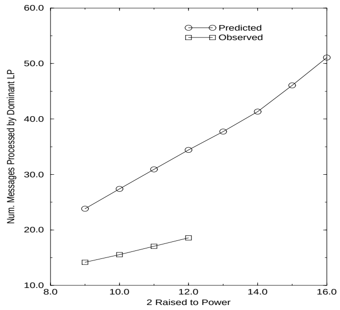

Figure 6: Scalability Results

7 Conclusions

In this paper we have presented an over-view of our approach to modeling the impact of adding aggressive processing to a non-aggressive windowing protocol. Our model predicts, and simulation studies support, that aggressive processing can, for a wide range of state saving and global synchronization costs, offer the potential for significant improvement in performance. Further, our model predicts the larger the aggressive window size the greater the improvement. As noted however the assumptions of our model begin to break down as the size of the aggressive window approaches the mean of the service time distribution. Thus the model is not able to predict the impact of a significant level of aggressive processing, or unlimited aggressive processing as occurs in Time Warp.

Our model predicts, and simulation studies support, that the aggressive algorithm scales well as the size of the simulation problem increases. This is an important area which merits further investigation.

REFERENCES

Chandy, K.M. and J. Misra 1979. A Case Study in the Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering*, SE-5,5 May 1979, 440-452.

Chandy, K., and R. Sherman 1989. The Conditional Event Approach to Distributed Simulation. *Proceedings of the 1989 SCS Multiconference on Distributed Simulation*, 93-99, January, 1989.

Dickens, P. and P. Reynolds 1990. SRADS with Local Rollback. *Proceedings of the 1990 SCS Multiconference on Distributed Simulation*, 161-164, January, 1990,

Dickens, P., D. M. Nicol, P. F. Reynolds, and J. M. Duva, 1993. Analysis of an Aggressive Global Windowing Algorithm. In preparation.

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

Lubachevsky B. 1988. Bounded Lag Distributed Discrete Event Simulation. *Proceedings of the 1988 SCS Multiconference on Distributed Simulation*, 183,191, January, 1988.

Lubachevsky B., A. Shwartz and A. Weiss 1989. Rollback Sometimes Works... If Filtered. *Proceedings of the 1989 Winter Simulation Conference.* 630-639, December, 1989.

Lubachevsky, B. 1989a. Scalability of the Bounded Lag Distributed Event Simulation. *Proceedings of the 1989 SCS Multiconference on Distributed Simulation* 100-105, January, 1989.

Madisetti, V., D. Hardaker and R. Fujimoto 1992. The MINDIX Operating System for Parallel Simulation. *Distributed Simulation*, SCS Simulation Series, Vol. 24, Num. 3, pgs. 65-74, Jan. 1992.

Nicol, D. 1993. The Cost of Conservative Synchronization in Parallel Discrete Event Simulation. *JACM*, to appear, April 1993.

Nicol, D.M., 1993a. Global Synchronization for Optimistic Parallel Discrete Event Simulation. *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, ed. R. Bagrodia and D. Jefferson, 27-34.San Diego, California.

Steinman, J. 1992. SPEEDES: A Unified Approach to Parallel Simulation. *Distributed Simulation*, SCS Simulation Series, Vol. 24, Num. 3, pgs. 75-84, Jan. 1992.

Turner, S. and M. Xu 1992. Performance Evaluation of the Bounded Time Warp Algorithm. *Distributed Simulation*, SCS Simulation Series, Vol. 24, Num. 3, pgs. 117-126, Jan. 1992.

AUTHOR BIOGRAPHIES

PHILLIP M. DICKENS, Ph.D., University of Virginia, 1992, is a staff scientist at ICASE at the NASA Langely Research Center. His research interests are in performance modeling and in the parallelization of existing sequential simulation languages.

DAVID M. NICOL, Ph. D., University of Virginia, 1985, is an Associate Professor at the College of William and Mary. His interests are in parallel simulation, performance analysis, and algorithms for mapping parallel workload.

PAUL F. REYNOLDS, JR., Ph.D., University of Texas at Austin, 1979, is an Associate Professor of Computer Science at the University of Virginia. He has published widely in the area of parallel computation, specifically in parallel simulation, and parallel language and algorithm design.

JOHN M. DUVA, Ph.D., Harvard University, 1985, is an Associate Professor of Applied Mathamatics at the University of Virginia. His current research includes the modeling consolidation processing of advanced metal matrix composites and failure initiation in notched composite laminae.