Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

An Optimal Repartitioning Decision Policy

David M. Nicol and Paul F. Reynolds, Jr.*
Department of Computer Science
Thornton Hall, University of Virginia
Charlottesville, Virginia 22903

Abstract

The automated partitioning of simulations for parallel execution is a timely research problem. A simulation's run-time performance depends heavily on the nature of the inputs the simulation responds to. Consequently, a simulation's run-time behavior varies as a function of time. Since a simulation's run-time behavior is generally too complex to analytically predict, partitioning algorithms must be *statistically based*: they base their partitioning decisions on the simulation's observed behavior. Simulations which are partitioned statistically are vulnerable to radical changes in the run-time dynamics of the simulation. In this paper we discuss a *dynamic repartitioning decision policy* which detects change in a simulation's run-time behavior and reacts to this change. This decision policy **optimally** balances the costs and potential benefits of repartitioning a running simulation.

1. Introduction

Most simulations are executed sequentially despite evidence that at least the most common class of simulations, namely discrete event simulations, can benefit significantly from distributed execution. While numerous protocols have been designed to support distributed discrete event simulation, the problem of partitioning a simulation for distribution over multiple processors still remains. The goal of partitioning is optimal or near-optimal performance as measured by the completion time of the simulation. The most promising partitioning strategy is *dynamic*; partitioning is performed as the simulation is executing. Critical to the effectiveness of a dynamic partitioning strategy are policies for detecting load changes, deciding when to calculate a new partition, and deciding when to effect a new partition. In this paper we discuss optimality results with respect to policies for determining when to repartition a simulation.

Dynamic partitioning holds the greatest promise not because of its naturalness and simplicity as much as the inadequacy of *static* partitioning. Static partitioning, where partitioning is performed once on a program before it begins execution, has the disadvantage of being totally inflexible to changes in the resource (processor) requirements of an executing distributed simulation. Static partitioning is akin to making a single scheduling decision for a queue of jobs in an operating system and then ignoring opportunities that arise as scheduled jobs block on other resources. We have observed that many distributed simulations do change their processing requirements as inputs change [1] .

Dynamic partitioning has the potential disadvantage of being expensive. Again, lessons from operating systems theory suggest that a simple strategy is required; complex processor scheduling strategies have rarely outperformed their simpler counterparts, due primarily to characteristically substantial growth in implementation costs as the strategy's complexity increases. A recent study of models of dynamic load sharing suggests that significant benefit can be realized from simple partitioning strategies [2].

*present address: ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, VA, 23665
This work was supported by Virginia's Center for Innovative Technology

The repartitioning decision policy we present here is statistically based. We assume that we can observe the current behavior of a simulation, and given a potential repartition of the simulation, we can predict its behavior. We assume that we are dealing with a stochastic system; process resource and communication requirements change as the simulation executes. Despite, or more appropriately because of, our assumptions regarding the stochastic nature of the system we are able to demonstrate the optimality of our policy. The optimality of our policy, in turn, dictates that any algorithm for performing dynamic partitioning must take our policy into account in order to be optimal.

2. Statistical Partitioning

Simulations are often thought of as functions; a set of inputs are presented resulting in a set of outputs. This function is usually is composed of a large number of components interacting in complicated ways. A central problem in distributed simulation is how to partition the simulation, assigning its components to different processors so that parallelism is exploited. The most realistic formulation of this problem assumes that the parallel system's resources such as processors, memory, and communication bandwidth are all limited. The simulation should be partitioned so that these resources are shared in such a way that the simulation's execution is substantially reduced over that of a sequential simulation. A good partitioning algorithm needs to examine the simulation's static structure and identify functional dependencies between components. It needs to consider how often a simulation's components are called upon for execution, and how much of a processor's time is required to execute each component. Given this information, the partitioning algorithm attempts to assign components to processors so that components that can be executed in parallel are, and tightly coupled components are assigned to the same processor. Components assigned to the same processor should be chosen so that their competition for the *CPU* is minimized. It is critical to observe that **the simulation's execution behavior depends on the inputs presented to the system.** Some inputs may cause very little simulation activity, while others may cause a great deal of activity. Furthermore, the level of interaction between components may vary as a function of the inputs. Thus a partitioning algorithm must attempt to create a partition which works well for every expected set of simulation input.

One approach to quantifying a simulation's run-time behavior is to mathematically model the behavior of the simulation components in response to probabilistically generated inputs. This approach usually fails to be computationally feasible, due to the complicated relationships between the simulation's components [1]. An alternate approach is to base the partitioning algorithm's assignments on *observations* of the way the simulation behaves. These observations are taken to be representative of the overall behavior of the simulation. Partitioning algorithms utilizing such observations are *statistical partitioning* algorithms.

Statistical partitioning algorithms must presume that future simulation system behavior resembles the observed behavior. The performance of a system partitioned with a statistical partitioning algorithm is hostage to this assumption. If the running behavior of the simulation were to drastically change, the observations governing the partitioning are no longer representa-

tive of the run-time behavior. Co-resident components whose executions were once nicely interleaved may now demand *CPU* service at the same time, causing bottlenecks. Separated components which once communicated infrequently may now communicate often, tying up the communication channel. To deal with this potential failing of statistical partitioning algorithms, we have developed a dynamic repartitioning decision process that anticipates and detects such a change, and then reacts to it. This decision process gives rise to a decision policy which balances the computational costs and risks of calculating a new partition with the potential speedup benefits. This decision policy is provably optimal.

3. Problem Statement

Consider a simulation whose components have already been distributed. We suppose that the total simulation session can be divided into subsessions that are in some sense identical. One convenient definition of a subsession is the simulation's response to one set of inputs. The sum of processor utilizations during a subsession is a measure of system performance during that subsession; we consider the subsessions to be probabilistically identical if the sequence of utilization sums over each subsession forms a weakly stationary (or covariance) stochastic process [3]. We call such a subsession a *system cycle*; let Z_i denote the sum of processor utilizations over the *ith* cycle.

We now consider the possibility that at some unknown time t, the sequence Z_t, $Z_{t+1}, \cdots$ forms a weakly stationary process that is different from $Z_0, Z_1, \cdots, Z_{t-1}$. If s is a time greater than or equal to t, we will say that a *change* has occurred by time s. If a change occurs and the mean measure Z_j is observed to decrease, we can conclude that the system is not running at the level of performance it once had. This drop in performance can be caused by a change in the dynamics of the running simulation; a change which causes bottlenecks due to the partitioning. The system performance might be improved then by repartitioning. We treat the problem of deciding whether or not a change has occured, and whether or not to calculate and adopt a new partition by a Markov decision process.

A *decision process* can be thought of as a sequence of actions; time is divided into intervals, and an action is chosen at the end of each interval as a function of the *decision policy*. Each possible action has an associated cost. The decision process we envision for our problem must determine, at the end of each time interval, whether or not a change has occurred. The process then has the option of either calculating a new partition, or continuing with the old one. If a new partition is calculated, it is tested against the old partition on the most recent workload profiles; this test determines if substantial enough improvement can be expected by using the new partition. The cost of the decision to repartition has two components. The system suffers delay due to the calculation and testing of the new partition; the second component is a benefit, or negative cost. This benefit is achieved if the new partition is found to be superior to the old one; the benefit is the expected differential in system finishing time between the system running under the old partition and the system running under the new partition. If the new partition is found superior, and is thus adopted, the decision process is considered to have *stopped*. If the new partition is found inferior to the old partition, we conclude that the change did not occur, and continue the decision procedure.

If the decision process does not choose to calculate a new partition, the system is run for the next interval of time under the old partition. The cost of this decision is a "lost opportunity" cost: this cost is incurred if the change has already occurred and future repartitioning benefits are achievable. The decision to continue for another interval under the old partition foregoes the benefit of the new partition for that interval.

The cost of a decision policy is the sum of the costs incurred at each decision dictated by that policy. An optimal decision policy is one with minimal expected cost. Intuitively, we see that the optimal policy must balance the costs and benefits of repartitioning with the costs of not repartitioning. We have determined an optimal decision policy for the repartitioning problem. Similar change detecting problems have been treated in [4] ; our work differs principally in that we require the repartitioning benefit to vary as a function of time. Likewise, the general statistical field of sequential analysis [5] treats the problem of deciding when to stop sampling; however, this sort of analysis generally assumes distributional knowledge of the quantities involved, and focuses on the minimization of the number of samples taken. This minimization metric gives equal cost to each sample; we will observe that a precise model of our problem will give observations unequal expected costs.

4. Problem Formulation

In this section we model the dynamic repartitioning problem as a Markov decision process. This requires a precise definition of Markov decision processes, and the formulation of dynamic partitioning within the confines of this definition. This formulation will be seen to depend on our ability at each point in time to calculate the probability that the anticipated change has already occurred. This probability is affected by a statistical procedure which examines recent system behavior and decides whether or not the change has occurred. We will treat the statistical issues of detecting change first, and then proceed to the larger task of formulating dynamic partitioning in terms of a Markov decision process.

4.1. Statistical Issues in Detecting Change

There are two issues we need to address. Most useful statistical tests assume that the measurements analyzed are independent; many assume normality as well. If we are to use such tests with confidence, we must first transform our measurements to fit the tests' assumptions. The second issue is simply how we intend to detect change. We propose solutions to both of these problems.

We can expect the sequence of system performance measurements to be correlated. Furthermore, the distribution of these measurements can be arbitrary. Before we can hope to confidently use statistical techniques on this data, we need to transform the data in some way to make it more amenable to analysis. The "batch means" [6] method is appropriate for this task. The resulting sequence of group means is the sequence analyzed.

We detect change in a system's dynamic behavior by analyzing the distribution of the transformed observations. We must examine the batch mean observations as they become available, and determine if, at some point in time, a significant and sustained difference in these observations' distribution occurs. Solutions to the so-called model identification problem [7] in statistics can be used to detect this change. As applied to our situation, the model identification problem is to decide whether two groups of independent normal observations are best described as being drawn from the same distribution, or from two different distributions.

Using a model identification approach, we detect distributional change in the sequence of normal random variables by creating a test cluster of observations. We assume the existence of a base cluster derived from initial observations taken before a change could have occurred. A test cluster is statistically compared with the base cluster; the test determines whether or not the two clusters are identically distributed. Positive indication of distributional difference is evidence for the change having already occurred.

The model selection criterion provides a statistical means of testing for change. As a statistical test, it may fail to correctly identify the true state of nature. If we treat the problem of identifying change in a Bayesian framework, the statistical test gives us certain information about the true state of nature. In particular, if we have a prior probability of the change having already occurred, we can calculate a posterior probability of the change having already occurred as a function of the test result. Calculation of the posterior probability requires knowledge of the statistical test's accuracy. We denote by α the probability that the model selection statistic falsely indicates a change (type I error); we denote by β the probability that the statistic fails to detect a change (type II error).

4.2. Model Formulation

A Markov decision process is a stochastic process which examines its state at different points in time; at each examination it chooses some action. Associated with each action is a cost, which depends on the state and the action. Conditioned on the action chosen, the next state of the process is chosen in accordance with some transition probability distribution. We may formulate the dynamic repartitioning decision problem as a Markov decision process [8]. We identify the points in time at which decisions are made, the decision process states, the state dependent action costs, and the state and action dependent transition probabilities. Each of these topics is addressed in turn.

Time Steps and Process States: The construction of the model selection test for change allows us to make this test every $b \cdot c$ system cycles, where b is the size of the batch means group and c is the number of observations in a cluster. We consider time to be divided into a sequence of *decision steps*, with $b \cdot c$ system cycles defining the period between two adjacent decision steps. Time n is thus considered to be the time corresponding to the *nth* decision step. Furthermore, we assume that the system runs for exactly N decision steps.

The definition of our formal model revolves around the probability that the anticipated change has already occurred. At time n, we denote this probability by p_n. The state of the decision process at time n is defined to be the pair $<p_n, n>$. Maintaining the probability that the change has already occurred is the central activity of the decision process.

Maintaining the Probability of Change: We suppose that the decision process is in state $<p, n>$. A number of probabilities are of interest to us. Let $p^*(p)$ denote the probability that the system *will have* changed by time $n+1$, conditioned on the value of $p_n = p$. This probability is dependent on prior knowledge of the distribution of the time of change. Supposing that such information is not precisely known, it is reasonable (and convenient) to assume that the failure rate of this distribution is some constant ϕ. Using simple conditioning arguments we have

$$p^*(p) = p + \phi \cdot (1 - p)$$
$$= (1 - \phi) \cdot p + \phi.$$

The probability $p^*(p)$ represents the probability of change at time $n+1$ given only the value of $p_n = p$ and foreknowledge of the change time failure rate. The probability of change by time $n+1$ is enhanced by an observation of the system at time $n+1$ since each observation yields additional information related to the probability of change. This posterior probability is given directly by Bayes' Theorem [9] . If $p_n = p$, and an observation at time $n + 1$ indicates change, change, the posterior probability p_{n+1} is equal to

$$p^c(p) = \frac{p^*(p) \cdot (1 - \beta)}{p^*(p) \cdot (1 - \beta) + (1 - p^*(p)) \cdot \alpha}.$$

Likewise, given a negative indication of change, the posterior probability is

$$p^{\overline{c}}(p) = \frac{p^*(p) \cdot \beta}{p^*(p) \cdot \beta + (1 - p^*(p)) \cdot (1 - \alpha)}.$$

These equations allow us then to maintain the probability that the change has already occurred as a function of the prior probability, and the result of the last test for change.

Decision Actions and Costs: Upon arrival at state $<p_n, n>$, the decision process has the option of choosing one of two actions. It may choose to *continue*, or it may choose to *test*. The decision to test is the decision to calculate and possibly adopt a new partition. The Markov decision model requires us to define action costs that depend on the state $<p_n, n>$. An understanding of these costs is obtained by considering the effects of the chosen action.

Test Decision Consequences and Costs: The decision to test initiates a two step process. The system is halted, and a new partition is calculated on the basis of recent system behavior. Given a new partition and performance traces of recent system behavior (e.g., observed precedences, activity execution times), we predict what the performance of the system would be under the new partition. [1] and [10] treat this in some detail. It is possible then to compare the performance of the system under the old and new partitions, and choose the better one. We will assume that the new partition is chosen if and only if the system has truly changed and is significant enough to warrant repartitioning. As a consequence, if the old partition is retained, the probability that the system has (significantly) changed by time n is considered to be 0. The system is restarted once the appropriate partition is selected. The decision process is considered to have stopped if the new partition is chosen. If the old partition is chosen, the process resumes, using $p_n = 0$ as its initial prior probability.

The computational delay of calculating and testing a new partition is assumed to be some constant D_d. If the new partition is chosen, an additional delay of D_r is incurred to physically effect the repartitioning. When the new partition is chosen, we expect a resultant speedup over the time to finish the session under the old partition. This differential speedup can be estimated when the system initially indicates change; the speedup, or gain over one decision step's worth of time is denoted by G. Given a finite number of N decision steps, the speedup resulting from a new partition chosen at time n is $G \cdot (N - n + 1)$.

Integrating all of these observations, we see that the *expected* cost of choosing to test at state $<p_n, n>$ is

$$E[Cost(<p_n, n>, test)] = D_d - p_n \cdot \left[G \cdot (N - n + 1) - D_r \right]$$

Continue Consequences and Costs: The consequences of the continue decision are substantially simpler than those of the test decision. The decision to continue simply allows the system to run for one more time period under the old partition. At the end of the time period the system is observed for change once again, and a new posterior probability of change is calculated. This posterior probability defines the decision process state at the end of this time period. The cost associated with the continue decision is a "lost opportunity" cost. By choosing to continue, the process foregoes any possible benefit from repartitioning in the next decision step's worth of system time. We presume that our partitioning algorithm can produce a better new partition only if the change has already occurred: the probability of this is p_n. We incur no cost by continuing if the change has not yet occurred. Consequently, the expected cost of choosing to continue at state $<p_n, n>$ is simply

$$E[Cost(<p_n, n>, continue)] = p_n \cdot G$$

State Transition Probabilities: After an action in state $<p_n, n>$, the process makes a transition; we first consider the transition after a decision to continue. By continuing, the system runs under the old partition until time $n+1$, at which point the system is again tested for change. The posterior probability that the system has changed by time $n+1$ defines the next process state $<p_{n+1}, n+1>$; p_{n+1} is equal either to $p^c(p_n)$ or $p^{\overline{c}}(p_n)$, depending on the outcome of the test for change at time $n+1$. The state transition probabilities out of $<p_n, n>$ after a continue decision are thus defined by the probability of observing a change at time $n+1$.

Let $q^c(p)$ denote the probability that the change test would report a change at time $n+1$, given that the probability of change by time n is p. We recall that α and β denote the type I and type II errors of the test procedure; again, direct conditioning arguments establish that

$$q^c(p) = p^*(p) \cdot (1 - \beta) + (1 - p^*(p)) \cdot \alpha.$$

The probability of observing no change at time $n+1$ is

$$q^{\overline{c}}(p) = 1 - q^c(p)$$
$$= p^*(p) \cdot \beta + (1 - p^*(p)) \cdot (1 - \alpha)$$

We now consider the transition probabilities out of $<p, n>$ after the decision to test. The probability that the new partition is adopted is just p; should this occur, the decision process

stops so that there is no next state. Conversely, the probability of rejecting the new partition is $1-p$. After rejection, we infer that $p = 0$; the state of the process at time $n+1$ is thus one of the two states reachable from state $<0,n>$, and depends again on the outcome of the change test at time $n+1$. Thus the probability of passing from $<p,n>$ into a particular state at time $n+1$ is $1-p$ times the probability of reaching that state from $<0,n>$.

5. Process Optimal Cost Function

The importance of Markov decision processes lies in a theorem which allows an optimal decision policy to be identified. This theorem is simply a statement of dynamic programming. Applied to our problem, this theorem states that given the process is in state $<p,n>$, the expected future cost of the optimal decision policy is the minimum of (i) the expected cost of choosing now to test, and thereafter using the optimal decision policy, and (ii) the expected cost of choosing now to continue, and thereafter using the optimal decision policy. We denote the expected future cost of the optimal decision policy from $<p,n>$ by $V(<p,n>)$. The formal statement of this theorem is aided by a further notational device. Given that the state at time n is $<p,n>$, we denote the mean expected future costs at $n+1$ by $E_v[<p,n>]$, and observe

$$E_v[<p,n>] = q^c(p)\cdot V(<p^c(p),n+1>) + q^{\overline{c}}(p)\cdot V(<p^{\overline{c}}(p),n+1>).$$

Then the relationship satisfied by the optimal cost function V is stated as

$$V(<p,n>) = \min\begin{cases} D_d - p\cdot\left[G\cdot(N-n+1) - D_r\right] + (1-p)\cdot E_v[<0,n>] \\ p\cdot G + E_v[<p,n>] \end{cases} \tag{1}$$

Furthermore, the optimal decision to make in state $<p,n>$ is the action associated with the cost function defining this minimum. The uppermost function on the right hand side of equation (1) is the optimal expected future cost function associated with the test decision, denoted by $ECT(<p,n>)$. The lower function is the optimal expected future cost function associated with the continue decision, and is denoted by $ECC(<p,n>)$.

Since $V<p,n>$ is the minimal expected future costs of the process in state $<p,n>$, we define $V(<p,N+1>) = 0$: the process never advances beyond time $N+1$.

6. Optimal Policy Structure

Any decision policy whose expected cost function satisfies equation (1) above is optimal. In [1] we have shown that the optimal decision policy is characterized by a sequence $\pi_1,\cdots,\pi_N$ of constants from the interval $[0,1]$. The optimal decision to make in state $<p,n>$ is to test if and only if $p > \pi_n$. Theorem 1 summarizes this development. For every $n = 1,2,\cdots,N$, $ECT(<p,n>)$ and $ECC(<p,n>)$ intersect at most once for $p \in [0,1]$. Furthermore, whenever $ECT(<p,n>)$ and $ECC(<p,n>)$ do intersect at $\pi_n \in [0,1]$, then $ECT(<p,n>) < ECC(<p,n>)$ for all $p > \pi_n$.

Supposing that the intersection points π_n are available, upon arrival at state $<p,n>$ the decision process compares p with the threshold π_n. If $p \leq \pi_n$, then $V(<p,n>)$ is equal to $ECC(<p,n>)$, so that the continue decision is optimal. On the other hand, if $p > \pi_n$, then the test decision is optimal. The utility of this result hinges then on our ability to determine the points of intersection π_n.

Equation (1) is seen to express the function $V(<\cdot,n>)$ in terms of $V(<\cdot,n+1>)$. Furthermore, $V(<p,N+1>) = 0$ for all p. We can therefore derive the points of intersection π_n numerically, by solving this recursive set of equations. An efficient method of solution is developed in [1].

7. Other Concerns

We have also addressed other concerns arising from the definition of this decision process. Some of the parameters used by the model cannot reasonably be estimated before some indication of change is observed. The problem faced by the system then, is the choice of action if no observation of change has yet occured. We have shown that under such circumstances, the continue decision is always optimal. Consequently, the model equations need not be solved until some indication of change occurs.

Another concern considers the parameter defining the expected repartitioning gain per decision step, G. The precise value of G cannot realistically be known. However, it may be possible to bound G from above and below. A point estimate of G might then be chosen from the interval defined by these bounds. We have shown that the points of intersection π_n are decreasing functions of G. This implies that the use of G's upper and lower bounds in equation (1) leads to tight upper and lower bounds on the true π_n. Using a point estimate for G is thus a reasonable heuristic: the resulting estimations $\hat{\pi}_n$ lie within known tight bounds on the true π_n.

8. Conclusions

The costs associated with dynamic repartitioning could easily overwhelm any potential benefits that might be gained from repartitioning. A cost effective repartitioning policy must balance its implementation costs against potential benefits. In this paper we have presented an optimal repartitioning policy that assumes a statistical model of execution behavior. With this assumption we have benefited twice; the policy we describe is optimal, due mainly to assumptions the stochastic nature of our model allows us to make, and the decision analysis our policy requires is inexpensive. That is not to say that all of our assumptions are cheap to implement. We have assumed that we have statistics on the past behavior of an existing partition and that we have sufficient information to probabilistically predict the future behavior of a proposed partition. The costs associated with using this information are not low.

We have not discussed architectures which we believe would support our partitioning policy efficiently and will not do so here. We note that all of the analysis required for our repartitioning policy could be carried on in parallel with an executing simulation. For this reason we are encouraged from a practical perspective by our results. From a theoretical perspective, the optimality of our repartitioning policy demonstrates that a relatively simple policy can be optimal. We have good reason to believe that dynamic partitioning can be cost effective.

References

[1] D. Nicol, The Automated Partitioning of Simulations for Parallel Execution, Ph.D. Dissertation, University of Virginia, August 1985.

[2] D. Eager, E. Lazowska and J. Zahorjan, Dynamic Load Sharing in Homogeneous Distributed Systems, Tech Report 84-10-01, University of Washington.

[3] S. Ross, Stochastic Processes, Wiley and Sons, New York, 1983.

[4] A. Rapoport, W.E. Stein and G.J. Burkheimer, Response Models for Detection of Change, D. Reidel Publishing Company, Boston, 1979.

[5] Z. Govindarajulu, Sequential Statistical Procedures, Academic Press, 1975.

[6] G. Fishman, "Grouping Observations in Digital Simulation", Management Science 24, (1978), 510-521.

[7] H. Bozedogan, S. Sclove, "Multi-Sample Cluster Analysis Using Akaike's Information Criterion", Annals of the Institute of Statistical Mathematics 36,1, (1983).

[8] S. Ross, Applied Probability Models with Optimization Applications, Holden-Day, San Fransico, 1970.

[9] S. Schmitt, An Elementary Introduction to Bayesian Statistics, Addison-Wesley, 1969.

[10] D. Nicol and P. Reynolds, "A Statistical Approach to Dynamic Partitioning", Proceedings of the SCS Multi-Conference, San Diego, January 1985, 53-56.

DAVID M. NICOL

David Nicol is a staff scientist with the Institute for Computer Applications in Science and Engineering, Hampton, Virginia. He was graduated in 1979 from Carleton College, Northfield, Minnesota, with a B.A. in mathematics. He spent three years as a programmer analyst with the Control Data Corporation, designing and developing distributed real-time signal processing systems. He entered the University of Virginia in 1982, taking an M.S. in computer science in 1983, and a Ph.D. in computer science in 1985. His research interests include distributed systems, parallel algorithms, and performance analysis. Dr. Nicol is a member of the Association for Computing Machinery, and the IEEE Computer Society.

ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, Virginia 23665
(804) 865-2513

PAUL F. REYNOLDS

Paul Reynolds received the B.A. degree from Ohio Northern University, Ada, Ohio, in 1970 and M.S. and Ph.D. degrees in computer science at the University of Texas at Austin in 1975 and 1979. He was an Instructor of computer science at the University of Texas at Austin in 1979-1980. Since the Fall of 1980 he has been an Assistant Professor of computer science at the University of Virginia. He has consulted for a number of large corporations, as well as government agencies. His research interests include distributed systems, language design, and performance evaluation. Dr. Reynolds is a member of the Association for Computing Machinery, and the IEEE Computer Society.

Department of Computer Science
Thornton Hall
University of Virginia
Charlottesville, Virginia 22903
(804) 924-1039