UNBOUNDEDLY PARALLEL SIMULATIONS VIA RECURRENCE RELATIONS FOR NETWORK AND RELIABILITY PROBLEMS

Albert G. Greenberg
Boris D. Lubachevsky

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974

Isi Mitrani

Computing Laboratory
University of Newcastle
Newcastle Upon Tyne, NE1 7RU
United Kingdom

ABSTRACT

New methods are presented for parallel simulation of discrete event systems that, when applicable, can usefully employ a number of processors much larger than the number of objects in the system being simulated. The simulation problem is posed using recurrence relations, and three algorithmic ideas are brought to bear on its solution: parallel prefix computation, parallel merging, and iterative folding. Efficient parallel simulations are given for the G/G/1 queue, a variety of queueing networks having a global first come first served structure (e.g., a series of queues with finite buffers), acyclic networks of queues, and networks of queues with feedbacks and cycles. In particular, the problem of simulating the arrival and departure times for the first N jobs to a single G/G/1 queue is solved in time proportional to $N/P + \log P$ using P processors.

1. INTRODUCTION

The simulation of a discrete event system is the process of generating a sample path, or trajectory, that represents the system state as a function of time. This normally entails the use of a global clock and an event list. In the last few years, much effort has been devoted to the task of splitting the simulation process into a number of sub-processes and executing the latter in parallel on different processors [Chandy and Misra 1979; Lubachevsky 1989; Misra 1986; Nicol 1988; Wagner and Lazowska 1989; Jefferson 1985]. For example, when simulating a queueing network, the idea might be to allocate each processor to a node, or a group of nodes, and let it handle the corresponding events, taking care of possible interactions with other processors. At best, the degree of parallelism obtained by such an approach will be equal to the number of nodes, and in general may be much smaller [Wagner and Lazowska 1989].

We propose new methods that do not limit the degree of parallelism in this way. Our methods require only single instruction, multiple data (SIMD) parallel processing. The methods can be efficiently implemented on the current generation [Thinking Machines Corporation 1989; Grondalski 1987] of SIMD massively parallel processors. Some preliminary benchmarks of simulations performed on an $8K$ processor Connection Machine (CM-2, Thinking Machines, Incorporated) are reported below. A more complete description of our work can be found in [Greenberg et al. 1990].

Our point of departure is to pose the simulation problem using recurrence relations. When the relations are of a certain type, computing their solution is a *parallel prefix problem* [Ladner and Fischer 1980]. The first algorithmic idea introduced here is to use fast algorithms for the parallel prefix problem [Greenberg et al. 1982; Hyafil and Kung 1977] to solve the recurrences and thereby simulate the system. The simplest, non-trivial system that yields to this approach is the G/G/1 queue, using the first come first served discipline. In this case, the recurrence relations define the sequences of job arrival and departure instants. We show that the arrival and departure times for the first N jobs can be computed in time proportional to $N/P + \log P$ using P processors. Memory requirements are moderate. If $B = O(N)$ memory locations are available, the time to simulate N jobs is proportional to

$$N/P\,(1 + P \log P/B),$$

which implies linear speedup with respect to N, the number of jobs, provided $B > P \log P$.

Other systems that have a global first come first served structure yield to the same approach, for example, acyclic fork/join networks [Baccelli et al. 1989], queues connected in a cycle, and queues in series with finite buffers [Mitra and Mitrani 1989]. We show that the first N events of any of these systems can be simulated in time proportional to $N/P + \log P$ using P processors, though the constant of proportionality may grow with the size of the network.

To extend the scope of our simulation methods to queueing networks that allow *overtaking* (meaning a pair of jobs A and B may exit a given node in the order A, B, and another in the order B, A), we combine fast parallel prefix algorithms with fast *parallel merging* algorithms to produce the job arrival and departure instants for each queue in the network. This leads to parallel simulations of acyclic networks of queues, which use P processors to produce the job arrival and departure times at each node for the first N jobs through the network in time order $N/P + \log P + \log N$.

To handle general networks of G/G/1 queues, with overtaking, feedbacks (the same job may visit the same node several times) and cycles (jobs may flow between a pair of nodes in both directions), breakdowns (jobs wait until service is restored), and priorities, a third algorithmic idea, *iterative folding*, is introduced wherein:

1. Complex dependencies (cause and effect relationships) between events are temporarily relaxed, and *estimates* are used for the timing of some of these events. A *tentative* sample path is then computed, using parallel prefix and merging algorithms.

2. The previously relaxed dependencies are then taken into account to update the estimates.

As this process is iterated, a wave of events, which begins at simulated time $t = 0$ and propagates toward $t = \infty$, converges to the correct values, and this convergence is guaranteed. Each iteration of this method, producing a new tentative sample path, is always fast. In particular, using P processors, order $B/P + \log B + \log P$ time is required to simulate a batch of B jobs. The efficiency of the method hinges on how many iterations are needed for convergence. A wide range of experiments for a variety of networks at a wide range of loads (and overloads) show that the number of iterations is small, about $\log B$ iterations for a batch of B jobs.

In this work, the simulation problems considered are such that all random variables can be precomputed, without regard to the state the system happens to be in. These variables are the job interarrival times, the routes the jobs take through the network, and their service demands along the route. To a limited degree, this sort of precomputation was exploited in [Lubachevsky 1987], and in the "future event list" technique [Nicol 1988]. Precomputing random choices may help accelerate the simulations of some networks where routing is state dependent. Work is in progress on applying our parallel simulation approach to the simulation of multiple access protocols such as Aloha [Abrahamson 1985; Greenberg et al. 1990]. We are also working on the simulation of loss systems (circuit switching networks) [Eick et al. 1990].

For the G/G/1 system, if the object of interest is the waiting time, it would also be possible to use Lindley's recurrences [Kleinrock 1975], rather than those presented below. However, Lindley's recurrences do not contain enough information to reconstruct the sample path. It was known that these recurrences provide a simple serial simulation algorithm for generating job waiting times for the G/G/1 queue (see, e.g., [Gaver and Thompson 1973]). Baccelli and his coworkers introduced recurrence relations generalizing Lindley's recurrences to describe embedded processes in a variety of first come first served stochastic systems [Baccelli et al. 1989; Baccelli 1989]. Baccelli [1989] recently characterized these systems as a restricted class of stochastic Petri nets. In this work the recurrences serve as the starting point for the study of ergodicity conditions and for the derivation of stochastic orderings and bounds.

Chandy and Sherman [1989] posed simulation as the problem of filling in a *space-time* rectangle containing the events. The task of filling in this rectangle is viewed as the problem of finding the unique fixed point X of a function F, $X = F(X)$, where X represents an assignment of events in the space-time rectangle (i.e., a possible sample path) and F expresses the dependencies that determine each event from neighboring events. This appealing view unifies earlier optimistic [Jefferson 1985] and conservative [Misra 1986] approaches to parallel simulation. Chandy and Sherman [1989] proposed using relaxation for finding the fixed point. Roughly, the idea is to tile the space-time rectangle in any advantageous manner, and assign one process to each tile with the task of filling in the events for that tile. To take into account information flowing across from neighboring tiles, it would appear to be necessary to serialize the computation. To avoid this serialization each tile process makes assumptions about the events in neighboring tiles. As information from the neighboring tiles is received, the tile process corrects the assumptions and repeats the simulation within the tile. We also approach simulation as the task of finding a fixed point. However, while Chandy and Sherman [1989] propose no specific tilings for fast convergence, we do (see also [Eick et al. 1990]). In addition, we specify massively parallel algorithms for propagating corrections to the errors in the event placement assumptions. Specifications that guarantee efficiency are important to the practical use of the ideas in [Chandy and Sherman 1989]; errors in the assumptions are inevitable, and relaxation alone can propagate corrections to these errors slowly. For example, applying the method of Chandy and Sherman [1989] to the problem of just computing the first N job arrival times to a G/G/1 queue results in serial convergence; order N time is consumed, regardless of the number of processors used. In contrast, our method for this problem is efficient because it rapidly propagates information about early events to guide the placement of later events.

2. IMPLEMENTATION

The parallel simulation methods proposed here are applicable to a wide variety of parallel architectures. The methods call on a small number of basic parallel processing operations: parallel prefix (sometimes called *scan*), reduce computation, linear recurrence computation, matrix multiplication, and parallel merging and sorting. These operations have been widely studied and programmed for a great variety of architectures, including systolic arrays, hypercubes, butterflies, ultracomputers, meshes, etc. [Leighton 1989]. These operations can all be implemented on SIMD parallel processors. Indeed, our simulations appear well suited for fine-grained "data parallel" machines, such as the Connection Machine [Thinking Machines Corporation 1989], where the computation proceeds via massively parallel synchronous transformations of arrays of data.

We implemented the simulation of series of G/G/1 queues on a CM-2 Connection Machine, having 8192 1-bit processing elements and 256 32-bit floating-point accelerators [Greenberg et al. 1990]. This simulation has a dual role: (i) to serve as a first benchmark of our methods, and (ii) to obtain results of current interests to queueing theorists. Recently, Srinivasan [1989] and Glynn and Whitt [1990] have obtained novel results on the transient behavior of series of single-server queues, which could be used to model the start-up behavior of a long production line or the transient flow of messages over a long path in a communication network. Glynn and Whitt [1990] prove limit theorems about the departure time of the k^{th} customer from the n^{th} queue as, for example, n tends to ∞ with k fixed, or with k increasing with n, say $k = \sqrt{n}$ or $k = n$. Though their analysis shows existence of limits, it does not specify the limits numerically except in a very few special cases. Thus, it is of interest to explore the numerical values through simulation, especially where the invariance principle of [Glynn and Whitt 1990] applies. Via the invariance principle the results can be applied universally, for essentially any service time distribution.

Table 1 gives timings for simulations of a single M/M/1 queue, and Table 2 gives timings for simulations of series of M/M/1 queues. In these runs just end-to-end response time statistics were collected. On long simulation runs, the CM-2 code is over 100 times faster than corresponding code on a MIPS RS2000 workstation; no special optimization tricks were tried in either code. The speed of the parallel series of queues code is about 17 billion simulated services per hour.

Table 1. Time to Simulate a Single M/M/1 Queue

Number of Jobs Simulated	Running Time (seconds)
2^{13}	.01
2^{15}	.02
2^{16}	.02
2^{17}	.03
2^{18}	.06
2^{19}	.12
2^{20}	.42
2^{21}	.80

Table 2. Time to Simulate 2^{21} = 2,097,152 Jobs Through a Series M/M/1 Queues

Number of Queues	Running Time (seconds)
10	4.87
100	45.11
1000	446.03

ACKNOWLEDGEMENTS

We are grateful to Otmar Schlunk for programming the CM-2, to Sandia National Labs and Thinking Machines, Incorporated for providing the computer time, and to Ward Whitt and Raj Srinivasan for their collaboration (in progress) on the series of queues experiments.

REFERENCES

Abrahamson, N. (1985), "Development of the ALOHANET," *IEEE Transactions on Information Theory IT-31*, 2, 119-123.

Baccelli, F. (1989), "Ergodic Theory of Stochastic Petri Nets," Technical Report 1037, INRIA-Sophia Antipolis, INRIA-Sophia 06565, Valbonne, France.

Baccelli, F., W. Massey, and D. Towsley (1989), "Acyclic Fork-join Queueing Networks," *Journal of the ACM 36*, 3, 615-642.

Chandy, K.M. and J. Misra (1979), "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs," *IEEE Transactions on Software Engineering SE-5*, 5.

Chandy, K.M. and B. Sherman (1989), "Space-Time and Simulation," In *Distributed Simulation 1989*, The Society for Computer Simulation, San Diego, CA, 53-57.

Eick, S., A.G. Greenberg, B.D. Lubachevsky, and A. Weiss (1990), "SIMD Relaxation for Parallel Simulations with Applications to Circuit-Switched Networks," in preparation.

Gaver, D.P. and G.L. Thompson (1973), *Programming and Probability Models in Operations Research*, Brooks/Cole Division of Wadsworth, Monterey, CA.

Glynn, P.W. and W. Whitt (1990), "Departures from Many Queues in Series," Technical Report, AT&T Bell Laboratories.

Greenberg, A.G., R.E. Ladner, M. Paterson, and Z. Galil (1982), "Efficient Parallel Algorithms for Linear Recurrence Computation," *Information Processing Letters 15*, 1, 31-35.

Greenberg, A.G., B.D. Lubachevsky, and I. Mitrani (1990), "Unboundedly Parallel Simulations Via Recurrence Relations," In *1990 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, ACM, New York, 1-12.

Greenberg, A.G., O.S. Schlunk, and B.D. Lubachevsky (1990), "Fast SIMD Parallel Simulations," in preparation.

Grondalski, R. (1987), "A Chip Set for a Massively Parallel Architecture," In *IEEE International Solid State Circuits Conference*, IEEE, New York, NY.

Hyafil, L. and H.T. Kung (1977), "The Complexity of Parallel Evaluation of Recurrences," *Journal of the ACM 24*, 513-521.

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

Kleinrock, L. (1975), *Queueing Systems, Volume 1: Theory*, Wiley, New York.

Ladner, R.E. and M.J. Fischer (1980), "Parallel Prefix Computation," *Journal of the ACM 27*, 831-838.

Leighton, T. (1989), "An Introduction to the Theory of Networks, Parallel Computation, and VLSI Design," Draft.

Lubachevsky, B.D. (1987), "Efficient Parallel Simulations of Asynchronous Cellular Arrays," *Complex Systems*, 1, 1099-1123.

Lubachevsky, B.D. (1989), "Efficient Distributed Event-Driven Simulation of Multiple-Loop Networks," *Communications of the ACM 32*, 1, 111.

Misra, J. (1986), "Distributed-Discrete Event Simulation," *ACM Computing Surveys 18*, 1, 39-66.

Mitra, D. and I. Mitrani (1989), "Control and Coordination Policies for Systems with Buffers," *1989 ACM Sigmetrics Performance Evaluation Review and Performance '89 17*, 1, 156-164.

Nicol, D.M. (1988), "Parallel Discrete-Event Simulation of FCFS Queueing Networks," In *Parallel Programming: Experience with Applications, Languages, and Systems*, ACM, New York, NY, 124-137.

Srinivasan, R. (1989), "Queues in Series Via Interacting Particle Systems," Technical Report, Department of Mathematics, University of Saskatchewan.

Thinking Machines Corporation (1989), *Connection Machine, Model CM-2 Technical Summary, Version 5.1*, Thinking Machines Corporation, Cambridge, MA.

Wagner, D.B. and E.D. Lazowska (1989), "Parallel Simulation of Queueing Networks: Limitations and Potentials," *1989 ACM Sigmetrics Performance Evaluation Review and Performance '89 17*, 1, 146-155.