

ORNL/TM-11064

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Multiprogramming a Distributed-Memory Multiprocessor

Michael R. Leuze
Lawrence W. Dowdy
Kee Hyun Park

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A03; Microfiche A01

Engineering Physics and Mathematics Division

Mathematical Sciences Section

**MULTIPROGRAMMING A DISTRIBUTED-MEMORY
MULTIPROCESSOR**

Michael R. Leuze †
Lawrence W. Dowdy †
Kee Hyun Park †

† Oak Ridge National Laboratory
Mathematical Sciences Section
P.O. Box 2009, Bldg. 9207-A
Oak Ridge, TN 37831-8083

† Department of Computer Science
Vanderbilt University
Nashville, TN 37235

Date Published: January, 1989

Research of the first author was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy. Research of the second author was supported in part by the Alexander von Humboldt Foundation while on leave to the University of Erlangen-Nürnberg, Erlangen, FRG.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC-05-84OR21400



Contents

1	Introduction	1
2	The Model	3
3	Adaptation of the Model	6
4	Numerical Experiments	9
5	Summary	12

MULTIPROGRAMMING A DISTRIBUTED-MEMORY MULTIPROCESSOR

Michael R. Leuze
Lawrence W. Dowdy
Kee Hyun Park

Abstract

The development of computing systems with large numbers of processors has been motivated primarily by the need to solve large, complex problems more quickly than is possible with uniprocessor systems. Traditionally, multiprocessor systems have been uniprogrammed, i.e., dedicated to the execution of a single set of related processes, since this approach provides the fastest response for an individual program once it begins execution. However, if the goal of a multiprocessor system is to minimize *average* response time or to maximize throughput, then multiprogramming must be considered.

In this paper, a model of a simple multiprocessor system with a two-program workload is reviewed; the model is then applied to an Intel iPSC/2 hypercube multiprocessor with a workload consisting of parallel wavefront algorithms for solving triangular systems of linear equations. Throughputs predicted by the model are compared with throughputs obtained experimentally from an actual system. The results provide validation for the model and indicate that significant performance improvements for multiprocessor systems are possible through multiprogramming.

1. Introduction

Multiprocessor systems with large numbers of processors are becoming commonplace. This development has been motivated principally by the need to solve large, complex problems that are intractable using conventional uniprocessor systems. Over the past 30 years, computing power (speed, capacity) has doubled approximately every two years. Until recently, these increases were achieved through miniaturization without resorting to changing the basic uniprocessor organization. However, further miniaturization has become increasingly difficult (and expensive) as fundamental physical limitations are approached. Hence, computer architects have begun to increase computing power by coupling large numbers of processors together into a single system, thus allowing them to work simultaneously on a single problem.

It is unlikely that such multiprocessor systems would have been developed without the motivation provided by large, complex problems. Nevertheless, now that multiprocessor systems do exist, it is important to determine how best to use such systems.

The behavior of many parallel programs on a given multiprocessor system can be characterized, at least in part, by speedup curves similar to the one depicted in Figure 1. The speedup curve indicates how effectively a program is able to use additional processors allocated to it. For a typical parallel program, there is a maximum num-

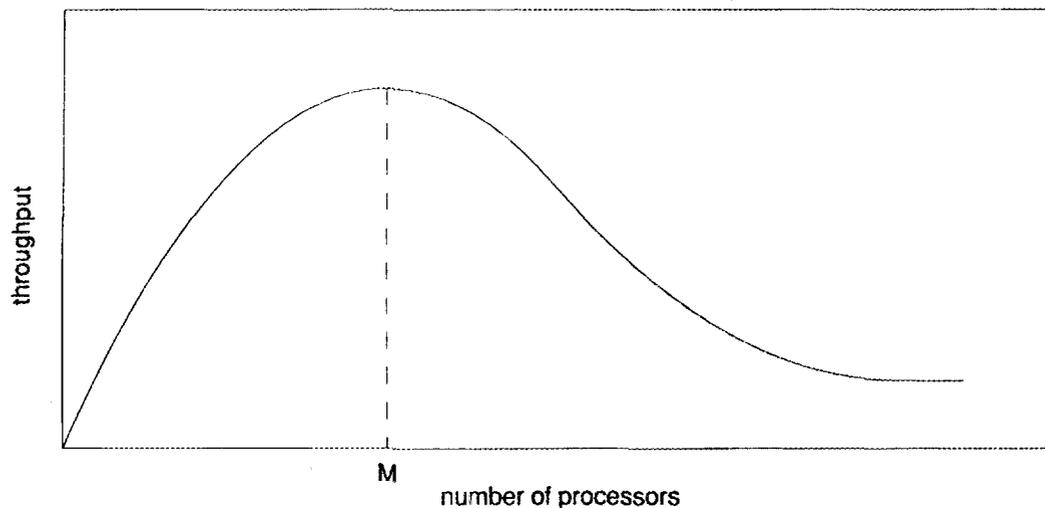


Figure 1: *Typical speedup curve for a parallel program. The horizontal axis indicates the number of processors allocated to the program; the vertical axis indicates throughput per unit time. M is the maximum number of processors the program can use effectively.*

ber of processors that can be used effectively. Allocating more than this number of processors will result in reduced throughput. The actual cause of this reduction in performance will vary from system to system, but may be the result of increased contention for the use of shared variables or of an increase in the volume of interprocessor communication.

If the goal of a system manager is to minimize response time for certain high-priority programs, system policies should be designed to allocate such a program the maximum number of processors it can use effectively. If this number exceeds the number of processors available, all available processors should be allocated. If, on the other hand, the system manager's goal is to maximize system throughput, i.e., to maximize the number of jobs completed per unit time, or to minimize *average* response time, a different processor allocation policy is needed. Note, as is illustrated in Figure 1, that a typical speedup curve initially increases rather rapidly but flattens as the point of optimal processor allocation is approached. Thus, the effectiveness with which a parallel program uses an additional processor depends on the number of processors already allocated to the program. Throughput tends to be increased more by assigning an additional processor to a program allocated few processors relative to its optimal number than by assigning it to a program allocated close to its optimal number of processors. Consequently, maximizing multiprocessor system throughput may require that two or more independent programs be executed simultaneously.

It is, therefore, worthwhile considering how a multiprocessor system can be multiprogrammed, that is, how it can be used to manage simultaneously two or more independent sets of related processes. An important question is how to partition the processors among the active programs. Processor partitioning can be implemented in a number of ways analogous to the partitioning of a computer system's primary memory. The simplest approach is to divide the processors into a number of fixed-size partitions. At any time, a processor partition executes at most a single program, i.e., a single collection of related processes. Each program is designed to run using a specific partition and can be executed only by that partition. A somewhat more restrictive approach may also be taken in which partitions are *private*; i.e., a partition and a program are paired, and as long as the program remains in the system, no other program may use its partition. An alternative approach is to allow programs to execute in any available fixed-size partition. Finally, partitions may be variable. As the processing needs of a workload change, the size and configuration of processor partitions may change in re-

sponse to these needs. In this paper, we consider modeling systems with two fixed-size, private partitions.

Section 2 contains a description of a simple model of a multiprocessor system with a two-program workload. In section 3, the model is adapted to an Intel iPSC/2 hypercube multiprocessor. Section 4 contains results from numerical experiments performed on the Intel iPSC/2 and compares throughputs obtained empirically to those predicted by the model.

2. The Model

In [2], a high-level model of a simple multiprocessor system whose processing elements (PEs) can be partitioned into two sets is described. The model assumes that program scheduling is done via a separate host processor, and that the parallel workload consists of two programs. The model description uses the following notation:

N —the total number of PEs in the multiprocessor.

n_i ($i = 1, 2$)—the number of PEs assigned to partition P_i . ($n_1 + n_2 = N$).

p_i ($i = 1, 2$)—the fraction of PEs assigned to partition P_i . ($p_i = n_i/N$.
 $p_1 + p_2 = 1$).

(A, B) —the current system state. “ A ” represents the location of program 1, either at the host (H) or in partition P_1 . “ B ” represents the location of program 2, either at the host (H) or in partition P_2 . The set of allowable states is $\{(H, H), (P_1, H), (H, P_2), (P_1, P_2)\}$.

$\mu_j(A, B)$ —the state-dependent service rate of device j , $j \in \{H, M\}$. “ H ” denotes the host; “ M ” denotes the multiprocessor. Service times are assumed to be exponentially distributed.

$S_M(A, B)$ —a state-dependent “interference factor” which represents degradation in multiprocessor performance due to contention for shared resources.

X —throughput of the multiprocessor system.

This model can be represented by the closed queuing network model of Figure 2.

The following assumptions are made with regard to the model:

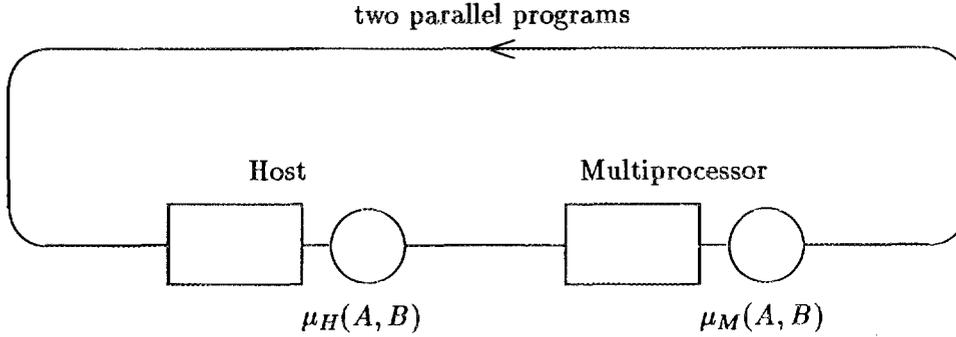


Figure 2: *Closed queuing network model of a multiprocessor system with its parallel workload.*

$\mu_H(A, B) = \lambda$. Service time at the host is state independent. The host treats both programs identically; i.e., the demands each program places on the host are statistically identical. Processor sharing at the host is assumed.

$\mu_M(P_1, H)$ and $\mu_M(H, P_2)$ are the “execution signatures” of the two parallel programs. Each term represents the processing rate of a program when it alone executes on the multiprocessor. Execution signatures depend not only on a program’s characteristics, but also on the specific architecture and on the size of the program’s partition. Expressions for execution signatures are suggested below. Execution signatures are analogous to speedup curves (see Figure 1).

$S_M(H, H) = S_M(P_1, H) = S_M(H, P_2) = 1$. When at most one program is executing on the multiprocessor, no performance degradation is experienced, since there is no contention for the shared resources of the multiprocessor.

$S_M(P_1, P_2) = s$. When both programs execute on the multiprocessor, the performance of each is degraded by a factor s ($0 \leq s \leq 1$), assumed to be a constant.

$\mu_M(P_1, P_2) = (\mu_M(P_1, H) + \mu_M(H, P_2)) \times S_M(P_1, P_2)$. When both programs are executing on the multiprocessor, the processing rate is the sum of the individual processing rates times the degradation factor $S_M(P_1, P_2)$.

Execution signatures of the form

$$\mu_M(P_1, H) = \frac{p_1}{D_{11}p_1 + D_{12}} \quad \text{and} \quad \mu_M(H, P_2) = \frac{p_2}{D_{21}p_2 + D_{22}} \quad (1)$$

have been suggested [2], where D_{i1} and D_{i2} are parameters dependent on characteristics of the specific architecture and of the specific program i . If $D_{i1} = 0$, program i experiences a linear speedup; if $D_{i2} = 0$, the execution rate of program i is constant, independent of the number of processors used.

The model is solved by applying standard techniques to the balance equations derived from the Markov diagram shown in Figure 3.

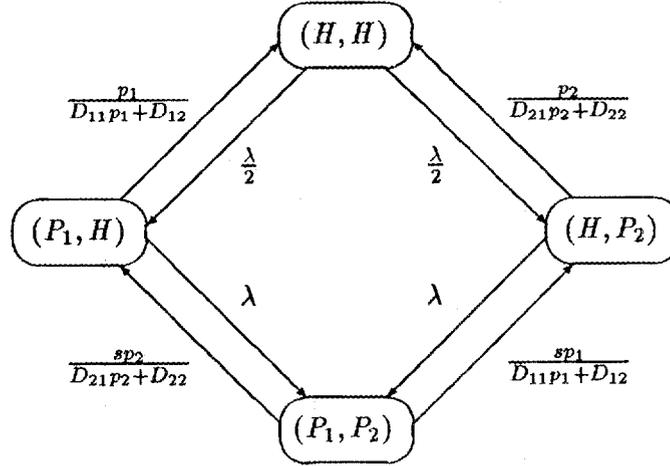


Figure 3: Markov diagram of the two-program multiprocessor system model. States correspond to locations of the programs; arcs are labeled with the transition flow rates.

Throughput as a function of the system parameters is

$$X(\lambda, D_{11}, D_{12}, D_{21}, D_{22}, p_1, p_2, s) = \frac{\lambda^2(p_2sD_{12} + p_1p_2sD_{11} + p_1p_2sD_{21} + p_1sD_{22}) + \lambda(2p_1p_2s)}{\lambda^2(p_2D_{12}D_{21} + p_1p_2D_{11}D_{21} + p_1D_{11}D_{22} + D_{12}D_{22}) + \lambda(p_2sD_{12} + p_1p_2sD_{21} + p_1p_2sD_{11} + p_1sD_{22}) + 2p_1p_2s} \quad (2)$$

3. Adaptation of the Model

3.1. The multiprocessor system

Experiments were performed on a 32-node Intel iPSC/2 multicomputer [4]. The iPSC/2 is a distributed-memory machine, connected in a binary n -cube configuration. Communication in the iPSC/2 is handled by a separate communications processor on each node, known as a “Direct-ConnectTM module.” The Direct-Connect module supports two modes of message-passing: short messages (100 bytes or less) are passed using a datagram technique; long messages (more than 100 bytes) are passed using a virtual circuit technique known as “worm-hole routing” [1]. Consequently, there is a jump in the cost of transmitting messages at the 100-byte boundary. The Direct-Connect module network requires only slightly longer to pass multi-hop messages than to pass single-hop messages. Consequently, the iPSC/2 appears to the user to be a fully-connected machine.

Adaptation of the model to the iPSC/2 is straightforward. There are only two major considerations: the manner in which the 32 nodes of the iPSC/2 are partitioned and the value used for the interference factor s . Preliminary experiments investigating possible ways to partition the 32 nodes of the iPSC/2 were conducted. The results of these experiments indicate that the critical factor in determining a program’s performance is the number of processors allocated to that program; the specific processors assigned to a program have little impact on its behavior, due to the “fully-connected” nature of the iPSC/2. Therefore, for convenience, program 1 is always run in a partition consisting of nodes numbered 0 through $i - 1$, and program 2 is always run in a partition consisting of nodes numbered i through 31 ($0 \leq i \leq 32$).

Experiments were also performed to determine the value of the interference factor s . The results of these experiments indicate that a program running in one partition has negligible impact on a program running in another partition. Consequently, s is set to one.

3.2. The parallel workload

The application programs used in these experiments are based on parallel wavefront algorithms [3] for solving triangular systems of linear equations. The wavefront idea produces both a column-oriented vector-sum algorithm and a row-oriented scalar-product algorithm. For these experiments, the row-oriented algorithm was chosen. In this algo-

rithm, messages consist of segments of the solution vector. Smaller segments increase the potential for parallelism but also increase the number of messages that must be passed. A performance trade-off exists, therefore, between potential parallelism and communication volume. Heath and Romine [3] have examined the issue of choosing an optimal segment size. The effect of varying the segment size (σ) is illustrated in Figures 4 and 5.

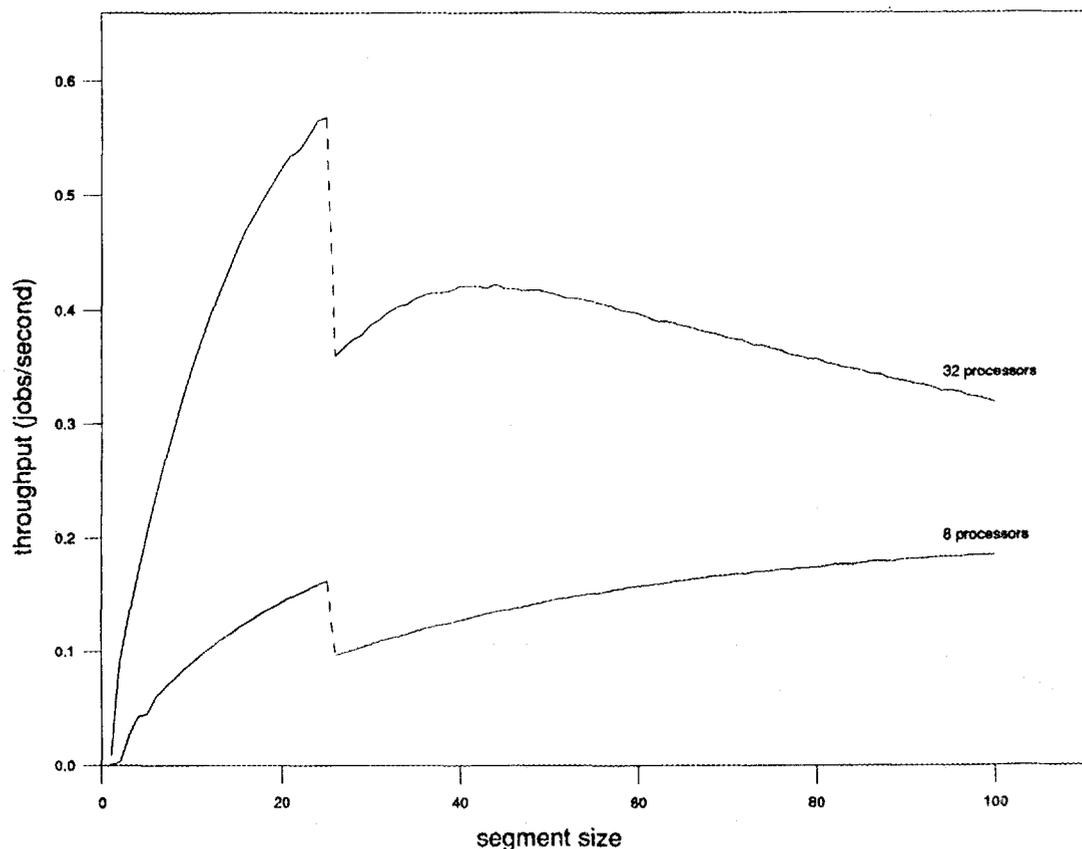


Figure 4: *Throughput as a function of segment size for 8 and for 32 processors solving a triangular system of order 1500 using the row-oriented wavefront algorithm. The dashed lines indicate the point at which the iPSC/2 switches message-passing modes from datagram service to virtual circuit service.*

Each program was written to run in two modes, a multiprocessor mode and a host mode. When in multiprocessor mode, a program solves a single triangular system in its private partition of the iPSC/2; when in host mode, a program executes a delay loop

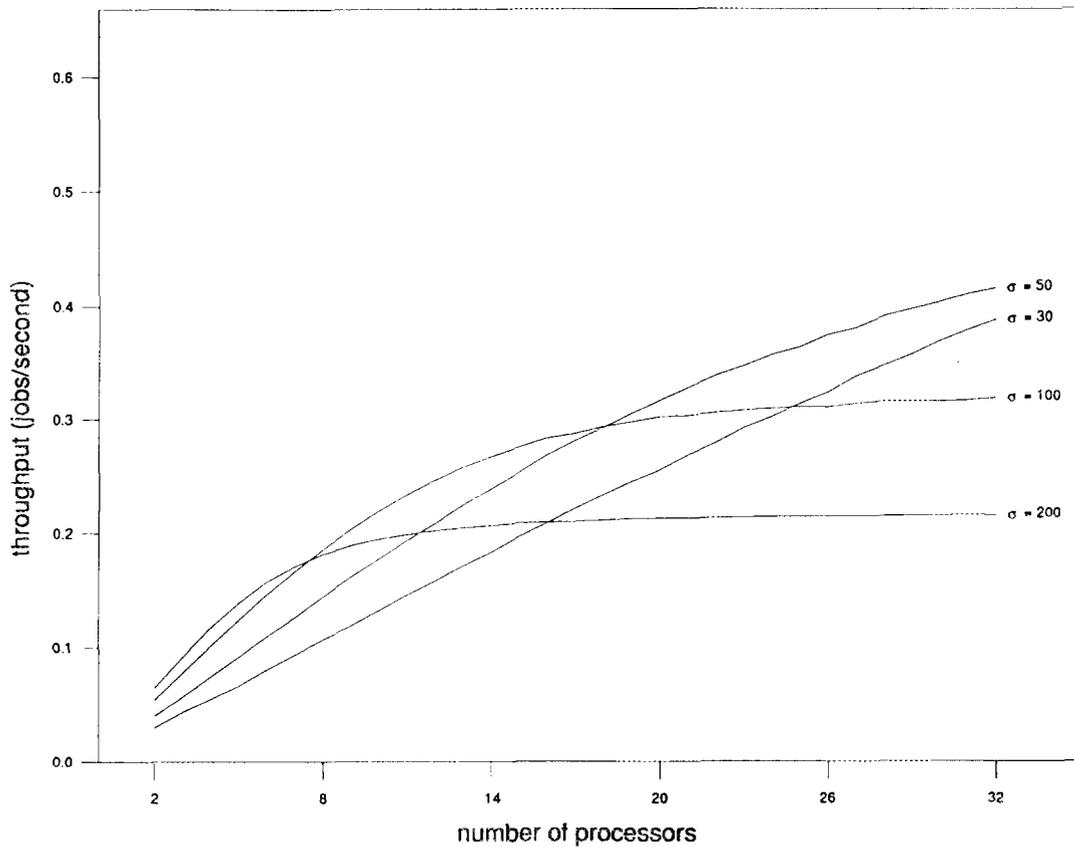


Figure 5: Execution signatures for the row-oriented wavefront algorithm on the iPSC/2 hypercube with a matrix of order 1500.

on the host machine (see Figure 6). The duration of a delay is randomly chosen from a predetermined probability distribution. The delay loop is used to simulate various host speeds, various amounts of required activity on the host, or various "loadings" placed on the multiprocessor by the host. A single experiment consists of the simultaneous execution of the two test programs for a predetermined period of time, during which each program cycles between its multiprocessor and host modes.

To study empirically the throughput of a partitioned system, two programs based on the row-oriented wavefront algorithm were selected. Program 1 solves a triangular system of order 1500 with a segment size of 10, and program 2 solves a triangular system of order 1800 with a segment size of 100. The empirically determined execution signatures of these two programs are illustrated in Figure 7. Corresponding execution

```

k = 222500.0 * tmin
  + 222500.0 * (tmax - tmin) * random() ;
j = 0 ;
for (i = 0; i < k; i++) j = (j+1)%1000 ;

```

Figure 6: Sample delay loop executed by a program in host mode. The routine `random()` generates a uniform random deviate between 0 and 1. The constant 222500.0 was empirically determined so that in this example, a delay of from `tmin` to `tmax` seconds would occur.

signatures of the form of equation (1) were determined by a non-linear least-squares fit. These execution signatures

$$\mu_M(P_1, H) = \frac{p_1}{0.1702p_1 + 2.6947} \quad \text{and} \quad \mu_M(H, P_2) = \frac{p_2}{2.4615p_2 + 1.1839} \quad (3)$$

are also illustrated in Figure 7.

Adapting the model (Section 2) to a specific architecture (Section 3.1) and to a specific workload (Section 3.2) results in parameter values listed in Table 1. The

<i>parameter</i>	<i>value</i>
D_{11}	0.1702
D_{12}	2.6947
D_{21}	2.4615
D_{22}	1.1839
s	1.0000

Table 1: Parameters for the theoretical model.

expression for throughput as a function of the partition size and host speed becomes

$$X(\lambda, p_1, p_2) = \frac{\lambda^2(2.6317p_1p_2 + 1.1839p_1 + 2.6947p_2) + \lambda(2p_1p_2)}{\lambda^2(0.4189p_1p_2 + 0.2015p_1 + 6.6330p_2 + 3.1903) + \lambda(2.6317p_1p_2 + 1.1839p_1 + 2.6947p_2) + 2p_1p_2s} \quad (4)$$

4. Numerical Experiments

The 32 nodes of the iPSC/2 were allocated so that program 1 was executed in partitions of size 0, 2, 4, ..., 32, with program 2 executing in the complementary partitions. Host

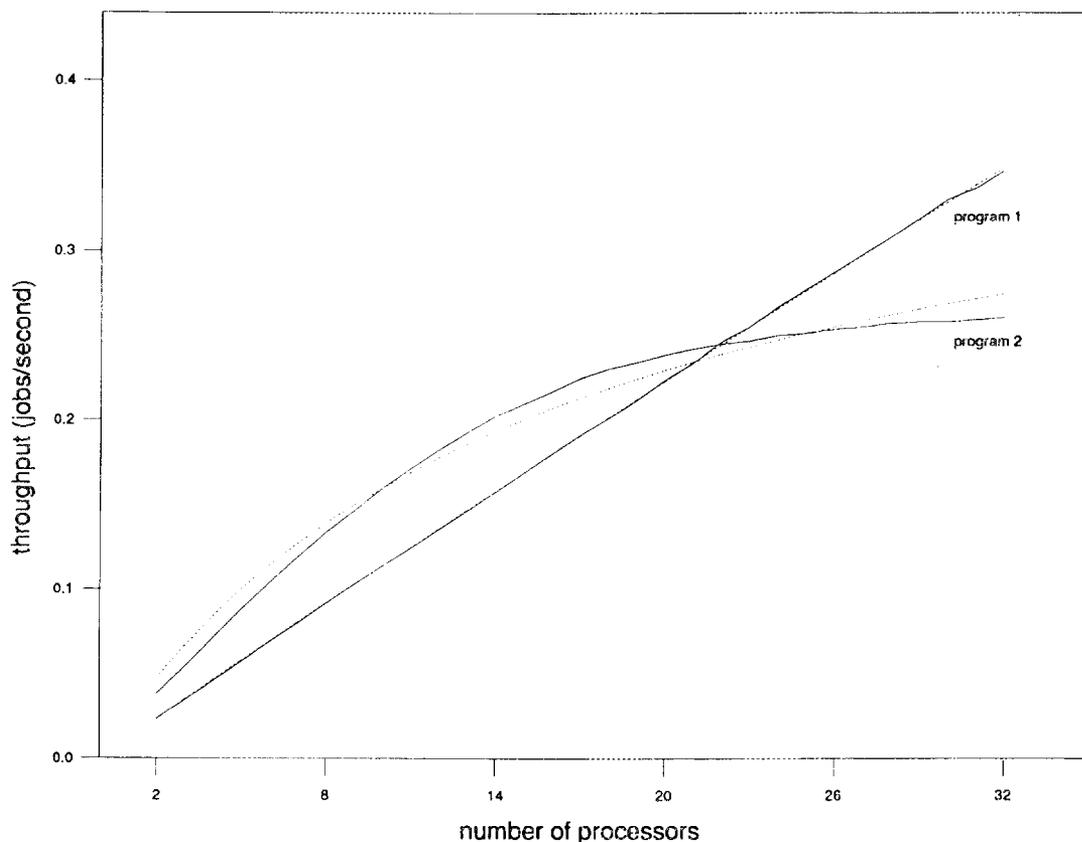


Figure 7: Execution signatures of the two test programs. The solid curves are empirically determined signatures; dotted curves are least-squares fits of signatures of the form of equation (1) to the empirical signatures.

delays were uniformly distributed between 2 and 4 seconds. The value for the speed of the host (λ) was 0.3268, the reciprocal of the sum of the average time spent in the delay loop (3 seconds) and a small measurable host-multiprocessor communication time (0.06 seconds). Each experiment was run for 10 minutes. System throughputs were both measured empirically and predicted using equation (4). The empirically determined and theoretically predicted throughputs are illustrated in Figure 8. Throughputs as high as 0.2134 jobs per second were observed. This throughput is 36.5% greater than the throughput for the same job mix in the corresponding uniprogramming environment, 0.1563 jobs per second.

The experimental and theoretical throughput values match quite well. The great-

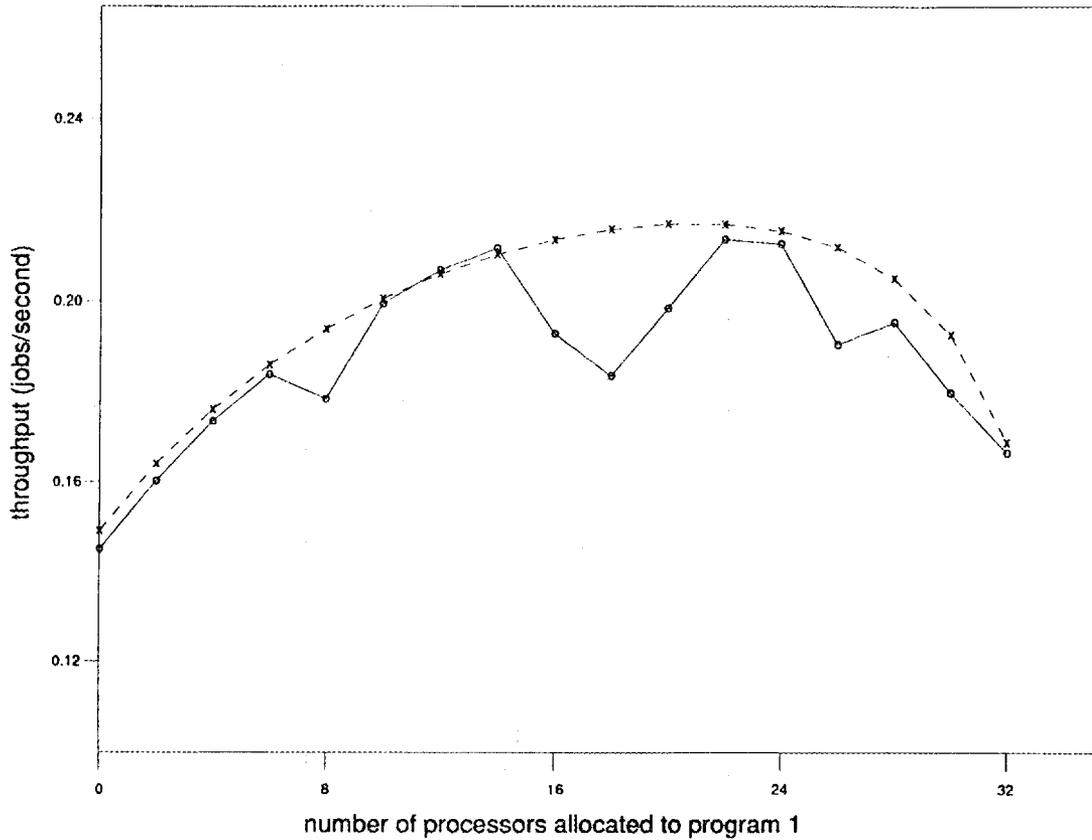


Figure 8: *Throughput as a function of the number of processors allocated to program 1. The solid line connects observed values. The dashed line connects values predicted by the theoretical model using the input values of Table 1.*

est discrepancy in throughput values occurs when 18 processors are assigned to program 1 and 14 processors are assigned to program 2. When allocated 18 processors, program 1 requires approximately 4.97 seconds to solve a triangular system; when allocated 14 processors, program 2 requires approximately 4.94 seconds to solve a triangular system. Thus, since each program places the same demands on the host, the average multiprocessor-host cycle times for both programs are nearly identical in length for this particular partitioning. When the program traces for this partitioning were examined, it was discovered that, despite the random delays incurred at the host, the programs tended to execute in lock-step, i.e., both programs were together at the host and then both programs were together at the multiprocessor. This tendency toward

synchronized behavior resulted in increased contention at the host and in a consequent decrease in throughput.

Other discrepancies between experimental and theoretical throughput values occur at the 8-24 and at the 26-6 partitionings. For these partitionings, the cycle time of one program was approximately twice the cycle time of the other. Again, a tendency toward synchronization was observed, with each multiprocessor phase of the slower program and every second multiprocessor phase of the faster program beginning at approximately the same time. Thus, every second host phase for the faster program was without contention; no host phase of the slower program was ever without contention.

Experiments were conducted to determine whether increasing the variance in the host delay, while maintaining a mean delay of 3.0 seconds, would counteract the tendency toward synchronization. Using a partition of 18 processors for program 1 and 14 processors for program 2, the minimum host delay was varied in 0.5 second intervals from 0.0 seconds to 3.0 seconds. Experimental results are summarized in Figure 9. As the variance in the host delay increases, the behavior of the two programs becomes more asynchronous. Consequently, contention at the host decreases, and throughput increases. With a host delay between 0 and 6 seconds, the observed throughput is less than 4% below the value predicted by the model.

Host-speed sensitivity experiments were also performed. Average delay at the host ($1/\lambda$) was varied from 0.5 to 10.0 seconds in increments of 0.5 seconds. All host-speed sensitivity experiments were conducted with 24 processors assigned to program 1 and 8 processors assigned to program 2. The empirically determined and theoretically predicted throughputs are illustrated in Figure 10. There is excellent agreement between experimental and model throughput values.

5. Summary

Although the development of most multiprocessor systems has been motivated by the need to execute single programs quickly, once an actual multiprocessor is in operation, system goals may change. For example, it may be important to complete the execution of a diverse workload as efficiently as possible, minimizing *average* response time or maximizing throughput. Pursuit of this goal leads to consideration of multiprogramming the multiprocessor system, which leads in turn to consideration of issues related to the partitioning of processors among the set of active programs.

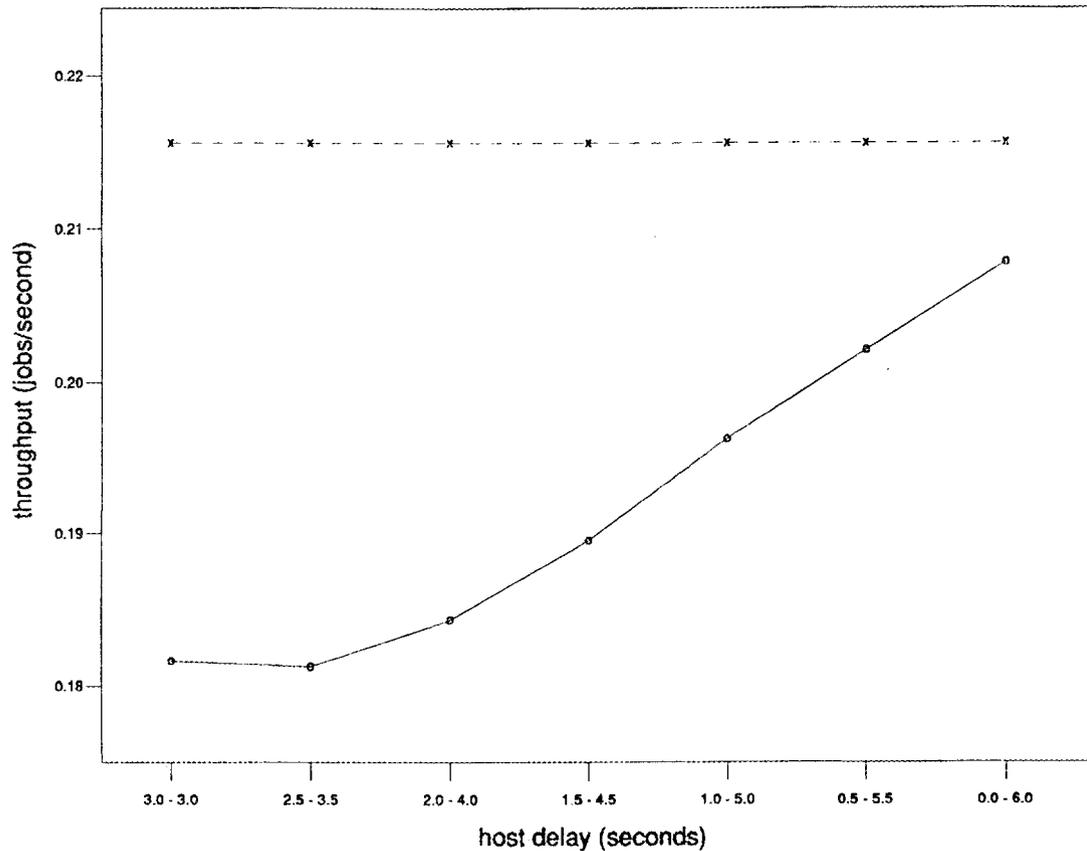


Figure 9: *Throughput as a function of variation in host delay. Program 1 was allocated 18 processors and program 2 was allocated 14 processors. Mean delay at the host was maintained at a constant 3.0 seconds. The dashed line represents throughput predicted by the theoretical model.*

In this paper, the adaptation of a model of a multiprocessor system with two fixed-size private partitions and a two-program workload to an actual system and an actual workload has been described. Validation experiments show a good match between theoretically predicted and empirically observed values. It is demonstrated that multiprogramming a distributed-memory multiprocessor can substantially improve performance. For the example selected, multiprocessor throughputs under multiprogramming that were as high as 0.2134 jobs per second were observed. For the same job mix in the corresponding uniprogramming environment, throughput would be 0.1563 jobs per second. Thus, an increase in throughput of 36.5% due to multiprogramming is observed.

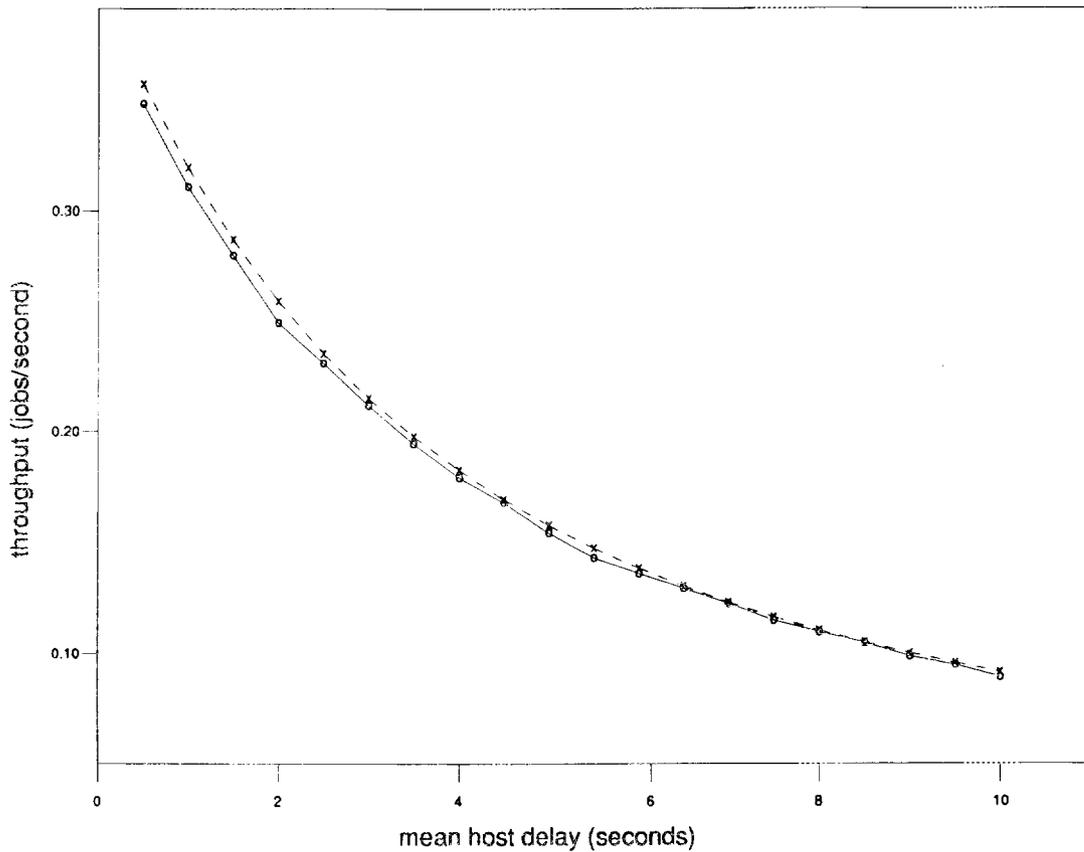


Figure 10: *Throughput as a function of the mean delay at the host. The solid line connects observed values. The dashed line connects values predicted by the theoretical model using the input values of Table 1.*

A number of extensions to this work are possible. The model of Section 2 can be generalized to account for more partitions and more programs. Increasing the number of partitions and programs will necessitate the use of heuristics in solving the model, since standard Markov analysis quickly becomes intractable. Other partitioning schemes, such as dynamic partitioning, can be incorporated into the model. The parallel workload can also be generalized. Different forms for the execution signatures can be considered, and the effects of different processing requirements at the host can be explored. For example, host requirements might vary from program to program, and the number of programs in the multiprogramming set may change as new programs enter the system and old programs complete and exit. All of these generalizations of

the model and of the workload should be validated experimentally. Because of the performance improvements that may be achieved through multiprogramming of multiprocessor systems, research in this area is important.

Acknowledgements

The authors wish to express their gratitude to Chuck Romine and to Pat Worley for helpful discussions. Their efforts have significantly improved the quality of this work.

References

- [1] William J. Dally and Charles L. Seitz, Deadlock-free message routing in multiprocessor interconnection networks, *IEEE Trans. on Comput.* **C-36**(5), May 1987, 547-553.
- [2] Lawrence W. Dowdy, On the partitioning of multiprocessor systems, *Technical Report*, Department of Computer Science, Vanderbilt University, Nashville, TN, 37235, March 1988.
- [3] Michael T. Heath and Charles H. Romine, Parallel solution of triangular systems on distributed-memory multiprocessors, *SIAM J. Sci. Statist. Comput.* **9**(3), May 1988, 558-588.
- [4] iPSC/2 User's Guide, Intel Scientific Computers, Beaverton, Oregon (1988).

ORNL/TM-11064

INTERNAL DISTRIBUTION

- | | |
|--------------------------|--------------------------------------|
| 1. B. R. Appleton | 26-30. R. C. Ward |
| 2. J. B. Drake | 31. P. H. Worley |
| 3. G. A. Geist | 32. A. Zucker |
| 4-5. R. F. Harbison | 33. J. J. Dorning (Consultant) |
| 6. M. T. Heath | 34. R. M. Haralick (Consultant) |
| 7-11. J. K. Ingersoll | 35. Central Research Library |
| 12-16. M. R. Leuze | 36. ORNL Patent Office |
| 17-21. F. C. Maienschein | 37. K-25 Plant Library |
| 22. E. G. Ng | 38. Y-12 Technical Library |
| 23. G. Ostrouchov | /Document Reference Station |
| 24. B. W. Peyton | 39. Laboratory Records - RC |
| 25. C. H. Romine | 40-41. Laboratory Records Department |

EXTERNAL DISTRIBUTION

42. Dr. Virgilio Almeida, Rua do Ouro, 958/401, 30210 Belo Horizonte, Brazil
43. Dr. Donald M. Austin, Office of Scientific Computing, Office of Energy Research, ER-7, Germantown Building, U.S. Department of Energy, Washington, DC 20545
44. Dr. Robert G. Babb, Department of Computer Science and Engineering, Oregon Graduate Center, 19600 N.W. Walker Road, Beaverton, OR 97006
45. Lawrence J. Baker, Exxon Production Research Company, P.O. Box 2189, Houston, TX 77252-2189
46. Dr. Jesse L. Barlow, Department of Computer Science, Pennsylvania State University, University Park, PA 16802
47. Dr. Chris Bischof, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
48. Prof. Åke Björck, Department of Mathematics, Linköping University, S-581 83 Linköping, Sweden

49. Dr. James C. Browne, Department of Computer Sciences, University of Texas, Austin, TX 78712
50. Dr. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307
51. Dr. Donald A. Calahan, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109
52. Dr. Tony Chan, Department of Mathematics, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90024
53. Dr. Jagdish Chandra, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709
54. Dr. Prasad Chintamaneni, Software Productivity Consortium, 1880 Campus Commons Drive N, Reston, VA 22071
55. Dr. Eleanor Chu, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
56. Prof. Tom Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853
57. Dr. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720
58. Prof. Andy Conn, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
59. Dr. Jane K. Cullum, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598
60. Dr. George Cybenko, Computer Science Department, University of Illinois, Urbana, IL 61801
61. Dr. George J. Davis, Department of Mathematics, Georgia State University, Atlanta, GA 30303
62. Dr. Jack J. Dongarra, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

63. Dr. Lawrence W. Dowdy, Department of Computer Science, Vanderbilt University, Nashville, TN 37235
64. Dr. Iain Duff, CSS Division, Harwell Laboratory, Didcot, Oxon OX11 0RA, England
65. Prof. Pat Eberlein, Department of Computer Science, SUNY/Buffalo, Buffalo, NY 14260
66. Dr. Stanley Eisenstat, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
67. Dr. Lars Elden, Department of Mathematics, Linköping University, S-581 83 Linköping, Sweden
68. Dr. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742
69. Dr. Albert M. Erisman, Boeing Computer Services, 565 Andover Park West, Tukwila, WA 98188
70. Dr. Peter Fenyés, General Motors Research Laboratory, Department 15, GM Technical Center, Warren, MI 48090
71. Prof. Patrick C. Fischer, Department of Computer Science, Vanderbilt University, Nashville, TN 37235
72. Prof. David Fisher, Department of Mathematics, Harvey Mudd College, Claremont, CA 91711
73. Dr. Geoffrey C. Fox, Booth Computing Center 158-79, California Institute of Technology, Pasadena, CA 91125
74. Dr. Paul O. Frederickson, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
75. Dr. Fred N. Fritsch, L-316, Mathematics and Statistics Division, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550
76. Dr. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650

77. Dr. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47405
78. Dr. David M. Gay, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
79. Dr. C. William Gear, Computer Science Department, University of Illinois, Urbana, IL 61801
80. Dr. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Road, Ottawa, Ontario, Canada K1A 0R8
81. Dr. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
82. Dr. John Gilbert, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304
83. Dr. Jacob D. Goldstein, The Analytic Sciences Corporation, 55 Walkers Brook Drive, Reading, MA 01867
84. Prof. Gene H. Golub, Department of Computer Science, Stanford University, Stanford, CA 94305
85. Dr. Karen D. Gordon, IDA/CSED, 1801 N. Beauregard Street, Alexandria, VA 22311
86. Dr. Joseph F. Grcar, Division 8331, Sandia National Laboratories, Livermore, CA 94550
87. Dr. Per Christian Hansen, Technical University of Denmark, Danish University Computing Center, UCI-C Lyngby, Building 305, DK-2800 Lyngby, Denmark
88. Dr. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P.O. Box 481, Houston, TX 77001
89. Dr. F. J. Helton, GA Technologies, P.O. Box 81608, San Diego, CA 92188
90. Prof. Dr. Ulrich Herzog, University of Erlangen-Nürnberg, Martensstrasse 3, 8520 Erlangen, FRG

91. Dr. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332
92. Dr. William H. Hooper, Mailstop W-425 (Data Networks), The MITRE Corporation, 7525 Colshire Drive, McLean, VA 22102
93. Dr. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550
94. Dr. Ilse Ipsen, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
95. Ms. Elizabeth Jessup, Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520
96. Prof. Barry Joe, Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1
97. Dr. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309
98. Dr. Bo Kagstrom, Institute of Information Processing, University of Umea, S-901 87 Umea, Sweden
99. Dr. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
100. Dr. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
101. Dr. Robert J. Kee, Applied Mathematics Division 8331, Sandia National Laboratories, Livermore, CA 94550
102. Ms. Virginia Klema, Statistics Center, E40-131, MIT, Cambridge, MA 02139
103. Dr. Richard Lau, Office of Naval Research, 1030 E. Green Street, Pasadena, CA 91101
104. Dr. Alan J. Laub, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106

105. Dr. Robert L. Launer, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709
106. Dr. Charles Lawson, MS 301-490, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109
107. Prof. Peter D. Lax, Director, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012
108. Dr. John G. Lewis, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346
109. Dr. Heather M. Liddell, Director, Center for Parallel Computing, Department of Computer Science and Statistics, Queen Mary College, University of London, Mile End Road, London E1 4NS, England
110. Dr. Joseph Liu, Department of Computer Science, York University, 4700 Keele Street, Downsview, Ontario, Canada M3J 1P3
111. Dr. Franklin Luk, Electrical Engineering Department, Cornell University, Ithaca, NY 14853
112. James G. Malone, General Motors Research Laboratories, Warren, MI 48090-9055
113. Dr. Thomas A. Manteuffel, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
114. Dr. Bernard McDonald, National Science Foundation, 1800 G Street, NW, Washington, DC 20550
115. Dr. Paul C. Messina, California Institute of Technology, Mail Code 158-79, Pasadena, CA 91125
116. Dr. Cleve Moler, Ardent Computers, 550 Del Ray Avenue, Sunnyvale, CA 94086
117. Dr. Brent Morris, National Security Agency, Ft. George G. Meade, MD 20755
118. Dr. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742
119. Maj. C. E. Oliver, Office of the Chief Scientist, Air Force Weapons Laboratory, Kirtland Air Force Base, Albuquerque, NM 87115

120. Dr. James M. Ortega, Department of Applied Mathematics, University of Virginia, Charlottesville, VA 22903
121. Prof. Chris Paige, Department of Computer Science, McGill University, 805 Sherbrooke Street W., Montreal, Quebec, Canada H3A 2K6
122. Dr. John F. Palmer, NCUBE Corporation, 915 E. LaVieve Lane, Tempe, AZ 85284
123. Prof. Roy P. Pargas, Department of Computer Science, Clemson University, Clemson, SC 29634-1906
124. Prof. Beresford N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720
125. Prof. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706
126. Dr. Alfredo Perez-Davila, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260
127. Dr. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650
128. Dr. Alex Pothén, Department of Computer Science, Pennsylvania State University, University Park, PA 16802
129. Prof. Daniel A. Reed, Department of Computer Science, Center for Supercomputing Research and Development, University of Illinois, Urbana, IL 61801
130. Dr. John K. Reid, CSS Division, Building 8.9, AERE Harwell, Didcot, Oxon, England OX11 0RA
131. Dr. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907
132. Dr. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore Laboratory, Livermore, CA 94550
133. Dr. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706

134. Dr. Ahmed H. Sameh, Computer Science Department, University of Illinois, Urbana, IL 61801
135. Dr. Michael Saunders, Systems Optimization Laboratory, Operations Research Department, Stanford University, Stanford, CA 94305
136. Dr. Larry Saxton, University of Regina, 53 Bobolink Bay, Regina, Saskatchewan, Canada S4S 4K2
137. Dr. Robert Schreiber, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180
138. Dr. Martin H. Schultz, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
139. Dr. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006
140. Dr. Lawrence F. Shampine, Mathematics Department, Southern Methodist University, Dallas, TX 75275
141. Dr. Kermit Sigmon, Department of Mathematics, University of Florida, Gainesville, FL 32611
142. Dr. Horst Simon, Mail Stop 258-5, NASA Ames Research Center, Moffett Field, CA 94035
143. Dr. Danny C. Sorensen, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439
144. Prof. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742
145. Dr. Kosmo D. Tatalias, Atlantic Aerospace Electronics Corporation, 6404 Ivy Lane, Suite 300, Greenbelt, MD 20770-1406
146. Prof. Satish K. Tripathi, Department of Computer Science, University of Maryland, College Park, MD 20742
147. Prof. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853

148. Dr. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665
149. Dr. Andrew B. White, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
150. Dr. Arthur Wouk, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709
151. Dr. Margaret Wright, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974
152. Dr. A. Yeremin, Department of Numerical Mathematics of the USSR Academy of Sciences, Gorki Street 11, Moscow, 103905, USSR
153. Prof. John Zahorjan, Department of Computer Science FR-35, University of Washington, Seattle, WA 98195
154. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P.O. Box 2001 Oak Ridge, TN 37831-8600
- 155-164. Office of Scientific & Technical Information, P.O. Box 62, Oak Ridge, TN 37831