ornl

ORNL/TM-11068

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

A Remote Host Facility for Intel Hypercubes

T. H. Dunigan

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

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

ORNL/TM-11068

Engineering Physics and Mathematics Division

Mathematical Sciences Section

A REMOTE HOST FACILITY FOR INTEL HYPERCUBES

T. H. Dunigan

Date Published - April 1989

The work was supported by the
Applied Mathematical Sciences subprogram
of the Office of Energy Research,
U. S. Department of Energy

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-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0299303 6

Table of Contents

Abstract 1

1. Introduction 1

2. User's Guide 2

3. Implementation 3

Acknowledgements 6

References 7

Appendix A: Sample host and node program 8

Appendix B: Remote Host Manual Page 9

A Remote Host Facility for Intel Hypercubes

T. H. Dunigan

Mathematical Sciences Section
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831

ABSTRACT

The structure and use of a remote host facility for controlling application programs on an Intel hypercube are described. The facility permits an alternate UNIX* host, such as a graphics workstation or supercomputer, connected by a TCP/IP network to the Intel cube manager processor to communicate with application programs running on the hypercube nodes. The facility supports both C and FORTRAN applications.

1. Introduction.

A hypercube is an ensemble of small computers interconnected in a communication network with the topology of an n-dimensional hypercube. Each processor, or node, has its own local memory and communication channels to n other nodes [5]. In addition, one or more nodes has a communication channel to a host processor, or cube manager. A hypercube application consists of one or more node programs and a host program. The host processor is used for building node programs, loading programs into the nodes, and controlling the application running on the nodes. The host and nodes work concurrently on an application and coordinate their computation by passing messages. The host program usually provides initial data values and reports final results.

A number of vendors, Ametek, Floating Point Systems, Intel, and Ncube, manufacture hypercube multiprocessors. This report describes a software facility to permit an alternate host to run the host portion of a hypercube application for Intel hypercubes. The Intel host processor, or cube manager, has limited I/O and graphics capability, so the remote host facility was developed to support alternate host processors that might have more advanced capabilities such as high-resolution graphics, windows, high-speed data collection, faster and higher capacity I/O, and possibly additional vector or parallel processing capability. The remote facility supports remote processors connected to the Intel host processor by Ethernet and the TCP/IP network software. (TCP/IP is a non-proprietary network package supported by most computer manufacturers.) The remote host application uses the same message-passing subroutines as an application running on the host processor. The facility has been used primarily on graphics workstations to

*UNIX is a trademark of AT&T.

provide animated graphics driven by the hypercube node processors. Applications that have used the remote host facility on graphics workstations include animated display of the solution of a 3-D wave equation [4], window-controlled display and traversal of the Mandelbrot set, contour plot display, and finite-element mesh display [1]. The remote host facility has been tested on VAX, Ardent, Sun, Sequent, Silicon Graphics Iris, IBM RT/AIX, and Tektronix processors. The C source files for the remote host library should be compatible with most UNIX systems supporting the Berkeley (BSD) TCP socket abstraction.

The next section describes how one builds a remote host application. The final section describes the implementation and performance of the remote host facility.

2. User's Guide.

A subroutine library, *librmthost.a*, on the remote host is required to build remote host applications, and the *rhdaemon* program on the Intel host processor is required to run the remote host application. The software distribution for the remote host facility includes source programs for the library and *rhdaemon* and a few sample programs. The user, or system manager, must compile and build the library on the remote host and compile and link *rhdaemon* on the cube manager. The distribution includes instructions for building and installing the library and daemon.

Table 1 illustrates the steps needed to build a remote host application in C. On the remote host, the remote host program, written in either C or FORTRAN, is compiled and linked with the remote host library *librmthost.a*. On the Intel host processor (cube manager), the user builds the node program, acquires a subcube, and loads the node programs [3]. (The remote host program can also acquire and load the cube with subroutines *getcube()* and *load()*.) Then *rhdaemon* is started on the cube manager, and the remote host program is started. When the remote host program terminates, *rhdaemon* will terminate. Appendix A contains a listing of a sample host and node program included in the software distribution.

Remote host	Cube manager
cc -o host host.c -lrmthost	*cc -o node node.c -node* *getcube* *load node* *rhdaemon*
host	

Table 1. Sample session.

For some remote hosts, one can copy an existing cube-manager host program to the remote host, rebuild it with the remote host library, and run it with no changes. However, one usually extends the host program to utilize the special features (graphics, I/O, etc.) of the remote host. The host application also may have to be modified if the representation of data on the remote host differs from the data representation on the Intel hypercube nodes. Floating point format, byte order, and character representation may differ. The remote host library provides some conversion subroutines so that hosts with

different data representations can communicate with the Intel hypercube(see Appendix B). For example, a Sun workstation has the same floating point format (IEEE 754) as the Intel node processors, but the byte-order is reversed. So a portion of a Sun remote host application that sends and receives a vector of single-precision (4-byte) floating point numbers would look like:

```
swap4(vector,DIM);
csend(TYPE2,vector,sizeof(vector),NODE,PID);
crecv(TYPE3,result,sizeof(result));
swap4(result,DIM);
```

The *DIM* elements of the single precision vector *vector* are swapped before sending the vector to the Intel cube. When the vector *result* is received by the Sun, the *DIM* elements are swapped to Sun byte-order. Subroutines are provided for swapping 2-, 4-, and 8-byte units for use with INTEGER*2, INTEGER*4 or REAL*4, and REAL*8 respectively. If the floating point format had differed, other subroutines would be called to convert from one format to the other. The library provides floating point conversion between single-precision, IEEE format floating point and VAX floating point.

3. Implementation.

The remote host facility consists of a library on the remote host and an intermediary process, *rhdaemon*, on the cube manager. When *rhdaemon* starts, it waits for a TCP/IP connection on a pre-defined port. When the remote host application is started, the TCP/IP connection is established, and *rhdaemon* processes each TCP/IP message from the remote host application.

The library on the remote host, *librmthost.a*, contains the Intel host-to-node communications routines (*csend, crecv*, etc.). Both the first-generation (iPSC/1) and second-generation (iPSC/2) subroutines are supported. When the remote host application first calls a subroutine in this library, the library attempts to establish a TCP/IP connection on the pre-defined port with *rhdaemon* on the cube manager. The library attempts to connect to the host *ipsc* unless the application provides another host name with the *rmthost()* function. The application may also specify the TCP port number to use when connecting to *rhdaemon* with the *rmtport()* function. The command *rhdaemon -p portnumber* can be used to specify the port number when invoking *rhdaemon*.

Once the TCP/IP connection is established between the remote host application and *rhdaemon*, then each Intel subroutine call issued by the remote host application is packaged as a TCP/IP message and sent to *rhdaemon*. *Rhdaemon* processes the message, usually issuing the equivalent subroutine on the cube manager, and, in most cases, sends a reply to the remote host application. The format of the messages between the remote host application and *rhdaemon* is illustrated in Table 2. The remote library routine sets the request type based on which subroutine is invoked, and *rhdaemon* uses the request type to select the corresponding cube manager subroutine. If *rhdaemon* is started in

request type
message id (e.g., for *msgwait()*)
node number
node process id
cube message type
maximum cube message length
cnode, cpid, ccode for *waitone()*
number of bytes received
length of buffer
buffer (cube message sent/received)

Table 2. Remote host message format.

debug mode (e.g., *rhdaemon -d 100*), then *rhdaemon* will produce on *stdout* on the cube manager a trace of interactions with the remote host application. This trace can be used for debugging.

The byte-order of the message header used by the remote host library must match that of the Intel cube manager, so the appropriate *makefile* must be selected when creating *librmthost.a*. In the *makefile* used to build *librmthost.a*, the C compiler directive *-DSUN* will reverse the byte-order for the 4-byte integers used in the message header.

The communication performance of the facility is less than that of a host program running directly on the cube manager. Data rates from a node to the cube manager range from 2 kilobytes/second to 910 kilobytes/second for message sizes ranging from 8 bytes to 8192 bytes. The data rate of the Ethernet interface of an Intel iPSC/2 cube manager in performing TCP/IP transfers ranges from 0.9 kilobytes/second to 81 kilobytes/second. The remote host facility transfers data with the nodes at rates from 0.08 kilobytes/second to 63 kilobytes/second, 10 to 20 times slower than the node-to-cube-manager data rate (see Figure 1). Nevertheless, the remote data transfer is fast enough to support animated graphics. (The data rates were gathered using an echo test [2], and a Sun 3/280 was used as the remote host.)

Intel now supports a similar facility in their second generation hypercube, iPSC/2 [3]. The Intel facility (version 2.4) supports only Sun workstations as remote hosts, but the facility provides additional remote services including remote cube commands and remote program building. The Intel facility is about a factor of two slower than our remote host facility — remote-host-to-node data rates range from 0.04 kilobytes/second to 40 kilobytes/second. (The Intel facility is slower because messages are passed through an additional process, *commser*, on the remote host.)

In the future, we plan to support multiple users of the remote host facility and modify *rhdaemon* to run continuously. To obtain a copy of the remote host facility or further information, send electronic mail to *dunigan@msr.epm.ornl.gov*.

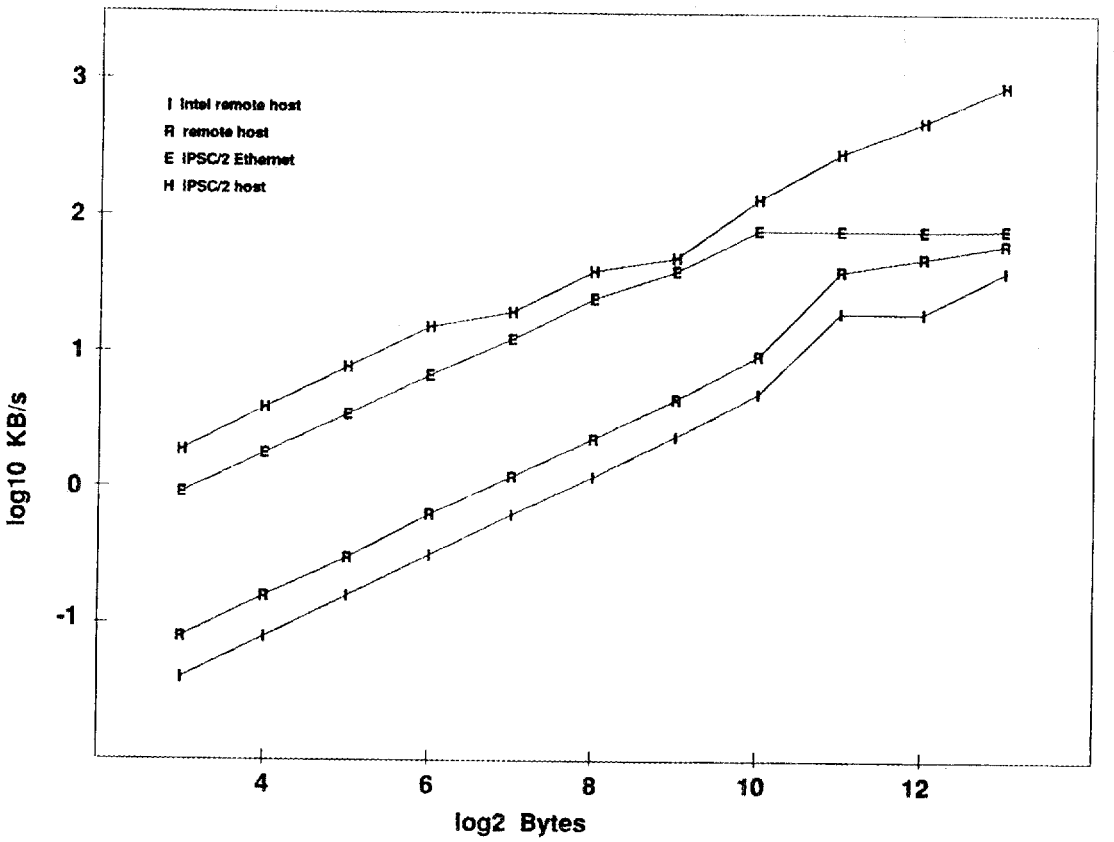

Figure 1. Host-to-node data rates for various message sizes.

Acknowledgements

The initial design and implementation of the remote host facility was begun when I visited the Christian Michelsen Institute, Bergen, Norway in the Summer of 1986. I am grateful to Paul Ferguson and Peter Bengst for their comments during the early phases of the project in Norway.

References

[1] J. B. Drake, D. L. Phelps, J. J. Roberts, *A simple graphics software system with applications to a distributed memory multiprocessor*, Tech. Rept. ORNL/TM-10540, Oak Ridge National Laboratory, Oak Ridge, TN (in preparation).

[2] T. H. Dunigan, *Performance of second generation hypercube*, Tech. Rept. ORNL/TM-10881, Oak Ridge National Laboratory, Oak Ridge, TN (1988).

[3] Intel, *iPSC/2 User's Guide*, Intel 311532-002, Portland, Oregon, March, 1988.

[4] C. Moler, P. Ross, *Solving the wave equation on the Intel iPSC*, Intel Marketing Literature, 1986.

[5] C. L. Seitz, *The cosmic cube*, Comm. ACM, **28** (1985), pp. 22-33.

Appendix A

Sample host and node program

```
/* iph.c  vector matrix(transpose) inner product using messages */
#include <stdio.h>
#define PID 0
#define DIM 4
int vector[DIM] = {2,3,1,4};
int matrix[DIM][DIM] = { 1,2,3,4, 2,3,1,0, 3,3,1,2, 4,3,2,1 };
int result[DIM];
main()
{
	/* Sun main task */
	int i, val, type, lth, node, pid;

	rmthost("ipsc");	/* select a cube manager host */
	getcube("thdring","d2m4",NULL,0);	/*request a 2D cube */
	setpid(PID);
	load("ipn",-1,PID);	/* load node program ipn */
	swap4(vector,DIM);
	for (i=0;i<DIM;i++){		/* start and send data to each node */
		swap4(matrix[i],DIM);
		csend(1,matrix[i],sizeof(matrix[i]),i,PID);
		csend(2,vector,sizeof(vector),i,PID);
	}
	i=DIM;
	while(i--){	/* wait for results */
		crecv(-1,&val,sizeof(int));
		swap4(&val,1);
		node = infonode();
		result[node] = val;
	}

	for(i=0;i<DIM;i++) printf(" %d",result[i]);
	printf(" (should be 27 14 24 23)0);
}



/* ipn.c  node program */
#define DIM 4
main()
{
	/* do inner product of two vectors */
	int v1[DIM], v2[DIM];
	int i, sum, me, node,pid,lth;

	me=mynode();
	crecv(1,v1,sizeof v1);
	crecv(2,v2,sizeof v2);
	lth = infocount();
	node = infonode();
	pid = infopid();
	sum=0;
	for(i=0;i< (lth/ sizeof(int));i++) sum += v1[i] * v2[i];
	csend(3,&sum,sizeof(int),node,pid);
}
```

Appendix B

Remote Host Manual Pages

RMTHOST(L) UNIX Programmer's Manual RMTHOST(L)

NAME

rmthost - remote host facility for the Intel iPSC hypercubes

SYNOPSIS

The *rmthost* facility enables the programmer to move his hypercube manager (host) application to another TCP/IP host machine. The remote host application still uses the same Intel subroutines (*load, csend*, etc.) to communicate with the hypercube node processors. The facility has been used to run graphics applications on a Sun, with the nodes providing the raw data to the Sun. Both the XENIX-iPSC/1 and System V-iPSC/2 Intel hypercube systems are supported.

DESCRIPTION

A library for C and FORTRAN on the remote host machine is provided that packages the hypercube messages into TCP/IP messages and communicates them to a daemon (*rhdaemon*) on the Intel 310 or 301. The daemon processes the messages executing the actual hypercube manager subroutines and relays the results back to the remote host.

On your remote host, build your host program with the *librmthost.a* library. *rlogin* to the hypercube manager; get the hypercube and load your node programs (your remote host program can also issue getcube() and load()); start *rhdaemon* (it is not privileged), usually in the background (&), then suspend your *rlogin* session (CTRL-Z) [or if you have multiple terminals or windows, then move to your host window] and start your host application. For the present release, when your host program terminates, *rhdaemon* will also terminate.

By default, the remote host will try to connect to the cube manager with a TCP/IP host name of *ipsc*. You may specify the name of the cube manager in your host application with the subroutine **rmthost(char *hostname)** and an alternate TCP port number with the subroutine **rmtport(int port)**. *Rhdaemon* may have optional arguments specifying a debug level and TCP port number. The form of the *rhdaemon* command is

rhdaemon -d debuglevel -p portnumber

The debug switch will log message requests from the remote host to *stdout* on the cube manager. A debug level of 100 gives a full trace of *rhdaemon* actions. (See the source code of *rhdaemon* for a more selective trace.)

FILES

The location of these files will depend on your installation.

bld	remote host build script for C
fbld	remote host build script for Fortran
librmthost.a	remote host interface library
rhdaemon	hypercube manager daemon

CAVEATS

Sun byte order differs from 80286/80386, so SUN host program must do swaps before *csend* and after *crecv*. The following functions are provided for the Sun programmer.

void swap2(short *vector,int elements);
void swap4(int *vector,int elements);
void swap8(double *vector,int elements);
These functions swap INTEGER*2, INTEGER*4 or REAL*4, and REAL*8 respectively (or their equivalents in C).

VAX floating point differs from the IEEE standard used on the hypercube nodes. The following single precision conversion functions are provided

void itovf(float *src, float *dst, int elements);
void vtoif(float *src, float *dst, int elements);

Default word sizes for a remote host and iPSC compilers may differ and can affect message length calculations. Declare remote host hypercube subroutine arguments as INTEGER (*int*) --e.g., node, pid, type, length.

The daemon exits when the host program terminates.

If your remote host program issues a *load()*, the node program must reside in the directory from which *rhdaemon* was started or a full path name must be specified in the *load()*.

The maximum host-to-node message size is 100,000 bytes.

AUTHOR

Tom Dunigan, Oak Ridge National Laboratory, dunigan@msr.epm.ornl.gov
A remote host facility for Intel hypercubes, Tech. Rept. ORNL/TM-11068, Oak Ridge National Laboratory.

INTERNAL DISTRIBUTION

1. B. R. Appleton
2. J. B. Drake
3-7. T. H. Dunigan
8. G. A. Geist
9-10. R. F. Harbison
11. M. T. Heath
12. M. R. Leuze
13-17. F. C. Maienschein
18. E. G. Ng
19. G. Ostrouchov
20-24 S. A. Raby
25. C. H. Romine
26-30. R. C. Ward
31. P. H. Worley
32. A. Zucker
33. J. J. Dorning (Consultant)
34. R. M. Haralick (Consultant)
35. Central Research Library
36. ORNL Patent Office
37. K-25 Plant Library
38. Y-12 Technical Library/ Document Reference Station
39. Laboratory Records - RC
40-41. Laboratory Records Department

EXTERNAL DISTRIBUTION

42. Dr. Donald M. Austin, Office of Scientific Computing, Office of Energy Research, ER-7, Germantown Building, U.S. Department of Energy, Washington, DC 20545

43. Dr. Robert G. Babb, Department of Computer Science and Engineering, Oregon Graduate Center, 19600 N.W. Walker Road, Beaverton, OR 97006

44. Lawrence J. Baker, Exxon Production Research Company, P.O. Box 2189, Houston, TX 77252-2189

45. Dr. Jesse L. Barlow, Department of Computer Science, Pennsylvania State University, University Park, PA 16802

46. Dr. Edward H. Barsis, Computer Science and Mathematics, P.O. Box 5800, Sandia National Laboratory, Albuquerque, NM 87185

47. Dr. Chris Bischof, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

48. Prof. Ake Bjorck, Department of Mathematics, Linkoping University, Linkoping 58183, 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 Office, P.O. Box 12211, Research Triangle Park, NC 27709

54. Dr. Eleanor Chu, Department of Computer Science, University of Waterloo, Ontario, Canada N2L 3G1

55. Prof. Tom Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853

56. Dr. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720

57. Prof. Andy Conn, Department of Combinatorics, and Optimization, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

58. Dr. Jane K. Cullum, IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598

59. Dr. George Cybenko, Computer Science Department, University of Illinois, Urbana, IL 61801

60. Dr. George J. Davis, Department of Mathematics, Georgia State University, Atlanta, GA 30303

61. Dr. Jack J. Dongarra, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

62. Dr. Iain Duff, CSS Division, Harwell Laboratory, Didcot, Oxon OX11 ORA, England

63. Prof. Pat Eberlein, Department of Computer Science, SUNY/Buffalo, Buffalo, NY 14260

64. Dr. Stanley Eisenstat, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

65. Dr. Lars Elden, Department of Mathematics, Linkoping University, 58183 Linkoping, Sweden

66. Dr. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742

67. Dr. Albert M. Erisman, Boeing Computer Services, 565 Andover Park West, Tukwila, WA 98188

68. Dr. Peter Fenyes, General Motors Research Laboratory, Department 15, GM Technical Center, Warren, MI 48090

69. Prof. David Fisher, Department of Mathematics, Harvey Mudd College, Claremont, CA 91711

70. Dr. Geoffrey C. Fox, Physics Departmtent, MS 356-48, California Institute of Technology, Pasadena, CA 91125

71. Dr. Paul O. Frederickson, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545

72. Dr. Fred N. Fritsch, L-300, Mathematics and Statistics Division, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550

73. Dr. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650

74. Dr. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47405

75. Dr. David M. Gay, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

76. Dr. C. William Gear, Computer Science Department, University of Illinois, Urbana, IL 61801

77. Dr. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Road, Ottawa, Ontario, Canada K1A 0R8

78. Dr. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

79. Dr. John Gilbert, Xerox Palo Alto Research Center, 3333 Coyote Hill Road Palo Alto, CA 94304

80. Dr. Jacob D. Goldstein, The Analytic Sciences Corporation, 55 Walkers Brook Drive, Reading, MA 01867

81. Prof. Gene H. Golub, Department of Computer Science, Stanford University, Stanford, CA 94305

82. Dr. Joseph F. Grcar, Division 8331, Sandia National Laboratories, Livermore, CA 94550

83. Dr. Per Christian Hansen, Copenhagen University Observatory, Øster Voldgade 3, DK-1350 Copenhagen K, Denmark

84. Dr. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P.O. Box 481, Houston, TX 77001

85. Dr. F. J. Helton, GA Technologies, P.O. Box 81608, San Diego, CA 92188

86. Dr. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332

87. Dr. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550

88. Dr. Ilse Ipsen, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

89. Ms. Elizabeth Jessup, Department of Computer Science, Yale University, P.O. Box 2158, Yale Station, New Haven, CT 06520

90. Prof. Barry Joe, Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada T6G 2H1

91. Dr. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309

92. Dr. Bo Kagstrom, Institute of Information Processing, University of Umea, 5-901 87 Umea, Sweden

93. Professor Malvyn Kalos, Courant Institute for Mathemtical Sciences, New York University, 251 Mercer Street, New York, NY 10012

94. Dr. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

95. Dr. Linda Kaufman, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

96. Dr. Robert J. Kee, Applied Mathematics Division 8331, Sandia National Laboratories, Livermore, CA 94550

97. Ms. Virginia Klema, Statistics Center, E40-131, MIT, Cambridge, MA 02139

98. Dr. Richard Lau, Office of Naval Research, 1030 E.Green Street, Pasadena, CA 91101

99. Dr. Alan J. Laub, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106

100. Dr. Robert L. Launer, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709

101. Dr. Charles Lawson, MS 301-490, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109

102. Prof. Peter D. Lax, Director, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012

103. Dr. John G. Lewis, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346

104. 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

105. Dr. Joseph Liu, Department of Computer Science, York University, 4700 Keele Street, Downsview, Ontario, Canada M3J 1P3

106. Dr. Franklin Luk, Electrical Engineering Department, Cornell University, Ithaca, NY 14853

107. James G. Malone, General Motors Research Laboratories, Warren, MI 48090-9055

108. Dr. Thomas A. Manteuffel, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545

109. Dr. Bernard McDonald, National Science Foundation, 1800 G Street, NW, Washington, DC 20550

110. Dr. Paul C. Messina, California Institute of Technology, Mail Code 158-79, Pasadena, CA 91125

111. Dr. Cleve Moler, Ardent Computers, 550 Del Ray Avenue, Sunnyvale, CA 94086

112. Dr. Brent Morris, National Security Agency, Ft. George G. Meade, MD 20755

113. Dr. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742

114. Maj. C. E. Oliver, Office of the Chief Scientist, Air Force Weapons Laboratory, Kirtland Air Force Base, Albuquerque, NM 87115

115. Dr. James M. Ortega, Department of Applied Mathematics, University of Virginia, Charlottesville, VA 22903

116. Prof. Chris Paige, Department of Computer Science, McGill University, 805 Sherbrooke Street W., Montreal, Quebec, Canada H3A 2K6

117. Dr. John F. Palmer, NCUBE Corporation, 915 E. LaVieve Lane, Tempe, AZ 85284

118. Prof. Roy P. Pargas, Department of Computer Science, Clemson University, Clemson, SC 29634-1906

119. Prof. Beresford N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720

120. Prof. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706

121. Dr. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650

122. Dr. Alex Pothen, Department of Computer Science, Pennsylvania State University, University Park, PA 16802

123. Dr. John K. Reid, CSS Division, Building 8.9, AERE Harwell, Didcot, Oxon, England OX11 0RA

124. Professor Werner C. Rheinboldt, Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA 15260

125. Dr. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907

126. Dr. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore Laboratory, Livermore, CA 94550

127. Dr. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706

128. Dr. Ahmed H. Sameh, Center for Supercomputing R&D, 1384 W. Springfield Avenue, University of Illinois, Urbana, IL 61801

129. Dr. Michael Saunders, Systems Optimization Laboratory, Operations Research Department, Stanford University, Stanford, CA 94305

130. Dr. Robert Schreiber, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180

131. Dr. Martin H. Schultz, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520

132. Dr. David S. Scott, Intel Scientific Computers, 15201 N.W.Greenbrier Parkway, Beaverton, OR 97006

133. Dr. Lawrence F. Shampine, Mathematics Department, Southern Methodist University, Dallas, TX 75275

134. Dr. Kermit Sigmon, Department of Mathematics, University of Florida, Gainesville, FL 32611

135. Dr. Danny C. Sorensen, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

136. Prof. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742

137. Dr. Kosmo D. Tatalias, Atlantic Aerospace Electronics Corporation, 6404 Ivy Lane, Suite 300, Breenbelt, MD 20770-1406

138. Prof. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853

139. Dr. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665

140. Dr. Andrew B. White, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545

141. Dr. Arthur Wouk, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709

142. Dr. Margaret Wright, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974

143. Dr. A. Yeremin, Department of Numerical Mathematics of the USSR Academy of Sciences, Gorki Street 11, Moscow, 103905, USSR

144. 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

145-154. Office of Scientific & Technical Information, P.O. Box 62, Oak Ridge, TN 37831