References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK: A portable linear algebra library for high-performance computers. In *Supercomputer 90*, New York, NY, 1990. IEEE Press.

[2] E. Anderson, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, S. Hammarling, and W. Kahan. Prospectus for an extension to lapack: A portable linear algebra library for high-performance computers. Computer Science Dept. Technical Report CS-90-118, University of Tennessee, Knoxville, TN, November 1990. (LAPACK Working Note #26).

[3] D. Dodson and J. Lewis. Issues relating to extension of the basic linear algebra subprograms. *ACM SIGNUM Newsletter*, 20(1):2–18, 1985.

[4] G. A. Geist, M. T. Heath, B. W. Peyton, and P. H. Worley. Picl: A portable instrumented communication library, c reference manual. Technical Report ORNL/TM-11130, Oak Ridge National Laboratory, Oak Ridge, TN, July 1990.

[5] G. A. Geist, M. T. Heath, B. W. Peyton, and P. H. Worley. A users' guide to picl: a portable instrumented communication library. Technical Report ORNL/TM-11616, Oak Ridge National Laboratory, Oak Ridge, TN, September 1990.

[6] A. Skjellum. Zipcode: A portable communication layer for high performance multicomputing - practice and experience. Numerical Mathematics Group UCRL Report JC-106725, Lawrence Livermore National Laboratory, Livermore, CA, March 1991.

Roldan Pozo
Center for Applied Parallel Processing
and Department of Computer Science
University of Colorado
Boulder, CO 80309
303-494-6072,roldan@cs.colorado.edu

Giuseppe Radicati
European Center for Scientific
and Engineering Computing (ECSEC)
IBM Italia spa
00147 Roma, via Giorgione 159, ITALY
39/6/5486.4527, radicati@romesc.vnet.ibm.com

Mikael Rannar
Institute of Information Processing
University of Umea
S-901-87 Umea, Sweden
46-90-166330, mr@cs.umu.se

Chuck Romine
Mathematical Sciences Section
Oak Ridge National Laboratory
Bldg. 9207-A
Oak Ridge, TN 37831
615-574-3148, romine@msr.epm.ornl.gov

Daniel Ruiz
CERFACS
42 Ave. G. Coriolis
31 057 Toulouse CEDEX
France
33-61-07-96-96 p9960, ruiz@orion.cerfacs.fr

Tony Skjellum
Lawrence Livermore National Laboratory
Numerical Mathematics Group
7000 East Avenue, L-316
Livermore, CA 94550
415-422-1161 FAX:415-423-2993, tony@helios.llnl.gov

Danny Sorensen
Dept. of Mathematical Sciences
P.O. Box 1892
Rice University
Houston, TX 77251
713-285-5193, sorensen@rice.edu

Chau-Wen Tseng
Box 1892
Department of Computer Science
Rice University
Houston, TX 77251
tseng@rice.edu

Phuong Vu
Cray Research, Inc.
655F Lone Oak Drive
Eagan, MN 55121
612-683-5615, pav@teak.cray.com

Robert van de Geijn
Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301
615-974-8295, rvdg@cs.utk.edu

Al Geist
Oak Ridge National Laboratory
P. O. Box Y
Building 9207A
Oak Ridge, TN 37830
615-574-3151, geist@msr.epm.ornl.gov

Fred Gustavson
Room 33-260
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598
914-945-1980, gustav@watson.ibm.com

Richard J. Hanson
IMSL, Inc.
2500 Permian Tower
2500 City West Blvd.
Houston, TX 77042-3020
(713) 782-6060 imsl!hanson@uunet.uu.net

Michael Heath
305 Talbot Laboratory
University of Illinois
104 South Wright Street
Urbana, IL 61801-2932
217-244-6915, heath@csrd.uiuc.edu

Adolfy Hoisie
Theory Center
Cornell University
631 E & TC Building
Ithaca, New York 14857
607-254-8786 AOH@CORNELLF.tn.cornell.edu

Lennart Johnsson
Thinking Machines Corp.
245 First Street
Cambridge MA 02142-1214
(617) 876-1111 johnsson@think.com

Paul Kinney
FPS Computing
3601 S.W. Murray Blvd.
Beaverton OR 97005
503 641-3151 ext. 3602, ptk@fps.com

Ed Kushner
Intel Scientific Computers
15201 NW. Greenbrier Pkwy.
Beaverton, Oregon 97006
503-629-7658, kushner@ssd.intel.com

Richard Lehoucq
IMSL, Inc.
2500 Permian Tower
2500 City West Boulevard
Houston, TX 77042
imsl!lehoucq@uunet.uu.net

Tim Leite
IMSL, Inc.
2500 Permian Tower
2500 City West Blvd.
Houston, TX 77042-3020
(713) 782-6060, imsl!leite@uunet.uu.net

Mike Leuze
Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge, TN 37830
615-574-3146, leuze@msr.epm.ornl.gov

Berna Massingill
Caltech
256-80
Pasadena, CA 91125
818-356-4603, berna@vlsi.cs.caltech.edu

Oliver McBryan
Department of Computer Science
University of Colorado at Boulder
Campus Box 425
Boulder, CO 80309-0425
303-665-0544, mcbryan@boulder.Colorado.EDU

Mo Mu
Computer Science Department
Purdue University
West Lafayette, Indiana 47907
317-494-0894, mu@cs.purdue.edu

Jim Patterson
Boeing Computer Services Company
Mail Stop 7L-21
P.O. Box 24346
Seattle, Washington 98124-0346
206-865-3510, pattersn@tritonia.boeing.com

List of Attendees

Ed Anderson
Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301
615-974-8295, eanderso@cs.utk.edu

Vasanth Bala
IBM T. J. Watson Research Ctr
P. O. Box 218
Yorktown Heights, NY 10598
vasb@ibm.com

Bridget Beattie
Dept. of Stats. and Comp. Math.
University of Liverpool
P.O. Box 147
Liverpool L69 3BX
England
011-20-525-6279, 011-44-051-794-4935

Martin Berzins
School of Computer Studies
Leeds University
Leeds LS2 9TJ
ENGLAND
+44 532 335449, martin@dcs.leeds.ac.uk

Ralph Brickner
C-3, MS-13265
Los Alamos National Lab
Los Alamos, NM 87545
rsb@lanl.gov

Bill Celmaster
Bolt, Beranek & Newman
10 Fawcett Street
Cambridge MA 02138
(617) 873-2837, wcelmast@bbn.com

Kuo-Ning Chiang
MacNeil-Schwendler
815 Colorado Blvd.
Los Angeles, CA 90041
213-258-9111

Krister Dackland
Institute of Information Processing
University of Umea
S-901-87 Umea, Sweden
46 90 166985, dacke@cs.umu.se

Michel Dayde
CERFACS
42 Ave G. Coriolis
31057 Toulouse Cedex
France
33 61-07-96-53, dayde@cerfacs.fr

Jim Demmel
Computer Science Div.
University of California
Berkeley, CA 94720
415-643-5386, demmel@arpa.berkeley.edu

Jack Dongarra
Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301
615-974-8295, dongarra@cs.utk.edu

Alan Edelman
Mathematics Department
University of California
Berkeley, CA 94720
415-528-5263, edelman@math.berkeley.edu

Erik Elmroth
Institute of Information Processing
University of Umea
S-901-87 Umea, Sweden
46-90-166986, elmroth@cs.umu.se

Vince Fernando
NAG Limited
Wilkinson House
Jordan Hill Road
Oxford, OX2 8DR
44-865-511245, nagkvf@vax.oxford.ac.uk

Geoffrey Fox
NPAC, 3-131 CST
111 College Place
Syracuse University
Syracuse, NY 13244-4100
gcf@nova.npac.syr.edu

Kyle Gallivan
C.S.R.D.
University of Illinois
305 Talbot Lab.
Champaign, IL 61820
217-244-0282, gallivan@csrd.uiuc.edu

`DIAG` is used by the trapezoidal matrix routines to specify whether or not the matrix is unit trapezoidal, as follows:

Value	Meaning
'`U`'	Unit trapezoidal
'`N`'	Non-unit trapezoidal

When `DIAG` is supplied as '`U`' the diagonal elements are not referenced or overwritten.

- `N` (Input) Length of vectors.
- `X` (Input) Vector of elements to be summed.
- `Y` (Output) Vector of results of summation.
- `IDEST` (Input) Index of process where result is to be accumulated.
- `MSGID` (Input) Identifier for this message.

Current BLACS routines:

Data Communication Routines:

```
_GESD ( M, N, A, LDA, IDEST, MSGID )
_GERV ( M, N, A, LDA, ISRC, MSGID )
_GEBC ( M, N, A, LDA, ISRC, MSGID )

_TRSD ( UPLO, DIAG, M, N, A, LDA, IDEST, MSGID )
_TRRV ( UPLO, DIAG, M, N, A, LDA, ISRC, MSGID )
_TRBC ( UPLO, DIAG, M, N, A, LDA, ISRC, MSGID )
```

Global Operations:

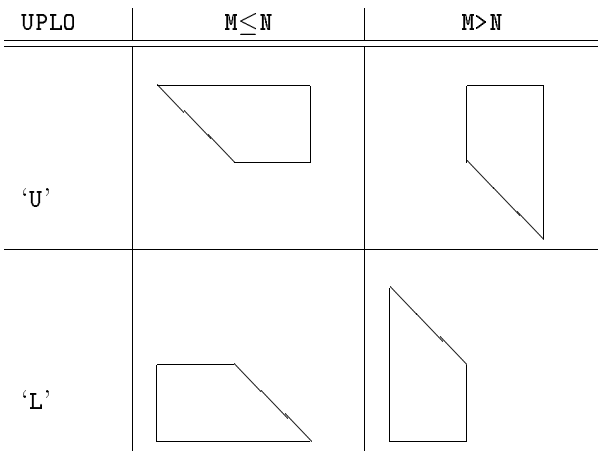
```
_GMAX ( N, X, Y, IY, IDEST, MSGID )
_GMIN ( N, X, Y, IY, IDEST, MSGID )
_GSUM ( N, X, Y, IDEST, MSGID )
```

Trapezoidal matrices:

`UPLO` is used by the Hermitian, symmetric, and trapezoidal matrix routines to specify whether the upper or lower trapezoid is being referenced as follows:

Value	Meaning
'U'	Upper trapezoid
'L'	Lower trapezoid

The shape of the trapezoid to be sent is determined by `M` and `N`:



Position Function

_ type of data to be communicated:
(Integer, Single or Double precision, etc.)

ZZZ function of the routine:
(MAXimum, MINimum, or SUM)

`_GMAX(N, X, Y, IY, IDEST, MSGID)`

Compare values of elements and produce the maximum (in absolute value?) (elementwise), i.e., for each `I=1,N`,

$$\texttt{Y(I)} = \max_{\texttt{allprocs}} \texttt{X(I)}.$$

Parameters:

• `N`	(Input)	Length of vectors.
• `X`	(Input)	Vector of elements to be compared.
• `Y`	(Output)	Vector of results of comparison.
• `IY`	(Output)	Vector of indices of the processes that gave the maximum values.
• `IDEST`	(Input)	Index of process where result is to be accumulated.
• `MSGID`	(Input)	Identifier for this message.

`_GMIN(N, X, Y, IY, IDEST, MSGID)`

Compare values of elements and produce the minimum (in absolute value?) (elementwise), i.e., for each `I=1,N`,

$$\texttt{Y(I)} = \min_{\texttt{allprocs}} \texttt{X(I)}.$$

Parameters:

• `N`	(Input)	Length of vectors.
• `X`	(Input)	Vector of elements to be compared.
• `Y`	(Output)	Vector of results of comparison.
• `IY`	(Output)	Vector of indices of the processes that gave the minimum values.
• `IDEST`	(Input)	Index of process where result is to be accumulated.
• `MSGID`	(Input)	Identifier for this message.

`_GSUM(N, X, Y, IDEST, MSGID)`

Sum the vector of elements.

Parameters:

- IDEST (Input) Index of destination process.
- MSGID (Input) Identifier for this message.

```
_GERV(M, N, A, LDA, ISRC, MSGID)
```

Receive array A from process ISRC.

```
_TRRV(UPLO, DIAG, M, N, A, LDA, ISRC, MSGID)
```

Receive trapezoidal array A from process ISRC.

Parameters:

- UPLO (Output) Specifies if the matrix is to be stored as a Lower or Upper trapezoidal matrix.
- DIAG (Output) Specifies whether or not the matrix is to be unit trapezoidal.
- M (Output) Row dimension of A.
- N (Output) Column dimension of A.
- A (Output) Array into which data is to be received.
- LDA (Input) Leading dimension of A.
- ISRC (Output) Index of source process.
- MSGID (Input) Identifier for this message.

```
_GEBC(M, N, A, LDA, ISRC, MSGID)
```

Send/receive matrix A to all processes from process with index ISRC.

```
_TRBC(UPLO, DIAG, M, N, A, LDA, ISRC, MSGID)
```

Send/receive trapezoidal matrix A to all processes from process with index ISRC.

Parameters:

- UPLO (Input/Output) Specifies if the matrix is stored as a Lower or Upper trapezoidal matrix.
- DIAG (Input/Output) Specifies whether or not the matrix is unit trapezoidal.
- M (Input/Output) Row dimension of A.
- N (Input/Output) Column dimension of A.
- A (Input/Output) Array of data to be sent/array into which data is to be received.
- LDA (Input) Leading dimension of A. LDA>=M
- ISRC (Input/Output) Index of source process.
- MSGID (Input) Identifier for this message.

Global Operators:

General form: _GZZZ

Appendix: "White Paper" Describing the BLACS

The BLACS are a set of communication routines that complement the level 1, 2 and 3 BLAS. The BLACS should provide tools for the implementation of numerical algorithms in linear algebra on distributed memory MIMD computers. The proposed routines cover the following operations:

Point-to-point send/receive
Broadcast
Probe
Global maximum, minimum and summation.

The naming convention is similar to the Level 2 and 3 BLAS.

Naming convensions: All names of BLACS subroutines are specified by at most six letters and are of the form `_XXYY` or `_GZZZ`.

Data Communication Routines:

General form: `_XXYY`

Position Function

`_` type of data to be communicated:
(Integer, Single or Double precision, etc.)

`XX` data structure:
(GEneral or TRapezoidal)

`YY` function of the routine:
(SenD, ReceiVe, or BroadCast)

```
_GESD(M, N, A, LDA, IDEST, MSGID)
```

Send matrix `A` to process `IDEST`.

```
_TRSD(UPLO, DIAG, M, N, A, LDA, IDEST, MSGID)
```

Send trapezoidal matrix `A` to process `IDEST`.

Parameters:

- `UPLO` (Input) Specifies if the matrix is stored as a Lower or Upper trapezoidal matrix.
- `DIAG` (Input) Specifies whether or not the matrix is unit trapezoidal.
- `M` (Input) Row dimension of `A`.
- `N` (Input) Column dimension of `A`.
- `A` (Input) Array of data to be sent.
- `LDA` (Input) Leading dimension of `A`. `LDA>=M`

3. I is fixed; N, $p = O(N^{1/2})$ and $M = O(N^{1/2})$ are increasing. Here we increase the number of processors and memory size proportionally to the matrix dimension n. This corresponds to laying out a fixed number of columns per memory.
4. N and M are fixed; I and $p = O(I)$ are increasing. Here we add more simultaneous independent problems as the number of processors grows.

- How portable should the codes be?

  Is it acceptable to have one set of codes for SIMD and one for MIMD machines? One set of codes for machines requiring explicit message passing and another for codes not requiring this? LAPACK is quite large, and we would like to avoid writing and maintaining different versions. Can we use facilities like the C preprocessors or other macro packages to make porting easier, or do we want to maintain the current "pure source code portability" model? We have already sacrificed this to some extent by assuming the BLAS are machine dependent, as well as certain tuning parameters, so perhaps we should take advantage of limited machine dependency elsewhere. Other issues include noninterference with the rest of the application (e.g., not assuming all processors are available to be dedicated to this problem, assuming multiprogramming of the nodes, not using global names for message IDs which may overlap with others).

- How soon should the BLACS and LAPACK code be developed?

  LAPACK is being designed now for distributed memory machines and needs a package like the BLACS. There is a great deal of activity both in designing new architectures as well as software systems (as evidenced by the presentations at this meeting). So while it might seem reasonable to wait and see which systems come out on top, we cannot afford to wait too long because people need to use this code. Furthermore, our experience in writing the code will help decide which systems should come out on top. We should try to avoid features in the BLACS that prevent them from being emulated by other lower level software.

It was agreed that the following steps be taken:

1. Obtain a reasonable consensus among those people actively interested in the area. This workshop is one step in the process.
2. It is too early to focus on a proposal at this time.
3. Another workshop was needed to bring out ideas. Perhaps in the Fall would be the right time frame. The Cornell people have volunteered to host the next meeting.
4. Examples of LU decomposition (and perhaps other routines) using the proposed BLACS as well as alternate systems would be desirable, both to compare for ease of expression and performance.

6 Discussion

Jim Demmel lead a discussion in the final session. What follows summarizes the points brought out during that session.

- What linear algebra functions should the BLACS support?

  This will depend upon the intended users of the BLACS more than anything else. If the BLACS are to be used for all applications, then a wider set of functions will have to be included.

- Which target machines should be supported?

  The machines fall into several types depending on whether they are SIMD or MIMD, and whether the memory is distributed or not, with the distributed memory machines being the ones of most immediate interest. Sometimes underlying message passing systems differ as to whether they are blocking or nonblocking, etc., as well. The package could be designed to work with both SIMD and MIMD models, however this may complicate the situation. For example, a code written in a language like Fortran D will look significantly different than one doing explicit message passing.

- Which data layouts should be supported?

  This is the most frequently asked question. There are clearly a great many more layouts possible than in LAPACK I, because of distributed memories of various sizes per processor, as well as the desire by some to support "multiple instance" routines, i.e., multiple independent input matrices describing independent problems. If these are laid out in separate memories, or "transposed" with part of each matrix in each memory, completely different algorithms might be needed. In addition, the user may only use some of the processors, or have his own special layout.

- How should performance scale?

  There are several different ways to scale problems. The efficiency E is a function of the input problem size $N = n^2$ (n = matrix dimension), the number of processors p, the memory per processor M, and the number of problem instances I. Scaling essentially means letting N, n, M and I grow in certain regimes without the efficiency E getting too low. Getting too low means being less than half as fast as a routine coded specifically for the machine. The following regimes are natural ones to consider trying to support:

  1. M and I are fixed; p and $N = O(p)$ are increasing. In other words, we add processors and increase the problem size proportionately. Since M is fixed, for large enough p this means that each memory has at most a subblock of the matrix, not even a full column.
  2. p and I are fixed; M and $N = O(M)$ are increasing. Here we increase the problem size as the memory grows. This corresponds to each memory storing a fixed fraction $1/p$ of the matrix.

Alan gave a personal perspective of the situation. Manufacturers of supercomputers are often slow to give users hands–on access to the underlying networks. Perhaps this is an appropriate strategy for them; their economic base consists of computational scientists, not computer scientists. It is our responsibility to decide what kinds of access to the networks we really want and need. At the very least, it is our responsibility to identify the communications routines that need to run fast on advanced architectures.

He also made the following remarks:

- SIMD/MIMD is not an issue. Synchronous/Asynchronous and Pipelining/Not Pipelining with arithmetic are issues.
- Not overlapping can cause the loss of at most a factor of 2, but makes programming portable. Without overlap, programs for LU factorization on all the machines can be made to look very much identical.
- A challenge (that is worth some thought) is whether overlapping of communication and arithmetic can be performed with a clean programming primitive.
- Subroutine names should be readable. The days of R2D2/C3PO are gone.
- Nearly all dense linear algebra programs on all machines use column (and/or row) wrapping. Thus, only parameters for row and column block sizes vary from machine to machine.
- Higher level abstractions can be illustrated with the idea of a column broadcast of column 1. It ought to look something like

  ```
  SPREAD(A(:,1))
  ```

  and should not depend on processor numbers or even whether the first columns of a matrix is in one or many processors. The software (person who writes the BLACS program) should figure out where the data must be sent and how it is to get there.
- Communications routines to perform matrix transpose (usually based on all-to-all personalized communication ideas) and matrix multiply steps (usually based on all-to-all broadcasting) are further needed.

Lennart Johnsson talked about the programming model used on the CM-2. He wants the communication primitives to work on arrays independent of the data layout. A number of CMSSL (CM Scientific Subroutine Library) routines performing the following operations were described:

- All–to–all broadcast
- Global reduction
- Matrix-vector multiply
- Matrix-matrix multiply

with data decomposition specifications; it can be viewed as a language for programming "data-parallel problems" that can be mapped efficiently onto different machine architectures, including both SIMD and MIMD distributed-memory machines.

Fortran D supports two levels of data decomposition—fine-grain problem mapping induced by the structure of the underlying computation, and coarse-grain machine mapping caused by translating the problem onto the finite resources of the machines. Data decompositions are specified using `DECOMPOSITION`, `ALIGN` and `DISTRIBUTE` statements, as shown in the following example.

```
REAL X(100,100)
DECOMPOSITION A(100,100)
ALIGN X(I,J) WITH A(I+1,J-1)
DISTRIBUTE A(BLOCK, *)
DISTRIBUTE A(*, CYCLIC)
```

Some of the other features of Fortran D include:

- Static and dynamic data decompositions
- Regular and irregular data decompositions
- Deterministic program semantics
- Support for reduction operations.

The goal of Fortran D is to ease the task of developing efficient parallel programs. Users do not need to explicitly insert parallel constructs, synchronization, or communications. Instead, the Fortran D compiler uses the data decomposition to automatically convert programs into single-program, multiple-data (SPMD) form with explicit message-passing. These node programs may take advantage of communications libraries such as Express or BLACS for efficient collective communications.

The data decomposition specifications in Fortran D are compatible with both Fortran 77 and Fortran 90. Researchers at Rice are developing the Fortran 77D compiler; the Fortran 90D compiler is being constructed at Syracuse. The Fortran D programming system includes a prototype compiler, automatic data partitioner, and static performance estimator. It is being developed in the context of the ParaScope parallel programming environment. Initial targets are the Intel iPSC/860 and Delta. It is planned that the Fortran77/90D compiler will be optimized for matrix algebra.

5 Other SIMD models

Alan Edelman and Lennart Johnsson spoke on an approach to the BLACS from the perspective of the CM2. They both felt that the BLACS, as proposed by van de Geijn, is at too low a level of abstraction and feel that a higher level is called for.

- Keep overheads low by utilizing advances in software technology—e.g., C macros, compiler in-line expansion
- Recognize existence of multiple application languages (C and Fortran-90, ...)
- Use a readable naming convention.

In addition, Zipcode will support PICL and other major protocols.

3.3 White paper on BLACS

Robert van de Geijn presented a "white paper" proposal for the BLACS. As described, the BLACS* are a set of communication routines that complement the Level 1, 2 and 3 BLAS. The communication library should provide the tool necessary for the implementation of numerical algorithms in linear algebra on distributed memory MIMD computers. The proposed routines can be be found in the Appendix.

The "white paper" outlines a proposed set of linear algebra communication routines for the specification and manipulation of data structures that arise when linear algebra algorithms are implemented on distributed memory multicomputers. The scope of the BLACS is intentionally limited. This package is not to be viewed as a complete communication library for all applications. It is intended primarily for software developers and to a lesser extent for experienced applications programmers in the area of dense numerical linear algebra. We see these routines complementing the existing Level 1, 2, and 3 BLAS; providing tools for the implementation of numerical algorithms in linear algebra for distributed memory MIMD machines.

It is important to realize what kind of algorithms inspired the BLACS. They are being developed as part of our effort to implement a subset of the LAPACK library to distributed memory MIMD architectures. In implementing these routines, we intend to perform minimal changes to the algorithms and codes, as well as maintain portability of the final product. The codes are written in FORTRAN77 in the SPMD (Single Program Multiple Data) paradigm. So far, matrices have been mapped to nodes using (column) panel-wrapped storage, where a matrix is partitioned into panels of constant width that are wrapped onto the p nodes so that panels $p + ki$, $k = 0, 1, \ldots$, are assigned to processors with index $i - 1$.

One result of the above decision is that it suffices to implement broadcast and combine operations in a way that involves all nodes, i.e., the nodes form a one–dimensional array as far as global operations are concerned, although a more complicated network can be utilized to speed up the execution of such operations.

4 Proposed Language: Fortran D

Geoffrey Fox from Syracuse University and Chau-Wen Tseng from Rice University discussed Fortran D, a language for portable parallel processing. Fortran D is a version of Fortran extended

*The word "BLACS" was first used by Alan Edelman in describing a set of communication routines.

- There are both a high level and a low level set of communication primitives in the package.

Disadvantages:

- A restricted programming model is supported
  - No nonblocking send/receives
  - Only one process per processor
  - No synchronous interprocessor communication.
- Packing and unpacking of data is the users responsibility.

A few points were mentioned with respect to collecting trace data. Tracing can be dangerous unless the library is used correctly. The volume of trace data can easily overwhelm and the overhead can be substantial.

3.2 Zipcode

Tony Skjellum spoke on a package called Zipcode [6] which provides a portable communications library. Zipcode works on top of the Reactive Kernel/Cosmic Environment. The reactive Kernel/Cosmic Environment is a portable "light-weight" multicomputer node operating system. The Reactive Kernal is implemented or emulated on the Intel iPSC/1, iPSC/2, iPSC/860, Symult S2010, and is emulated in shared memory computers such as the BBN TC2000 as well as networks of homogeneous NFS-connected workstations.

The key features of Zipcode are its design for extensibility, allowing the definition of many classes of communication and hence message receipt selectivity; support for abstraction of process lists into convenient working groups for communication; the ability to define many non-interfering communication contexts based on process lists with instantiation at runtime rather than compile-time; and the derivation of additional communication contexts through inheritance.

Skjellum provides a set of basic requirements for the BLACS:

- Support data-distribution-independent programming—applications define layout
- Provide logical 2D process grid abstraction
- Support global operations on logical 2D grids and subgrids
- Don't interfere with or constrain other communication patterns of an application
- Recognize possibility of multiple logical process grids in an application
- Don't interfere with MPMD programming
- Support application-defined pivoting and exploit communication pipelining

- The purpose of the BLAS and BLACS
- Their intended audience
- Criteria for inclusion
- View of the BLACS as software or specification
- Standardization of performance
- Testing and certification
- Conventions.

Jack Dongarra spoke on the development of LAPACK and the proposed extension to the project, the development of a core set of routines for distributed memory computers. The LAPACK project has generated approximately 500,000 lines of Fortran and in taking the package to distributed memory machines, software reuse is important. The LAPACK project [1] will be completed before the end of 1991, and the plan calls for a follow up project [2] which will address Fortran 90, C, distributed memory architectures, and trying to exploit IEEE arithmetic. Critical to the development of a distributed memory version of the linear algebra library is a portable/standard set of communication routines.

3 Existing Communication Libraries

3.1 PICL

Al Geist described a message passing interface for distributed memory parallel computer that was designed and in use at Oak Ridge National Laboratory called PICL (Portable Instrumented Communication Library) [4, 5]. The package is intended to aid in portable software development and to provide performance characterization through instrumentation and data visualization. The PICL package has been implemented on a number of platforms: Cogent, Intel iPSC/1, Intel iPSC/2, Intel iPSC/860, nCUBE, Symult S2010, Cosmic Environment, Linda, Unix System, and X-Windows. The library is available in C and Fortran. The performance monitoring is provided through event-driven tracing capability. The routines in PICL produce time-stamped records of processor activities that serve as input to a visualization tool called ParaGraph also developed at Oak Ridge [4].

Some of the advantages of a package of this form are:

- A portable program can be designed by using the library.
- The overhead of using the library is minimal when compared to the native commands.
- Parameter values are checked for validity during run time.
- The library is available in C and Fortran.
- Instrumentation can be provided automatically for performance and debugging monitoring.

LAPACK Working Note 34: Workshop on the BLACS

J.J. Dongarra

February 6, 1992

Abstract: Forty-three people met on March 28, 1991, to discuss a set of Basic Linear Algebra Communication Subprograms (BLACS). This set of routines is motivated by the needs of distributed memory computers.

1 Introduction

This is an informal report of a workshop on the BLACS held in Houston on March 28, 1991. The workshop was organized by Jack Dongarra and attended by 43 people. Similar workshops were held in conjunction with the development of the Basic Linear Algebra Subprograms (BLAS). The purpose of the Houston workshop was to consider preliminary proposals for a set of basic communication routines used in library routines for solving problems in linear algebra.

The principle conclusions reached in this workshop were as follows:

a) The motivation for the BLACS is to increase portability, efficiency and modularity at a high level.

b) The audience for the BLACS are mathematical software experts and people with large scale scientific computations to perform.

c) A systematic effort must be made to achieve a *de facto* standard for the BLACS. Further discussion would take place in the Fall 1991 at Cornell.

d) The discussion opened the possibility of a higher level of expressing than reflected in the "white paper".

2 General Issues Related to the BLACS

Richard Hanson discussed general issues he felt should be borne in mind when defining the BLACS. The issues fell under a number of headings and follow a report by David Dodson and John Lewis published in SIGNUM [3].

This work was supported in part by the National Science Foundation, under grant ASC-8715728 and the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615.

CS - 91 - 134

LAPACK Working Note 34: Workshop on the BLACS

J.J. Dongarra

Computer Science Department
University of Tennessee
Knoxville, TN 37996-1301

and

Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge, TN 37831

February 6, 1992

This work was supported in part by the National Science Foundation, under grant ASC-8715728 and the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615.