Tutorial on

High Performance Linear Algebra

Jack Dongarra
University of Tennessee
and
Oak Ridge National Laboratory

Goals of a Parallel Architecture

- **High Performance:**
  - **Performance > fastest uniprocessors**
- **Competitive cost-performance:**
  - **Cost-performance is competitive with workstations**
- **Scalable performance and cost-performance**
- **General-purpose: cost-effective for a wide range of applications**

Scalable Multiprocessors

What is Required?

- **Must scale the local memory bandwidth linearly.**
- **Must scale the global interprocessor communication bandwidth.**
- **Scaling memory bandwidth cost-effectively requires separate, distributed memories.**
- **Cost-effectiveness also requires best price-performance in individual processors.**

What we get

- **Compelling Price/Performance**
- **Tremendous scalability**
- **Tolerable entry price**
- **Tackle intractable problems**

Performance Numbers Comparing Various RISC Processors

Using the Linpack Benchmark

(All based on actual runs.)
Linpack n=100 based on Fortran code only.

Ax=b n=1000 is based on a blocked algorithm (LAPACK routine)
using the Level 3 BLAS as provided by the vendor.

Machine	MHz	Cycle time nsec	Linpack n=100 Mflops	% peak	Ax = b n=1000 Mflops	% peak	Theor Peak Mflops
DEC Alpha	300	3.3	140	23%	411	69%	600
IBM Power2	66	15	130	48%	236	89%	266
SGI Power	75	13.3	101	33%	260	87%	300
HP 735	100	10	61	31%	107	54%	198
DEC Alpha	200	5	43	22%	155	78%	200
DEC Alpha	182	5	39	21%	141	77%	182
IBM RS-580	66	15	38	30%	104	83%	125
DEC Alpha	160	6	36	23%	114	71%	160
DEC Alpha	150	7	30	20%	107	71%	150
IBM RS-550	42	24	26	31%	70	83%	84
HP 730/750	66	15	24	36%	47	71%	66
SGI Indy2/Crim	100	10	15	30%	32	64%	50
IBM RS-530	25	40	15	30%	42	84%	50
KSR (1 proc)	40	25	15	38%	31	78%	40
Intel i860	40	25	10	25%	34	85%	40
CRAY T90	500	2	???	??%	???	??%	2000
CRAY C90	238	4.2	387	41%	902	95%	952
CRAY J90	100	10	115	51%	193	96%	200
CRAY Y-MP	166	6	161	48%	324	97%	333
CRAY X-MP	118	8.5	121	51%	218	93%	235
CRAY 3	481	2.1	241	25%			962
CRAY 2	244	4.1	120	25%	384	79%	488
CRAY 1	80	12.5	27	17%	110	69%	160

Table 1: Multiprocessor Latency and Bandwidth.

Machine	OS	Latency $n = 0$ (μs)	Bandwidth $n = 10^6$ (MB/s)	$n_{1/2}$ bytes	Theoretical Bandwidth (MB/s)
Convex SPP1000 (PVM)	SPP-UX 3.0.4.1	76	11	1000	250
Convex SPP1000 (sm 1-n)	SPP-UX 3.0.4.1	2.5	82	1000	250
Convex SPP1000 (sm m-n)	SPP-UX 3.0.4.1	12	59	1000	250
Convex SPP1200 (PVM)	SPP-UX 3.0.4.1	63	15	1000	250
Convex SPP1200 (sm 1-n)	SPP-UX 3.0.4.1	2.2	92	1000	250
Convex SPP1200 (sm m-n)	SPP-UX 3.0.4.1	11	71	1000	250
Cray T3D (sm)	MAX 1.2.0.2	3	128	363	300
Cray T3D (PVM)	MAX 1.2.0.2	21	27	1502	300
Intel Paragon	OSF 1.0.4	29	154	7236	175
Intel Paragon	SUNMOS 1.6.2	25	171	5856	175
Intel Delta	NX 3.3.10	77	8	900	22
Intel iPSC/860	NX 3.3.2	65	3	340	3
Intel iPSC/2	NX 3.3.2	370	2.8	1742	3
IBM SP-1	MPL	270	7	1904	40
IBM SP-2	MPI	35	35	3263	40
KSR-1	OSF R1.2.2	73	8	635	32
Meiko CS2 (sm)	Solaris 2.3	11	40	285	50
Meiko CS2	Solaris 2.3	83	43	3559	50
nCUBE 2	Vertex 2.0	154	1.7	333	2.5
nCUBE 1	Vertex 2.3	384	0.4	148	1
NEC Cenju-3	Env. Rel 1.5d	40	13	900	40
NEC Cenju-3 (sm)	Env. Rel 1.5d	34	25	400	40
SGI	IRIX 6.1	10	64	799	1200
TMC CM-5	CMMD 2.0	95	9	962	10
Ethernet	TCP/IP	500	0.9		1.2
FDDI	TCP/IP	900	9.7		12
ATM-100	TCP/IP	900	3.5		12

Table 2: Computation Performance.

Machine	OS	Clock cycle MHz	(nsec)	Linpack 100 Mfls	(ops/cl)	Linpack 1000 Mfls	(ops/cl)	Latency us	(cl)
Convex SPP1000 (PVM)	SPP-UX 3.0.4.1	100	(10)	48	(.48)	123	(1.23)	76	(7600)
Convex SPP1000 (sm 1-n)								2.6	(260)
Convex SPP1000 (sm m-n)								11	(1080)
Convex SPP1200 (PVM)	SPP-UX 3.0.4.1	100	(8.33)	65	(.54)	123	(1.02)	63	(7560)
Convex SPP1200 (sm 1-n)								2.2	(264)
Convex SPP1200 (sm m-n)								11	(1260)
Cray T3D (sm)	MAX 1.2.0.2	150	(6.67)	38	(.25)	94	(.62)	3	(450)
Cray T3D (PVM)								21	(3150)
Intel Paragon	OSF 1.0.4	50	(20)	10	(.20)	34	(.68)	29	(1450)
Intel Paragon	SUNMOS 1.6.2							25	(1250)
Intel Delta	NX 3.3.10	40	(25)	9.8	(.25)	34	(.85)	77	(3080)
Intel iPSC/860	NX 3.3.2	40	(25)	9.8	(.25)	34	(.85)	65	(2600)
Intel iPSC/2	NX 3.3.2	16	(63)	.37	(.01)	–	(–)	370	(5920)
IBM SP-1	MPL	62.5	(16)	38	(.61)	104	(1.66)	270	(16875)
IBM SP-2	MPI	66	(15.15)	130	(1.97)	236	(3.58)	35	(2310)
KSR-1	OSF R1.2.2	40	(25)	15	(.38)	31	(.78)	73	(2920)
Meiko CS2 (MPI)	Solaris 2.3	90	(11.11)	24	(.27)	97	(1.08)	83	(7470)
Meiko CS2 (sm)								11	(990)
nCUBE 2	Vertex 2.0	20	(50)	.78	(.04)	2	(.10)	154	(3080)
nCUBE 1	Vertex 2.3	8	(125)	.10	(.01)	–	(–)	384	(3072)
NEC Cenju-3	Env Rev 1.5d	75	(13.3)	23	(.31)	39	(.52)	40	(3000)
NEC Cenju-3(sm)	Env Rev 1.5d	75	(13.3)	23	(.31)	39	(.52)	34	(2550)
SGI Power Challenge	IRIX 6.1	90	(11.11)	126	(1.4)	308	(3.42)	10	(900)
TMC CM-5	CMMD 2.0	32	(31.25)	–	(–)	–	(–)	95	(3040)

The Situation:

Parallel scientific applications are typically

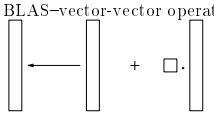
- written from scratch, or manually adapted from sequential programs
- using simple SPMD model
- in Fortran or C
- with explicit message passing primitives

Which means

- similar components are coded over and over again,
- codes are difficult to develop and maintain
- debugging particularly unpleasant...
- difficult to reuse components
- not really designed for long-term software solution

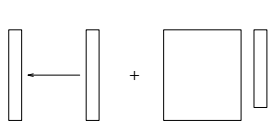
Level 1, 2, and 3 BLAS

- Level 1 BLAS–vector-vector operations



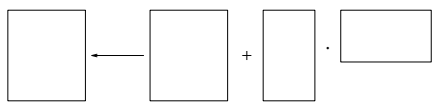
$y \leftarrow y + \alpha x$, also dot product, ...

- Level 2 BLAS–matrix-vector operations



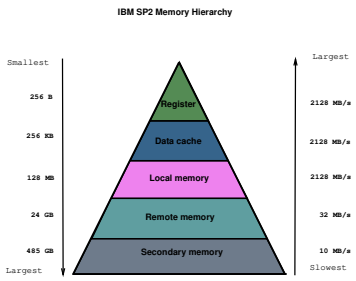
$y \leftarrow y + Ax$, also triangular solve, rank-1 update

- Level 3 BLAS–matrix-matrix operations



$A \leftarrow A + BC$, also block triangular solve, rank-k update

Why Higher Level BLAS?

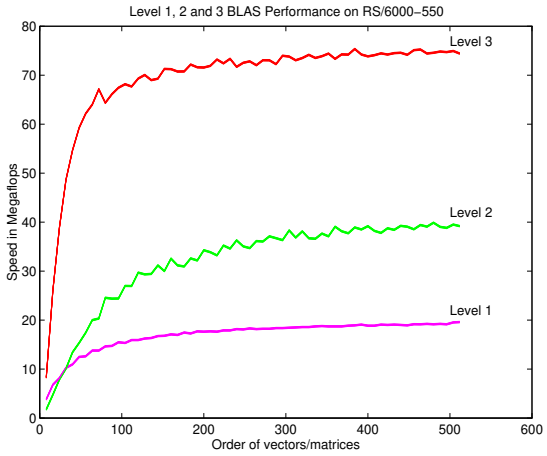


- Can only do arithmetic on data at top
  $\longrightarrow$ keep active data as close to top of hierarchy as possible
- Higher level BLAS lets us do this:

		mem ref	flops	flops/mem ref
Level 1 BLAS	$y \leftarrow y + \alpha x$	$3n$	$2n$	$2/3$
Level 2 BLAS	$y \leftarrow y + Ax$	n^2	$2n^2$	2
Level 3 BLAS	$A \leftarrow A + BC$	$4n^2$	$2n^3$	$n/2$

- On parallel machines
  higher level BLAS $\longrightarrow$increase granularity $\longrightarrow$ lower synchronization cost

BLAS – BASICS



- Development of blocked algorithms using Level 3 BLAS,
- $\Rightarrow$ LAPACK, high performance, portability.

BLAS – REFERENCES

- BLAS software and documentation can be obtained via:
  - WWW: `http://www.netlib.org/blas`,
  - (anonymous) ftp `netlib2.cs.utk.edu`:
    `cd blas; get index`
  - email `netlib@ornl.gov` with the message:
    `send index from blas`
- Comments and questions can be addressed at
  `lapack@cs.utk.edu`

- C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, *Basic Linear Algebra Subprograms for Fortran Usage*, ACM Transactions on Mathematical Software, 5:308–325, 1979.
- J. Dongarra, J. Du Croz, S. Hammarling, and R. Hanson, *An Extended Set of Fortran Basic Linear Algebra Subprograms*, ACM Transactions on Mathematical Software, 14(1):1–32, (1988).
- J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling, *A Set of Level 3 Basic Linear Algebra Subprograms*, ACM Transactions on Mathematical Software, 16(1):1–17, 1990.

Blocked Algorithms have been developed for:

- LU factorization
- Cholesky factorization
- Factorization of symmetric indefinite matrices
- Matrix inversion
- Banded LU and Cholesky factorization
- QR factorization
- Form Q or Q^TB
- Orthogonal reduction to:
  - Hessenberg
  - symmetric tridiagonal
  - bidiagonal
- Block QR iteration for nonsymmetric eigenvalue problems

LAPACK

- Linear algebra library in Fortran 77:
  - solution of systems of equations
  - solution for eigenvalue problems
- Combine algorithms from LINPACK and EISPACK into a single package
- Efficient on a wide range of high-performance computers
  - Single and double precision routines
  - Complex counterparts
- User interface similar to LINPACK
- Built on the Level 1, 2, and 3 BLAS

LAPACK – CONTENTS

- **Basic problems:**

Linear systems:	$Ax = b$
Least squares:	$min_x \|Ax - b\|_2$
Singular value decomposition:	$A = U\Sigma V^T$
Eigenvalues and vectors:	$Ax = \lambda x, Ax = \lambda Bx$

- **Factorizations:**

$A = LU$, LL^T, QR, QRP, RQ, QL, LQ
$A = V\Lambda V^T$ (SEP and NEP), $A = U\Sigma V^T$ (SVD)
$A - \lambda B = V(\Lambda_A - \lambda\Lambda_B)V^T$, $AB = V\Lambda V^{-1}$
(GSEP, GNEP and GSVD)

- **Dense, banded, symmetric packed, ...**
  (no general sparse yet)
- **Real and complex, single and double precision**
- **Error bounds, condition estimation** for (almost) everything
- **Easy-to-use drivers plus computational kernels**
- **Uses BLAS for high performance**
  (manufacturers optimize BLAS)
- **(Trans)portable FORTRAN 77 (+BLAS)**, 579K lines
  (248K library, 351K testing and timing).

LAPACK timing results for a cray Y/MP (Mflops)

Name	Matrix size N					
	32	64	128	256	512	1024
LU						
SGETRF (1 proc)	40	108	195	260	290	304
(2 proc)	32	91	229	408	532	588
(4 proc)	32	90	260	588	914	1097
(8 proc)	32	90	205	375	1039	1974
Cholesky						
SPOTRF (1 proc)	34	95	188	259	289	301
(2 proc)	29	84	221	410	539	594
(4 proc)	29	84	252	598	952	1129
(8 proc)	29	84	273	779	1592	2115
QR Factorization						
SGEQRF (1 proc)	54	139	225	275	294	301
(2 proc)	50	134	256	391	505	562
(4 proc)	50	136	292	612	891	1060
(8 proc)	50	133	328	807	1476	1937

The BLAS as the Key to Portability

Speed in megaflops of Level 2 and Level 3 BLAS operations on a CRAY C90

(all matrices are of order 1000; U is upper triangular)

Number of processors:	1	2	4	8	16
Level 2: $y \leftarrow \alpha Ax + \beta y$	899	1780	3491	6783	11207
Level 3: $C \leftarrow \alpha AB + \beta C$	900	1800	3600	7199	14282
Level 2: $x \leftarrow Ux$	852	1620	3063	5554	6953
Level 3: $B \leftarrow UB$	900	1800	3574	7147	13281
Level 2: $x \leftarrow U^{-1}x$	802	1065	1452	1697	1558
Level 3: $B \leftarrow U^{-1}B$	896	1792	3578	7155	14009

Level 1 BLAS : for vector operations, such as $y \leftarrow \alpha x + y$

Level 2 BLAS : for matrix-vector operations, such as $y \leftarrow \alpha Ax + \beta y$

Level 3 BLAS : for matrix-matrix operations, such as $C \leftarrow \alpha AB + \beta C$

Here, A, B and C are matrices, x and y are vectors, and α and β are scalars.

LAPACK – REFERENCES

- LAPACK software and documentation can be obtained via:
  - WWW: `http://www.netlib.org/lapack`,
  - WWW: `http://www.netlib.org/lapack/lawns`,
  - WWW: `http://www.netlib.org/c++/lapack++`,
  - (anonymous) ftp `netlib2.cs.utk.edu`:
    - cd lapack; get index
    - cd lapack/lawns; get index
    - cd linalg; get index
    - cd c++; get index
  - email `netlib@ornl.gov` with the message:

    `send index from lapack`
- Comments and questions can be addressed at

  `lapack@cs.utk.edu`

- E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide, Second Edition*, SIAM, Philadelphia, PA, 1995.

LAPACK / ScaLapack
A Scalable Library for Distributed Memory Machines

- **LAPACK success (from HP-48G to CRAY C-90)**
  - **Targets workstations, vector machines, shared memory**
  - **427,269 requests since then (as of 8/95)**
  - **Dense and banded matrices**
  - **Linear systems, least squares, eigenproblems**
  - **Manual available from SIAM, html, Japanese translation**
  - **Second release and manual in 9/94**
  - **Used by Cray, IBM, Convex, NAG, IMSL, ...**
- **Provide useful scientific program libraries for computational scientists working on new parallel architectures**
  - **Move to new generation high performance machines - SP-2, T3D, Paragon, Clusters of WS, ...**
  - **Add functionality (generalized eigenproblem, sparse matrices, ...)**

- **New challenges**
  - **No standard software**
    Fortran 90, HP Fortran, PVM, MPIbut ...
  - **Many flavors of message passing**
    Need standard - BLACS
  - **Highly varying ratio of**

$$r = \frac{\text{computation speed}}{\text{communication speed}}$$

  - **Many ways to layout data**
  - **Fastest parallel algorithm sometimes less stable numerically**

Needs

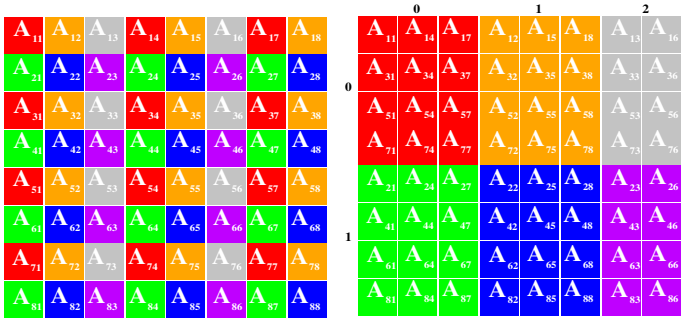
- **Standards**
  - **Portable**
  - **Vendor-supported**
  - **Scalable**
- **High efficiency**
- **Protect programming investment, if possible**
- **Avoid low-level details**
- **Scalability**

ScaLAPACK – OVERVIEW

Goal: Port Linear Algebra PACKage (LAPACK) to distributed-memory environments.

- **Scalability and Performance**
  - Optimized compute and communication engines
  - Blocked data access (level 3 BLAS) yields good node performance
  - 2-D Block-cyclic data decomposition yields good load balance
- **Portability**
  - BLAS: optimized compute engine
  - BLACS: communication kernel (PVM, MPI, ...)
- **Simple calling interface**
  - Stay as close as possible to LAPACK in calling sequence, storage, etc.
- **Ease of Programming / Maintenance**
  - Modularity: Build rich set of linear algebra tools
    - BLAS
    - BLACS
    - PBLAS
  - Whenever possible, use reliable LAPACK algorithms.

2-DIMENSIONAL BLOCK CYCLIC DISTRIBUTION



Global (left) and distributed (right) views of matrix

- **Ensure good load balance ⇒ Performance and scalability**,
- Encompass a large number (but not all) data distribution schemes,
- Need redistribution routines to go from distribution to the other,
- Not intuitive.

Communication Library for Linear Algebra

Basic Linear Algebra Communication Subroutines

Purpose of the BLACS:

- A design tool, they are a conceptual aid in design and coding.
- Associate widely recognized mnemonic names with communication operations, improve program readability and the self-documenting quality of code.
- Promote efficiency by identifying frequently-occurring operations of linear algebra which can be optimized on various computers.
- Program portability is improved through standardization of kernels without giving up efficiency since optimized versions will be available.

BLACS – BASICS

- Processes are embedded in a **2-dimensional grid**,

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

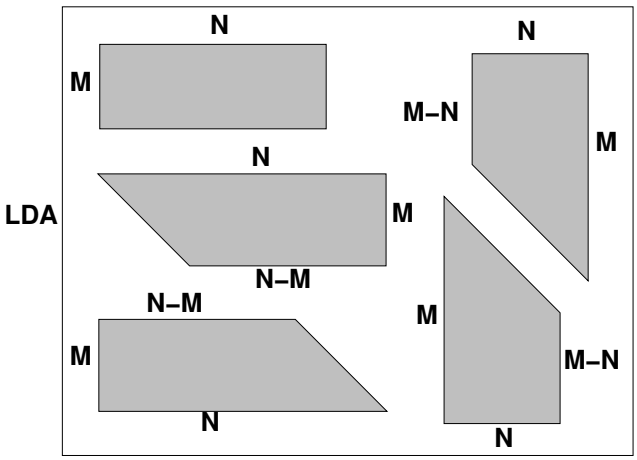
- An operation which involves more than one sender and one receiver is called a **"scoped operation"**. Using a 2D-grid, there are 3 natural scopes:

SCOPE	MEANING
Row	All processes in a process row participate.
Column	All processes in a process column participate.
All	All processes in the process grid participate.

→ use of `SCOPE` parameter.

BLACS – BASICS

• **Operations are matrix based and ID-less**



• **4 Types of BLACS routines: point-to-point communication, broadcast, combine operations and support routines.**

BLACS – COMMUNICATION ROUTINES

1. **Send/Recv**

Send (sub)matrix from one process to another:

□xxSD2D(ICONTXT, M, N, A, LDA, RDEST, CDEST)

□xxRV2D(ICONTXT, M, N, A, LDA, RSRC, CSRC)

□ (Data type)	xx (Matrix type)
I: INTEGER, S: REAL, D: DOUBLE PRECISION, C: COMPLEX, Z: DOUBLE COMPLEX.	GE: GEneral rectangular matrix, TR: TRapezoidal matrix.

```
CALL BLACS_GRIDINFO( ICONTXT, NPROW, NPCOL, MYROW, MYCOL )

IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
   CALL DGESD2D( ICONTXT, 5, 1, X, 5, 1, 0 )
ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
   CALL DGERV2D( ICONTXT, 5, 1, Y, 5, 0, 0 )
END IF
```


BLACS – COMMUNICATION ROUTINES

2. **Broadcast**

Send (sub)matrix to all processes or subsection of processes in SCOPE, using various distribution patterns (TOP):

□xxBS2D(ICONTXT, SCOPE, TOP, M, N, A, LDA)

□xxBR2D(ICONTXT, SCOPE, TOP, M, N, A, LDA, RSRC, CSRC)

SCOPE	TOP
'Row'	' ' (default)
'Column'	'Increasing Ring'
'All'	'1-tree' ...

```
CALL BLACS_GRIDINFO( ICONTXT, NPROW, NPCOL, MYROW, MYCOL )

IF( MYROW.EQ.0 ) THEN
   IF( MYCOL.EQ.0 ) THEN
      CALL DGEBS2D( ICONTXT, 'Row', ' ', 2, 2, A, 3 )
   ELSE
      CALL DGEBR2D( ICONTXT, 'Row', ' ', 2, 2, A, 3, 0, 0 )
   END IF
END IF
```


BLACS – COMBINE OPERATIONS

3. **Global combine operations**

Perform element-wise SUM, |MAX|, |MIN| operations, on rectangular matrices:

```
□GSUM2D( ICONTXT, SCOPE, TOP, M, N, A, LDA,
         RDEST, CDEST )

□GAMX2D( ICONTXT, SCOPE, TOP, M, N, A, LDA,
         RA, CA, RCFLAG, RDEST, CDEST )

□GAMN2D( ICONTXT, SCOPE, TOP, M, N, A, LDA,
         RA, CA, RCFLAG, RDEST, CDEST )
```

- RDEST = -1 indicates that the result of the operation should be left on all processes selected by SCOPE,
- For |MAX|, |MIN|, when RCFLAG = -1, RA and CA are not referenced; otherwise, these arrays are set on output with the coordinates of the process owning the corresponding maximum (resp. minimum) element in absolute value of A. Both arrays must be at least of size $M \times N$ and their leading dimension is specified by RCFLAG.

BLACS – ADVANCED TOPICS

The **BLACS context** is the BLACS mechanism for partitioning communication space. A defining property of a context is that a message in a context cannot be sent or received in another context. The BLACS context includes the definition of a grid, and each process coordinates in it.

It allows the user to

- create arbitrary groups of processes,
- create multiple overlapping and/or disjoint grids,
- isolate each process grid so that grids do not interfere with each other.

→ the BLACS context is compatible with the MPI communicator

BLACS – REFERENCES

- BLACS software and documentation can be obtained via:
  - WWW: `http://www.netlib.org/blacs`,
  - (anonymous) ftp `ftp.netlib.org`: `cd blacs; get index`
  - email `netlib@www.netlib.org` with the message:

    `send index from blacs`
- Comments and questions can be addressed to

  `blacs@cs.utk.edu`

- J. Dongarra and R. van de Geijn, *Two dimensional Basic Linear Algebra Communication Subprograms*, Technical Report UT CS-91-138, LAPACK Working Note #37, University of Tennessee, 1991.
- R. C. Whaley, *Basic Linear Algebra Communication Subprograms: Analysis and Implementation Across Multiple Parallel Architectures*, Technical Report UT CS-94-234, LAPACK Working Note #73, University of Tennessee, 1994.
- J. Dongarra and R. C. Whaley, *A User's Guide to the BLACS v1.0*, Technical Report UT CS-95-281, LAPACK Working Note #94, University of Tennessee, 1995.
- Message Passing Interface Forum, *MPI: A Message Passing Interface Standard*, *International Journal of Supercomputer Applications and High Performance Computing*, 8(3–4), 1994.
- A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine. A Users' Guide*

and Tutorial for Networked Parallel Computing, The MIT Press, Cambridge, Massachusetts, 1994.

PBLAS – INTRODUCTION

Parallel Basic Linear Algebra Subprograms for distributed memory MIMD computers.

- **Similar functionality** as the BLAS: distributed vector-vector, matrix-vector and matrix-matrix operations,
- **Simplification of the parallelization** of dense linear algebra codes: especially when implemented on top of the BLAS,
- **Clarity:** code is shorter and easier to read,
- **Modularity:** gives programmer larger building blocks,
- **Program portability:** machine dependency are confined to the PBLAS (BLAS and BLACS).

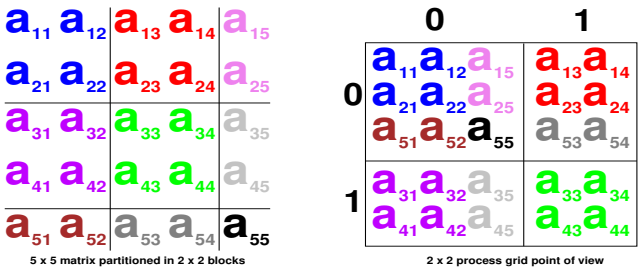
PBLAS – STORAGE CONVENTIONS

- An `M_`-by-`N_` **matrix is block partitionned** and these `MB_`-by-`NB_` blocks are distributed according to the

  2-dimensional block-cyclic scheme
  $\Rightarrow$ **load balanced computations, scalability**,

- Locally, the scattered **columns are stored contiguously** (FORTRAN "Column-major")

  $\Rightarrow$ **re-use of the BLAS** (leading dimension `LLD_`).

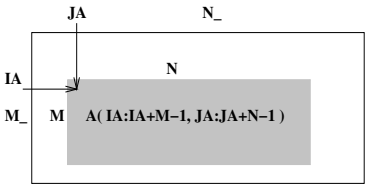


Descriptor `DESC_`: 8-Integer array describing the matrix layout, containing `M_`, `N_`, `MB_`, `NB_`, `RSRC_`, `CSRC_`, `CTXT_` and `LLD_`, where (`RSRC_`, `CSRC_`) are the coordinates of the process owning the first matrix entry in the grid specified by `CTXT_`.

Ex: `M_=N_=5`, `MB_=NB_=5`, `RSRC_=CSRC_= 0`, `LLD_` $\geq$ `3` (in process row 0), and `2` (in process row 1).

PBLAS – ARGUMENT CONVENTIONS

- **Global view of the matrix operands**, allowing global addressing of distributed matrices (hiding complex local indexing),



- **Code reusability**, interface very close to sequential BLAS:

```
CALL  DGEXXX( M, N, A( IA, JA ), LDA )

CALL  DGEMM( 'No Transpose', 'No Transpose',
$     M-J-JB+1, N-J-JB+1, JB, -ONE, A(J+JB,J),
$     LDA, A(J,J+JB), LDA, ONE, A(J+JB,J+JB),
$     LDA )
                          ↓
CALL PDGEXXX( M, N, A, IA, JA, DESCA )

CALL PDGEMM( 'No Transpose', 'No Transpose',
$     M-J-JB+1, N-J-JB+1, JB, -ONE, A, J+JB,
$     J, DESCA, A, J, J+JB, DESCA, ONE, A,
$     J+JB, J+JB, DESCA )
```


PBLAS – SPECIFICATIONS

```
DGEMV( TRANS, M, N, ALPHA, A, LDA, X, INCX,
       BETA, Y, INCY )
                          ↕
PDGEMV( TRANS, M, N, ALPHA, A, IA, JA, DESCA,
        X, IX, JX, DESCX, INCX,
        BETA, Y, IY, JY, DESCY, INCY )
```

- In the PBLAS, the increment specified for vectors is always global. So far only `INCX=1` and `INCX=DESCX(1)` are supported.

BLAS	PBLAS
`INTEGER LDA`	`INTEGER IA, JA, DESCA( 8 )`
`INTEGER INCX`	`INTEGER INCX, IX, JX, DESCX( 8 )`
`A, LDA`	`A, IA, JA, DESCA`
`X, INCX`	`X, IX, JX, DESCX, INCX`

- PBLAS matrix transposition routine (REAL):

```
PDTRAN( M, N, ALPHA, A, IA, JA, DESCA, BETA,
        C, IC, JC, DESCC )
```

- Level 1 BLAS functions have become PBLAS subroutines. Ouput Scalar correct in operand scope.

>> SPMD PROGRAMMING MODEL <<

PBLAS – REFERENCES

- PBLAS software and documentation can be obtained via:
  - WWW: `http://www.netlib.org/scalapack`,
  - (anonymous) ftp `ftp.netlib.org`:

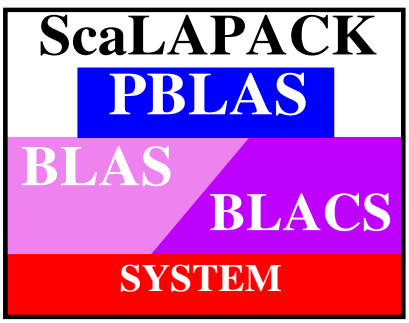
    `cd scalapack; get index`
  - email `netlib@www.netlib.org` with the message:

    `send index from scalapack`
- Comments and questions can be addressed to

  `scalapack@cs.utk.edu`

- J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker and R. C. Whaley, *A Proposal for a Set of Parallel Basic Linear Algebra Subprograms*, Technical Report UT CS-95-292, LAPACK Working Note #100, University of Tennessee, 1995.

ScaLAPACK IMPLEMENTATION



BLAS (Performance and Portability)

- Blocked data access (level 3 BLAS) yields good local performance

BLACS (Performance and Portability)

- Dedicated machine dependent communication kernel,
- 2D array-based and ID-less communication,
- Code re-entrance, multiple overlapping process grids,
- Safe co-existence with other parallel libraries.

Parallel Level 2 and 3 BLAS: PBLAS

Goal

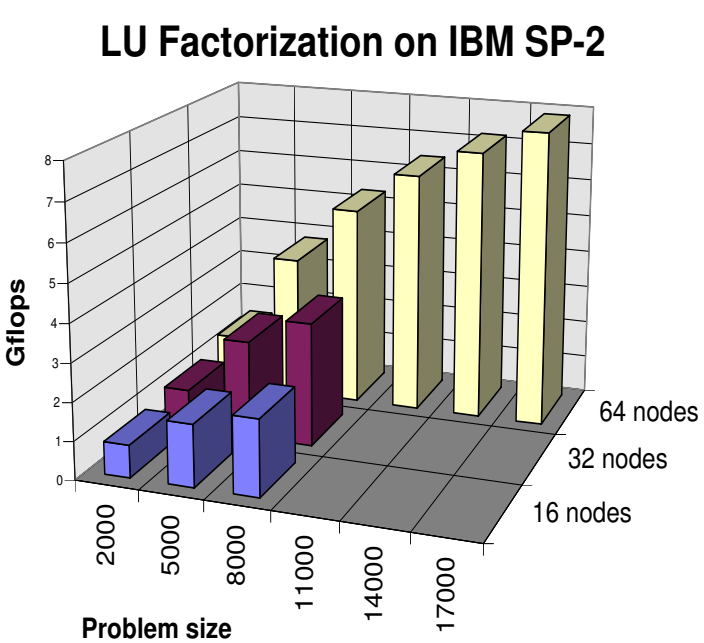
- Port sequential library for shared memory machine that use the BLAS to distributed memory machines with little effort.
- Reuse the existing software by hiding the message passing in a set of BLAS routines.
- Parallel implementation of the BLAS that understands how the matrix is layed out and when called can perform not only the operation but the required data transfer.

LAPACK $\rightarrow$ ScaLAPACK

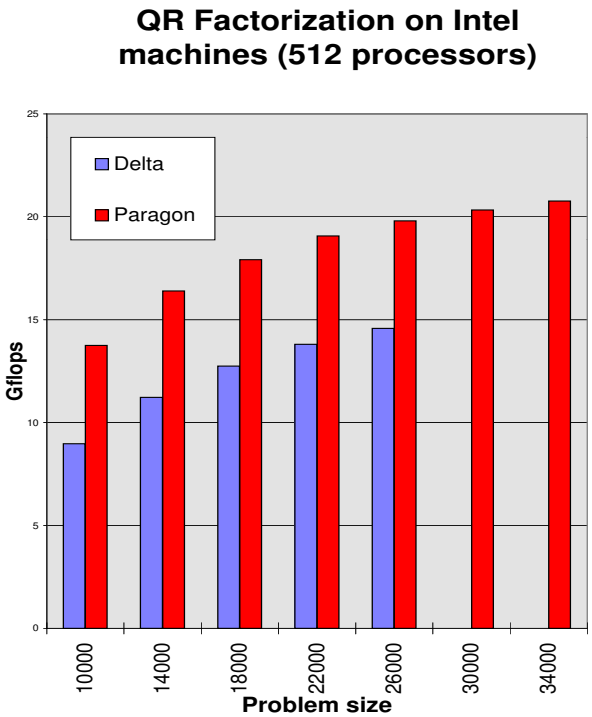
BLAS $\rightarrow$ PBLAS (BLAS, BLACS)

- Quality (maintenance)
- Portability (F77 - C)
- Efficiency - Reuseability (BLAS, BLACS)
- Hide Parallelism in (P)BLAS

PERFORMANCE RESULTS

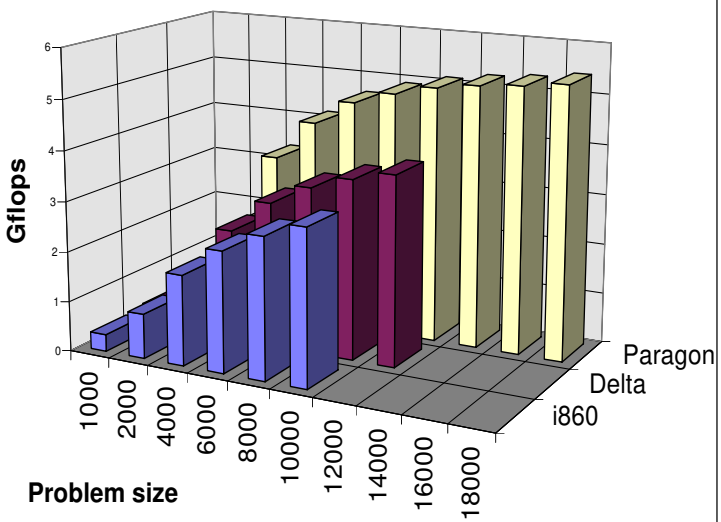


PERFORMANCE RESULTS

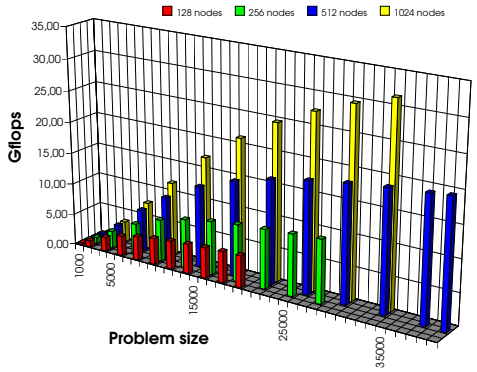


PERFORMANCE RESULTS

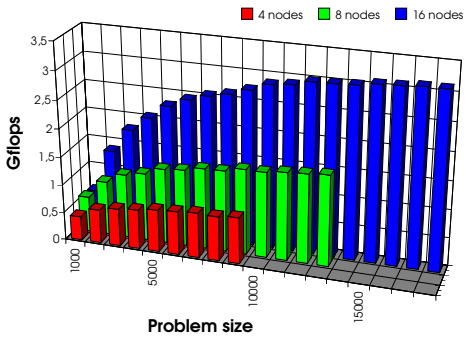
QR Factorization on 128 nodes Intel Machines



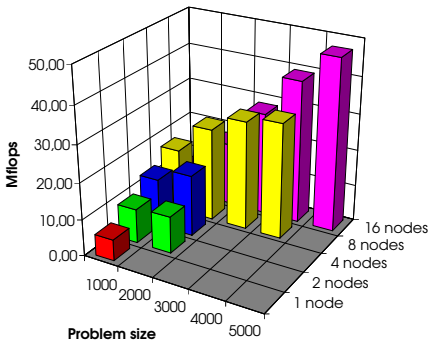
LU Factorization on Intel Paragon

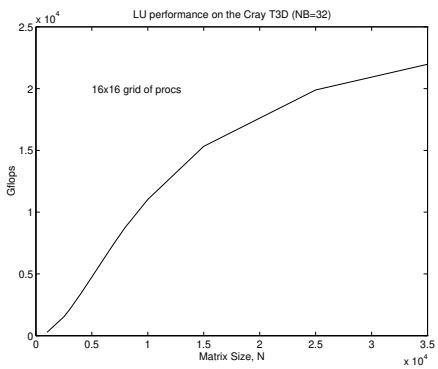


LU Factorization on IBM SP-2



LU Factorization on Sparc-5 (Ethernet)

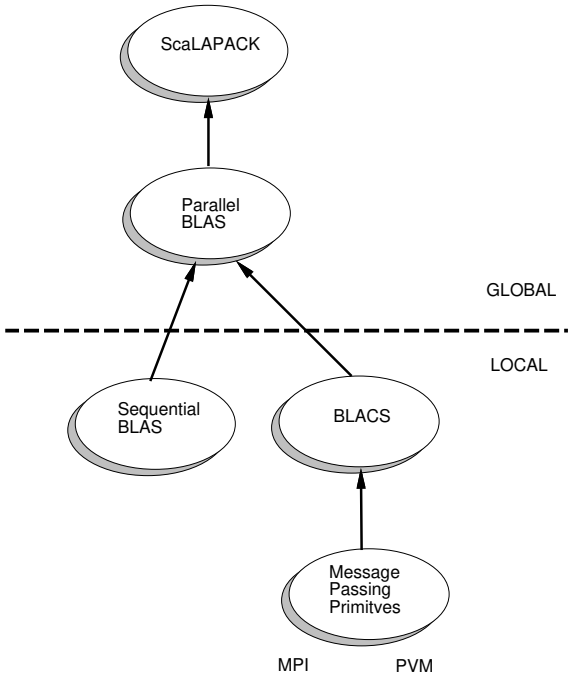




WHAT IS AVAILABLE TO YOU ?

- **Tools**
  - BLACS (CMMD, MPL, NX, PVM)
  - PBLAS
  - REDISTRIBUTION
  - PUMMA
- **Factorizations**
  - LU + solve
  - Cholesky + solve
  - QR, QL, LQ, RQ, QRP
- **Matrix inversions**
- Q, QC, Q^TC, CQ, CQ^T
- **Reductions**
  - to Hessenberg form
  - to symmetric tridiagonal form
  - to bidiagonal form
- **Symmetric eigensolver**

Software Framework



PVM calls within the BLACS

PVM (Parallel Virtual Machine) is a software package that permits a heterogeneous collection of serial, parallel, and vector computers hooked together by a network to appear as one large computer.

BLACS (Basic Linear Algebra Communication Subroutines) is a software package to perform communication at the same level as the BLAS are for computation.

- **BLACS have been written in terms of calls to PVM.**
- **Allows network of workstations to be used with minimal change.**

MPI – Message Passing Interface

Standard for Message Passing

- **BLACS → MPI underway**

SCALAPACK – ONGOING WORK

- **HPF**
  - HPF supports the 2D-block-cyclic distribution used by ScaLAPACK
  - HPF like calling sequence (Global indexing scheme)
- **Portability: MPI implementation of the BLACS**
- **More flexibility added to the PBLAS**
- **More testing and timing programs**
- **Condition estimators**
- **Iterative refinement of linear system solutions**
- **SVD**
- **Linear Least Square solvers**
- **Banded systems**
- **Non symmetric eigensolvers**
- ...

SCALAPACK – REFERENCES

- ScaLAPACK software and documentation can be obtained via:
  - WWW: `http://www.netlib.org/scalapack`,
  - WWW: `http://www.netlib.org/lapack/lawns`,
  - (anonymous) ftp `netlib2.cs.utk.edu`:
    - cd scalapack; get index
    - cd lapack/lawns; get index
  - email `netlib@ornl.gov` with the message:

    `send index from scalapack`
- Comments and questions can be addressed at

  `scalapack@cs.utk.edu`

- LAPACK Working Notes:

  #43, #55, #57, #58, #61, #65, #73, #80, #86, #91,

  #92, **#93**, **#94**, **#95**, #96, **#100**.

- J. Dongarra and D. Walker, *Software Libraries for Linear Algebra Computations on High Performance*

Computers, SIAM Review, Vol. 37, (2), pp. 151 – 180, 1995.