Scalapack : A Library for Parallel Linear Algebra

Xiaoye S. Li

xiaoye@nersc.gov

Scientific Computing Group

ERSUG, April 8 1997

Outline

- ScaLAPACK : dense, band matrix

  Functionality, software infrastructure, data distribution, performance

- Sparse matrix libraries
  - SuperLU : direct solver (XSL)
  - PETSc : PDE-oriented, including sparse solvers (W. Saphir)
  - Aztec : Krylov methods and preconditioners (J. Wu)

LAPACK and ScaLAPACK

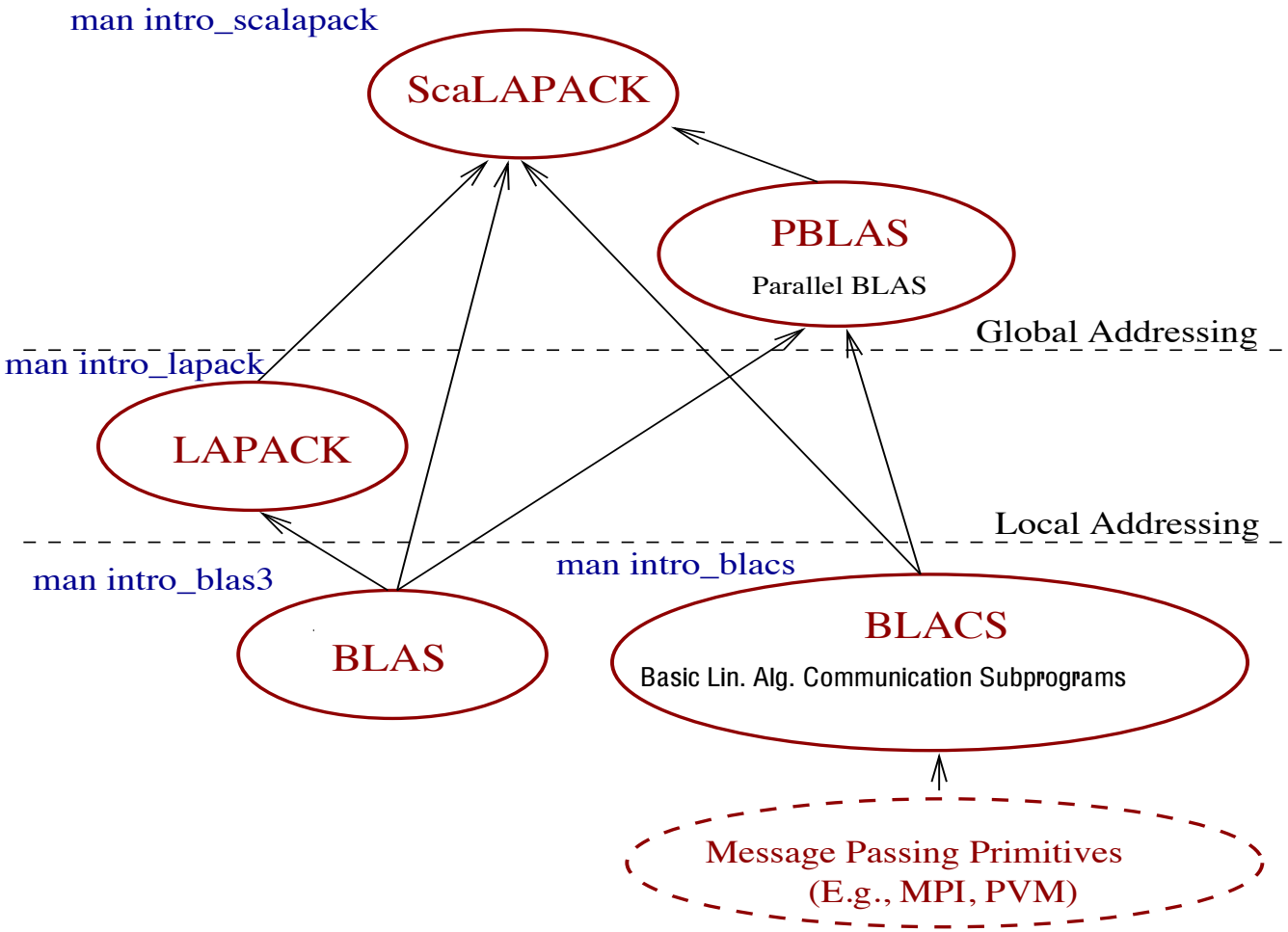
	LAPACK	ScaLAPACK
Machines	Workstations, Vector, SMP	Distributed memory, DSM
Based on	BLAS	BLAS, BLACS
Functionality	Linear systems Least squares Eigenproblems	Linear systems Least squares Eigenproblems (less than LAPACK)
Matrix types	dense, band	dense, band out-of-core
Data types	real, complex	real, complex
Error bounds	Complete	A few
Languages	F77 or C	F77 and C
Interfaces to	C++, F90	HPF
Manual?	Yes	Yes
Where?	www.netlib.org/lapack/	www.netlib.org/scalapack/

Achieve High Performance and Portability

- Use block-partitioned algorithms to increase data reuse ratio:
  $$\frac{\#\text{flops}}{\#\text{memory references}}$$
  - Reference locality $\Rightarrow$ good for memory hierarchy: register, cache(s), memory (local, remote), disk

- High-level algorithms call low-level BLAS (Basic Linear Algebra Subprograms)
  - Level 3 BLAS (matrix-matrix operations) fastest

ScaLAPACK Software Components

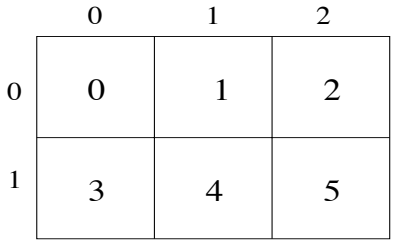


Matrix Partition and Placement

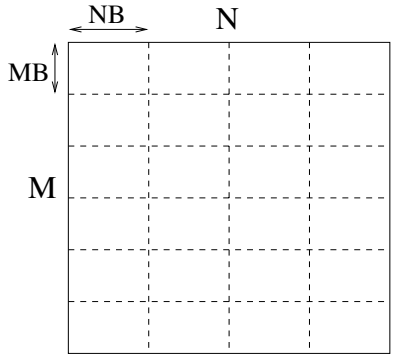
- Each PE only owns a portion of the global matrix (or submatrix)
  - The notion of local versus global data
- User's responsibility to distribute submatrices onto multiple PEs
- Examples of how to distribute matrix:
  1. Submatrix on each PE is generated locally
  2. One PE reads the matrix from a file, then sends pieces to other PEs
     $\Rightarrow$ May require message passing to do this
- 2-D block-cyclic layout is used in ScaLAPACK
  - Analyses of many algorithms to justify this layout
  - Supported in HPF

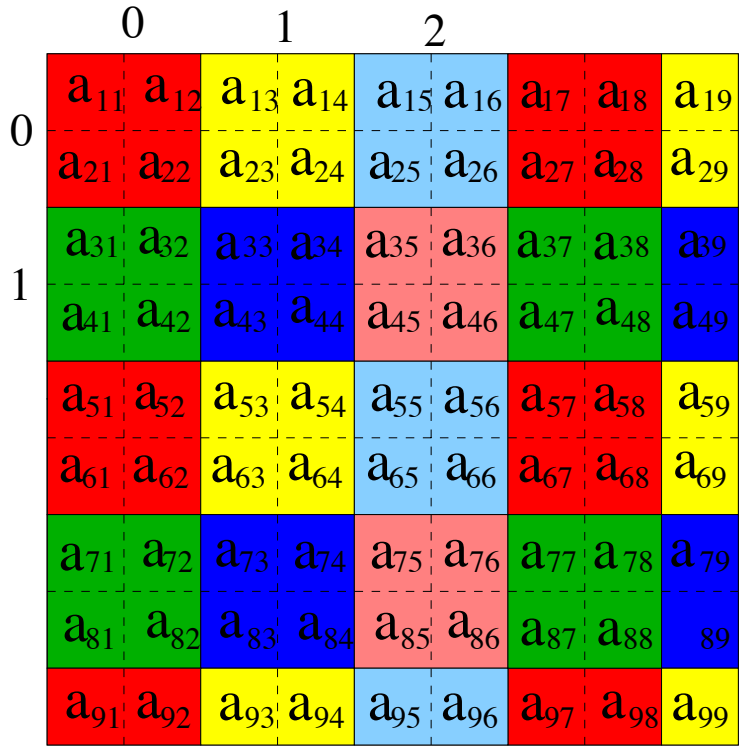
2-D Block-Cyclic Layout (in BLACS)

- Process grid
  - $(0 : nprocs - 1) \rightarrow P \times Q$ rectangular grid
  - Default assumes "row-major order" numbering

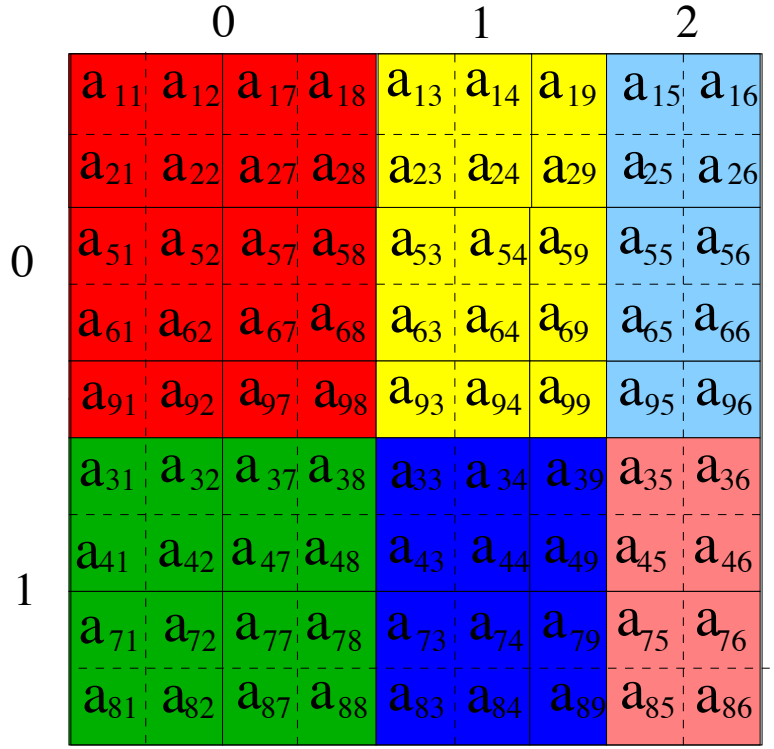


- Matrix to process mapping
  - An M-by-N matrix is partitioned into MB-by-NB blocks
  - The blocks are cyclically distributed across the process grid





Global view



Local (distributed) view

- 9×9 matrix of blocks 2×2, distributed over 2×3 process grid

Array Object Descriptor

DESC_()	Symbolic name	Scope	Definition
1	DTYPE_A	global	Descriptor type DTYPE_A=1 for dense matrices (Band, tridiagonal, out-of-core available too)
2	CTXT_A	global	BLACS context handle, indicating the BLACS process grid over which matrix A is distributed
3	M_A	global	Number of rows in the global array A
4	N_A	global	Number of columns in the global array A
5	MB_A	global	Blocking factor used to distribute the rows of A
6	NB_A	global	Blocking factor used to distribute the columns of A
7	RSRC_A	global	Process row over which the first row of the array A is distributed
8	CSRC_A	global	Process column over which the first column of the array A is distributed
9	LLD_A	local	Leading dimension of the local array LLD_A $\geq$ MAX(1, LOC$_r$(M_A))

Performance of LU (MFLOPS(PDGESV))

Machine	Process grid	Block size	N		
			5000	10000	15000
Cray T3E-600	1x4	32	884		
	2x8		2680	3356	3602
	4x16		6912	10299	12547
IBM SP2	1x4	50	603		
	2x8		1543	2149	
	4x16		3017	5596	7057
Intel XP/S MP Paragon	1x4	32	282		
	2x8		865		
	4x16		2084	3344	3963
Berkeley NOW	2x8	32	1310	1547	2436
	4x16		3091	4263	4560

Performance of Symmetric Eigensolver (TIME(PDSYEVX))

- Bisection + inverse iteration

Machine	Process grid	Block size	N		
			1000	2000	4000
Cray T3E-600	1x4	32	15	76	
	2x8		7	29	164
	4x16		5	17	64
IBM SP2	1x4	50	36		
	2x8		18	76	
	4x16		16	52	213
Intel XP/S MP Paragon	1x4	32	96		
	2x8		35	142	
	4x16		19	60	260
Berkeley NOW	1x4	32	28		
	2x8		15	64	
	4x8		14	47	608

Documentation, examples ...

- T3E on-line help
  - http://www.nersc.gov/hardware/hardware/T3E.html/#math
  - Examples: /usr/local/pkg/scg/SCALAPACK-1.5/examples/
  - To use:

```
module load scalapack
module help scalapack
```

- Documentation on netlib
  - ScaLAPACK Users' Guide
  - Installation Guide
  - LAPACK working notes

- More example programs
  - http://www.netlib.org/scalapack/examples/

SuperLU

- Sparse unsymmetric linear systems solver using Gaussian elim.
- http://www.nersc.gov/research/SCG/Sherry/
- Installed on J90 (killeen) ... `module help superlu_mt`

	SuperLU	SuperLU_MT
Machines	Uniprocessors	SMPs
Based on	BLAS	BLAS, Multithreading
Functionality	Sparsity orderings GEPP, solves Iterative refinement	Same
Matrix types	General sparse	Same
Data types	real, complex	real
Languages	C F77 interface	Same
Where?	www.netlib.org/ scalapack/prototype	Same

SuperLU Performance

- Large matrices on Origin 2000

Matrix	N	$nnz(A)$	P=1	P=8	P=18	Speedup
EX11	16614	1096948	209	33	20	10
RAEFSKY4	19779	1316789	229	39	25	9
BBMAT	38744	1771722	605	166	64	9
VAVASIS3	41092	1683902	598	136	75	8
TWOTONE	120750	1224224	313	58	38	8

- A 3-D flow calculation (EX11)

Machine	CPUs	Speedup	Mflops	Percent peak
C90	8	6	2583	33%
J90	16	12	831	25%
Power Challenge	12	7	1002	23%
Origin 2000	20	10	1335	17%
AlphaServer 8400	8	7	781	17%

SuperLU On T3E (In Progress)

- 2-D (nonuniform) block-cyclic layout to enhance scalability
- Factorization scales well
- Triangular solve needs improvement
- Performance

	N	nnz(A)	P=32	P=128	P=256	P=480	Mflops
bbmat	38744	1771722	93	33	23		1856
ecl32	51993	380415	69	23	17	15	7934
fidapm11	22294	623554	27	10	8		2003

- Software will be publically available when ready