



Connecting communities
through HPC

Linear Algebra Libraries for High-Performance Computing: Scientific Computing with Multicore and Accelerators

Presenters

Prof. Jack Dongarra (8:30 – 10:00)

University of Tennessee & Oak Ridge National Lab

Dr. Jakub Kurzak (10:30 – 12:00)

University of Tennessee

Prof. James Demmel (1:30 – 3:00)

University of California Berkeley

Dr. Michael Heroux (3:30 – 5:00)

Sandia National Laboratory

Overview of Dense Numerical Linear Algebra Libraries

- **BLAS**: kernel for dense linear algebra
- **LAPACK**: sequential dense linear algebra
- **ScaLAPACK**: parallel distributed dense linear algebra

L	A	P	A	C	K
L	-A	P	-A	C	-K
L	A	P	A	-C	-K
L	-A	P	-A	-C	K
L	A	-P	-A	C	K
L	-A	-P	A	C	-K

Linear **A**lgebra **PACK**age

What do you mean by performance?

.. What is a x flop/s?

- **x flop/s is a rate of execution, some number of floating point operations per second.**
 - Whenever this term is used it will refer to 64 bit floating point operations and the operations will be either addition or multiplication.
- **Tflop/s refers to trillions (10^{12}) of floating point operations per second and**
- **Pflop/s refers to 10^{15} floating point operations per second.**

.. What is the theoretical peak performance?

- **The theoretical peak is based not on an actual performance from a benchmark run, but on a paper computation to determine the theoretical peak rate of execution of floating point operations for the machine.**
- **The theoretical peak performance is determined by counting the number of floating-point additions and multiplications (in full precision) that can be completed during a period of time, usually the cycle time of the machine.**
- **For example, an Intel Xeon 5570 quad core at 2.93 GHz can complete 4 floating point operations per cycle or a theoretical peak performance of 11.72 GFlop/s per core or 46.88 Gflop/s for the socket.**

What Is LINPACK?

- .. LINPACK is a package of mathematical software for solving problems in linear algebra, mainly dense linear systems of linear equations.
- .. LINPACK: "LINear algebra PACKage"
 - Written in Fortran 66
- .. The project had its origins in 1974
- .. The project had four primary contributors: myself when I was at Argonne National Lab, Jim Bunch from the University of California-San Diego, Cleve Moler who was at New Mexico at that time, and Pete Stewart from the University of Maryland.
- .. LINPACK as a software package has been largely superseded by LAPACK, which has been designed to run efficiently on shared-memory, vector supercomputers.

Computing in 1974

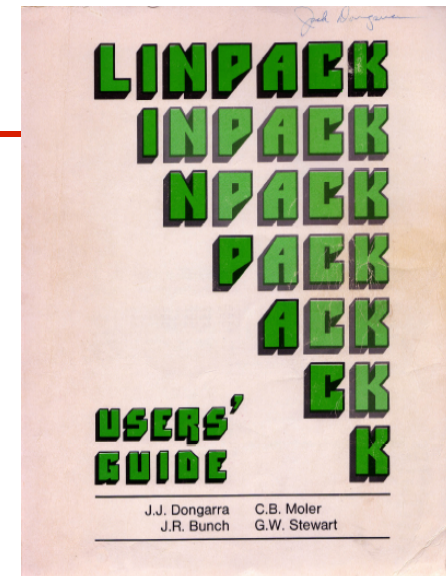
- .. **High Performance Computers:**
 - **IBM 370/195, CDC 7600, Univac 1110, DEC PDP-10, Honeywell 6030**
- .. **Fortran 66**
- .. **Trying to achieve software portability**
- .. **Run efficiently**
- .. **BLAS (Level 1)**
 - **Vector operations**
- .. **Software released in 1979**
 - **About the time of the Cray 1**

LINPACK Benchmark?

- .. The Linpack Benchmark is a measure of a computer's floating-point rate of execution.
 - It is determined by running a computer program that solves a dense system of linear equations.
- .. Over the years the characteristics of the benchmark has changed a bit.
 - In fact, there are three benchmarks included in the Linpack Benchmark report.
- .. LINPACK Benchmark
 - Dense linear system solve with LU factorization using partial pivoting
 - Operation count is: $\frac{2}{3} n^3 + O(n^2)$
 - Benchmark Measure: MFlop/s
 - Original benchmark measures the execution rate for a Fortran program on a matrix of size 100x100.

Accidental Benchmark

- Appendix B of the Linpack Users' Guide
 - Designed to help users extrapolate execution time for Linpack software package
- First benchmark report from 1977:
 - Cray 1 to DEC PDP-10



$\frac{2}{3} N^3$ ops time

UNIT = 10**6 TIME/(1/3 100**3 + 100**2)

Facility	TIME N=100 secs.	UNIT micro- secs.	Computer	Type	Compiler
NCAR	14.0	.049	0.14	CRAY-1	S CFT, Assembly BLAS
LASL	4.64	.148	0.43	CDC 7600	S FTN, Assembly BLAS
NCAR	3.58	.192	0.56	CRAY-1	S CFT
LASL	3.27	.210	0.61	CDC 7600	S FTN
Argonne	2.31	.297	0.86	IBM 370/195	D H
NCAR	1.91	.359	1.05	CDC 7600	S Local
Argonne	1.77	.388	1.33	IBM 3033	D H
NASA Langley	1.40	.489	1.42	CDC Cyber 175	S FTN
U. Ill. Urbana	1.34	.506	1.47	CDC Cyber 175	S Ext. 4.6
LLL	1.24	.554	1.61	CDC 7600	S CHAT, No optimize
SLAC	1.19	.579	1.69	IBM 370/168	D H Ext., Fast mult.
Michigan	1.09	.631	1.84	Amdahl 470/V6	D H
Toronto	.772	.890	2.59	IBM 370/165	D H Ext., Fast mult.
Northwestern	.477	1.44	4.20	CDC 6600	S FTN
Texas	.356	1.93*	5.63	CDC 6600	S RUN
China Lake	.352	1.95*	5.69	Univac 1110	S V
Yale	.265	2.59	7.53	DEC KL-20	S F20
Bell Labs	.197	3.46	10.1	Honeywell 6080	S Y
Wisconsin	.197	3.49	10.1	Univac 1110	S V
Iowa State	.194	3.54	10.2	Intel AS/5 mod3	D H
U. Ill. Chicago	.148	4.10	11.9	IBM 370/158	D G1
Purdue	.174	5.69	16.6	CDC 6500	S FUN
U. C. San Diego	.062	13.1	38.2	Burroughs 6700	S H
Yale	.040	17.1*	49.9	DEC KA-10	S F40

* TIME(100) = (100/75)**3 SGEFA(75) + (100/75)**2 SGESL(75)

High Performance Linpack (HPL)

Benchmark Name	Matrix dimension	Optimizations allowed	Parallel Processing
Linpack 100	100	compiler	— ^a
Linpack 1000 ^b	1000	hand, code replacement	— ^c
Linpack Parallel	1000	hand, code replacement	Yes
HPLinpack ^d	arbitrary	hand, code replacement	Yes

^a Compiler parallelization possible.

^b Also known as TPP (Toward Peak Performance) or Best Effort

^c Multiprocessor implementations allowed.

^d Highly-Parallel LINPACK Benchmark is also known as NxN Linpack Benchmark or High Parallel Computing (HPC).

A brief history of (Dense) Linear Algebra software

.. But the BLAS-1 weren't enough

- Consider AXPY ($y = \alpha \cdot x + y$): $2n$ flops on $3n$ read/writes
- Computational intensity = $(2n)/(3n) = 2/3$
- Too low to run near peak speed (read/write dominates)

.. So the BLAS-2 were developed (1984-1986)

- Standard library of 25 operations (mostly) on matrix/vector pairs
 - "GEMV": $y = \alpha \cdot A \cdot x + \beta \cdot y$, "GER": $A = A + \alpha \cdot x \cdot y^T$,
 $x = T^{-1} \cdot x$
 - Up to 4 versions of each (S/D/C/Z), 66 routines, 18K LOC
- Why BLAS 2 ? They do $O(n^2)$ ops on $O(n^2)$ data
- So computational intensity still just $\sim (2n^2)/(n^2) = 2$
 - OK for vector machines, but not for machine with caches

A brief history of (Dense) Linear Algebra software

.. The next step: BLAS-3 (1987-1988)

- Standard library of 9 operations (mostly) on matrix/matrix pairs
 - "GEMM": $C = \alpha \cdot A \cdot B + \beta \cdot C$, $C = \alpha \cdot A \cdot A^T + \beta \cdot C$, $B = T^{-1} \cdot B$
 - Up to 4 versions of each (S/D/C/Z), 30 routines, 10K LOC
- Why BLAS 3 ? They do $O(n^3)$ ops on $O(n^2)$ data
- So computational intensity $(2n^3)/(4n^2) = n/2$ - big at last!
 - Good for machines with caches, other mem. hierarchy levels

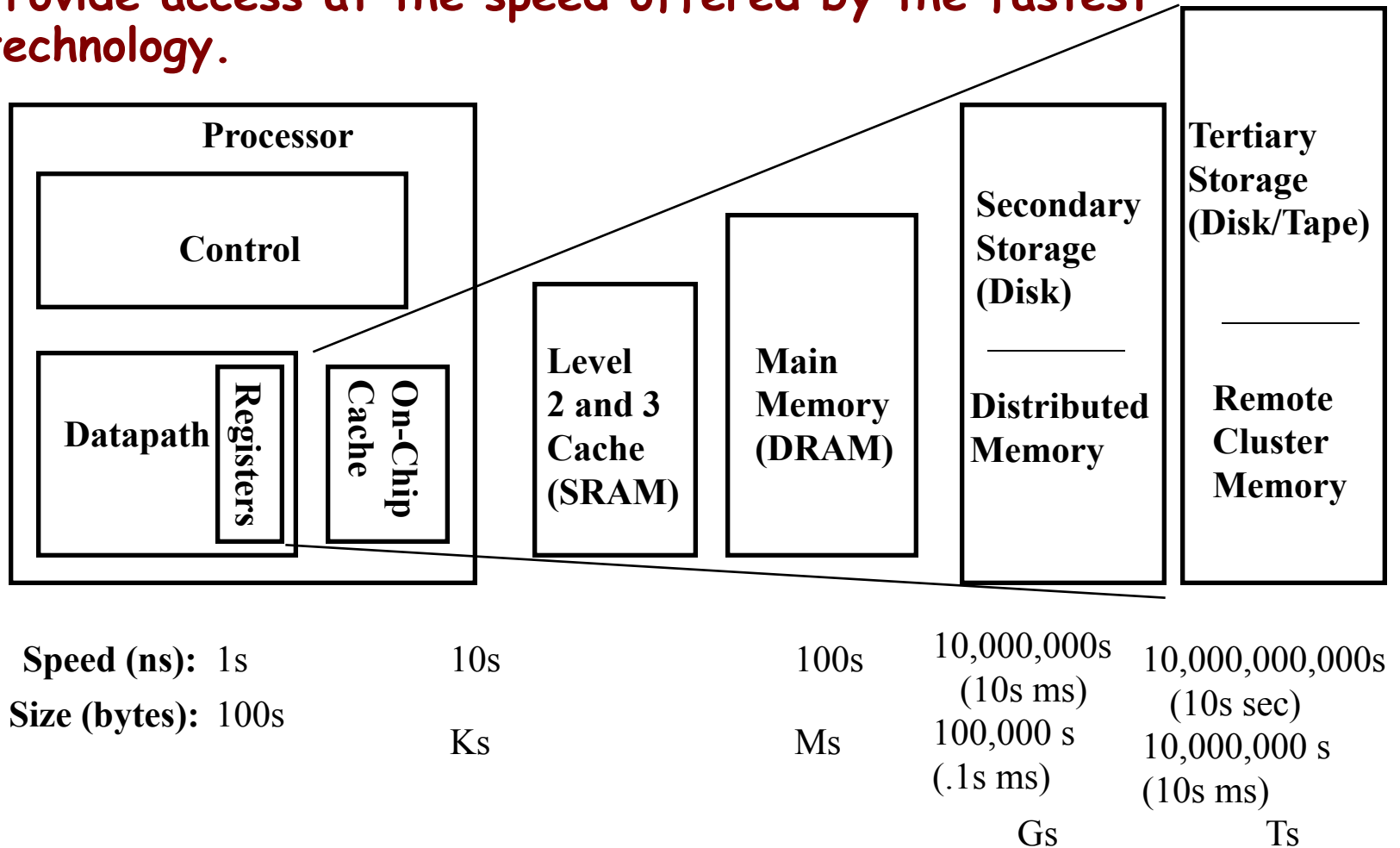
.. How much BLAS1/2/3 code so far (all at www.netlib.org/blas)

- Source: 142 routines, 31K LOC, Testing: 28K LOC
- Reference (unoptimized) implementation only
 - Ex: 3 nested loops for GEMM

Memory Hierarchy

By taking advantage of the principle of locality:

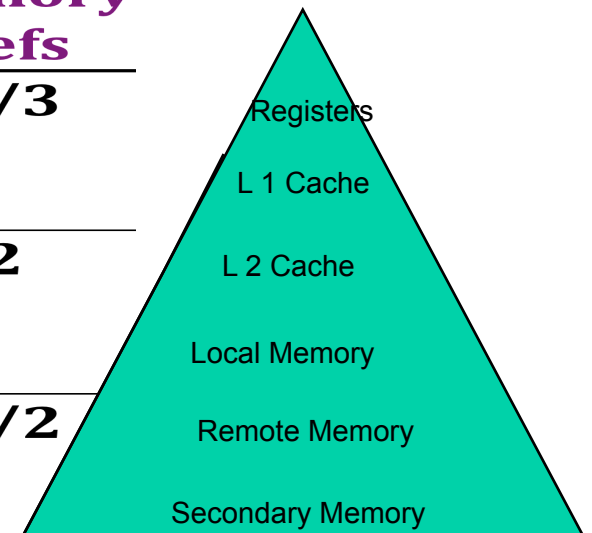
- Present the user with as much memory as is available in the cheapest technology.
- Provide access at the speed offered by the fastest technology.



Why Higher Level BLAS?

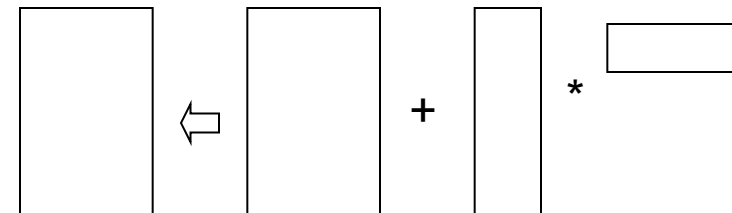
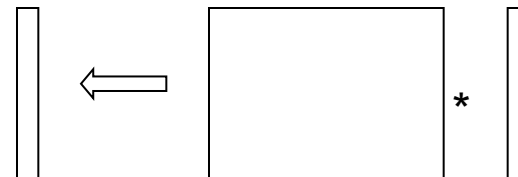
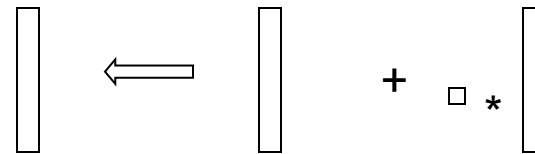
- Can only do arithmetic on data at the top of the hierarchy
- Higher level BLAS lets us do this

BLAS	Memory Refs	Flops	Flops/ Memory Refs
Level 1 $y = y + \alpha x$	$3n$	$2n$	$2/3$
Level 2 $y = y + Ax$	n^2	$2n^2$	2
Level 3 $C = C + AB$	$4n^2$	$2n^3$	$n/2$



Level 1, 2 and 3 BLAS

- **Level 1 BLAS**
Vector-Vector
operations
- **Level 2 BLAS**
Matrix-Vector
operations
- **Level 3 BLAS**
Matrix-Matrix
operations



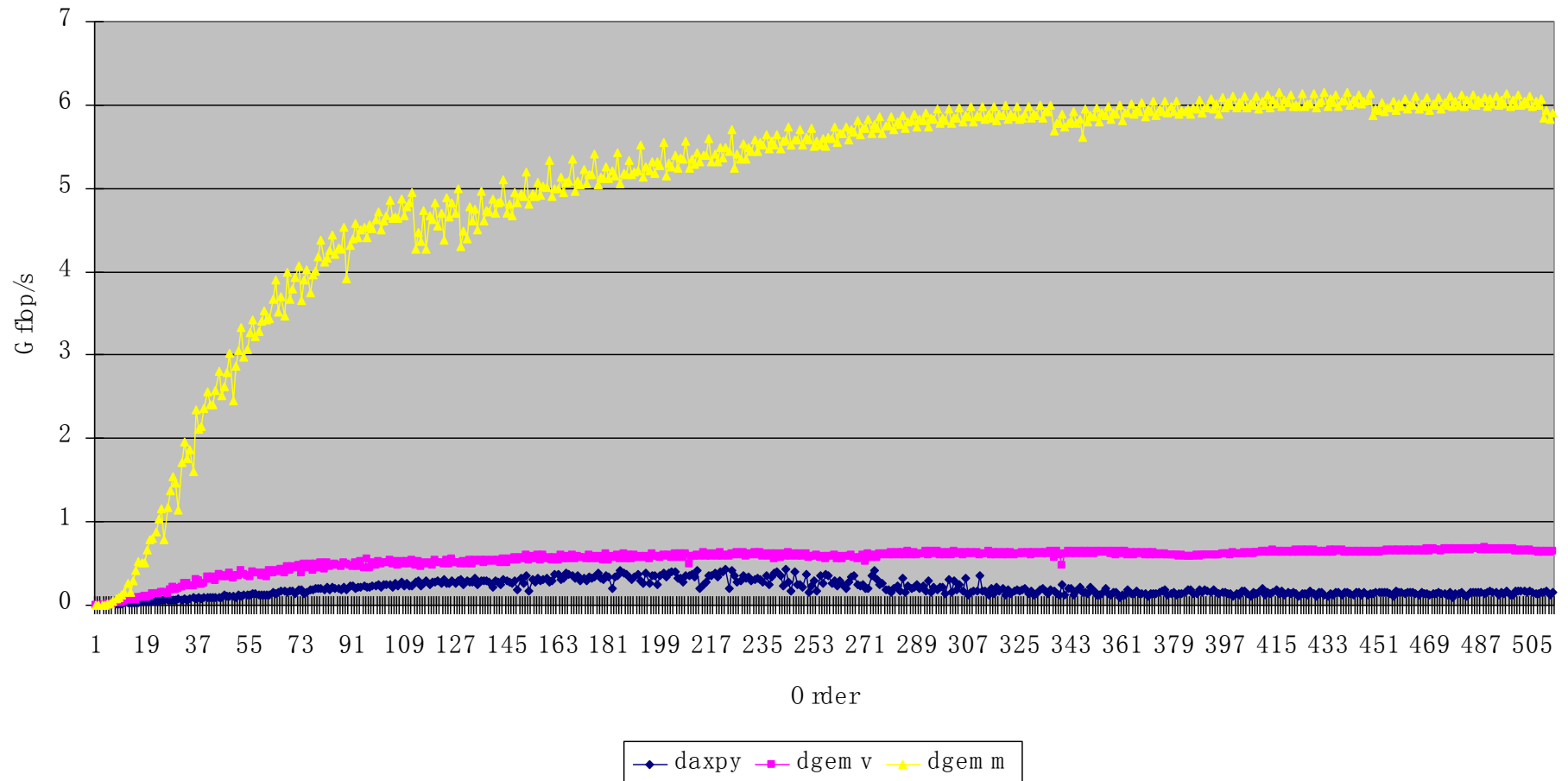
Level 1, 2 and 3 BLAS

Before (2007)

3.4 GHz EM 64T Xeon M KL8.1

Peak: 6.8 G fbp/s

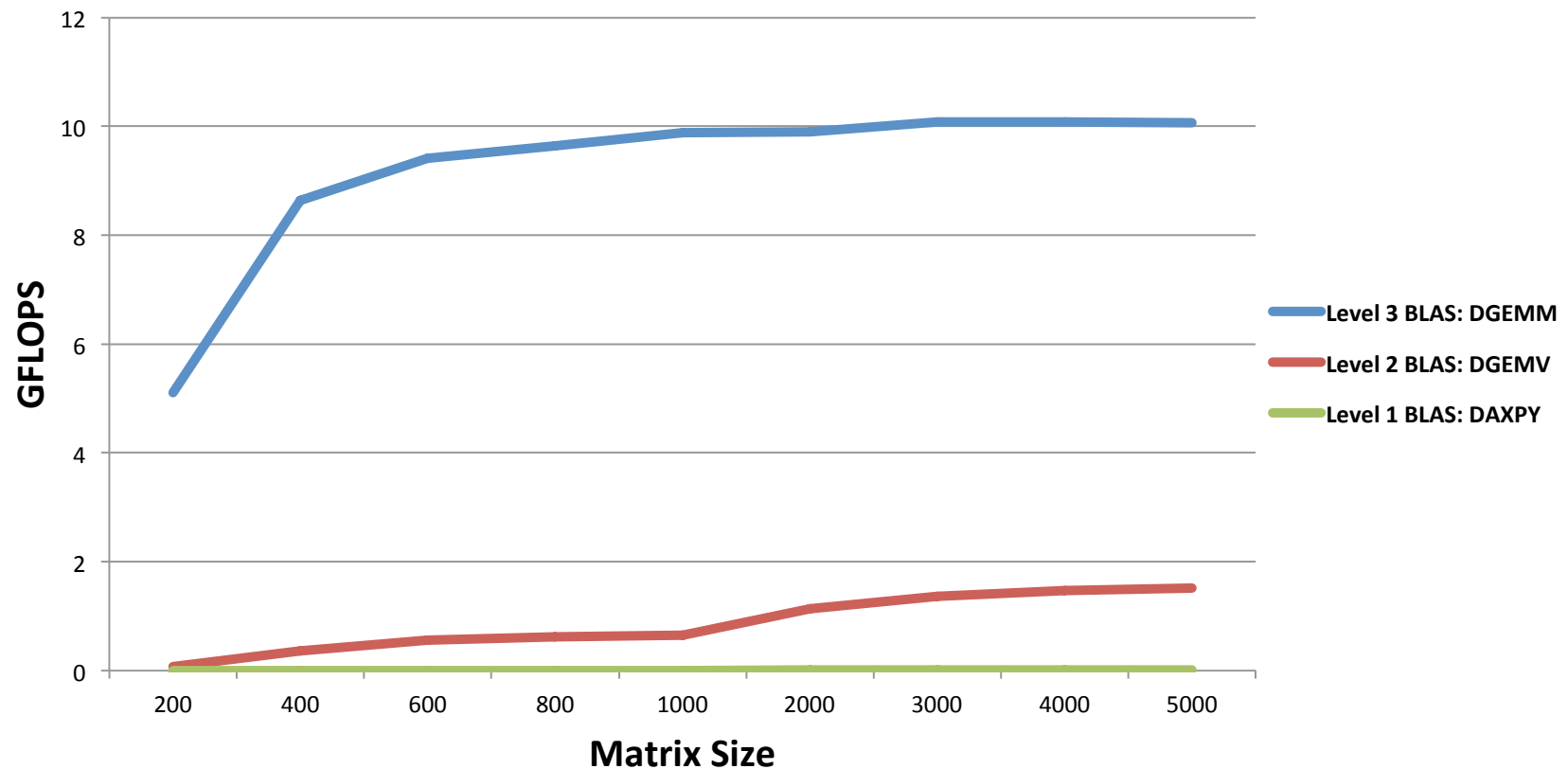
`gcc -fcommon -fno-strict-aliasing -funroll-loops -O3`



Level 1, 2 and 3 BLAS

Now (2011)

AMD Opteron 8439 SE Processor (6 cores total @ 2.8Ghz)
Using 1 core 11.2 Gflop/s theoretical peak



Level 1 BLAS

	dim	scalar	vector	vector	scalars	5-element array
SUBROUTINE xROTG (A, B, C, S)	
SUBROUTINE xROTMG(D1, D2, A, B,	PARAM)
SUBROUTINE xROT (N,		X,	INCX,	Y, INCY,	C, S)	
SUBROUTINE xROTM (N,		X,	INCX,	Y, INCY,	PARAM)	
SUBROUTINE xSWAP (N,		X,	INCX,	Y, INCY)		
SUBROUTINE xSCAL (N,	ALPHA,	X,	INCX)			
SUBROUTINE xCOPY (N,		X,	INCX,	Y, INCY)		
SUBROUTINE xAXPY (N,	ALPHA,	X,	INCX,	Y, INCY)		
FUNCTION xDOT (N,		X,	INCX,	Y, INCY)		
FUNCTION xDOTU (N,		X,	INCX,	Y, INCY)		
FUNCTION xDOTC (N,		X,	INCX,	Y, INCY)		
FUNCTION xxDOT (N,		X,	INCX,	Y, INCY)		
FUNCTION xNRM2 (N,		X,	INCX)			
FUNCTION xASUM (N,		X,	INCX)			
FUNCTION IxAMAX (N,		X,	INCX)			

Generate plane rotation
 Generate modified plane rotation
 Apply plane rotation
 Apply modified plane rotation
 $x \leftrightarrow y$
 $x \leftarrow \alpha x$
 $y \leftarrow x$
 $y \leftarrow \alpha x + y$
 $dot \leftarrow x^T y$
 $dot \leftarrow x^T y$
 $dot \leftarrow x^H y$
 $dot \leftarrow \alpha + x^T y$
 $nrm2 \leftarrow \|x\|_2$
 $asum \leftarrow \|re(x)\|_1 + \|im(x)\|_1$
 $amax \leftarrow 1^{\text{st}} k \ni |re(x_k)| + |im(x_k)|$
 $= \max(|re(x_i)| + |im(x_i)|)$

prefixes
 S, D
 S, D
 S, D
 S, D
 S, D, C, Z
 S, D, C, Z, CS, ZD
 S, D, C, Z
 S, D, C, Z
 S, D, DS
 C, Z
 C, Z
 SDS
 S, D, SC, DZ
 S, D, SC, DZ
 S, D, C, Z

Level 2 BLAS

options	dim	b-width	scalar	matrix	vector	scalar	vector
xGEMV (TRANS,	M, N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
xGBMV (TRANS,	M, N,	KL, KU,	ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
xHEMV (UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
xHBMV (UPLO,	N, K,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
xHPMV (UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)
xSYMV (UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
xSBMV (UPLO,	N, K,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)
xSPMV (UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)
xTRMV (UPLO, TRANS, DIAG,	N,		A,	LDA,	X, INCX)		
xTBMV (UPLO, TRANS, DIAG,	N, K,		A,	LDA,	X, INCX)		
xTPMV (UPLO, TRANS, DIAG,	N,		AP,	X, INCX)			
xTRSV (UPLO, TRANS, DIAG,	N,		A,	LDA,	X, INCX)		
xTBSV (UPLO, TRANS, DIAG,	N, K,		A,	LDA,	X, INCX)		
xTPSV (UPLO, TRANS, DIAG,	N,		AP,	X, INCX)			
options	dim	scalar	vector	vector	matrix		
xGER (M, N,	ALPHA,	X,	INCX,	Y, INCY,	A, LDA)	
xGERU (M, N,	ALPHA,	X,	INCX,	Y, INCY,	A, LDA)	
xGERC (M, N,	ALPHA,	X,	INCX,	Y, INCY,	A, LDA)	
xHER (UPLO,	N,	ALPHA,	X,	INCX,	A, LDA)		
xHPR (UPLO,	N,	ALPHA,	X,	INCX,	AP)		
xHER2 (UPLO,	N,	ALPHA,	X,	INCX,	Y, INCY,	A, LDA)	
xHPR2 (UPLO,	N,	ALPHA,	X,	INCX,	Y, INCY,	AP)	
xSYR (UPLO,	N,	ALPHA,	X,	INCX,	A, LDA)		
xSPR (UPLO,	N,	ALPHA,	X,	INCX,	AP)		
xSYR2 (UPLO,	N,	ALPHA,	X,	INCX,	Y, INCY,	A, LDA)	
xSPR2 (UPLO,	N,	ALPHA,	X,	INCX,	Y, INCY,	AP)	

$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$
 $y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $y \leftarrow \alpha Ax + \beta y$
 $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$
 $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$
 $x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$
 $x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$
 $x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$
 $x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$

S, D, C, Z
 S, D, C, Z
 C, Z
 C, Z
 C, Z
 S, D
 S, D
 S, D
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z

$A \leftarrow \alpha xy^T + A, A - m \times n$
 $A \leftarrow \alpha xy^T + A, A - m \times n$
 $A \leftarrow \alpha xy^H + A, A - m \times n$
 $A \leftarrow \alpha xx^H + A$
 $A \leftarrow \alpha xx^H + A$
 $A \leftarrow \alpha xy^H + y(\alpha x)^H + A$
 $A \leftarrow \alpha xy^H + y(\alpha x)^H + A$
 $A \leftarrow \alpha xx^T + A$
 $A \leftarrow \alpha xx^T + A$
 $A \leftarrow \alpha xy^T + \alpha yx^T + A$
 $A \leftarrow \alpha xy^T + \alpha yx^T + A$

S, D
 C, Z
 C, Z
 C, Z
 C, Z
 C, Z
 C, Z
 S, D
 S, D
 S, D

Level 3 BLAS

options	dim	scalar	matrix	matrix	scalar	matrix
xGEMM (TRANSA, TRANSB,	M, N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)
xSYMM (SIDE, UPLO,	M, N,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)
xHEMM (SIDE, UPLO,	M, N,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)
xSYRK (UPLO, TRANS,	N, K,	ALPHA,	A, LDA,	BETA,	C, LDC)	
xHERK (UPLO, TRANS,	N, K,	ALPHA,	A, LDA,	BETA,	C, LDC)	
xSYR2K(UPLO, TRANS,	N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)
xHER2K(UPLO, TRANS,	N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)
xTRMM (SIDE, UPLO, TRANSA,	DIAG, M, N,	ALPHA,	A, LDA,	B, LDB)		
xTRSM (SIDE, UPLO, TRANSA,	DIAG, M, N,	ALPHA,	A, LDA,	B, LDB)		

$C \leftarrow \alpha op(A) op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$
 $C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^T$
 $C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$
 $C \leftarrow \alpha AA^T + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$
 $C \leftarrow \alpha AA^H + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$
 $C \leftarrow \alpha AB^T + \alpha BA^T + \beta C, C \leftarrow \alpha A^T B + \alpha B^T A + \beta C, C - n \times n$
 $C \leftarrow \alpha AB^H + \alpha BA^H + \beta C, C \leftarrow \alpha A^H B + \alpha B^H A + \beta C, C - n \times n$
 $B \leftarrow \alpha op(A) B, B \leftarrow \alpha B op(A), op(A) = A, A^T, A^H, B - m \times n$
 $B \leftarrow \alpha op(A^{-1}) B, B \leftarrow \alpha B op(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$

S, D, C, Z
 S, D, C, Z
 C, Z
 S, D, C, Z
 C, Z
 S, D, C, Z
 C, Z
 S, D, C, Z
 S, D, C, Z
 S, D, C, Z

A brief history of (Dense) Linear Algebra software

LAPACK - "Linear Algebra PACKage" - uses BLAS-3 (1989 - now)

- Ex: Obvious way to express Gaussian Elimination (GE) is adding multiples of one row to other rows - BLAS-1
 - How do we reorganize GE to use BLAS-3 ? (details later)
- Contents of LAPACK (summary)
 - Algorithms we can turn into (nearly) 100% BLAS 3
 - Linear Systems: solve $Ax=b$ for x
 - Least Squares: choose x to minimize $\|Ax - b\|_2$
 - Algorithms that are only 50% BLAS 3 (so far)
 - "Eigenproblems": Find λ and x where $Ax = \lambda x$
 - Singular Value Decomposition (SVD): $(A^T A)x = \sigma^2 x$
 - Generalized problems (eg $Ax = \lambda Bx$)
 - Error bounds for everything
 - Lots of variants depending on A 's structure (banded, $A=A^T$, etc)
- How much code? (Release 3.4, Nov 2011) (www.netlib.org/lapack)
 - Source: 1674 routines, 490K LOC, Testing: 448K LOC

A brief history of (Dense) Linear Algebra software

- .. **Is LAPACK parallel?**
 - Only if the BLAS are parallel (possible in shared memory)
- .. **ScaLAPACK - “Scalable LAPACK” (1995 - now)**
 - For distributed memory - uses MPI
 - More complex data structures, algorithms than LAPACK
 - Only (small) subset of LAPACK's functionality available
 - All at www.netlib.org/scalapack

LAPACK

- .. <http://www.netlib.org/lapack/>
- .. LAPACK (Linear Algebra Package) provides routines for
 - solving systems of simultaneous linear equations,
 - least-squares solutions of linear systems of equations,
 - eigenvalue problems,
 - and singular value problems.
- .. LAPACK relies on BLAS
- .. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers.
- .. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

LAPACK is in
FORTRAN
Column Major

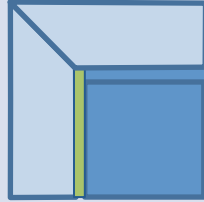
LAPACK is
SEQUENTIAL

LAPACK is a
REFERENCE
implementation

A new generation of algorithms?

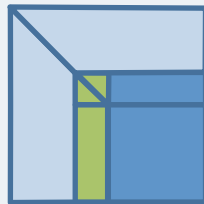
Algorithms follow hardware evolution along time.

LINPACK (80's)
(Vector operations)



Rely on
- Level-1 BLAS operations

LAPACK (90's)
(Blocking, cache friendly)



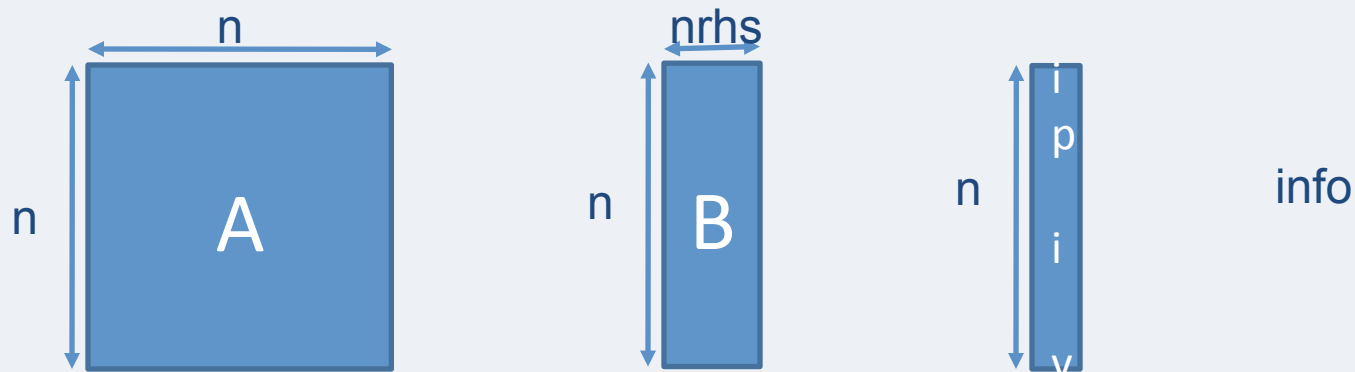
Rely on
- Level-3 BLAS operations

Example with GESV

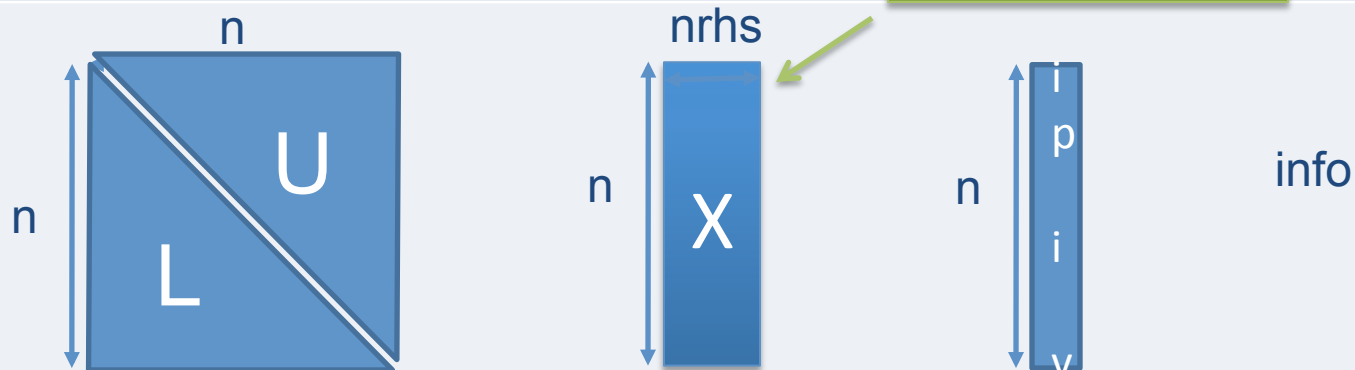
Solve a system of linear equations using a LU factorization

```
subroutine dgesv( n, nrhs, A, lda, ipiv, b, ldb, info )
```

input:



output:



Functionalities in LAPACK

Type of Problem	Acronyms
Linear system of equations	SV
Linear least squares problems	LLS
Linear equality-constrained least squares problem	LSE
General linear model problem	GLM
Symmetric eigenproblems	SEP
Nonsymmetric eigenproblems	NEP
Singular value decomposition	SVD
Generalized symmetric definite eigenproblems	GSEP
Generalized nonsymmetric eigenproblems	GNEP
Generalized (or quotient) singular value decomposition	GSVD (QSVD)

LAPACK Software

- First release in February **1992**
- Version **3.4.0** released in November 2011
- **LICENSE:** Mod-BSD, freely-available software package - Thus, it can be included in commercial software packages (and has been). We only ask that proper credit be given to the authors.
- Open SVN repository
- **Multi-OS**
 - *nix, Mac OS/X, Windows
- **Multi-build** support (cmake)
 - make, xcode, nmake, VS studio, Eclipse, etc..
- **LAPACKe: Standard C language APIs for LAPACK** (In collaboration with INTEL)
 - **2 layers of interface**
 - **High-Level Interface** : Workspace allocation and NAN Check
 - **Low-Level Interface**
- Prebuilt Libraries for **Windows**
- Extensive test suite
- **Forum and User support:** <http://icl.cs.utk.edu/lapack-forum/>

Latest Algorithms

Since release 3.0 of LAPACK

3.1

- Hessenberg QR algorithm with the small bulge multi-shift QR algorithm together with aggressive early deflation. [**2003 SIAM SIAG LA Prize** winning algorithm of Braman, Byers and Mathias]
- Improvements of the Hessenberg reduction subroutines. [G. Quintana-Ortí and van de Geijn]
- New MRRR eigenvalue algorithms [**2006 SIAM SIAG LA Prize** winning algorithm of Dhillon and Parlett]
- New partial column norm updating strategy for QR factorization with column pivoting. [Drmač and Bujanovic]
- Mixed Precision Iterative Refinement for exploiting fast single precision hardware for GE, PO [Langou's]

3.2

- Variants of various factorization (LU, QR, Chol) [Du]
- RFP (Rectangular Full Packed) format [Gustavson, Langou]
- XBLAS and Extra precise iterative refinement for GESV [Demmel et al.].
- New fast and accurate Jacobi SVD [**2009 SIAM SIAG LA Prize**, Drmač and Veselić]
- Pivoted Cholesky [Lucas]
- Better multishift Hessenberg QR algorithm with early aggressive deflation [Byers]

3.3

- Complete CS decomposition [Sutton]
- Level-3 BLAS symmetric indefinite solve and symmetric indefinite inversion [Langou's]
- Since LAPACK 3.3, all routines in are now thread-safe

LAPACK 3.4.0

- xGEQRT: QR factorization (improved interface).
Contribution by Rodney James, UC Denver.
xGEQRT is analogous to xGEQRF with a modified interface which enables better performance when the blocked reflectors need to be reused. The companion subroutines xGEMQRT apply the reflectors.
- xGEQRT3: recursive QR factorization.
Contribution by Rodney James, UC Denver.
The recursive QR factorization enable cache-oblivious and enable high performance on tall and skinny matrices.
- xTPQRT: Communication-Avoiding QR sequential kernels.
Contribution by Rodney James, UC Denver.
These subroutines are useful for updating a QR factorization and are used in sequential and parallel Communication Avoiding QR. These subroutines support the general case Triangle on top of Pentagone which includes as special cases the so-called triangle on top of triangle and triangle on top of square. This is the right-looking version of the subroutines and the routine is blocked. The T matrices and the block size are part of the interface. The companion subroutines xTPMQRT apply the reflectors.
- xSYEVK: LDLT with rook pivoting and fast Bunch-Parlett pivoting.
Contribution by Craig Lucas.
These subroutines enables better stability than the Bunch-Kaufman pivoting scheme (xSYEV) currently used in LAPACK. The computational time is slightly higher.

Resources

Reference Code:

- Reference code: (current version 3.3.1)
<http://www.netlib.org/lapack/lapack.tgz>
- LAPACK build for windows (current version 3.3.1)
<http://icl.cs.utk.edu/lapack-for-windows/lapack>
- LAPACKe: Standard C language APIs for LAPACK (in collaboration with INTEL):
http://www.netlib.org/lapack/#_standard_c_language_apis_for_lapack
- Remi's wrappers (wrapper for Matlab users):
<http://icl.cs.utk.edu/~delmas/lapwrapmw.htm>

Vendor Libraries:

more or less same as the BLAS: MKL, ACML, VECLIB, ESSL, etc... (**WARNING**: some implementations are just a subset of LAPACK)

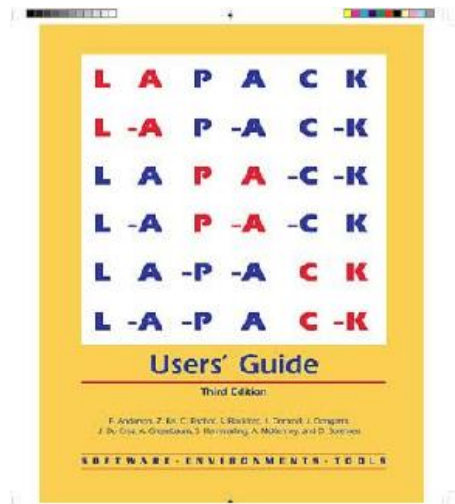
Documentation:

- LAPACK Users' guide:
<http://www.netlib.org/lapack/lug/>
- LAPACK Working notes (in particular LAWN 41)
<http://www.netlib.org/lapack/lawns/downloads/>
- LAPACK release notes
<http://www.netlib.org/lapack/lapack-3.1.0.changes>
- LAPACK NAG example and auxiliary routines
<http://www.nag.com/lapack-ex/lapack-ex.html>
- **CRC Handbook of Linear Algebra, Leslie Hogben ed, Packages of Subroutines for Linear Algebra**, Bai, Demmel, Dongarra, Langou, and Wang, Section 75: pages 75-1,75-24, CRC Press, 2006.
<http://www.netlib.org/netlib/utk/people/JackDongarra/PAPERS/CRC-LAPACK-2005.pdf>

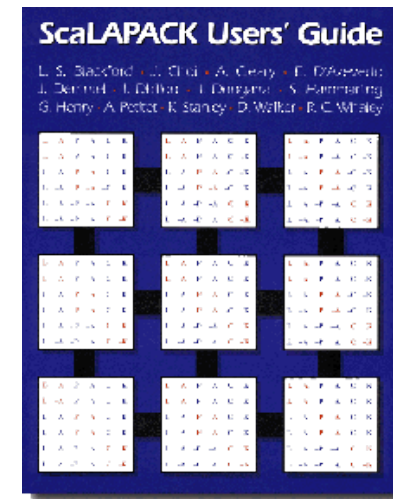
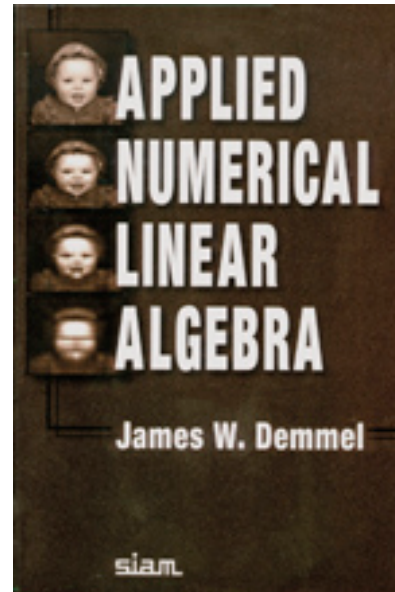
Support:

- LAPACK forum: (more than 1000 topics)
<http://icl.cs.utk.edu/lapack-forum/>
- LAPACK mailing-list:
lapack@cs.utk.edu
- LAPACK mailing-list archive:
<http://icl.cs.utk.edu/lapack-forum/archives/>

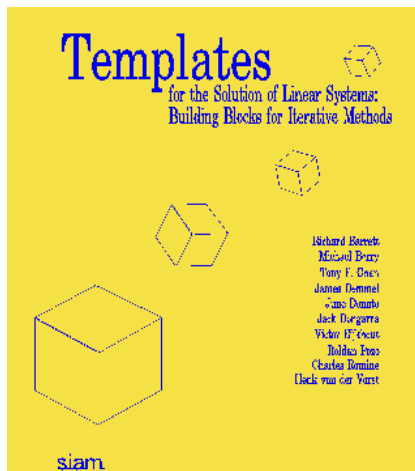
Organizing Linear Algebra – in books



www.netlib.org/lapack



www.netlib.org/scalapack



www.netlib.org/templates



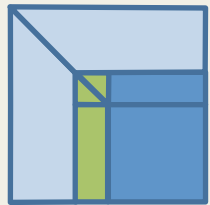
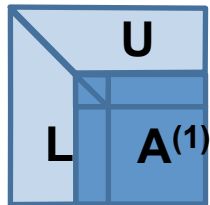
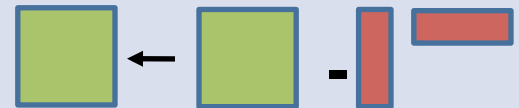
www.cs.utk.edu/~dongarra/etemplates

Parallelization of LU and QR.

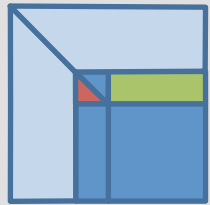
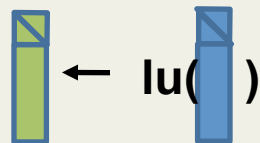
Parallelize the update:

- Easy and done in any reasonable software.
- This is the $2/3n^3$ term in the FLOPs count.
- Can be done efficiently with LAPACK+multithreaded BLAS

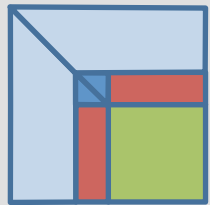
dgemm



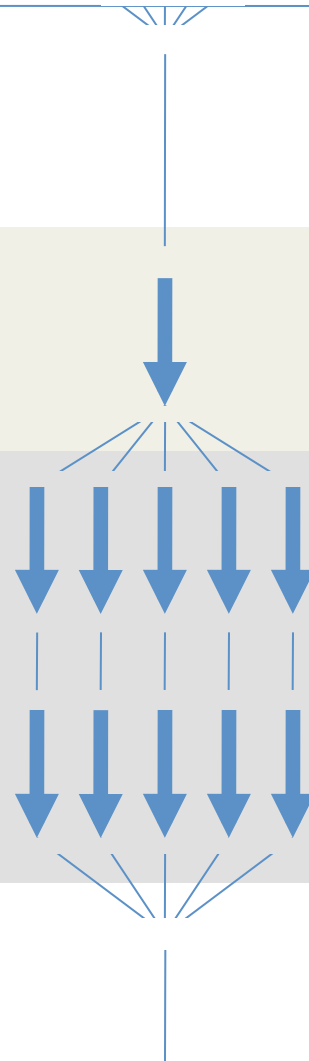
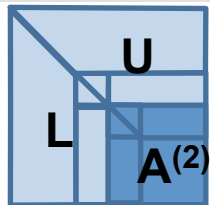
dgetf2



dtrsm (+ dswp)

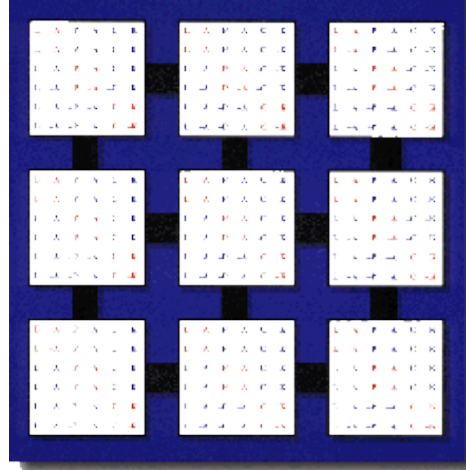


dgemm



Overview of Dense Numerical Linear Algebra Libraries

- **BLAS**: kernel for dense linear algebra
- **LAPACK**: sequential dense linear algebra
- **ScaLAPACK**: parallel distributed dense linear algebra



Scalable Linear Algebra PACKage

- Library of software dealing with dense & banded routines
- Distributed Memory - Message Passing
- MIMD Computers and Networks of Workstations
- Clusters of SMPs

ScaLAPACK

- <http://www.netlib.org/scalapack/>
- ScaLAPACK (Scalable Linear Algebra Package) provides routines for
 - solving systems of simultaneous linear equations,
 - least-squares solutions of linear systems of equations,
 - eigenvalue problems,
 - and singular value problems.
- Relies on LAPACK / BLAS and BLACS / MPI
- Includes PBLAS (Parallel BLAS)

ScaLAPACK is in
FORTRAN and C

ScaLAPACK is for
PARALLEL
DISTRIBUTED

ScaLAPACK is a
REFERENCE
implementation

Programming Style


- .. **SPMD Fortran 77 with object based design**
- .. **Built on various modules**
 - **PBLAS Interprocessor communication**
 - **BLACS**
 - PVM, MPI, IBM SP, CRI T3, Intel, TMC
 - Provides right level of notation.
 - **BLAS**
- .. **LAPACK software expertise/quality**
 - **Software approach**
 - **Numerical methods**

Overall Structure of Software

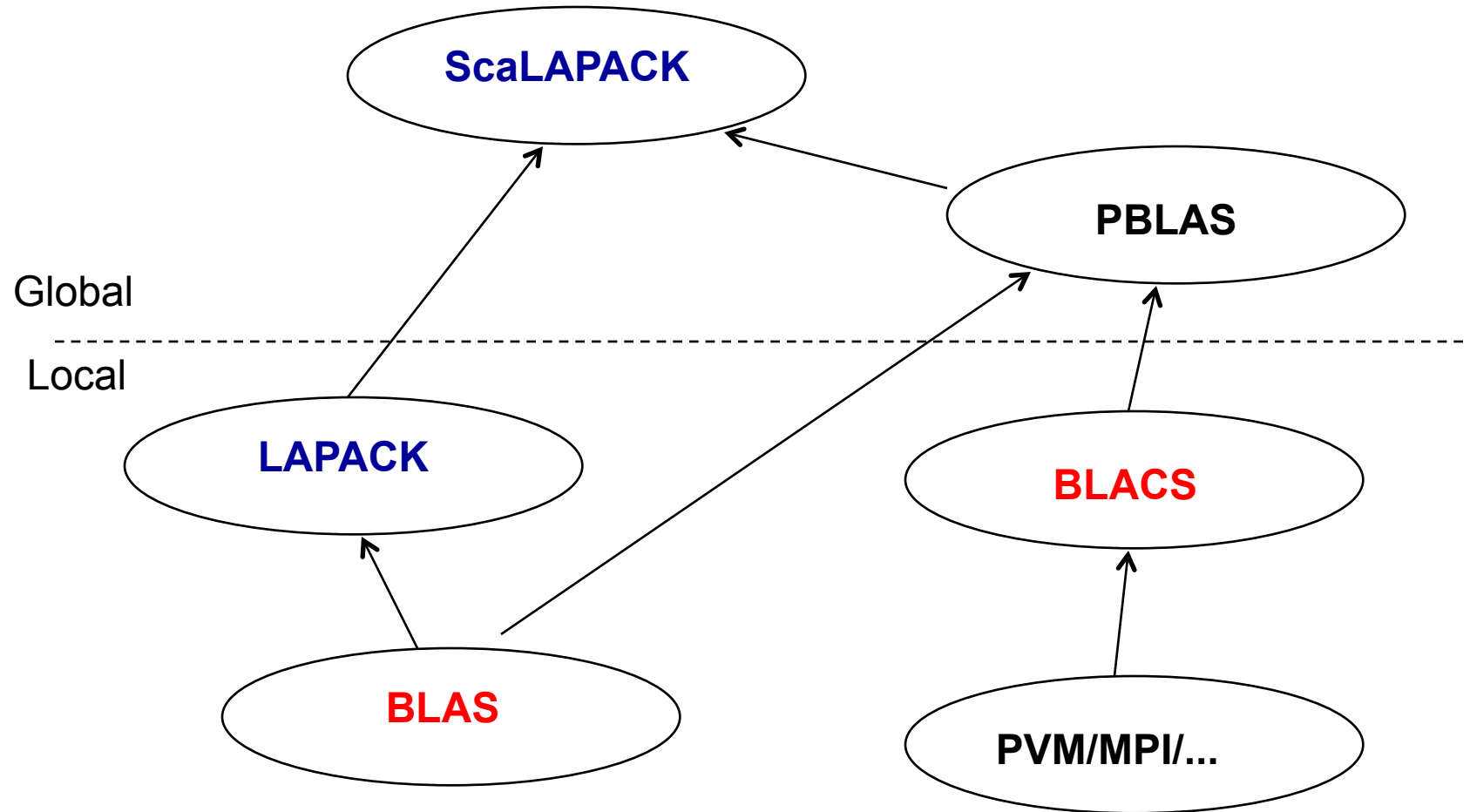
.. Object based - Array descriptor

- Contains information required to establish mapping between a global array entry and its corresponding process and memory location.
- Provides a flexible framework to easily specify additional data distributions or matrix types.
- Currently dense, banded, & out-of-core

.. Using the concept of context

- Similar to the BLAS in functionality and naming.
- Built on the BLAS and BLACS
- Provide global view of matrix
`CALL DGEXXX (M, N, A(IA, JA), LDA,...)`

`CALL PDGEXXX(M, N, A, IA, JA, DESCA,...)`

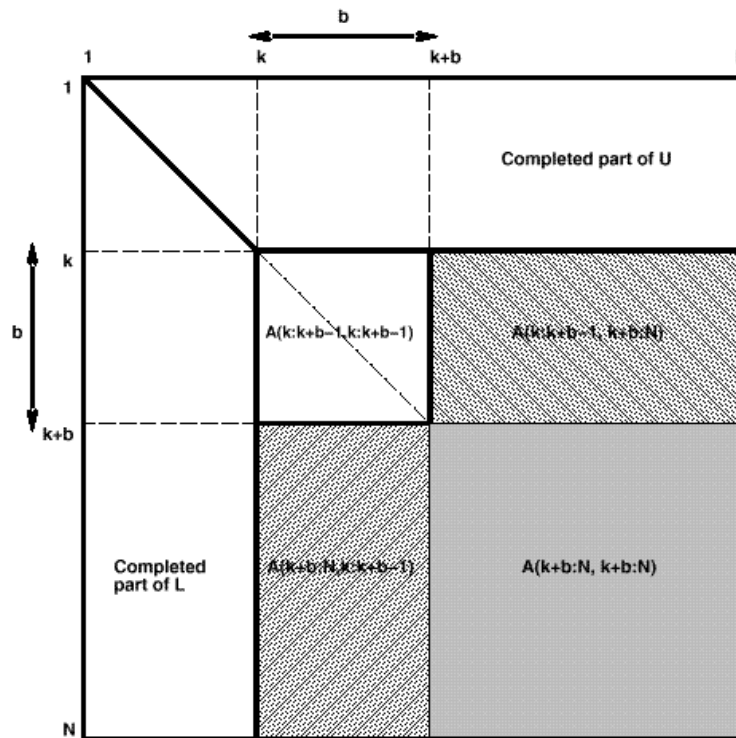
ScaLAPACK Structure



Choosing a Data Distribution

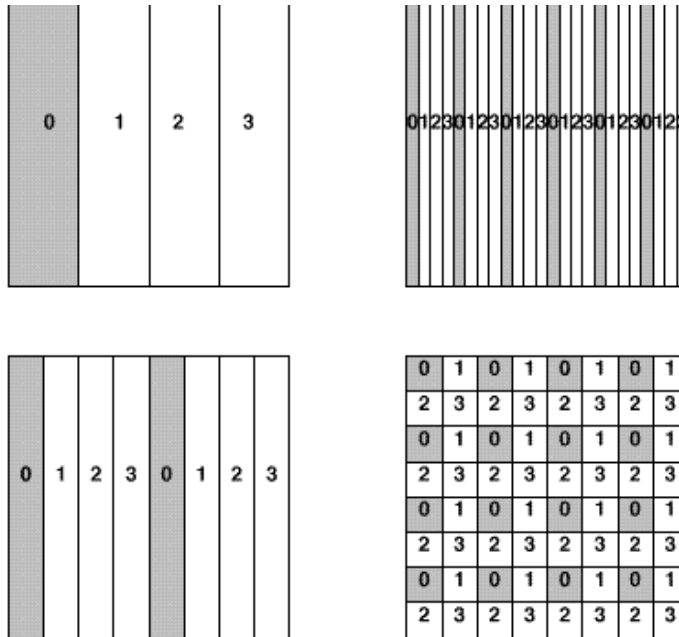
.. **Main issues are:**

- **Load balancing**
- **Use of the Level 3 BLAS**



Possible Data Layouts

- 1D block and cyclic column distributions

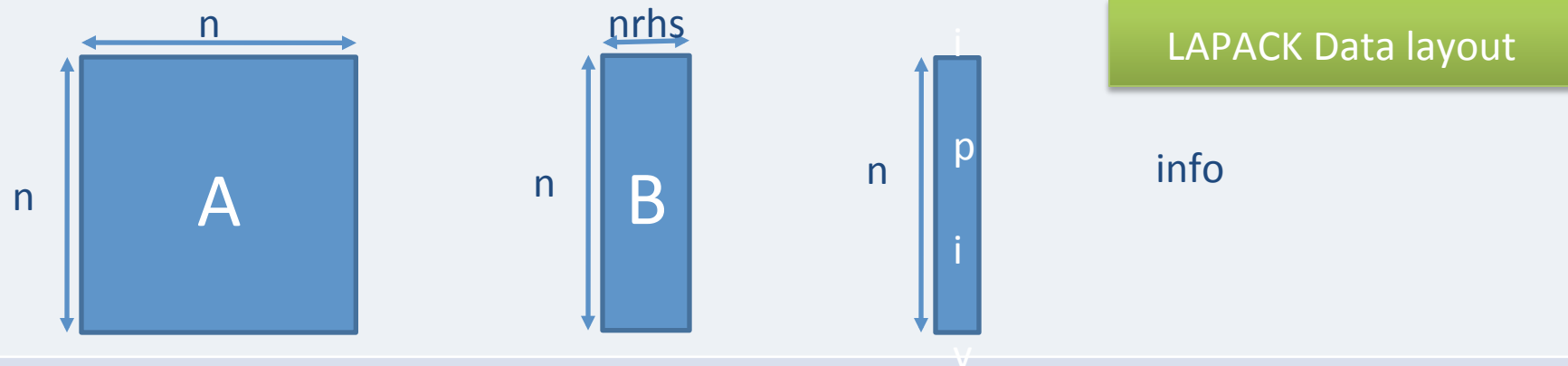


- 1D block-cycle column and 2D block-cyclic distribution
- 2D block-cyclic used in ScaLAPACK for dense matrices

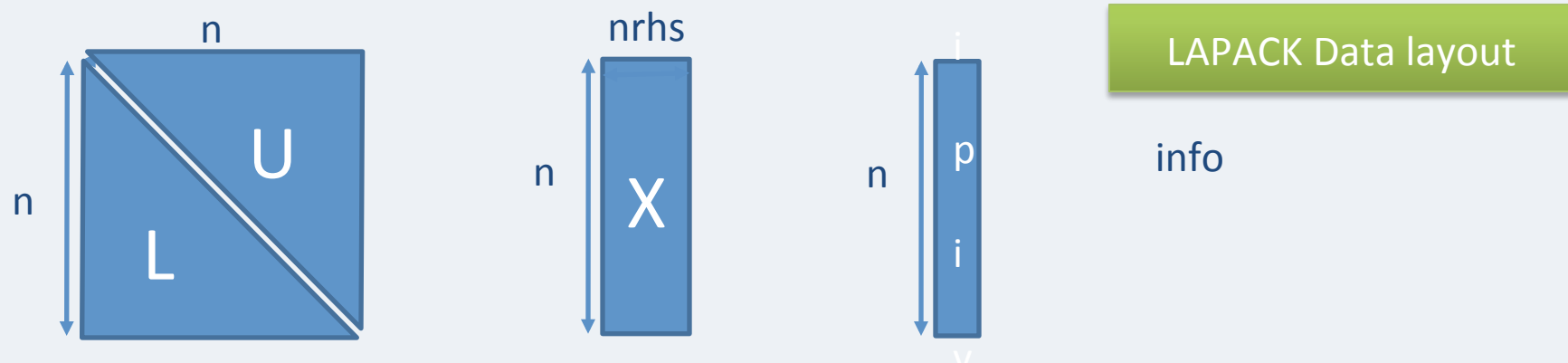
From LAPACK to ScaLAPACK

[LAPACK] subroutine **dgesv**(n, nrhs, a(ia,ja), lda, ipiv, b(ib,jb), ldb, info)

input:



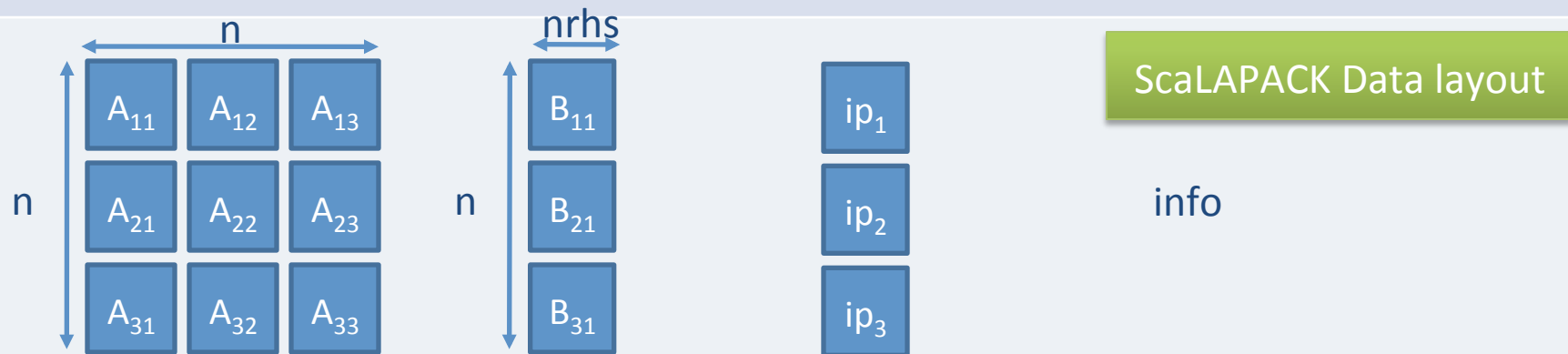
output:



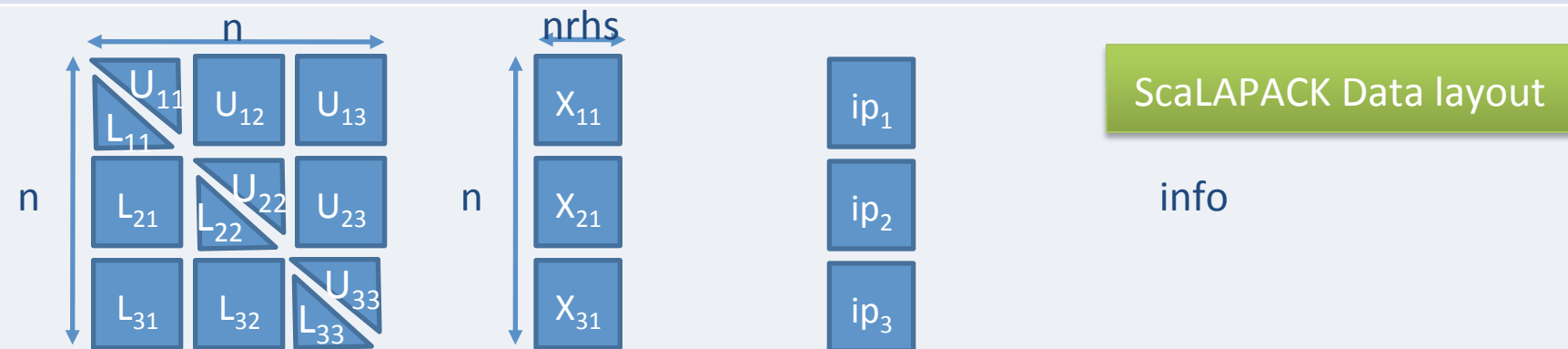
From LAPACK to ScaLAPACK

[LAPACK] subroutine **dgesv**(n, nrhs, a(ia,ja), lda, ipiv, b(ib,jb), ldb, info)

input:



output:

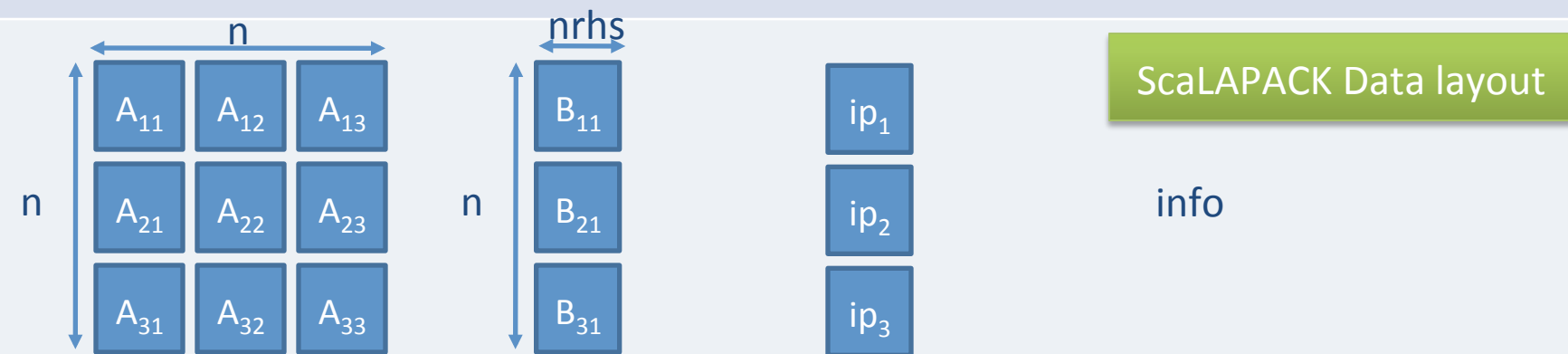


From LAPACK to ScaLAPACK

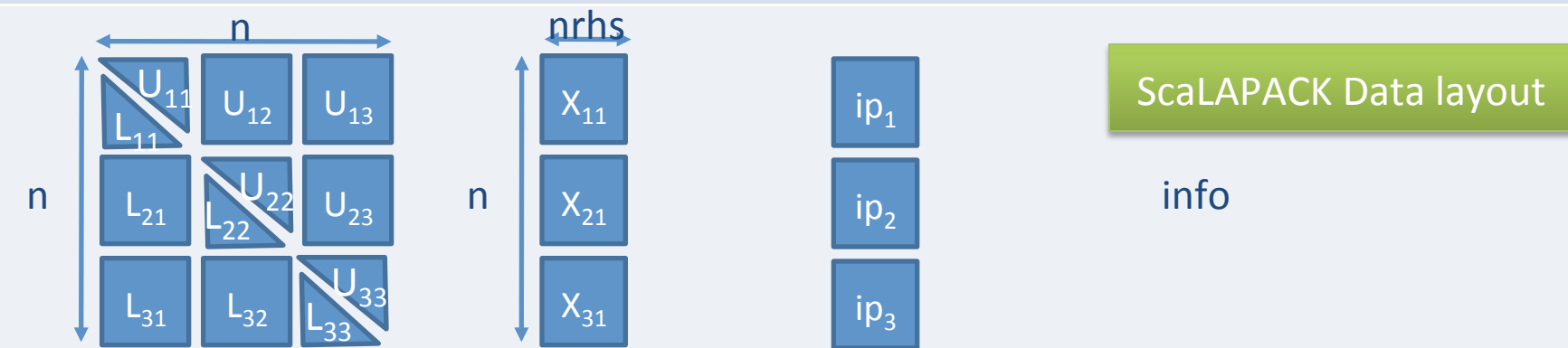
[LAPACK] subroutine **dgesv**(n, nrhs, a(ia,ja), lda, ipiv, b(ib,jb), ldb, info)

[ScaLAPACK] subroutine **pdgesv**(n, nrhs, a, ia, ja, desca, ipiv, b, ib, jb, descb, info)

input:



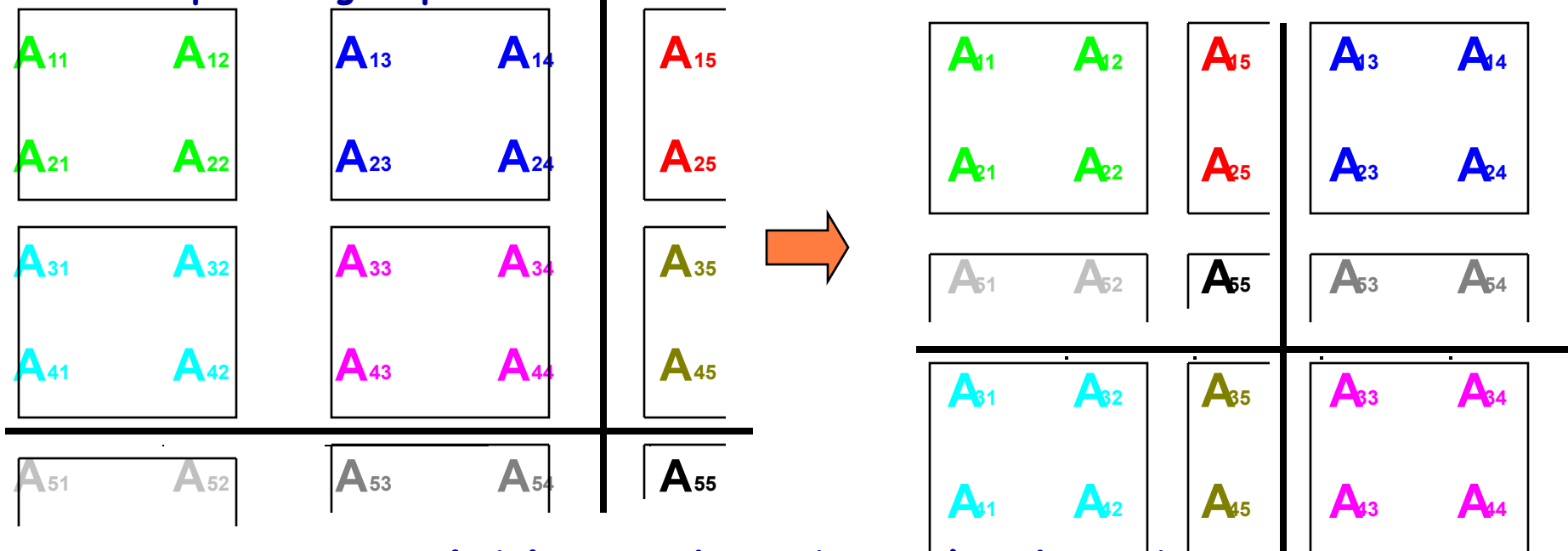
output:



Distribution and Storage

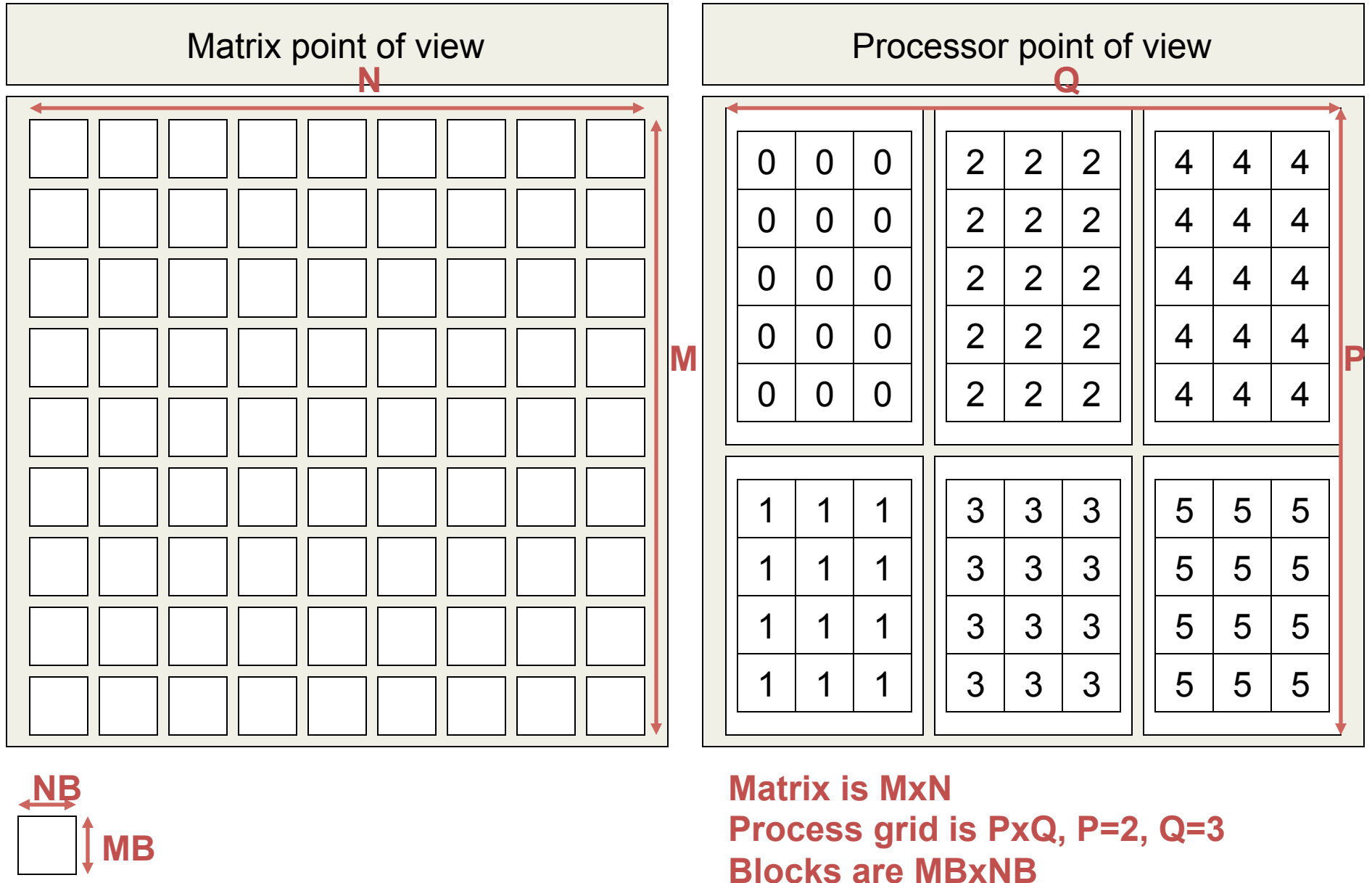
- Matrix is block-partitioned & maps blocks
- Distributed 2-D block-cyclic scheme

5x5 matrix partitioned in 2x2 blocks
2x2 process grid point of view



- Routines available to distribute/redistribute data.

2D Block Cyclic Layout



2D Block Cyclic Layout

Matrix point of view

Processor point of view

0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5

2D Block Cyclic Layout

Matrix point of view

0	2	4						
1	3	5						

Processor point of view

0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5

2D Block Cyclic Layout

Matrix point of view

0	2	4	0	2	4			
1	3	5	1	3	5			

Processor point of view

0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5

2D Block Cyclic Layout

Matrix point of view

0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4

Processor point of view

<table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>2</td><td>2</td><td>2</td></tr></table>	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	<table><tr><td>4</td><td>4</td><td>4</td></tr><tr><td>4</td><td>4</td><td>4</td></tr><tr><td>4</td><td>4</td><td>4</td></tr><tr><td>4</td><td>4</td><td>4</td></tr><tr><td>4</td><td>4</td><td>4</td></tr></table>	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
0	0	0																																													
0	0	0																																													
0	0	0																																													
0	0	0																																													
0	0	0																																													
2	2	2																																													
2	2	2																																													
2	2	2																																													
2	2	2																																													
2	2	2																																													
4	4	4																																													
4	4	4																																													
4	4	4																																													
4	4	4																																													
4	4	4																																													
<table><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	1	1	1	<table><tr><td>3</td><td>3</td><td>3</td></tr><tr><td>3</td><td>3</td><td>3</td></tr><tr><td>3</td><td>3</td><td>3</td></tr><tr><td>3</td><td>3</td><td>3</td></tr></table>	3	3	3	3	3	3	3	3	3	3	3	3	<table><tr><td>5</td><td>5</td><td>5</td></tr><tr><td>5</td><td>5</td><td>5</td></tr><tr><td>5</td><td>5</td><td>5</td></tr><tr><td>5</td><td>5</td><td>5</td></tr></table>	5	5	5	5	5	5	5	5	5	5	5	5									
1	1	1																																													
1	1	1																																													
1	1	1																																													
1	1	1																																													
3	3	3																																													
3	3	3																																													
3	3	3																																													
3	3	3																																													
5	5	5																																													
5	5	5																																													
5	5	5																																													
5	5	5																																													

2D Block Cyclic Layout

Matrix point of view

0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4

Processor point of view

0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
0	0	0	2	2	2	4	4	4
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5
1	1	1	3	3	3	5	5	5

2D Block Cyclic Layout

Matrix point of view


















0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4
1	3	5	1	3	5	1	3	5
0	2	4	0	2	4	0	2	4

Processor point of view















<table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<table><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>2</td><td>2</td><td>2</td></tr><tr><td>2</td><td>2</td><td>2</td></tr></table>	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	<table><tr><td>4</td><td>4</td><td>4</td></tr><tr><td>4</td><td>4</td><td>4</td></tr><tr><td>4</td><td>4</td><td>4</td></tr><tr><td>4</td><td>4</td><td>4</td></tr><tr><td>4</td><td>4</td><td>4</td></tr></table>	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
0	0	0																																													
0	0	0																																													
0	0	0																																													
0	0	0																																													
0	0	0																																													
2	2	2																																													
2	2	2																																													
2	2	2																																													
2	2	2																																													
2	2	2																																													
4	4	4																																													
4	4	4																																													
4	4	4																																													
4	4	4																																													
4	4	4																																													
<table><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	1	1	1	<table><tr><td>3</td><td>3</td><td>3</td></tr><tr><td>3</td><td>3</td><td>3</td></tr><tr><td>3</td><td>3</td><td>3</td></tr><tr><td>3</td><td>3</td><td>3</td></tr></table>	3	3	3	3	3	3	3	3	3	3	3	3	<table><tr><td>5</td><td>5</td><td>5</td></tr><tr><td>5</td><td>5</td><td>5</td></tr><tr><td>5</td><td>5</td><td>5</td></tr><tr><td>5</td><td>5</td><td>5</td></tr></table>	5	5	5	5	5	5	5	5	5	5	5	5									
1	1	1																																													
1	1	1																																													
1	1	1																																													
1	1	1																																													
3	3	3																																													
3	3	3																																													
3	3	3																																													
3	3	3																																													
5	5	5																																													
5	5	5																																													
5	5	5																																													
5	5	5																																													

2D Block Cyclic Layout

Matrix point of view

































								
	3	5	1	3	5	1	3	5
	2	4	0	2	4	0	2	4
	3	5	1	3	5	1	3	5
	2	4	0	2	4	0	2	4
	3	5	1	3	5	1	3	5
	2	4	0	2	4	0	2	4
	3	5	1	3	5	1	3	5
	2	4	0	2	4	0	2	4

Processor point of view








											
	0	0	2			2			4		
	0	0	2			2			4		
	0	0	2			2			4		
	0	0	2			2			4		
	1	1	3			3			5		
	1	1	3			3			5		
	1	1	3			3			5		
	1	1	3			3			5		








2D Block Cyclic Layout




Matrix point of view

								
								
		4	0	2	4	0	2	4
		5	1	3	5	1	3	5
		4	0	2	4	0	2	4
		5	1	3	5	1	3	5
		4	0	2	4	0	2	4
		5	1	3	5	1	3	5
		4	0	2	4	0	2	4







Processor point of view




		
	0	0
	0	0
	0	0
	0	0

		
	2	2
	2	2
	2	2
	2	2

		
4	4	4
4	4	4
4	4	4
4	4	4

	1	1
	1	1
	1	1

		
	3	3
	3	3
	3	3

		
5	5	5
5	5	5
5	5	5

2D Block Cyclic Layout

Matrix point of view

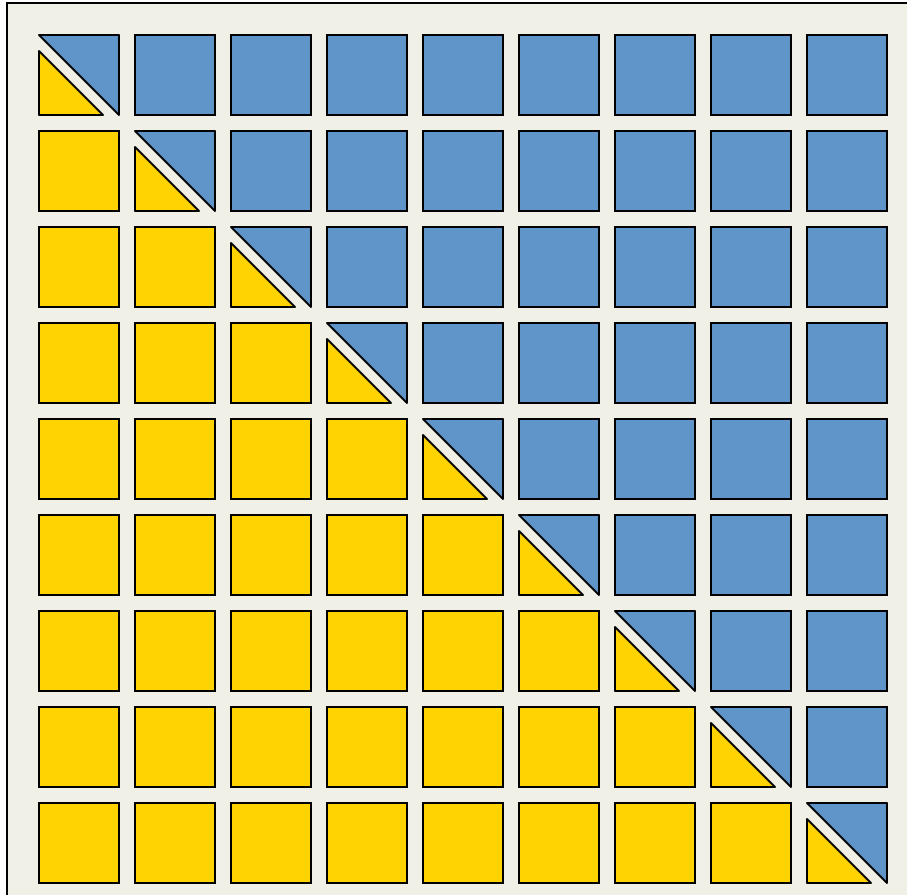
			1	3	5	1	3	5
			0	2	4	0	2	4
			1	3	5	1	3	5
			0	2	4	0	2	4
			1	3	5	1	3	5
			0	2	4	0	2	4

Processor point of view

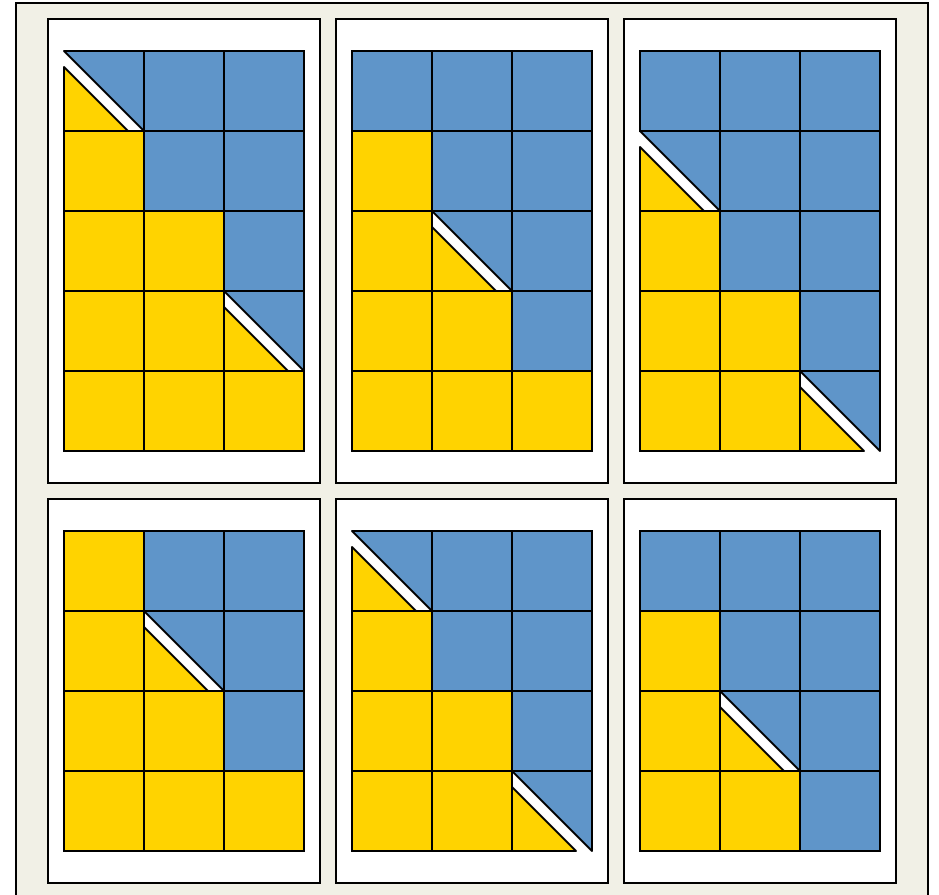
<table> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>0</td><td></td></tr> </table>							0	0		0	0		0	0		<table> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td>2</td><td>2</td><td></td></tr> <tr><td>2</td><td>2</td><td></td></tr> <tr><td>2</td><td>2</td><td></td></tr> </table>							2	2		2	2		2	2		<table> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td>4</td><td>4</td><td></td></tr> <tr><td>4</td><td>4</td><td></td></tr> <tr><td>4</td><td>4</td><td></td></tr> </table>							4	4		4	4		4	4	
0	0																																														
0	0																																														
0	0																																														
2	2																																														
2	2																																														
2	2																																														
4	4																																														
4	4																																														
4	4																																														
<table> <tr><td></td><td></td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </table>				1	1		1	1		1	1		<table> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td>3</td><td>3</td><td></td></tr> <tr><td>3</td><td>3</td><td></td></tr> <tr><td>3</td><td>3</td><td></td></tr> </table>							3	3		3	3		3	3		<table> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td>5</td><td>5</td><td></td></tr> <tr><td>5</td><td>5</td><td></td></tr> <tr><td>5</td><td>5</td><td></td></tr> </table>							5	5		5	5		5	5				
1	1																																														
1	1																																														
1	1																																														
3	3																																														
3	3																																														
3	3																																														
5	5																																														
5	5																																														
5	5																																														

2D Block Cyclic Layout

Matrix point of view



Processor point of view



Parallelism in ScaLAPACK

- .. **Level 3 BLAS block operations**
 - All the reduction routines
- .. **Pipelining**
 - QR Algorithm, Triangular Solvers, classic factorizations
- .. **Redundant computations**
 - Condition estimators
- .. **Static work assignment**
 - Bisection
- .. **Task parallelism**
 - Sign function eigenvalue computations
- .. **Divide and Conquer**
 - Tridiagonal and band solvers, symmetric eigenvalue problem and Sign function
- .. **Cyclic reduction**
 - Reduced system in the band solver
- .. **Data parallelism**
 - Sign function

Functionalities in LAPACK

Type of Problem	Acronyms
Linear system of equations	SV
Linear least squares problems	LLS
Linear equality-constrained least squares problem	LSE
General linear model problem	GLM
Symmetric eigenproblems	SEP
Nonsymmetric eigenproblems	NEP
Singular value decomposition	SVD
Generalized symmetric definite eigenproblems	GSEP
Generalized nonsymmetric eigenproblems	GNEP
Generalized (or quotient) singular value decomposition	GSVD (QSVD)

Functionnnalities in ScaLAPACK

Type of Problem	Acronyms
Linear system of equations	SV
Linear least squares problems	LLS
Linear equality-constrained least squares problem	LSE
General linear model problem	GLM
Symmetric eigenproblems	SEP
Nonsymmetric eigenproblems	NEP
Singular value decomposition	SVD
Generalized symmetric definite eigenproblems	GSEP
Generalized nonsymmetric eigenproblems	GNEP
Generalized (or quotient) singular value decomposition	GSVD (QSVD)

Major Changes to Software

- **Must rethink the design of our software**
 - **Another disruptive technology**
 - Similar to what happened with cluster computing and message passing
 - **Rethink and rewrite the applications, algorithms, and software**
- **Numerical libraries for example will change**
 - **For example, both LAPACK and ScaLAPACK will undergo major changes to accommodate this**

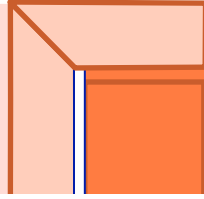


A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

Software/Algorithms follow hardware evolution in time

LINPACK (70's)
(Vector operations)



Rely on
- Level-1 BLAS
operations

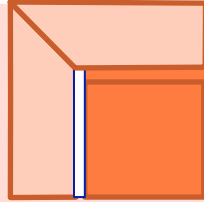


A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

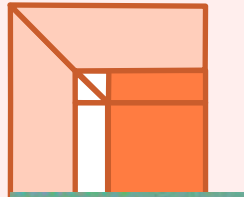
Software/Algorithms follow hardware evolution in time

LINPACK (70's)
(Vector operations)



Rely on
- Level-1 BLAS
operations

LAPACK (80's)
(Blocking, cache
friendly)



Rely on
- Level-3 BLAS
operations

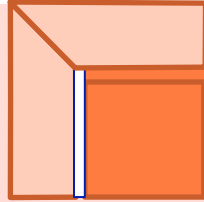


A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

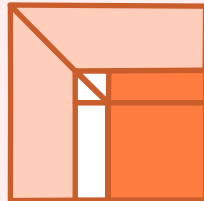
Software/Algorithms follow hardware evolution in time

LINPACK (70's)
(Vector operations)



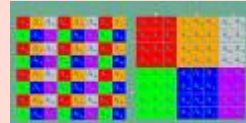
Rely on
- Level-1 BLAS
operations

LAPACK (80's)
(Blocking, cache
friendly)



Rely on
- Level-3 BLAS
operations

ScaLAPACK (90's)
(Distributed Memory)

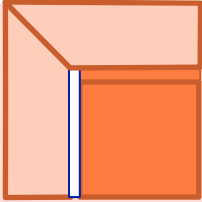

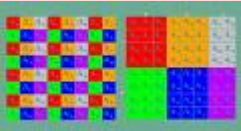



Rely on
- PBLAS Mess Passing



A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

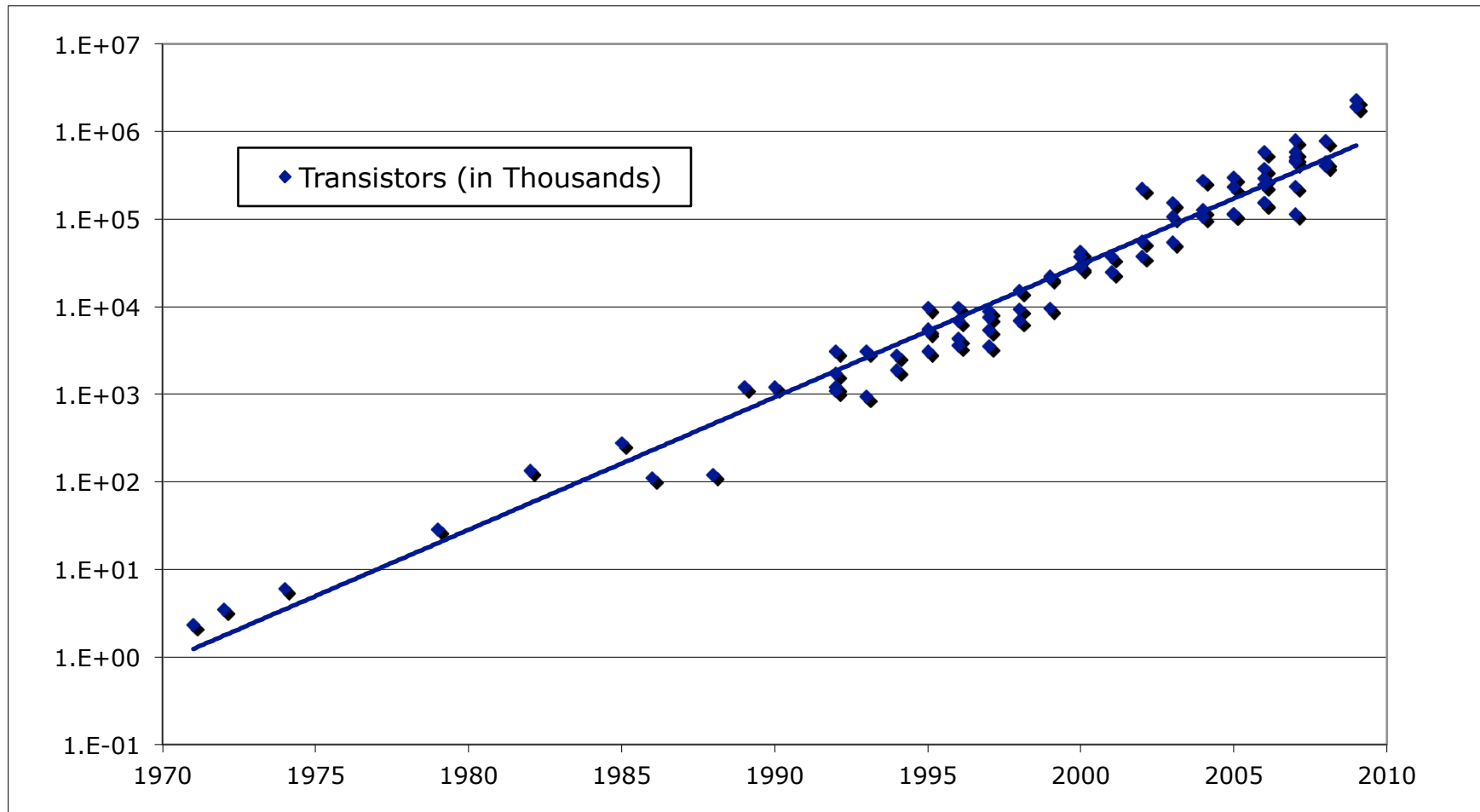
Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels

Those new algorithms

- have a very **low granularity**, they scale very well (multicore, petascale computing, ...)
- **removes a lots of dependencies** among the tasks, (multicore, distributed computing)
- **avoid latency** (distributed computing, out-of-core)
- **rely on fast kernels**

Those new algorithms need new kernels and rely on efficient scheduling algorithms.

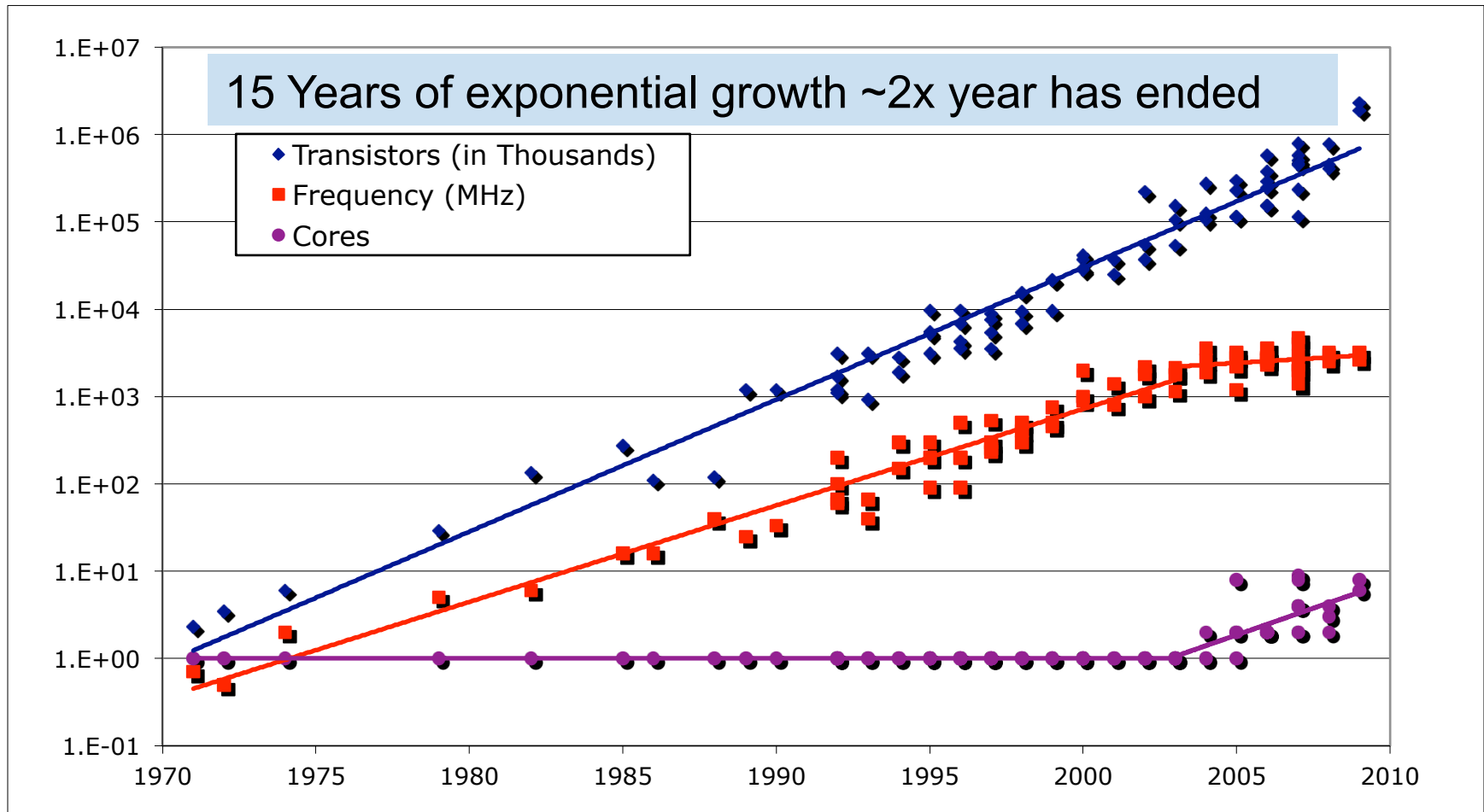
Moore's Law is Alive and Well



Data from Kunle Olukotun, Lance Hammond, Herb Sutter,
Burton Smith, Chris Batten, and Krste Asanović

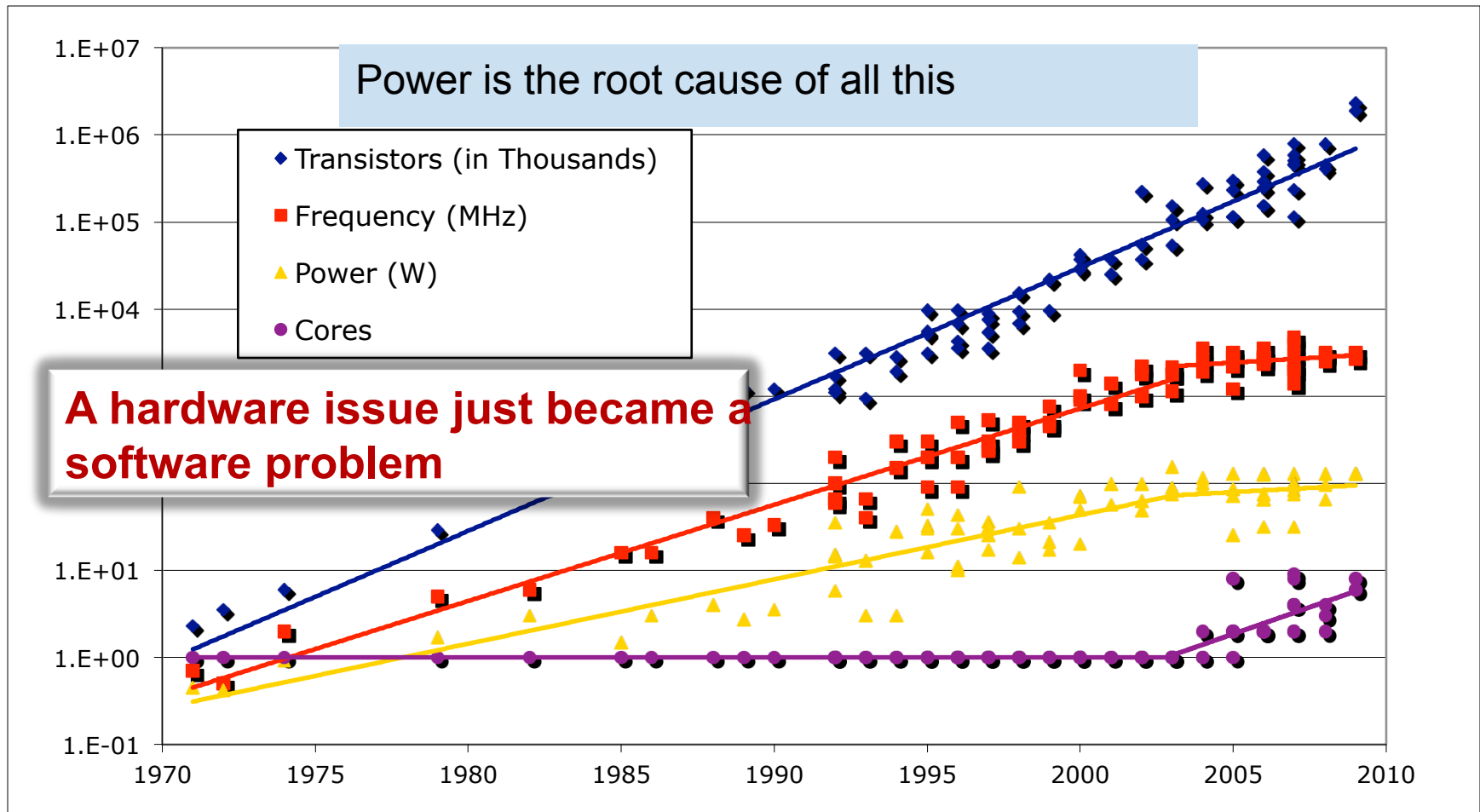


But Clock Frequency Scaling Replaced by Scaling Cores / Chip



Data from Kunle Olukotun, Lance Hammond, Herb Sutter,
Burton Smith, Chris Batten, and Krste Asanović

Performance Has Also Slowed, Along with Power



Data from Kunle Olukotun, Lance Hammond, Herb Sutter,
Burton Smith, Chris Batten, and Krste Asanović

Power Cost of Frequency

- Power \propto Voltage² x Frequency (V^2F)
- Frequency \propto Voltage
- Power \propto Frequency³

	Cores	V	Freq	Perf	Power	PE (Bops/watt)
Superscalar	1	1	1	1	1	1
"New" Superscalar	1X	1.5X	1.5X	1.5X	3.3X	0.45X

Power Cost of Frequency

- Power \propto Voltage² x Frequency (V^2F)
- Frequency \propto Voltage
- Power \propto Frequency³

	Cores	V	Freq	Perf	Power	PE (Bops/watt)
Superscalar	1	1	1	1	1	1
"New" Superscalar	1X	1.5X	1.5X	1.5X	3.3X	0.45X
Multicore	2X	0.75X	0.75X	1.5X	0.8X	1.88X

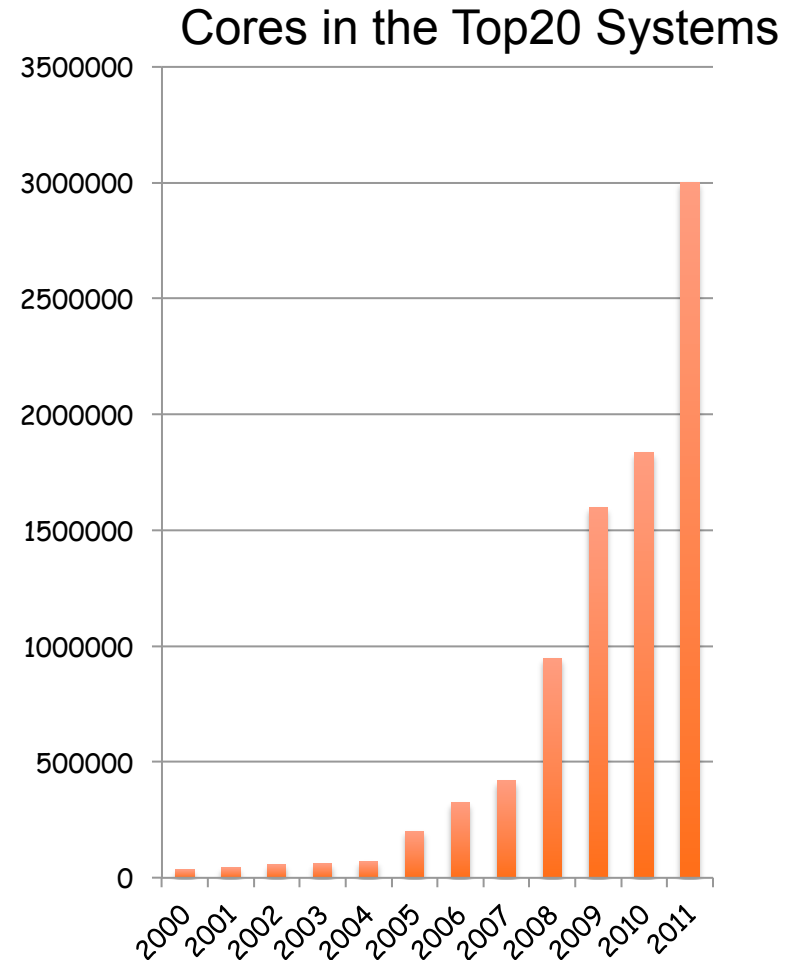
(Bigger # is better)

50% more performance with 20% less power

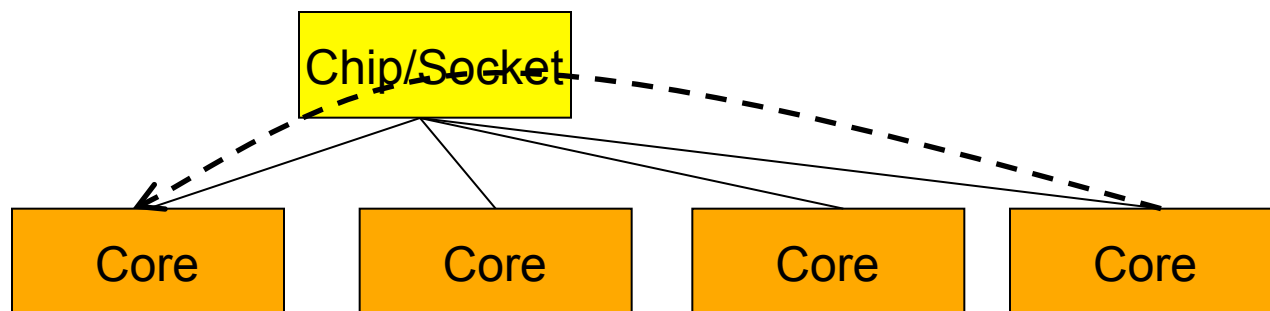
Preferable to use multiple slower devices, than one superfast device

Moore's Law Reinterpreted

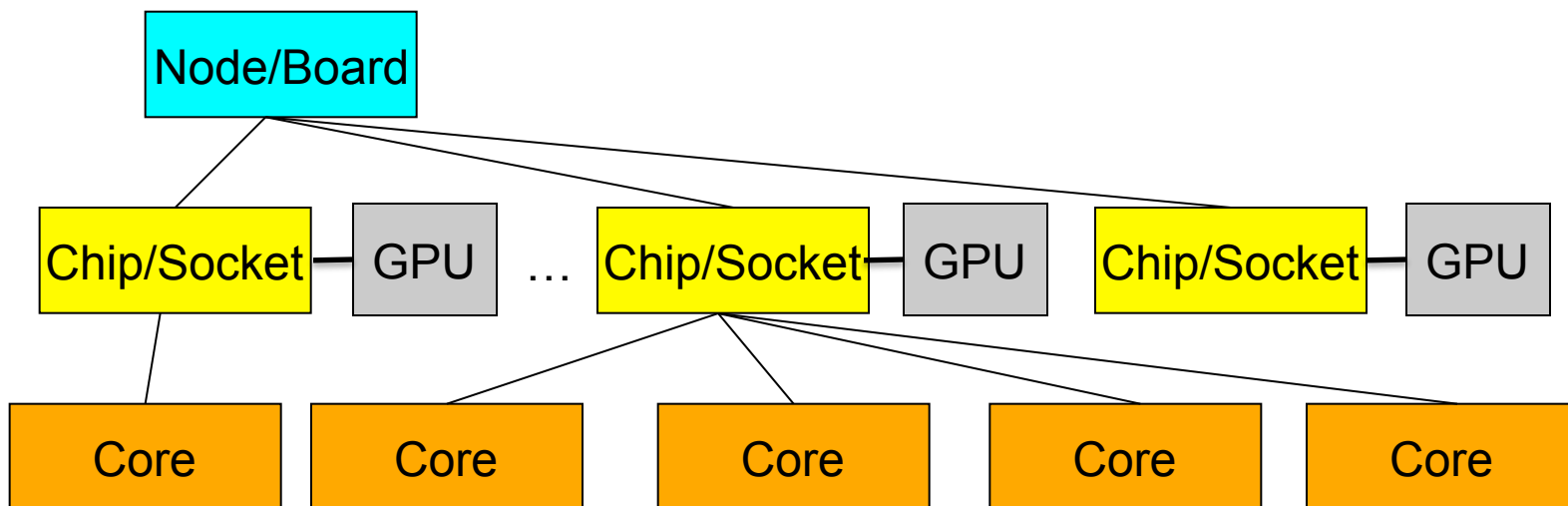
- .. **Number of cores per chip doubles every 2 year, while clock speed decreases (not increases).**
 - **Need to deal with systems with millions of concurrent threads**
 - **Future generation will have billions of threads!**
 - **Need to be able to easily replace inter-chip parallelism with intro-chip parallelism**
- .. **Number of threads of execution doubles every 2 year**



Example of typical parallel machine

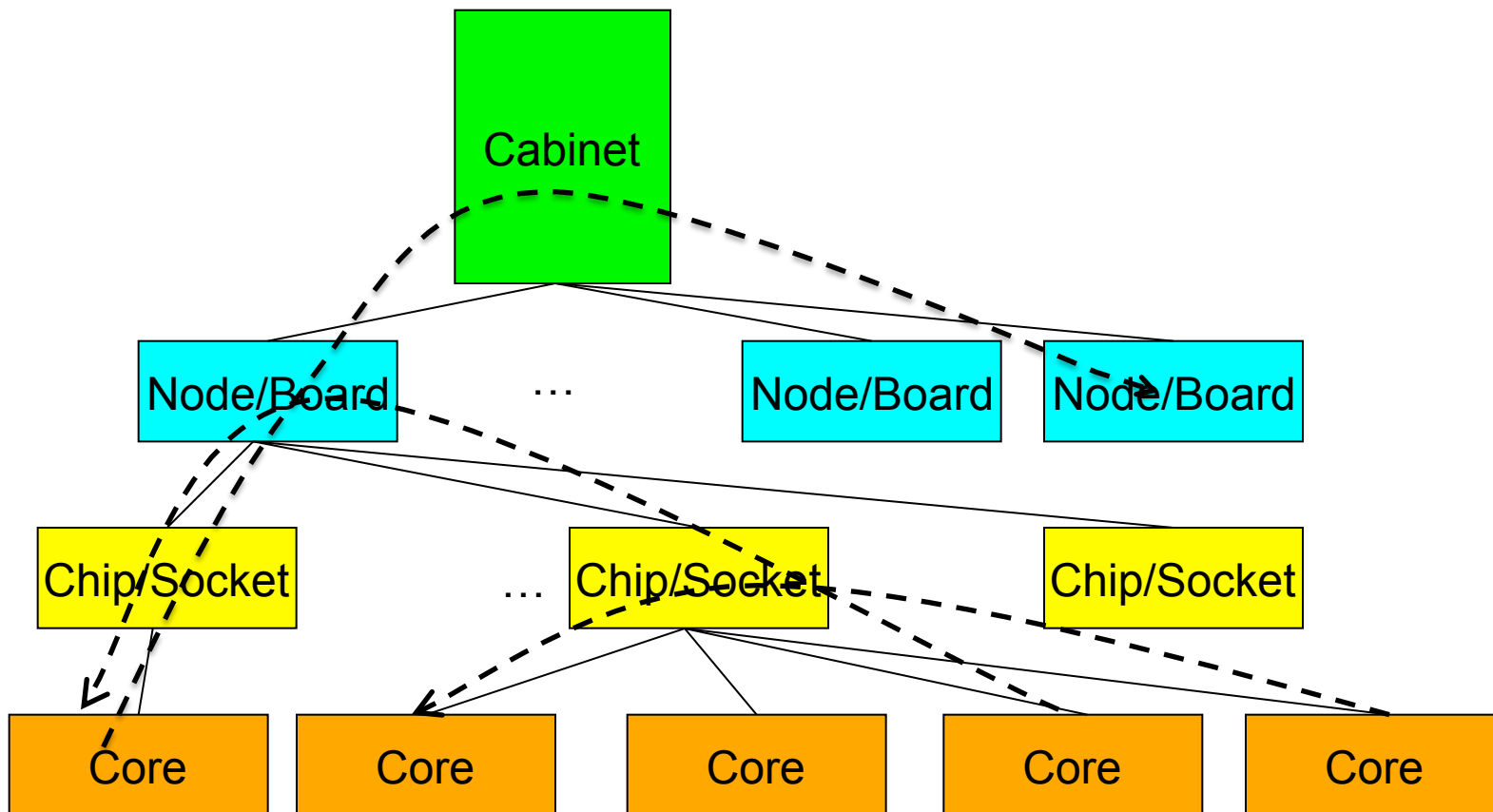


Example of typical parallel machine



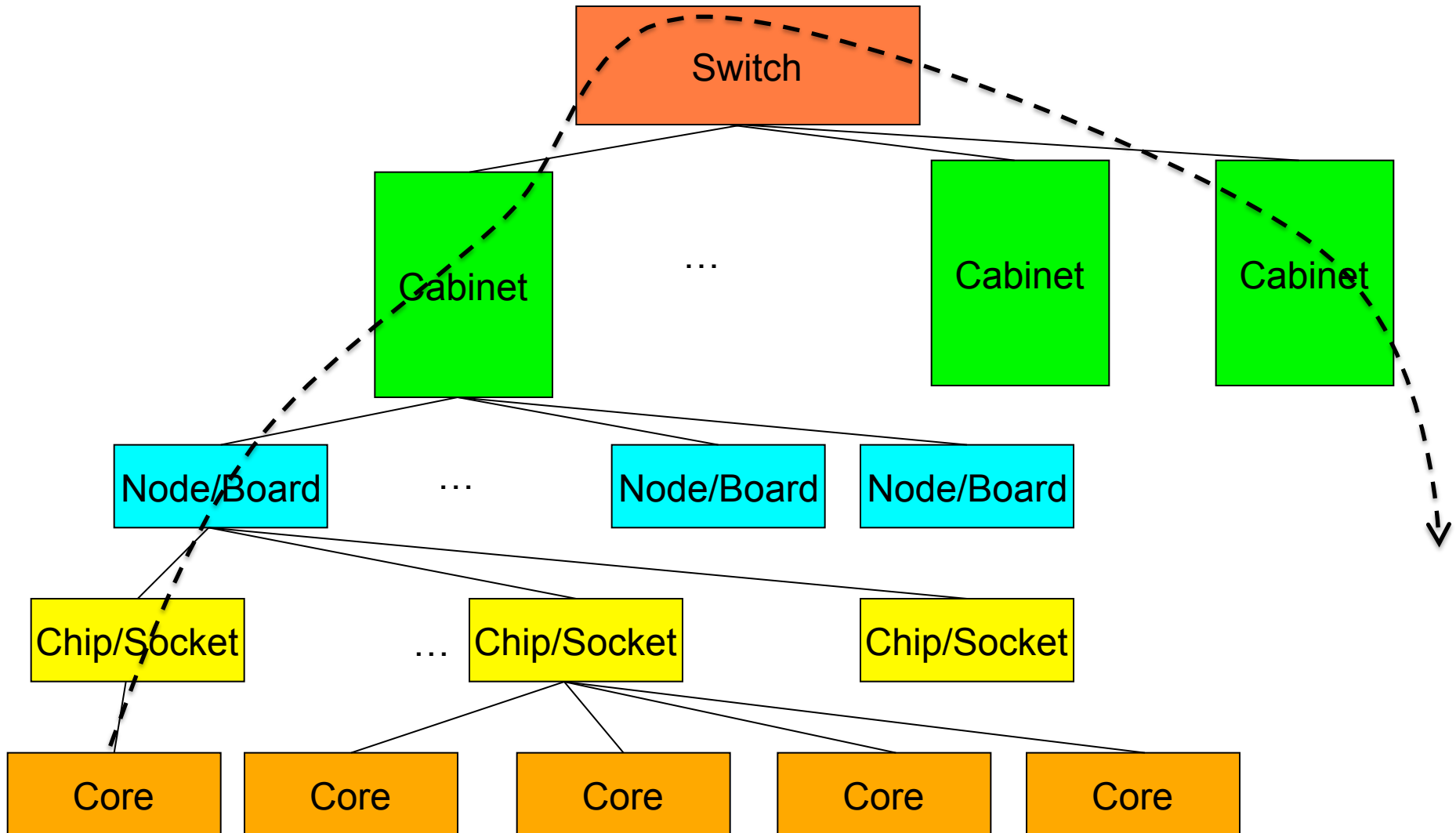
Example of typical parallel machine

Shared memory programming between processes on a board and
a combination of shared memory and distributed memory programming
between nodes and cabinets



Example of typical parallel machine

Combination of shared memory and distributed memory programming





November 2011: The TOP10

Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak
1	RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx + custom	Japan	705,024	10.5	93
2	Nat. SuperComputer Center in Tianjin	Tianhe-1A, NUDT Intel + Nvidia GPU + custom	China	186,368	2.57	55
3	DOE / OS Oak Ridge Nat Lab	Jaguar, Cray AMD + custom	USA	224,162	1.76	75
4	Nat. Supercomputer Center in Shenzhen	Nebulea, Dawning Intel + Nvidia GPU + IB	China	120,640	1.27	43
5	GSIC Center, Tokyo Institute of Technology	Tusbame 2.0, HP Intel + Nvidia GPU + IB	Japan	73,278	1.19	52
6	DOE / NNSA LANL & SNL	Cielo, Cray AMD + custom	USA	142,272	1.11	81
7	NASA Ames Research Center/NAS	Plelades SGI Altix ICE 8200EX/8400EX + IB	USA	111,104	1.09	83
8	DOE / OS Lawrence Berkeley Nat Lab	Hopper, Cray AMD + custom	USA	153,408	1.054	82
9	Commissariat a l'Energie Atomique (CEA)	Tera-10, Bull Intel + IB	France	138,368	1.050	84
10	DOE / NNSA Los Alamos Nat Lab	Roadrunner, IBM AMD + Cell GPU + IB	USA	122,400	1.04	76

Critical Issues for Peta and Exascale Algorithms

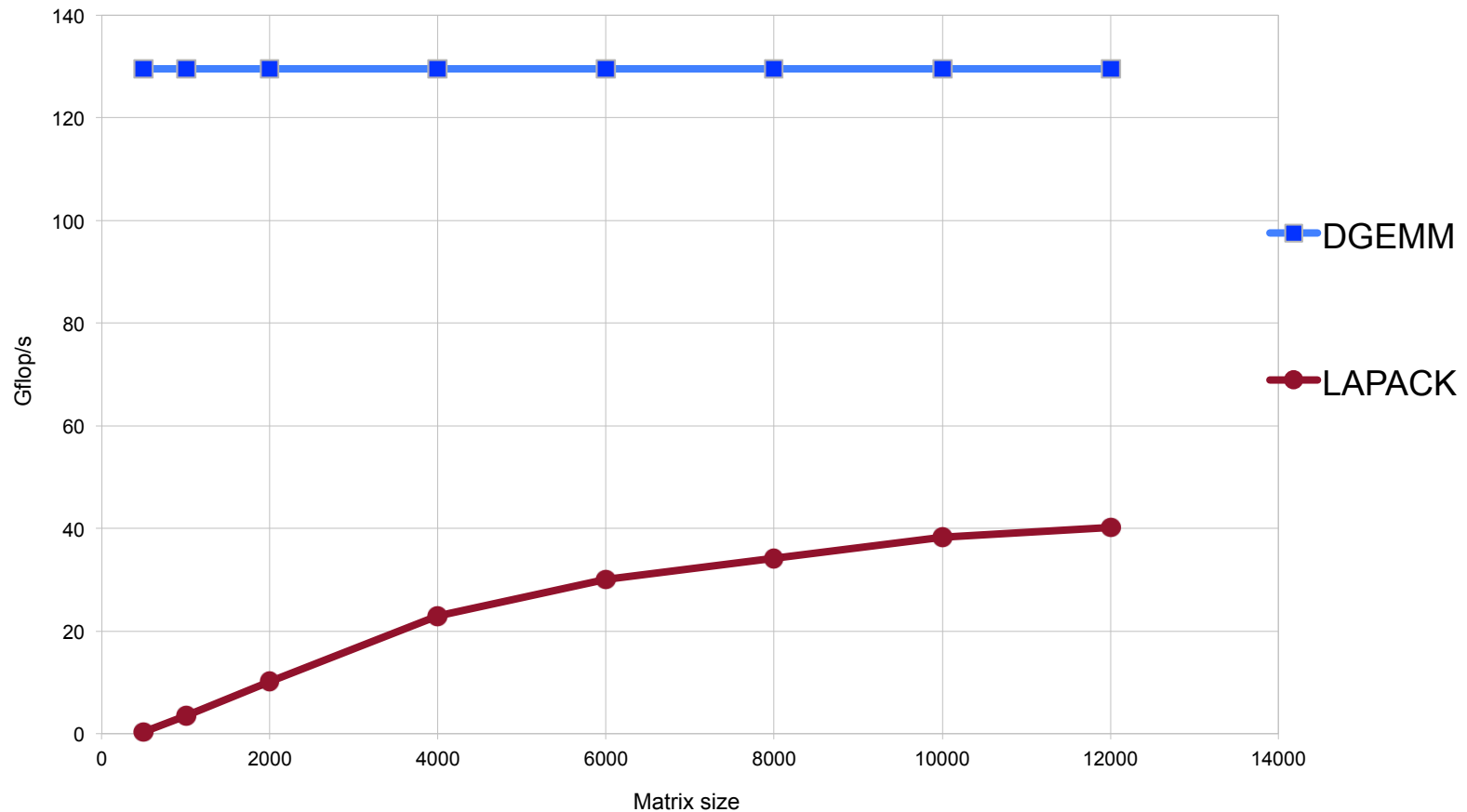
- .. **Synchronization-reducing algorithms**
 - Break Fork-Join model
- .. **Communication-reducing algorithms**
 - Use methods which have lower bound on communication
- .. **Mixed precision methods**
 - 2x speed of ops and 2x speed for data movement
- .. **Autotuning**
 - Today's machines are too complicated, build "smarts" into software have experiment to optimize.
- .. **Fault resilient algorithms**
 - Implement algorithms that can recover from failures/bit flips
- .. **Reproducibility of results**
 - Today we can't guarantee this. We understand the issues, but some of our "colleagues" have a hard time with this.

Major Changes to Software

- **Must rethink the design of our software**
 - **Another disruptive technology**
 - Similar to what happened with cluster computing and message passing
 - **Rethink and rewrite the applications, algorithms, and software**
- **Numerical libraries for example will change**
 - **For example, both LAPACK and ScaLAPACK will undergo major changes to accommodate this**

LAPACK LU - Intel64 - 16 cores

DGETRF - Intel64 Xeon quad-socket quad-core (16 cores) - th. peak 153.6 Gflop/s

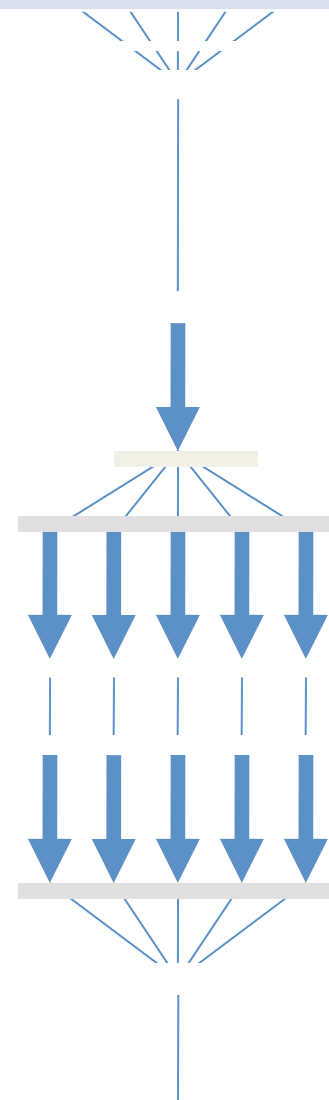
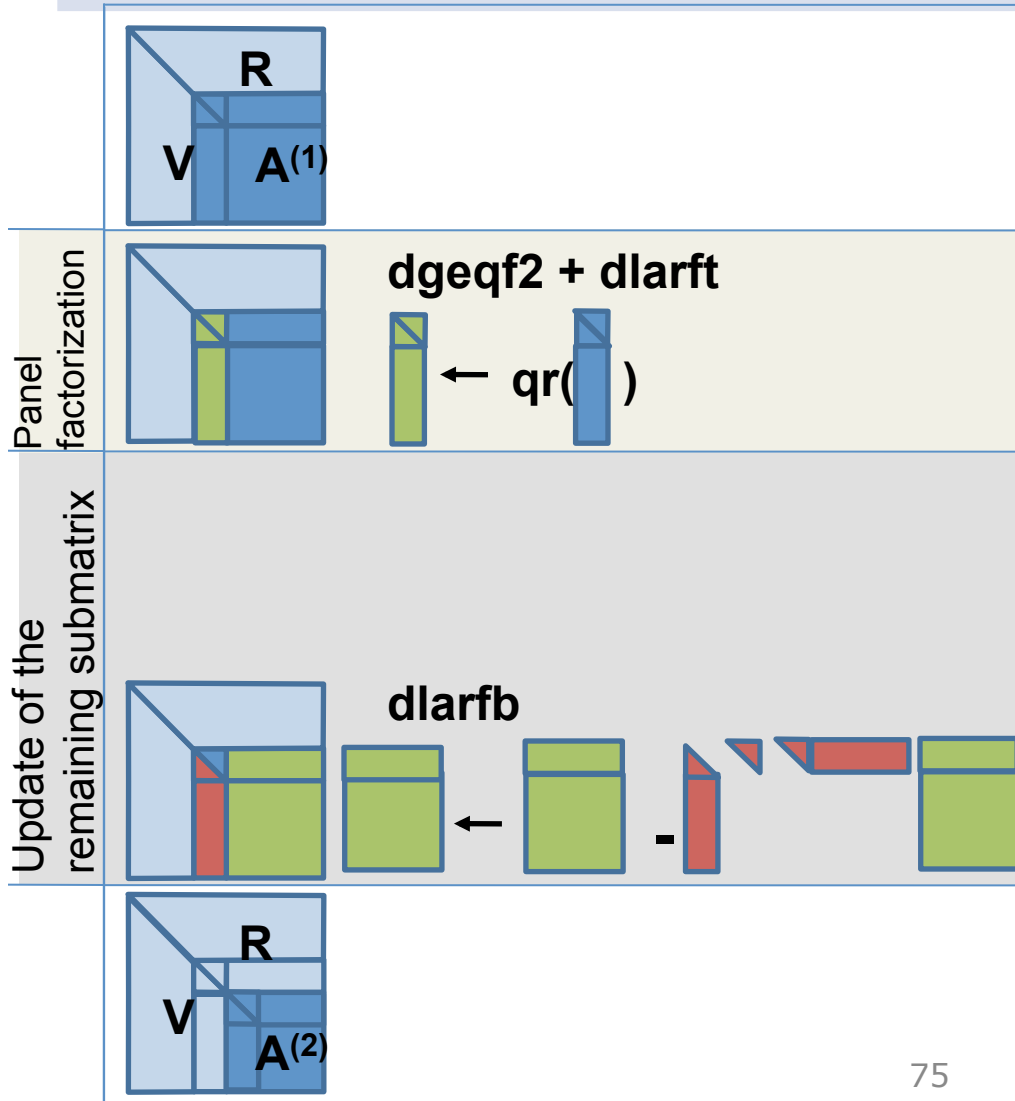


Parallelization of QR Factorization

Parallelize the update:

- Easy and done in any reasonable software.
- This is the $2/3n^3$ term in the FLOPs count.
- Can be done efficiently with LAPACK+multithreaded BLAS

dgemm



Fork - Join parallelism
Bulk Sync Processing

PLASMA: Parallel Linear Algebra s/w for Multicore Architectures

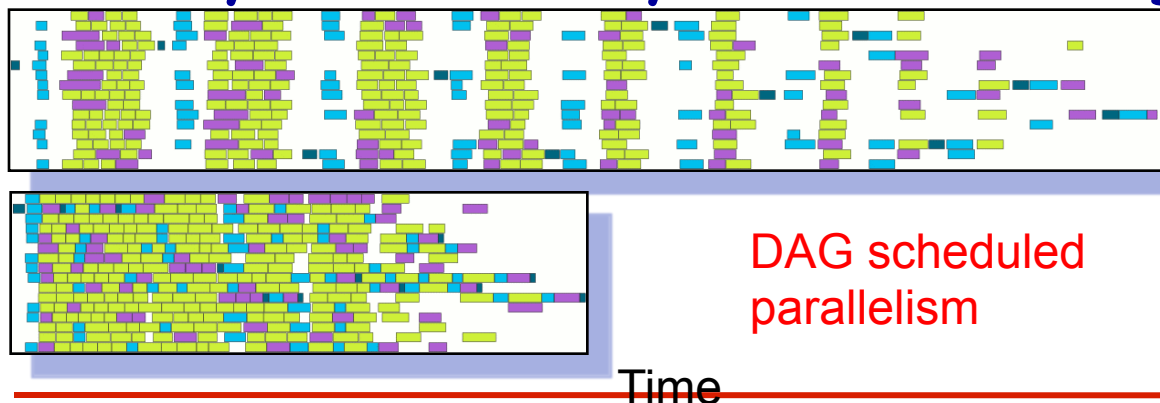
➤ Objectives

- High utilization of each core
- Scaling to large number of cores
- Shared or distributed memory

➤ Methodology

- Dynamic DAG scheduling
- Split phases task generation and execution
- Explicit parallelism/Implicit communication
- Fine granularity / block data layout

➤ Arbitrary DAG with dynamic scheduling



DAG scheduled
parallelism

