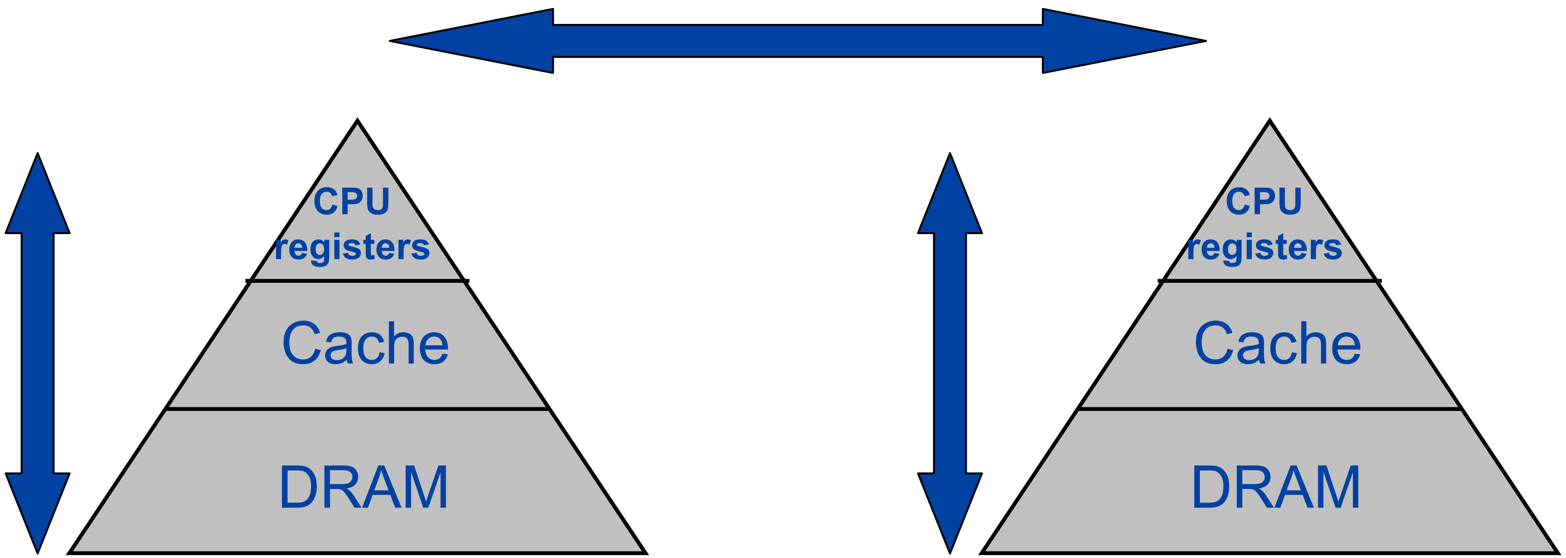
Avoiding communication in linear algebra

Laura Grigori
INRIA Saclay
Paris Sud University

Motivation

Algorithms spend their time

- in doing useful computations (flops)
- or in moving data
  - between different levels of the memory hierarchy
  - and between processors



Motivation

- Time to move data >> time per flop

Running time =
#flops * **time_per_flop** +
#words_moved / **bandwidth** +
#messages * **latency**

Improvements per year

DRAM	Network
23%	**26%**
5%	**15%**

- Gap steadily and exponentially growing over time

"There is an old network saying: Bandwidth problems can be cured with money. Latency problems are harder because the speed of light is fixed -- you can't bribe God." Anonymous

*"We are going to hit the **memory wall**, unless something basic changes"* [W. Wulf, S. McKee, 95]

- And we are also going to hit the "***interconnect network wall***"

Motivation

- The communication problem needs to be taken into account higher in the computing stack
- A paradigm shift in the way the numerical algorithms are devised is required
- Communication avoiding algorithms - a novel perspective for numerical linear algebra
  - Minimize volume of communication
  - Minimize number of messages
  - Minimize over multiple levels of memory/parallelism
  - Allow redundant computations (preferably as a low order term)

Plan

- Motivation
- Selected past work on reducing communication
- Communication complexity of linear algebra operations
- Communication avoiding for dense linear algebra
  - LU, LU_PRRP, QR, Rank Revealing QR factorizations
  - Often not in ScaLAPACK or LAPACK
  - Algorithms for multicore processors
- Communication avoiding for sparse linear algebra
  - Sparse Cholesky factorization
  - Iterative methods and preconditioning
- Conclusions

Selected past work on reducing communication

- Only few examples shown, many references available

A. Tuning

- Overlap communication and computation, at most a factor of 2 speedup

B. Ghosting

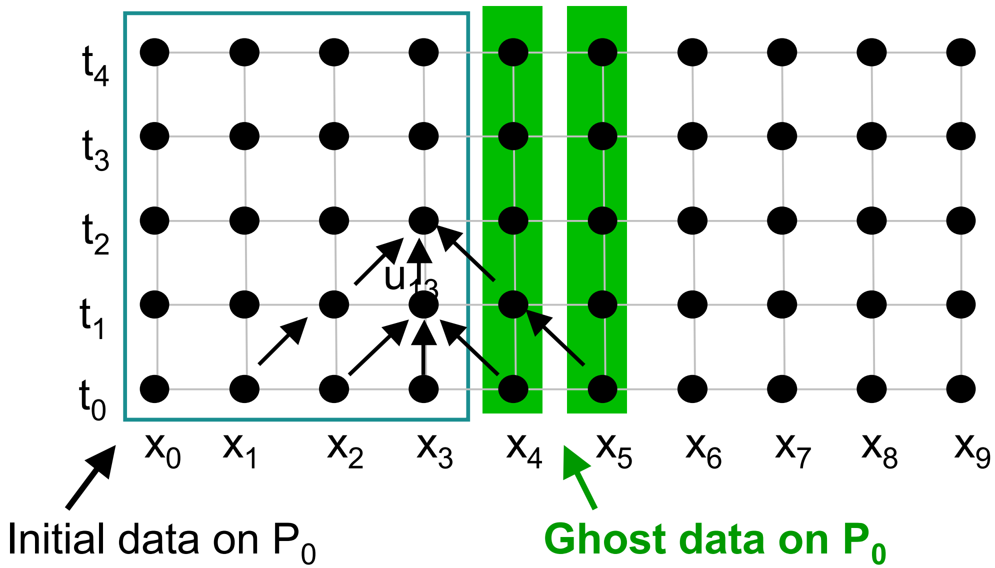
- Standard approach in *explicit methods*
- Store redundantly data from neighboring processors for future computations

Example of a parabolic PDE

$$u_t = \alpha \Delta u$$

with a finite difference,
the solution at a grid point is:

$$u_{i,j+1} = u(x_i, t_{j+1}) = f(u_{i-1,j}, u_{ij}, u_{i+1,j})$$



Selected past work on reducing communication

C. Same operation, different schedule of the computation

Block algorithms for dense linear algebra

- Barron and Swinnerton-Dyer, 1960
  - LU factorization used to solve a system with 31 equations - first subroutine written for EDSAC 2
  - Block LU factorization used to solve a system with 100 equations using an auxiliary magnetic-tape
  - The basis of the algorithm used in LAPACK

Cache oblivious algorithms for dense linear algebra

- recursive Cholesky, LU, QR (Gustavson ‘97, Toledo ‘97, Elmroth and Gustavson ‘98, Frens and Wise ‘03, Gustavson ‘97, Ahmed and Pingali ‘00)

Selected past work on reducing communication

D. Same algebraic framework, different numerical algorithm

More opportunities for reducing communication, may affect stability

Dense LU-like factorization (Barron and Swinnerton-Dyer, 60)

- LU-like factorization based on pairwise pivoting and its block version
  $PA = L_1 L_2 \ldots L_n U$
- With small modifications, minimizes communication between two levels of fast-slow memory
- Stable for small matrices, unstable for nowadays matrices

Communication Complexity of Dense Linear Algebra

- Matrix multiply, using $2n^3$ flops (sequential or parallel)
  - Hong-Kung (1981), Irony/Tishkin/Toledo (2004)
  - Lower bound on Bandwidth = Ω (#flops / $M^{1/2}$)
  - Lower bound on Latency = Ω (#flops / $M^{3/2}$)

- Same lower bounds apply to LU using reduction
  - Demmel, LG, Hoemmen, Langou 2008

$$\begin{pmatrix} I & & -B \\ A & I & \\ & & I \end{pmatrix} = \begin{pmatrix} I & & \\ A & I & \\ & & I \end{pmatrix} \cdot \begin{pmatrix} I & & -B \\ & I & AB \\ & & I \end{pmatrix}$$

- And to almost all direct linear algebra [Ballard, Demmel, Holtz, Schwartz, 09]

Sequential algorithms and communication bounds

Algorithm	Minimizing #words (not #messages)	Minimizing #words and #messages
Cholesky	LAPACK	[Gustavson, 97] [Ahmed, Pingali, 00]
LU	LAPACK (few cases) [Toledo,97], [Gustavson, 97] both use partial pivoting	[LG, Demmel, Xiang, 08] [Khabou, Demmel, LG, Gu, 12] uses tournament pivoting
QR	LAPACK (few cases) [Elmroth,Gustavson,98]	[Frens, Wise, 03], 3x flops [Demmel, LG, Hoemmen, Langou, 08] uses different representation of Q
RRQR	?	[Branescu, Demmel, LG, Gu, Xiang 11] uses tournament pivoting, 3x flops

- Only several references shown for block algorithms (LAPACK), cache-oblivious algorithms and communication avoiding algorithms

2D Parallel algorithms and communication bounds

- If memory per processor = n^2 / P, the lower bounds become
  #words_moved ≥ $\Omega (n^2 / P^{1/2})$, #messages ≥ $\Omega (P^{1/2})$

Algorithm	Minimizing #words (not #messages)	Minimizing #words and #messages
Cholesky	ScaLAPACK	ScaLAPACK
LU	ScaLAPACK uses partial pivoting	[LG, Demmel, Xiang, 08] [Khabou, Demmel, LG, Gu, 12] uses tournament pivoting
QR	ScaLAPACK	[Demmel, LG, Hoemmen, Langou, 08] uses different representation of Q
RRQR	?	[Branescu, Demmel, LG, Gu, Xiang 11] uses tournament pivoting, 3x flops

- Only several references shown, block algorithms (ScaLAPACK) and communication avoiding algorithms

Scalability of communication optimal algorithms

- 2D communication optimal algorithms, $M = 3 \cdot n^2/P$

  (matrix distributed over a $P^{1/2}$-by- $P^{1/2}$ grid of processors)

  $T_P = O(n^3)\gamma + \Omega(n^2 / P^{1/2})\beta + \Omega(P^{1/2})\alpha$

  - Isoefficiency: $\mathbf{n^3 \propto P^{1.5}}$ and $\mathbf{n^2 \propto P}$
  - For GEPP, $\mathbf{n^3 \propto P^{2.25}}$ [Grama et al, 93]

- 3D communication optimal algorithms, $M = 3 \cdot P^{1/3}(n^2/P)$

  (matrix distributed over a $P^{1/3}$-by- $P^{1/3}$-by- $P^{1/3}$ grid of processors)

  $T_P = O(n^3)\gamma + \Omega(n^2 / P^{2/3})\beta + \Omega(\log(P))\alpha$

  - Isoefficiency: $\mathbf{n^3 \propto P}$ and $\mathbf{n^2 \propto P^{2/3}}$

- 2.5D algorithms with $M = 3 \cdot c \cdot (n^2/P)$, and 3D algorithms exist for matrix multiplication and LU factorization
  - References: Dekel et al 81, Agarwal et al 90, 95, Johnsson 93, McColl and Tiskin 99, Irony and Toledo 02, Solomonik and Demmel 2011

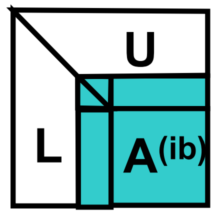
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a $P = P_r \times P_c$ grid of processors

For ib = 1 to n-1 step b

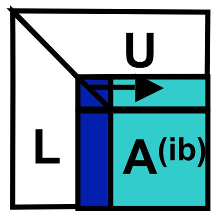
$A^{(ib)}$ = A(ib:n, ib:n)

#messages



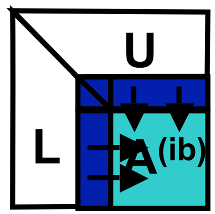
(1) Compute panel factorization $O(n \log_2 P_r)$

- find pivot in each column, swap rows



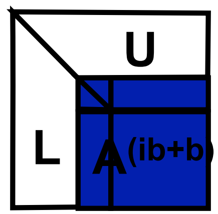
(2) Apply all row permutations $O(n/b(\log_2 P_c + \log_2 P_r))$

- broadcast pivot information along the rows
- swap rows at left and right



(3) Compute block row of U $O(n/b \log_2 P_c)$

- broadcast right diagonal block of L of current panel



(4) Update trailing matrix $O(n/b(\log_2 P_c + \log_2 P_r))$

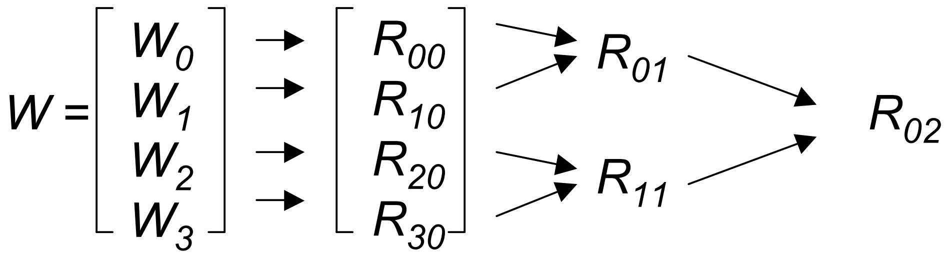
- broadcast right block column of L
- broadcast down block row of U

TSQR: QR factorization of a tall skinny matrix using Householder transformations

- QR decomposition of m x b matrix W, m >> b
  - P processors, block row layout

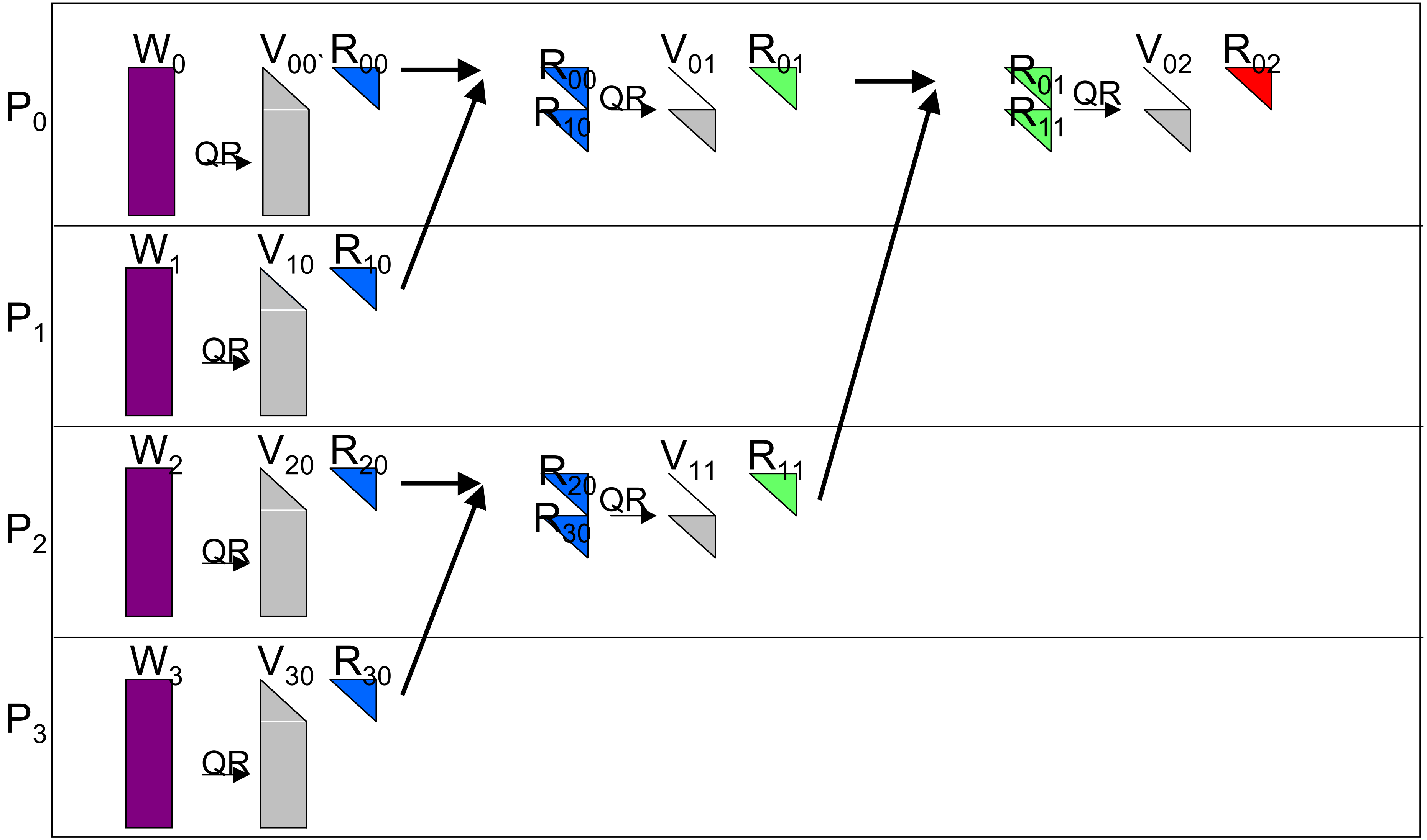
- Classic Parallel Algorithm
  - Compute Householder vector for each column
  - Number of messages ∝ b log P

- Communication Avoiding Algorithm
  - Reduction operation, with QR as operator
  - Number of messages ∝ log P



J. Demmel, LG, M. Hoemmen, J. Langou, 08

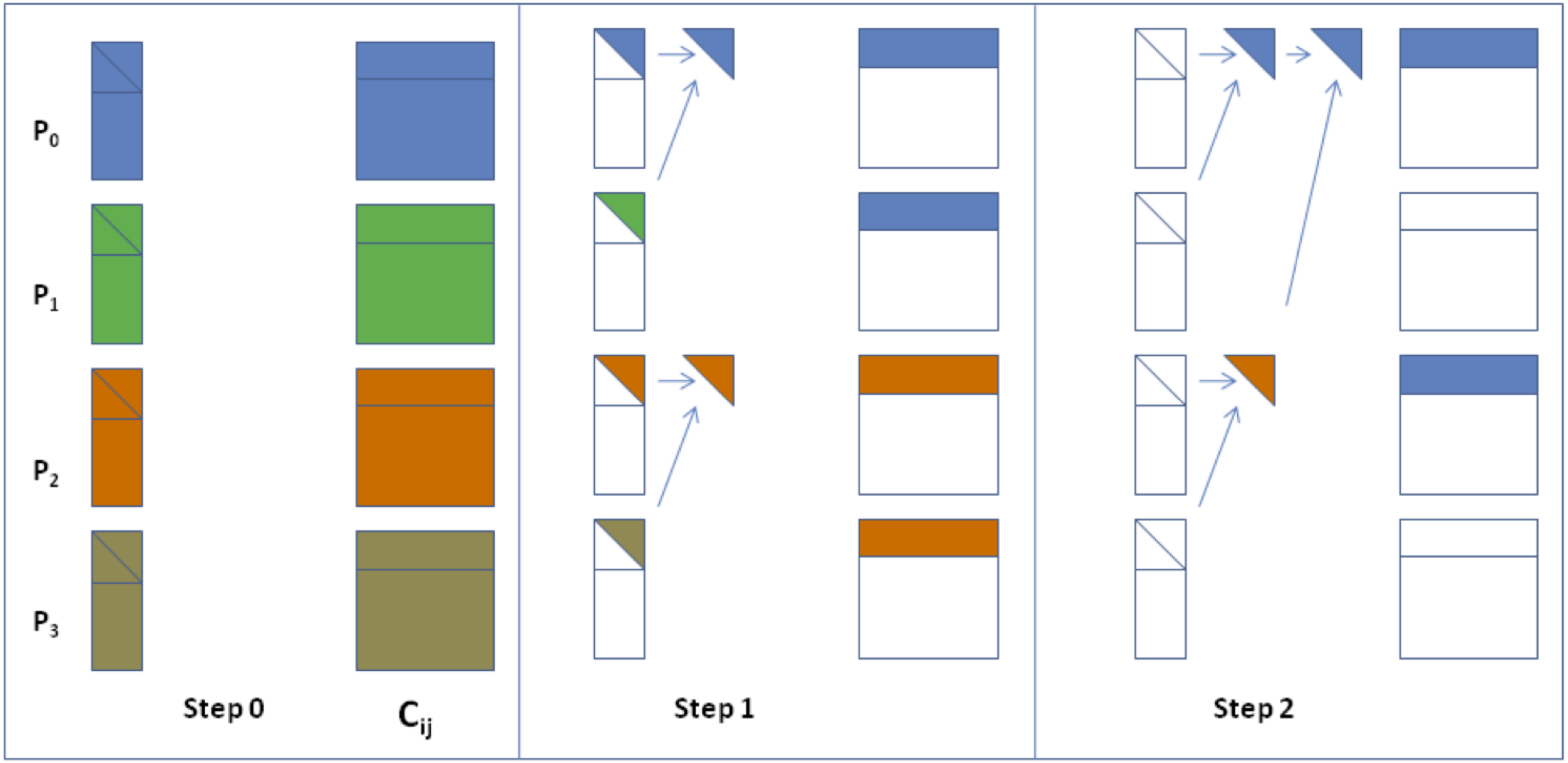
Parallel TSQR



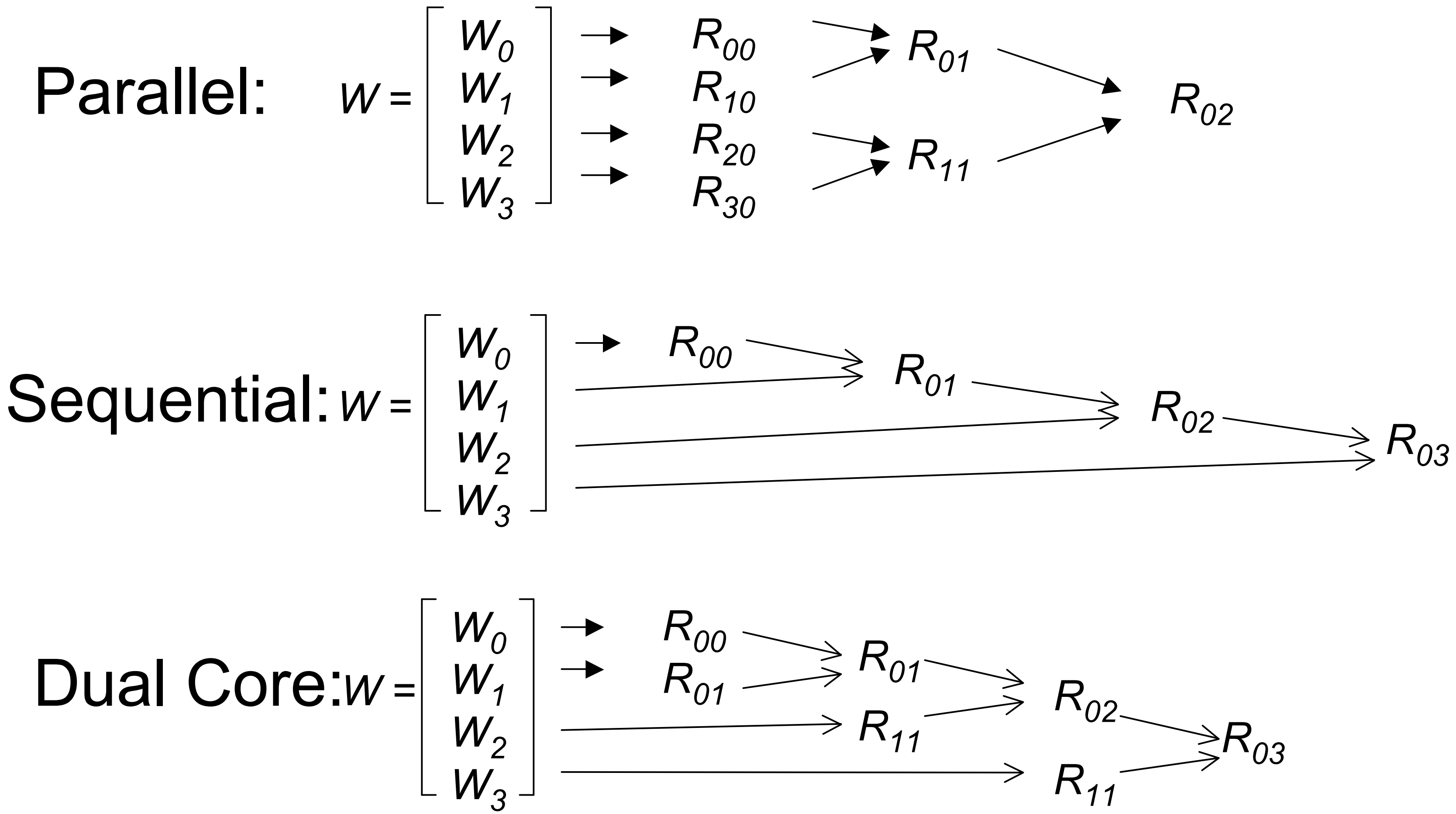
References: Golub, Plemmons, Sameh 88, Pothen, Raghavan, 89, Da Cunha, Becker, Patterson, 02

CAQR for general matrices

- Use TSQR for panel factorizations
- Update the trailing matrix - triggered by the reduction tree used for the panel factorization



Flexibility of TSQR and CAQR algorithms



Reduction tree will depend on the underlying architecture, could be chosen dynamically

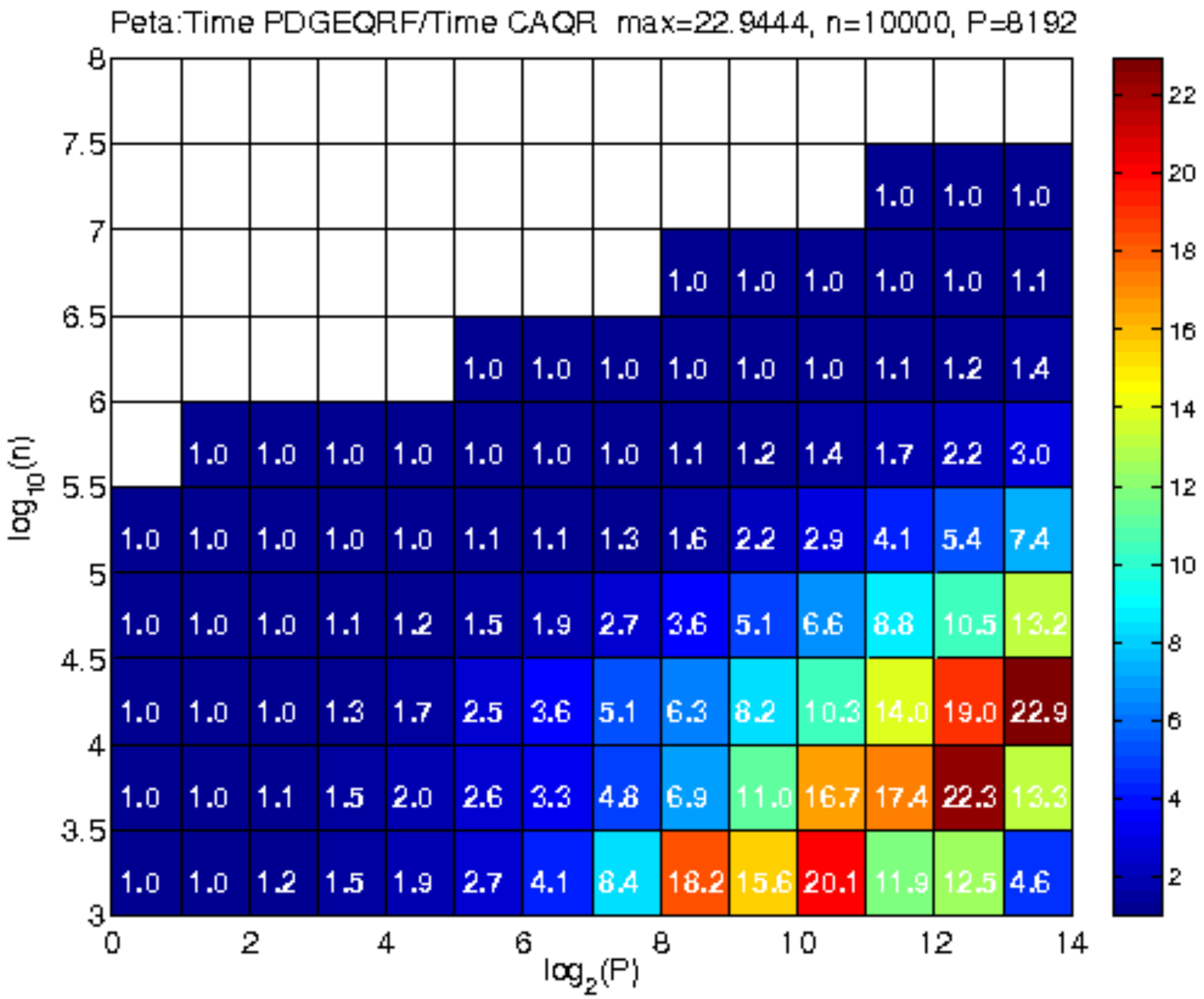
Source slide: J. Demmel

Performance of TSQR vs Sca/LAPACK

- Parallel
  - Intel Xeon (two socket, quad core machine), 2010
    - Up to **5.3x speedup** (8 cores, 10^5 x 200)
  - Pentium III cluster, Dolphin Interconnect, MPICH, 2008
    - Up to **6.7x speedup** (16 procs, 100K x 200)
  - BlueGene/L, 2008
    - Up to **4x speedup** (32 procs, 1M x 50)
  - QR computed locally using recursive algorithm (Elmroth-Gustavson) – enabled by TSQR

- See [Demmel, LG, Hoemmen, Langou, SISC 12], [Donfack, LG, IPDPS 10].

Modeled Speedups of CAQR vs ScaLAPACK



Petascale
up to 22.9x

IBM Power 5
up to 9.7x

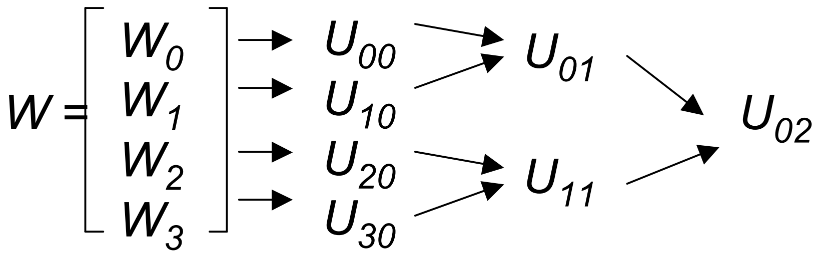
“Grid”
up to 11x

Petascale machine with 8192 procs, each at 500 GFlops/s, a bandwidth of 4 GB/s.

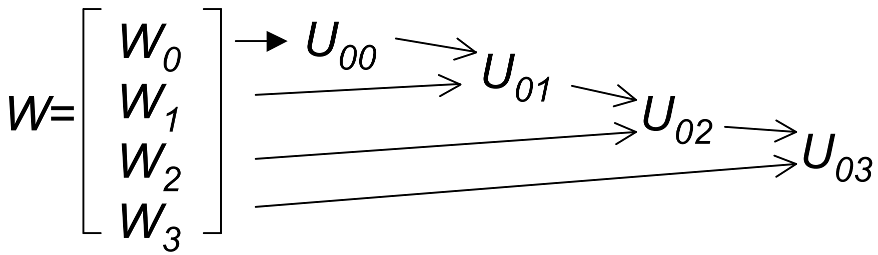
$$\gamma = 2 \cdot 10^{-12} s, \alpha = 10^{-5} s, \beta = 2 \cdot 10^{-9} s / word.$$

Obvious generalization of TSQR to LU

- Block parallel pivoting:
  - uses a binary tree and is optimal in the parallel case



- Block pairwise pivoting:
  - uses a flat tree and is optimal in the sequential case
  - used in PLASMA for multicore architectures and FLAME for out-of-core algorithms and for multicore architectures



Stability of the LU factorization

- The backward stability of the LU factorization of a matrix A of size n-by-n

$$\left\| |L| \cdot |U| \right\|_\infty \leq (1 + 2(n^2 - n) g_w) \|A\|_\infty$$

depends on the growth factor

$$g_W = \frac{\max_{i,j,k} |a_{ij}^k|}{\max_{i,j} |a_{ij}|}$$ where a_{ij}^k are the values at the k-th step.

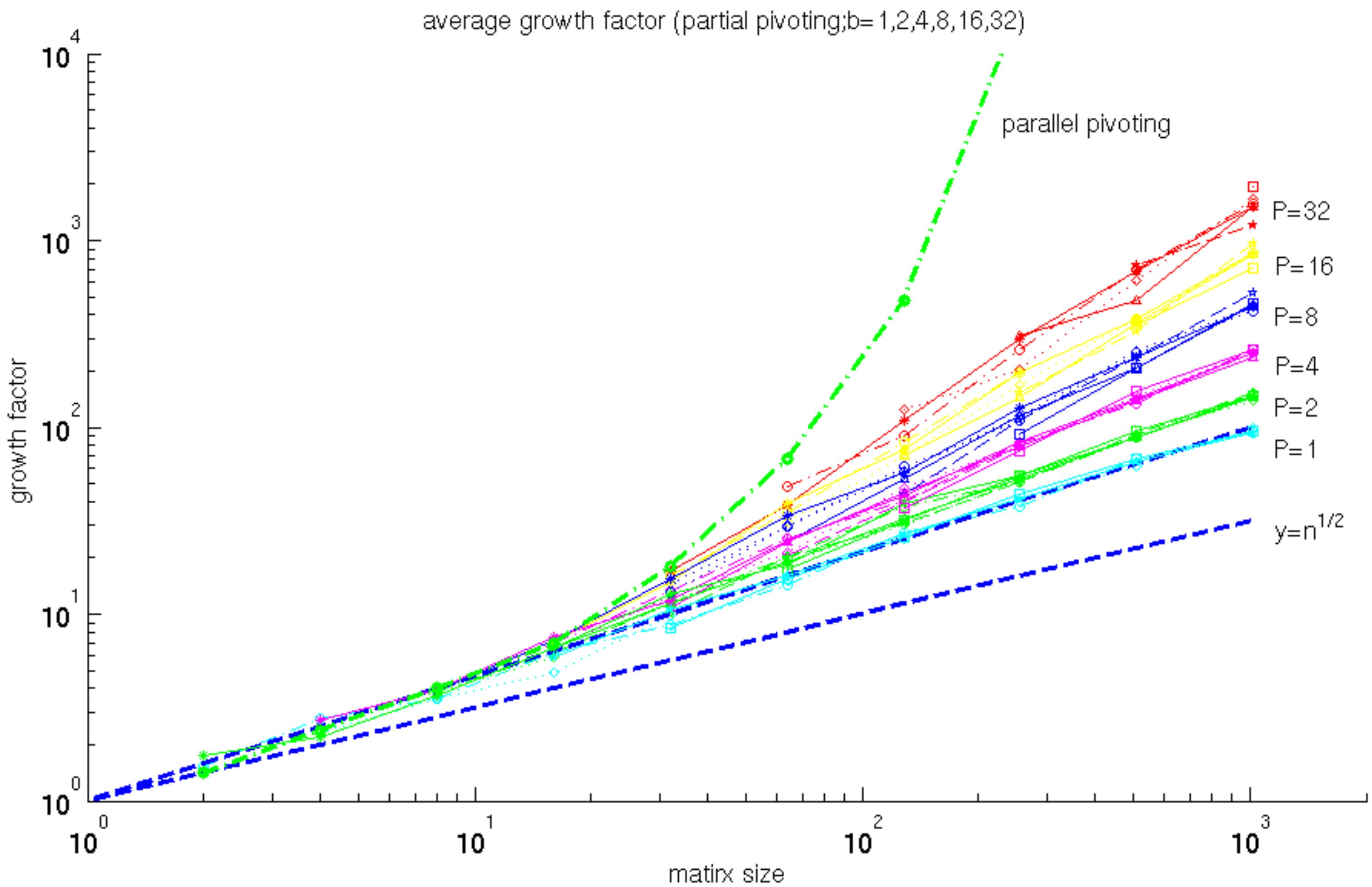
- $g_W \leq 2^{n-1}$, but in practice it is on the order of $n^{2/3}$ -- $n^{1/2}$

- Two reasons considered to be important for the average case stability [Trefethen and Schreiber, 90] :

- the multipliers in L are small,

- the correction introduced at each elimination step is of rank 1.

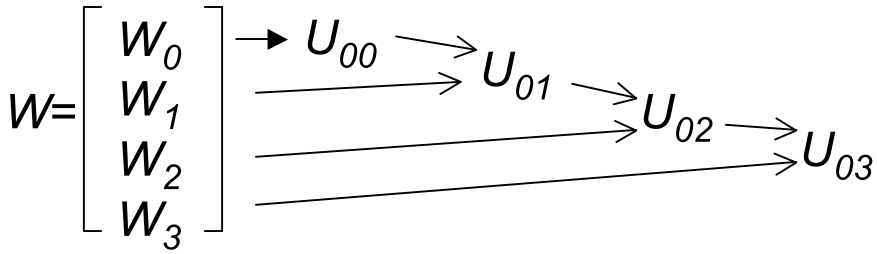
Block parallel pivoting

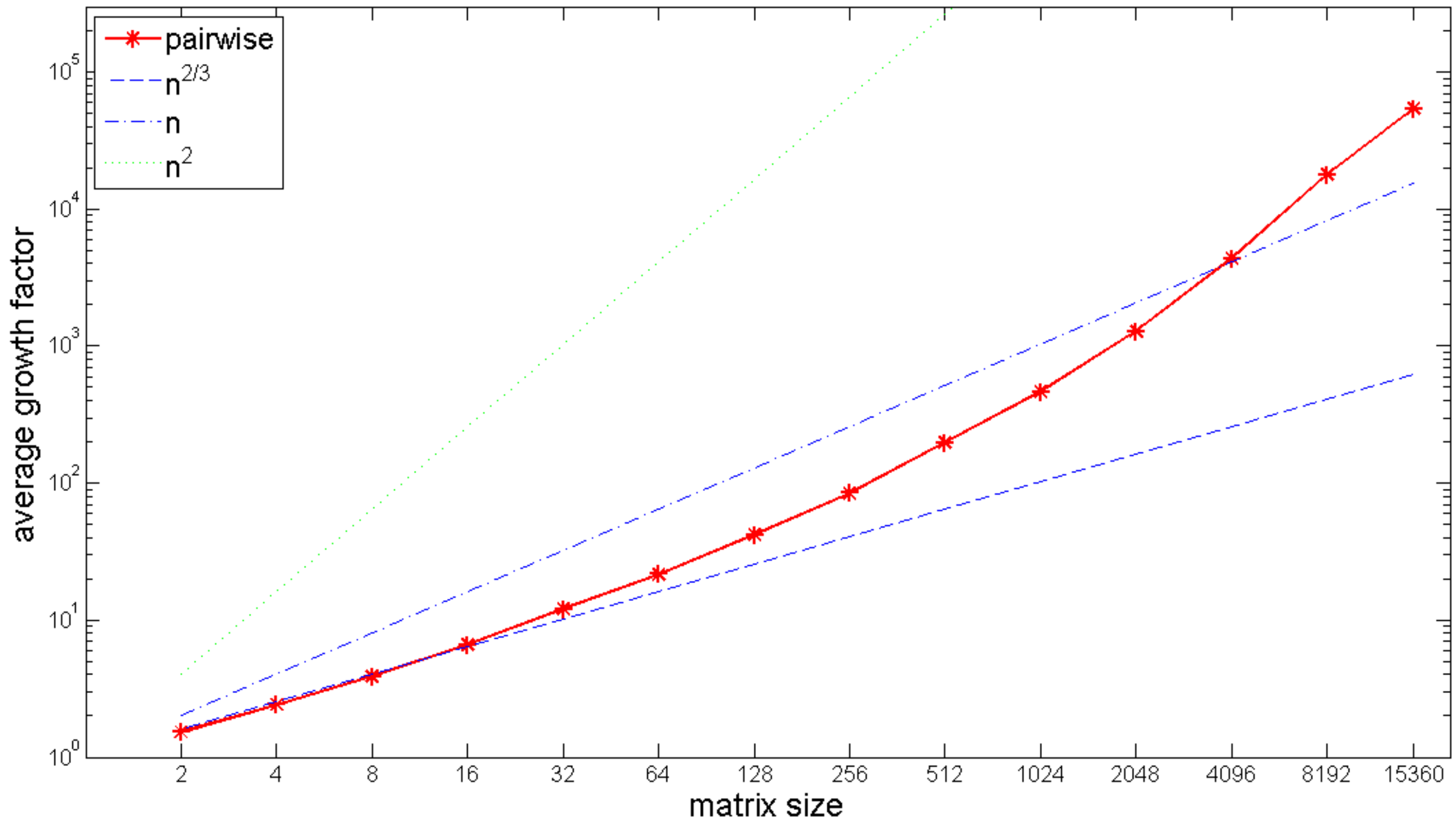


- Unstable for large number of processors P
- When P=number rows, it corresponds to parallel pivoting, known to be unstable (Trefethen and Schreiber, 90)

Block pairwise pivoting

- Results shown for random matrices
- Will become unstable for large matrices





Tournament pivoting - the overall idea

- At each iteration of a block algorithm

$$A = \begin{pmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{21}}^{n-b} \\ A_{21} & A_{22} \end{pmatrix} \begin{matrix} \}\, b \\ \}\, n-b \end{matrix} \text{, where } \quad W = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$$

  - Preprocess W to find at low communication cost good pivots for the LU factorization of W, return a permutation matrix P.
  - Permute the pivots to top, ie compute PA.
  - Compute LU with no pivoting of W, update trailing matrix.

$$PA = \begin{pmatrix} L_{11} & \\ L_{21} & I_{n-b} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ & A_{22} - L_{21}U_{12} \end{pmatrix}$$

Tournament pivoting

P_0

$$W_0 = \begin{pmatrix} 2 & 4 \\ 0 & 1 \\ 2 & 0 \\ 1 & 2 \end{pmatrix} = \Pi_0 L_0 U_0 \qquad \Pi_0^T W_0 = \begin{pmatrix} 2 & 4 \\ 2 & 0 \end{pmatrix}$$

P_1

$$W_1 = \begin{pmatrix} 2 & 0 \\ 0 & 0 \\ 4 & 1 \\ 1 & 0 \end{pmatrix} = \Pi_1 L_1 U_1 \qquad \Pi_1^T W_1 = \begin{pmatrix} 4 & 1 \\ 2 & 0 \end{pmatrix}$$

P_2

$$W_2 = \begin{pmatrix} 0 & 1 \\ 1 & 4 \\ 0 & 0 \\ 0 & 2 \end{pmatrix} = \Pi_2 L_2 U_2 \qquad \Pi_2^T W_2 = \begin{pmatrix} 1 & 4 \\ 0 & 2 \end{pmatrix}$$

P_3

$$W_3 = \begin{pmatrix} 2 & 1 \\ 0 & 2 \\ 1 & 0 \\ 4 & 2 \end{pmatrix} = \Pi_3 L_3 U_3 \qquad \Pi_3^T W_3 = \begin{pmatrix} 4 & 2 \\ 0 & 2 \end{pmatrix}$$

P_0 (from P_0, P_1):

$$\overline{W}_0 = \begin{pmatrix} 2 & 4 \\ 2 & 0 \\ 4 & 1 \\ 2 & 0 \end{pmatrix} = \overline{\Pi}_0 \overline{L}_0 \overline{U}_0 \qquad \overline{\Pi}_0^T \overline{W}_0 = \begin{pmatrix} 4 & 1 \\ 2 & 4 \end{pmatrix}$$

P_2 (from P_2, P_3):

$$\overline{W}_2 = \begin{pmatrix} 1 & 4 \\ 0 & 2 \\ 4 & 2 \\ 0 & 2 \end{pmatrix} = \overline{\Pi}_2 \overline{L}_2 \overline{U}_2 \qquad \overline{\Pi}_2^T \overline{W}_2 = \begin{pmatrix} 4 & 2 \\ 1 & 4 \end{pmatrix}$$

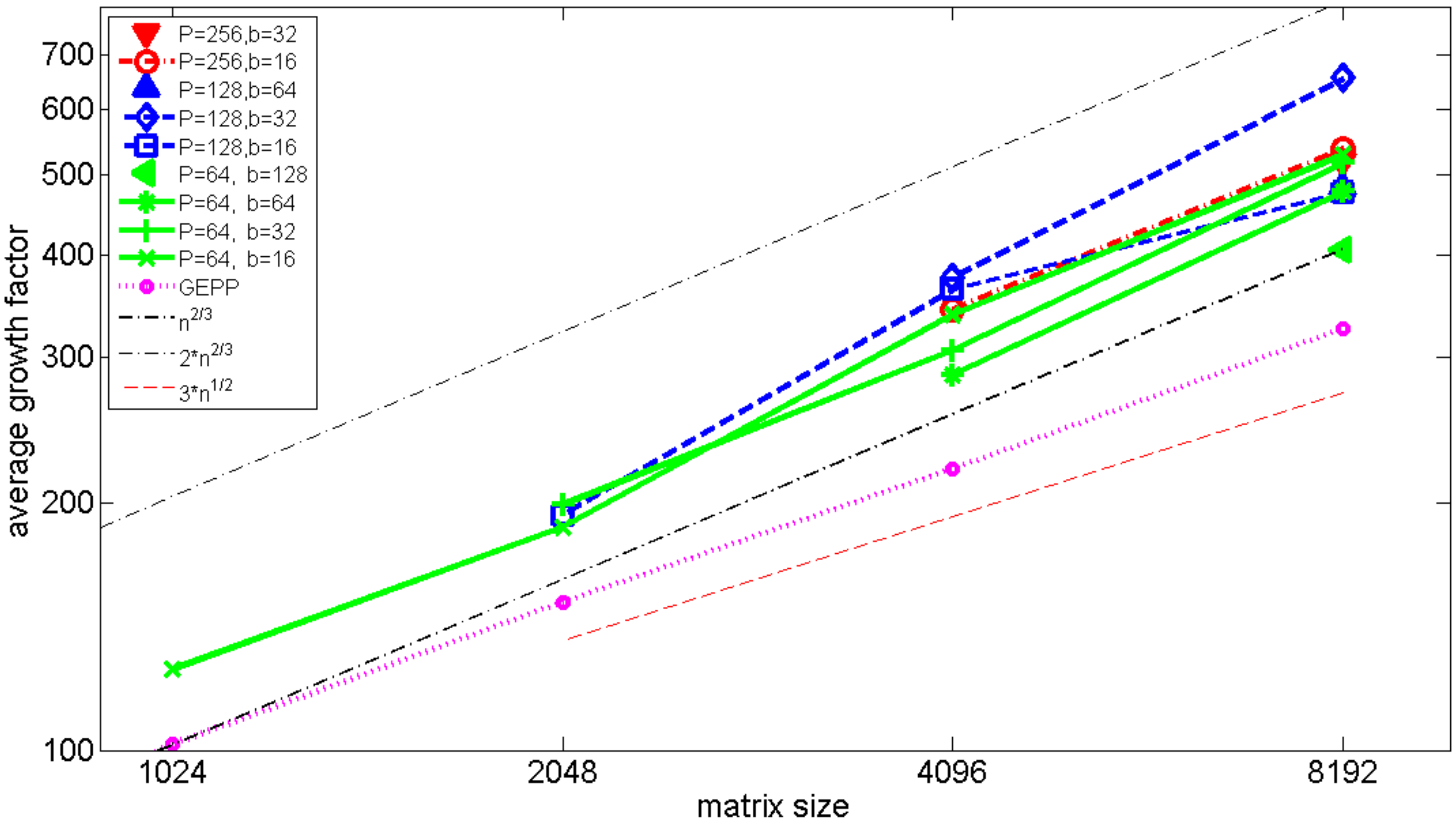
P_0 (from P_0, P_2):

$$\underline{W}_0 = \begin{pmatrix} 4 & 1 \\ 2 & 4 \\ 4 & 2 \\ 1 & 4 \end{pmatrix} = \underline{\Pi}_0 \underline{L}_0 \underline{U}_0 \qquad \underline{\Pi}_0^T \underline{W}_0 = \begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix}$$

Good pivots for factorizing W

time →

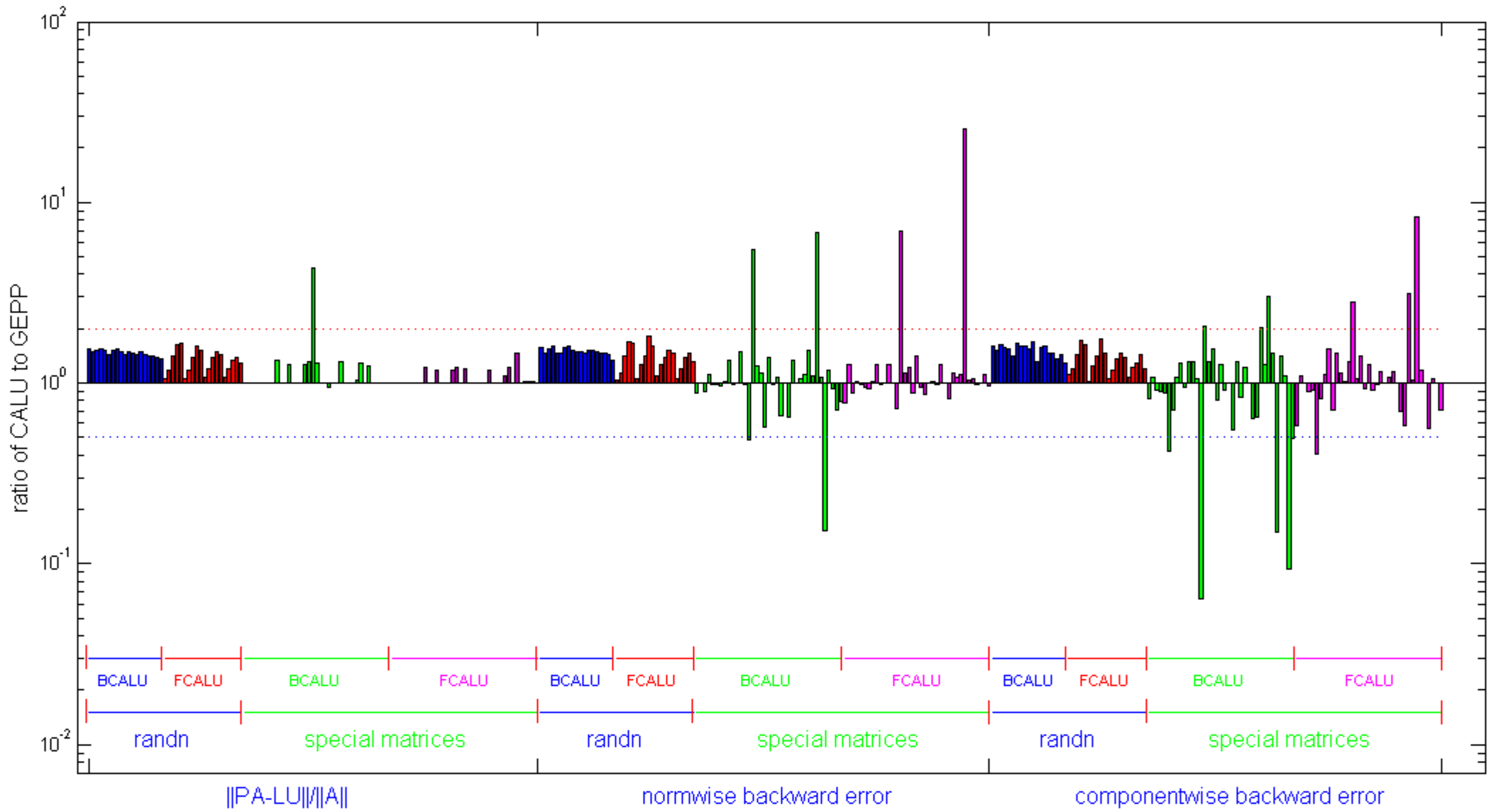
Growth factor for binary tree based CALU



- Random matrices from a normal distribution
- Same behaviour for all matrices in our test, and |L| <= 4.2

Stability of CALU (experimental results)

- Results show ||PA-LU||/||A||, normwise and componentwise backward errors, for random matrices and special ones
  - See [LG, Demmel, Xiang, 2010] for details
  - BCALU denotes binary tree based CALU and FCALU denotes flat tree based CALU



Our “proof of stability” for CALU

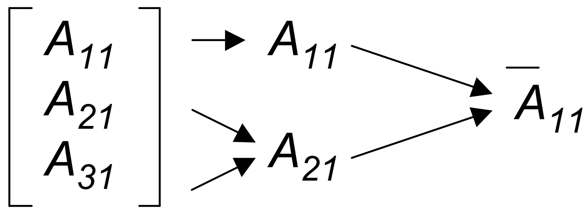
- CALU as stable as GEPP in following sense:

  CALU process on a matrix A is equivalent to GEPP process on a larger matrix G whose entries are blocks of A and blocks of zeros.

- Example of one step of tournament pivoting:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{pmatrix}$$

tournament pivoting:



$$G = \begin{pmatrix} \bar{A}_{11} & & \bar{A}_{12} \\ A_{21} & A_{21} & \\ & -A_{31} & A_{32} \end{pmatrix}$$

- Proof possible by using original rows of A during tournament pivoting (not the computed rows of U).

Growth factor of different pivoting strategies

- Matrix of size m-by-n, reduction tree of height H=log(P).
- (CA)LU_PRRP select pivots using strong rank revealing QR (A. Khabou, J. Demmel, LG, M. Gu, 2012)
- "In practice" means observed/expected/conjectured values.

	CALU	GEPP	CALU_PRRP	LU_PRRP
Upper bound	$2^{n(\log(P)+1)-1}$	2^{n-1}	$(1+2b)^{(n/b)\log(P)}$	$(1+2b)^{(n/b)}$
In practice	$n^{2/3}$ -- $n^{1/2}$	$n^{2/3}$ -- $n^{1/2}$	$(n/b)^{2/3}$ -- $(n/b)^{1/2}$	$(n/b)^{2/3}$ -- $(n/b)^{1/2}$

→ Better bounds

- For a matrix of size 10^7-by-10^7 (using petabytes of memory)

  $n^{1/2} = 10^{3.5}$

- When will Linpack have to use the QR factorization for solving linear systems ?

Performance vs ScaLAPACK

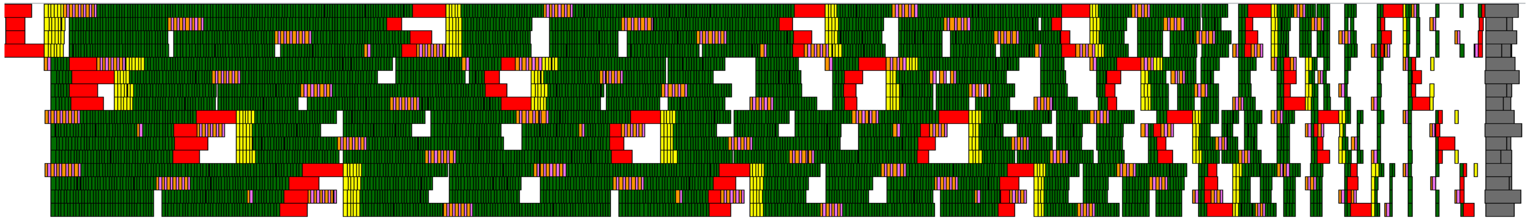
- Parallel TSLU (LU on tall-skinny matrix)
  - IBM Power 5
    - Up to **4.37x** faster (16 procs, 1M x 150)
  - Cray XT4
    - Up to **5.52x** faster (8 procs, 1M x 150)

- Parallel CALU (LU on general matrices)
  - Intel Xeon (two socket, quad core)
    - Up to **2.3x** faster (8 cores, 10^6 x 500)
  - IBM Power 5
    - Up to **2.29x** faster (64 procs, 1000 x 1000)
  - Cray XT4
    - Up to **1.81x** faster (64 procs, 1000 x 1000)

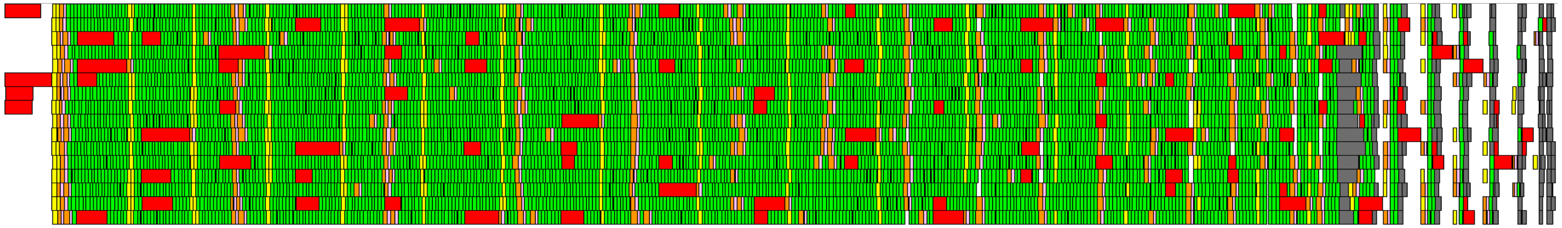
- Details in SC08 (LG, Demmel, Xiang), IPDPS'10 (S. Donfack, LG).

Scheduling CALU's Task Dependency Graph

- Static scheduling
  - \+ Good locality of data
  - \- Ignores noise



- Dynamic scheduling
  - \+ Keeps cores busy
  - \- Poor usage of data locality
  - \- Can have large dequeue overhead

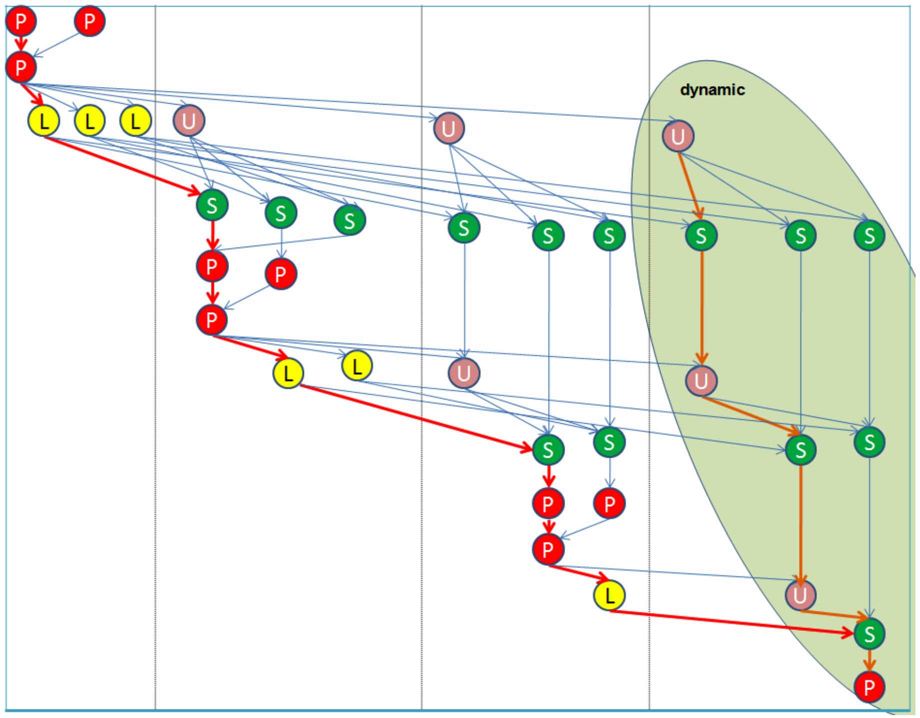


Lightweight scheduling

- A self-adaptive strategy to provide
  - A good trade-off between load balance, data locality, and dequeue overhead.
  - Performance consistency
  - Shown to be efficient for regular mesh computation [B. Gropp and V. Kale]

Combined static/dynamic scheduling:

- A thread executes in priority its statically assigned tasks
- When no task ready, it picks a ready task from the dynamic part
- The size of the dynamic part is guided by a performance model

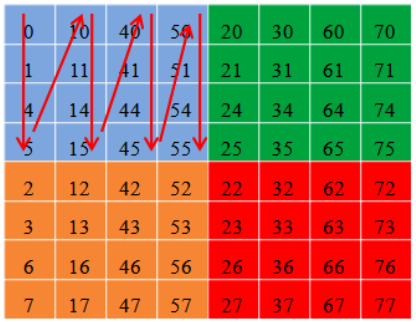


S. Donfack, LG, B. Gropp, V. Kale, 2012

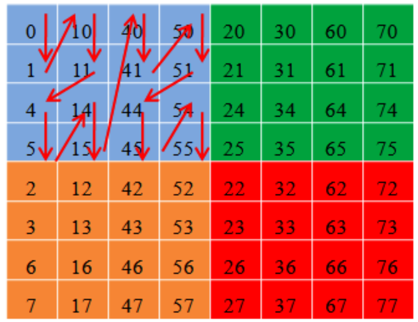
Impact of data layout on performance

Data layouts:

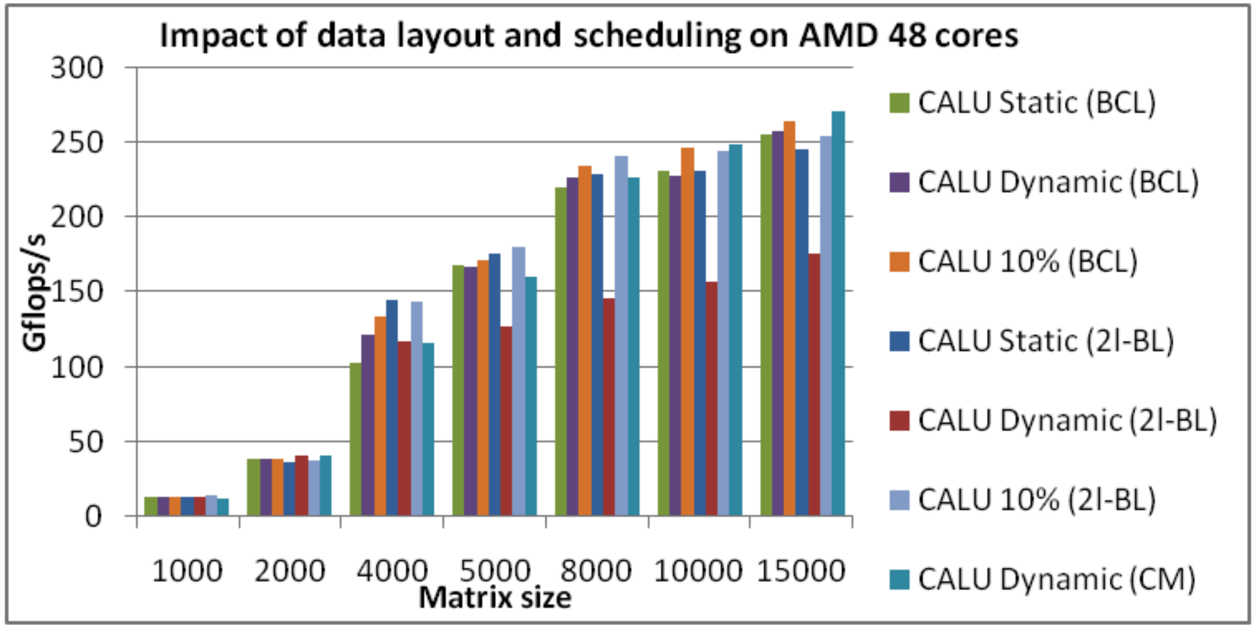
- CM : Column major order
- BCL : Each thread stores its data using CM
- 2l-BL : Each thread stores its data in blocks



Block cyclic layout (BCL)

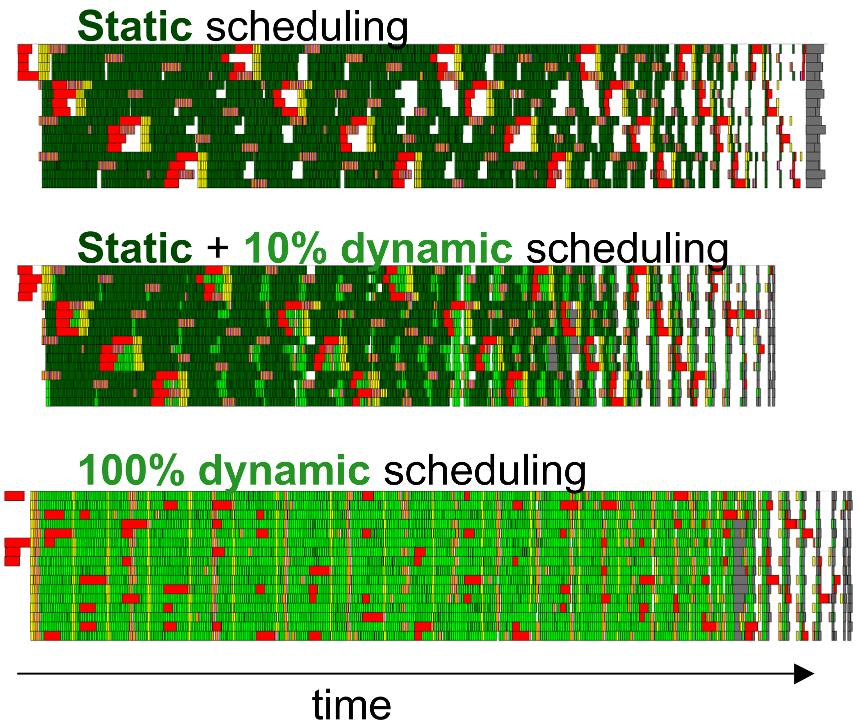


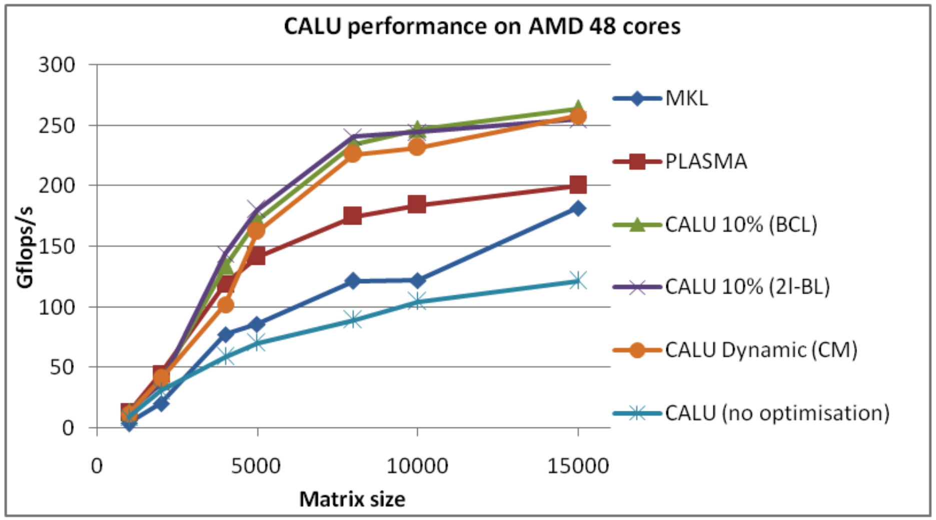
Two level block layout (2l-BL)



Four socket, twelve cores machine based on AMD Opteron processor (U. of Tennessee).

Best performance of CALU on multicore architectures





- Reported performance for PLASMA uses LU with block pairwise pivoting.

Plan

- Motivation
- Selected past work on reducing communication
- Communication complexity of linear algebra operations
- Communication avoiding for dense linear algebra
  - LU, LU_PRRP, QR, Rank Revealing QR factorizations
  - Often not in ScaLAPACK or LAPACK
  - Algorithms for multicore processors
- Communication avoiding for sparse linear algebra
  - Sparse Cholesky factorization
  - Iterative methods and preconditioning
- Conclusions

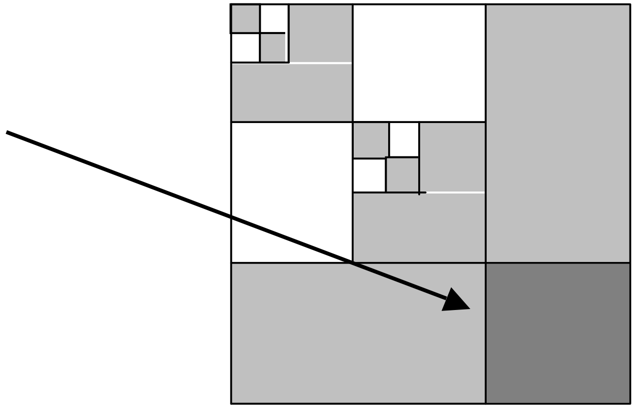
Sparse Cholesky factorization for 2D/3D regular grids

- Matrix A from a finite difference operator on a regular grid of dimension $s \geq 2$ with k^s nodes.

- Its Cholesky L factor contains a dense lower triangular matrix of size $k^{s-1} \times k^{s-1}$.

  $\#$ words_moved $\geq \Omega((k^{3(s-1)}/(2P)) / M^{1/2})$

  $\#$ messages $\geq \Omega((k^{3(s-1)}/(2P)) / M^{3/2})$



- PSPASES with an optimal layout minimizes communication
  - Uses nested dissection to reorder the matrix
  - Distributes the matrix using the subtree-to-subcube algorithm
- Sequential multifrontal algorithm minimizes communication
  - Every dense multifrontal matrix is factored using an optimal dense Cholesky

- But in general for sparse matrix operations, the known lower bounds on communication can become vacuous

Communication in Krylov subspace methods

Iterative methods to solve Ax =b

- Find a solution x_k from $x_0 + K_k(A, r_0)$, where $K_k(A, r_0) = span\{r_0, A r_0, \ldots, A^{k-1} r_0\}$ such that the Petrov-Galerkin condition $b - Ax_k \perp L_k$ is satisfied.

- For numerical stability, an orthonormal basis $\{q_1, q_2, \ldots, q_k\}$ for $K_k(A, r_0)$ is computed (CG, GMRES, BiCGstab,…)

- Each iteration requires
  - Sparse matrix vector product
  - Dot products for the orthogonalization process

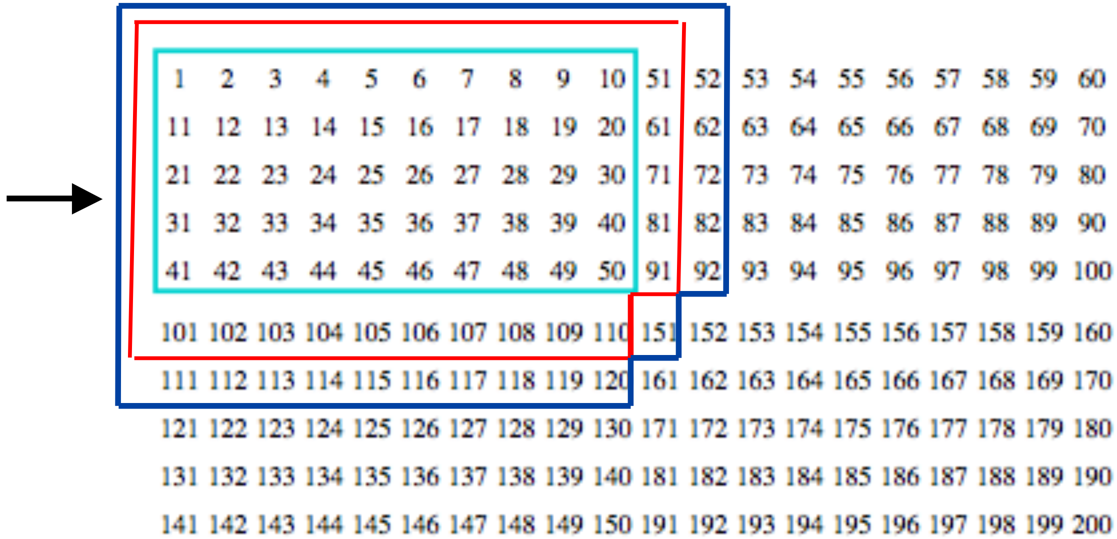
- *S-step Krylov subspace methods*
  - Unroll s iterations, orthogonalize every s steps

- Van Rosendale '83, Walker '85, Chronopoulous and Gear '89, Erhel '93, Toledo '95, Bai, Hu, Reichel '91 (Newton basis), Joubert and Carey '92 (Chebyshev basis), etc.
- Recent references: G. Atenekeng, B. Philippe, E. Kamgnia (to enable multiplicative Schwarz preconditioner), J. Demmel, M. Hoemmen, M. Mohiyuddin, K. Yellick (to minimize communication, next slide)

S-step Krylov subspace methods

- To avoid communication, unroll s steps, ghost necessary data,
  - generate a set of vectors W for the Krylov subspace $K_k\,(A, r_0)$
  - orthogonalize the vectors using TSQR(W)

Domain and ghost data
to compute A^2 x
with no communication



Example: 5 point stencil 2D grid
partitioned on 4 processors

- A factor of O(s) less data movement in the memory hierarchy
- A factor of O(s) less messages in parallel

Research opportunities and limitations

Length of the basis "s" is limited by
- Size of ghost data
- Loss of precision

Cost for a 3D regular grid, 7 pt stencil

s-steps	Memory	Flops
GMRES	$O(s\ n/P)$	$O(s\ n/P)$
CA-GMRES	$O(s\ n/P)+$ $O(s\ (n/P)^{2/3})+$ $O(s^2\ (n/P)^{1/3})$	$O(s\ n/P)+$ $O(s^2\ (n/P)^{2/3})+$ $O(s^3\ (n/P)^{1/3})$

Preconditioners: few identified so far to work with s-step methods
- Highly decoupled preconditioners: Block Jacobi
- Hierarchical, semiseparable matrices (M. Hoemmen, J. Demmel)

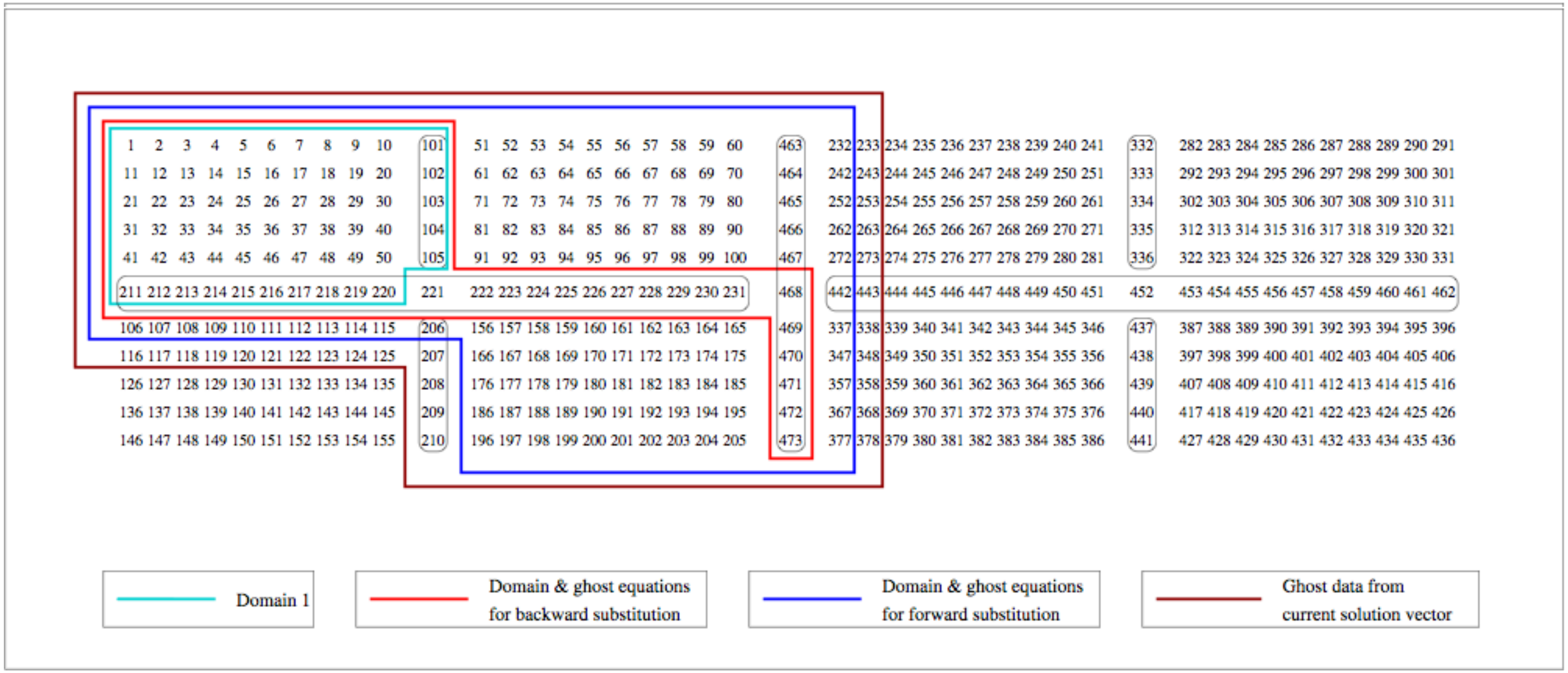
A look at three classes of preconditioners
- Incomplete LU factorizations (joint work with S. Moufawad)
- Two level preconditioners in DDM
- Deflation techniques through preconditioning

ILU0 with nested dissection and ghosting

Let α_0 be the set of equations to be solved by one processor
For *j = 1* to *s* do
  Find β_j = *ReachableVertices (G(U),* α_{j-1}*)*
  Find γ_j = *ReachableVertices (G(L),* β_j*)*
  Find δ_j = *Adj (G(A),* γ_j*)*
  Set $\alpha_j = \delta_j$
end

Ghost data required:
x(δ*), A(*γ,δ*),*
L(γ,γ*), U(*β, β*)*

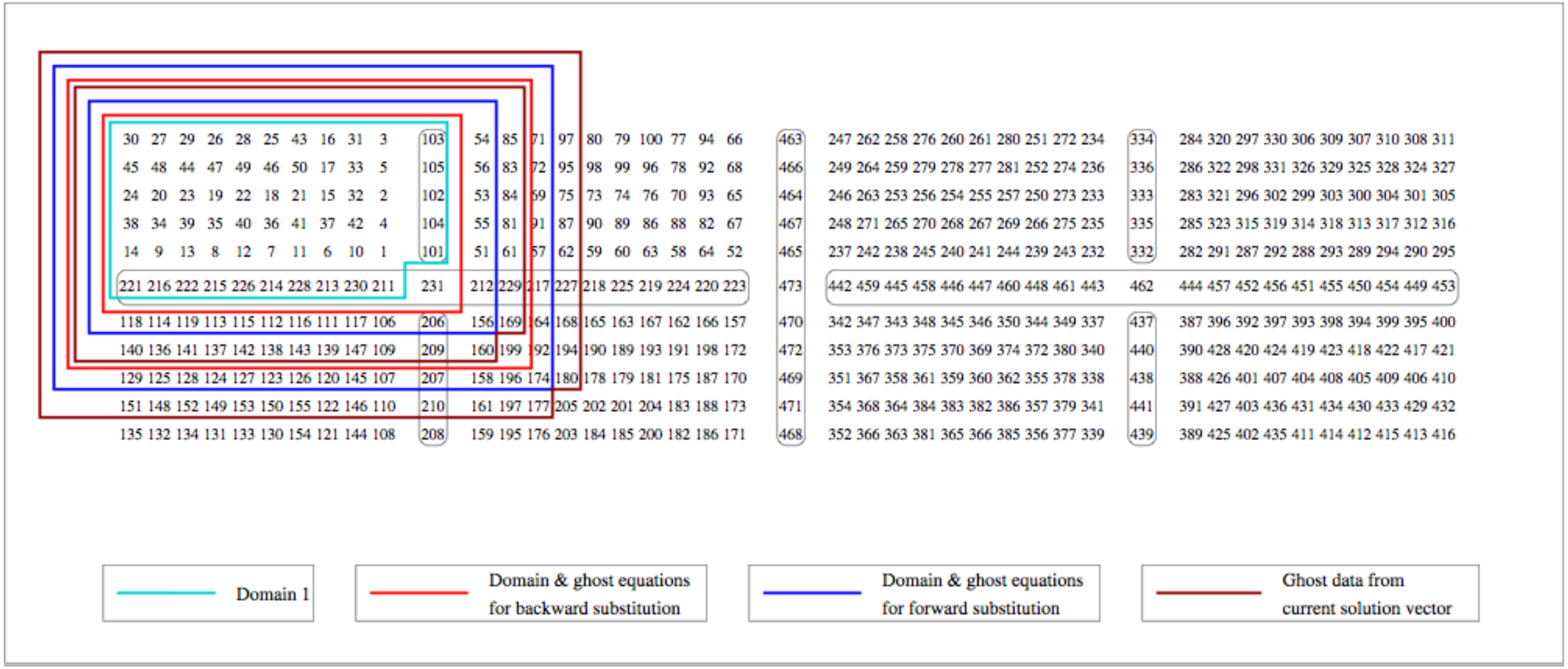
$\Rightarrow$ Half of the work performed on one processor



5 point stencil on a 2D grid

CA-ILU0 with alternating reordering and ghosting

- Reduce volume of ghost data by reordering the vertices:
  - First number the vertices at odd distance from the separators
  - Then number the vertices at even distance from the separators
- CA-ILU0 computes a standard ILU0 factorization



5 point stencil on a 2D grid

Two level preconditioners

In the unified framework of (Tang et al. 09), let :

$$P := I - A\,Q, \qquad Q := Z\,E^{-1}\,Z^T, \qquad E := Z^T A\,Z$$

where

M is the first level preconditioner (eg based on additive Schwarz)

Z is the deflation subspace matrix of full rank

E is the coarse grid correction, a small dense invertible matrix

P is the deflation matrix

Examples of preconditioners:

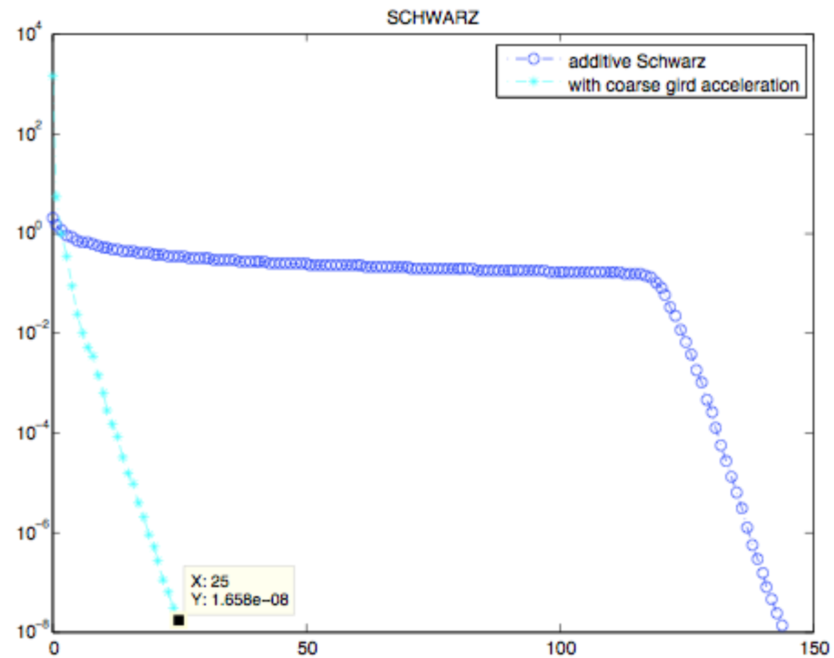
$$P_{ADD} = M^{-1} + Z\,E^{-1}\,Z^T, \qquad P_{ADEF2} = P^T M^{-1} + Z\,E^{-1}\,Z^T$$ (Mandel 1993)

- DDM - Z and Z^T are the restriction and prolongation operators based on subdomains, E is a coarse grid, P is a subspace correction
- Deflation - Z contains the vectors to be deflated
- Multigrid - interpretation possible

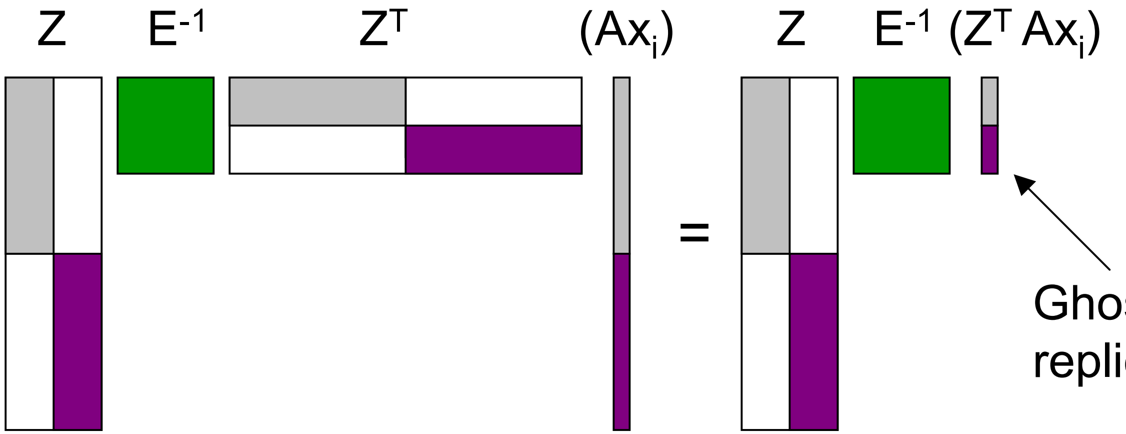
Two level preconditioners

P_{ADD} for a Poisson-like problem, using Z defined as in (Nicolaides 1987):

$$Z = \begin{bmatrix} 1_{\Omega_1} & 0 & \cdots & 0 \\ 0 & 1_{\Omega_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1_{\Omega_P} \end{bmatrix}$$



Courtesy of F. Nataf



Ghosting requires replicating A on each processor

Conclusions

- Introduced a new class of communication avoiding algorithms that minimize communication
  - Attain theoretical lower bounds on communication
  - Minimize communication at the cost of redundant computation
  - Are often faster than conventional algorithms in practice

- Remains a lot to do for sparse linear algebra
  - Communication bounds, communication optimal algorithms
  - Numerical stability of s-step methods
  - Preconditioners - limited by the memory size, not flops
- And BEYOND
  - Our homework for the next years !

Conclusions

- Many previous results
  - Only several cited, many references given in the papers
  - Flat trees algorithms for QR factorization, called tiled algorithms used in the context of
    - Out of core - Gunter, van de Geijn 2005
    - Multicore, Cell processors - Buttari, Langou, Kurzak and Dongarra (2007, 2008), Quintana-Orti, Quintana-Orti, Chan, van Zee, van de Geijn (2007, 2008)

- Upcoming related talks at this conference:
  - MS50: Innovative algorithms for eigenvalue and singular value decomposition, Friday
  - MS59: Communication in Numerical Linear Algebra, Friday PM
  - CP15: A Class of Fast Solvers for Dense Linear Systems on Hybrid GPU-multicore Machines, M. Baboulin, Friday PM
  - CP15: Communication-Avoiding QR: LAPACK Kernels Description, Implementation, Performance and Example of Application, R. James, Friday PM

Collaborators, funding

Collaborators:

- A. Branescu, INRIA, S. Donfack, INRIA, A. Khabou, INRIA, M. Jacquelin, INRIA, S. Moufawad, INRIA, H. Xiang, University Paris 6
- J. Demmel, UC Berkeley, B. Gropp, UIUC, M. Gu, UC Berkeley, M. Hoemmen, UC Berkeley, J. Langou, CU Denver, V. Kale, UIUC

Funding: ANR Petal and Petalh projects, ANR Midas, Digiteo Xscale NL, COALA INRIA funding

Further information:

http://www-rocq.inria.fr/who/Laura.Grigori/

References

Results presented from:

- J. Demmel, L. Grigori, M. F. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, UCB-EECS-2008-89, 2008, published in SIAM journal on Scientific Computing, Vol. 34, No 1, 2012.
- L. Grigori, J. Demmel, and H. Xiang, *Communication avoiding Gaussian elimination*, Proceedings of the IEEE/ACM SuperComputing SC08 Conference, November 2008.
- L. Grigori, J. Demmel, and H. Xiang, *CALU: a communication optimal LU factorization algorithm*, SIAM. J. Matrix Anal. & Appl., 32, pp. 1317-1350, 2011.
- M. Hoemmen's Phd thesis, *Communication avoiding Krylov subspace methods*, 2010.
- L. Grigori, P.-Y. David, J. Demmel, and S. Peyronnet, *Brief announcement: Lower bounds on communication for sparse Cholesky factorization of a model problem*, ACM SPAA 2010.
- S. Donfack, L. Grigori, and A. Kumar Gupta, *Adapting communication-avoiding LU and QR factorizations to multicore architectures*, Proceedings of IEEE International Parallel & Distributed Processing Symposium IPDPS, April 2010.
- S. Donfack, L. Grigori, W. Gropp, and V. Kale, *Hybrid static/dynamic scheduling for already optimized dense matrix factorization* , Proceedings of IEEE International Parallel & Distributed Processing Symposium IPDPS, 2012.
- A. Khabou, J. Demmel, L. Grigori, and M. Gu, *LU factorization with panel rank revealing pivoting and its communication avoiding version*, LAWN 263, 2012.
- L. Grigori, S. Moufawad, *Communication avoiding incomplete LU preconditioner*, in preparation, 2012