Avoiding communication in linear algebra

Laura Grigori
ALPINES
INRIA Rocquencourt - LJLL, UPMC

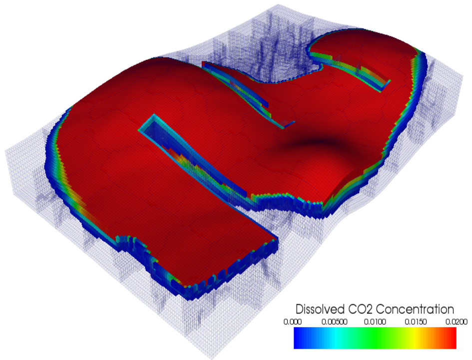
Plan

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

Data driven science

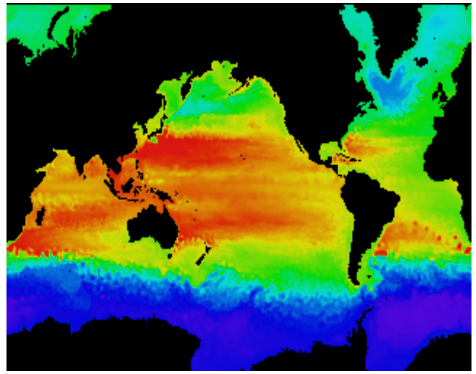
Numerical simulations require increasingly computing power as data sets grow exponentially

CO2 Underground storage



Source: T. Guignon, IFPEN

Climate modeling



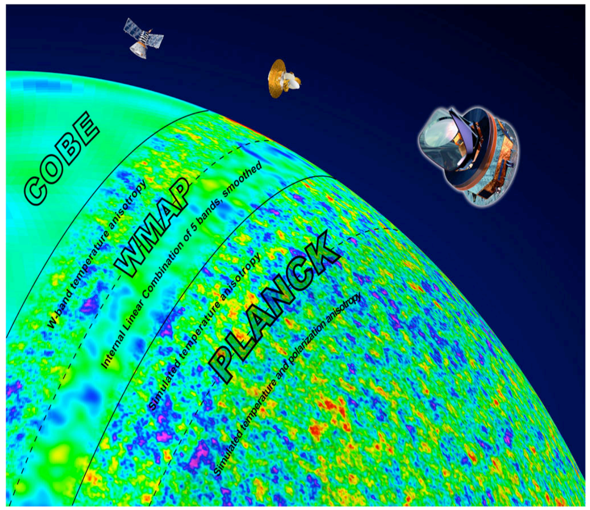
http://www.epm.ornl.gov/chammp/chammp.html

Figures from astrophysics:

- Produce and analyze multi-frequency 2D images of the universe when it was 5% of its current age.
- COBE (1989) collected 10 gigabytes of data, required 1 Teraflop per image analysis.
- PLANCK (2010) produced 1 terabyte of data, requires 100 Petaflops per image analysis.
- CMBPol (2020) is estimated to collect .5 petabytes of data, will require 100 Exaflops per image analysis.

Source: J. Borrill, LBNL, R. Stompor, Paris 7

Astrophysics: CMB data analysis



http://www.scidacreview.org/0704/html/cmb.html

Motivation - the communication wall

- Runtime of an algorithm is the sum of:
  - #flops x **time_per_flop**
  - #words_moved / **bandwidth**
  - #messages x **latency**

- Time to move data >> time per flop
  - Gap steadily and exponentially growing over time

Annual improvements			
Time/flop		Bandwidth	Latency
59%	Network	**26%**	**15%**
	DRAM	**23%**	**5%**

- Performance of an application is less than 10% of the peak performance

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

Motivation

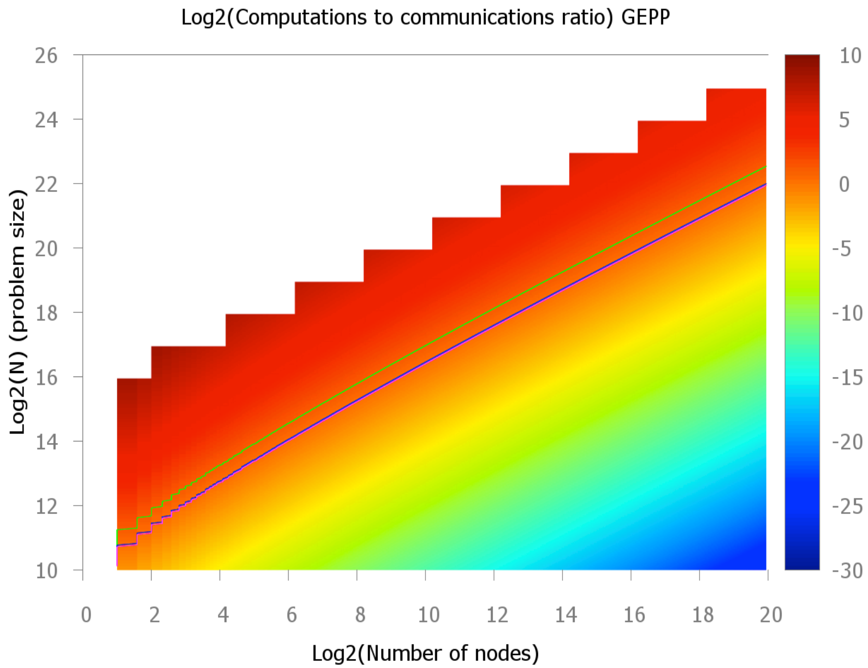
- The communication problem needs to be taken into account higher in the computing stack
- A paradigm shift in the way the numerical algorithms are devised is required
- Communication avoiding algorithms - a novel perspective for numerical linear algebra
  - Minimize volume of communication
  - Minimize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

Previous work on reducing communication

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

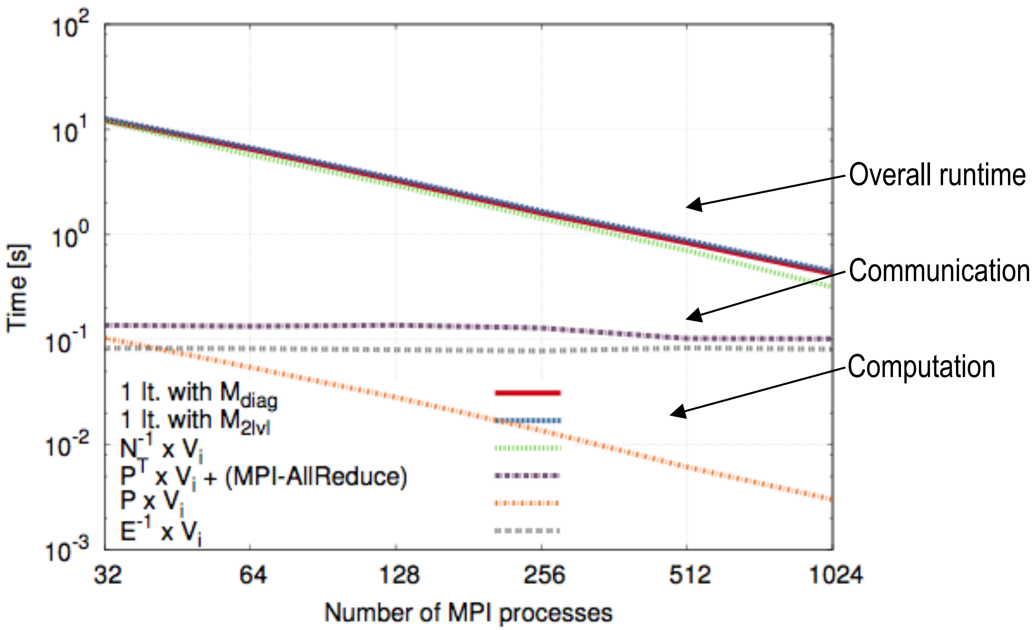
- Ghosting
  - Store redundantly data from neighboring processors for future computations

- Scheduling
  - Block algorithms for linear algebra
    - Barron and Swinnerton-Dyer, 1960
    - ScaLAPACK, Blackford et al 97
  - Cache oblivious algorithms for linear algebra
    - Gustavson 97, Toledo 97, Frens and Wise 03, Ahmed and Pingali 00

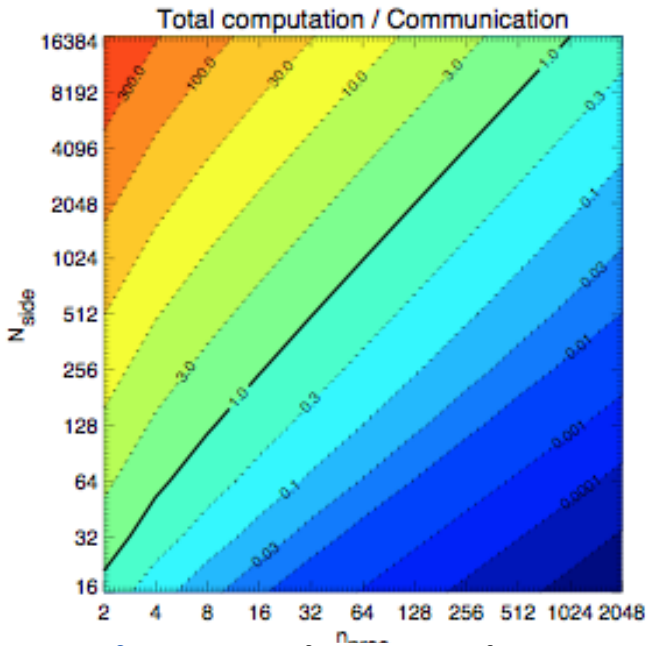


Communication in CMB data analysis

- Map-making problem
  - Find the best map *x* from observations *d*, scanning strategy *A*, and noise N^{-1}
  - Solve generalized least squares problem involving sparse matrices of size 10^{12}-by-10^{7}
- Spherical harmonic transform (SHT)
  - Synthesize a sky image from its harmonic representation
    - Computation over rows of a 2D object (summation of spherical harmonics)
    - Communication to transpose the 2D object
    - Computation over columns of the 2D object (FFTs)



Map making, with R. Stompor, M. Szydlarski
Results obtained on Hopper, Cray XE6, NERSC



SHT, with R. Stompor, M. Szydlarski
Simulation on a petascale computer

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]

2D Parallel algorithms and communication bounds

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

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	ScaLAPACK	[Branescu, Demmel, LG, Gu, Xiang 11] uses tournament pivoting, 3x flops

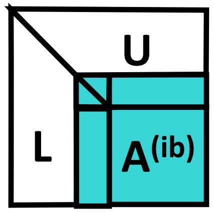
- Only several references shown, block algorithms (ScaLAPACK) and communication avoiding algorithms

LU factorization (as in ScaLAPACK pdgetrf)

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

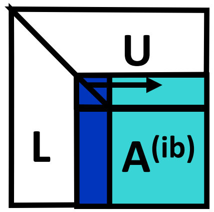
For ib = 1 to n-1 step b

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



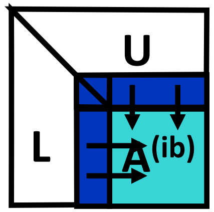
(1) Compute panel factorization $O(n \log_2 P_r)$

- find pivot in each column, swap rows



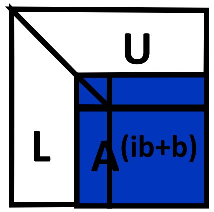
(2) Apply all row permutations $O(n/b(\log_2 P_c + \log_2 P_r))$

- 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

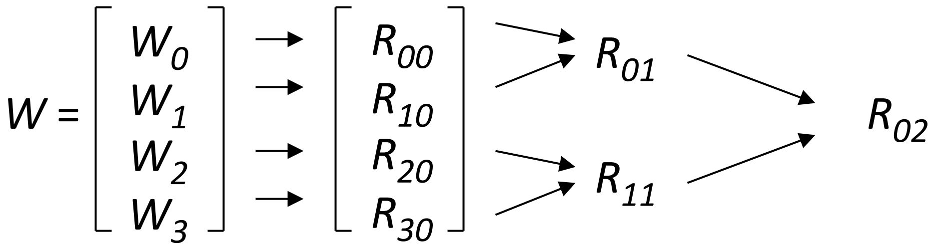


(4) Update trailing matrix $O(n/b(\log_2 P_c + \log_2 P_r))$

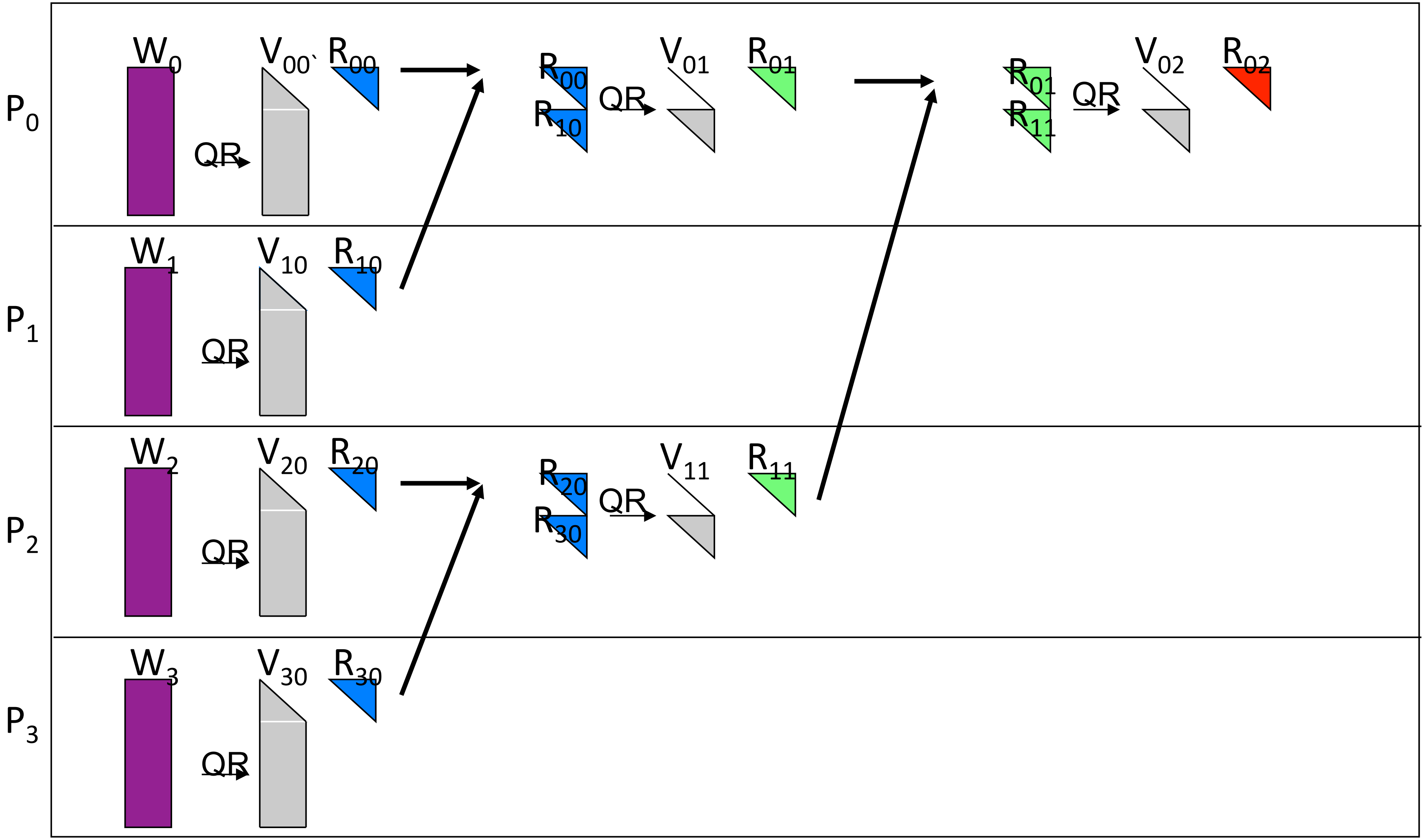
- broadcast right block column of L
- broadcast down block row of U

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



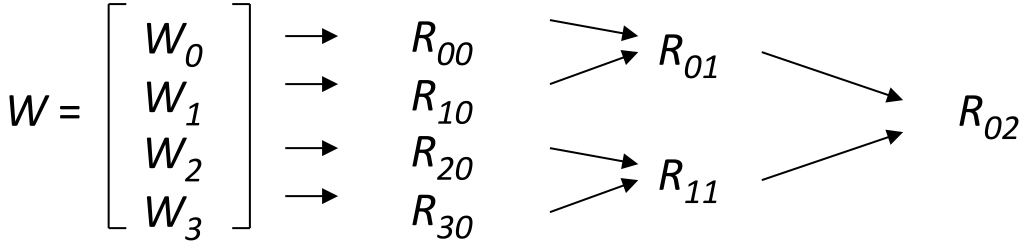
Parallel TSQR



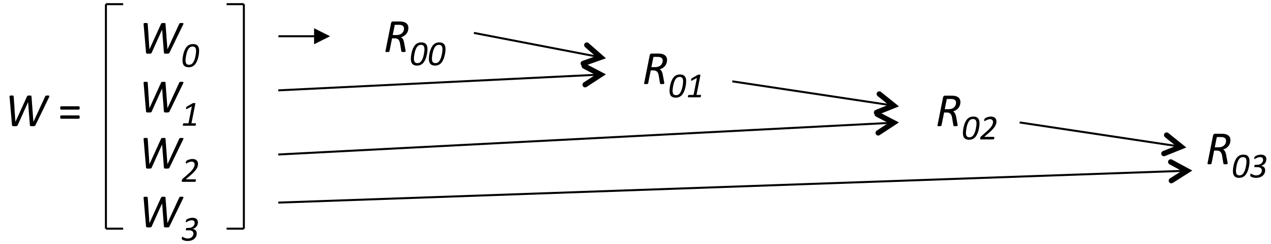
References: Golub, Plemmons, Sameh 88, Pothen, Raghavan, 89, Da Cunha, Becker, Patterson, 02

Flexibility of TSQR and CAQR algorithms

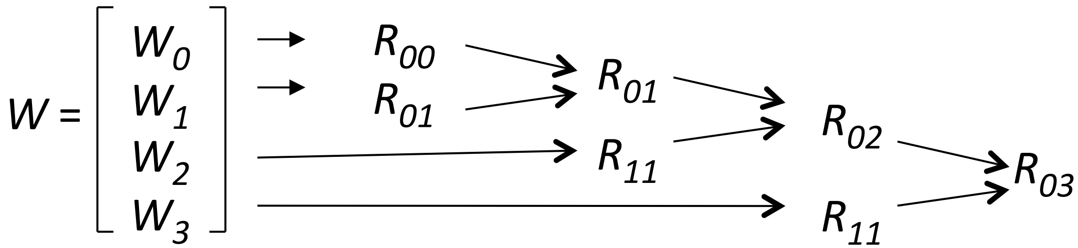
Parallel:



Sequential:

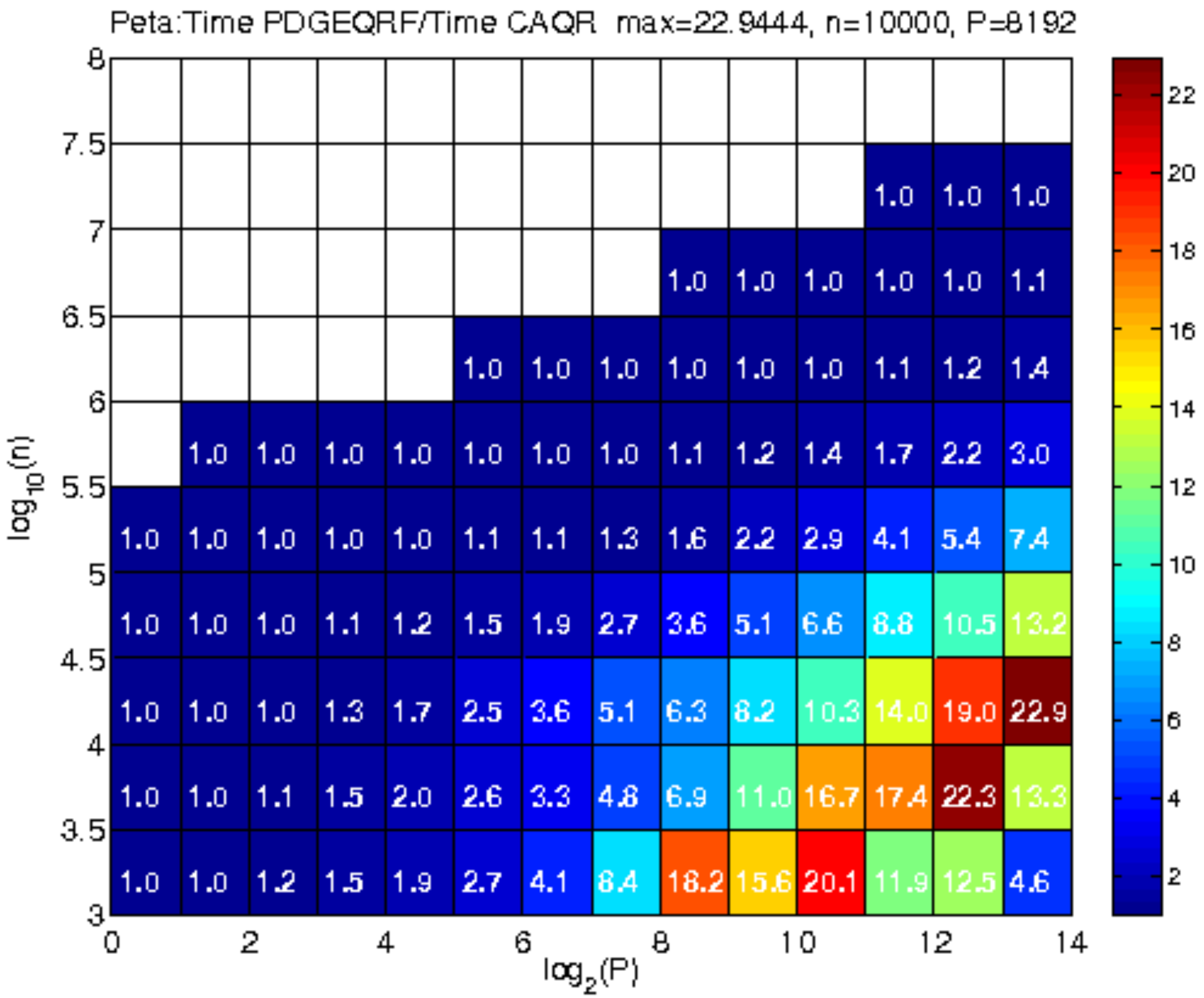


Dual Core:



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

Modeled Speedups of CAQR vs ScaLAPACK



Petascale
up to 22.9x

IBM Power 5
up to 9.7x

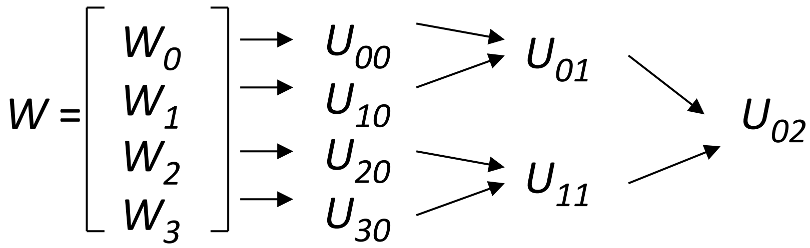
“Grid”
up to 11x

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

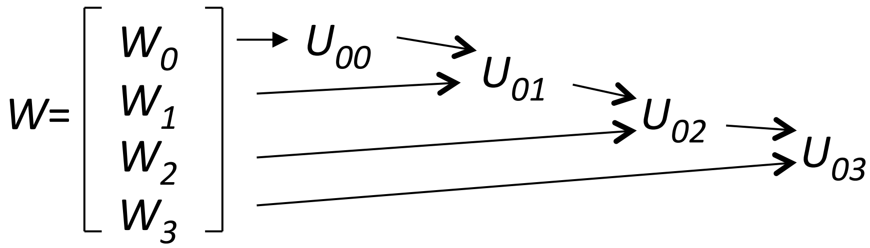
$$\gamma = 2\cdot 10^{-12}s, \alpha = 10^{-5}s, \beta = 2\cdot 10^{-9}s/word.$$

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
  - introduced by Barron and Swinnerton-Dyer, 1960: block LU factorization used to solve a system with 100 equations on EDSAC 2 computer using an auxiliary magnetic-tape
  - 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

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

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.

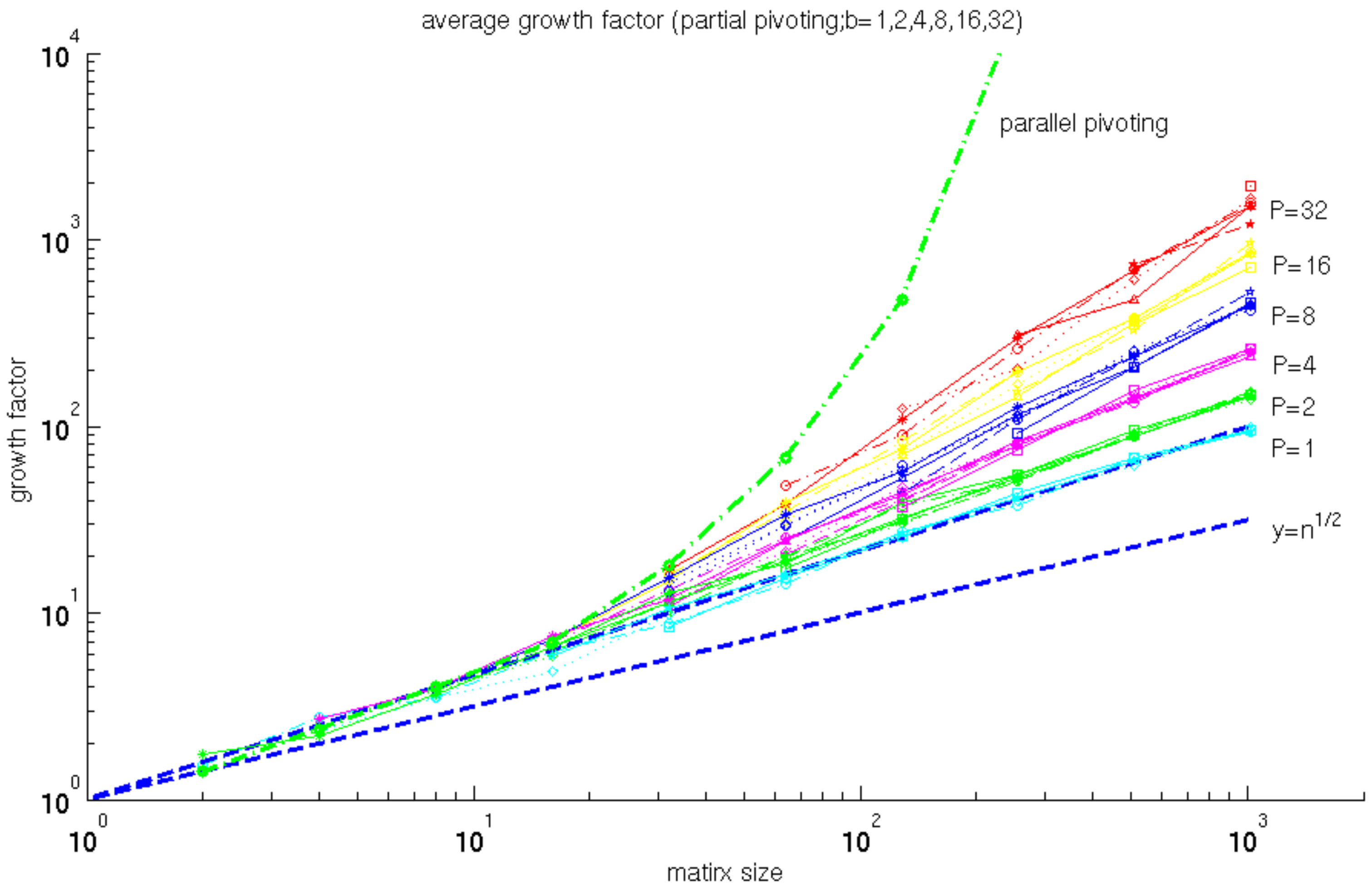
- $g_W \le 2^{n-1}$, but in practice it is on the order of $n^{2/3}$ -- $n^{1/2}$

- 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)

Tournament pivoting - the overall idea

- At each iteration of a block algorithm

$$A = \begin{array}{c} \begin{matrix} \quad b & \qquad n-b \end{matrix} \\ \begin{pmatrix} \overbrace{A_{11}} & \overbrace{A_{21}} \\ A_{21} & A_{22} \end{pmatrix} \end{array} \begin{array}{l} \\ \left.\right\} b \\ \left.\right\} n-b \end{array} \text{, where } \quad W = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$$

  - 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

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

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

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

Good pivots for factorizing W

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

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

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

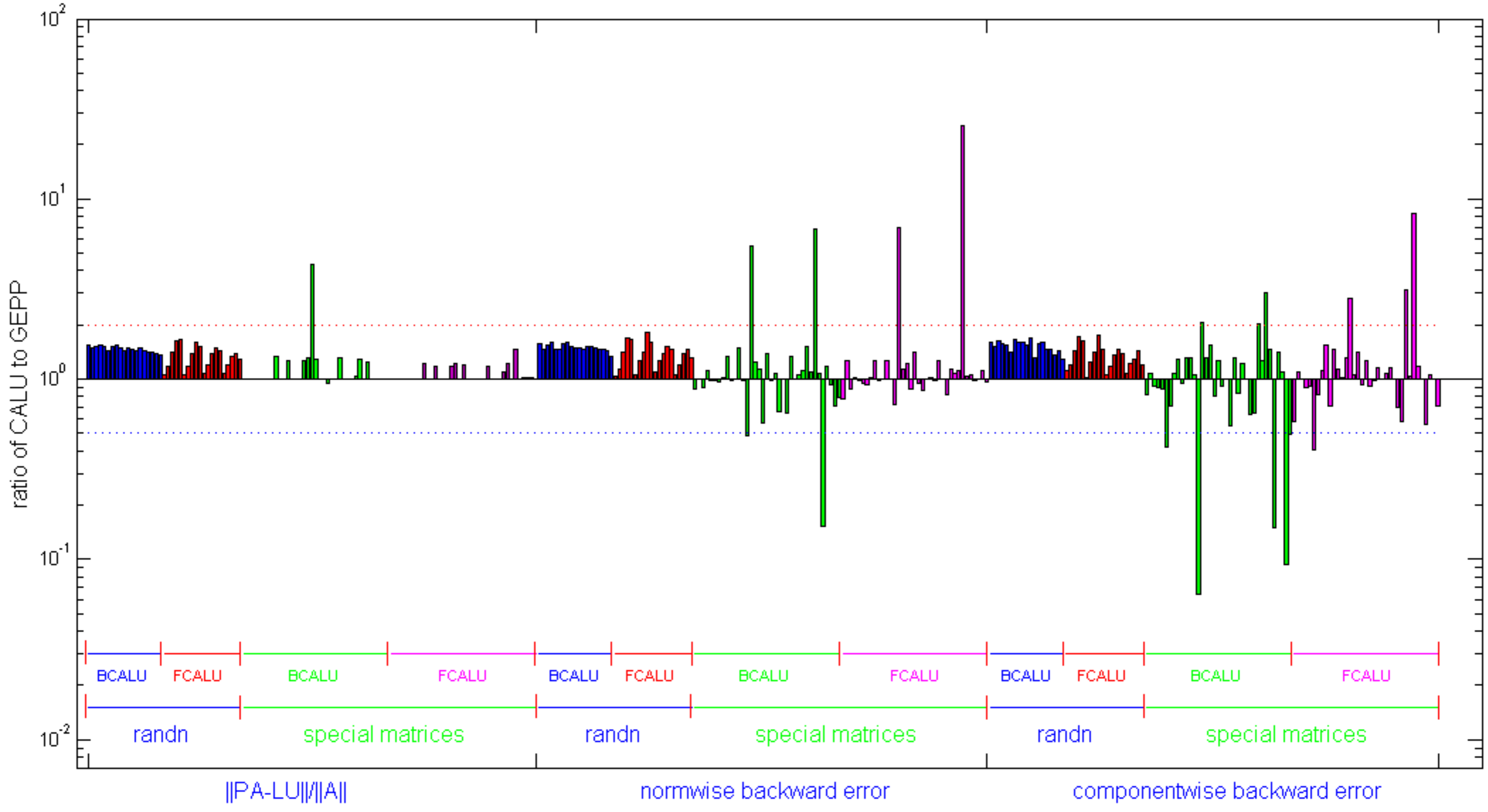
P_3

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

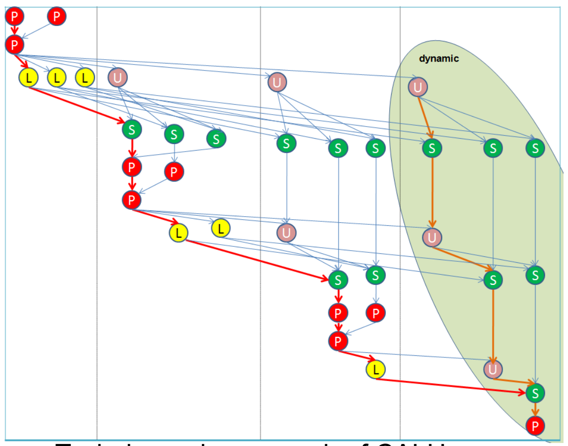
time $\longrightarrow$

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

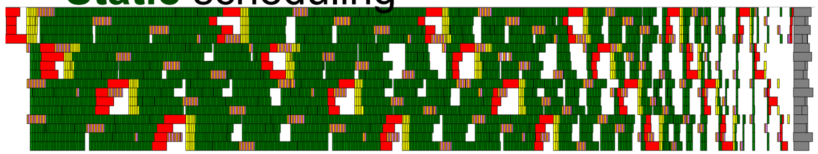


Lightweight scheduling for CALU

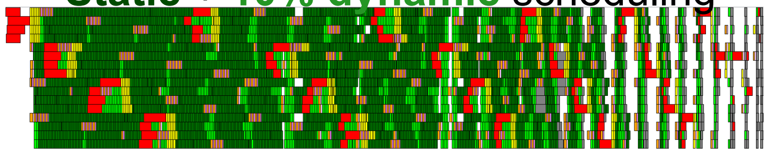


Task dependency graph of CALU

Static scheduling

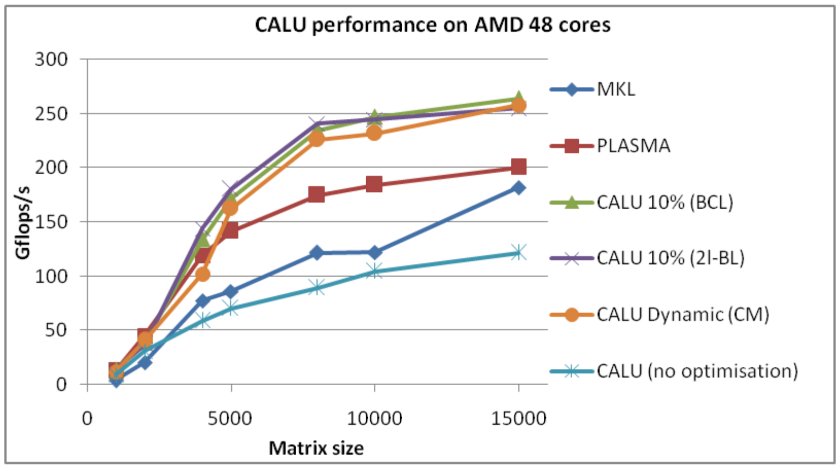


Static + **10% dynamic** scheduling



100% dynamic scheduling

time



Donfack, LG, Gropp, Kale, IPDPS 2012

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
  - Iterative methods and preconditioning
- Conclusions

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)

Minimizing communication in iterative solvers

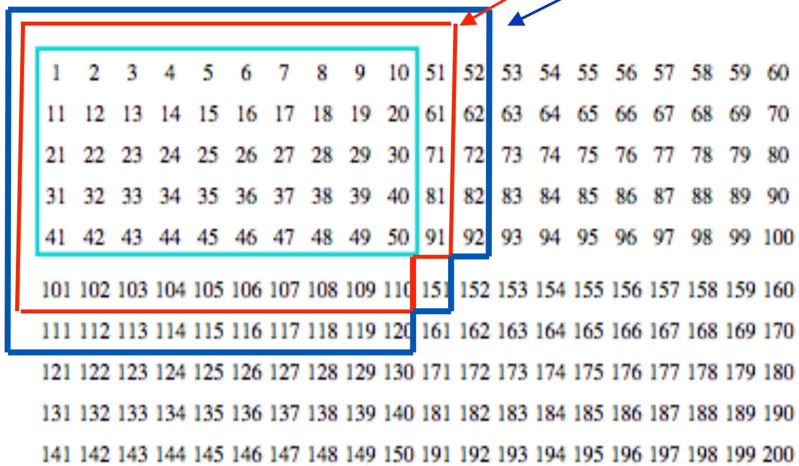
- To minimize communication
  - Generate a set of s vectors (Ab, A^2b, …, A^sb)
  - Orthogonalize the s vectors, check convergence

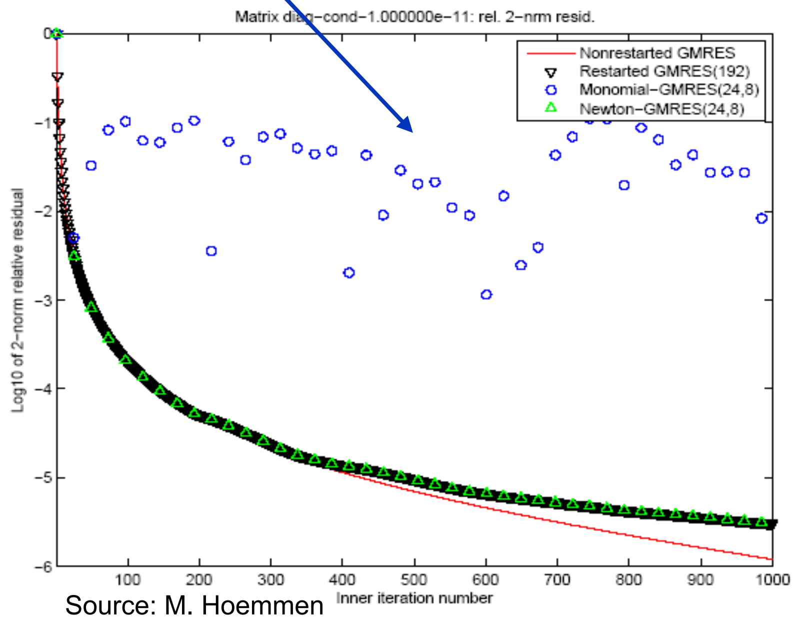
} O(log P) messages, optimal

However

- Important instability problem to address (monomial basis)
- CA-preconditioners to further decrease the number of iterations

Domain and ghost data to compute Ax, A^2 x, …
on one processor with no communication





Source: M. Hoemmen

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

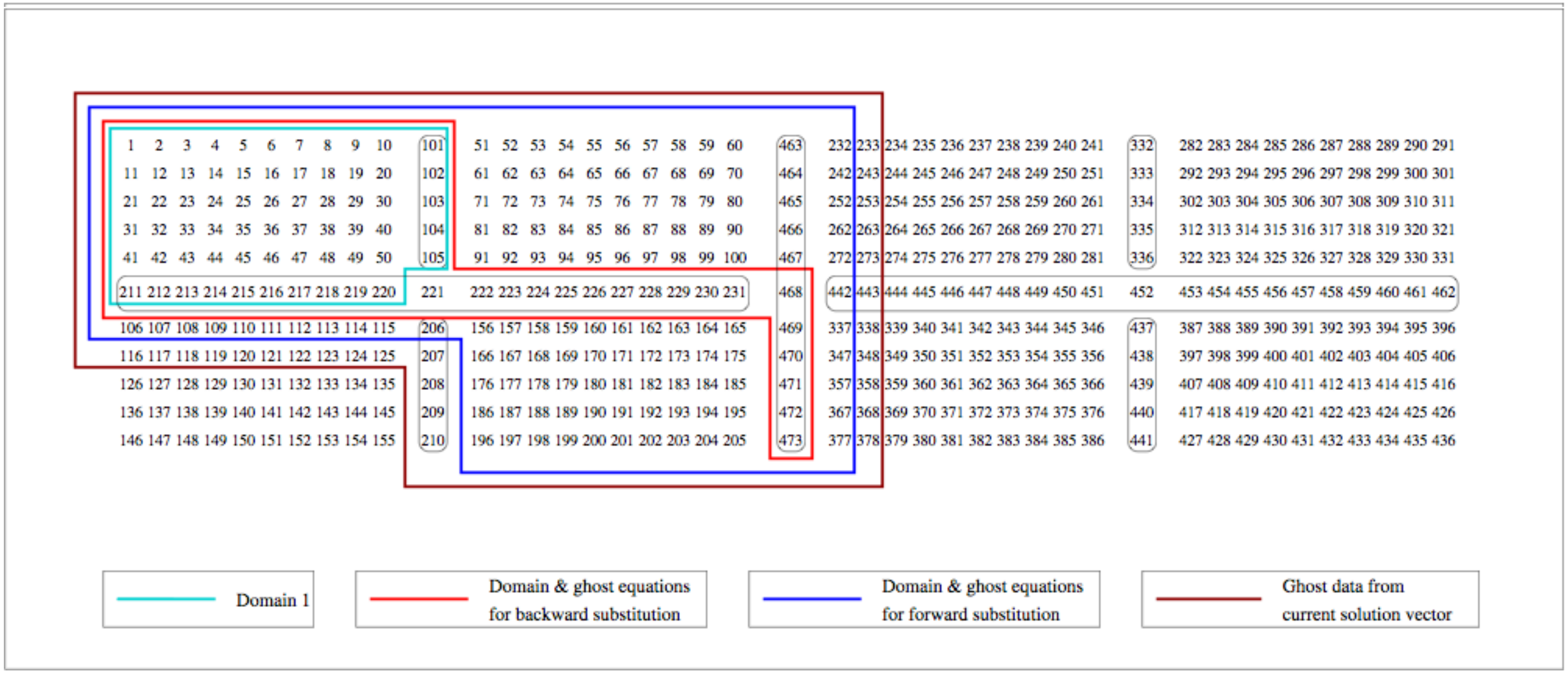
- Incomplete LU factorizations (joint work with S. Moufawad, talk in MS later today)
- Two level preconditioners in DDM
- Deflation techniques through preconditioning

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* $(G(U), \alpha_{j-1})$
  Find γ_j = *ReachableVertices* $(G(L), \beta_j)$
  Find δ_j = *Adj* $(G(A), \gamma_j)$
  Set $\alpha_j = \delta_j$
end

Ghost data required:
$x(\delta)$, $A(\gamma,\delta)$,
$L(\gamma,\gamma)$, $U(\beta, \beta)$

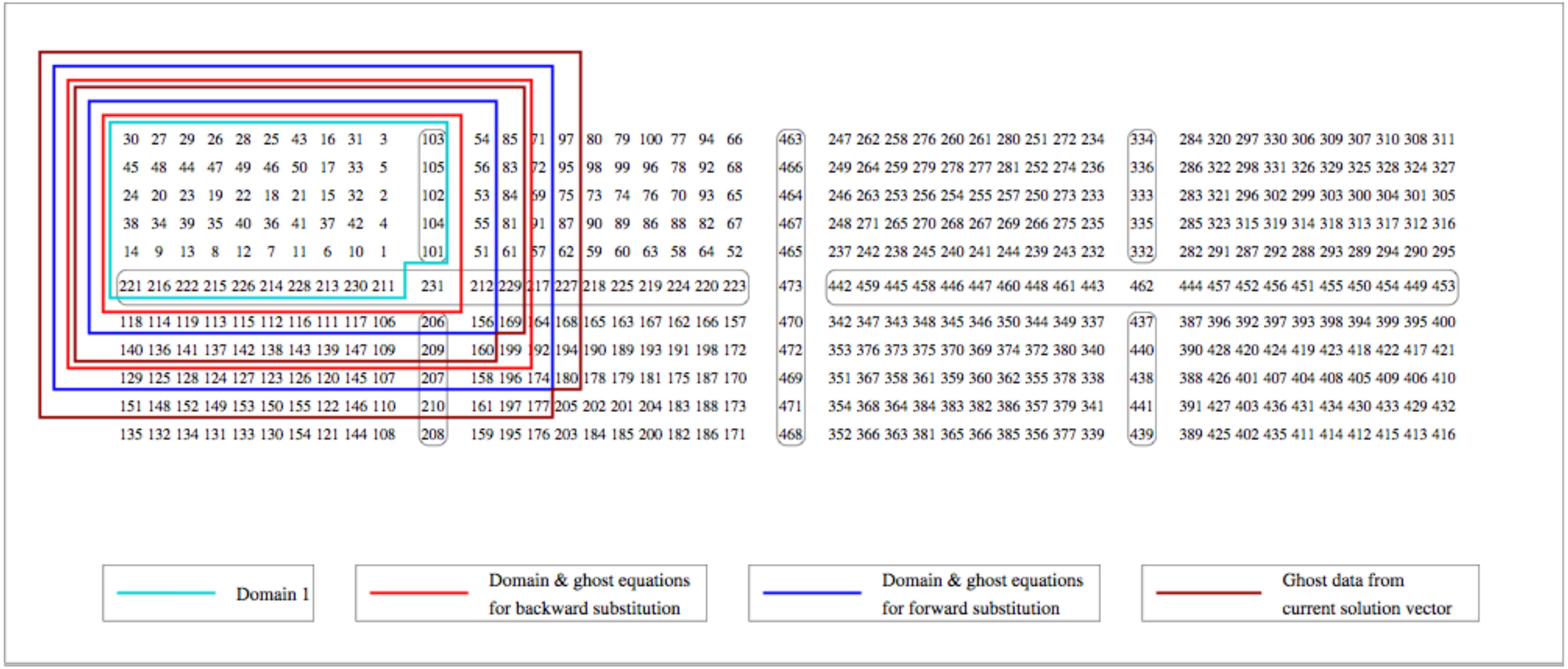
$\Rightarrow$ Half of the work performed on one processor



5 point stencil on a 2D grid

CA-ILU0 with AMML reordering and ghosting

- Reduce volume of ghost data by reordering the vertices using Alternating Min-Max Layers (AMML) reordering:
  - 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

Comparison with Block Jacobi

- Block Jacobi is another preconditioner which does not require communication for one step of an iterative method
- Tests for a boundary value problem (provided by Achdou, Nataf), 40x40x40 grid

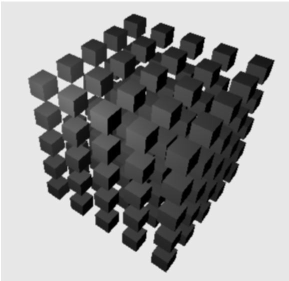
$$-div(\kappa(x)\nabla u) = f \quad in\,\Omega$$

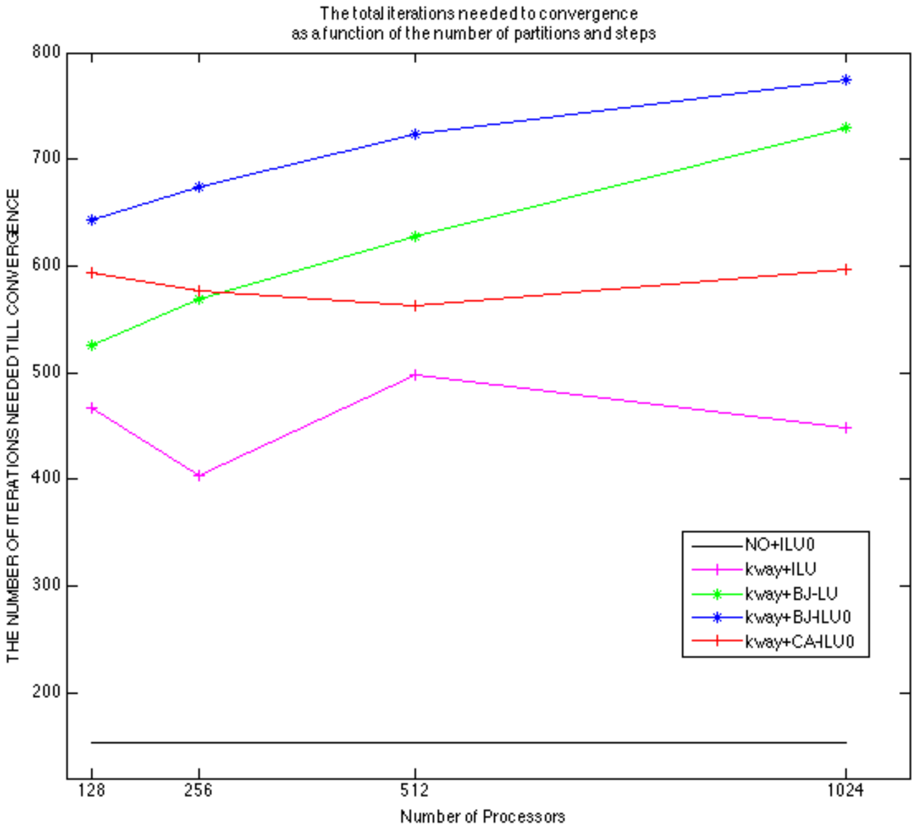
$$u = 0 \quad on\,\partial\Omega_D$$

$$\frac{\partial u}{\partial n} = 0 \quad on\,\partial\Omega_N$$

$$\Omega = [0,1]^3, \partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$$

κ jumps from 1 to 10^3

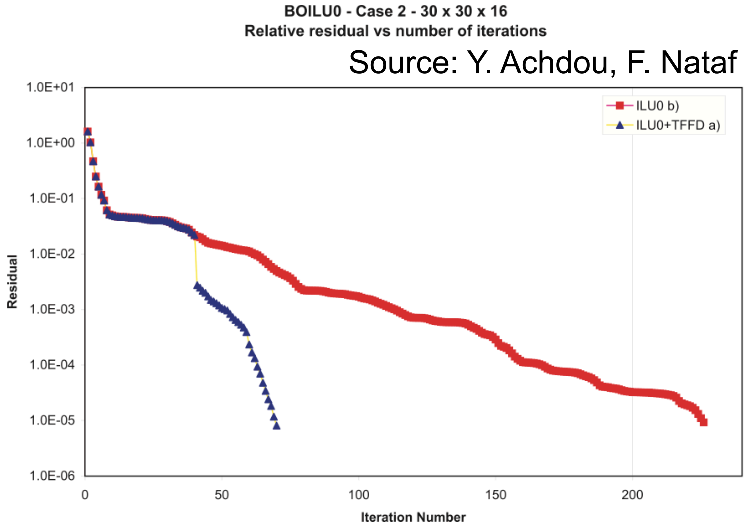




Methods tested:

- Natural ordering NO+ILU0
- CA-ILU0 - kway+AMML(1)+ILU0
- Block Jacobi using LU - BJ+ILU0
- Block Jacobi using ILU0 - BJ-ILU0

Challenge in getting scalable preconditioners



Many preconditioners (as ILU) have plateaus in the convergence, often due to the presence of few low eigenvalues

Direction preserving factorization

- Preconditioner M satisfies a filtering property
  $MT = AT$ or $T^T M = T^T A$
- Filtering vectors T are chosen to improve the convergence

Block Filtering (BFD) and Nested Filtering (NFF) Preconditioners

R. Fezzani, LG, P. Kumar, R. Lacroix, F. Nataf, L. Qu, K. Wang

- Algebraic preconditioners based on nested dissection and block/nested factorization
- Every Schur complement is approximated to satisfy the filtering property:

$$L_{ik} D_{kk}^{-1} U_{kj} t = L_{ik} F_{kj} U_{kj} t, \text{ e.g. } F_{kj} = Diag\left(\left(D_{kk}^{-1} U_{kj} t\right)./\left(U_{kj} t\right)\right)$$

Preserving directions of interest

- Pointwise approximate factorization satisfying a row-sum criteria, Dupont, Kendall, and Rachford (1968), Gustafsson (1978)
  - Improves the condition number of the preconditioned matrix for matrices arising from finite difference approximation of second order elliptic equations
- Nested factorization, Appleyard, Cheshire (1983)
  - If $t^T r_0 = 0$, then at any iteration $t^T r_k = 0$, ensures a mass conservation property
- Filtering factorization, Wagner, Wittum (1997), Achdou, Nataf (2001)

- Direction preserving semiseparable approximation of SPD matrices, Gu, Li, Vassilevski (2010)
  - If the near null-space of the original fine grid matrix is preserved, then view the preconditioner as a coarse discretization matrix
  - Conditioning analysis performed by Napov, components dropped are orthogonal to components preserved
- Multigrid methods
  - Bootstrup AMG (Brandt, Brannick, Kahl, and Livshits)

Results for a boundary value problem

- SKY (provided by Achdou, Nataf), discretized on a 225x225x225 grid (11.3 millions unknowns) and 400x400x400 grid (64 millions unknowns, 447 millions nonzeros)

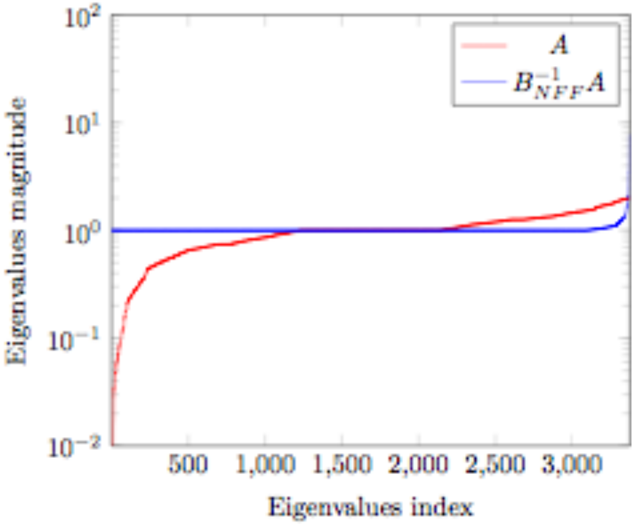
$$-div(\kappa(x)\nabla u) = f \quad in\,\Omega \qquad \Omega = [0,1]^3, \partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$$

$$u = 0 \quad on\,\partial\Omega_D$$

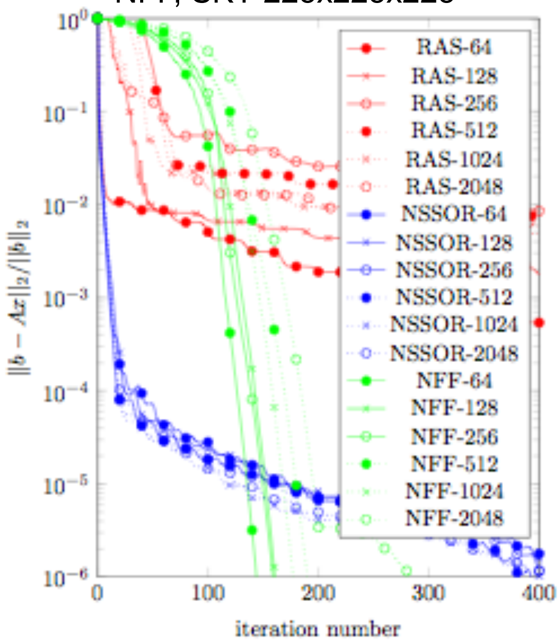
$$\frac{\partial u}{\partial n} = 0 \quad on\,\partial\Omega_N$$

- Tests use GMRES (PETSc), tolerance = 10^{-8}

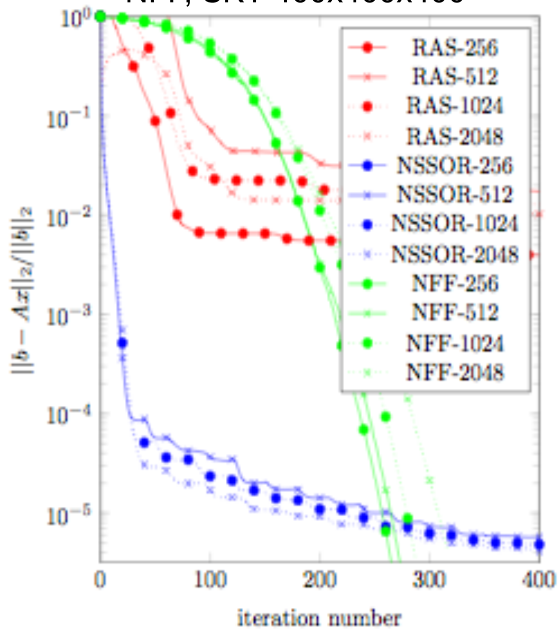
SKY 15x15x15



NFF, SKY 225x225x225



NFF, SKY 400x400x400



Comparison with Restricted Additive Schwarz (RAS)

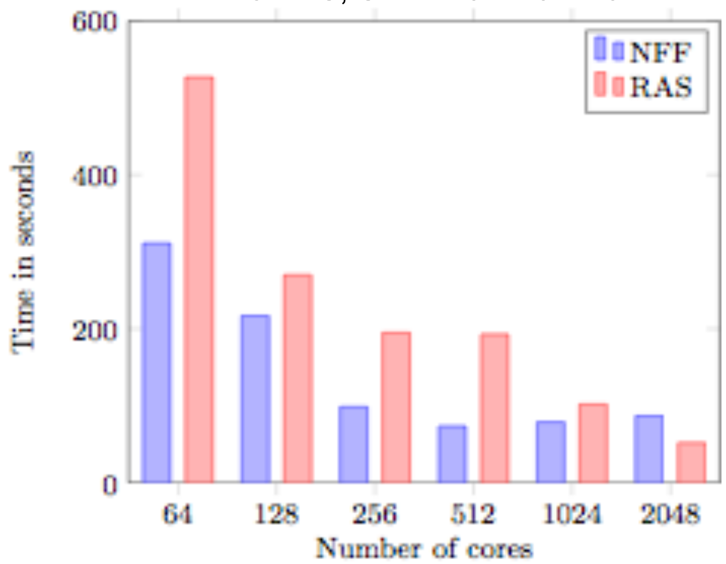
Settings:

- Curie supercomputer based on Bullx system, nodes composed of two eight-core Intel Sandy Bridge.
- Subdomains solved using Pardiso, separators solved using MUMPS.
- GMRES and RAS from PETSc.

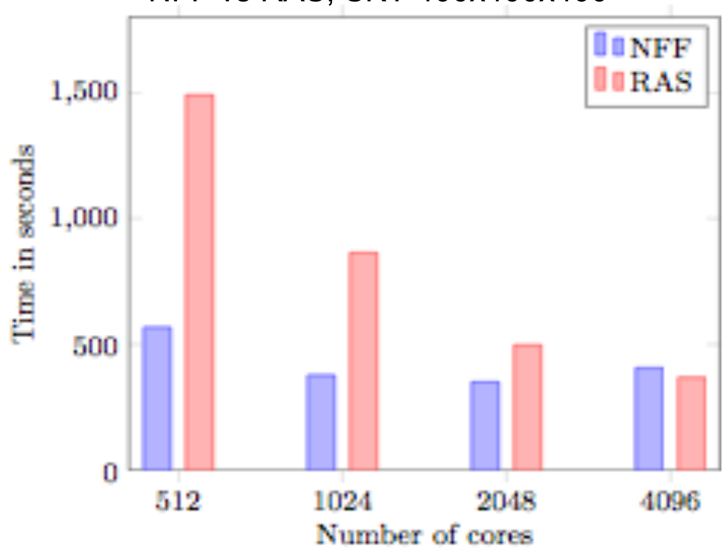
NFF vs RAS, SKY 400x400x400

Subdom	Iteration	Error	Iteration	Error
256	5489	5.9e-7	268	2.2e-6
512	6126	2.7e-6	273	3.2e-6
1024	7163	1.8e-6	289	2.6e-6
2048	10000	3.7e-6	317	3.8e-6

NFF vs RAS, SKY 225x225x225



NFF vs RAS, SKY 400x400x400



Best student paper finalist, Qu, LG, Nataf, SC'13 (talk in MS later today)

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
  - Alternatives as block iterative methods, pipelined iterative methods (Vanroose et al., talk in MS later today)
  - Preconditioners - limited by memory and communication, not flops
- And BEYOND

Collaborators, funding

Collaborators:

- S. Donfack, INRIA, A. Khabou, INRIA, M. Jacquelin, INRIA, L. Qu, Paris 11, F. Nataf, CNRS, S. Moufawad, INRIA, H. Xiang, Wuhan University
- 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