Comunication Optimal Algorithms for Numerical Linear Algebra

Laura Grigori
INRIA Saclay – Ile de France

June 11, 2009

Joint work with J. Demmel, M. Hoemmen (UC Berkeley), J. Langou (CU Denver), and H. Xiang (Paris 6)

Plan

- Research context
- Communication-optimal operations for linear algebra, first results in dense LU and QR factorizations
- Conclusions and future work

Motivation and challenges

- Running time of an algorithm is sum of 3 terms:
  - # flops * time_per_flop
  - # words moved / bandwidth
  - # messages * latency
- Exponentially growing gaps between
  - Time_per_flop << 1/Network BW << Network Latency
    - **Improving 59%/year vs 26%/year vs 15%/year**
  - Time_per_flop << 1/Memory BW << Memory Latency
    - **Improving 59%/year vs 23%/year vs 5.5%/year**
- Goal : reorganize linear algebra to *avoid* communication
  - Not just *hiding* communication
  - Arbitrary speedups possible

Communication Lower Bounds for Dense Linear Algebra – Summary of theory

- Matrix multiply, using $2n^3$ flops (sequential or parallel)
  - Hong-Kung (1981), Irony/Tishkin/Toledo (2004)
  - Lower bound on Bandwidth = Ω (#flops / (local/fast memory size)$^{1/2}$)
  - Lower bound on Latency = Ω (#flops / (local/fast memory size)$^{3/2}$)
  - Attained by usual block algorithm (sequential), Cannon (parallel)
- Same lower bounds apply to LU, QR and Cholesky
  - Assumption: $O(n^3)$ algorithms; LU is easy, Cholesky trickier, QR subtle
- LAPACK and ScaLAPACK do *not* attain these bounds for LU, QR
  - ScaLAPACK attains bandwidth lower bound
    - But sends $O((mn/P)^{1/2})$ times more messages
  - LAPACK attains neither; $O((m^2/W)^{1/2})$ times more bandwidth
- But new LU, QR do attain them, mod polylog factors
  - LU requires a new pivoting scheme, still stable
  - QR requires new representation of Q, $O(n^2)$ more flops
  - Sequential Recursive LU and QR minimize bandwidth, not latency

– Cholesky: ScaLAPACK attains lower bounds, LAPACK just bandwidth

– Joint work with G. Ballard, J. Demmel, L. Grigori, M. Hoemmen, O. Holtz, J. Langou, O. Schwartz

Courtesy of J. Demmel

Lower Bounds on Communication for parallel LU

- Matrix multiplication lower bounds on communication bandwidth (Hong,Kung 1981, Irony/Toledo/Tishkin, 2004) extended to provide latency bounds:
  - each processor has $O(n^2/P)$ memory

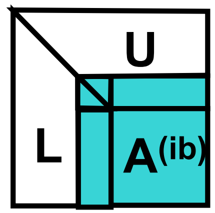
$$\#\,words \geq \Omega\left(\frac{n^2}{\sqrt{P}}\right) \qquad \#\,messages \geq \Omega\left(\sqrt{P}\right)$$

- Bounds hold for LU using a simple example:

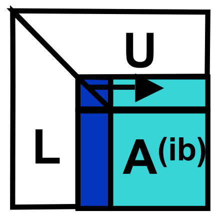
$$\begin{pmatrix} I & & -B \\ A & I & \\ & & I \end{pmatrix} = \begin{pmatrix} I & & \\ A & I & \\ & & I \end{pmatrix} \cdot \begin{pmatrix} I & & -B \\ & I & AB \\ & & I \end{pmatrix}$$

LU factorization (as in ScaLAPACK pdgetrf)

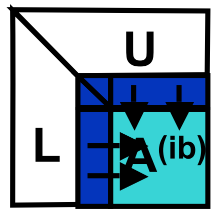
LU factorization on a Pr by Pc grid of processors
For ib = 1 to n-1 step b
    $A^{(ib)}$ = A(ib:n, ib:n)



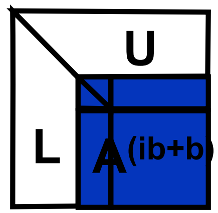
(1) Compute panel factorization (pdgetf2) $O(n \log_2 P_r)$
    - find pivot in each column, swap rows



(2) Apply all row permutations (pdlaswp) $O(n/b(\log_2 P_c + \log_2 P_r))$
    - broadcast pivot information along the rows
    - swap rows at left and right



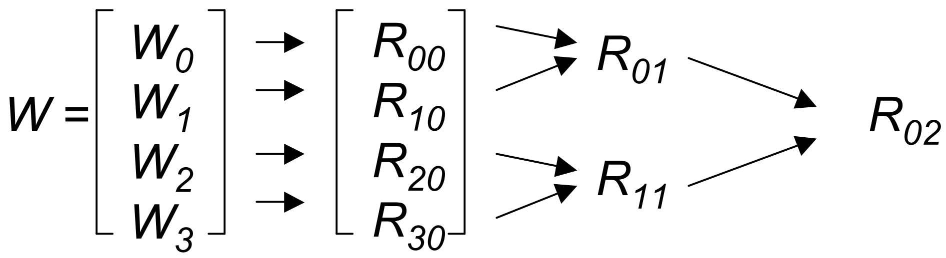
(3) Compute block row of U (pdtrsm) $O(n/b \log_2 P_c)$
    - broadcast right diagonal block of L of current panel



(4) Update trailing matrix (pdgemm) $O(n/b(\log_2 P_c + \log_2 P_r))$
    - broadcast right block column of L
    - broadcast down block row of U

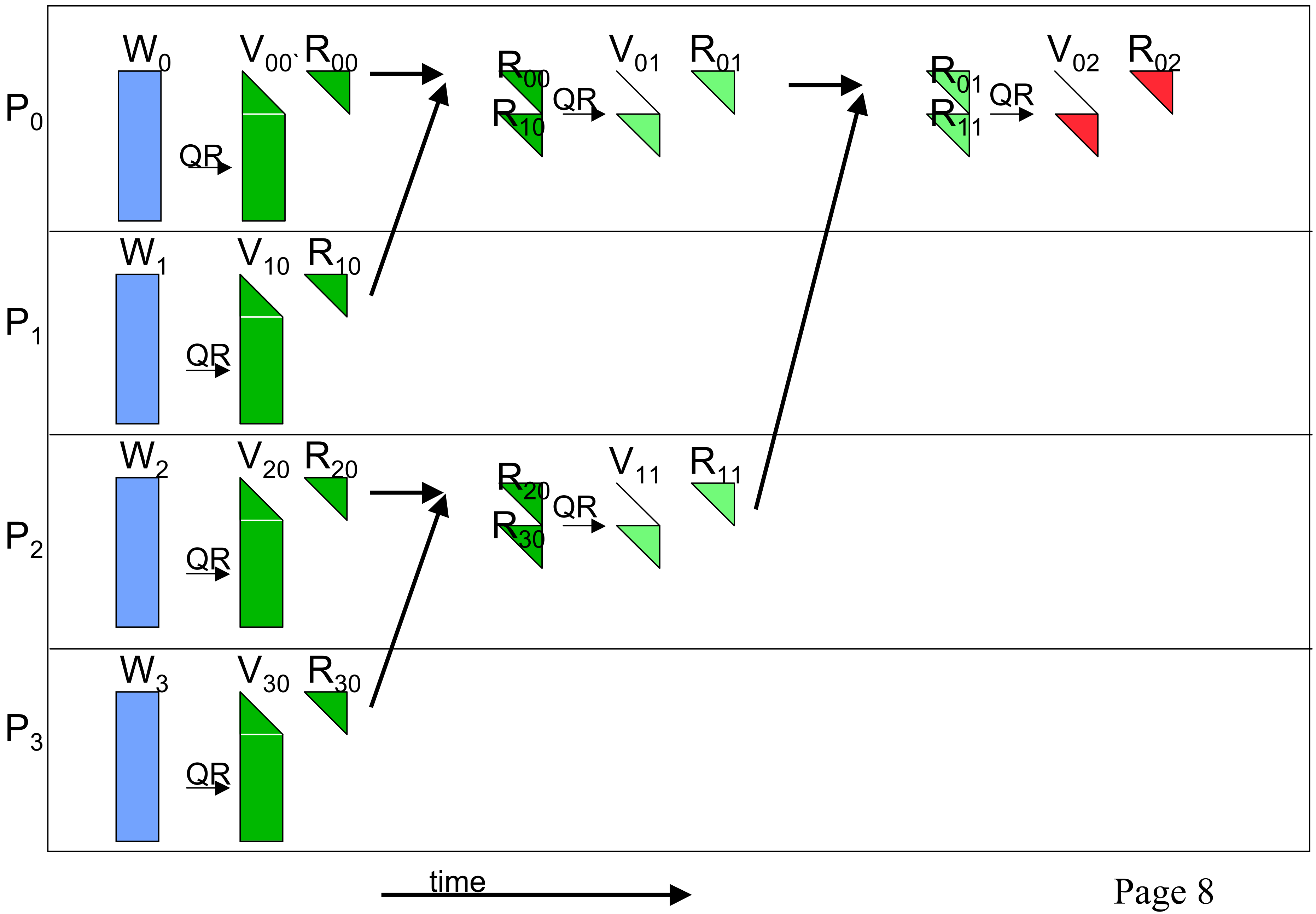
TSQR: an approach for QR factorization of a tall skinny matrix using Householder transformations

- QR decomposition of m x n matrix W, m >> n
  - TSQR = “Tall Skinny QR”
  - P processors, block row layout
- Usual Parallel Algorithm
  - Compute Householder vector for each column
  - Number of messages $\propto$ n log P
- Communication Avoiding Algorithm
  - Reduction operation, with QR as operator
  - Number of messages $\propto$ log P



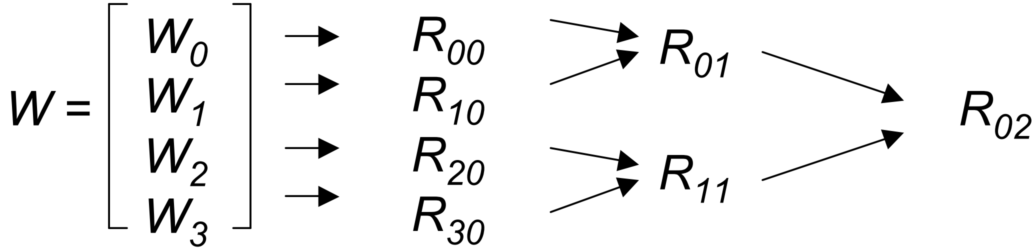
- Joint work with J. Demmel, M. Hoemmen, J. Langou

Parallel TSQR

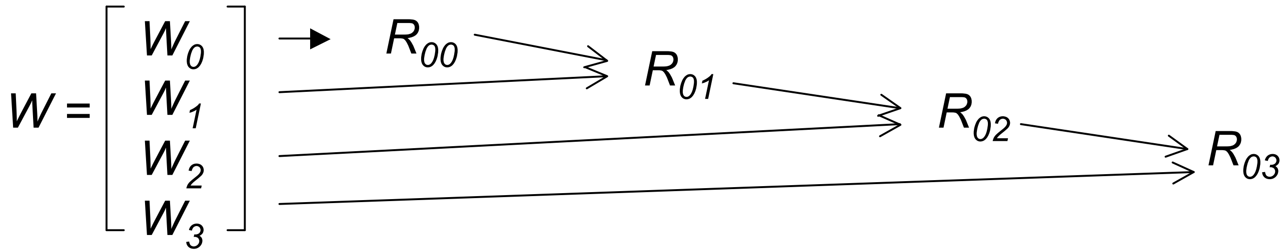


Minimizing Communication in TSQR

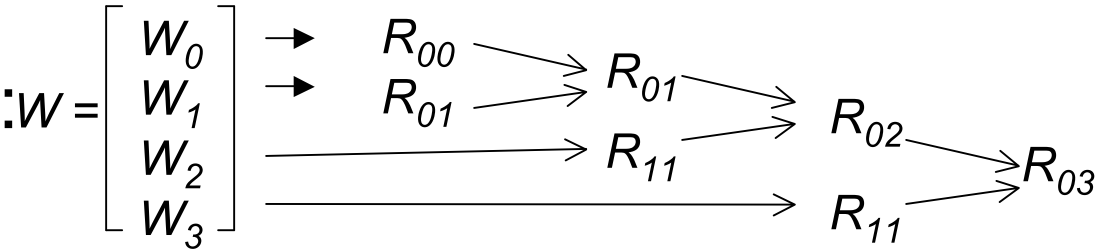
Parallel:



Sequential:



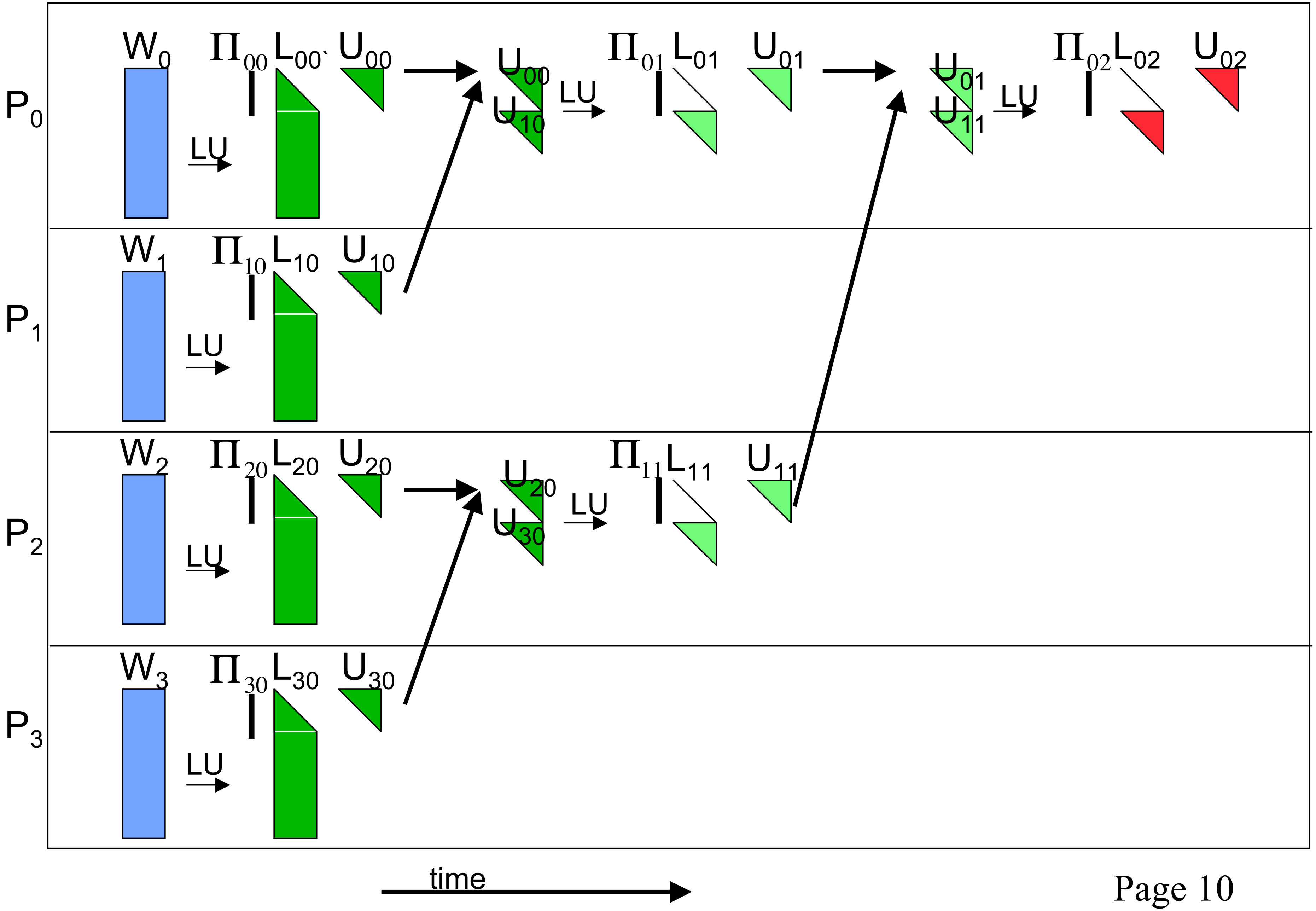
Dual Core:



Multicore / Multisocket / Multirack / Multisite / Out-of-core: ?

Choose reduction tree dynamically

Obvious generalization of TSQR to LU



Stability of the LU factorization

- Consider the growth factor

$$\rho = \frac{\max_{i,j,k} \left| a_{ij}^{k} \right|}{\max_{i,j} \left| a_{ij} \right|} \quad \text{where} \quad a_{ij}^{(k)} \text{ are the values at the k-th step.}$$

- Experiments performed for various distribution of matrices with n < 1024 [Trefethen and Schreiber ’90] showed that the average growth factor normalized by the standard deviation of the initial matrix elements is:

  - close to $n^{2/3}$ for partial pivoting, $n^{1/2}$ for complete pivoting.

- Two reasons considered to be important for the average case stability:

  - the multipliers in L are small,

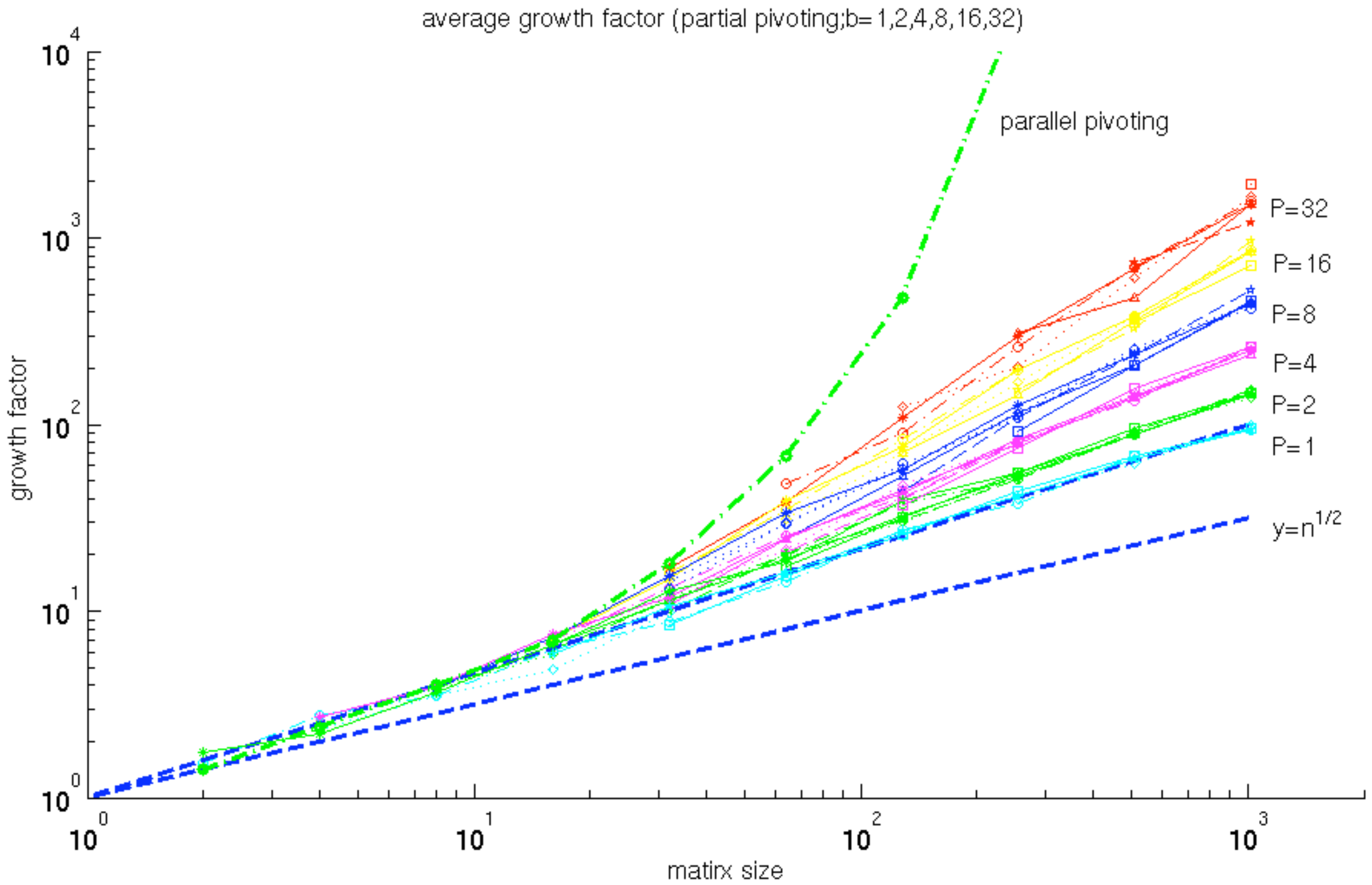
  - the correction introduced at each elimination step is of rank 1.

Other strategies:

- pairwise pivoting considered reasonably stable (low rank correction).

- TSLU involves a rank-P update at each step.

Growth factor for TSLU based factorization



- Unstable for large P and large matrices.
- When P equals the number of rows, TSLU is equivalent to parallel pivoting.

Courtesy of H. Xiang

Making TSLU stable - preprocessing step to find good pivots

P_0

$$\overset{W_0}{\begin{pmatrix} 2 & 4 \\ 0 & 1 \\ 2 & 0 \\ 1 & 2 \end{pmatrix}} = \Pi_0 L_0 U_0 \qquad \overset{\Pi_0^T W_0}{\begin{pmatrix} 2 & 4 \\ 2 & 0 \end{pmatrix}} \longrightarrow \overset{\overline{W}_0}{\begin{pmatrix} 2 & 4 \\ 2 & 0 \\ 4 & 1 \\ 2 & 0 \end{pmatrix}} = \overline{\Pi}_0 \overline{L}_0 \overline{U}_0 \qquad \overset{\overline{\Pi}_0^T \overline{W}_0}{\begin{pmatrix} 4 & 1 \\ 2 & 4 \end{pmatrix}} \longrightarrow \overset{\underline{W}_0}{\begin{pmatrix} 4 & 1 \\ 2 & 4 \\ 4 & 2 \\ 1 & 4 \end{pmatrix}} = \underline{\Pi}_0 \underline{L}_0 \underline{U}_0 \qquad \overset{\underline{\Pi}_0^T \underline{W}_0}{\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix}}$$

Good pivots for factorizing W

P_1

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

P_2

$$\overset{W_2}{\begin{pmatrix} 0 & 1 \\ 1 & 4 \\ 0 & 0 \\ 0 & 2 \end{pmatrix}} = \Pi_2 L_2 U_2 \qquad \overset{\Pi_2^T W_2}{\begin{pmatrix} 1 & 4 \\ 0 & 2 \end{pmatrix}} \longrightarrow \overset{\overline{W}_2}{\begin{pmatrix} 1 & 4 \\ 0 & 2 \\ 4 & 2 \\ 0 & 2 \end{pmatrix}} = \overline{\Pi}_2 \overline{L}_2 \overline{U}_2 \qquad \overset{\overline{\Pi}_2^T \overline{W}_2}{\begin{pmatrix} 4 & 2 \\ 1 & 4 \end{pmatrix}}$$

P_3

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

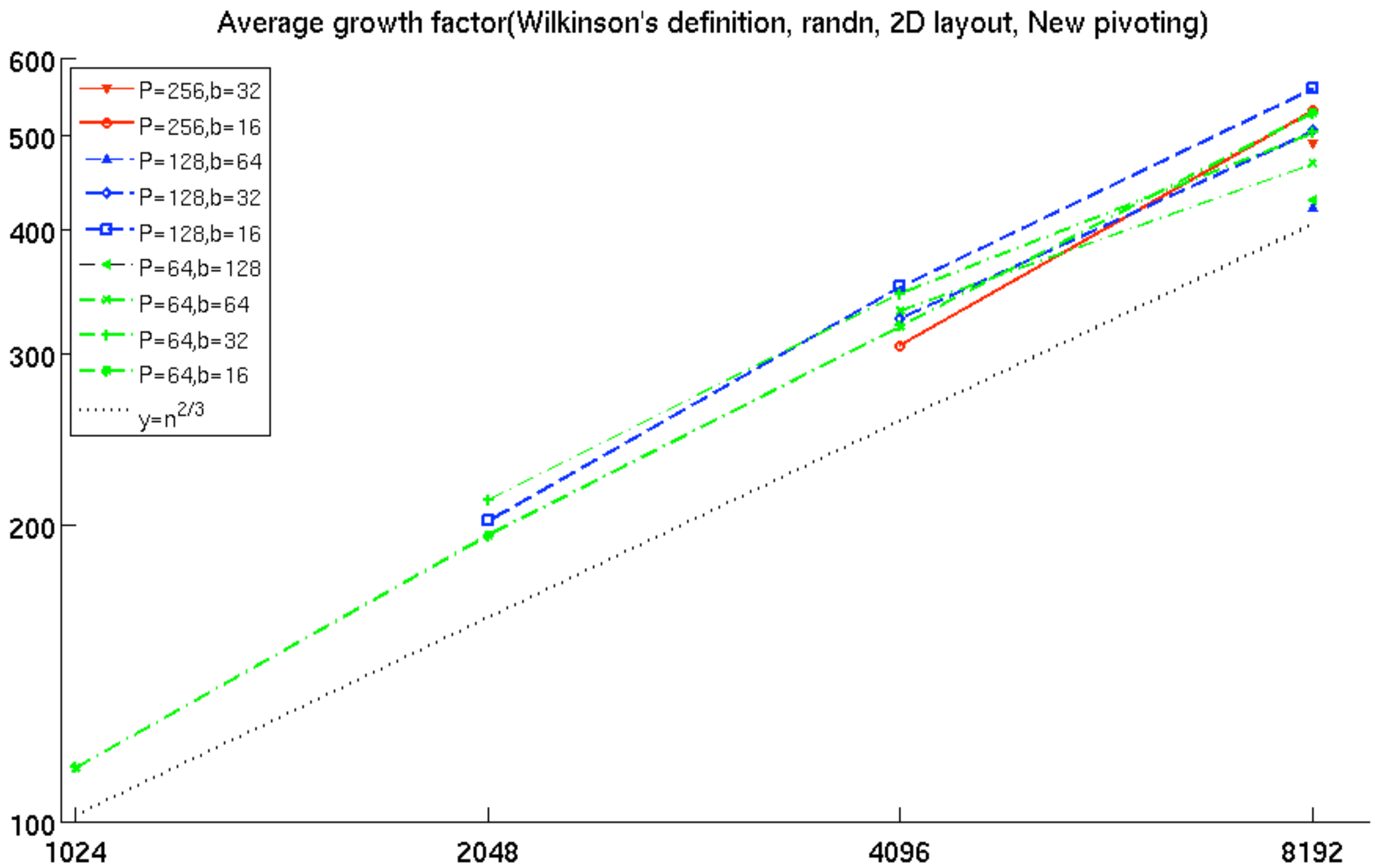
time $\longrightarrow$

Making TSLU stable - the overall idea

- At each node in tree, TSLU selects b pivot rows from 2b candidates from its 2 child nodes
- At each node, do LU on 2b *original* rows selected by child nodes, not U factors from child nodes
- When TSLU done, permute b selected rows to top of original matrix, redo b steps of LU without pivoting

- CALU – Communication Avoiding LU for general A
  - Use TSLU for panel factorizations
  - Apply to rest of matrix
  - Cost: redundant panel factorizations
- Benefit:
  - Stable in practice, but *not* same pivot choice as GEPP
  - b times fewer messages overall - faster
- Joint work with J. Demmel, H. Xiang

Growth factor for CALU approach



Like threshold pivoting with worst case threshold = .33 , so |L| <= 3

Testing shows about same residual as GEPP

Stability of CALU

- Consider the *m-by-n* matrix *W*, where W_1 and W_3 are *b-by-b*, and suppose the permutation returned by TSLU is the identity.

$$W = \begin{pmatrix} W_1 & W_5 \\ W_2 & W_6 \\ \hline W_3 & W_7 \\ W_4 & W_8 \end{pmatrix}$$

TSLU: $\begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} \begin{matrix} \to W_1 \\ \\ \to W_3 \end{matrix} \begin{matrix} \searrow \\ \nearrow \end{matrix} W_1$

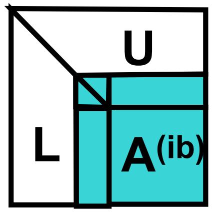
- After the factorization of first panel by CALU, W_8^s (the Schur complement of W_8) is not bounded as in GEPP, but:

$$\begin{pmatrix} W_1 & & W_1 & W_5 \\ & W_3 & W_3 & W_7 \\ W_3 & & 2W_3 & W_7 \\ & -W_4 & & -W_8 \end{pmatrix} = \widehat{L} \cdot \begin{pmatrix} I_{3b} & I_{n-b} \\ & -W_8^s \end{pmatrix} \cdot \widehat{U}$$

- Pivot growth is no worse than PP applied to a different matrix whose entries are the same as the entries of the original matrix.

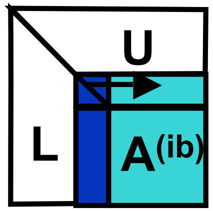
CALU – a communication avoiding LU factorization

- Consider a 2D grid of P processors P_r-by-P_c , using a 2D block cyclic layout with square blocks of size b.



For ib = 1 to n-1 step b

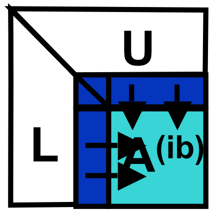
  $A^{(ib)}$ = A(ib:n, ib:n)



(1) Find permutation for current panel using TSLU $O(n/b\log_2 P_r)$

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

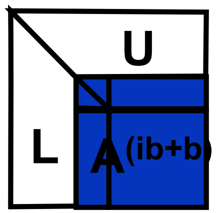
- broadcast pivot information along the rows of the grid



(3) Compute panel factorization (dtrsm)

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

- broadcast right diagonal part of L of current panel



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

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

LU for General Matrices

- Cost of CALU vs ScaLAPACK's PDGETRF
  - n x n matrix on $P^{1/2}$ x $P^{1/2}$ processor grid, block size b
  - Flops: $(2/3)n^3/P + (3/2)n^2 b / P^{1/2}$ vs $(2/3)n^3/P + n^2 b/P^{1/2}$
  - Bandwidth: $n^2 \log P/P^{1/2}$ vs same
  - Latency: $3 n \log P / \mathbf{b}$ vs $1.5 n \log P + 3.5 n \log P / b$

- Close to optimal (modulo log P factors)
  - Assume: $O(n^2/P)$ memory/processor, $O(n^3)$ algorithm,
  - Choose b near $n / P^{1/2}$ (its upper bound)
  - Bandwidth lower bound: $\Omega(n^2 / P^{1/2})$ – just log(P) smaller
  - Latency lower bound: $\Omega(P^{1/2})$ – just polylog(P) smaller

Evaluation of the performance

- Experiments performed on two platforms at NERSC:

IBM p575 POWER 5 system, 111 compute nodes, 8 processors per node

- each processor is clocked at 1.9 GHz, theoretical peak of 7.6 GFLOPs/sec.
- each node has 32 GB memory.
- MPI Point to Point internode Latency is 5 usec.
- peak Bandwidth is 3100 MB/sec.

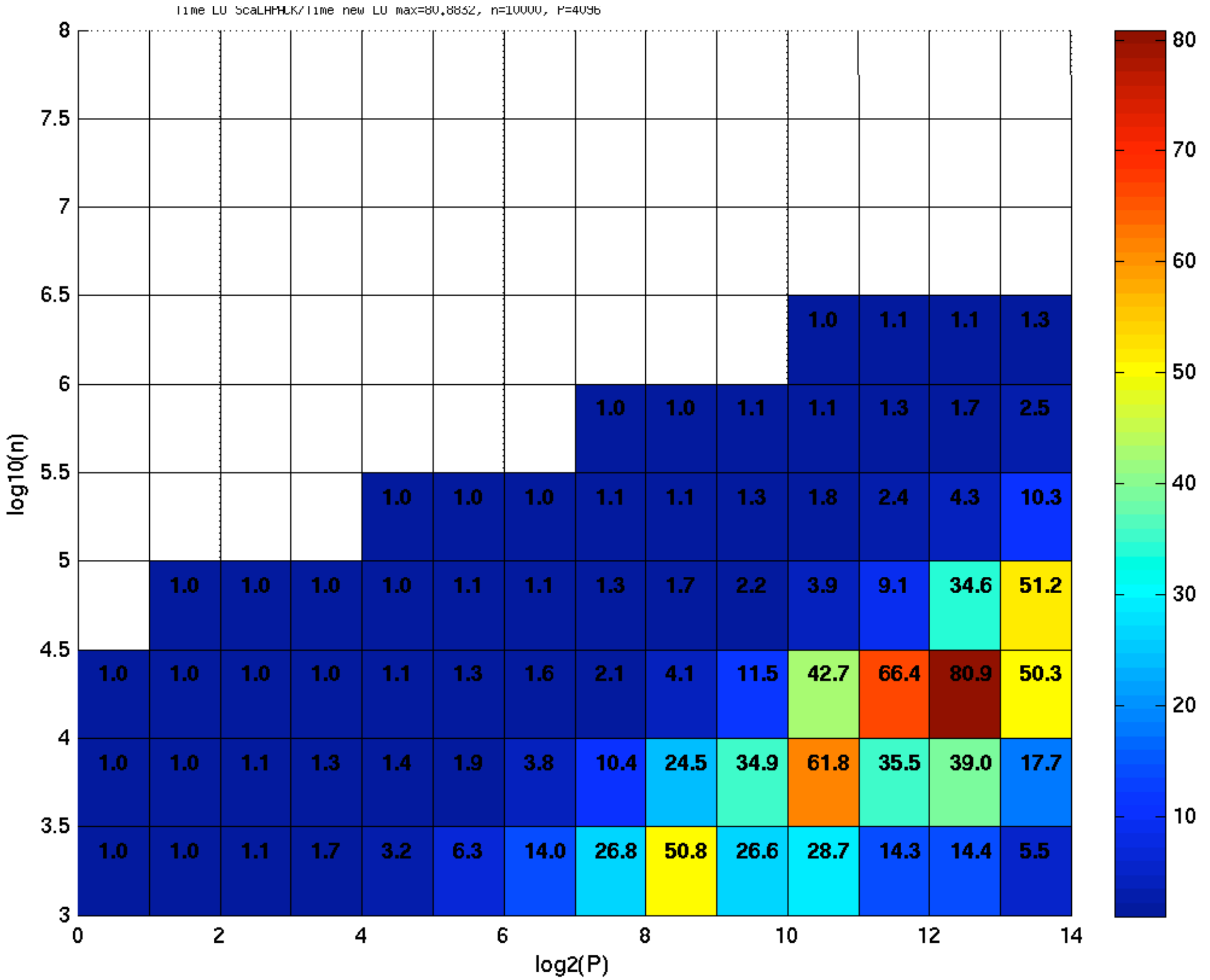
Opteron cluster with 356 dual-processor nodes

- each node has 6 GB memory
- each processor is clocked at 2.2 GHz, theoretical peak of 4.4 GFLOPs/sec.
- Switch MPI Unidirectional Latency is 4.5 usec.
- peak Switch MPI Unidirectional Bandwidth is 620 MB/sec.

Performance vs ScaLAPACK

- TSQR
  - Pentium III cluster, Dolphin Interconnect, MPICH
    - Up to **6.7x speedup** (16 procs, 100K x 200)
  - BlueGene/L - Up to **4x speedup** (32 procs, 1M x 50)
- TSLU
  - 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)
- CALU
  - 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)
- All use recursive algorithms (Toledo, Elmroth-Gustavson) locally.

Speedup prediction for a Petascale machine - up to 81x



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

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

Conclusions

- Possible to minimize communication complexity of much dense and sparse linear algebra
  - Practical speedups
  - Approaching theoretical lower bounds

- The new algorithms minimize the number of messages exchanged at the cost of some redundant computation.

- Recent work extends the bounds to sparse matrix multiplication and sparse direct factorizations.

Conclusions

- *Lots* of prior work, some recent work
  - Idea of binary reduction tree for parallel QR previously studied by
    - Golub, Plemmons, Sameh 1988 - first to suggest the idea
    - Pothen, Raghavan, 1989 - implement it using logP messages
  - Flat trees algorithms, 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).
- And some ideas are *new*
  - Parallel CAQR, CALU
  - Bounds on communication

Future Work

- Many open problems

- Automatic tuning - choose the right communication pattern/tree.
- Extend optimality proofs to general architectures
- Which preconditioners work?