Communication avoiding parallel algorithms for dense matrix factorizations

Edgar Solomonik

Department of EECS, UC Berkeley

October 2013

Outline

Collaborators

- Advising: James Demmel, Kathy Yelick
- Communication lower bounds: Nicholas Knight, Erin Carson
- QR factorization: Grey Ballard, Mathias Jacquelin, Laura Grigori

What costs do we consider?

We count three architectural costs

- α – network processor-to-processor latency
- β – time to transfer a word of data between two processors
- γ – time to perform a floating point operation on local data

We consider three algorithmic costs

- S – number of messages sent
- W – number of words of data moved
- F – number of local floating point operations performed

The time of the algorithm is then bound by these parameters

$$\max(S \cdot \alpha, W \cdot \beta, F \cdot \gamma) \leq \text{execution time} \leq S \cdot \alpha + W \cdot \beta + F \cdot \gamma.$$

How do we measure algorithmic costs?

Would like to measure costs "along the critical path"

- The "critical path" is the longest execution path during execution
- Would like to obtain lower and upper bounds on the length of the critical path

Communication lower bound approach

- Define a computation according to its higher level dependency structure
- Consider all possible parallel schedules and prove bound on shortest path

Algorithmic analysis

- Define a parallel schedule for a computation
- Consider all paths in the schedule and bound their length

Parallel communication lower bounds

For matrix multiplication of n-by-n matrices, in 1981 Hong and Kung proved

$$W = \Omega\left(\frac{n^3}{\sqrt{M}}\right)$$

In 2004, Irony, Tiskin, and Toledo proved

$$W_p = \Omega\left(\frac{n^3}{p \cdot \sqrt{M}}\right)$$

In 2010, Ballard, Demmel, Holtz, and Schwartz showed that this bound also holds for LU and QR factorizations, among other algorithms.

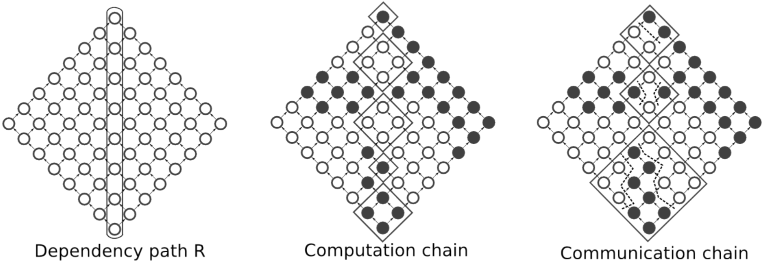
Dependency bubble expansion

Recall that a balanced vertex separator Q of a graph $G = (V, E)$, splits $V - Q = W_1 + W_2$ so that $min(|W_1|, |W_2|) \geq \frac{1}{4}|V|$ and $E = W_1 \times (Q + W_1) + W_2 \times (Q + W_2)$.

Definition (Dependency bubble cross-section expansion)

If $B(R)$ is the dependency bubble formed around path R, the **bubble cross-section expansion**, $E(R)$ is the minimum size of a balanced vertex separator of $B(R)$.

Dependency bubble expansion along path



General latency lower-bound based on bubble expansion

Theorem (Bubble Expansion Theorem)

Let P be a dependency path in G, such that any subsequence $R \subset P$, has bubble cross-section expansion $E(R) = \Omega(\epsilon(|R|))$ and bubble size $|B(R)| = \Omega(\sigma(|R|))$, where $\epsilon(b) = b_1^d$,and $\sigma(b) = b_2^d$ for positive integers d_1, d_2 The bandwidth and latency costs of any parallelization of G must obey the relations

$$F = \Omega(\sigma(b) \cdot |P|/b), \qquad W = \Omega(\epsilon(b) \cdot |P|/b), \qquad S = \Omega(|P|/b)$$

for all $b \in [1, |P|]$.

Appliation: LU factorization

We can use bubble expansion to prove better latency lower bounds for LU factorization. LU factorization of square matrices gives a cubic DAG $v_{ijk} = (L_{ik} \cdot U_{kj})$, where

$$A_{ij} = \sum_{k \leq \min(i,j)} L_{ik} \cdot U_{kj}.$$

Theorem (Latency-bandwidth Trade-off in LU Factorization)

The parallel computation of lower-triangular L and upper-triangular U such that $A = LU$ where all matrices are n-by-n, must incur flops cost F, latency cost S, and bandwidth cost W, such that

$$W \cdot S = \Omega(n^2) \ \ \textit{and} \ \ F \cdot S^2 = \Omega(n^3)$$

Parallel Householder QR

Distribute the starting matrix $A^0 = A$ in a block-cyclic fashion

- For the ith column in the matrix
  1. Compute the norm of the column (requires communication of norm)
  2. Compute the Householder vector y_i from the column and its norm
  3. Update the trailing matrix $A^{i+1} = (I - \tau_i y_i {y_i}^T)A^i$ (requires communication of y_i)
- Continue on to the next column of the trailing matrix

Aggregation of the trailing matrix update

The algorithm in the previous slide uses BLAS 2 routines, would like to use BLAS 3

- We can aggregate k Householder vectors into larger matrices using (Puglisi 1992)

$$\prod_{i=1}^{k}(I - \tau_i y_i y_i^T) = I - YTY^T$$

- T can be computed from Y using the identity

$$Y^T Y = T^{-1} + T^{-T}$$

- Using the aggregated form we may update the trailing matrix via matrix multiplications

Communication costs of Householder QR

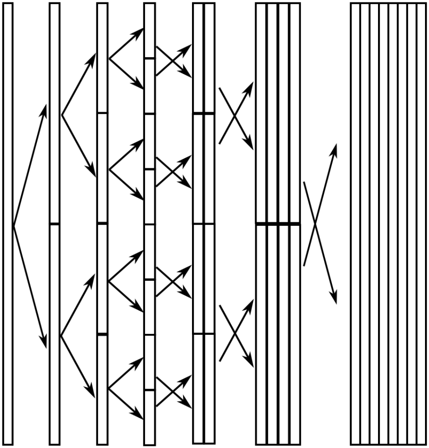
Assuming a square matrix and processor grid the cost of Householder QR is

$$\gamma \cdot (2mn^2 - 2n^3/3)/p + \beta \cdot (mn + n^2)/\sqrt{p} + \alpha \cdot n \log p.$$

- Note that the bandwidth cost does not have a $\log p$ factor, because we can broadcast n words in $O(n)$ time
- The bandwidth cost is optimal so long as there is no extra available memory
- However, the latency cost is much higher than the lower bound ($O(n \log p)$ vs $O(\sqrt{p})$)

Collective communication via recursive halving and doubling

Diagram for broadcast on 8 processors:

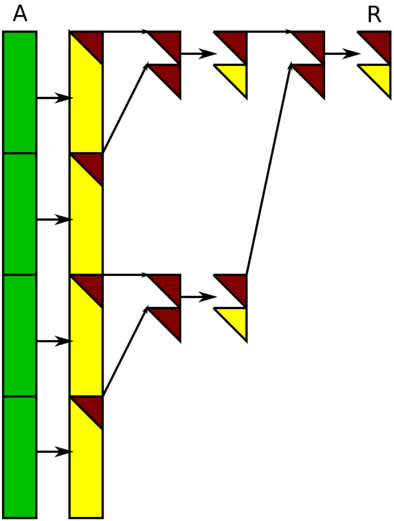


Communication-avoiding QR (CAQR)

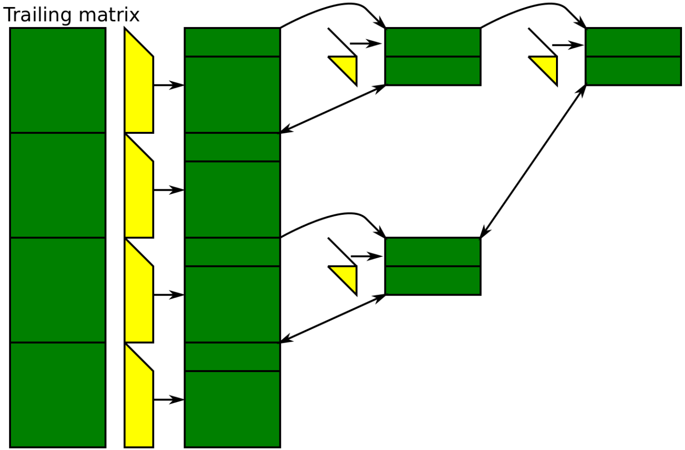
In paralle, CAQR reduces the latency cost over Householder QR. We outline the algorithm for a n-by-n matrix below,

- Distribute the matrix in a block-cyclic layout
- For each panel of width $b \ll n$ perform a Tall-Skinny-QR (TSQR)
- TSQR gives a tree representation which implictly represents the orthogonal matrix
- Apply the tree represenation obtained from TSQR to the orthognal matrix

Tall-skinny QR factorization



CAQR trailing matrix update



Communication costs of CAQR

Assuming square matrices and binary tree TSQR, apply-Q^T, CAQR has the costs

$$\gamma \cdot \left(\frac{2mn^2 - 2n^3/3}{p} \right) + \beta \cdot \left(\frac{2mn + n^2 \log p}{\sqrt{p}} \right) + \alpha \cdot \left(\frac{7}{2} \sqrt{p} \log^3 p \right).$$

- Major latency improvement over Householder QR (where it was $O(n \log p)$)
- Less efficient in terms of bandwidth due to term $O(n^2 \log p / \sqrt{p})$
- Computational overheads involved in apply-Q^T necessitate $\log^3 p$ latency cost factor

Basis kernel representations

In 1996, Sun and Bischof detailed many “basis-kernel” representations of an orthogonal matrix

$$Q = I - YTY^T = I - \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} T \begin{bmatrix} Y_1^T & Y_2^T \end{bmatrix}. \qquad (1)$$

- Y is refferred to as the “basis” and is not necessarily triangular
- T is called the “kernel” and cn also have any structure

Losing the triangular structure restrictions allows the definition of new representations.

Yamamoto's basis kernel representation

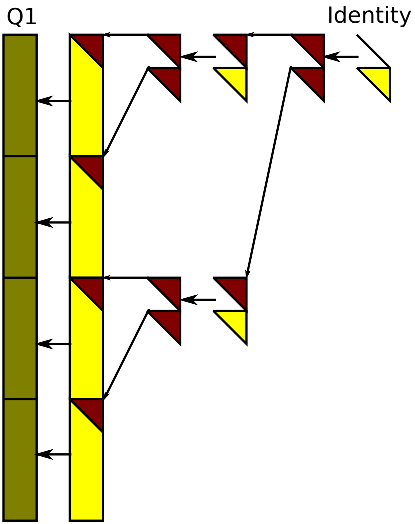
At SIAM ALA 2013, Yusaku Yamomoto presented a method to construct a basis kernel representation from the implicit TSQR representation

- Having computed a TSQR of size n-by-b construct the first b columns of Q, $[Q_1; Q_2]$ explicitly
- Now construct a basis-kernel representation as follows

$$Q = I - \tilde{Y}\tilde{T}\tilde{Y}^T = I - \begin{bmatrix} Q_1 - I \\ Q_2 \end{bmatrix} \left[I - Q_1\right]^{-T} \begin{bmatrix} (Q_1 - I)^T & Q_2^T \end{bmatrix}$$

  where $I - Q_1$ is assumed to be nonsingular.
- The assumption that $I - Q_1$ is nonsingular can be dropped by replacing I with a diagonal sign matrix S chosen so that $S - Q_1$ is nonsingular

Forming the first columns of the orthogonal matrix



Reconstructing the Householder vectors

Yamamoto's basis-kernel representation does not have the same structure as Householder vectors, which we want for software engineering reasons. Our method builds on Yamamoto's to obtain this representation,

- Form the first b columns of the orthogonal matrix, $[Q_1; Q_2]$ as done by Yamamoto
- Compute the LU factorization of $[Q_1 - S; Q_2]$ picking elements of S to be the opposite sign of the diagonal entry of the next column of the trailing matrix
- The L factor from the LU factorization will be the Householder vectors Y!
- The U factor from the LU factorization will be $-TY_1^T S$!

Uniqueness in exact arithmetic

Lemma

Given an orthonormal $m \times b$ matrix Q, let the compact QR decomposition of Q given by the CAQR-HR algorithm be

$$Q = \left(\begin{bmatrix} I_n \\ 0 \end{bmatrix} - YTY_1^T \right) S,$$

where Y is unit lower triangular, Y_1 is the top $b \times b$ block of Y, and T is the upper triangular $b \times b$ matrix satisfying $T^{-1} + T^T = Y^T Y$. Then S is a diagonal sign matrix, and $Q - \begin{bmatrix} S \\ 0 \end{bmatrix}$ has a unique LU decomposition given by

$$Q - \begin{bmatrix} S \\ 0 \end{bmatrix} = Y \cdot (-TY_1^T S). \tag{2}$$

Stability proof for Householder reconstruction

Proven by Grey Ballard et al.

Lemma

In floating point arithmetic, given an orthonormal $m \times b$ matrix Q, CAQR-HR computes factors S, $\tilde{Y}$, and $\tilde{T}$ such that

$$\left\| QS - \left(\begin{bmatrix} I \\ 0 \end{bmatrix} - \tilde{Y}\tilde{T}\tilde{Y}_1^T \right) \right\|_F \leq f_3(b, \varepsilon).$$

where

$$f_3(b, \varepsilon) = 2\gamma_b \left(b^2 + \left(1 + \sqrt{2} \right) b \right)$$

and Y_1 is given by the first b rows of Y.

Tall-skinny numerical stability experiments

Results collected by Mathias Jacquelin

ρ	κ	$\|A - QR\|_2$	$\|I - Q^T Q\|_2$
1e-01	5.1e+02	2.2e-15	9.3e-15
1e-03	5.0e+04	2.2e-15	8.4e-15
1e-05	5.1e+06	2.3e-15	8.7e-15
1e-07	5.0e+08	2.4e-15	1.1e-14
1e-09	5.0e+10	2.3e-15	9.9e-15
1e-11	4.9e+12	2.5e-15	1.0e-14
1e-13	5.0e+14	2.2e-15	8.8e-15
1e-15	5.0e+15	2.4e-15	9.7e-15

Error of CAQR-HR on tall and skinny matrices ($m = 1000, b = 200$)

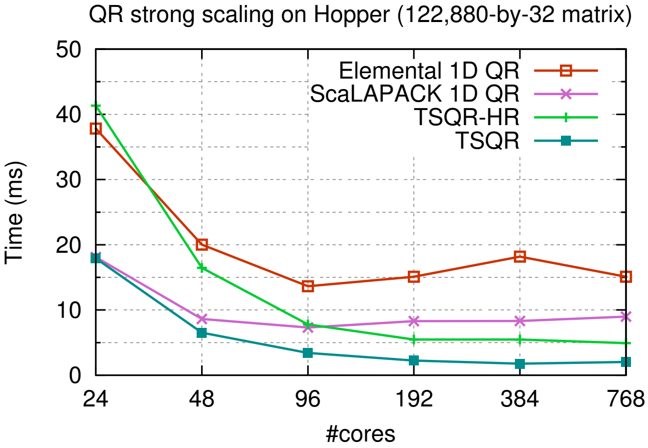
Square numerical stability experiments

Results collected by Mathias Jacquelin

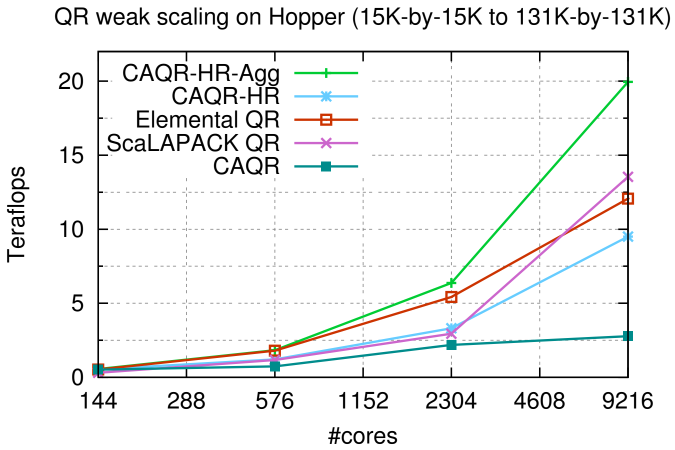
Matrix type	κ	$\|A - QR\|_2$	$\|I - Q^T Q\|_2$
$A = 2 * rand(m) - 1$	2.1e+03	4.3e-15 (256)	2.8e-14 (2)
Golub-Klema-Stewart	2.2e+20	0.0e+00 (2)	0.0e+00 (2)
Break 1 distribution	1.0e+09	1.0e-14 (256)	2.8e-14 (2)
Break 9 distribution	1.0e+09	9.9e-15 (256)	2.9e-14 (2)
$U \Sigma V^T$ with exponential distribution	4.2e+19	2.0e-15 (256)	2.8e-14 (2)
The devil's stairs matrix	2.3e+19	2.4e-15 (256)	2.8e-14 (2)
KAHAN matrix, a trapezoidal matrix	5.6e+56	0.0e+00 (2)	0.0e+00 (2)
Matrix ARC130 from Matrix Market	6.0e+10	8.8e-19 (16)	2.1e-15 (2)
Matrix FS_541_1 from Matrix Market	4.5e+03	5.8e-16 (64)	1.8e-15 (256)
DERIV2: second derivative	1.2e+06	2.8e-15 (256)	4.6e-14 (2)
FOXGOOD: severely ill-posed problem	5.7e+20	2.4e-15 (256)	2.8e-14 (2)

Errors of CAQR-HR on square matrices ($m = 1000$). The numbers in parentheses give the panel width yielding largest error.

Tall-skinny QR performance on Cray XE6



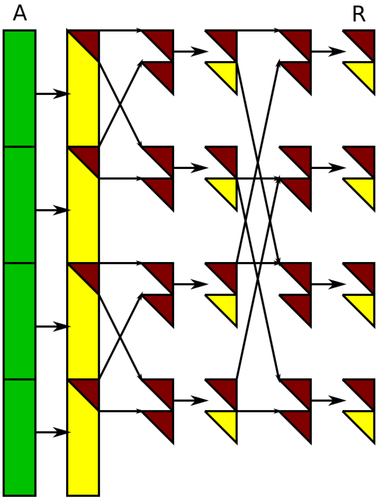
Square matrix QR performance on Cray XE6



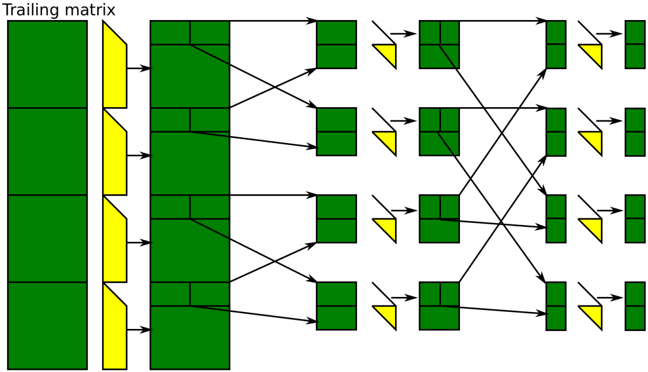
Repeating the mistake

- The original CAQR paper made the mistake of mislabeling the performance of Householder QR, due to being unaware of linear-scaling collective algorithms.
- We made the mistake of mislabeling the performance of CAQR by not considering more efficient trees for the apply-Q^T stage
- But as far as we know, everyone else made the same mistake

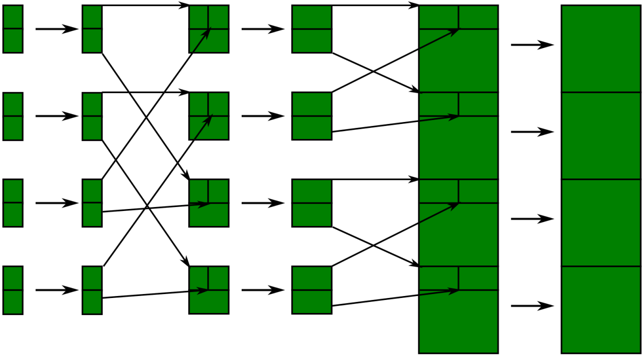
Butterfly TSQR



Butterfly TSQR apply Q^T: recursive halving stage



Butterfly TSQR apply Q^T: recursive doubling stage

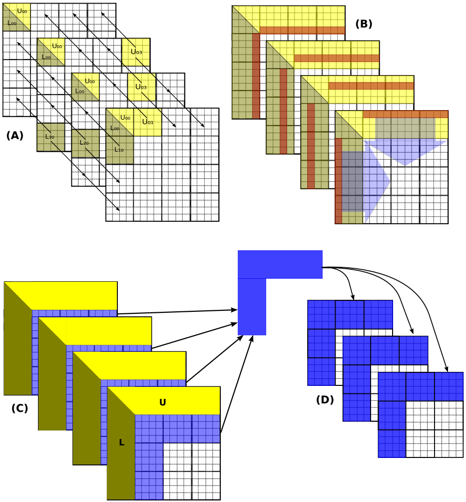


2.5D recursive LU

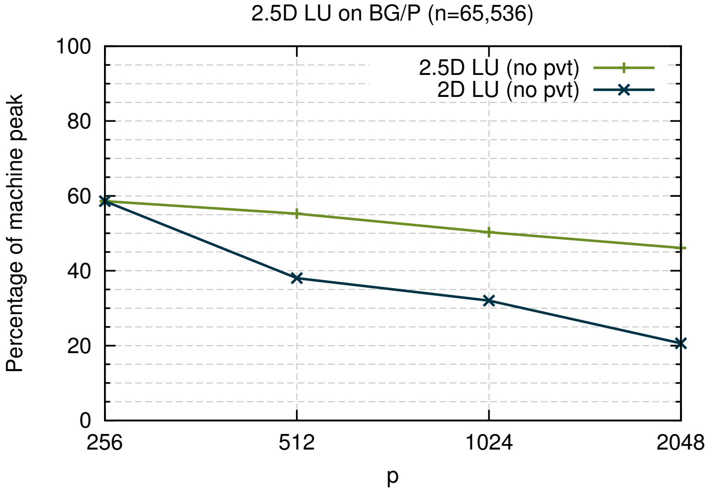
$A = L \cdot U$ where L is lower-triangular and U is upper-triangular

- A 2.5D recursive algorithm with no pivoting [A. Tiskin 2002]
- Tiskin gives algorithm under the BSP model
  - Bulk Synchronous Parallel
  - considers communication and synchronization
- We give an alternative distributed-memory adaptation and implementation
- Also, we lower-bound the latency cost

2.5D LU factorization



2.5D LU strong scaling (without pivoting)



2.5D LU with pivoting

$A = P \cdot L \cdot U$, where P is a permutation matrix

- 2.5D generic pairwise elimination (neighbor/pairwise pivoting or Givens rotations (QR)) [A. Tiskin 2007]
  - pairwise pivoting does not produce an explicit L
  - pairwise pivoting may have stability issues for large matrices
- Our approach uses tournament pivoting, which is more stable than pairwise pivoting and gives L explicitly
  - pass up rows of A instead of U to avoid error accumulation

Tournament pivoting

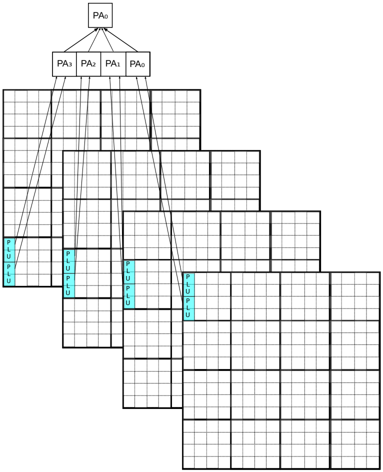
Partial pivoting is not communication-optimal on a blocked matrix

- requires message/synchronization for each column
- $O(n)$ messages needed

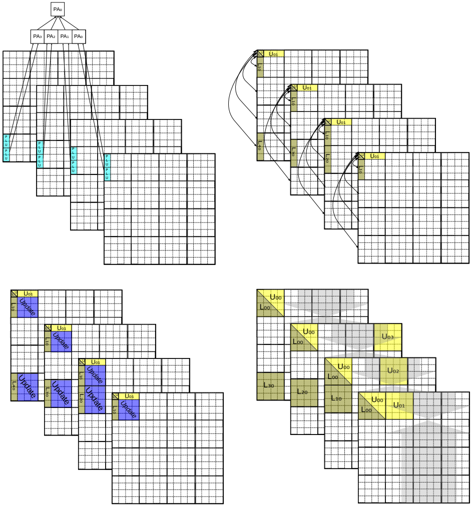
Tournament pivoting is communication-optimal

- performs a tournament to determine best pivot row candidates
- passes up 'best rows' of A

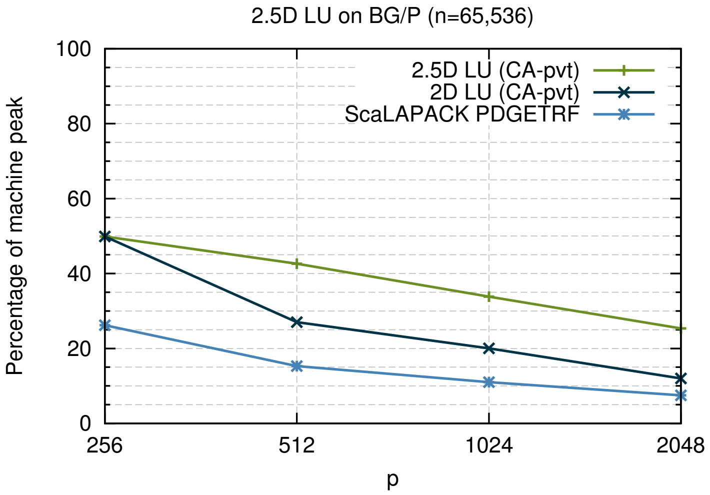
2.5D LU factorization with tournament pivoting



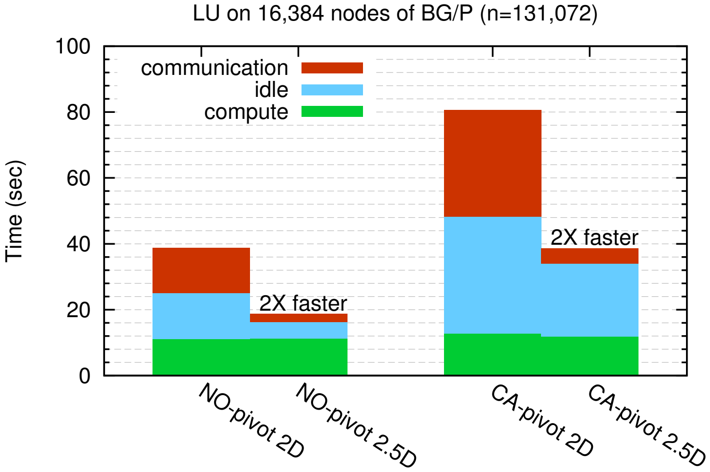
2.5D LU factorization with tournament pivoting



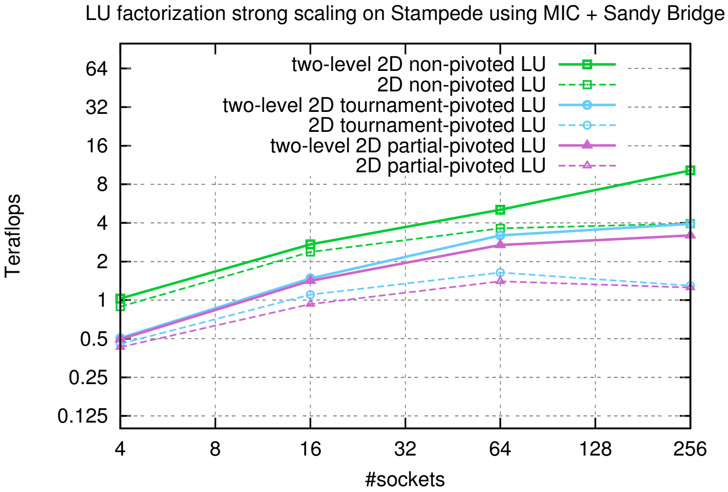
Strong scaling of 2.5D LU with tournament pivoting



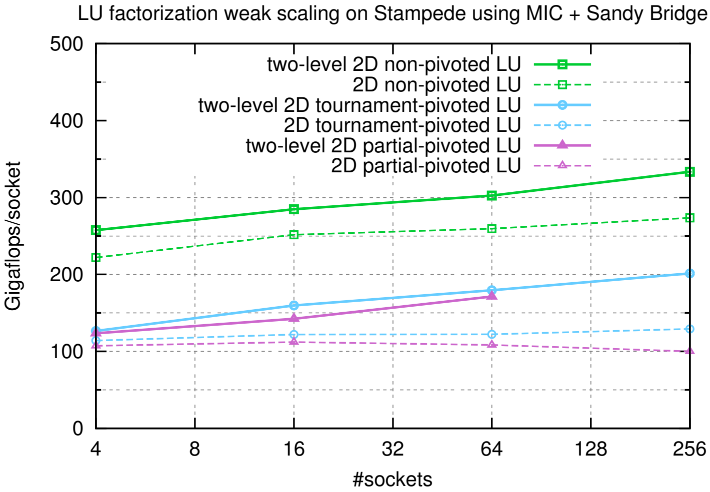
2.5D LU on 65,536 cores



2.5D LU on hybrid architectures



2.5D LU on hybrid architectures



Conclusions

- Know your collective communication algorithms!
- ScaLAPACK should have two block sizes
- Avoid communication

Future work: symmetric eigensolve

Backup slides