Communication-Avoiding LU factorization with Panel Rank Revealing Pivoting

Amal Khabou

Université Paris Sud 11, INRIA Saclay France

September 6, 2011

Collaborators

- Jim Demmel, UC Berkeley
- Laura Grigori, INRIA
- Ming Gu, UC Berkeley

Plan

- Motivation : just work, less talk

Plan

- Motivation : just work, less talk
- Communication Avoiding LU (CALU)

Plan

- Motivation : just work, less talk
- Communication Avoiding LU (CALU)
- LU with Panel Rank Revealing Pivoting (LU_PRRP)

Plan

- Motivation : just work, less talk
- Communication Avoiding LU (CALU)
- LU with Panel Rank Revealing Pivoting (LU_PRRP)
- Communication Avoiding LU_PRRP (CALU_PRRP)

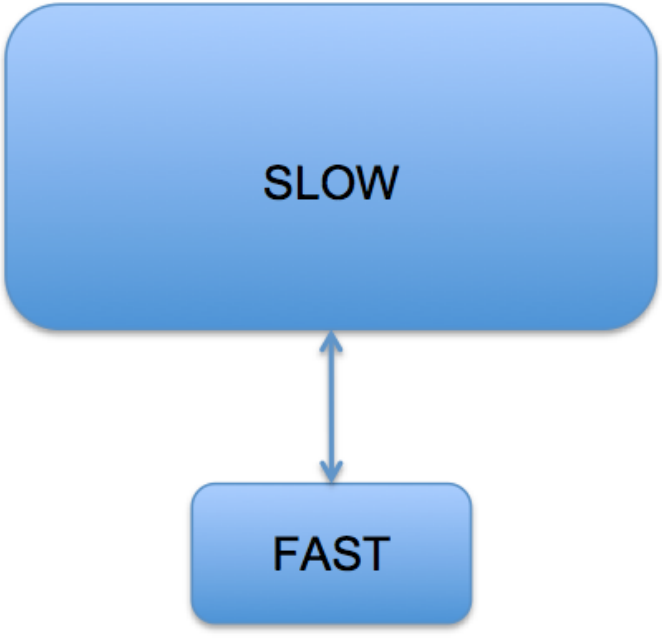
Plan

- Motivation : just work, less talk
- Communication Avoiding LU (CALU)
- LU with Panel Rank Revealing Pivoting (LU_PRRP)
- Communication Avoiding LU_PRRP (CALU_PRRP)
- Conclusions and future work

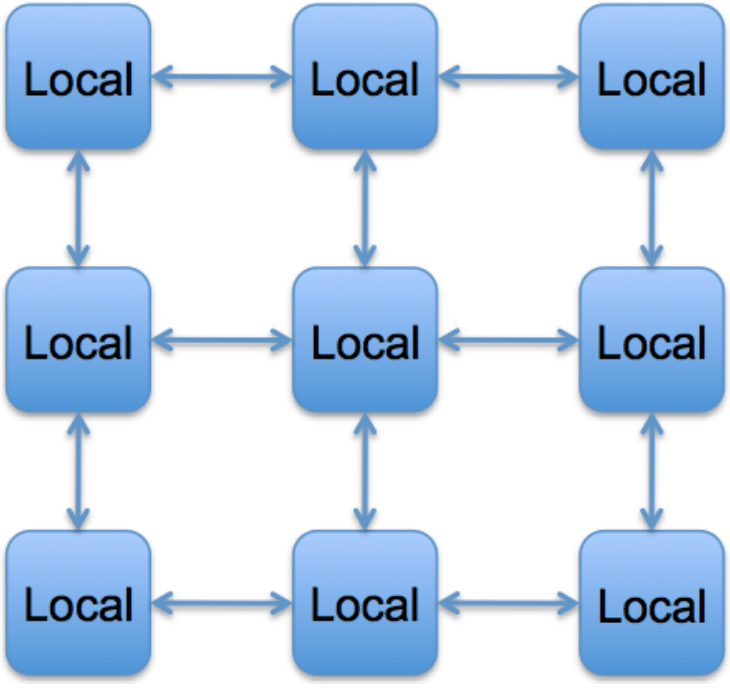
Memory Model

By *communication* we mean

- moving data within memory hierarchy on a sequential computer
- moving data between processors on a parallel computer



Sequential



Parallel

Communication Cost Model

Measure communication in terms of *messages* and *words*

- Flop cost: γ
- Cost of message of size *w* words: $\alpha + \beta w$
- Total running time of an algorithm (ignoring overlap):

$$\alpha \cdot (\text{\# messages}) + \beta \cdot (\text{\# words}) + \gamma \cdot (\text{\# flops})$$

- think of α as latency+overhead cost, β as inverse bandwidth

As flop rates continue to improve more quickly than data transfer rates, the relative cost of communication (the first two terms) grows larger

Lower bound for all direct linear algebra

Let M = local/ fast memory size, general lower bounds:
[Ballard et al., 2011]

$$\#\text{ words moved} = \Omega\left(\frac{\#\text{ flops}}{\sqrt{M}}\right)$$

$$\#\text{ messages} = \Omega\left(\frac{\#\text{ flops}}{M^{\frac{3}{2}}}\right)$$

Lower bound for all direct linear algebra

Let M = local/ fast memory size, general lower bounds:
[Ballard et al., 2011]

$$\#\text{ words moved} = \Omega\left(\frac{\#\text{ flops}}{\sqrt{M}}\right)$$

$$\#\text{ messages} = \Omega\left(\frac{\#\text{ flops}}{M^{\frac{3}{2}}}\right)$$

Goal : reorganize linear algebra to :

- minimize # words moved
- minimize # messages
- conserve numerical stability (better: to improve)
- don't increase the flop count (too much)

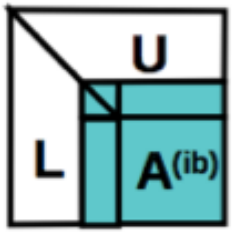
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a $P = P_r \times P_c$ grid of processors
For $ib = 1$ to $n - 1$step b $A^{(ib)} = A(ib : n, ib : n)$

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

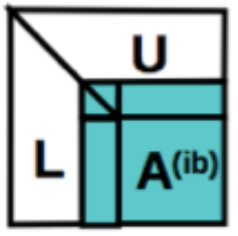
1. Compute panel factorization
   - find pivot in each column, swap rows.



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

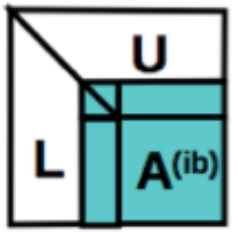
1. Compute panel factorization # messages=$O(n \log_2 P_r)$
   - find pivot in each column, swap rows.



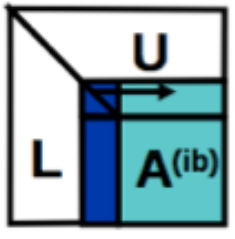
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a $P = P_r \times P_c$ grid of processors
For $ib = 1$ to $n - 1$step b $A^{(ib)} = A(ib : n, ib : n)$

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



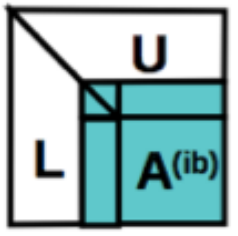
2. Apply all row permutations
   - broadcast pivot information along the rows
   - swap rows at left and right.



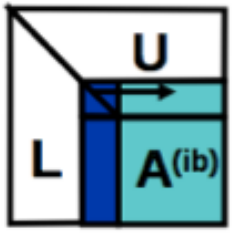
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a $P = P_r \times P_c$ grid of processors
For $ib = 1$ to $n - 1$step b $A^{(ib)} = A(ib : n, ib : n)$

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



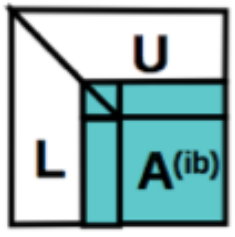
2. Apply all row permutations # messages=$O(n/b(\log_2 P_r + \log_2 P_c))$
   - broadcast pivot information along the rows
   - swap rows at left and right.



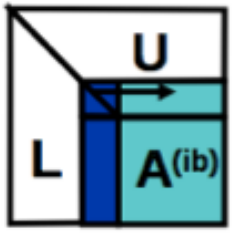
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a $P = P_r \times P_c$ grid of processors
For $ib = 1$ to $n - 1$step b $A^{(ib)} = A(ib : n, ib : n)$

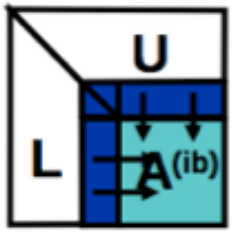
1. Compute panel factorization # messages=$O(n \log_2 P_r)$
   - find pivot in each column, swap rows.



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



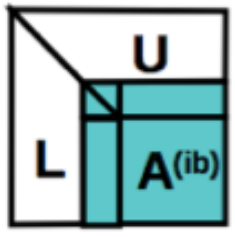
3. Compute block row of U
   - broadcast right diagonal block of L of current panel.



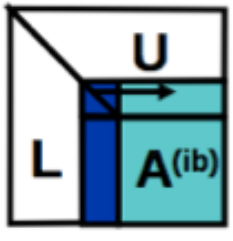
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a $P = P_r \times P_c$ grid of processors
For $ib = 1$ to $n - 1$step b $A^{(ib)} = A(ib : n, ib : n)$

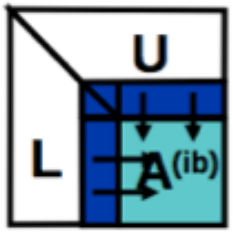
1. Compute panel factorization # messages=$O(n \log_2 P_r)$
   - find pivot in each column, swap rows.



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



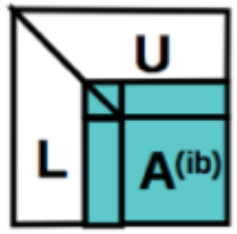
3. Compute block row of U # messages=$O(n/b \log_2 P_c)$
   - broadcast right diagonal block of L of current panel.



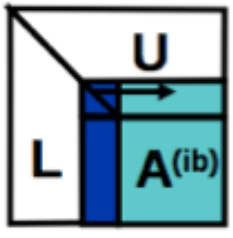
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a $P = P_r \times P_c$ grid of processors
For $ib = 1$ to $n - 1$step b $A^{(ib)} = A(ib : n, ib : n)$

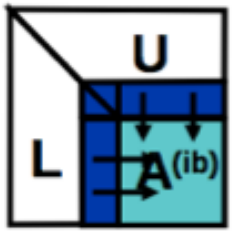
1. Compute panel factorization # messages=$O(n \log_2 P_r)$
   - find pivot in each column, swap rows.



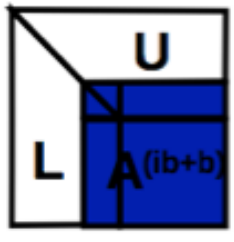
2. Apply all row permutations # messages=$O(n/b(\log_2 P_r + \log_2 P_c))$
   - broadcast pivot information along the rows
   - swap rows at left and right.



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



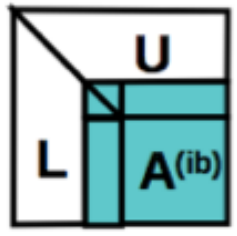
4. Update trailing matrix
   - broadcast right block column of L.
   - broadcast down block row of U.



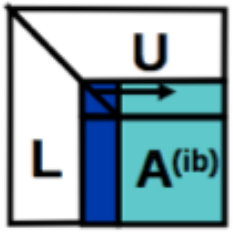
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a $P = P_r \times P_c$ grid of processors
For $ib = 1$ to $n - 1$step b $A^{(ib)} = A(ib : n, ib : n)$

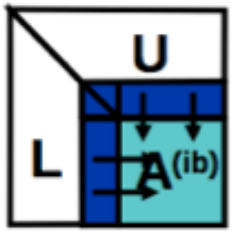
1. Compute panel factorization # messages=$O(n \log_2 P_r)$
   - find pivot in each column, swap rows.



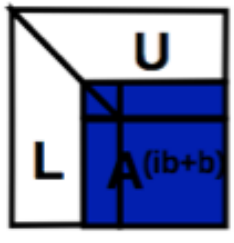
2. Apply all row permutations # messages=$O(n/b(\log_2 P_r + \log_2 P_c))$
   - broadcast pivot information along the rows
   - swap rows at left and right.



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



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



LU factorization (ScaLAPACK) upper bounds

2D algorithm $P = P_r \times P_c$, M = $O\left(\frac{n^2}{P}\right)$, Lower bounds :
[Ballard et al., 2011]

$$\#\text{ words moved} = \Omega\left(\frac{n^2}{\sqrt{P}}\right)$$

$$\#\text{ messages} = \Omega\left(\sqrt{P}\right)$$

LU factorization (ScaLAPACK) upper bounds

2D algorithm $P = P_r \times P_c$, M = $O\left(\frac{n^2}{P}\right)$, Lower bounds : [Ballard et al., 2011]

$$\#\text{ words moved} = \Omega\left(\frac{n^2}{\sqrt{P}}\right) \qquad \#\text{ messages} = \Omega\left(\sqrt{P}\right)$$

LU with partial pivoting (*ScaLAPACK*) $P = \sqrt{P} \times \sqrt{P}$, b = $\frac{n}{\sqrt{P}}$ (upper bound) :

Factor exceeding lower bounds for #words moved	Factor exceeding lower bounds for #messages
$\log P$	$\left(\frac{n}{\sqrt{P}}\right) \log P$

LU factorization (ScaLAPACK) upper bounds

2D algorithm $P = P_r \times P_c$, M = $O\left(\frac{n^2}{P}\right)$, Lower bounds : [Ballard et al., 2011]

$$\#\text{ words moved} = \Omega\left(\frac{n^2}{\sqrt{P}}\right) \qquad \#\text{ messages} = \Omega\left(\sqrt{P}\right)$$

LU with partial pivoting (*ScaLAPACK*) $P = \sqrt{P} \times \sqrt{P}$, b = $\frac{n}{\sqrt{P}}$ (upper bound) :

Factor exceeding lower bounds for #words moved	Factor exceeding lower bounds for #messages
log P	$\left(\frac{n}{\sqrt{P}}\right)$ log P

The lower bounds are attained by CALU : LU factorization with tournament pivoting [Grigori et al., 2010]

Performance Models of Parallel CALU vs PDGETRF

For $n \times n$ matrix, with an optimal layout for CALU :

$$P_r = \sqrt{P} \ , \ P_c = \sqrt{P} \ \text{ and } b = \log^{-2}\left(\sqrt{P}\right) \cdot \sqrt{\frac{n^2}{P}},$$

Performance Models of Parallel CALU vs PDGETRF

For $n \times n$ matrix, with an optimal layout for CALU :

$$P_r = \sqrt{P} \ , \ P_c = \sqrt{P} \ \text{ and } b = \log^{-2}\left(\sqrt{P}\right) \cdot \sqrt{\frac{n^2}{P}},$$

	Parallel CALU	Lower bound
# messages	$3\sqrt{P}\log^3 P$	$\Omega(\sqrt{P})$
# words	$\frac{n^2}{\sqrt{P}}\left(2\log^{-1} P + 1.25\log P\right)$	$\Omega(\frac{n^2}{\sqrt{P}})$
# flops	$\frac{1}{P}\frac{2n^3}{3} + \frac{5n^3}{2P\log^2 P} + \frac{5n^3}{3P\log^3 P}$	$\frac{1}{P}\left(\frac{2n^3}{3}\right)$

Performance Models of Parallel CALU vs PDGETRF

For $n \times n$ matrix, with an optimal layout for CALU :

$$P_r = \sqrt{P} \ , \ P_c = \sqrt{P} \ \text{ and } b = \log^{-2}\left(\sqrt{P}\right) \cdot \sqrt{\frac{n^2}{P}},$$

	Parallel CALU	Lower bound
# messages	$3\sqrt{P}\log^3 P$	$\Omega(\sqrt{P})$
# words	$\frac{n^2}{\sqrt{P}}\left(2\log^{-1} P + 1.25\log P\right)$	$\Omega(\frac{n^2}{\sqrt{P}})$
# flops	$\frac{1}{P}\frac{2n^3}{3} + \frac{5n^3}{2P\log^2 P} + \frac{5n^3}{3P\log^3 P}$	$\frac{1}{P}\left(\frac{2n^3}{3}\right)$

	PDGETRF	Lower bound
# messages	$n\log P + 7\sqrt{P}\log^3 P$	$\Omega(\sqrt{P})$
# words	$\frac{n^2}{\sqrt{P}}\left(\log^{-1} P + \log P\right)$	$\Omega(\frac{n^2}{\sqrt{P}})$
# flops	$\frac{1}{P}\frac{2n^3}{3} + \frac{2n^3}{P\log^2 P}$	$\frac{1}{P}\left(\frac{2n^3}{3}\right)$

Stability of the LU factorization (1/2)

Consider the growth factor [Wilkinson, 1961]

$$g_W = \frac{max_{i,j,k}|a_{i,j}^{(k)}|}{max_{i,j}|a_{i,j}|}$$

where $a_{i,j}^{(k)}$ denotes the entry in position (i, j) obtained after k steps of elimination.

Stability of the LU factorization (1/2)

Consider the growth factor [Wilkinson, 1961]

$$g_W = \frac{max_{i,j,k}|a_{i,j}^{(k)}|}{max_{i,j}|a_{i,j}|}$$

where $a_{i,j}^{(k)}$ denotes the entry in position (i, j) obtained after k steps of elimination.
Worst case pivot growth :

- Partial pivoting : $g_W \leq 2^{n-1}$
- Complete pivoting :
  $g_W \leq n^{1/2}(2.3^{1/2}...n^{1/(n-1)})^{1/2} \sim cn^{1/2}n^{1/4\log_2 n}$

Stability of the LU factorization (1/2)

Consider the growth factor [Wilkinson, 1961]

$$g_W = \frac{max_{i,j,k}|a_{i,j}^{(k)}|}{max_{i,j}|a_{i,j}|}$$

where $a_{i,j}^{(k)}$ denotes the entry in position (i, j) obtained after k steps of elimination.
Worst case pivot growth :

- Partial pivoting : $g_W \leq 2^{n-1}$
- Complete pivoting :
  $g_W \leq n^{1/2}(2.3^{1/2}...n^{1/(n-1)})^{1/2} \sim cn^{1/2}n^{1/4\log_2 n}$

- For partial pivoting, the upper bound is attained for the Wilkinson matrix.

Stability of the LU factorization (2/2)

$$g_T = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\sigma_A}$$

a statistical model for the average growth factor where, σ_A is the standard deviation of the initial element distribution of A, introduced by [Trefethen and Schreiber, 1990].

Stability of the LU factorization (2/2)

$$g_T = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\sigma_A}$$

a statistical model for the average growth factor where, σ_A is the standard deviation of the initial element distribution of A, introduced by [Trefethen and Schreiber, 1990].
Experiments performed for various distribution of matrices with n < 1024

Stability of the LU factorization (2/2)

$$g_T = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\sigma_A}$$

a statistical model for the average growth factor where, σ_A is the standard deviation of the initial element distribution of A, introduced by [Trefethen and Schreiber, 1990].
Experiments performed for various distribution of matrices with n < 1024
Average case pivot growth :

- Partial pivoting : $g_T \sim n^{2/3}$
- Complete pivoting : $g_T \sim n^{1/2}$

TSLU : Tournament pivoting - the overall idea

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

TSLU : Tournament pivoting - the overall idea

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W

TSLU : Tournament pivoting - the overall idea

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - Find at low communication cost good pivots for the LU factorization of the panel W, return a permutation matrix Π (preprocessing step).

TSLU : Tournament pivoting - the overall idea

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - Find at low communication cost good pivots for the LU factorization of the panel W, return a permutation matrix Π (preprocessing step).
  - Apply Π to the input matrix A.

TSLU : Tournament pivoting - the overall idea

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - Find at low communication cost good pivots for the LU factorization of the panel W, return a permutation matrix Π (preprocessing step).
  - Apply Π to the input matrix A.
  - Compute LU with no pivoting of ΠW, update trailing matrix.

TSLU : Tournament pivoting - the overall idea

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \text{ where, } \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - Find at low communication cost good pivots for the LU factorization of the panel W, return a permutation matrix Π (preprocessing step).
  - Apply Π to the input matrix A.
  - Compute LU with no pivoting of ΠW, update trailing matrix.
- benefit: b times fewer messages overall for the panel factorization - faster

Stability of CALU

Worst case analysis of pivot growth factor for a reduction tree of height H : CALU vs GEPP [Grigori et al., 2010]

	matrix of size $m \times (b+1)$	
	TSLU(b,H)	GEPP
g_W upper bound	$2^{b(H+1)}$	2^b
	matrix of size $m \times n$	
	CALU	GEPP
g_W upper bound	$2^{n(H+1)-1}$	2^{n-1}

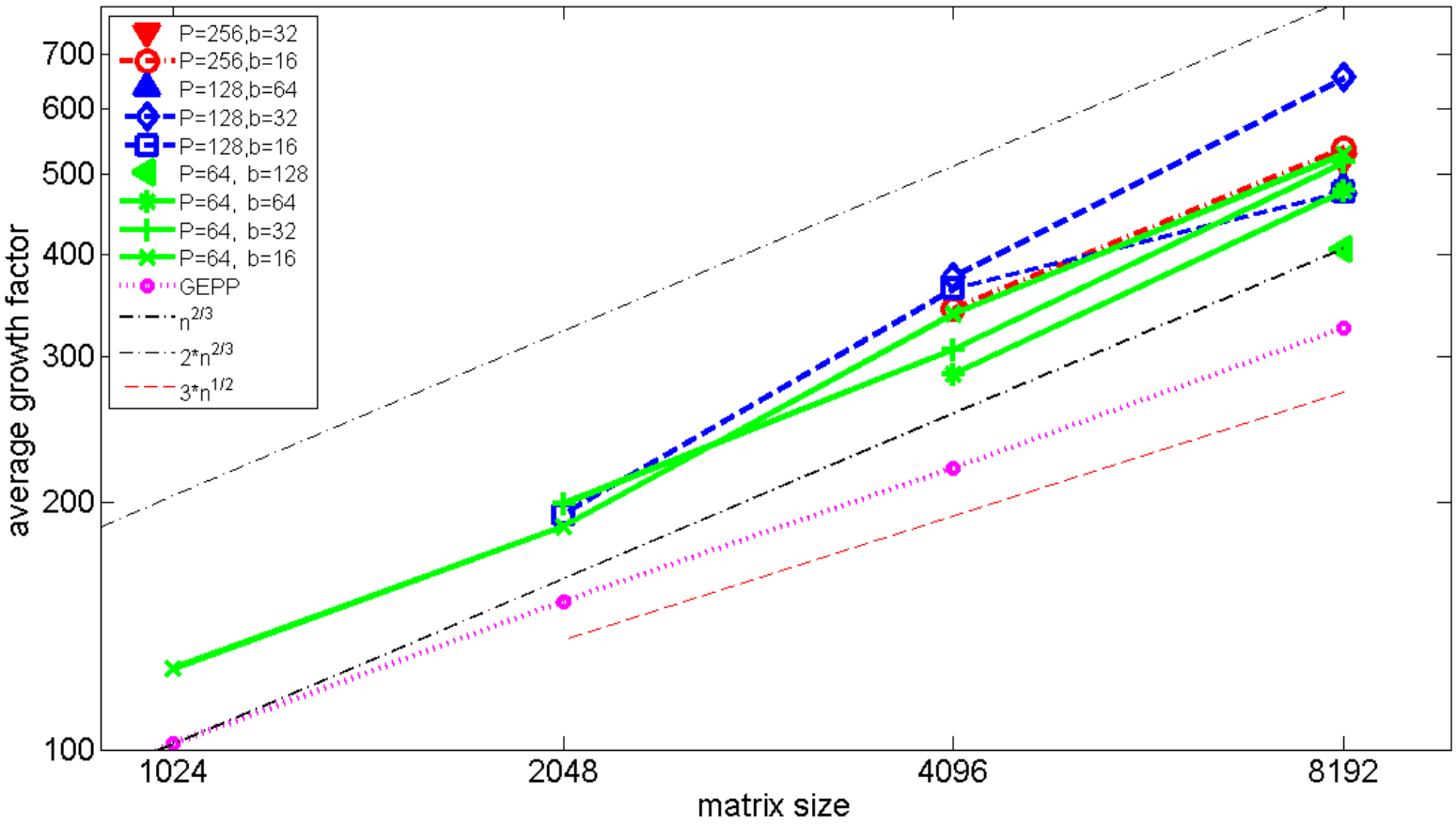
Stability of CALU

Worst case analysis of pivot growth factor for a reduction tree of height H : CALU vs GEPP [Grigori et al., 2010]

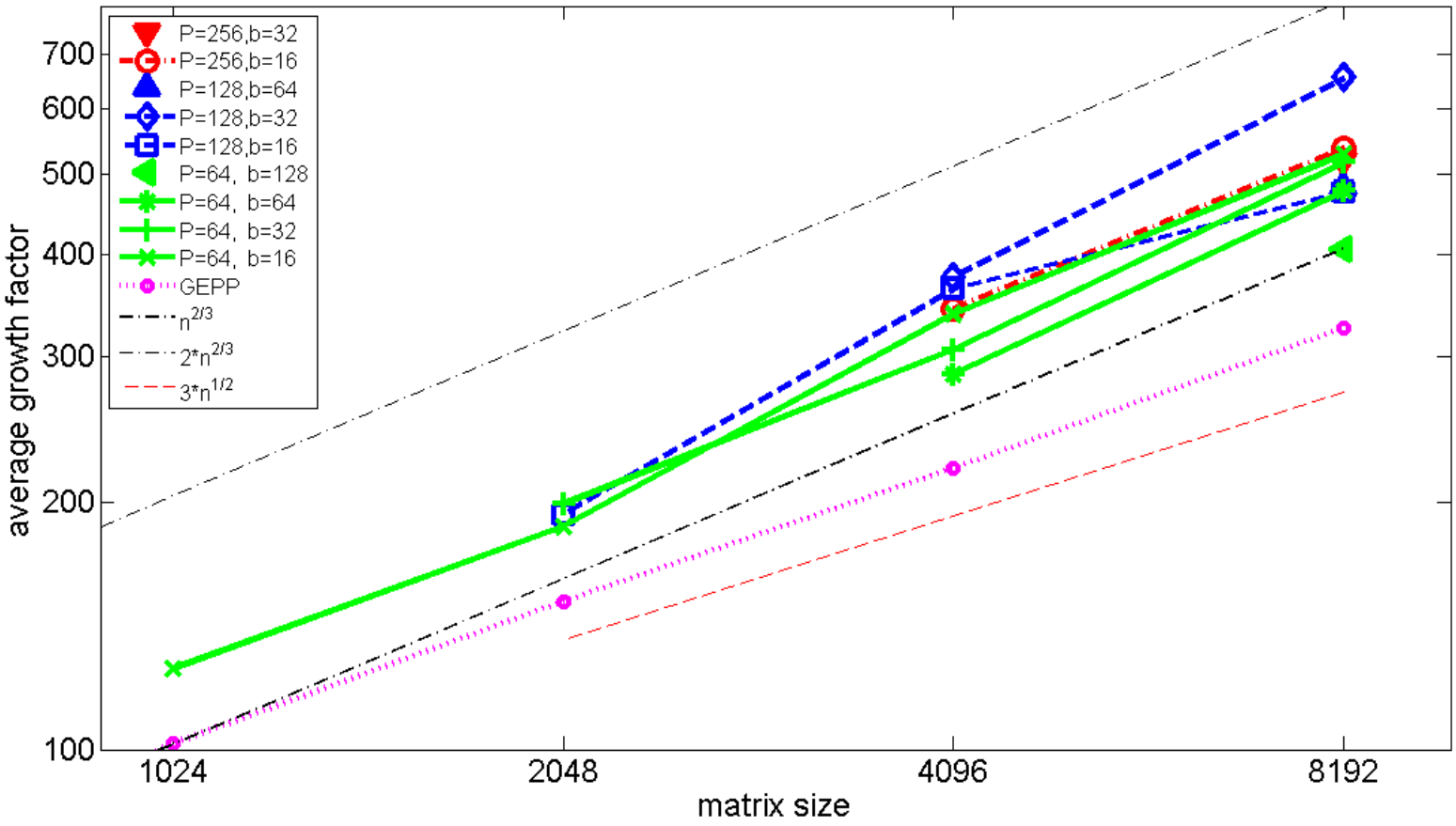
	matrix of size $m \times (b+1)$	
	TSLU(b,H)	GEPP
g_W upper bound	$2^{b(H+1)}$	2^b
	matrix of size $m \times n$	
	CALU	GEPP
g_W upper bound	$2^{n(H+1)-1}$	2^{n-1}

- The upper bound of CALU is worse than GEPP.
- Upper bound for $|L|$ is 2^{bH}, attained $2^{(b-2)H-b+1}$, but CALU is still stable in practice.
- There are Wlikinson-like matrices where GEPP is stable but CALU gives exponential growth, and vice-versa.

Growth factor g_T for binary tree based CALU



Growth factor g_T for binary tree based CALU



- Random matrices from a normal distribution.
- The growth factor g_T grows as $\sim (1.5)n^{2/3}$.

Motivation

- CALU is communication optimal.
- CALU is as stable as GEPP in practice.
- CALU is worse than GEPP in terms of pivot growth.

- Our goal: to improve the stability of both GEPP and CALU.

Strong Rank Revealing QR

To improve the stability of LU:

- Pivoting strategy based on the Strong RRQR factorization

$$A^T \Pi = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}$$

Strong Rank Revealing QR

To improve the stability of LU:

- Pivoting strategy based on the Strong RRQR factorization

$$A^T \Pi = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}$$

This factorization satisfies three conditions [Gu and Eisenstat, 1996]:

- every singular value of R_{11} is large.
- every singular value of R_{22} is small.
- every element of $R_{11}^{-1} R_{12}$ could be bounded by a given threshold τ.

Strong Rank Revealing QR

Result: $A^T \Pi = QR$ with $\|R_{11}^{-1} R_{12}\|_{max} \leq \tau$
Compute $A^T \Pi = QR$ RRQR with column pivoting ;
while *there exist i and j such that* $\left|\left[R_{11}^{-1} R_{12}\right]_{ij}\right| > \tau$ **do**
  Compute the QR factorization of $R\,\Pi_{ij}$ (QR updates) ;
end

Algorithm: Strong RRQR

Strong Rank Revealing QR

Result: $A^T\Pi = QR$ with $\|R_{11}^{-1}R_{12}\|_{max} \leq \tau$
Compute $A^T\Pi = QR$ RRQR with column pivoting ;
while *there exist i and j such that* $\left|\left[R_{11}^{-1}R_{12}\right]_{ij}\right| > \tau$ **do**
  Compute the QR factorization of $R\,\Pi_{ij}$ (QR updates) ;
end

Algorithm: Strong RRQR

- After each swap the determinant of R_{11} increases by at least τ.
- There are only a finite number of permutations.
- Strong RRQR does $O(mn^2)$ flops in the worst case.

LU_PRRP : LU with Panel Rank Revealing Pivoting

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

LU_PRRP : LU with Panel Rank Revealing Pivoting

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W

LU_PRRP : LU with Panel Rank Revealing Pivoting

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - perform the Strong RRQR factorization of the transpose of the panel W, find a permutation matrix Π

$$W^T \Pi = Q \begin{bmatrix} R_{11} & R_{12} \end{bmatrix} \text{ such as } D = \|R_{12}^T (R_{11}^{-1})^T\| \leq \tau$$

LU_PRRP : LU with Panel Rank Revealing Pivoting

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - perform the Strong RRQR factorization of the transpose of the panel W, find a permutation matrix Π

$$W^T \Pi = Q \begin{bmatrix} R_{11} & R_{12} \end{bmatrix} \text{ such as } D = \|R_{12}^T (R_{11}^{-1})^T\| \leq \tau$$

  - Apply Π^T to the input matrix A.

LU_PRRP : LU with Panel Rank Revealing Pivoting

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - perform the Strong RRQR factorization of the transpose of the panel W, find a permutation matrix Π

$$W^T \Pi = Q \begin{bmatrix} R_{11} & R_{12} \end{bmatrix} \text{ such as } D = \|R_{12}^T (R_{11}^{-1})^T\| \leq \tau$$

  - Apply Π^T to the input matrix A.
  - update trailing matrix as following.

$$\hat{A} = \Pi^T A = \begin{bmatrix} I_b & \\ D & I_{m-b} \end{bmatrix} \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ & \bar{\hat{A}}_{22} \end{bmatrix}$$

where

$$\bar{\hat{A}}_{22} = \hat{A}_{22} - D\hat{A}_{12}$$

LU_PRRP : LU with Panel Rank Revealing Pivoting

Result: Block LU factorization
for *j from 1 to* $\frac{n}{b}$ **do**
  Compute $A_j^T \Pi_j = Q_j R_j$ using Strong RRQR ;
  $D := (R_j(1:b, 1:b)^{-1} R_j(1:b, b+1:m-(j-1)b))^T$;
  Pivoting : apply the permutation matrix Π_j^T on the entire matrix A :
  $\hat{A} = \Pi_j^T A$;
  Update : $\hat{A}(jb+1:m, jb+1:n) :=$
  $\hat{A}(jb+1:m, jb+1:n) - D\hat{A}(j-1)b+1:jb, jb+1:n)$;
end

Algorithm: LU_PRRP

LU_PRRP : LU with Panel Rank Revealing Pivoting

Result: Block LU factorization
for j *from 1 to* $\frac{n}{b}$ **do**
  Compute $A_j^T \Pi_j = Q_j R_j$ using Strong RRQR ;
  $D := (R_j(1:b, 1:b)^{-1} R_j(1:b, b+1:m-(j-1)b))^T$;
  Pivoting : apply the permutation matrix Π_j^T on the entire matrix A :
  $\hat{A} = \Pi_j^T A$;
  Update : $\hat{A}(jb+1:m, jb+1:n) :=$
  $\hat{A}(jb+1:m, jb+1:n) - D\hat{A}(j-1)b+1:jb, jb+1:n)$;
end

Algorithm: LU_PRRP

- Block LU factorization, to get the full LU, GEPP is performed on the $b \times b$ diagonal block.
- The total cost is about $\frac{2}{3}n^3 + O(n^2 b)$ flops
- Is only $O(n^2 b)$ flops more than GEPP.

Stability of LU_PRRP (1/2)

Worst case analysis of pivot growth factor : LU_PRRP vs GEPP .

	matrix of size $m \times (b+1)$	
	LU_PRRP	GEPP
g_W upper bound	$1 + \tau b$	2^b
	matrix of size $m \times n$	
	LU_PRRP	GEPP
g_W upper bound	$(1 + \tau b)^{\frac{n}{b}}$	2^{n-1}

Stability of LU_PRRP (1/2)

Worst case analysis of pivot growth factor : LU_PRRP vs GEPP .

	matrix of size $m \times (b+1)$	
	LU_PRRP	GEPP
g_W upper bound	$1 + \tau b$	2^b
	matrix of size $m \times n$	
	LU_PRRP	GEPP
g_W upper bound	$(1 + \tau b)^{\frac{n}{b}}$	2^{n-1}

- LU_PRRP is more stable than GEPP.
- More resistant to pathological cases : Wilkinson, Foster[Foster, 1994].

Stability of LU_PRRP (2/2)

The upper bound for different panel sizes with a parameter $\tau = 2$

Stability of LU_PRRP (2/2)

The upper bound for different panel sizes with a parameter $\tau = 2$

b	g_W
8	$(1.425)^n$
16	$(1.244)^n$
32	$(1.139)^n$
64	$(1.078)^n$
128	$(1.044)^n$

Stability of LU_PRRP (2/2)

The upper bound for different panel sizes with a parameter $\tau = 2$

For the different panel sizes, LU_PRRP is more stable than GEPP.

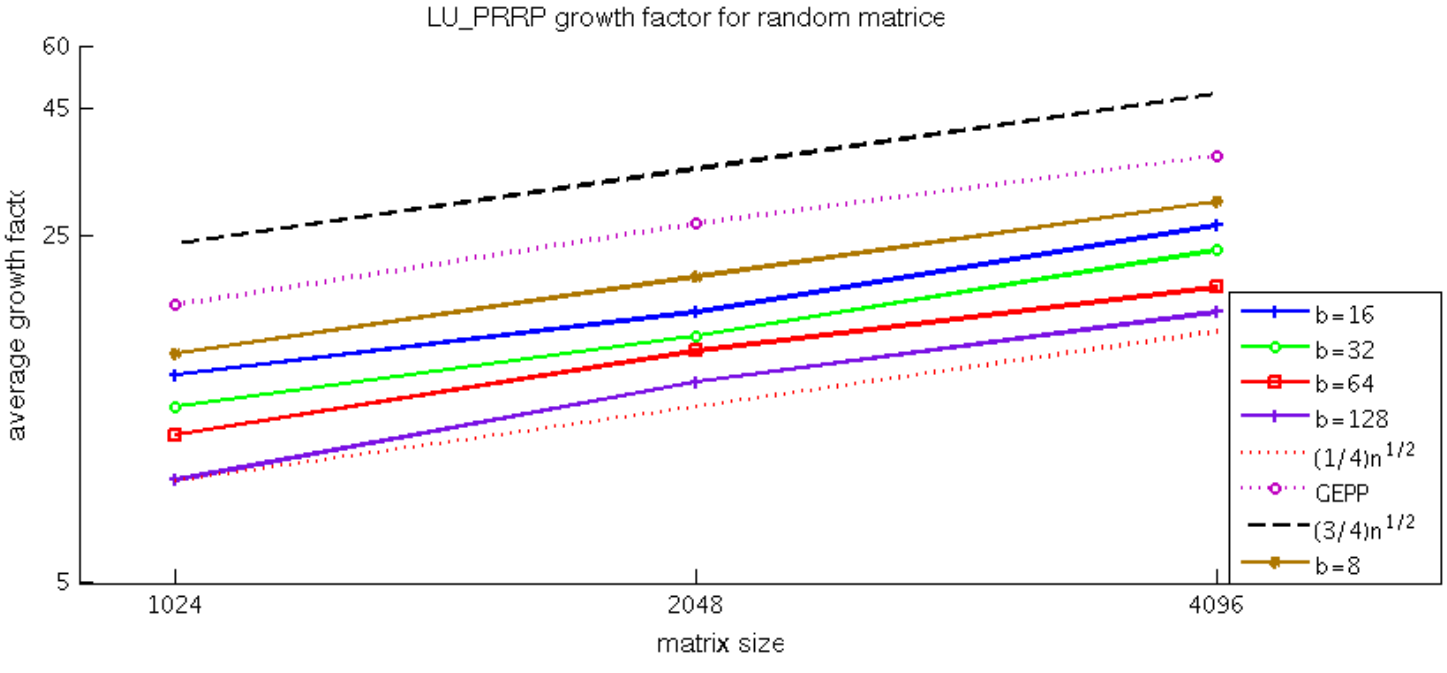
b	g_W
8	$(1.425)^n$
16	$(1.244)^n$
32	$(1.139)^n$
64	$(1.078)^n$
128	$(1.044)^n$

Stability of LU_PRRP (2/2)

The upper bound for different panel sizes with a parameter $\tau = 2$

For the different panel sizes, LU_PRRP is more stable than GEPP.

b	g_W
8	$(1.425)^n$
16	$(1.244)^n$
32	$(1.139)^n$
64	$(1.078)^n$
128	$(1.044)^n$



CALU_PRRP: Communication-Avoiding LU_PRRP

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

CALU_PRRP: Communication-Avoiding LU_PRRP

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W

CALU_PRRP: Communication-Avoiding LU_PRRP

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - Find at low communication cost b pivot rows for the QR factorization of the panel W, return a permutation matrix Π (preprocessing step).

CALU_PRRP: Communication-Avoiding LU_PRRP

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - Find at low communication cost b pivot rows for the QR factorization of the panel W, return a permutation matrix Π (preprocessing step).
  - Apply Π^T to the input matrix A.

CALU_PRRP: Communication-Avoiding LU_PRRP

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - Find at low communication cost b pivot rows for the QR factorization of the panel W, return a permutation matrix Π (preprocessing step).
  - Apply Π^T to the input matrix A.
  - Compute QR with no pivoting of $W^T\Pi$, update trailing matrix as for the LU_PRRP algorithm.

CALU_PRRP: Communication-Avoiding LU_PRRP

- Consider a block algorithm that factors an $n \times n$ matrix A by traversing panels of b columns.

$$A = \begin{bmatrix} \overbrace{A_{11}}^{b} & \overbrace{A_{12}}^{n-b} \\ A_{21} & A_{22} \end{bmatrix} \textit{where}, \quad W = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}$$

- For each panel W
  - Find at low communication cost b pivot rows for the QR factorization of the panel W, return a permutation matrix Π (preprocessing step).
  - Apply Π^T to the input matrix A.
  - Compute QR with no pivoting of $W^T\Pi$, update trailing matrix as for the LU_PRRP algorithm.
  - Perform GEPP on the $b \times b$ diagonal block (to obtain the full LU factorization).

CALU_PRRP: preprocessing step

- Consider the panel W, partitioned over $P = 4$ processors as

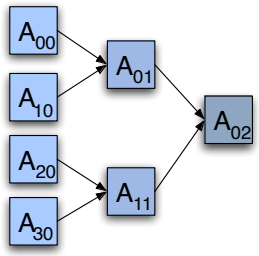
$$W = \begin{pmatrix} A_{00} \\ A_{10} \\ A_{20} \\ A_{30} \end{pmatrix},$$

CALU_PRRP: preprocessing step

- Consider the panel W, partitioned over $P = 4$ processors as

$$W = \begin{pmatrix} A_{00} \\ A_{10} \\ A_{20} \\ A_{30} \end{pmatrix},$$

- The panel factorization uses the following binary tree :

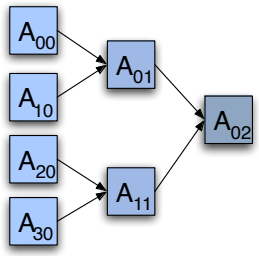


CALU_PRRP: preprocessing step

- Consider the panel W, partitioned over $P = 4$ processors as

$$W = \begin{pmatrix} A_{00} \\ A_{10} \\ A_{20} \\ A_{30} \end{pmatrix},$$

- The panel factorization uses the following binary tree :



1. Compute Strong RRQR factorization of the transpose of each block A_{i0} of the panel W, find a permutation Π_0

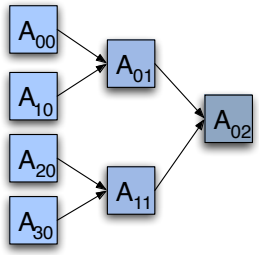
$$\begin{bmatrix} A_{00}^T \Pi_{00} \\ A_{10}^T \Pi_{10} \\ A_{20}^T \Pi_{20} \\ A_{30}^T \Pi_{30} \end{bmatrix} = \begin{bmatrix} Q_{00} R_{00} \\ Q_{10} R_{10} \\ Q_{20} R_{20} \\ Q_{30} R_{30} \end{bmatrix}$$

CALU_PRRP: preprocessing step

- Consider the panel W, partitioned over $P = 4$ processors as

$$W = \begin{pmatrix} A_{00} \\ A_{10} \\ A_{20} \\ A_{30} \end{pmatrix},$$

- The panel factorization uses the following binary tree :



1. Compute Strong RRQR factorization of the transpose of each block A_{i0} of the panel W, find a permutation Π_0

$$\begin{bmatrix} A_{00}^T \Pi_{00} \\ A_{10}^T \Pi_{10} \\ A_{20}^T \Pi_{20} \\ A_{30}^T \Pi_{30} \end{bmatrix} = \begin{bmatrix} Q_{00} R_{00} \\ Q_{10} R_{10} \\ Q_{20} R_{20} \\ Q_{30} R_{30} \end{bmatrix}$$

2. Perform $\log P$ times Strong RRQR factorizations of $2b \times b$ blocks, find permutations Π_1 and Π_2

$$A_{01}^T \Pi_{01} = \left[(A_{00}^T \Pi_{00})(:, 1:b), A_{10}^T \Pi_{10})(:, 1:b) \right]^T \Pi_{01} = Q_{01} R_{01}$$

$$A_{11}^T \Pi_{11} = \left[(A_{20}^T \Pi_{20})(:, 1:b), A_{30}^T \Pi_{30})(:, 1:b) \right]^T \Pi_{11} = Q_{11} R_{11}$$

$$A_{02}^T \Pi_{02} = \left[(A_{01}^T \Pi_{01})(:, 1:b), A_{11}^T \Pi_{11})(:, 1:b) \right]^T \Pi_{02} = Q_{02} R_{02}$$

Stability of CALU_PRRP

Worst case analysis of pivot growth factor for a reduction tree of height H : CALU_PRRP vs CALU.

	matrix of size $m \times (b+1)$	
	TSLU_PRRP(b,H)	TSLU(b,H)
g_W upper bound	$(1+\tau b)^{H+1}$	$2^{b(H+1)}$
	matrix of size $m \times n$	
	CALU_PRRP	CALU
g_W upper bound	$(1+\tau b)^{\frac{n}{b}(H+1)-1}$	$2^{n(H+1)-1}$

Stability of CALU_PRRP

Worst case analysis of pivot growth factor for a reduction tree of height H : CALU_PRRP vs CALU.

	matrix of size $m \times (b+1)$	
	TSLU_PRRP(b,H)	TSLU(b,H)
g_W upper bound	$(1+\tau b)^{H+1}$	$2^{b(H+1)}$
	matrix of size $m \times n$	
	CALU_PRRP	CALU
g_W upper bound	$(1+\tau b)^{\frac{n}{b}(H+1)-1}$	$2^{n(H+1)-1}$

- CALU_PRRP is more stable than CALU.
- For the binary reduction tree, it is more stable than GEPP when $b \geq log(\tau b) \log P$ (b = $\frac{n}{\sqrt{P}}$)

Outline of the proof of stability for CALU_PRRP

- CALU_PRRP as stable as LU_PRRP in following sense:
  the Schur complement obtained after each step of performing CALU_PRRP on a matrix A is equivalent to the Schur complement obtained after performing LU_PRRP on a larger matrix A_{LU_PRRP} whose entries are blocks of A, sometimes slightly perturbed, or zeros.

Outline of the proof of stability for CALU_PRRP

- Let A be an $m \times n$ matrix partitioned as

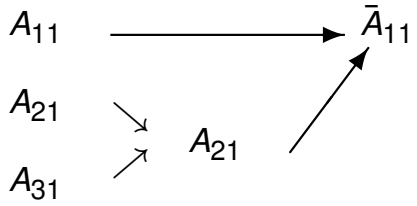
$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{pmatrix},$$

Outline of the proof of stability for CALU_PRRP

- Let A be an $m \times n$ matrix partitioned as

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

- The panel factorization uses the following reduction tree :

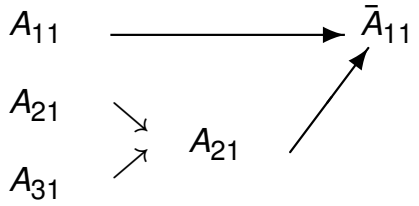


Outline of the proof of stability for CALU_PRRP

- Let A be an $m \times n$ matrix partitioned as

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

- The panel factorization uses the following reduction tree :



- After the factorization of first panel by CALU_PRRP, A_{32}^s (the Schur complement of A_{32}) is not bounded as in LU_PRRP,

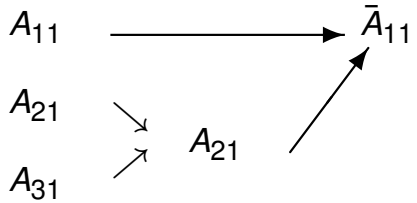
$$\begin{pmatrix} \Pi_{11}^T & \Pi_{12}^T & \\ \Pi_{21}^T & \Pi_{22}^T & \\ & & I_{m-2b} \end{pmatrix} \cdot \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{pmatrix} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \\ A_{31} & A_{32} \end{pmatrix} = \begin{pmatrix} I_b & & \\ A_{21}\bar{A}_{11}^{-1} & I_b & \\ A_{31}\bar{A}_{11}^{-1} & & I_b \end{pmatrix} \cdot \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ & \bar{A}_{22}^s \\ & A_{32}^s \end{pmatrix}$$

Outline of the proof of stability for CALU_PRRP

- Let A be an $m \times n$ matrix partitioned as

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

- The panel factorization uses the following reduction tree :



- After the factorization of first panel by CALU_PRRP, A_{32}^s (the Schur complement of A_{32}) is not bounded as in LU_PRRP,

$$\begin{pmatrix} \Pi_{11}^T & \Pi_{12}^T & \\ \Pi_{21}^T & \Pi_{22}^T & \\ & & I_{m-2b} \end{pmatrix} \cdot \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{pmatrix} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \\ A_{31} & A_{32} \end{pmatrix} = \begin{pmatrix} I_b & & \\ A_{21}\bar{A}_{11}^{-1} & I_b & \\ A_{31}\bar{A}_{11}^{-1} & & I_b \end{pmatrix} \cdot \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ & \bar{A}_{22}^s \\ & A_{32}^s \end{pmatrix}$$

- but A_{32}^s can be obtained by LU_PRRP on a larger matrix A_{LU_PRRP} formed from blocks of A:

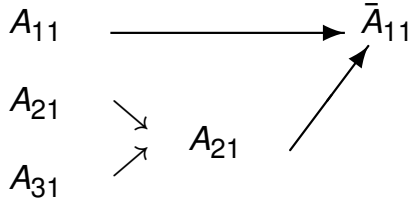
$$A_{LU\_PRRP} = \begin{pmatrix} \bar{A}_{11} & & \bar{A}_{12} \\ A_{21} & A_{21} & \\ & -A_{31} & A_{32} \end{pmatrix} = \begin{pmatrix} I_b & & \\ A_{21}\bar{A}_{11}^{-1} & I_b & \\ & -A_{31}A_{21}^{-1} & I_b \end{pmatrix} \cdot \begin{pmatrix} \bar{A}_{11} & & \bar{A}_{12} \\ & A_{21} & -A_{21}\bar{A}_{11}^{-1}\bar{A}_{12} \\ & & A_{32}^s \end{pmatrix}$$

Outline of the proof of stability for CALU_PRRP

- Let A be an $m \times n$ matrix partitioned as

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

- The panel factorization uses the following reduction tree :



- After the factorization of first panel by CALU_PRRP, A_{32}^s (the Schur complement of A_{32}) is not bounded as in LU_PRRP,

$$\begin{pmatrix} \Pi_{11}^T & \Pi_{12}^T & \\ \Pi_{21}^T & \Pi_{22}^T & \\ & & I_{m-2b} \end{pmatrix} \cdot \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{pmatrix} = \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \\ A_{31} & A_{32} \end{pmatrix} = \begin{pmatrix} I_b & & \\ A_{21}\bar{A}_{11}^{-1} & I_b & \\ A_{31}\bar{A}_{11}^{-1} & & I_b \end{pmatrix} \cdot \begin{pmatrix} \bar{A}_{11} & \bar{A}_{12} \\ & \bar{A}_{22}^s \\ & A_{32}^s \end{pmatrix}$$

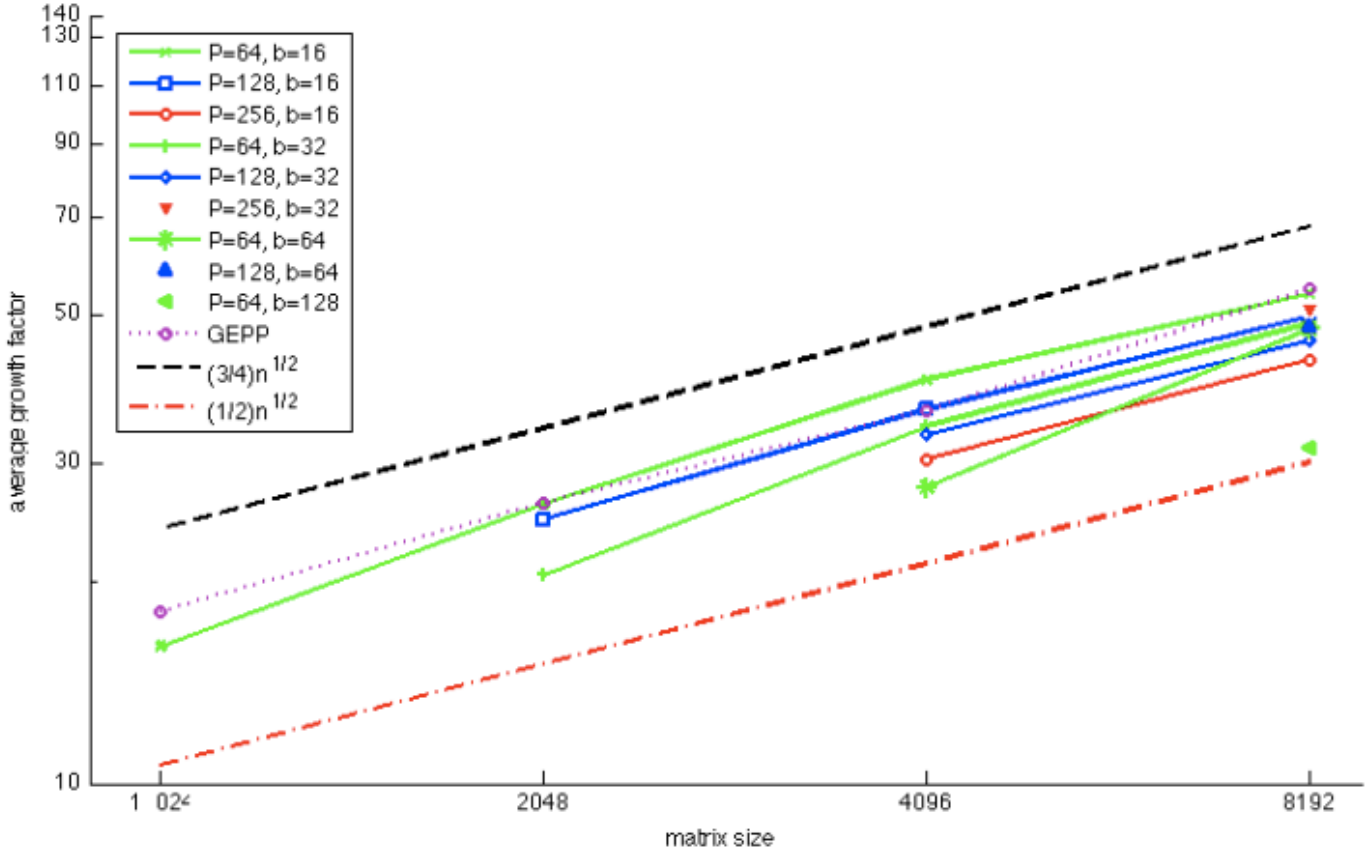
- but A_{32}^s can be obtained by LU_PRRP on a larger matrix A_{LU_PRRP} formed from blocks of A:

$$A_{LU\_PRRP} = \begin{pmatrix} \bar{A}_{11} & & \bar{A}_{12} \\ A_{21} & A_{21} & \\ & -A_{31} & A_{32} \end{pmatrix} = \begin{pmatrix} I_b & & \\ A_{21}\bar{A}_{11}^{-1} & I_b & \\ & -A_{31}A_{21}^{-1} & I_b \end{pmatrix} \cdot \begin{pmatrix} \bar{A}_{11} & & \bar{A}_{12} \\ & A_{21} & -A_{21}\bar{A}_{11}^{-1}\bar{A}_{12} \\ & & A_{32}^s \end{pmatrix}$$

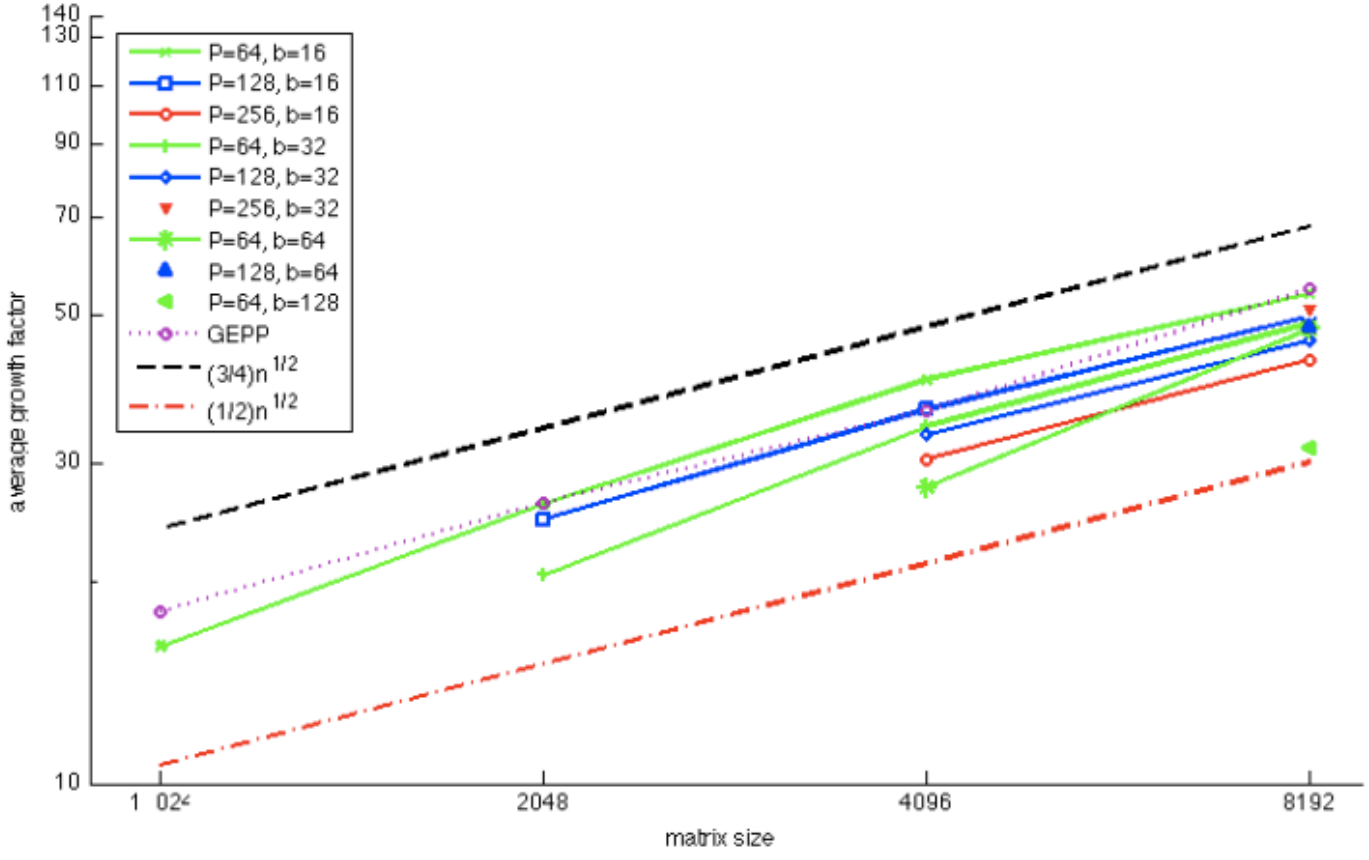
- Strong RRQR on A_{LU_PRRP} does not permute and

$$A_{32} - (-A_{31}A_{21}^{-1})(-A_{21}\bar{A}_{11}^{-1}\bar{A}_{12}) = A_{32} - A_{31}\bar{A}_{11}^{-1}\bar{A}_{12} = A_{32}^s$$

CALU_PRRP: experimental results



CALU_PRRP: experimental results



- CALU_PRRP is as stable as GEPP .
- QR with column pivoting is sufficient to attain the bound τ in practice.
- $(1/2)n^{1/2} \leq g_W \leq (3/4)n^{1/2}$

Conclusion

- Results
  - The new algorithm LU_PRRP is more stable than GEPP.
  - Its Communication Avoiding version, CALU_PRRP, does an optimal amount of communication as CALU.
  - CALU_PRRP is more stable than CALU in terms of worst case pivot growth.
  - CALU_PRRP is more stable than GEPP in terms of worst case pivot growth when $b \geq log(\tau b) \log P$.

Conclusion

- Results
  - The new algorithm LU_PRRP is more stable than GEPP.
  - Its Communication Avoiding version, CALU_PRRP, does an optimal amount of communication as CALU.
  - CALU_PRRP is more stable than CALU in terms of worst case pivot growth.
  - CALU_PRRP is more stable than GEPP in terms of worst case pivot growth when $b \geq log(\tau b) \log P$.
- Ongoing work
  - Hybrid CALU : GEPP at each node of the reduction tree, then additional Strong RRQR on the L factor, work in progress to fix stability issues.
  - Alternative algorithms : Flat tree LU_PRRP and Binary tree LU_PRRP schemes: for each node of the reduction tree, perform LU_PRRP : Strong RRQR on the block then update of the corresponding trailing matrix.

Future work

- Implementation of LU_PRRP as a routine for *LAPACK*.
- Performance comparison between CALU_PRRP and CALU on multicore architectures.

References I

Ballard, G., Demmel, J., Holtz, O., and Schwartz, O. (2011).
Minimizing communication in linear algebra.
to appear in SIAM Journal on Matrix Analysis and Applications.

Foster, L. V. (1994).
Gaussian Elimination with Partial Pivoting Can Fail in Practice.
SIAM J. Matrix Anal. Appl., 15:1354–1362.

Grigori, L., Demmel, J., and Xiang, H. (2010).
CALU: a Communication Optimal LU Factorization Algorithm.
to appear in SIAM Journal on Matrix Analysis and Applications.

Gu, M. and Eisenstat, S. C. (1996).
Efficient Algorithms For Computing A Strong Rank Reaviling QR Factorization.
SIAM J. SCI.COMPUT., 17:848–896.

Trefethen, L. N. and Schreiber, R. S. (1990).
Average-case stability of Gaussian elimination.
SIAM J. Matrix Anal. Appl., 11(3):335–360.

Wilkinson, J. H. (1961).
Error analysis of direct methods of matrix inversion.
J. Assoc. Comput. Mach., 8:281–330.