

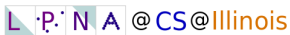
Communication-avoiding factorization algorithms

Edgar Solomonik

Department of Computer Science, University of Illinois at Urbana-Champaign

Conference on Fast Direct Solvers, Purdue University

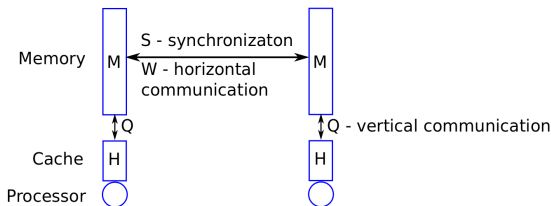
November 10, 2018



Beyond computational complexity

Algorithms should minimize communication, not just computation

- communication and synchronization cost more **energy** than flops
- two types of communication (data movement):



- **vertical** (intranode memory–cache)
- **horizontal** (internode network transfers)
- parallel algorithm design involves tradeoffs: computation vs communication vs synchronization
- parameterized algorithms provide optimality and flexibility

Cost model for parallel algorithms

We use the **Bulk Synchronous Parallel (BSP) model** (L.G. Valiant 1990)

- execution is subdivided into S supersteps, each associated with a global **synchronization** (cost α)
- at the start of each superstep, processors interchange messages, then they perform local computation
- if the **maximum amount of data** sent or received by any process is w_i (work done is f_i and amount of memory traffic is q_i) at superstep i then the BSP time is

$$T = \sum_{i=1}^S \alpha + w_i \cdot \beta + q_i \cdot \nu + f_i \cdot \gamma = O(S \cdot \alpha + W \cdot \beta + Q \cdot \nu + F \cdot \gamma)$$

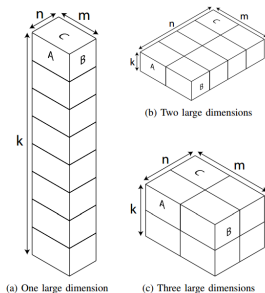
where typically $\alpha \gg \beta \gg \nu \gg \gamma$

- we mention vertical communication cost only when it exceeds $Q = O(F/\sqrt{H} + W)$ where H is cache size

Communication complexity of matrix multiplication

Multiplication of $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{k \times n}$ can be done in $O(1)$ supersteps with **communication cost** $W = O\left(\left(\frac{mnk}{p}\right)^{2/3}\right)$ provided sufficient memory and sufficiently large p

- when $m = n = k$, 3D blocking gets $O(p^{1/6})$ improvement over 2D¹
- when m, n, k are unequal, need appropriate processor grid²



¹J. Berntsen, Par. Comp., 1989; A. Aggarwal, A. Chandra, M. Snir, TCS, 1990; R.C. Agarwal, S.M. Balle, F.G. Gustavson, M. Joshi, P. Palkar, IBM, 1995; F.W. McColl, A. Tiskin, Algorithmica, 1999; ...

²J. Demmel, D. Eliahu, A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, O. Spillinger 2013

Communication complexity of dense matrix kernels

For $n \times n$ **Cholesky** with p processors

$$F = O(n^3/p), \quad W = O(n^2/p^\delta), \quad S = O(p^\delta)$$

given memory to store $p^{2\delta-1}$ copies of the matrix for any $\delta = [1/2, 2/3]$.

Can achieve similar costs for LU, QR, and the symmetric eigenvalue problem (modulo logarithmic factors on synchronization), but algorithmic changes (as opposed to parallel schedules) are necessary.

triangular solve	square TRSM $\checkmark^3$	rectangular TRSM $\checkmark^4$
LU with pivoting	pairwise pivoting $\checkmark^5$	tournament pivoting $\checkmark^6$
QR factorization	Givens on square $\checkmark^3$	Householder on rect. $\checkmark^7$
SVD (sym. eig.)	singular values only $\checkmark^8$	singular vectors $\times$

³ B. Lipshitz, MS thesis 2013

⁴ T. Wicky, E.S., T. Hoefer, IPDPS 2017

⁵ A. Tiskin, FGCS 2007

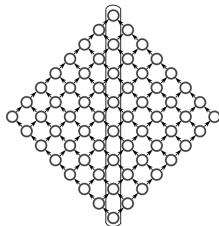
⁶ E.S., J. Demmel, EuroPar 2011

⁷ E.S., G. Ballard, T. Hoefer, J. Demmel, SPAA 2017

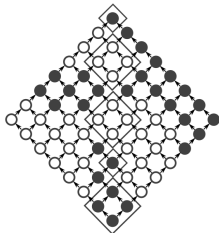
Tradeoffs between costs based on dependency graphs

Definition ((ϵ, σ)-path-expander)

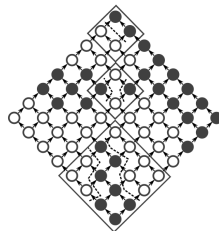
Graph $G = (V, E)$ is a (ϵ, σ) -**path-expander** if there exists a path $(u_1, \dots, u_n) \subset V$, such that the dependency interval $[u_i, u_{i+b}]_G$ for each i, b has size $\Theta(\sigma(b))$ and a minimum cut of size $\Omega(\epsilon(b))$.



Dependency chain P



Monochrome dependency intervals



Multicolored dependency intervals

- computation-synchronizatoin tradeoff in diamond DAG⁸: $F \cdot S = \Omega(n^2)$
- extends to triangular solve, matrix factorization, and iterative methods⁹

⁸C.H. Papadimitriou, J.D. Ullman, SIAM JC, 1987

⁹E.S., E. Carson, N. Knight, J. Demmel, JPDC 2017

Tradeoffs between costs

Definition ((ϵ, σ)-path-expander)

Graph $G = (V, E)$ is a (ϵ, σ) -**path-expander** if there exists a path $(u_1, \dots, u_n) \subset V$, such that the dependency interval $[u_i, u_{i+b}]_G$ for each i, b has size $\Theta(\sigma(b))$ and a minimum cut of size $\Omega(\epsilon(b))$.

Theorem (Path-expander communication lower bound)

*Any parallel schedule of an algorithm with a (ϵ, σ) -**path-expander** dependency graph about a path of length n and some $b \in [1, n]$ incurs computation (F), communication (W), and synchronization (S) costs:*

$$F = \Omega(\sigma(b) \cdot n/b), \quad W = \Omega(\epsilon(b) \cdot n/b), \quad S = \Omega(n/b).$$

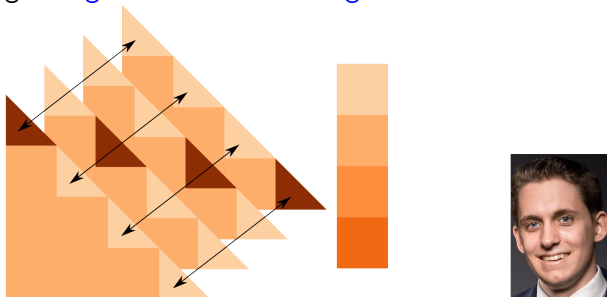
Corollary (Computation-sync. and bandwidth-sync. tradeoffs)

If $\sigma(b) = b^d$ and $\epsilon(b) = b^{d-1}$, the above theorem yields,

$$F \cdot S^{d-1} = \Omega(n^d), \quad W \cdot S^{d-2} = \Omega(n^{d-1}).$$

New algorithms can circumvent lower bounds

For TRSM, we can achieve a lower synchronization/communication cost by performing **triangular inversion on diagonal blocks**



- MS thesis work by Tobias Wicky¹⁰
- **decreases synchronization cost** by $O(p^{2/3})$ on p processors with respect to known algorithms
- optimal communication for **any number of right-hand sides**

¹⁰T. Wicky, E.S., T. Hoefer, IPDPS 2017

QR factorization of tall-and-skinny matrices

Consider the reduced factorization $\mathbf{A} = \mathbf{Q}\mathbf{R}$ with $\mathbf{A}, \mathbf{Q} \in \mathbb{R}^{m \times n}$ and $\mathbf{R} \in \mathbb{R}^{n \times n}$ when $m \gg n$ (in particular $m \geq np$)

- $\mathbf{A}$ is tall-and-skinny, each processor owns a block of rows
- Householder-QR requires $S = \Theta(n)$ supersteps, $W = O(n^2)$ comm.
- **TSQR**¹¹ row-wise divide-and-conquer, $W = O(n^2 \log p)$, $S = O(\log p)$

$$\begin{bmatrix} \mathbf{Q}_1 \mathbf{R}_1 \\ \mathbf{Q}_2 \mathbf{R}_2 \end{bmatrix} = \begin{bmatrix} \text{TSQR}(\mathbf{A}_1) \\ \text{TSQR}(\mathbf{A}_2) \end{bmatrix}, \mathbf{Q}_{12} \mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & \\ & \mathbf{Q}_2 \end{bmatrix} \mathbf{Q}_{12}$$

- **TSQR-HR**¹² Householder rep. $\mathbf{I} - \mathbf{Y}\mathbf{T}\mathbf{Y}$, $W = O(n^2 \log p)$, $S = O(\log p)$
- **Cholesky-QR2**¹³ stable so long as $\kappa(\mathbf{A}) \leq 1/\sqrt{\epsilon}$, achieves $W = O(n^2)$, $S = O(1)$, Cholesky-QR3¹⁴ gets same and is unconditionally stable

¹¹J. Demmel, L. Grigori, M. Hoemmen, J. Langou 2012

¹²G. Ballard, J. Demmel, L. Grigori, M. Jacquelin, H.-D. Nguyen, E.S. 2014

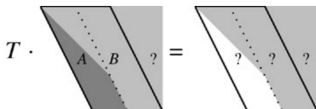
¹³Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, T. Fukaya 2015

¹⁴T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, Y. Yanagisawa 2018

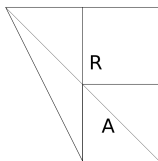
QR factorization of square matrices

Square matrix QR algorithms generally use 1D QR for panel factorization

- algorithms in ScaLAPACK, Elemental, DPLASMA use **2D layout**, generally achieve $W = O(n^2/\sqrt{p})$ cost
- Tiskin's 3D QR algorithm¹⁵ achieves $W = O(n^2/p^{2/3})$ communication



- however, requires **slanted-panel matrix embedding**



which is highly inefficient for rectangular (tall-and-skinny) matrices

¹⁵ A. Tiskin 2007, "Communication-efficient generic pairwise elimination"

Communication-avoiding rectangular QR

For $A \in \mathbb{R}^{m \times n}$ existing algorithms are optimal when $m = n$ and $m \gg n$

- cases with $n < m < np$ underdetermined equations are important
- new algorithm¹⁶
 - subdivide p processors into m/n groups of pn/m processors
 - perform row-recursive QR (TSQR) with tree of height $\log_2(m/n)$
 - compute each tree-node elimination $Q_{12}R = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}$ using Tiskin's QR with pn/m or more processors
- note: interleaving rows of R_1 and R_2 gives a **slanted panel**
- obtains ideal communication cost for any m, n , generally

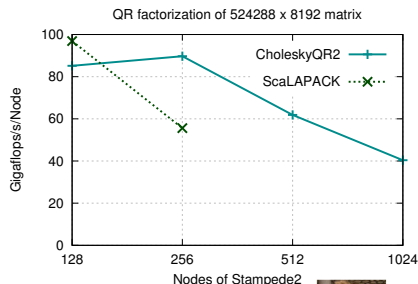
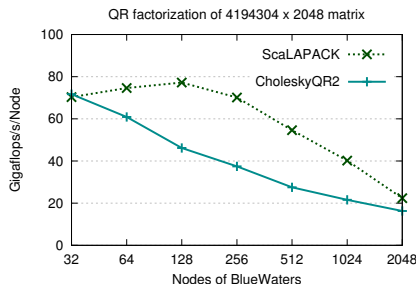
$$W = O\left(\left(\frac{mn^2}{p}\right)^{2/3}\right)$$

¹⁶E.S., G. Ballard, J. Demmel, and T. Hoeftler, SPAA 2017

Cholesky-QR2 for rectangular matrices

Cholesky-QR2¹⁷ with 3D Cholesky gives a practical 3D QR algorithm¹⁸

- Compute $A = \hat{Q}\hat{R}$ using Cholesky-QR $A^T A = \hat{R}^T \hat{R}$, $\hat{Q} = A\hat{R}^{-1}$
- Correct approximate factorization by Cholesky-QR $Q\bar{R} = \hat{Q}$, $R = \bar{R}\hat{R}$
- Simple algorithm to achieve minimize comm. and sync. for any m, n, p



Analysis and implementation by PhD student Edward Hutter



¹⁷T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, Y. Yamamoto 2014

¹⁸E. Hutter, E.S. 2018

Tridiagonalization

Reducing the symmetric matrix $A \in \mathbb{R}^{n \times n}$ to a tridiagonal matrix

$$T = Q^T A Q$$

via a **two-sided orthogonal transformation** is most costly in diagonalization (eigenvalue computation, SVD similar)

- can be done by **successive subcolumn QR factorizations**

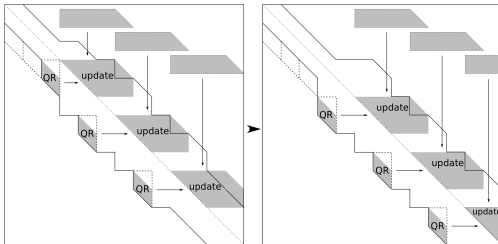
$$T = \underbrace{Q_1^T \cdots Q_{n-2}^T}_{Q^T} A \underbrace{Q_1 \cdots Q_{n-2}}_Q$$

- two-sided updates harder to parallelize than one-sided
- each update requires a BSP superstep and reading A from memory
- can use n/b QRs on panels of b subcolumns to go to band-width $b + 1$
- $b = 1$ gives direct tridiagonalization

Successive band reduction (SBR)

After reducing to a banded matrix, we need to transform the banded matrix to a tridiagonal one

- fewer nonzeros lead to lower computational cost, $F = O(n^2 b/p)$
- however, transformations introduce **fill/bulges**
- bulges must be chased down the band¹⁹



- communication- and synchronization-efficient **1D SBR algorithm** known for small band-width²⁰

¹⁹ Lang 1993; Bischof, Lang, Sun 2000

²⁰ Ballard, Demmel, Knight 2012

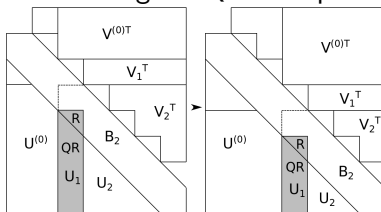
Communication-efficient eigenvalue computation

Previous work (start-of-the-art): **two-stage tridiagonalization**

- implemented in ELPA, can outperform ScaLAPACK²¹
- with $n = n/\sqrt{p}$, 1D SBR gives $W = O(n^2/\sqrt{p})$, $S = O(\sqrt{p} \log^2(p))$ ²²

New results²³: **many-stage tridiagonalization**

- $\Theta(\log(p))$ intermediate band-widths to achieve $W = O(n^2/p^{2/3})$
- communication-efficient rectangular QR with processor groups



- 3D SBR (each QR and matrix multiplication update parallelized)

²¹Auckenthaler, Bungartz, Huckle, Krämer, Lang, Willems 2011

²²Ballard, Demmel, Knight 2012

²³E.S., G. Ballard, J. Demmel, T. Hoeftler, SPAA 2017

Symmetric eigensolver results summary

Algorithm	W	Q	S
ScaLAPACK	$n^2/\sqrt{p}$	n^3/p	$n \log(p)$
ELPA	$n^2/\sqrt{p}$	-	$n \log(p)$
two-stage + 1D-SBR	$n^2/\sqrt{p}$	$n^2 \log(n)/\sqrt{p}$	$\sqrt{p}(\log^2(p) + \log(n))$
many-stage	$n^2/p^{2/3}$	$n^2 \log(p)/p^{2/3}$	$p^{2/3} \log^2 p$

- costs are asymptotic (same computational cost F for eigenvalues)
- W – horizontal (interprocessor) communication
- Q – vertical (memory–cache) communication excluding $W + F/\sqrt{H}$ where H is cache size
- S – synchronization cost (number of supersteps)

Summary of new communication avoiding algorithms

- communication-efficient **QR factorization** algorithm
 - optimal communication cost for any matrix dimensions
 - variants that trade-off some accuracy guarantees for performance
- communication-efficient **symmetric eigensolver** algorithm
 - reduce matrix to successively smaller band-width
 - uses concurrent executions of 3D matrix multiplication and 3D QR

Practical implications

- ELPA demonstrated efficacy of two-stage approach, **our work motivates 3+ stages**
- partial parallel implementation is competitive but no speed-up

Future work

- back-transformations to compute **eigenvectors** in less computational complexity than $F = O(n^3 \log(p)/p)$
- QR with **column pivoting** / low-rank SVD / sparse factorization

Acknowledgements

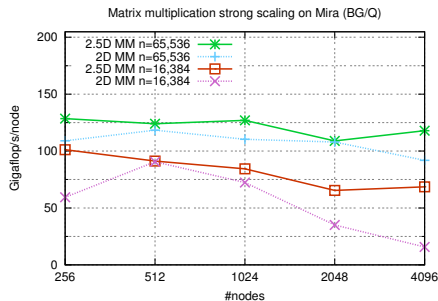
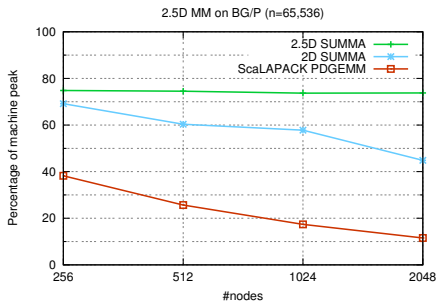
Collaborators on this work

- Edward Hutter (Department of Computer Science, University of Illinois at Urbana-Champaign)
- Grey Ballard (Department of Computer Science, Wake Forest University)
- James Demmel (Department of Computer Science and Department of Mathematics, University of California, Berkeley)
- Tobias Wicky (Department of Computer Science, ETH Zurich)
- Torsten Hoefer (Department of Computer Science, ETH Zurich)
- Erin Carson (Courant Institute of Mathematical Sciences, NYU)
- Nicholas Knight (Courant Institute of Mathematical Sciences, NYU)

Computational resources and funding

- DOE Computational Science Graduate Fellowship
- ETH Zurich Postdoctoral Fellowship
- XSEDE/TACC (Stampede2) and NCSA (BlueWaters)

Communication-efficient matrix multiplication



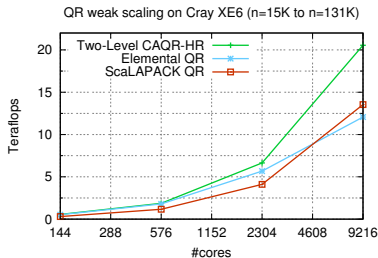
12X speed-up, 95% reduction in comm. for $n = 8K$ on 16K nodes of BG/P

Communication-efficient QR factorization

- Householder form can be reconstructed quickly from TSQR²⁴

$$Q = I - YTY^T \quad \Rightarrow \quad \text{LU}(I - Q) \rightarrow (Y, TY^T)$$

- Householder aggregation yields performance improvements

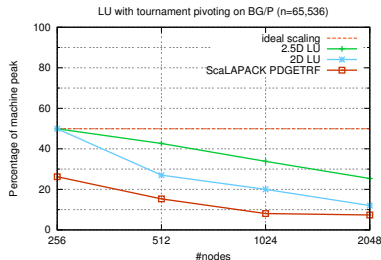


²⁴ Ballard, Demmel, Grigori, Jacquelin, Nguyen, S., IPDPS, 2014

Communication-efficient LU factorization

For any $c \in [1, p^{1/3}]$, use cn^2/p memory per processor and obtain

$$W_{\text{LU}} = O(n^2/\sqrt{cp}), \quad S_{\text{LU}} = O(\sqrt{cp})$$



- LU with pairwise pivoting²⁵ extended to tournament pivoting²⁶
- first implementation of a communication-optimal LU algorithm¹¹

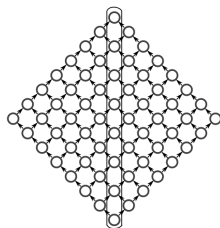
²⁵Tiskin, FGCS, 2007

²⁶S., Demmel, Euro-Par, 2011

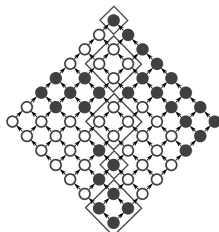
Tradeoffs in the diamond DAG

Computation vs synchronization tradeoff for the $n \times n$ diamond DAG,²⁷

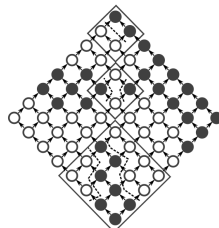
$$F \cdot S = \Omega(n^2)$$



Dependency chain P



Monochrome dependency intervals



Multicolored dependency intervals

We generalize this idea²⁸

- additionally consider horizontal communication
- allow arbitrary (polynomial or exponential) interval expansion

²⁷Papadimitriou, Ullman, SIAM JC, 1987

²⁸S., Carson, Knight, Demmel, SPAA 2014 (extended version, JPDC 2016)

Tradeoffs involving synchronization

We apply tradeoff lower bounds to dense linear algebra algorithms, represented via dependency hypergraphs:²⁹

For triangular solve with an $n \times n$ matrix,

$$F_{\text{TRSV}} \cdot S_{\text{TRSV}} = \Omega(n^2)$$

For Cholesky of an $n \times n$ matrix,

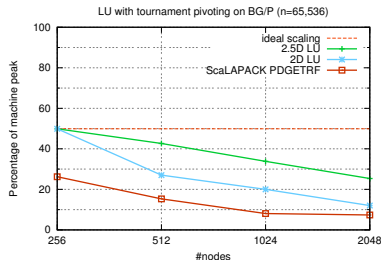
$$F_{\text{CHOL}} \cdot S_{\text{CHOL}}^2 = \Omega(n^3) \quad W_{\text{CHOL}} \cdot S_{\text{CHOL}} = \Omega(n^2)$$

²⁹S., Carson, Knight, Demmel, SPAA 2014 (extended version, JPDC 2016)

Communication-efficient LU factorization

For any $c \in [1, p^{1/3}]$, use cn^2/p memory per processor and obtain

$$W_{\text{LU}} = O(n^2/\sqrt{cp}), \quad S_{\text{LU}} = O(\sqrt{cp})$$



- LU with pairwise pivoting³⁰ extended to tournament pivoting³¹
- first implementation of a communication-optimal LU algorithm¹⁰

³⁰Tiskin, FGCS, 2007

³¹S., Demmel, Euro-Par, 2011