A communication-avoiding parallel algorithm for the symmetric eigenvalue problem

Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

Householder Symposium XX

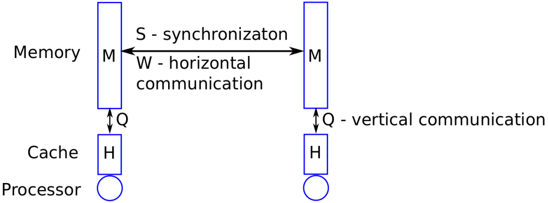
June 19, 2017

L P N A @CS@Illinois

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

BSP model definition

The Bulk Synchronous Parallel (BSP) model[1] is a theoretical execution/cost model for parallel algorithms

- 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 m_i at superstep i then the horizontal communication cost is

$$T = \sum_{i=1}^{s} \alpha + m_i \cdot \beta$$

[1] Valiant 1990

Cost model for parallel algorithms

In addition to computation and BSP horizontal communication cost, we consider vertical communication cost

- F – computation cost (local computation)
- Q – vertical communication cost (memory–cache traffic)
- W – horizontal communication cost (interprocessor communication)
- S – synchronization cost (number of supersteps)

Symmetric eigenvalue problem

Given a dense symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ find diagonal matrix $\mathbf{D}$ so

$$\mathbf{AX} = \mathbf{XD}$$

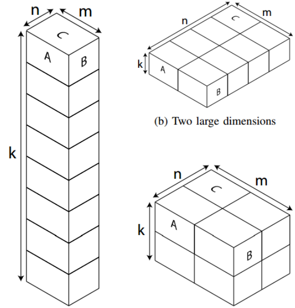
where $\mathbf{X}$ is an orthogonal matrix composed of eigenvectors of $\mathbf{A}$

- diagonalization – reduction of $\mathbf{A}$ to diagonal matrix $\mathbf{D}$
- computing the SVD has very similar computational structure
- we focus on tridiagonalization (bidiagonalization for SVD), from which standard approaches (e.g. MRRR) can be used
- core building blocks:
  - matrix multiplication
  - QR factorization

Parallel matrix multiplication

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

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



(a) One large dimension (b) Two large dimensions (c) Three large dimensions

[2] Berntsen, Par. Comp., 1989; Agarwal, Chandra, Snir, TCS, 1990; Agarwal, Balle, Gustavson, Joshi, Palkar, IBM, 1995; McColl, Tiskin, Algorithmica, 1999; ...

[3] Demmel, Eliahu, Fox, Kamil, Lipshitz, Schwartz, Spillinger 2013

Bandwidth-efficient QR and diagonalization

Goal: achieve the same communication complexity for QR and diagonalization as for matrix multiplication

- synchronization complexity expected to be higher

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

  product of communication and synchronization cost must be greater than the square of the number of columns

- general strategy
  1. use communication-efficient matrix-multiplication for QR
  2. use communication-efficient QR for diagonalization

QR factorization of tall-and-skinny matrices

Consider the reduced factorization $\mathbf{A} = \mathbf{QR}$ 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, $W = O(n^2)$
- Cholesky-QR2, TSQR, and HR-TSQR require $S = \Theta(\log(p))$ supersteps
  - Cholesky-QR2[4]: stable so long as $\kappa(\mathbf{A}) \leq 1/\sqrt{\epsilon}$, $W = O(n^2)$

$$\mathbf{L} = \mathrm{Chol}(\mathbf{A}^T\mathbf{A}), \mathbf{Z} = \mathbf{A}\mathbf{L}^{-T}, \bar{\mathbf{L}} = \mathrm{Chol}(\mathbf{Z}^T\mathbf{Z}), \mathbf{Q} = \mathbf{Z}\bar{\mathbf{L}}^{-T}, \mathbf{R} = \bar{\mathbf{L}}^T\mathbf{L}^T$$

  - TSQR[5]: row-recursive divide-and-conquer, $W = O(n^2\log(p))$

$$\begin{bmatrix} \mathbf{Q}_1\mathbf{R}_1 \\ \mathbf{Q}_2\mathbf{R}_2 \end{bmatrix} = \begin{bmatrix} \mathrm{TSQR}(\mathbf{A}_1) \\ \mathrm{TSQR}(\mathbf{A}_2) \end{bmatrix}, [\mathbf{Q}_{12}, \mathbf{R}] = \mathrm{QR}\left(\begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix}\right), \mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_2 \end{bmatrix} \mathbf{Q}_{12}$$

  - TSQR-HR[6]: TSQR with Householder-reconstruction, $W = O(n^2\log(p))$

[4] Yamamoto, Nakatsukasa, Yanagisawa, Fukaya 2015

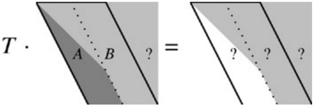
[5] Demmel, Grigori, Hoemmen, Langou 2012

[6] Ballard, Demmel, Grigori, Jacquelin, Nguyen, S. 2014

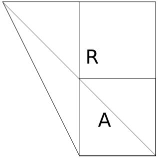
QR factorization of square matrices

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 algorithm[7] 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

[7] Tiskin 2007, "Communication-efficient generic pairwise elimination"

Communication-avoiding rectangular QR

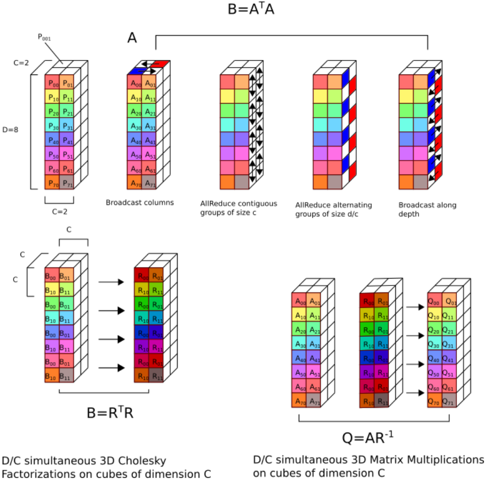
For $\mathbf{A} \in \mathbb{R}^{m \times n}$ existing algorithms are optimal when $m = n$ and $m \gg n$

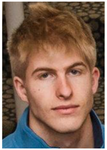
- cases with $n < m < np$ underdetermined equations are important
- new algorithm
  - subdivide p processors into m/n groups of pn/m processors
  - perform row-recursive QR (TSQR) with tree of height $\log_2(m/n)$
  - compute each tree-node elimination $\text{QR}\left(\begin{bmatrix}\mathbf{R_1} \\ \mathbf{R_2}\end{bmatrix}\right)$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

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

Cholesky-QR2 for rectangular matrices

Cholesky-QR2 with 3D Cholesky provides a simple 3D QR algorithm for well-conditioned rectangular matrices





work by Edward Hutter (PhD student at UIUC)

Tridiagonalization

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

$$\mathbf{T} = \mathbf{Q}^T \mathbf{A} \mathbf{Q}$$

via a two-sided orthogonal transformation is most costly in diagonalization

- can be done by successive column QR factorizations

$$\mathbf{T} = \underbrace{\mathbf{Q_1^T} \cdots \mathbf{Q_n^T}}_{\mathbf{Q}^T} \mathbf{A} \underbrace{\mathbf{Q_1} \cdots \mathbf{Q_n}}_{\mathbf{Q}}$$

- two-sided updates harder to manage than one-sided
- can use n/b QRs on panels of b columns to go to band-width $b+1$
- $b = 1$ gives direct tridiagonalization

Writing the orthogonal transformation in Householder form, we get

$$\underbrace{(\mathbf{I} - \mathbf{U}\mathbf{T}\mathbf{U}^T)^T}_{\mathbf{Q}^T}\mathbf{A}\underbrace{(\mathbf{I} - \mathbf{U}\mathbf{T}\mathbf{U}^T)}_{\mathbf{Q}} = \mathbf{A} - \mathbf{U}\mathbf{V}^T - \mathbf{V}\mathbf{U}^T$$

where $\mathbf{U}$ are Householder vectors and $\mathbf{V}$ is

$$\mathbf{V}^T = \mathbf{T}\mathbf{U}^T + \frac{1}{2}\mathbf{T}^T\mathbf{U}^T \underbrace{\mathbf{A}\mathbf{U}}_{\text{challenge}} \mathbf{T}\mathbf{U}^T$$

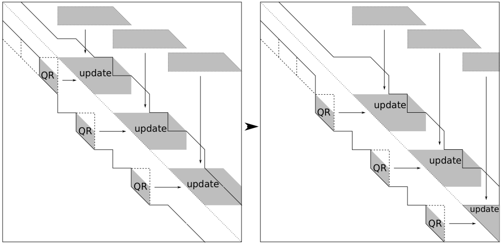
- when performing two-sided updates, computing $\mathbf{AU}$ dominates cost
- if $b = 1$, $\mathbf{U}$ is a column-vector, and $\mathbf{AU}$ is dominated by vertical communication cost (moving $\mathbf{A}$ between memory and cache)
- idea: reduce to banded matrix ($b \gg 1$) first[8]

[8] Auckenthaler, Bungartz, Huckle, Krämer, Lang, Willems 2011

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 band[9]



- communication- and synchronization-efficient 1D SBR algorithm known for small band-width[10]

[9] Lang 1993; Bischof, Lang, Sun 2000

[10] Ballard, Demmel, Knight 2012

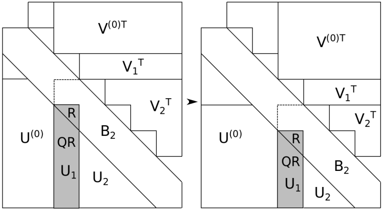
Communication-efficient eigenvalue computation

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

- implemented in ELPA, can outperform ScaLAPACK[11]
- with $n = n/\sqrt{p}$, 1D SBR gives $W = O(n^2/\sqrt{p})$, $S = O(\sqrt{p}\log^2(p))$[12]

New results[13]: many-stage tridiagonalization

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



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

[11] Auckenthaler, Bungartz, Huckle, Krämer, Lang, Willems 2011

[12] Ballard, Demmel, Knight 2012

[13] S., Ballard, Demmel, Hoefler 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 $W + F/\sqrt{H}$
- S – synchronization cost (number of supersteps)

Conclusion

Summary of contributions

- 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

Acknowledgements

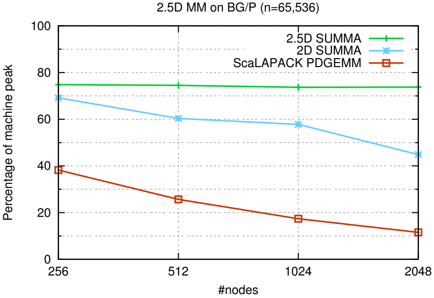
Talk based on joint work with

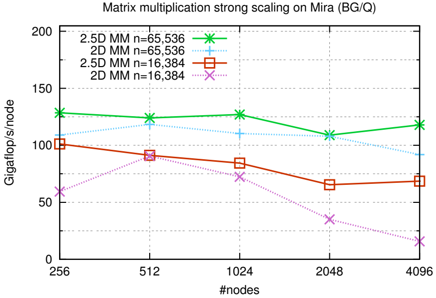
- Edward Hutter (UIUC)
- Grey Ballard (Wake Forest University)
- James Demmel (UC Berkeley)
- Torsten Hoefler (ETH Zurich)

For more details see "E.S., Grey Ballard, James Demmel, and Torsten Hoefler, *A communication-avoiding parallel algorithm for the symmetric eigenvalue problem*, SPAA 2017."

Backup slides

Communication-efficient matrix multiplication

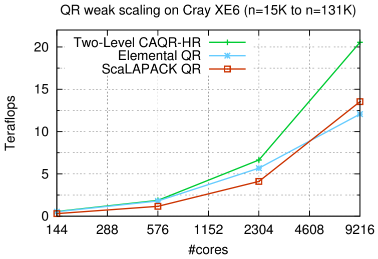




12X speed-up, 95% reduction in comm. for $n = 8\text{K}$ on 16K nodes of BG/P

Communication-efficient QR factorization

- Householder form can be reconstructed quickly from TSQR[14]
$$\mathbf{Q} = \mathbf{I} - \mathbf{Y}\mathbf{T}\mathbf{Y}^T \quad \Rightarrow \quad \mathrm{LU}(\mathbf{I} - \mathbf{Q}) \rightarrow (\mathbf{Y}, \mathbf{T}\mathbf{Y}^T)$$
- Householder aggregation yields performance improvements

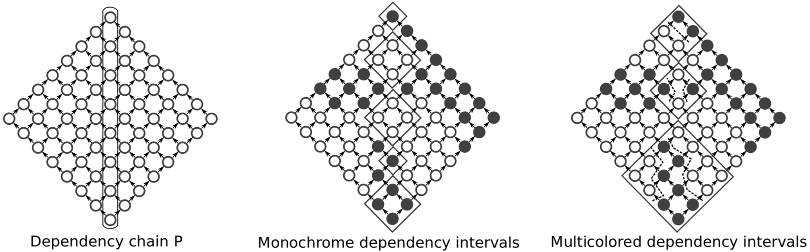


[14] Ballard, Demmel, Grigori, Jacquelin, Nguyen, S., IPDPS, 2014

Tradeoffs in the diamond DAG

Computation vs synchronization tradeoff for the $n \times n$ diamond DAG,[15]

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



We generalize this idea[16]

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

[15] Papadimitriou, Ullman, SIAM JC, 1987

[16] S., 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:[17]
For triangular solve with an $n \times n$ matrix,

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

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

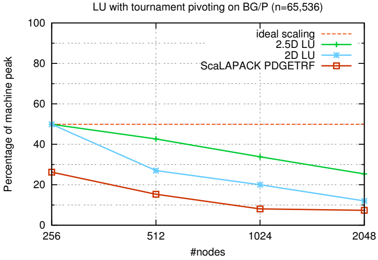
$$F_{\text{CHOL}} \cdot S_{\text{CHOL}}^2 = \Omega\left(n^3\right) \qquad W_{\text{CHOL}} \cdot S_{\text{CHOL}} = \Omega\left(n^2\right)$$

[17] S., 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

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



- LU with pairwise pivoting[18] extended to tournament pivoting[19]
- first implementation of a communication-optimal LU algorithm[10]

---

[18] Tiskin, FGCS, 2007

[19] S., Demmel, Euro-Par, 2011