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
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\textsuperscript{1} is a theoretical execution/cost model for parallel algorithms

- execution is subdivided into \( s \) supersteps, each associated with a global synchronization (cost \( \alpha \))
- 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

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

\textsuperscript{1}Valiant 1990
In addition to computation and BSP horizontal communication cost, we consider \textit{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 \( A \in \mathbb{R}^{n \times n} \) find diagonal matrix \( D \) so

\[
AX = XD
\]

where \( X \) is an orthogonal matrix composed of eigenvectors of \( A \)

- **diagonalization** – reduction of \( A \) to diagonal matrix \( D \)
- 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 \( A \in \mathbb{R}^{m \times k} \) 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 sufficiently memory and sufficiently large \( p \)

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

---

\[\text{References:}\]

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

- 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
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 $\kappa(\mathbf{A}) \leq 1/\sqrt{\varepsilon}$, $W = O(n^2)$

\[
\mathbf{L} = \text{Chol}(\mathbf{A}^T \mathbf{A}), \quad \mathbf{Z} = \mathbf{A} \mathbf{L}^{-T}, \quad \bar{\mathbf{L}} = \text{Chol}(\mathbf{Z}^T \mathbf{Z}), \quad \mathbf{Q} = \mathbf{Z} \bar{\mathbf{L}}^{-T}, \quad \mathbf{R} = \bar{\mathbf{L}}^T \mathbf{L}^T
\]

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

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

- 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
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\textsuperscript{7} achieves $W = O(n^2/p^{2/3})$ communication, however, requires slanted-panel matrix embedding.

\textsuperscript{7}Tiskin 2007, “Communication-efficient generic pairwise elimination”

which is highly inefficient for rectangular (tall-and-skinny) matrices.
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 QR($\begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix}$) 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 with 3D Cholesky provides a simple 3D QR algorithm for well-conditioned rectangular matrices

work by Edward Hutter (PhD student at UIUC)
Reducing the symmetric matrix $A \in \mathbb{R}^{n \times n}$ to a tridiagonal matrix $T = Q^T AQ$

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

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

\[
T = \underbrace{Q_1^T \cdots Q_n^T}_Q \underbrace{A Q_1 \cdots Q_n}_T
\]

- 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

\[ \left( I - UTU^T \right)^T A \left( I - UTU^T \right) = A - UV^T - VU^T \]

where \( U \) are Householder vectors and \( V \) is

\[ V^T = TU^T + \frac{1}{2} T^T U^T A U \quad TU^T \]

- when performing two-sided updates, computing \( AU \) dominates cost
- if \( b = 1 \), \( U \) is a column-vector, and \( AU \) is dominated by vertical communication cost (moving \( A \) 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

![Diagram of SBR process](image)

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

---

9. Lang 1993; Bischof, Lang, Sun 2000
10. Ballard, Demmel, Knight 2012
Previous work (start-of-the-art): two-stage tridiagonalization
- implemented in ELPA, can outperform ScaLAPACK\textsuperscript{11}
- with $n = n/\sqrt{p}$, 1D SBR gives $W = O(n^2/\sqrt{p})$, $S = O(\sqrt{p} \log^2(p))$\textsuperscript{12}

New results\textsuperscript{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)

\textsuperscript{11} Auckenthaler, Bungartz, Huckle, Krämer, Lang, Willems 2011
\textsuperscript{12} Ballard, Demmel, Knight 2012
\textsuperscript{13} S., Ballard, Demmel, Hoefler 2017
Symmetric eigensolver results summary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$W$</th>
<th>$Q$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScaLAPACK</td>
<td>$n^2/\sqrt{p}$</td>
<td>$n^3/p$</td>
<td>$n\log(p)$</td>
</tr>
<tr>
<td>ELPA</td>
<td>$n^2/\sqrt{p}$</td>
<td>-</td>
<td>$n\log(p)$</td>
</tr>
<tr>
<td>two-stage + 1D-SBR</td>
<td>$n^2/\sqrt{p}$</td>
<td>$n^2 \log(n)/\sqrt{p}$</td>
<td>$\sqrt{p}(\log^2(p) + \log(n))$</td>
</tr>
<tr>
<td>many-stage</td>
<td>$n^2/p^{2/3}$</td>
<td>$n^2 \log p/p^{2/3}$</td>
<td>$p^{2/3} \log^2 p$</td>
</tr>
</tbody>
</table>

- 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
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.”
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 TSQR\(^{14}\)
  \[ Q = I - YTY^T \quad \Rightarrow \quad LU(I - Q) \rightarrow (Y, TY^T) \]
- 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:\textsuperscript{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,

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

\textsuperscript{17}S., Carson, Knight, Demmel, SPAA 2014 (extended version, JPDC 2016)
For any $c \in [1, p^{1/3}]$, use $cn^2/p$ memory per processor and obtain

$$W_{LU} = O\left(\frac{n^2}{\sqrt{cp}}\right), \quad S_{LU} = O\left(\sqrt{cp}\right)$$

- 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