Communication-avoiding Cholesky-QR2 for rectangular matrices

Edward Hutter and Edgar Solomonik

Laboratory for Parallel Numerical Algorithms
Department of Computer Science
University of Illinois at Urbana-Champaign

March 9, 2018

L·P·N·A @CS@Illinois
Abstract

- **Novelty:**
  - new distributed-memory QR factorization algorithm
  - extends recent Cholesky-QR2 algorithm to rectangular matrices
  - exploits a tunable processor grid to provably reduce communication
  - utilizes first distributed-memory implementation of recursive 3D Cholesky factorization

- **Benefits**
  - practical, flexible, and achieves minimal communication (data movement between processors)

- **Drawbacks**
  - matrix must be sufficiently well-conditioned
  - machine must have sufficient memory
Motivation for reducing parallel algorithmic costs

Communication cost model:

- $\alpha$ - cost of sending a message over a network
- $\beta$ - cost of injecting a byte of data into a network
- $\gamma$ - cost of a floating point operation

Current cost trend for distributed-memory machines:

- $\alpha \gg \beta \gg \gamma$

Goal: A QR factorization algorithm that prioritizes minimizing synchronization and communication cost
2D and 3D QR algorithm characteristics

- 2D algorithms are communication-optimal assuming minimal memory footprint
- 3D algorithms take advantage of extra memory to reduce communication
  - exist in theory but have not been implemented or studied in practice.\(^1\) \(^2\)

\(^1\) A. Tiskin 2007, "Communication-efficient generic pairwise elimination"
\(^2\) E. Solomonik et al., "A communication-avoiding parallel algorithm for the symmetric eigenvalue problem"
Parallel QR factorization cost comparison

- Scalapack PGEQRF method is a 2D Householder QR
- CA-QR utilizes TSQR along panels to reduce synchronization
- Communication-avoiding CholeskyQR2 is a tunable Cholesky-based QR factorization algorithm
- We will compare both the theoretical cost and performance

\[
T_{2D \text{ Householder QR}} = \mathcal{O} \left( n \log P \cdot \alpha + \frac{mn}{\sqrt{P}} \cdot \beta \right)
\]

\[
T_{CAQR-HR} = \mathcal{O} \left( \sqrt{P} \log^2 P \cdot \alpha + \frac{mn}{\sqrt{P}} \cdot \beta \right)
\]

\[
T_{CA-\text{CholeskyQR2}} = \mathcal{O} \left( \left( \frac{Pn}{m} \right)^{\frac{2}{3}} \log P \cdot \alpha + \left( \frac{n^2 m}{P} \right)^{\frac{2}{3}} \cdot \beta \right)
\]
Cholesky-QR algorithm

\[
\begin{align*}
[Q, R] & \leftarrow \text{CholeskyQR}(A) \\
B & \leftarrow A^T A \\
R & \leftarrow \text{Cholesky}(B) \\
Q & \leftarrow AR^{-1}
\end{align*}
\]

- \(A^T A = R^T Q^T QR = R^T R\) if \(Q\) is orthogonal
- Both Cholesky and triangular solve are backwards stable, yet sensitive to conditioning of \(B\)
- Cholesky can break down due to numerical error, losing positive-definiteness
- CholeskyQR is not stable, deviation from orthogonality of computed \(Q\) is \(O(\kappa(A)^2 \cdot \epsilon)\), where \(\epsilon\) is machine epsilon
### Cholesky-QR2 algorithm

\[
[Q, R] \leftarrow \text{CholeskyQR2} (A)
\]

\[
Q_1, R_1 \leftarrow \text{CholeskyQR} (A) \\
Q, R_2 \leftarrow \text{CholeskyQR} (Q_1) \\
R \leftarrow R_2 R_1
\]

- By performing CholeskyQR 2x, the residual and deviation from orthogonality are \( \mathcal{O} (\epsilon) \) if \( \kappa (A) = \mathcal{O} \left( \frac{1}{\sqrt{\epsilon}} \right) \)

- Proposed as a replacement for TSQR for tall-and-skinny matrices
  - Lower theoretical communication cost by \( \mathcal{O} (\log P) \), better performance, and simpler implementation
  - TSQR is unconditionally stable

---

\(^3\) Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 .."

\(^4\) Y. Yamamoto et al., "CholeskyQR2: A communication-avoiding algorithm"
1D CholeskyQR2

$A^T = [A_1^T, A_2^T, A_3^T, A_4^T, A_5^T, A_6^T, A_7^T, A_8^T]$ = $B = A^T A$

$R = \text{Chol}(B)$
$Q = AR^{-1}$
1D CholeskyQR2

\[ T_{\text{CholeskyQR2\_1D}} (m, n, P) = \mathcal{O} \left( \log P \cdot \alpha + n^2 \cdot \beta + \left( \frac{n^2m}{P} + n^3 \right) \cdot \gamma \right) \]
**Figure:** 3D algorithm for square matrix multiplication

$$C = AB$$

1. Bersten 1989, "Communication-efficient matrix multiplication on hypercubes"
2. Aggarwal, Chandra, Snir 1990, "Communication complexity of PRAMs"
3. Agarwal et al. 1995, "A three-dimensional approach to parallel matrix multiplication"
Figure: 3D algorithm for square matrix multiplication

\[ T_{3D-MM}(n, P) = O \left( \log P \cdot \alpha + \frac{n^2}{P^{2/3}} \cdot \beta + \frac{n^3}{P} \cdot \gamma \right) \]

\[ C = AB \]

1. Bersten 1989, "Communication-efficient matrix multiplication on hypercubes"
2. Aggarwal, Chandra, Snir 1990, "Communication complexity of PRAMs"
3. Agarwal et al. 1995, "A three-dimensional approach to parallel matrix multiplication"
3D recursive CholeskyInverse

\[
\begin{bmatrix} L, L^{-1} \end{bmatrix} \leftarrow \text{CholeskyInverse}(A)
\]

\[
\begin{bmatrix} L_{11}, L_{11}^{-1} \end{bmatrix} \leftarrow \text{CholeskyInverse}(A_{11})
\]

\[L_{21} \leftarrow A_{21}L_{11}^{-T}\]

\[
\begin{bmatrix} L_{22}, L_{22}^{-1} \end{bmatrix} \leftarrow \text{CholeskyInverse}(A_{22} - L_{21}L_{21}^{T})
\]

\[L_{21}^{-1} \leftarrow -L_{22}^{-1}L_{21}L_{11}^{-1}\]

\[
T_{\text{CholeskyInverse3D}}(n, P) = 2T^{\alpha-\beta} \left(\frac{n}{2}, P\right) + \mathcal{O}(3D \text{ Matrix Multiplication})
\]
### 3D recursive CholeskyInverse

\[
\begin{bmatrix} L_{11} & L_{11}^{-1} \end{bmatrix} \leftarrow \text{CholeskyInverse}(A)
\]

\[
L_{21} \leftarrow A_{21} L_{11}^{-T}
\]

\[
\begin{bmatrix} L_{22} & L_{22}^{-1} \end{bmatrix} \leftarrow \text{CholeskyInverse}(A_{22} - L_{21} L_{21}^{-T})
\]

\[
L_{21}^{-1} \leftarrow -L_{22}^{-1} L_{21} L_{11}^{-1}
\]

\[
T_{\text{CholeskyInverse3D}}(n, P) = 2 T^{\alpha - \beta} \left( \frac{n}{2}, P \right) + \mathcal{O}(3D \text{ Matrix Multiplication})
\]

\[
T_{\text{CholeskyInverse3D}}(n, P) = \mathcal{O} \left( P^{2 \frac{2}{3}} \log P \cdot \alpha + \frac{n^2}{P^{2 \frac{2}{3}}} \cdot \beta + \frac{n^3}{P} \cdot \gamma \right)
\]
TRSM3D is an option to reduce computation cost by 2 at the highest levels of recursion.

- Skip two matrix multiplications by not obtaining $L_{21}^{-1}$.
- Utilize diagonally inverted blocks and iterate along column panels.
- Two 3D Matrix Multiplications per iteration to solve for panel and update trailing matrix.
Figure: $A^T A$ over a tunable $c \times d \times c$ processor grid
Figure: Start with a tunable $c \times d \times c$ processor grid
Figure: Broadcast columns of $A$

Cost: $2 \log_2 c \cdot \alpha + \frac{2mn}{dc} \cdot \beta$
Figure: Reduce contiguous groups of size $c$

Cost: $2 \log_2 c \cdot \alpha + \frac{2n^2}{c^2} \cdot \beta + \frac{n^2}{c^2} \cdot \gamma$
Figure: Allreduce alternating groups of size $\frac{d}{c}$

Cost: $2 \log_2 \frac{d}{c} \cdot \alpha + \frac{2n^2}{c^2} \cdot \beta + \frac{n^2}{c^2} \cdot \gamma$
Figure: Broadcast missing pieces of $B$ along depth

Cost: $2 \log_2 c \cdot \alpha + \frac{2n^2}{c^2} \cdot \beta$
Figure: $\frac{d}{c}$ simultaneous 3D CholeskyInverse on cubes of dimension $c$

Cost: $\mathcal{O} \left( c^2 \log c^3 \cdot \alpha + \frac{n^2}{c^2} \cdot \beta + \frac{n^3}{c^3} \cdot \gamma \right)$
Figure: \( \frac{d}{c} \) simultaneous 3D matrix multiplication or TRSM on cubes of dimension \( c \)

\[
\begin{align*}
Q &= AR^{-1} \\
\text{Cost: } O &\left(2 \log_2 c^3 \cdot \alpha + \left(\frac{4mn}{dc} + \frac{n^2 + nc}{c^2}ight) \cdot \beta + \frac{n^2m}{c^2d} \cdot \gamma\right)
\end{align*}
\]
Figure: Tunable Cholesky-QR2 Algorithm

\[ B = A^T A \]

Broadcast columns
- AllReduce contiguous groups of size \( c \)
- AllReduce alternating groups of size \( d/c \)
- Broadcast along depth

\[ B = R^T R \]

\[ Q = AR^{-1} \]

d/c simultaneous 3D Cholesky Inverses on cubes of dimension \( c \)
d/c simultaneous 3D Matrix Multiplications or TRSM on cubes of dimension \( c \)

Edward Hutter
Parallel 3D Cholesky-QR2
By setting \( \frac{m}{d} = \frac{n}{c} \) and enforcing \( P = c^2 d \), we are able to solve for the optimal grid and minimize communication.

<table>
<thead>
<tr>
<th></th>
<th>1D CholeskyQR2</th>
<th>2D Householder QR</th>
<th>2D CAQR-HR</th>
<th>optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td># of messages</td>
<td>( \mathcal{O} (\log P) )</td>
<td>( n \log P )</td>
<td>( \sqrt{P} \log P )</td>
<td>( \mathcal{O} \left( \left( \frac{P n}{m} \right)^{\frac{2}{3}} \log P \right) )</td>
</tr>
<tr>
<td># of words</td>
<td>( \mathcal{O} \left( n^2 \right) )</td>
<td>( \mathcal{O} \left( \frac{mn}{\sqrt{P}} \right) )</td>
<td>( \mathcal{O} \left( \frac{mn}{\sqrt{P}} \right) )</td>
<td>( \mathcal{O} \left( \left( \frac{n^2 m}{P} \right)^{\frac{2}{3}} \right) )</td>
</tr>
<tr>
<td># of flops</td>
<td>( \mathcal{O} \left( \frac{n^2 m}{P} + n^3 \right) )</td>
<td>( \mathcal{O} \left( \frac{mn^2}{P} \right) )</td>
<td>( \mathcal{O} \left( \frac{mn^2}{P} \right) )</td>
<td>( \mathcal{O} \left( \frac{n^2 m}{P} \right) )</td>
</tr>
<tr>
<td>Memory footprint</td>
<td>( \mathcal{O} \left( \frac{mn}{P} + n^2 \right) )</td>
<td>( \mathcal{O} \left( \frac{mn}{P} \right) )</td>
<td>( \mathcal{O} \left( \frac{mn}{P} \right) )</td>
<td>( \mathcal{O} \left( \left( \frac{n^2 m}{P} \right)^{\frac{2}{3}} \right) )</td>
</tr>
</tbody>
</table>
Implementation and performance testing

- All code written from scratch in modern C++ and MPI
- This is a first implementation
  - TRSM not used in results
  - No overlap in computation and communication
  - No topology-aware mapping
  - No threading except for multi-threaded BLAS
- For each processor count and matrix size:
  - Scalapack PGEQRF was tuned over block sizes and processor grid dimensions
  - CA-CholeskyQR2 was tuned over range of 3D grid dimensions
- Best performance numbers were chosen to compare
Cray XC40 system at Argonne Leadership Computing Facility

- Compute nodes: Intel Knights Landing
  - Each node is a single Xeon Phi chip with 64 cores, 16 GB MCDRAM, 192GB DDR4
  - Each core supports 4 hardware threads, up to 256 threads per node

- Interconnect topology: Cray Aries dual-place Dragonfly with 10 groups

Edward Hutter

Parallel 3D Cholesky-QR2
**Architecture details**

- **Cray XC40 system at Argonne Leadership Computing Facility**
  - Compute nodes: Intel Knights Landing
    - Each node is a single Xeon Phi chip with 64 cores, 16 GB MCDRAM, 192GB DDR4
    - Each core supports 4 hardware threads, up to 256 threads per node
  - Interconnect topology: Cray Aries dual-place Dragonfly with 10 groups
- **Our configuration:**
  - 16 MPI processes per node, 4 OpenMP threads per process, 2 hyperthreads per core
  - MKL 2018 libraries used for BLAS and LAPACK
  - MKL 2018 SCALAPACK library used as comparison for benchmarking
  - \((d, c)\) pairs for \(c \times d \times c\) processor grid given for each data point
Weak scaling on Theta (XC40)

Cholesky-QR2, \(m = \#\text{nodes} \times 256, n = 128\)
ScaLAPACK QR, \(m = \#\text{nodes} \times 256, n = 128\)
Weak scaling on Theta (XC40)

- **Cholesky-QR2**, $m=\text{#nodes} \times 128$, $n=512$
- **ScaLAPACK QR**, $m=\text{#nodes} \times 128$, $n=512$

Edward Hutter  Parallel 3D Cholesky-QR2
Weak scaling on Theta (XC40)

- Cholesky-QR2, $m = \text{#nodes} \times 512$, $n = 1024$
- ScaLAPACK QR, $m = \text{#nodes} \times 512$, $n = 1024$

Edward Hutter
Parallel 3D Cholesky-QR2
Strong scaling on Theta (XC40)

Cholesky-QR2, $m=16384$, $n=256$

ScalAPACK QR, $m=16384$, $n=256$

Edward Hutter  Parallel 3D Cholesky-QR2
CholeskyQR2 is outperforming Scalapack on rectangular matrices.

Scalapack is outperforming CholeskyQR2 on tall-and-skinny and near-square matrices.

Both do not show strong scaling to 1024 nodes.

Flop cost is a concern.

- HouseHolder QR performs $2mn^2 - \frac{2n^3}{3}$ flops
- CholeskyQR2 performs $4mn^2 + \frac{5n^3}{3}$ flops

Communication improvement of $O(P^{\frac{1}{6}})$

- Perhaps $P$ is not large enough to make a difference?
Future work

Numerical library integration is our focus. Much work remains
- optimize and tune the code, more large scale numerical tests to better understand performance
- evaluate more tunable grid shapes to further analyze patterns in performance
- develop memory tunable variants for machines without enough memory
- reduce the flop cost and improve the stability

MS87 talk at 3:45 on fixing the stability of CholeskyQR2

Acknowledgement: Argonne Leadership Computing Facility
The advantage of using a tunable grid lies in the ability to frame the shape of the grid around the shape of rectangular $m \times n$ matrix $A$. Optimal communication can be attained by ensuring that the grid perfectly fits the dimensions of $A$, or that the dimensions of the grid are proportional to the dimensions of the matrix. We derive the cost for the optimal ratio $\frac{m}{d} = \frac{n}{c}$ below. Using equation $P = c^2 d$ and $\frac{m}{d} = \frac{n}{c}$, solve for $d, c$ in terms of $m, n, P$.

Solving the system of equations yields $c = (\frac{Pn}{m})^{\frac{1}{3}}$, $d = (\frac{Pm^2}{n^2})^{\frac{1}{3}}$. We can plug these values into the cost of CholeskyQR2_Tunable to find the optimal cost.

$$T_{\text{CholeskyQR2_Tunable}}^{\alpha-\beta} \left( m, n, \left(\frac{Pn}{m}\right)^{\frac{1}{3}}, \left(\frac{Pm^2}{n^2}\right)^{\frac{1}{3}} \right) = O \left( \left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P \cdot \alpha \right)$$

$$+ \left(\frac{Pm^2}{n^2}\right)^{\frac{1}{3}} \left(\frac{Pn}{m}\right)^{\frac{2}{3}} \beta + \left(\frac{n^2m}{P}\right)^{\frac{2}{3}} + \left(\frac{n^2m}{P}\right)^{\frac{2}{3}} \cdot \gamma \right)$$

$$= O \left( \left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P \cdot \alpha + \left(\frac{n^2m}{P}\right)^{\frac{2}{3}} \beta + \left(\frac{n^2m}{P}\right)^{\frac{2}{3}} \cdot \gamma \right)$$

<table>
<thead>
<tr>
<th>Grid shape</th>
<th>Metric</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td># of messages</td>
<td>$O \left( \left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P \right)$</td>
</tr>
<tr>
<td></td>
<td># of words</td>
<td>$O \left( \left(\frac{n^2m}{P}\right)^{\frac{2}{3}} \right)$</td>
</tr>
<tr>
<td></td>
<td># of flops</td>
<td>$O \left( \frac{n^2m}{P} \right)$</td>
</tr>
<tr>
<td></td>
<td>Memory footprint</td>
<td>$O \left( \left(\frac{n^2m}{P}\right)^{\frac{2}{3}} \right)$</td>
</tr>
</tbody>
</table>