QR Factorization over Tunable Processor Grids

Edward Hutter

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

July 12, 2017
Abstract

• What:
  ▶ new distributed-memory QR factorization algorithm
  ▶ extends the existing Cholesky-QR2 algorithm to matrices of an arbitrary size and shape
  ▶ takes advantage of a tunable processor grid to provably reduce communication

• Why:
  ▶ practical
  ▶ flexible
  ▶ achieves minimal communication

• Who:
  ▶ Edward Hutter and Edgar Solomonik
Communication model

- Let's define a communication model
  \[\alpha - \beta - \gamma\text{ model}\]

- This model's advantages and disadvantages lie in its simplicity
  - Allows each processor to send and/or receive one message at a time
  - Not "blocking", not synchronous, but blocks to completion
  - No overlap in communication and computation
Key terms

- Lets explore some key terms
  - Synchronization (Latency) cost - $\alpha$
    - Elapsed time between request and reception of the first byte of data in a message
  - Horizontal (Vertical) bandwidth cost - $\beta$
    - Cost to move a byte of data among processors over a network (between levels in a memory hierarchy)
  - Horizontal (Vertical) communication cost - $c_1 \cdot \alpha + c_2 \cdot \beta$
    - Linear combination of synchronization (latency) cost and horizontal (vertical) bandwidth cost
  - Flop cost - $\gamma$
    - Cost to compute a floating point operation with register-resident data
  - Critical path - $c_1 \cdot \alpha + c_2 \cdot \beta + c_3 \cdot \gamma$
    - Most expensive chain of dependent execution in the DAG representing the parallel algorithm
  - Stability
    - Ability of an algorithm to suppress input and approximation errors
Why the need to redesign existing numerical linear algebra algorithms?

- Trend in modern machines shows a widening gap between speedups in flop rate, bandwidth, and latency: $\alpha \gg \beta \gg \gamma$

- Algorithm performance on modern machines is dependent on the movement of data

- Therefore, existing algorithms not achieving minimal communication will become less performant over time

- In response, a new field of research has emerged with the goal to redesign existing algorithms from the bottom up in order to achieve minimal communication along the critical path.
Problem Definition

\[ A = QR \]

- \( A \) is dense \( m \times n \)
- \( Q \) is \( m \times n \) and orthogonal
- \( R \) is upper triangular \( n \times n \)
Motivation

- Efficient QR factorization algorithms are needed in many applications within the fields of scientific computing and machine learning.
- QR factorization is a key step to solving least squares problems.
Competing QR factorization algorithms

- **Parallel Householder**
  - Restriction: requires $n \log_2 P$ synchronizations between processors

- **Cholesky-QR2**
  - Restriction: memory footprint of order $O(n^2)$ for $m \times n$ matrices

- **TSQR**
  - Restriction: although unconditionally stable and 2x as communication efficient as Cholesky-QR2, scales only for tall-and-skinny matrices

- **Other recent theoretical QR factorization algorithms**
  - Restriction: Not (yet) implemented due to impracticality

**Bottom line:** Our new tunable algorithm takes the best qualities from the above algorithms, achieving practicality, flexibility, and minimal communication for matrices with a condition number of order $O\left(\frac{1}{\sqrt{\epsilon}}\right)$
Cholesky-QR2 Algorithm

Algorithm 2.1 \([Q, R] \leftarrow \text{CholeskyQR} (A, m, n)\)

Require: \(A\) is \(m \times n\)

1: \(W \leftarrow \text{seq-Syrk} (A, m, n)\)
2: \(R^T \leftarrow \text{seq-Cholesky} (W, n)\)
3: \(Q \leftarrow \text{seq-MM} (A, R^{-1}, m, n, n)\)

Ensure: \(A = QR\), where \(Q\) is \(m \times n\) orthogonal, \(R\) is \(n \times n\) upper triangular

Algorithm 2.2 \([Q, R] \leftarrow \text{CholeskyQR2} (A, m, n)\)

Require: \(A\) is \(m \times n\)

1: \(Q_1, R_1 \leftarrow \text{CholeskyQR} (A)\)
2: \(Q, R_2 \leftarrow \text{CholeskyQR} (Q_1)\)
3: \(R \leftarrow \text{seq-MM}(R_2, R_1, n, n, n)\)

Ensure: \(A = QR\), where \(Q\) is \(m \times n\) orthogonal, \(R\) is \(n \times n\) upper triangular
To devise an efficient parallel CholeskyQR2 algorithm, we need efficient algorithms for:

- Collective communication
- Matrix multiplication
- Cholesky factorization

Some useful costs

\[
T_{\text{seq-MM}}^{\alpha - \beta} (m, n, k) = \mathcal{O}(mnk) \cdot \gamma \\
T_{\text{seq-Subtract}}^{\alpha - \beta} (m, n) = \mathcal{O}(mn) \cdot \gamma \\
T_{\text{seq-Syrk}}^{\alpha - \beta} (m, n) = \mathcal{O}(mn^2) \cdot \gamma \\
T_{\text{seq-Cholesky}}^{\alpha - \beta} (n) = \mathcal{O}(n^3) \cdot \gamma \\
T_{\text{seq-Trilv}}^{\alpha - \beta} (n) = \mathcal{O}(n^3) \cdot \gamma
\]

We also want to define a unit-step function as follows: \( \delta (x) = \begin{cases} 
0 & \text{if } x \leq 1 \\
1 & \text{if } x > 1
\end{cases} \)
Collective communication - Broadcast

Figure: Broadcast

\[ T_{\text{Bcast}}^{\alpha-\beta}(n, P) = 2 \log_2 P \cdot \alpha + 2n\delta(P) \cdot \beta \]
Collective communication - AllReduce

Figure: AllReduce

$$T^{\alpha-\beta}_{\text{AllReduce}}(n, P) = 2 \log_2 P \cdot \alpha + 2n\delta(P) \cdot \beta + n\delta(P) \cdot \gamma$$
Collective communication - AllGather

$$T_{\text{AllGather}}^{\alpha-\beta} (n, P) = \log_2 P \cdot \alpha + n\delta(P) \cdot \beta$$
Problem Definition

\[ C = AB \]

- \( A \) is dense \( m \times k \)
- \( B \) is dense \( k \times n \)
- \( C \) is dense \( m \times n \)

\[
C[i,j] = \sum_{k=1}^{n} A[i,k]B[k,j]
\]

\[
\vec{C}_i = \sum_{i=1}^{n} A\vec{B}_i
\]
Figure: 2D Parallel Matrix Multiplication

\[
P_{\frac{1}{2}} = P_{00} P_{01} P_{02} P_{03} P_{10} P_{11} P_{12} P_{13} P_{20} P_{21} P_{22} P_{23} P_{30} P_{31} P_{32} P_{33}
\]

\[
A_{00} A_{01} A_{02} A_{03} A_{10} A_{11} A_{12} A_{13} A_{20} A_{21} A_{22} A_{23} A_{30} A_{31} A_{32} A_{33}
\]

\[
B_{00} B_{01} B_{02} B_{03} B_{10} B_{11} B_{12} B_{13} B_{20} B_{21} B_{22} B_{23} B_{30} B_{31} B_{32} B_{33}
\]

\[
C_{21} = A_{23} B_{31} + A_{20} B_{01} + A_{21} B_{11} + A_{22} B_{21}
\]

How many partial products? \( p^{\frac{1}{2}} \)

If processor grid was size \( k \times k \) \( \rightarrow \) \( k \) partial products

\[
C_{ij} = A_{i0} B_{0j} + A_{i1} B_{1j} + A_{i2} B_{2j} + A_{i3} B_{3j}
\]
Figure: Matrix Multiplication 3D Algorithm

C = AB

Broadcast across rows
Broadcast along columns
AllReduce along depth

Cyclic distribution
Algorithm 2.3 $[C] \leftarrow \text{MatrixMultiplication3D}(A, B, m, n, k, \Pi, i, j, k)$

Require: $\Pi$ has $P$ processors arranged in a 3D grid. Matrices $A$ and $B$ are replicated on $\Pi[:, :, k], \forall k \in [0, P^{\frac{1}{3}} - 1]$. Each processor $\Pi[i, j, k]$ owns a cyclic partition of $m \times n$ matrix $A$ and $n \times k$ matrix $B$. We call these local matrices $A_{ij}$ and $B_{ij}$, respectively. These matrix partitions are condensed into 1D row-major arrays of size $\frac{mn}{P^{\frac{2}{3}}}$ and $\frac{nk}{P^{\frac{2}{3}}}$, respectively. Let $X, Y,$ and $Z$ be temporary arrays with the same distribution as $A$ and $B$.

1: Bcast($A_{kj}, X_{ij}, k, \Pi[:, :, k]$)  
   $\triangleright$ Broadcast from root $k$ across row $i$

2: Bcast($B_{ik}, Y_{ij}, k, \Pi[i, :, k]$)  
   $\triangleright$ Broadcast from root $k$ along column $j$

3: $Z_{ij} \leftarrow \text{seq-MM}(X_{ij}, Y_{ij}, \frac{n}{P^{\frac{1}{3}}})$

4: Allreduce($Z_{ij}, C_{ij}, \Pi[i, j, :]$)  
   $\triangleright$ AllReduce along the depth of 3D grid

Ensure: $C = AB$, where $C$ is $m \times k$ and distributed the same way as $A$ and $B$. 
Table: Costs of MatrixMultiplication3D.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2 \log_2 P \frac{1}{3} \cdot \alpha + \frac{2mn\delta(P)}{P^2} \cdot \beta$</td>
</tr>
<tr>
<td>2</td>
<td>$2 \log_2 P \frac{1}{3} \cdot \alpha + \frac{2nk\delta(P)}{P^2} \cdot \beta$</td>
</tr>
<tr>
<td>3</td>
<td>$O\left(\frac{mnk}{P}\right) \cdot \gamma$</td>
</tr>
<tr>
<td>4</td>
<td>$2 \log_2 P \frac{1}{3} \cdot \alpha + \frac{2mk\delta(P)}{P^2} \cdot \beta + \frac{mk\delta(P)}{P^2} \cdot \gamma$</td>
</tr>
</tbody>
</table>

$$T_{\text{MatrixMultiplication3D}}^\alpha - \beta (m, n, k, P) = 6 \log_2 P \frac{1}{3} \cdot \alpha + \frac{(2mn + 2nk + 2mk) \delta(P)}{P^2} \cdot \beta$$

$$+ O\left(\frac{mk\delta(P)}{P^3} + \frac{mnk}{P}\right) \cdot \gamma$$

$$= O\left(\log P \cdot \alpha + \frac{(mn + nk + mk) \delta(P)}{P^2} \cdot \beta + \frac{mnk}{P} \cdot \gamma\right)$$
Cholesky Factorization

Problem Definition

\[ A = LL^T \]

- \( A \) is dense \( n \times n \), symmetric and positive definite
- \( L \) is lower triangular \( n \times n \)
Derivation of recursive Cholesky Factorization

\[
\begin{bmatrix}
A_{11} & A_{21} \\
A_{21} & A_{22}
\end{bmatrix}
= \begin{bmatrix}
L_{11} & L_{21} \\
L_{21} & L_{22}
\end{bmatrix}
\begin{bmatrix}
L_{11}^T & L_{21}^T \\
L_{21}^T & L_{22}
\end{bmatrix}
\]

\[
A_{11} = L_{11}L_{11}^T
\]

\[
A_{21} = L_{21}L_{11}^T
\]

\[
A_{22} = L_{21}L_{21}^T + L_{22}L_{22}^T
\]

\[
L_{11} = \text{Cholesky}(A_{11})
\]

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

\[
L_{22} = \text{Cholesky}(A_{22} - L_{21}L_{21}^T)
\]

\[
\begin{bmatrix}
L_{11} & 0 \\
L_{21} & L_{22}
\end{bmatrix}
\begin{bmatrix}
L_{11}^{-1} & L_{12}^{-1} \\
L_{21} & L_{22}
\end{bmatrix}
= \begin{bmatrix}
I_n & 0 \\
0 & I_n
\end{bmatrix}
\]

\[
L_{11}^{-1} = (L_{11})^{-1}
\]

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

\[
L_{22}^{-1} = (L_{22})^{-1}
\]

\[
\begin{bmatrix}
L_{11} & L_{11}^{-1} \\
L_{21} & L_{22}
\end{bmatrix}
\begin{bmatrix}
L_{22}^{-1} & 0 \\
-L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1}
\end{bmatrix}
= \text{CholeskyInverse}(A)
\]

\[
L = \begin{bmatrix}
L_{11} & 0 \\
L_{21} & L_{22}
\end{bmatrix}
\]

\[
L^{-1} = \begin{bmatrix}
L_{11}^{-1} & 0 \\
-L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1}
\end{bmatrix}
\]
Figure: Need for transpose with cyclic distribution
Figure: CholeskyFactorization3D Algorithm
Algorithm 2.4 \( [L, L^{-1}] \leftarrow \text{CholeskyFactorization3D} (A, n, n_o, \Pi, i, j, k) \)

**Require:** \( \Pi \) has \( P \) processors arranged in a 3D grid. Matrix \( A \) is of dimension \( n \), symmetric, and positive definite. \( A \) is replicated on \( \Pi[\cdot, \cdot, k], \forall k \in [0, P \frac{1}{3} - 1] \). Each processor \( \Pi[i, j, k] \) owns a cyclic partition of \( A \) known as \( A_{ij} \). \( A_{ij} \) is packed into a 1D array of size \( \left( \frac{n}{2} \right) \left( \frac{n}{2} + 1 \right) \). Let \( n_o \) be the matrix dimension in which we call the base case. Let \( \text{TopLeft} \) and \( \text{BottomRight} \) be lower triangular portions of dimension \( \frac{n}{2} \) square submatrices in the upper-left quadrant and lower-right quadrants of a matrix, respectively. Let \( \text{BottomLeft} \) be the square submatrix in the lower-left quadrant. Let \( T, W, X, Y, \) and \( Z \) be temporary arrays, distributed the same way as \( A \).

1: if \( n = n_o \) then
2: \( \text{AllGather} (A_{ij}, T_{ij}, \Pi[\cdot, \cdot, k]) \)
3: \( L_{ij} \leftarrow \text{seq-Cholesky} (T_{ij}, n) \)
4: \( L_{ij}^{-1} \leftarrow \text{seq-Trilv} (L_{ij}, n) \)
5: else
6: \( L[\text{TopLeft}], L^{-1}[\text{TopLeft}] \leftarrow \text{CholeskyFactorization3D} \left( A[\text{TopLeft}], \frac{n}{2}, n_o, \Pi, i, j, k \right) \)
7: \( W_{ij} \leftarrow \text{Transpose} \left( L_{ij}^{-1}[\text{TopLeft}], \Pi[j, i, k] \right) \)
8: \( L[\text{BottomLeft}] \leftarrow \text{MatrixMultiply3D} \left( A[\text{BottomLeft}], W^T, \frac{n}{2}, \frac{n}{2}, \frac{n}{2}, \Pi, i, j, k \right) \)
9: \( X_{ij} \leftarrow \text{Transpose} \left( L_{ij}[\text{BottomLeft}], \Pi[j, i, k] \right) \)
10: \( Y \leftarrow \text{MatrixMultiply3D} \left( L[\text{BottomLeft}], X^T, \frac{n}{2}, \frac{n}{2}, \frac{n}{2}, \Pi, i, j, k \right) \)
11: \( Z_{ij} \leftarrow \text{seq-Subtract} \left( A_{ij}[\text{BottomRight}], Y_{ij}, \frac{n}{2} \right) \)
12: \( L[\text{BottomRight}], L^{-1}[\text{BottomRight}] \leftarrow \text{CholeskyFactorization3D} \left( Z, \frac{n}{2}, n_o, \Pi, i, j, k \right) \)
13: \( Y \leftarrow \text{MatrixMultiply3D} \left( L[\text{BottomLeft}], L^{-1}[\text{TopLeft}], \frac{n}{2}, \frac{n}{2}, \frac{n}{2}, \Pi, i, j, k \right) \)
14: \( W \leftarrow (-1) \cdot L^{-1}[\text{BottomRight}] \)
15: \( L^{-1}[\text{BottomLeft}] \leftarrow \text{MatrixMultiply3D} \left( W, Y, \frac{n}{2}, \frac{n}{2}, \frac{n}{2}, \Pi, i, j, k \right) \)

**Ensure:** \( A = LL^T, L^{-1} = (L)^{-1} \), where matrices \( L \) and \( L^{-1} \) are distributed the same way as \( A \).
Table: Costs of CholeskyFactorization3D.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$\log_2 P \frac{2}{3} \cdot \alpha + n_o^2 \delta (P) \cdot \beta$</td>
</tr>
<tr>
<td>3</td>
<td>$\mathcal{O} \left( \frac{n_o^3}{P} \right) \cdot \gamma$</td>
</tr>
<tr>
<td>4</td>
<td>$\mathcal{O} \left( \frac{n_o^3}{P} \right) \cdot \gamma$</td>
</tr>
<tr>
<td>7</td>
<td>$\delta (P) \cdot \alpha + \frac{n}{2} \frac{(n^2 + 1) \delta (P)}{2} \cdot \beta$</td>
</tr>
<tr>
<td>8</td>
<td>$2 \log_2 P \cdot \alpha + \frac{(5n^2 + 2n) \delta (P)}{4P \frac{2}{3}} \cdot \beta + \mathcal{O} \left( \frac{n^3}{P} \right) \cdot \gamma$</td>
</tr>
<tr>
<td>9</td>
<td>$\delta (P) \cdot \alpha + \frac{n^2 \delta (P)}{4P \frac{2}{3}} \cdot \beta$</td>
</tr>
<tr>
<td>10</td>
<td>$2 \log_2 P \cdot \alpha + \frac{3n^2 \delta (P)}{2P \frac{2}{3}} \cdot \beta + \mathcal{O} \left( \frac{n^3}{P} \right) \cdot \gamma$</td>
</tr>
<tr>
<td>11</td>
<td>$\mathcal{O} \left( \frac{n^2}{P \frac{2}{3}} \right) \cdot \gamma$</td>
</tr>
<tr>
<td>13</td>
<td>$2 \log_2 P \cdot \alpha + \frac{(5n^2 + 2n) \delta (P)}{4P \frac{2}{3}} \cdot \beta + \mathcal{O} \left( \frac{n^3}{P} \right) \cdot \gamma$</td>
</tr>
<tr>
<td>14</td>
<td>$\mathcal{O} \left( \frac{n^2}{P \frac{2}{3}} \right) \cdot \gamma$</td>
</tr>
<tr>
<td>15</td>
<td>$2 \log_2 P \cdot \alpha + \frac{(5n^2 + 2n) \delta (P)}{4P \frac{2}{3}} \cdot \beta + \mathcal{O} \left( \frac{n^3}{P} \right) \cdot \gamma$</td>
</tr>
</tbody>
</table>
Further analysis of CholeskyFactorization3D

$$T_{\text{CholeskyBaseCase}}^{{\alpha - \beta}} (n_0, P) = \log_2 P \frac{2}{3} \cdot \alpha + n_0^2 \delta (P) \cdot \beta + \mathcal{O} \left( n_0^3 \right) \cdot \gamma$$

$$= \mathcal{O} \left( \log P \cdot \alpha + n_0^2 \delta (P) \cdot \beta + n_0^3 \cdot \gamma \right)$$

Choice of $n_0$ depends on the non-recursive communication cost. Because $\frac{n}{2^z} = n_0$, our algorithm must compute $\frac{n}{n_0}$ AllGathers.

$$T_{\text{CholeskyFactorization3D}}^{{\alpha - \beta}} (n, P) = 2 T_{\text{CholeskyFactorization3D}}^{{\alpha - \beta}} \left( \frac{n}{2}, P \right) + (8 \log_2 P + 2 \delta (P)) \cdot \alpha$$

$$+ \frac{22.5 n^2 + 7 n}{4 P^2} \cdot \beta + \mathcal{O} \left( \frac{n^3}{P} \right) \cdot \gamma$$

$$= 2 T_{\text{CholeskyFactorization3D}}^{{\alpha - \beta}} \left( \frac{n}{2}, P \right) + \mathcal{O} \left( \log P \cdot \alpha + \frac{n^2 \delta (P)}{P^2} \cdot \beta + \frac{n^3}{P} \cdot \gamma \right)$$

$$= 2^z T_{\text{CholeskyBaseCase}}^{{\alpha - \beta}} (n_0, P) + \sum_{q=0}^{z-1} 2^q \cdot \mathcal{O} \left( \log P \cdot \alpha + \frac{\left( \frac{n}{2^q} \right)^2 \delta (P)}{P^2} \cdot \beta + \frac{\left( \frac{n}{2^q} \right)^3}{P} \cdot \gamma \right)$$

$$= \mathcal{O} \left( \frac{n \log P}{n_0} \cdot \alpha + n n_0 \delta (P) \cdot \beta + n n_0^2 \cdot \gamma \right) + \mathcal{O} \left( \frac{n \log P}{n_0} \cdot \alpha + \frac{n^2 \delta (P)}{P^2} \cdot \beta + \frac{n^3}{P} \cdot \gamma \right)$$

Choice of $\frac{n}{n_0}$ creates a tradeoff between the synchronization cost and the communication cost. We elect to match the communication cost at the expense of an increase in synchronization, giving the relation $n_o = \frac{n}{P^{\frac{2}{3}}}$. The final cost of the 3D algorithm is the following:

$$T_{\text{CholeskyFactorization}} (n, P) = \mathcal{O} \left( P^{\frac{2}{3}} \log P \cdot \alpha + \frac{n^2 \delta (P)}{P^2} \cdot \beta + \frac{n^3}{P} \cdot \gamma \right)$$
Figure: CholeskyQR2_1D Algorithm

\[ A^T = A_1^T A_2^T A_3^T A_4^T A_5^T A_6^T A_7^T A_8^T = B_1 \]

\[ B = R^T R \]

\[ Q = AR^{-1} \]
Algorithm 2.5 \([Q, R] \leftarrow \text{CholeskyQR\_1D}(A, m, n, \Pi_1, p)\)

Require: \(\Pi_1\) has \(P\) processors arranged in a 1D grid. Each processor \(\Pi_1[p]\) owns a (cyclic) blocked partition of \(m \times n\) input matrix \(A\) known as \(A_p\). \(A_p\) is packed into a 1D array of size \(\frac{mn}{p}\), where it owns a rectangular piece of size \(\frac{m}{p} \times n\). Let \(X\) and \(Y\) be temporary arrays.

1: \(X_p \leftarrow \text{seq-Syrk}(A_p, m, n)\) \(\triangleright X_p \leftarrow A_p^T A_p\)
2: \(\text{AllReduce}(X_p, Y_p, \Pi_1)\) \(\triangleright Y \leftarrow A^T A\)
3: \(R^T \leftarrow \text{seq-Cholesky}(Y, n)\) \(\triangleright Y = R^T R\)
4: \(R^{-T} \leftarrow \text{seq-Trilnv}(R^T, n)\) \(\triangleright R^{-T} \leftarrow (R^T)^{-1}\)
5: \(Q_p \leftarrow \text{seq-MM}(A_p, R^{-1}, m, n, n)\) \(\triangleright Q \leftarrow A R^{-1}\)

Ensure: \(A = QR\), where \(Q\) is distributed the same as \(A\), \(R\) is an upper triangular matrix of dimension \(n\) owned locally by every processor and packed into a 1D array of size \(\frac{n(n+1)}{2}\).

Algorithm 2.6 \([Q, R] \leftarrow \text{CholeskyQR2\_1D}(A, m, n, \Pi_1, p)\)

Require: Same requirements as Algorithm 3.

1: \(X, Y \leftarrow \text{CholeskyQR\_1D}(A, m, n, \Pi_1, p)\)
2: \(Q, Z \leftarrow \text{CholeskyQR\_1D}(X, m, n, \Pi_1, p)\)
3: \(R \leftarrow \text{seq-MM}(Z, Y, n, n, n)\)

Ensure: Same requirements as Algorithm 2.5.
Cost analysis for CholeskyQR_1D

Table: Costs of CholeskyQR_1D.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\mathcal{O} \left( \frac{n^2 m}{P} \right) \cdot \gamma$</td>
</tr>
<tr>
<td>2</td>
<td>$2 \log_2 P \cdot \alpha + n^2 \delta(P) \cdot \beta + \frac{n^2}{2} \cdot \gamma$</td>
</tr>
<tr>
<td>3</td>
<td>$\mathcal{O} \left( n^3 \right) \cdot \gamma$</td>
</tr>
<tr>
<td>4</td>
<td>$\mathcal{O} \left( n^3 \right) \cdot \gamma$</td>
</tr>
<tr>
<td>5</td>
<td>$\mathcal{O} \left( \frac{n^2 m}{P} \right) \cdot \gamma$</td>
</tr>
</tbody>
</table>

$T_{\text{CholeskyQR_1D}}^{\alpha - \beta} (m, n, P) = 2 \log_2 P \cdot \alpha + n^2 \delta(P) \cdot \beta + \mathcal{O} \left( \frac{n^2 m}{P} + n^3 \right) \cdot \gamma$  \hspace{1cm} (1)

Table: Costs of CholeskyQR2_1D.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2 \log_2 P \cdot \alpha + n^2 \delta(P) \cdot \beta + \mathcal{O} \left( \frac{n^2 m}{P} + n^3 \right) \cdot \gamma$</td>
</tr>
<tr>
<td>2</td>
<td>$2 \log_2 P \cdot \alpha + n^2 \delta(P) \cdot \beta + \mathcal{O} \left( \frac{n^2 m}{P} + n^3 \right) \cdot \gamma$</td>
</tr>
<tr>
<td>3</td>
<td>$\mathcal{O} \left( n^3 \right) \cdot \gamma$</td>
</tr>
</tbody>
</table>
Further analysis for CholeskyQR\_1D

\[ T_{\text{CholeskyQR2}\_1D}^{\alpha-\beta} (m, n, P) = 4 \log_2 P \cdot \alpha + 2n^2 \delta (P) \cdot \beta + \mathcal{O} \left( \frac{n^2 m}{P} + n^3 \right) \cdot \gamma \]

\[ = \mathcal{O} \left( \log P \cdot \alpha + n^2 \delta (P) \cdot \beta + \left( \frac{n^2 m}{P} + n^3 \right) \cdot \gamma \right) \]

Varying \( \frac{m}{P} \) and \( n \) can lead to different asymptotic costs and advantages and disadvantages in practice.

**Table:** Costs of CholeskyQR2\_1D with varying block sizes.

<table>
<thead>
<tr>
<th>( \frac{m}{P} )</th>
<th>( \mathcal{O} \left( \log P \cdot \alpha + n^2 \delta (P) \cdot \beta + \frac{n^2 m}{P} \cdot \gamma \right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{m}{P} &gt; n )</td>
<td>( \mathcal{O} \left( \log P \cdot \alpha + n^2 \delta (P) \cdot \beta + \frac{n^2 m}{P} \cdot \gamma \right) )</td>
</tr>
<tr>
<td>( \frac{m}{P} \leq n )</td>
<td>( \mathcal{O} \left( \log P \cdot \alpha + n^2 \delta (P) \cdot \beta + n^3 \cdot \gamma \right) )</td>
</tr>
</tbody>
</table>

This algorithm achieves poor scalability in communication, computation, and memory footprint. Regardless of \( P \), the AllGather distributes an \( n \times n \) matrix onto each processor and as \( n \) grows the matrix won’t fit into a reasonably sized memory.
Figure: Cholesky-QR2 3D Algorithm

B = A^T A

B = R^T R

Q = AR^{-1}

3D Cholesky Factorization

3D Matrix Multiplication

Broadcast columns to get corresponding A^T

Reduce along columns

Broadcast along depth
Algorithm 2.7 \([Q, R] \leftarrow \text{CholeskyQR}_3\text{D}(A, m, n, \Pi, i, j, k)\)

Require: \(\Pi\) has \(P\) processors arranged in a 3D grid. \(A\) is \(m \times n\) and is replicated on \(\Pi[:, :, k], \forall k \in [0, P\frac{1}{3} - 1]\). Each processor \(\Pi[i, j, k]\) owns a (cyclic) blocked partition of \(A\) known as \(A_{ji}\). \(A_{ji}\) is packed into a 1D array of size \(\frac{mn}{P\frac{2}{3}}\), where it owns a rectangular piece of size \(\frac{m}{P\frac{1}{3}} \times \frac{n}{P\frac{1}{3}}\). Let \(W, X, Y, Z,\) and \(R^{-1}\) be temporary arrays distributed the same as \(A\).

1: Bcast \((A_{jk}, W_{ji}, k, \Pi[:, :, k])\)  \(\triangleright\) Broadcast from root \(k\) across row \(i\)

2: \(X_{ji} \leftarrow \text{seq-MM} \left(\frac{n}{P\frac{1}{3}}, \frac{m}{P\frac{1}{3}}, \frac{n}{P\frac{1}{3}}\right)\)

3: Reduce \((X_{ji}, Y_{ki}, k, \Pi[i, :, k])\)  \(\triangleright\) Reduce along each column to root \(k\)

4: Bcast \((Y_{ki}, Z_{ji}, k, \Pi[i, j, :])\)  \(\triangleright\) Every 2D slice owns same matrix \(B = A^T\)

5: \(R^T, R^{-T} \leftarrow \text{CholeskyFactorization3D} \left(\frac{n}{P\frac{2}{3}}, \Pi, i, j, k\right)\)

6: \(Q \leftarrow \text{MatrixMultiplication3D} \left(A, R^{-1}, m, n, n, \Pi, i, j, k\right)\)

Ensure: \(A = QR\), where \(Q\) and \(R\) are distributed the same as \(A\). \(Q\) is \(m \times n\) and \(R\) is an upper triangular matrix of dimension \(n\).

Algorithm 2.8 \([Q, R] \leftarrow \text{CholeskyQR2}_3\text{D}(A, m, n, \Pi, i, j, k)\)

Require: Same requirements as Algorithm 5.

1: \(X, Y \leftarrow \text{CholeskyQR}_3\text{D}(A, m, n, \Pi, i, j, k)\)
2: \(Q, Z \leftarrow \text{CholeskyQR}_3\text{D}(X, m, n, \Pi, i, j, k)\)
3: \(R \leftarrow \text{MatrixMultiplication3D}(Z, Y, n, n, n, \Pi, i, j, k)\)

Ensure: Same requirements as Algorithm 7.
Cost analysis of CholeskyQR_3D

Table: Costs of CholeskyQR_3D.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(2 \log_2 P \frac{1}{3} \cdot \alpha + \frac{2nm\delta(P)}{P\frac{2}{3}} \cdot \beta)</td>
</tr>
<tr>
<td>2</td>
<td>(O \left( \frac{n^2 m}{P} \right) \cdot \gamma)</td>
</tr>
<tr>
<td>3</td>
<td>(2 \log_2 P \frac{1}{3} \cdot \alpha + \frac{2n^2 \delta(P)}{P\frac{2}{3}} \cdot \beta + \frac{n^2 \delta(P)}{P\frac{2}{3}} \cdot \gamma)</td>
</tr>
<tr>
<td>4</td>
<td>(2 \log_2 P \frac{1}{3} \cdot \alpha + \frac{2n^2 \delta(P)}{P\frac{2}{3}} \cdot \beta)</td>
</tr>
<tr>
<td>5</td>
<td>(O \left( P\frac{2}{3} \log P \cdot \alpha + \frac{n^2 \delta(P)}{P\frac{2}{3}} \cdot \beta + \frac{n^3}{P} \cdot \gamma \right))</td>
</tr>
<tr>
<td>6</td>
<td>(2 \log_2 P \cdot \alpha + \left( \frac{4mn+n^2+nP\frac{1}{3}}{P\frac{2}{3}} \right) \delta(P) \cdot \beta + O \left( \frac{n^2 m}{P} \right) \cdot \gamma)</td>
</tr>
</tbody>
</table>

\[T_{\text{CholeskyQR}_3D}^{\alpha-\beta} (m, n, P) = O \left( P\frac{2}{3} \log P \cdot \alpha + \left( \frac{n^2 + nm}{P\frac{2}{3}} \right) \delta(P) \cdot \beta + \frac{n^2 m + n^3}{P} \cdot \gamma \right)\] (2)
Further analysis of CholeskyQR\_3D

Table: Costs of CholeskyQR\_2\_3D.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( O \left( \frac{P^2}{3} \log P \cdot \alpha + \frac{(n^2 + nm) \delta(P)}{P^2} \cdot \beta + \frac{n^2m + n^3}{P} \cdot \gamma \right) )</td>
</tr>
<tr>
<td>2</td>
<td>( O \left( \frac{P^2}{3} \log P \cdot \alpha + \frac{(n^2 + nm) \delta(P)}{P^2} \cdot \beta + \frac{n^2m + n^3}{P} \cdot \gamma \right) )</td>
</tr>
<tr>
<td>3</td>
<td>( 2 \log_2 P \cdot \alpha + \frac{(3n^2 + 6nP^\frac{1}{3}) \delta(P)}{2P^2} \cdot \beta + O \left( \frac{n^3}{P^3} \right) \cdot \gamma )</td>
</tr>
</tbody>
</table>

\[ T_{\text{CholeskyQR\_2\_3D}}^{\alpha - \beta} (m, n, P) = O \left( \frac{P^2}{3} \log P \cdot \alpha + \frac{(n^2 + nm) \delta(P)}{P^2} \cdot \beta + \frac{n^2m + n^3}{P} \cdot \gamma \right) \] (3)

Table: Costs of CholeskyQR\_2\_3D with varying block sizes.

| \( m > n \) | \( O \left( \frac{P^2}{3} \log P \cdot \alpha + \frac{nm\delta(P)}{P^2} \cdot \beta + \frac{n^2m}{P} \cdot \gamma \right) \) |
| \( m \leq n \) | \( O \left( \frac{P^2}{3} \log P \cdot \alpha + \frac{n^2\delta(P)}{P^2} \cdot \beta + \frac{n^3}{P} \cdot \gamma \right) \) |

This algorithm is most communication efficient when \( m = n \).
Figure: Tunable Cholesky-QR2 Algorithm
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

\[ B = R^T R \]

D/C simultaneous 3D Cholesky Factorizations on cubes of dimension \( C \)

D/C simultaneous 3D Matrix Multiplications on cubes of dimension \( C \)
Algorithm 2.9 \([Q, R] \leftarrow \text{CholeskyQR\textunderscore Tunable}(A, m, n, \Pi_T, i, j, k)\)

Require: \(\Pi_T\) has \(P\) processors arranged in a tunable grid of size \(c \times d \times c\) for any integer \(c\) in range \([0, \lfloor P^{\frac{1}{3}} \rfloor - 1]\). \(A\) is \(m \times n\) and is replicated on \(\Pi_T[:, :, k], \forall k \in [0, c - 1]\). Each processor \(\Pi_T[i, j, k]\) owns a (cyclic) blocked partition of \(A\) known as \(A_{ji}\). \(A_{ji}\) is packed into a 1D array of size \(\frac{mn}{dc}\), where it owns a rectangular piece of size \(\frac{m}{d} \times \frac{n}{c}\). Let \(W, X, Y, Z,\) and \(R^{-1}\) be temporary arrays distributed the same as \(A\).

1: \(\text{Bcast}(A_{jk}, W_{ji}, k, \Pi_T[:, :, k])\) \(\triangleright\) Broadcast from root \(k\) across row \(i\)
2: \(X_{ji} \leftarrow \text{seq-MM}(W_{ji}^T, A_{ji}, \frac{n}{c}, \frac{m}{d}, \frac{n}{c})\)
3: \(\text{AllReduce}(X_{ji}, Y_{ji}, \Pi_T[i, c \cdot \lfloor \frac{i}{c} \rfloor : (c + 1) \cdot \lfloor \frac{i}{c} \rfloor, k])\) \(\triangleright\) AllReduce among groups of \(c\) along each column
4: \(\text{AllReduce}(Y_{ji}, Z_{ji}, \Pi_T[i, c : c : d, k])\) \(\triangleright\) AllReduce among groups of \(c\) of size \(c\) distance away along each column
5: \(\text{Bcast}(Z_{ji}, Z_{ji}, k, \Pi_T[i, j, :])\) \(\triangleright\) Every 2D slice owns the same matrix \(B = A^T\)
6: Define \(\Pi_3 \leftarrow \Pi_T[:, c \cdot \lfloor \frac{i}{c} \rfloor : (c + 1) \cdot \lfloor \frac{i}{c} \rfloor, :]\) \(\triangleright\) Split rectangular processor grid into cubic grid of dimension \(c\)
7: \(R^T, R^{-T} \leftarrow \text{CholeskyFactorization3D}(Z, n, \frac{n}{2}, \Pi_3, i, j \mod c, k)\)
8: \(Q \leftarrow \text{MatrixMultiplication3D}(A, R^{-1}, m, n, \Pi_3, i, j \mod c, k)\)

Ensure: \(A = QR\), where \(Q\) and \(R\) are distributed the same as \(A\). \(Q\) is \(m \times n\) and \(R\) is an upper triangular matrix of dimension \(n\).
Algorithm 2.10 \([Q, R] \leftarrow \text{CholeskyQR2\_Tunable} (A, m, n, \Pi_T, i, j, k)\)

Require: Same requirements as Algorithm 7.

1: \(X, Y \leftarrow \text{CholeskyQR\_Tunable} (A, m, n, \Pi_T, i, j, k)\)
2: \(Q, Z \leftarrow \text{CholeskyQR\_Tunable} (X, m, n, \Pi_T, i, j, k)\)
3: Define \(\Pi_1 \leftarrow \Pi_T[\cdot : c \cdot \left\lfloor \frac{j}{c} \right\rfloor : (c + 1) \cdot \left\lfloor \frac{j}{c} \right\rfloor,:]\)
4: \(R \leftarrow \text{MatrixMultiplication3D} (Z, Y, n, \Pi_1, i, j \mod c, k)\)

Ensure: Same requirements as Algorithm 2.9.
Cost analysis for CholeskyQR_Tunable

Table: Costs of CholeskyQR_Tunable.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2 \log_2 c \cdot \alpha + \frac{2mn\delta(c)}{dc} \cdot \beta$</td>
</tr>
<tr>
<td>2</td>
<td>$O \left( \frac{n^2 m}{c^2 d} \right)$</td>
</tr>
<tr>
<td>3</td>
<td>$2 \log_2 c \cdot \alpha + \frac{2n^2\delta(c)}{c^2} \cdot \beta + \frac{n^2\delta(c)}{c^2} \cdot \gamma$</td>
</tr>
<tr>
<td>4</td>
<td>$2 \log_2 \frac{d}{c} \cdot \alpha + \frac{2n^2\delta(d)}{c^2} \cdot \beta + \frac{n^2\delta(d)}{c^2} \cdot \gamma$</td>
</tr>
<tr>
<td>5</td>
<td>$2 \log_2 c \cdot \alpha + \frac{2mn\delta(c)}{dc} \cdot \beta$</td>
</tr>
<tr>
<td>6</td>
<td>$O \left( c^2 \log c^3 \cdot \alpha + \frac{n^2\delta(c)}{c^2} \cdot \beta + \frac{n^3}{c^3} \cdot \gamma \right)$</td>
</tr>
<tr>
<td>7</td>
<td>$2 \log_2 c^3 \cdot \alpha + \left( \frac{4mn\delta(c)}{dc} + \frac{(n^2 + nc)\delta(c)}{c^2} \right) \cdot \beta + O \left( \frac{n^2 m}{c^2 d} \right) \cdot \gamma$</td>
</tr>
</tbody>
</table>

$$T_{\text{CholeskyQR-Tunable}}^{\alpha - \beta} (m, n, c, d) = O \left( c^2 \log c + \log \frac{d}{c} \right) \cdot \alpha$$

$$+ \left( \frac{mn\delta(c)}{dc} + \frac{n^2\delta(c)}{c^2} + \frac{n^2\delta(d)}{c^2} \right) \cdot \beta + \left( \frac{n^3}{c^3} + \frac{n^2 m}{c^2 d} \right) \cdot \gamma$$

$$= O \left( c^2 \log P \cdot \alpha + \frac{cmn\delta(c) + n^2d\delta(P)}{dc^2} \cdot \beta + \frac{n^3d + n^2mc}{c^3d} \cdot \gamma \right)$$

(4)
Cost analysis continued

Table: Costs of CholeskyQR2_Tunable.

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$O \left( c^2 \log P \cdot \alpha + \frac{c m n \delta(c) + n^2 d \delta(P)}{d c^2} \cdot \beta + \frac{n^3 d + n^2 m c}{c^3 d} \cdot \gamma \right) $</td>
</tr>
<tr>
<td>2</td>
<td>$O \left( c^2 \log P \cdot \alpha + \frac{c m n \delta(c) + n^2 d \delta(P)}{d c^2} \cdot \beta + \frac{n^3 d + n^2 m c}{c^3 d} \cdot \gamma \right) $</td>
</tr>
<tr>
<td>3</td>
<td>$2 \log_2 c^3 \cdot \alpha + \frac{3 n^2 + 6 n c \delta(c)}{2 c^2} \cdot \beta + O \left( \frac{n^3}{c^3} \right) \cdot \gamma$</td>
</tr>
</tbody>
</table>

$$T^{\alpha-\beta}_{\text{CholeskyQR2_Tunable}}(m, n, c, d) = O \left( c^2 \log P \cdot \alpha ight.$$ 
$$+ \frac{c m n \delta(c) + n^2 d \delta(P)}{d c^2} \cdot \beta + \frac{n^3 d + n^2 m c}{c^3 d} \cdot \gamma \right)$$ (5)
Further analysis of CholeskyQR\_Tunable

We can show that costs attained by CholeskyQR2\_Tunable correctly interpolates between the costs of CholeskyQR2\_1D and CholeskyQR2\_3D. Note that our $c \times d \times c$ grid requires $P = c^2d$ and $d \geq c$.

\[
T_{\text{CholeskyQR2\_Tunable}}^{\alpha - \beta} (m, n, 1, P) = O \left( 1^2 \log P \cdot \alpha + \frac{mn\delta (1) + n^2P\delta (P)}{P \cdot 1^2} \cdot \beta + \frac{n^3P + n^2m}{1^3 \cdot P} \cdot \gamma \right) 
\]
\[
= O \left( \log P \cdot \alpha + n^2\delta (P) \cdot \beta + \left( n^3 + \frac{n^2m}{P} \right) \cdot \gamma \right) 
\]

(6)

\[
T_{\text{CholeskyQR2\_Tunable}}^{\alpha - \beta} \left( m, n, P^{\frac{1}{3}}, P^{\frac{1}{3}} \right) = O \left( P^{\frac{2}{3}} \log P \cdot \alpha + \frac{mnP^{\frac{1}{3}} \delta \left( P^{\frac{1}{3}} \right) + n^2P^{\frac{1}{3}} \delta \left( P \right)}{P} \cdot \beta + \frac{n^3P^{\frac{1}{3}} + n^2mP^{\frac{1}{3}}}{P^{\frac{4}{3}}} \cdot \gamma \right) 
\]
\[
= O \left( P^{\frac{2}{3}} \log P \cdot \alpha + \frac{\left( n^2 + nm \right) \delta \left( P \right)}{P^{\frac{2}{3}}} \cdot \beta + \frac{n^3 + n^2m}{P} \cdot \gamma \right) 
\]

(7)
## Costs attained by CholeskyQR2 algorithm variants

<table>
<thead>
<tr>
<th>Grid shape</th>
<th>Metric</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \times P \times 1$</td>
<td># of messages</td>
<td>$\mathcal{O}(\log P)$</td>
</tr>
<tr>
<td></td>
<td># of words</td>
<td>$\mathcal{O}(n^2 \delta(P))$</td>
</tr>
<tr>
<td></td>
<td># of flops</td>
<td>$\mathcal{O}\left(\frac{n^2 m}{P} + n^3\right)$</td>
</tr>
<tr>
<td></td>
<td>Memory footprint</td>
<td>$\mathcal{O}\left(\frac{mn}{P} + n^2\right)$</td>
</tr>
<tr>
<td>$P^{\frac{1}{3}} \times P^{\frac{1}{3}} \times P^{\frac{1}{3}}$</td>
<td># of messages</td>
<td>$\mathcal{O}\left(P^{\frac{2}{3}} \log P\right)$</td>
</tr>
<tr>
<td></td>
<td># of words</td>
<td>$\mathcal{O}\left(\frac{n^2 + nm}{P^{\frac{2}{3}}} \delta(P)\right)$</td>
</tr>
<tr>
<td></td>
<td># of flops</td>
<td>$\mathcal{O}\left(\frac{n^2 m + n^3}{P}\right)$</td>
</tr>
<tr>
<td></td>
<td>Memory footprint</td>
<td>$\mathcal{O}\left(\frac{mn + n^2}{P^{\frac{2}{3}}}\right)$</td>
</tr>
<tr>
<td>$c \times d \times c$</td>
<td># of messages</td>
<td>$\mathcal{O}\left(c^2 \log P\right)$</td>
</tr>
<tr>
<td></td>
<td># of words</td>
<td>$\mathcal{O}\left(\frac{cmn \delta(c) + n^2 d \delta(P)}{dc^2}\right)$</td>
</tr>
<tr>
<td></td>
<td># of flops</td>
<td>$\mathcal{O}\left(\frac{n^3 d + n^2 mc}{c^3 d}\right)$</td>
</tr>
<tr>
<td></td>
<td>Memory footprint</td>
<td>$\mathcal{O}\left(\frac{mnc + n^2 d}{c^2 d}\right)$</td>
</tr>
</tbody>
</table>
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

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

$$T_{\text{CholeskyQR2-Tunable}}^{\alpha - \beta} \left( m, n, \left( \frac{Pn}{m} \right)^{\frac{1}{3}}, \left( \frac{Pm^2}{n^2} \right)^{\frac{1}{3}} \right) = \mathcal{O} \left( \left( \frac{Pn}{m} \right)^{\frac{2}{3}} \log P \cdot \alpha \right)$$

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

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

<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>$\mathcal{O} \left( \left( \frac{Pn}{m} \right)^{\frac{2}{3}} \log P \right)$</td>
</tr>
<tr>
<td></td>
<td># of words</td>
<td>$\mathcal{O} \left( \left( \frac{n^2 m}{P} \right)^{\frac{2}{3}} \delta (P) \right)$</td>
</tr>
<tr>
<td></td>
<td># of flops</td>
<td>$\mathcal{O} \left( \frac{n^2 m}{P} \right)$</td>
</tr>
<tr>
<td></td>
<td>Memory footprint</td>
<td>$\mathcal{O} \left( \left( \frac{n^2 m}{P} \right)^{\frac{2}{3}} \right)$</td>
</tr>
</tbody>
</table>
Implementation and results

- All code written from scratch in C++11 and MPI
- Sitting in github right now, and waiting for final tune up before running on Blue Waters
- Optimistic, but ready to apply optimization strategies (Charmworks has taught me a lot in this regard through work on the CASI Solver project)
- Results will probably be ready sometime in August