communication-optimal QR factorizations: performance and scalability on varying architectures

Edward Hutter and Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

Blue Waters Symposium 2019

L P N A @CS@Illinois

Motivation for reducing algorithmic communication costs

Communication and synchronization increasingly dominating algorithm performance on modern architectures

Motivation for reducing algorithmic communication costs

Communication and synchronization increasingly dominating algorithm performance on modern architectures

$\alpha - \beta - \gamma$ cost model

- α - cost to send zero-byte message
- β - cost to inject byte of data into network
- γ - cost to perform flop with register-resident data

Motivation for reducing algorithmic communication costs

Communication and synchronization increasingly dominating algorithm performance on modern architectures

$\alpha - \beta - \gamma$ cost model

- α - cost to send zero-byte message
- β - cost to inject byte of data into network
- γ - cost to perform flop with register-resident data

Architectural trend: $\alpha \gg \beta \gg \gamma$

Motivation for reducing algorithmic communication costs

Communication and synchronization increasingly dominating algorithm performance on modern architectures

$\alpha - \beta - \gamma$ cost model

- α - cost to send zero-byte message
- β - cost to inject byte of data into network
- γ - cost to perform flop with register-resident data

Architectural trend: $\alpha \gg \beta \gg \gamma$

Communication-avoiding algorithms for **most** dense matrix factorizations present in numerical libraries

Motivation for reducing algorithmic communication costs

Communication and synchronization increasingly dominating algorithm performance on modern architectures

$\alpha - \beta - \gamma$ cost model

- α - cost to send zero-byte message
- β - cost to inject byte of data into network
- γ - cost to perform flop with register-resident data

Architectural trend: $\alpha \gg \beta \gg \gamma$

Communication-avoiding algorithms for **most** dense matrix factorizations present in numerical libraries

Goal: A QR factorization algorithm that prioritizes minimizing synchronization and communication cost

Motivation for reducing algorithmic communication costs

Communication and synchronization increasingly dominating algorithm performance on modern architectures

$\alpha - \beta - \gamma$ cost model

- α - cost to send zero-byte message
- β - cost to inject byte of data into network
- γ - cost to perform flop with register-resident data

Architectural trend: $\alpha \gg \beta \gg \gamma$

Communication-avoiding algorithms for **most** dense matrix factorizations present in numerical libraries

Goal: A QR factorization algorithm that prioritizes minimizing synchronization and communication cost

Our team uses BlueWaters to assess the scalability of new algorithms for numerical tensor algebra at massively large scale

Architecture trends: machine balance decreasing

machine	launch year	peak node perf (Gflops/s)	peak injection bandwidth (Gwords/sec)	machine balance (words/flop)
ASCI Red	1997	0.666	0.4	1/1.665
ANL BG/P	2007	13.6	1	1/13.6
ONL Jaguar	2009	124.8	2.2	1/56
ANL BG/Q	2012	205	2	1/102.5
NCSA BlueWaters (XE)	2012	313.6	9.6	1/32
NCSA BlueWaters (XK)	2012	1320	9.6	1/137.5
ORNL Titan	2013	1320	8	1/165
ANL Theta	2017	3000+	10.2	1/294
TACC Stampede2	2017	3000+	12.5	1/240
LLNL Sierra	2018	28000	12.5	1/2240
ORNL Summit	2018	44000	12.5	1/3520

Architecture trends: machine balance decreasing

machine	launch year	peak node perf (Gflops/s)	peak injection bandwidth (Gwords/sec)	machine balance (words/flop)
ASCI Red	1997	0.666	0.4	1/1.665
ANL BG/P	2007	13.6	1	1/13.6
ONL Jaguar	2009	124.8	2.2	1/56
ANL BG/Q	2012	205	2	1/102.5
NCSA BlueWaters (XE)	2012	313.6	9.6	1/32
NCSA BlueWaters (XK)	2012	1320	9.6	1/137.5
ORNL Titan	2013	1320	8	1/165
ANL Theta	2017	3000+	10.2	1/294
TACC Stampede2	2017	3000+	12.5	1/240
LLNL Sierra	2018	28000	12.5	1/2240
ORNL Summit	2018	44000	12.5	1/3520

Higher arithmetic intensity →higher performance on new architectures

Architecture trends: machine balance decreasing

machine	launch year	peak node perf (Gflops/s)	peak injection bandwidth (Gwords/sec)	machine balance (words/flop)
ASCI Red	1997	0.666	0.4	1/1.665
ANL BG/P	2007	13.6	1	1/13.6
ONL Jaguar	2009	124.8	2.2	1/56
ANL BG/Q	2012	205	2	1/102.5
NCSA BlueWaters (XE)	2012	313.6	9.6	1/32
NCSA BlueWaters (XK)	2012	1320	9.6	1/137.5
ORNL Titan	2013	1320	8	1/165
ANL Theta	2017	3000+	10.2	1/294
TACC Stampede2	2017	3000+	12.5	1/240
LLNL Sierra	2018	28000	12.5	1/2240
ORNL Summit	2018	44000	12.5	1/3520

Higher arithmetic intensity →higher performance on new architectures

BlueWaters **not** a favorable machine for communication-avoiding algorithms

Communication-avoiding Cholesky-QR2 (CA-CQR2)

3D algorithms utilize available extra memory to reduce communication asymptotically.

Communication-avoiding Cholesky-QR2 (CA-CQR2)

3D algorithms utilize available extra memory to reduce communication asymptotically.

We introduce CA-CQR2, a novel practical 3D QR factorization algorithm

Communication-avoiding Cholesky-QR2 (CA-CQR2)

3D algorithms utilize available extra memory to reduce communication asymptotically.

We introduce CA-CQR2, a novel practical 3D QR factorization algorithm

- extends CholeskyQR2 algorithm to arbitary $m \times n$ matrices across P processes

Communication-avoiding Cholesky-QR2 (CA-CQR2)

3D algorithms utilize available extra memory to reduce communication asymptotically.

We introduce CA-CQR2, a novel practical 3D QR factorization algorithm

- extends CholeskyQR2 algorithm to arbitary $m \times n$ matrices across P processes
- requires $\mathcal{O}\left(\left(\mathbf{Pm^2/n^2}\right)^{\mathbf{1/6}}\right)$ less communication than known 2D QR algorithms

Communication-avoiding Cholesky-QR2 (CA-CQR2)

3D algorithms utilize available extra memory to reduce communication asymptotically.

We introduce CA-CQR2, a novel practical 3D QR factorization algorithm

- extends CholeskyQR2 algorithm to arbitary $m \times n$ matrices across P processes
- requires $\mathcal{O}\left(\left(\mathbf{Pm^2/n^2}\right)^{\mathbf{1/6}}\right)$ less communication than known 2D QR algorithms
- incurs a number of (increasingly profitable) tradeoffs
  - $2-4$x more flops than Householder QR)
  - matrix must be sufficiently well-conditioned
  - requires $\mathcal{O}\left(\left(\mathbf{Pm/n}\right)^{\mathbf{1/3}}\right)$ more memory than known 2D QR algorithms

Communication-avoiding Cholesky-QR2 (CA-CQR2)

3D algorithms utilize available extra memory to reduce communication asymptotically.

We introduce CA-CQR2, a novel practical 3D QR factorization algorithm

- extends CholeskyQR2 algorithm to arbitary $m \times n$ matrices across P processes
- requires $\mathcal{O}\left(\left(\mathbf{Pm^2/n^2}\right)^{\mathbf{1/6}}\right)$ less communication than known 2D QR algorithms
- incurs a number of (increasingly profitable) tradeoffs
  - $2-4$x more flops than Householder QR)
  - matrix must be sufficiently well-conditioned
  - requires $\mathcal{O}\left(\left(\mathbf{Pm/n}\right)^{\mathbf{1/3}}\right)$ more memory than known 2D QR algorithms

All algorithms will be measured along the critical path instead of a volume measure

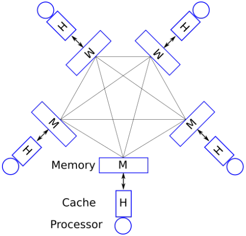
Communication-avoiding Cholesky-QR2 (CA-CQR2)

3D algorithms utilize available extra memory to reduce communication asymptotically.

We introduce CA-CQR2, a novel practical 3D QR factorization algorithm

- extends CholeskyQR2 algorithm to arbitary $m \times n$ matrices across P processes
- requires $\mathcal{O}\left(\left(\mathbf{Pm^2/n^2}\right)^{1/6}\right)$ less communication than known 2D QR algorithms
- incurs a number of (increasingly profitable) tradeoffs
  - $2-4$x more flops than Householder QR)
  - matrix must be sufficiently well-conditioned
  - requires $\mathcal{O}\left(\left(\mathbf{Pm/n}\right)^{1/3}\right)$ more memory than known 2D QR algorithms

All algorithms will be measured along the critical path instead of a volume measure



T_near-neighbor-exchange(n,P) = α + n*β

T_all-reduce(n,P) = f(P)*α + f(P)n*β

Figure: Horizontal (internode network) communication along critical path

QR Strong scaling performance

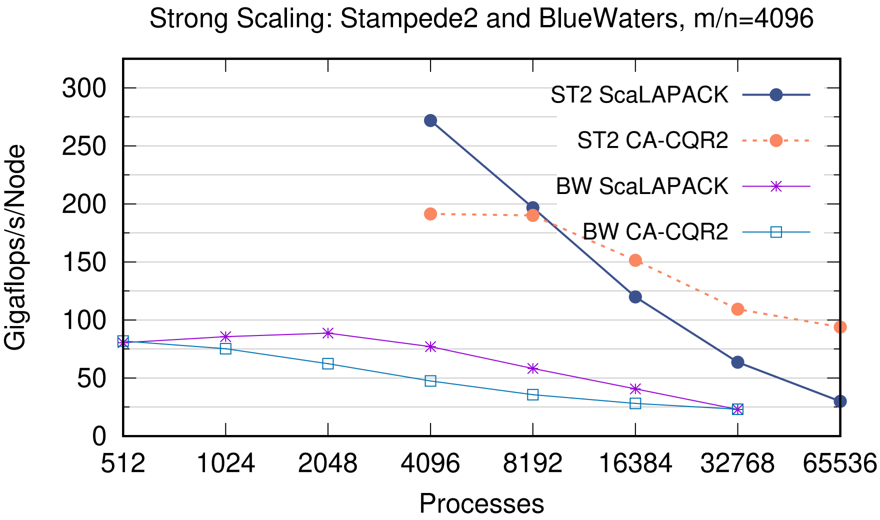


Figure: Strong scaling for $m \times n$ matrices

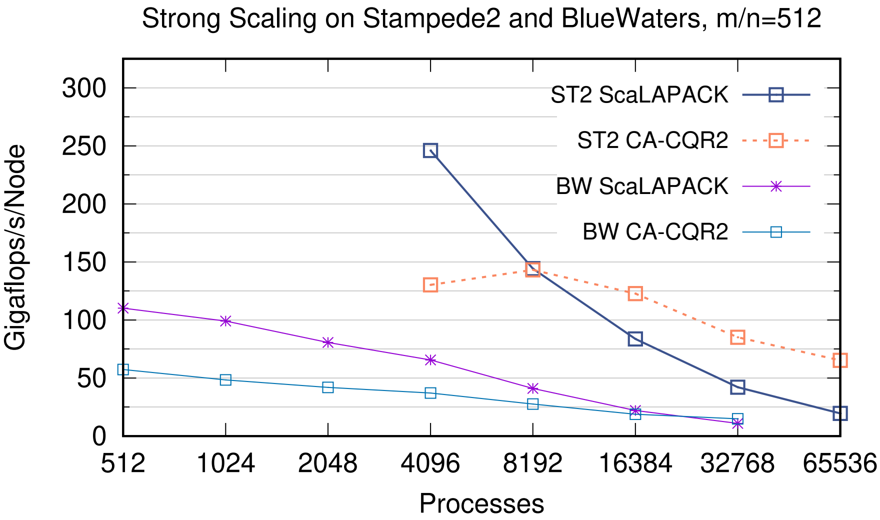


Figure: Strong scaling for $m \times n$ matrices

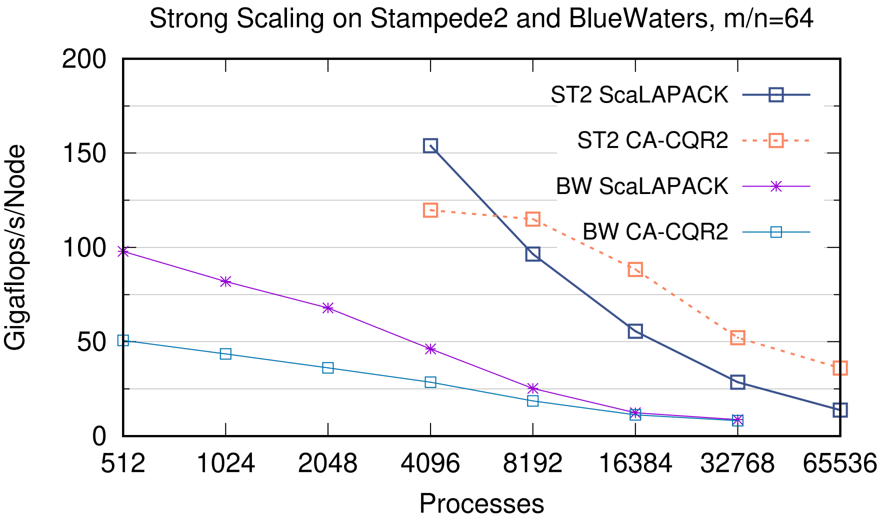


Figure: Strong scaling for $m \times n$ matrices

QR Strong scaling performance

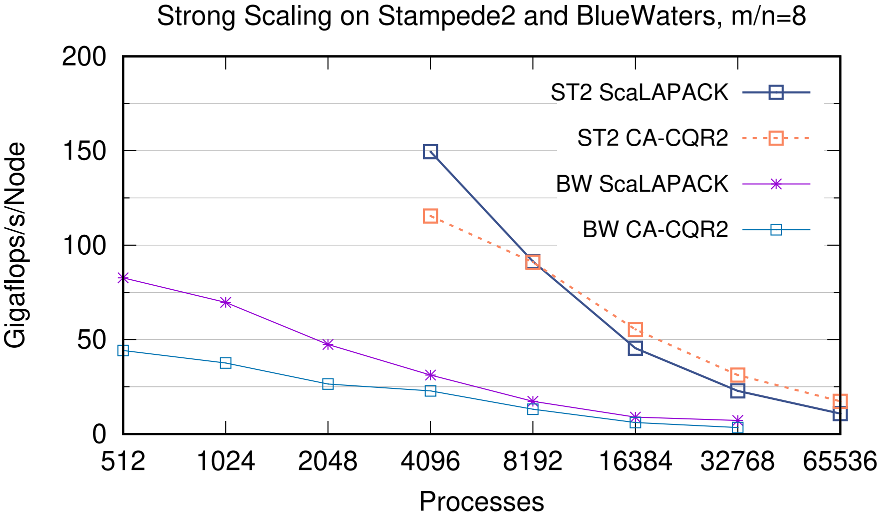


Figure: Strong scaling for $m \times n$ matrices

QR Strong scaling performance

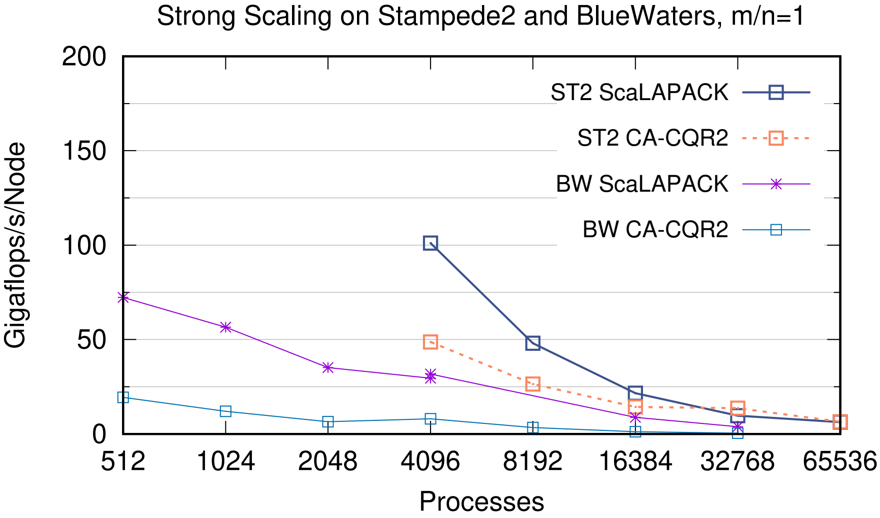


Figure: Strong scaling for $m \times n$ matrices

ScaLAPACK's PGEQRF is communication-optimal assuming minimal memory (2D)

$$T^{\alpha,\beta}_{\text{PGEQRF}} = \mathcal{O}\left(n \log P \cdot \alpha + \frac{mn}{\sqrt{P}} \cdot \beta \right) \qquad M_{\text{PGEQRF}} = \mathcal{O}(\frac{mn}{P})$$

[1] J. Demmel et al., "Communication-optimal Parallel and Sequential QR and LU Factorizations", SISC 2012

[2] A. Tiskin, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

[3] E. Solomonik et al., "A communication-avoiding parallel algorithm for the symmetric eigenvalue problem", SPAA 2017

[4] G. Ballard et al., "A 3D Parallel Algorithm for QR Decomposition", SPAA 2018

[5] E. Hutter et al., "Communication-avoiding CholeskyQR2 for rectangular matrices", IPDPS 2019

Competing costs of parallel QR factorization of $A_{m \times n}$

ScaLAPACK's PGEQRF is communication-optimal assuming minimal memory (2D)

$$T^{\alpha,\beta}_{\text{PGEQRF}} = \mathcal{O}\left(n \log P \cdot \alpha + \frac{mn}{\sqrt{P}} \cdot \beta \right) \qquad M_{\text{PGEQRF}} = \mathcal{O}(\frac{mn}{P})$$

CAQR factors panels using TSQR to reduce synchronization[1] (2D)

$$T^{\alpha,\beta}_{\text{CAQR}} = \mathcal{O}\left(\sqrt{P} \log^2 P \cdot \alpha + \frac{mn}{\sqrt{P}} \cdot \beta \right) \qquad M_{\text{CAQR}} = \mathcal{O}(\frac{mn}{P})$$

[1] J. Demmel et al., "Communication-optimal Parallel and Sequential QR and LU Factorizations", SISC 2012

[2] A. Tiskin, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

[3] E. Solomonik et al., "A communication-avoiding parallel algorithm for the symmetric eigenvalue problem", SPAA 2017

[4] G. Ballard et al., "A 3D Parallel Algorithm for QR Decomposition", SPAA 2018

[5] E. Hutter et al., "Communication-avoiding CholeskyQR2 for rectangular matrices", IPDPS 2019

Competing costs of parallel QR factorization of $A_{m\times n}$

ScaLAPACK's PGEQRF is communication-optimal assuming minimal memory (2D)

$$T^{\alpha,\beta}_{\text{PGEQRF}} = \mathcal{O}\left(n\log P\cdot\alpha + \frac{mn}{\sqrt{P}}\cdot\beta\right) \qquad M_{\text{PGEQRF}} = \mathcal{O}(\frac{mn}{P})$$

CAQR factors panels using TSQR to reduce synchronization[1] (2D)

$$T^{\alpha,\beta}_{\text{CAQR}} = \mathcal{O}\left(\sqrt{P}\log^2 P\cdot\alpha + \frac{mn}{\sqrt{P}}\cdot\beta\right) \qquad M_{\text{CAQR}} = \mathcal{O}(\frac{mn}{P})$$

CA-CQR2 leverages extra memory to reduce communication (3D)

$$T^{\alpha,\beta}_{\text{CA-CQR2}} = \mathcal{O}\left(\left(\frac{Pn}{m}\right)^{\frac{2}{3}}\log P\cdot\alpha + \left(\frac{n^2m}{P}\right)^{\frac{2}{3}}\cdot\beta\right) \qquad M_{\text{CA-CQR2}} = \mathcal{O}\left(\left(\frac{n^2m}{P}\right)^{\frac{2}{3}}\right)$$

---

[1] J. Demmel et al., "Communication-optimal Parallel and Sequential QR and LU Factorizations", SISC 2012

[2] A. Tiskin, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

[3] E. Solomonik et al., "A communication-avoiding parallel algorithm for the symmetric eigenvalue problem", SPAA 2017

[4] G. Ballard et al., "A 3D Parallel Algorithm for QR Decomposition", SPAA 2018

[5] E. Hutter et al., "Communication-avoiding CholeskyQR2 for rectangular matrices", IPDPS 2019

Competing costs of parallel QR factorization of $A_{m \times n}$

ScaLAPACK's PGEQRF is communication-optimal assuming minimal memory (2D)

$$T^{\alpha,\beta}_{\text{PGEQRF}} = \mathcal{O}\left(n \log P \cdot \alpha + \frac{mn}{\sqrt{P}} \cdot \beta\right) \qquad M_{\text{PGEQRF}} = \mathcal{O}\left(\frac{mn}{P}\right)$$

CAQR factors panels using TSQR to reduce synchronization[1] (2D)

$$T^{\alpha,\beta}_{\text{CAQR}} = \mathcal{O}\left(\sqrt{P} \log^2 P \cdot \alpha + \frac{mn}{\sqrt{P}} \cdot \beta\right) \qquad M_{\text{CAQR}} = \mathcal{O}\left(\frac{mn}{P}\right)$$

CA-CQR2 leverages extra memory to reduce communication (3D)

$$T^{\alpha,\beta}_{\text{CA-CQR2}} = \mathcal{O}\left(\left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P \cdot \alpha + \left(\frac{n^2 m}{P}\right)^{\frac{2}{3}} \cdot \beta\right) \qquad M_{\text{CA-CQR2}} = \mathcal{O}\left(\left(\frac{n^2 m}{P}\right)^{\frac{2}{3}}\right)$$

3D algorithms exist in theory[2] [3] [4], but **CA-CQR2 is the first practical approach**[5]

[1] J. Demmel et al., "Communication-optimal Parallel and Sequential QR and LU Factorizations", SISC 2012

[2] A. Tiskin, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

[3] E. Solomonik et al., "A communication-avoiding parallel algorithm for the symmetric eigenvalue problem", SPAA 2017

[4] G. Ballard et al., "A 3D Parallel Algorithm for QR Decomposition", SPAA 2018

[5] E. Hutter et al., "Communication-avoiding CholeskyQR2 for rectangular matrices", IPDPS 2019

Instability of Cholesky-QR

QR factorization algorithms used in practice stem from processes of orthogonal triangularization for their superior numerical stability

$$Q_n Q_{n-1} \dots Q_1 A = R$$

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

Instability of Cholesky-QR

QR factorization algorithms used in practice stem from processes of orthogonal triangularization for their superior numerical stability

$$Q_n Q_{n-1} \dots Q_1 A = R$$

The Cholesky-QR algorithm is a simple algorithm that follows a numerically unstable process of triangular orthogonalization

$$A R_1^{-1} R_2^{-1} \dots R_n^{-1} = Q$$

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

Instability of Cholesky-QR

QR factorization algorithms used in practice stem from processes of orthogonal triangularization for their superior numerical stability

$$Q_n Q_{n-1} \ldots Q_1 A = R$$

The Cholesky-QR algorithm is a simple algorithm that follows a numerically unstable process of triangular orthogonalization

$$A R_1^{-1} R_2^{-1} \ldots R_n^{-1} = Q$$

$[Q, R] \leftarrow$ **Cholesky-QR** (A)

$B \leftarrow A^T A$	$\triangleright$ B may be indefinite!
$R^T R \leftarrow B$	$\triangleright$ Possible failure in Cholesky factorization!
$Q \leftarrow A R^{-1}$	$\triangleright$ R may have lost all accuracy! Q may lost orthogonality!

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

Instability of Cholesky-QR

QR factorization algorithms used in practice stem from processes of orthogonal triangularization for their superior numerical stability

$$Q_n Q_{n-1} \dots Q_1 A = R$$

The Cholesky-QR algorithm is a simple algorithm that follows a numerically unstable process of triangular orthogonalization

$$A R_1^{-1} R_2^{-1} \dots R_n^{-1} = Q$$

$[Q, R] \leftarrow$ **Cholesky-QR** (A)

$B \leftarrow A^T A$ $\triangleright$ B may be indefinite!

$R^T R \leftarrow B$ $\triangleright$ Possible failure in Cholesky factorization!

$Q \leftarrow A R^{-1}$ $\triangleright$ R may have lost all accuracy! Q may lost orthogonality!

CholeskyQR2 leverages near-perfect conditioning of Q in a second iteration[1]

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

Scalability of Cholesky-QR2

Cholesky-QR2 (CQR2) can achieve superior performance on tall-and-skinny matrices[1]

[1] T. Fukaya et al., "CholeskyQR2: A communication-avoiding algorithm", ScalA 2014

Scalability of Cholesky-QR2

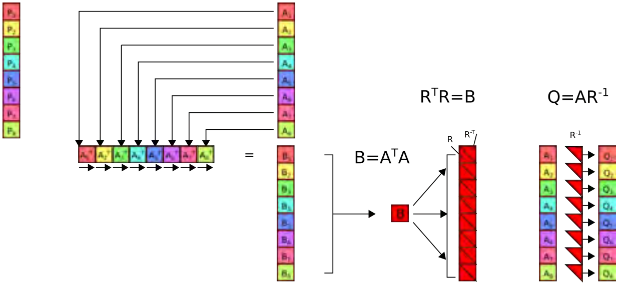
Cholesky-QR2 (CQR2) can achieve superior performance on tall-and-skinny matrices[1]

- Householder QR - $2mn^2 - \frac{2n^3}{3}$ flops, Cholesky-QR2 - $4mn^2 + \frac{5n^3}{3}$ flops

[1] T. Fukaya et al., "CholeskyQR2: A communication-avoiding algorithm", ScalA 2014

Cholesky-QR2 (CQR2) can achieve superior performance on tall-and-skinny matrices[1]

- Householder QR - $2mn^2 - \frac{2n^3}{3}$ flops, Cholesky-QR2 - $4mn^2 + \frac{5n^3}{3}$ flops

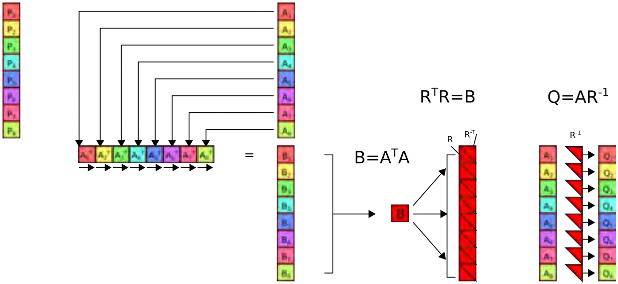


[1] T. Fukaya et al., "CholeskyQR2: A communication-avoiding algorithm", ScalA 2014

Scalability of Cholesky-QR2

Cholesky-QR2 (CQR2) can achieve superior performance on tall-and-skinny matrices[1]

- Householder QR - $2mn^2 - \frac{2n^3}{3}$ flops, Cholesky-QR2 - $4mn^2 + \frac{5n^3}{3}$ flops



CQR2 attains minimal communication cost (by $\mathcal{O}(\log P)$), yet simple implementation

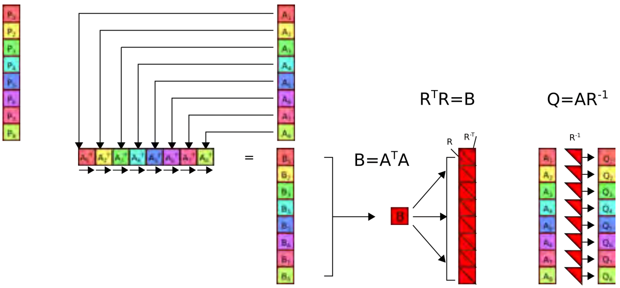
$$T_{\text{Cholesky-QR2}}(m,n,P) = \mathcal{O}\left(\log P \cdot \alpha + n^2 \cdot \beta + \left(\frac{n^2 m}{P} + n^3\right) \cdot \gamma\right)$$

[1] T. Fukaya et al., "CholeskyQR2: A communication-avoiding algorithm", ScalA 2014

Scalability of Cholesky-QR2

Cholesky-QR2 (CQR2) can achieve superior performance on tall-and-skinny matrices[1]

- Householder QR - $2mn^2 - \frac{2n^3}{3}$ flops, Cholesky-QR2 - $4mn^2 + \frac{5n^3}{3}$ flops



CQR2 attains minimal communication cost (by $\mathcal{O}(\log P)$), yet simple implementation

$$T_{\text{Cholesky-QR2}}(m,n,P) = \mathcal{O}\left(\log P \cdot \alpha + n^2 \cdot \beta + \left(\frac{n^2 m}{P} + n^3\right) \cdot \gamma\right)$$

CA-CQR2 parallelizes Cholesky-QR2 over a 3D processor grid, **efficiently factoring any rectangular matrix**

[1] T. Fukaya et al., "CholeskyQR2: A communication-avoiding algorithm", ScalA 2014

CA-CQR2's communication-optimal parallelization

CA-CQR2 leverages known 3D algorithms for matrix multiplication[1] and Cholesky factorization[2]

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

[2] A. Tiskin 2007, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

CA-CQR2's communication-optimal parallelization

CA-CQR2 leverages known 3D algorithms for matrix multiplication[1] and Cholesky factorization[2]

A tunable 3D processor grid of dimensions $c \times d \times c$ determines the replication factor (c), the communication reduction ($\sqrt{c}$), and the number of simultaneous instances of 3D algorithms (d/c)

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

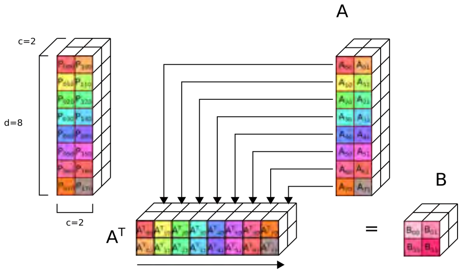
[2] A. Tiskin 2007, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

CA-CQR2's communication-optimal parallelization

CA-CQR2 leverages known 3D algorithms for matrix multiplication[1] and Cholesky factorization[2]

A tunable 3D processor grid of dimensions $c \times d \times c$ determines the replication factor (c), the communication reduction ($\sqrt{c}$), and the number of simultaneous instances of 3D algorithms (d/c)

Figure: Computation of Gram matrix $A^T A$



Cost: $\mathcal{O}((\log c + \log d/c) \cdot \alpha + \left(\frac{mn}{dc} + \frac{n^2}{c^2}\right) \cdot \beta + \left(\frac{mn^2}{dc^2} + \frac{n^2}{c^2}\right) \cdot \gamma$

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

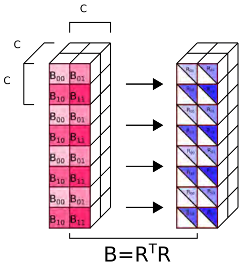
[2] A. Tiskin 2007, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

CA-CQR2's communication-optimal parallelization

CA-CQR2 leverages known 3D algorithms for matrix multiplication[1] and Cholesky factorization[2]

A tunable 3D processor grid of dimensions $c \times d \times c$ determines the replication factor (c), the communication reduction ($\sqrt{c}$), and the number of simultaneous instances of 3D algorithms (d/c)

Figure: $\frac{d}{c}$ simultaneous 3D Cholesky on cubes of dimension c



Cost: $\mathcal{O}\left(c^2 \log c^3 \cdot \alpha + \frac{n^2}{c^2} \cdot \beta + \frac{n^3}{c^3} \cdot \gamma\right)$

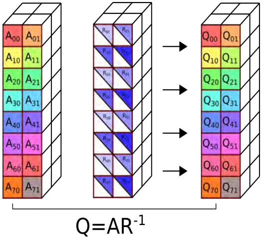
[1] Bersten 1989, "Communication-efficient matrix multiplication on hypercubes", Aggarwal, Chandra, Snir 1990, "Communication complexity of PRAMs", Agarwal et al. 1995, "A three-dimensional approach to parallel matrix multiplication"

[2] A. Tiskin 2007, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

CA-CQR2 leverages known 3D algorithms for matrix multiplication[1] and Cholesky factorization[2]

A tunable 3D processor grid of dimensions $c \times d \times c$ determines the replication factor (c), the communication reduction ($\sqrt{c}$), and the number of simultaneous instances of 3D algorithms (d/c)

Figure: $\frac{d}{c}$ simultaneous 3D MatMul / TRSM on cubes of dimension c



Cost: $\mathcal{O}\left(\log c^3 \cdot \alpha + \frac{n^2}{c^2} \cdot \beta + \frac{n^3}{c^3} \cdot \gamma\right)$

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

[2] A. Tiskin 2007, "Communication-efficient generic pairwise elimination", Future Generation Computer Systems 2007

Algorithmic cost analysis: CA-CQR2 vs. competition

CA-CQR2's cost expression expresses tunable tradeoffs

$$T^{\alpha-\beta}_{\text{CA-CQR2}}(m,n,c,d) = \mathcal{O}\left(c^2\log(d/c)\cdot\alpha + \left(\frac{mn}{dc}+\frac{n^2}{c^2}\right)\cdot\beta + \left(\frac{mn^2}{c^2d}+\frac{n^3}{c^3}\right)\cdot\gamma\right)$$

CA-CQR2's cost expression expresses tunable tradeoffs

$$T^{\alpha-\beta}_{\text{CA-CQR2}}(m,n,c,d) = \mathcal{O}\left(c^2\log(d/c)\cdot\alpha + \left(\frac{mn}{dc}+\frac{n^2}{c^2}\right)\cdot\beta + \left(\frac{mn^2}{c^2d}+\frac{n^3}{c^3}\right)\cdot\gamma\right)$$

Requiring each processor to own a square submatrix ($\frac{m}{d}=\frac{n}{c}$) and enforcing $P=c^2d$,
CA-CQR2 finds an optimal processor grid that supports minimal communication

Algorithmic cost analysis: CA-CQR2 vs. competition

CA-CQR2's cost expression expresses tunable tradeoffs

$$T^{\alpha-\beta}_{\text{CA-CQR2}}(m,n,c,d) = \mathcal{O}\left(c^2\log(d/c)\cdot\alpha + \left(\frac{mn}{dc} + \frac{n^2}{c^2}\right)\cdot\beta + \left(\frac{mn^2}{c^2 d} + \frac{n^3}{c^3}\right)\cdot\gamma\right)$$

Requiring each processor to own a square submatrix ($\frac{m}{d} = \frac{n}{c}$) and enforcing $P = c^2 d$,
CA-CQR2 finds an optimal processor grid that supports minimal communication

	1D Cholesky-QR2
messages	$\mathcal{O}(\log P)$
words	$\mathcal{O}\left(n^2\right)$
flops	$\mathcal{O}\left(\frac{n^2 m}{P} + n^3\right)$
memory	$\mathcal{O}\left(\frac{mn}{P} + n^2\right)$

Algorithmic cost analysis: CA-CQR2 vs. competition

CA-CQR2's cost expression expresses tunable tradeoffs

$$T^{\alpha-\beta}_{\text{CA-CQR2}}(m,n,c,d) = \mathcal{O}\left(c^2 \log(d/c)\cdot\alpha + \left(\frac{mn}{dc} + \frac{n^2}{c^2}\right)\cdot\beta + \left(\frac{mn^2}{c^2 d} + \frac{n^3}{c^3}\right)\cdot\gamma\right)$$

Requiring each processor to own a square submatrix ($\frac{m}{d} = \frac{n}{c}$) and enforcing $P = c^2 d$,
CA-CQR2 finds an optimal processor grid that supports minimal communication

	1D Cholesky-QR2	**2D ScaLAPACK**
messages	$\mathcal{O}(\log P)$	$\mathcal{O}(n \log P)$
words	$\mathcal{O}(n^2)$	$\mathcal{O}(\frac{mn}{\sqrt{P}})$
flops	$\mathcal{O}\left(\frac{n^2 m}{P} + n^3\right)$	$\mathcal{O}(\frac{mn^2}{P})$
memory	$\mathcal{O}\left(\frac{mn}{P} + n^2\right)$	$\mathcal{O}(\frac{mn}{P})$

Algorithmic cost analysis: CA-CQR2 vs. competition

CA-CQR2's cost expression expresses tunable tradeoffs

$$T^{\alpha-\beta}_{\text{CA-CQR2}}(m,n,c,d) = \mathcal{O}\left(c^2 \log(d/c)\cdot\alpha + \left(\frac{mn}{dc} + \frac{n^2}{c^2}\right)\cdot\beta + \left(\frac{mn^2}{c^2 d} + \frac{n^3}{c^3}\right)\cdot\gamma\right)$$

Requiring each processor to own a square submatrix ($\frac{m}{d} = \frac{n}{c}$) and enforcing $P = c^2 d$,
CA-CQR2 finds an optimal processor grid that supports minimal communication

	1D Cholesky-QR2	**2D ScaLAPACK**	**2D CAQR**
messages	$\mathcal{O}(\log P)$	$\mathcal{O}(n \log P)$	$\mathcal{O}\left(\sqrt{P}\log^2 P\right)$
words	$\mathcal{O}(n^2)$	$\mathcal{O}(\frac{mn}{\sqrt{P}})$	$\mathcal{O}(\frac{mn}{\sqrt{P}})$
flops	$\mathcal{O}\left(\frac{n^2 m}{P} + n^3\right)$	$\mathcal{O}(\frac{mn^2}{P})$	$\mathcal{O}(\frac{mn^2}{P})$
memory	$\mathcal{O}\left(\frac{mn}{P} + n^2\right)$	$\mathcal{O}(\frac{mn}{P})$	$\mathcal{O}(\frac{mn}{P})$

Algorithmic cost analysis: CA-CQR2 vs. competition

CA-CQR2's cost expression expresses tunable tradeoffs

$$T^{\alpha-\beta}_{\text{CA-CQR2}}(m,n,c,d) = \mathcal{O}\left(c^2 \log(d/c)\cdot\alpha + \left(\frac{mn}{dc} + \frac{n^2}{c^2}\right)\cdot\beta + \left(\frac{mn^2}{c^2 d} + \frac{n^3}{c^3}\right)\cdot\gamma\right)$$

Requiring each processor to own a square submatrix ($\frac{m}{d} = \frac{n}{c}$) and enforcing $P = c^2 d$,
CA-CQR2 finds an optimal processor grid that supports minimal communication

	1D Cholesky-QR2	2D ScaLAPACK	2D CAQR	3D CA-CQR2
messages	$\mathcal{O}(\log P)$	$\mathcal{O}(n \log P)$	$\mathcal{O}\left(\sqrt{P}\log^2 P\right)$	$\mathcal{O}\left(\left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P\right)$
words	$\mathcal{O}\left(n^2\right)$	$\mathcal{O}(\frac{mn}{\sqrt{P}})$	$\mathcal{O}(\frac{mn}{\sqrt{P}})$	$\mathcal{O}\left(\left(\frac{n^2 m}{P}\right)^{\frac{2}{3}}\right)$
flops	$\mathcal{O}\left(\frac{n^2 m}{P} + n^3\right)$	$\mathcal{O}(\frac{mn^2}{P})$	$\mathcal{O}(\frac{mn^2}{P})$	$\mathcal{O}\left(\frac{n^2 m}{P}\right)$
memory	$\mathcal{O}\left(\frac{mn}{P} + n^2\right)$	$\mathcal{O}(\frac{mn}{P})$	$\mathcal{O}(\frac{mn}{P})$	$\mathcal{O}\left(\left(\frac{n^2 m}{P}\right)^{\frac{2}{3}}\right)$

Algorithmic cost analysis: CA-CQR2 vs. competition

CA-CQR2's cost expression expresses tunable tradeoffs

$$T^{\alpha-\beta}_{\text{CA-CQR2}}(m,n,c,d) = \mathcal{O}\left(c^2 \log(d/c)\cdot\alpha + \left(\frac{mn}{dc} + \frac{n^2}{c^2}\right)\cdot\beta + \left(\frac{mn^2}{c^2 d} + \frac{n^3}{c^3}\right)\cdot\gamma\right)$$

Requiring each processor to own a square submatrix ($\frac{m}{d} = \frac{n}{c}$) and enforcing $P = c^2 d$,
CA-CQR2 finds an optimal processor grid that supports minimal communication

	1D Cholesky-QR2	**2D ScaLAPACK**	**2D CAQR**	**3D CA-CQR2**
messages	$\mathcal{O}(\log P)$	$\mathcal{O}(n \log P)$	$\mathcal{O}\left(\sqrt{P}\log^2 P\right)$	$\mathcal{O}\left(\left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P\right)$
words	$\mathcal{O}\left(n^2\right)$	$\mathcal{O}(\frac{mn}{\sqrt{P}})$	$\mathcal{O}(\frac{mn}{\sqrt{P}})$	$\mathcal{O}\left(\left(\frac{n^2 m}{P}\right)^{\frac{2}{3}}\right)$
flops	$\mathcal{O}\left(\frac{n^2 m}{P} + n^3\right)$	$\mathcal{O}(\frac{mn^2}{P})$	$\mathcal{O}(\frac{mn^2}{P})$	$\mathcal{O}\left(\frac{n^2 m}{P}\right)$
memory	$\mathcal{O}\left(\frac{mn}{P} + n^2\right)$	$\mathcal{O}(\frac{mn}{P})$	$\mathcal{O}(\frac{mn}{P})$	$\mathcal{O}\left(\left(\frac{n^2 m}{P}\right)^{\frac{2}{3}}\right)$

Minimal communication cost in a QR factorization is reflected by the surface area of the cubic volume of $\mathcal{O}(mn^2/P)$ computation

Implementation and Experiment setup

We factor $m \times n$ matrices with $m \gg n$ to highlight the effect CA-CQR2's communication reduction and algorithmic tradeoffs have on performance

[1] Intel Knights Landing (KNL) cluster at TACC

[2] Cray XE/XK hybrid machine at NCSA

We factor $m \times n$ matrices with $m \gg n$ to highlight the effect CA-CQR2's communication reduction and algorithmic tradeoffs have on performance

[1] Intel Knights Landing (KNL) cluster at TACC

[2] Cray XE/XK hybrid machine at NCSA

Implementation and Experiment setup

We factor $m \times n$ matrices with $m \gg n$ to highlight the effect CA-CQR2's communication reduction and algorithmic tradeoffs have on performance

Scaling studies highlight **interplay between CA-CQR2's increased arithmetic intensity and an architecture's machine balance**

- ratio of peak-flops to network bandwidth is 8x higher on Stampede2[1] than BlueWaters[2]

[1] Intel Knights Landing (KNL) cluster at TACC

[2] Cray XE/XK hybrid machine at NCSA

Implementation and Experiment setup

We factor $m \times n$ matrices with $m \gg n$ to highlight the effect CA-CQR2's communication reduction and algorithmic tradeoffs have on performance

Scaling studies highlight **interplay between CA-CQR2's increased arithmetic intensity and an architecture's machine balance**

- ratio of peak-flops to network bandwidth is 8x higher on Stampede2[1] than BlueWaters[2]

We show only the **most-performant variants at each node count** of CA-CQR2 and ScaLAPACK's PGEQRF

- ScaLAPACK tuned over 2D processor grid dimensions and block sizes
- CA-CQR2 tuned over processor grid dimensions d and c
- each tested/tuned over a number of resource configurations
- both algorithms use Householder's flop cost in determining performance

[1] Intel Knights Landing (KNL) cluster at TACC
[2] Cray XE/XK hybrid machine at NCSA

Table: Strong scaling: CA-CQR2 performance relative to ScaLAPACK

	m/n	computation	512 PEs	1024 PEs	2048 PEs	4096 PEs	8192 PEs	16384 PEs	32768 PEs	65536 PEs
BlueWaters	4096	2.00x	1.01x	0.88x	0.70x	0.62x	0.62x	0.73x	1.00x	-
BlueWaters	512	2.00x	0.51x	0.48x	0.51x	0.56x	0.66	0.86x	1.36x	-
BlueWaters	64	2.02x	0.51x	0.53x	0.53x	0.61x	0.73x	0.91x	0.92	-
BlueWaters	8	2.20x	0.53x	0.54x	0.55x	0.72x	0.75x	0.67x	0.47x	-
Blue Waters	1	4.25x	0.26x	0.21x	0.18x	0.27x	0.21x	0.13x	0.13x	-
Stampede2	4096	2.00x	-	-	-	0.70x	1.02x	1.27x	1.72x	3.13x
Stampede2	512	2.00x	-	-	-	0.52x	0.99x	1.47x	2.01x	3.34x
Stampede2	64	2.02x	-	-	-	0.77x	1.19x	1.59x	1.82x	2.61x
Stampede2	8	2.20x	-	-	-	0.77x	1.00x	1.21x	1.36x	1.60x
Stampede2	1	4.25x	-	-	-	0.48x	0.55x	0.66x	1.41x	1.02x

Deeper analysis into Strong Scaling results

Table: Strong scaling: CA-CQR2 performance relative to ScaLAPACK

	m/n	computation	512 PEs	1024 PEs	2048 PEs	4096 PEs	8192 PEs	16384 PEs	32768 PEs	65536 PEs
BlueWaters	4096	2.00x	1.01x	0.88x	0.70x	0.62x	0.62x	0.73x	1.00x	-
BlueWaters	512	2.00x	0.51x	0.48x	0.51x	0.56x	0.66	0.86x	1.36x	-
BlueWaters	64	2.02x	0.51x	0.53x	0.53x	0.61x	0.73x	0.91x	0.92	-
BlueWaters	8	2.20x	0.53x	0.54x	0.55x	0.72x	0.75x	0.67x	0.47x	-
Blue Waters	1	4.25x	0.26x	0.21x	0.18x	0.27x	0.21x	0.13x	0.13x	-
Stampede2	4096	2.00x	-	-	-	0.70x	1.02x	1.27x	1.72x	3.13x
Stampede2	512	2.00x	-	-	-	0.52x	0.99x	1.47x	2.01x	3.34x
Stampede2	64	2.02x	-	-	-	0.77x	1.19x	1.59x	1.82x	2.61x
Stampede2	8	2.20x	-	-	-	0.77x	1.00x	1.21x	1.36x	1.60x
Stampede2	1	4.25x	-	-	-	0.48x	0.55x	0.66x	1.41x	1.02x

Deeper analysis into Strong Scaling results

Table: Strong scaling: CA-CQR2 performance relative to ScaLAPACK

	m/n	computation	512 PEs	1024 PEs	2048 PEs	4096 PEs	8192 PEs	16384 PEs	32768 PEs	65536 PEs
BlueWaters	4096	2.00x	1.01x	0.88x	0.70x	0.62x	0.62x	0.73x	1.00x	-
BlueWaters	512	2.00x	0.51x	0.48x	0.51x	0.56x	0.66	0.86x	1.36x	-
BlueWaters	64	2.02x	0.51x	0.53x	0.53x	0.61x	0.73x	0.91x	0.92	-
BlueWaters	8	2.20x	0.53x	0.54x	0.55x	0.72x	0.75x	0.67x	0.47x	-
Blue Waters	1	4.25x	0.26x	0.21x	0.18x	0.27x	0.21x	0.13x	0.13x	-
Stampede2	4096	2.00x	-	-	-	0.70x	1.02x	1.27x	1.72x	3.13x
Stampede2	512	2.00x	-	-	-	0.52x	0.99x	1.47x	2.01x	3.34x
Stampede2	64	2.02x	-	-	-	0.77x	1.19x	1.59x	1.82x	2.61x
Stampede2	8	2.20x	-	-	-	0.77x	1.00x	1.21x	1.36x	1.60x
Stampede2	1	4.25x	-	-	-	0.48x	0.55x	0.66x	1.41x	1.02x

Table: Strong scaling: CA-CQR2 performance relative to ScaLAPACK

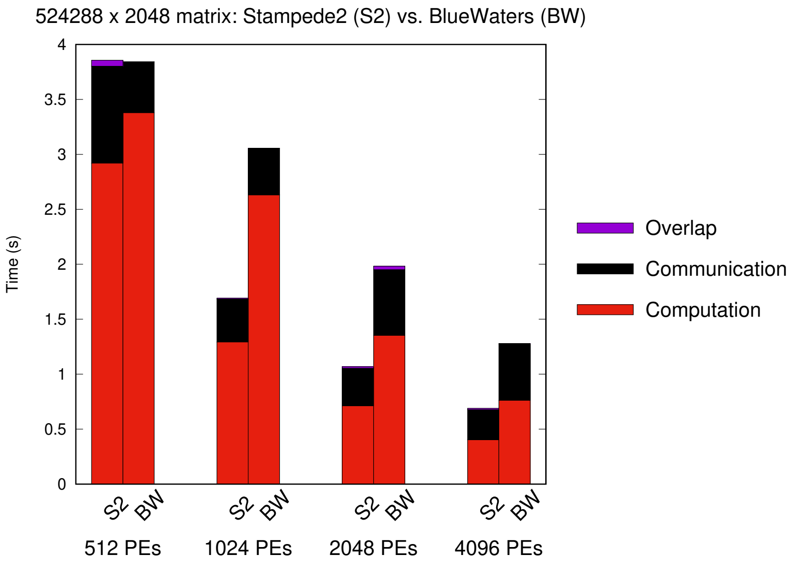
	m/n	computation	512 PEs	1024 PEs	2048 PEs	4096 PEs	8192 PEs	16384 PEs	32768 PEs	65536 PEs
BlueWaters	4096	2.00x	1.01x	0.88x	0.70x	0.62x	0.62x	0.73x	1.00x	-
BlueWaters	512	2.00x	0.51x	0.48x	0.51x	0.56x	0.66	0.86x	1.36x	-
BlueWaters	64	2.02x	0.51x	0.53x	0.53x	0.61x	0.73x	0.91x	0.92	-
BlueWaters	8	2.20x	0.53x	0.54x	0.55x	0.72x	0.75x	0.67x	0.47x	-
Blue Waters	1	4.25x	0.26x	0.21x	0.18x	0.27x	0.21x	0.13x	0.13x	-
Stampede2	4096	2.00x	-	-	-	0.70x	1.02x	1.27x	1.72x	3.13x
Stampede2	512	2.00x	-	-	-	0.52x	0.99x	1.47x	2.01x	3.34x
Stampede2	64	2.02x	-	-	-	0.77x	1.19x	1.59x	1.82x	2.61x
Stampede2	8	2.20x	-	-	-	0.77x	1.00x	1.21x	1.36x	1.60x
Stampede2	1	4.25x	-	-	-	0.48x	0.55x	0.66x	1.41x	1.02x

Deeper analysis into Strong Scaling results

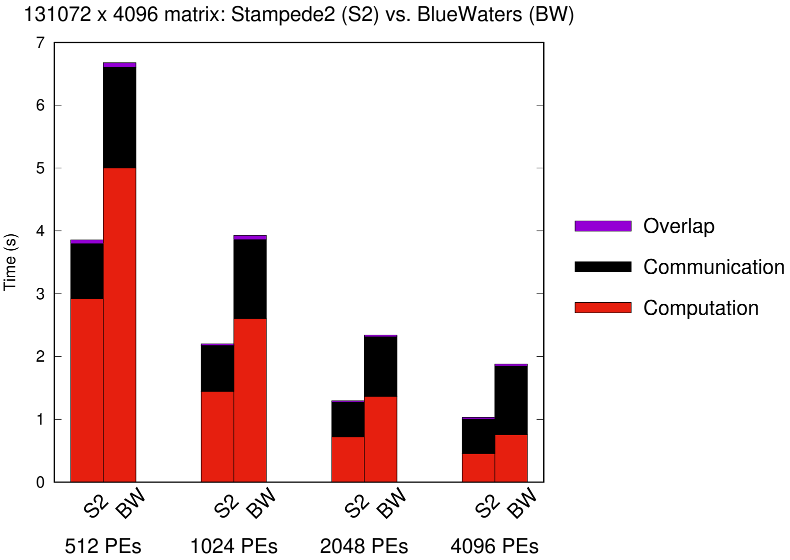
Table: Strong scaling: CA-CQR2 performance relative to ScaLAPACK

	m/n	computation	512 PEs	1024 PEs	2048 PEs	4096 PEs	8192 PEs	16384 PEs	32768 PEs	65536 PEs
BlueWaters	4096	2.00x	1.01x	0.88x	0.70x	0.62x	0.62x	0.73x	1.00x	-
BlueWaters	512	2.00x	0.51x	0.48x	0.51x	0.56x	0.66	0.86x	1.36x	-
BlueWaters	64	2.02x	0.51x	0.53x	0.53x	0.61x	0.73x	0.91x	0.92	-
BlueWaters	8	2.20x	0.53x	0.54x	0.55x	0.72x	0.75x	0.67x	0.47x	-
Blue Waters	1	4.25x	0.26x	0.21x	0.18x	0.27x	0.21x	0.13x	0.13x	-
Stampede2	4096	2.00x	-	-	-	0.70x	1.02x	1.27x	1.72x	3.13x
Stampede2	512	2.00x	-	-	-	0.52x	0.99x	1.47x	2.01x	3.34x
Stampede2	64	2.02x	-	-	-	0.77x	1.19x	1.59x	1.82x	2.61x
Stampede2	8	2.20x	-	-	-	0.77x	1.00x	1.21x	1.36x	1.60x
Stampede2	1	4.25x	-	-	-	0.48x	0.55x	0.66x	1.41x	1.02x

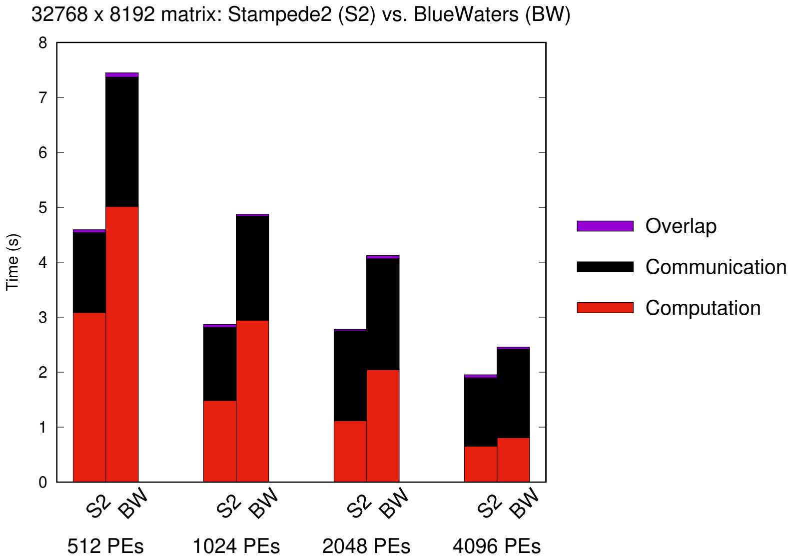
QR Strong scaling critical path analysis



QR Strong scaling critical path analysis



QR Strong scaling critical path analysis



Analysis and Future Work

CA-CQR2's performance improvements over ScaLAPACK on Stampede2 range from **1.1 - 3.3x at 1024 nodes**

[1] Our preprint detailing CA-CQR2 can be found at https://arxiv.org/abs/1710.08471

[2] Our C++ implementation can be found at https://github.com/huttered40/CA-CQR2

Analysis and Future Work

CA-CQR2's performance improvements over ScaLAPACK on Stampede2 range from **1.1 - 3.3x at 1024 nodes**

CA-CQR2 leverages current and future architectural trends

- machines with highest ratio of peak node performance to peak injection bandwidth will benefit most
- asymptotic communication reductuction increasingly evident as we scale, despite overheads in synchronization and computation

[1] Our preprint detailing CA-CQR2 can be found at https://arxiv.org/abs/1710.08471

[2] Our C++ implementation can be found at https://github.com/huttered40/CA-CQR2

CA-CQR2's performance improvements over ScaLAPACK on Stampede2 range from **1.1 - 3.3x at 1024 nodes**

CA-CQR2 leverages current and future architectural trends

- machines with highest ratio of peak node performance to peak injection bandwidth will benefit most
- asymptotic communication reductuction increasingly evident as we scale, despite overheads in synchronization and computation

These results motivate increasingly wide overdetermined systems, a **critical use case for solving linear least squares and eigenvalue problems**

[1] Our preprint detailing CA-CQR2 can be found at https://arxiv.org/abs/1710.08471

[2] Our C++ implementation can be found at https://github.com/huttered40/CA-CQR2

Analysis and Future Work

CA-CQR2's performance improvements over ScaLAPACK on Stampede2 range from **1.1 - 3.3x at 1024 nodes**

CA-CQR2 leverages current and future architectural trends

- machines with highest ratio of peak node performance to peak injection bandwidth will benefit most
- asymptotic communication reductuction increasingly evident as we scale, despite overheads in synchronization and computation

These results motivate increasingly wide overdetermined systems, a **critical use case for solving linear least squares and eigenvalue problems**

Offloading computation to GPUs on XK nodes is a work in progress

[1] Our preprint detailing CA-CQR2 can be found at https://arxiv.org/abs/1710.08471

[2] Our C++ implementation can be found at https://github.com/huttered40/CA-CQR2

Analysis and Future Work

CA-CQR2's performance improvements over ScaLAPACK on Stampede2 range from **1.1 - 3.3x at 1024 nodes**

CA-CQR2 leverages current and future architectural trends

- machines with highest ratio of peak node performance to peak injection bandwidth will benefit most
- asymptotic communication reductuction increasingly evident as we scale, despite overheads in synchronization and computation

These results motivate increasingly wide overdetermined systems, a **critical use case for solving linear least squares and eigenvalue problems**

Offloading computation to GPUs on XK nodes is a work in progress

Our study shows that **communication-optimal parallel QR factorizations can achieve superior performance and scaling up to thousands of nodes**[1] [2]

[1] Our preprint detailing CA-CQR2 can be found at https://arxiv.org/abs/1710.08471

[2] Our C++ implementation can be found at https://github.com/huttered40/CA-CQR2

Cyclops Tensor Framework (CTF)

https://github.com/cyclops-community/ctf

LPNA @CS@Illinois cyclops Cyclops Community

Cyclops Tensor Framework Developer Community

- MPI sparse/dense tensors + OpenMP and CUDA acceleration

```
Matrix<int> A(n, n, AS|SP, World(MPI_COMM_WORLD));
Tensor<float> T(order, is_sparse, dims, syms, ring, world);
T.read(...); T.write(...); T.slice(...); T.permute(...);
```

Cyclops Tensor Framework (CTF)

https://github.com/cyclops-community/ctf

L P N A @CS@Illinois

Cyclops Community

Cyclops Tensor Framework Developer Community

- MPI sparse/dense tensors + OpenMP and CUDA acceleration

```
Matrix<int> A(n, n, AS|SP, World(MPI_COMM_WORLD));
Tensor<float> T(order, is_sparse, dims, syms, ring, world);
T.read(...); T.write(...); T.slice(...); T.permute(...);
```

- parallel contraction/summation/transformation of tensors

```
Z["abij"]   += V["ijab"];                         // C++
W["mnij"]   += 0.5*W["mnef"]*T["efij"];     // C++
M["ij"] += Function<>([](double x){ return 1/x; })(v["j"]);
W.i("mnij") << 0.5*W.i("mnef")*T.i("efij") // Python
[Z,SC,C] = Z.i("abk").svd("abc","kc",rank) // Python
einsum("mnef,efij->mnij",W,T)  // numpy-style Python
```

Cyclops Tensor Framework (CTF)

https://github.com/cyclops-community/ctf

L P N A @CS@Illinois cyclops Cyclops Community

Cyclops Tensor Framework Developer Community

- MPI sparse/dense tensors + OpenMP and CUDA acceleration

```
Matrix<int> A(n, n, AS|SP, World(MPI_COMM_WORLD));
Tensor<float> T(order, is_sparse, dims, syms, ring, world);
T.read(...); T.write(...); T.slice(...); T.permute(...);
```

- parallel contraction/summation/transformation of tensors

```
Z["abij"]   += V["ijab"];                         // C++
W["mnij"]   += 0.5*W["mnef"]*T["efij"];     // C++
M["ij"] += Function<>([](double x){ return 1/x; })(v["j"]);
W.i("mnij") << 0.5*W.i("mnef")*T.i("efij") // Python
[Z,SC,C] = Z.i("abk").svd("abc","kc",rank) // Python
einsum("mnef,efij->mnij",W,T)  // numpy-style Python
```

- Cyclops applications (some using Blue Waters): tensor decomposition, tensor completion, tensor networks (DMRG), quantum chemistry, quantum circuit simulation, graph algorithms, bioinformatics

Acknowledgements

We'd also like to acknowledge NCSA and TACC for providing benchmarking resources

- Texas Advanced Computing Center (TACC) via Stampede2[2]
- National Center for Supercomputing Applications (NCSA) via Blue Waters[3]

I'd like to acknowledge the Department of Energy and Krell Institute for supporting this research via awarding me a DOE Computational Science Graduate Fellowship[1]

[1] Grant number DE-SC0019323

[2] Allocation TG-CCR180006

[3] Awards OCI-0725070 and ACI-1238993

Conditional stability of Cholesky-QR2

The Cholesky-QR2 algorithm *can* achieve stability through iterative refinement[1]

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

[2] T. Fukaya et al., "Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices", Arxiv 2018

The Cholesky-QR2 algorithm *can* achieve stability through iterative refinement[1]

$[Q, R] \leftarrow$ **Cholesky-QR2** (A)

$Z, R_1 \leftarrow CQR(A)$
$Q, R_2 \leftarrow CQR(Z)$
$R \leftarrow R_2 R_1$

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

[2] T. Fukaya et al., "Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices", Arxiv 2018

The Cholesky-QR2 algorithm *can* achieve stability through iterative refinement[1]

$[Q, R] \leftarrow$ **Cholesky-QR2** (A)

$Z, R_1 \leftarrow CQR(A)$
$Q, R_2 \leftarrow CQR(Z)$
$R \leftarrow R_2 R_1$

- leverages near-perfect conditioning of Z in a second iteration[1]

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

[2] T. Fukaya et al., "Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices", Arxiv 2018

Conditional stability of Cholesky-QR2

The Cholesky-QR2 algorithm *can* achieve stability through iterative refinement[1]

$[Q, R] \leftarrow$ **Cholesky-QR2** (A)

$Z, R_1 \leftarrow CQR(A)$
$Q, R_2 \leftarrow CQR(Z)$
$R \leftarrow R_2 R_1$

- leverages near-perfect conditioning of Z in a second iteration[1]
- $A = ZR_1 = QR_2R_1$, from $A^T A = R_1^T Z^T Z R_1 = R_1^T R_2^T Q^T Q R_2 R_1$, where R_2 corrects initial R_1

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

[2] T. Fukaya et al., "Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices", Arxiv 2018

Conditional stability of Cholesky-QR2

The Cholesky-QR2 algorithm *can* achieve stability through iterative refinement[1]

$[Q, R] \leftarrow$ **Cholesky-QR2** (A)

$Z, R_1 \leftarrow CQR(A)$
$Q, R_2 \leftarrow CQR(Z)$
$R \leftarrow R_2 R_1$

- leverages near-perfect conditioning of Z in a second iteration[1]
- $A = ZR_1 = QR_2R_1$, from $A^T A = R_1^T Z^T Z R_1 = R_1^T R_2^T Q^T Q R_2 R_1$, where R_2 corrects initial R_1
- numerical breakdown still possible if first iteration loses positive definiteness in $A^T A$ via $\kappa(A) \leq 1/\sqrt{\epsilon}$

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015
[2] T. Fukaya et al., "Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices", Arxiv 2018

The Cholesky-QR2 algorithm *can* achieve stability through iterative refinement[1]

$[Q, R] \leftarrow$ **Cholesky-QR2** (A)

$Z, R_1 \leftarrow CQR(A)$
$Q, R_2 \leftarrow CQR(Z)$
$R \leftarrow R_2 R_1$

- leverages near-perfect conditioning of Z in a second iteration[1]
- $A = ZR_1 = QR_2R_1$, from $A^TA = R_1^TZ^TZR_1 = R_1^TR_2^TQ^TQR_2R_1$, where R_2 corrects initial R_1
- numerical breakdown still possible if first iteration loses positive definiteness in A^TA via $\kappa(A) \leq 1/\sqrt{\epsilon}$

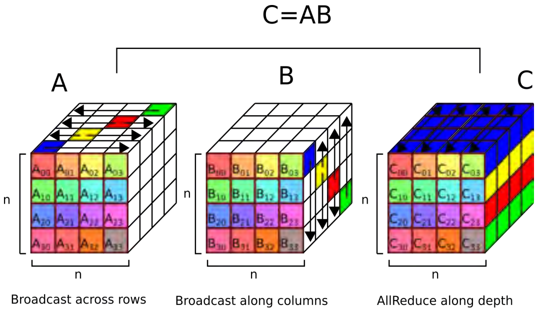
Shifted Cholesky-QR[2] can attain a stable factorization for any matrix $\kappa(A) \leq 1/\epsilon$

- the eigenvalues of A^TA are shifted to prevent loss of positive definiteness
- three Cholesky-QR iterations required, essentially $3-6$x more flops than Householder approaches

[1] Y. Yamamoto et al., "Roundoff Error Analysis of the CholeskyQR2 algorithm", Electron. Trans. Numer. Anal. 2015

[2] T. Fukaya et al., "Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices", Arxiv 2018

Figure: 3D algorithm for square matrix multiplication [1] [2] [3]



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

[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"

We can embed the recursive definitions of Cholesky factorization and triangular inverse to find matrices R, R^{-1}

Tuning the recursion tree yields a tradeoff in horizontal bandwidth and synchronization[1]

$$[L, L^{-1}] \leftarrow \textbf{CholeskyInverse}\,(A)$$

$$\begin{bmatrix} L_{11} & L_{11}^{-1} \end{bmatrix} \leftarrow \textbf{CholeskyInverse}(A_{11})$$
$$L_{21} \leftarrow A_{21} L_{11}^{-T}$$
$$\begin{bmatrix} L_{22} & L_{22}^{-1} \end{bmatrix} \leftarrow \textbf{CholeskyInverse}(A_{22} - L_{21} L_{21}^{T})$$
$$L_{21}^{-1} \leftarrow -L_{22}^{-1} L_{21} L_{11}^{-1}$$

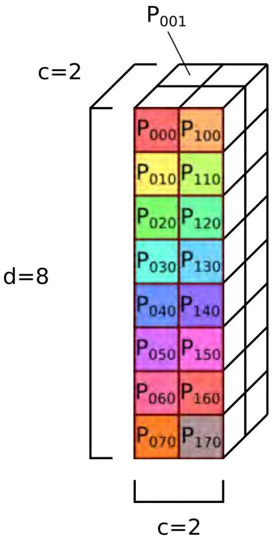
$$T_{\text{CholeskyInverse3D}}(n, P) = \mathcal{O}\left(P^{\frac{2}{3}} \log P \cdot \alpha + \frac{n^2}{P^{\frac{2}{3}}} \cdot \beta + \frac{n^3}{P} \cdot \gamma \right)$$

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

[1] A. Tiskin 2007, "Communication-efficient generic pairwise elimination"

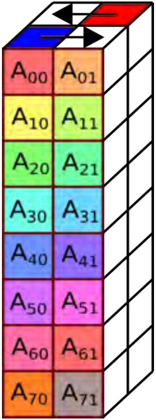
CA-CQR2 – Computation of Gram matrix

Figure: Start with a tunable $c \times d \times c$ processor grid



CA-CQR2 – Computation of Gram matrix

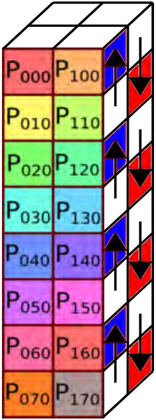
Figure: Broadcast columns of A



Cost: $2\log_2 c \cdot \alpha + \frac{2mn}{dc} \cdot \beta$

CA-CQR2 – Computation of Gram matrix

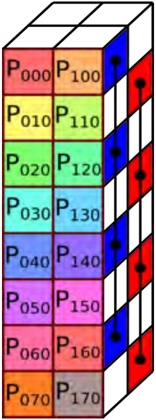
Figure: Reduce contiguous groups of size c



Cost: $2\log_2 c \cdot \alpha + \frac{2n^2}{c^2} \cdot \beta + \frac{n^2}{c^2} \cdot \gamma$

CA-CQR2 – Computation of Gram matrix

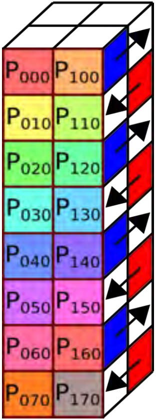
Figure: Allreduce alternating groups of size $\frac{d}{c}$



Cost: $2\log_2 \frac{d}{c} \cdot \alpha + \frac{2n^2}{c^2} \cdot \beta + \frac{n^2}{c^2} \cdot \gamma$

CA-CQR2 – Computation of Gram matrix

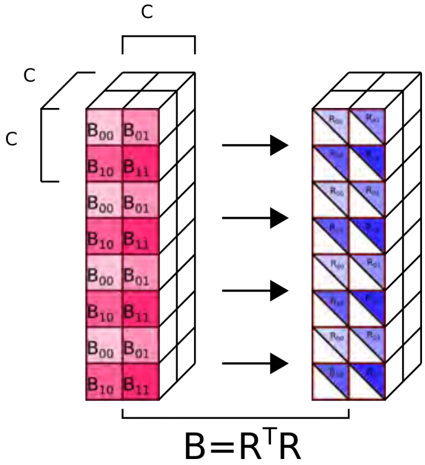
Figure: Broadcast missing pieces of B along depth



Cost: $2\log_2 c \cdot \alpha + \frac{2n^2}{c^2} \cdot \beta$

CA-CQR2 – Computation of CholeskyInverse

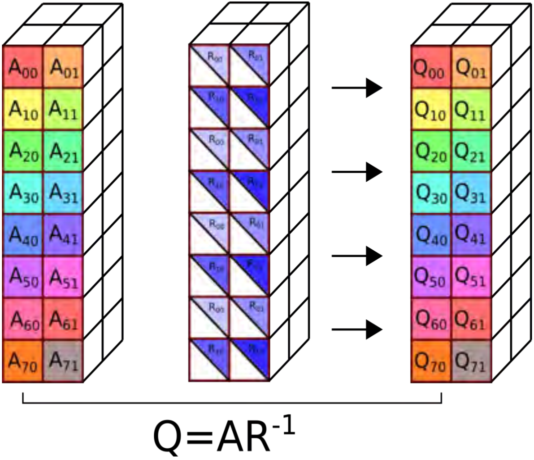
Figure: $\frac{d}{c}$ simultaneous 3D CholeskyInverse on cubes of dimension c



Cost: $\mathcal{O}\left(c^2 \log c^3 \cdot \alpha + \frac{n^2}{c^2} \cdot \beta + \frac{n^3}{c^3} \cdot \gamma\right)$

CA-CQR2 – Computation of triangular solve

Figure: $\frac{d}{c}$ simultaneous 3D matrix multiplication or TRSM on cubes of dimension c



Cost: $\mathcal{O}(\log_2 c^3 \cdot \alpha + \left(\frac{mn}{dc} + \frac{n^2+nc}{c^2}\right) \cdot \beta + \frac{n^2 m}{c^2 d} \cdot \gamma)$

Optimum cost of CholesyQR2_Tunable

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 $c = \left(\frac{Pn}{m}\right)^{\frac{1}{3}}$, $d = \left(\frac{Pm^2}{n^2}\right)^{\frac{1}{3}}$. We can plug these values into the cost of Cholesky-QR2_Tunable to find the optimal cost.

$$
\begin{aligned}
T^{\alpha-\beta}_{\text{Cholesky-QR2\_Tunable}}\left(m, n, \left(\frac{Pn}{m}\right)^{\frac{1}{3}}, \left(\frac{Pm^2}{n^2}\right)^{\frac{1}{3}}\right) &= \mathcal{O}\Bigg(\left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P \cdot \alpha \\
&+ \frac{\left(\frac{Pn}{m}\right)^{\frac{1}{3}} mn + n^2 \left(\frac{Pm^2}{n^2}\right)^{\frac{1}{3}}}{\left(\frac{Pm^2}{n^2}\right)^{\frac{1}{3}} \left(\frac{Pn}{m}\right)^{\frac{2}{3}}} \cdot \beta + \frac{n^3 \left(\frac{Pm^2}{n^2}\right)^{\frac{1}{3}} + n^2 m \left(\frac{Pn}{m}\right)^{\frac{1}{3}}}{\left(\frac{Pn}{m}\right)\left(\frac{Pm^2}{n^2}\right)^{\frac{1}{3}}} \cdot \gamma\Bigg) \qquad (1) \\
&= \mathcal{O}\left(\left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P \cdot \alpha + \left(\frac{n^2 m}{P}\right)^{\frac{2}{3}} \cdot \beta + \frac{n^2 m}{P} \cdot \gamma\right)
\end{aligned}
$$

Grid shape	Metric	Cost
optimal	# of messages	$\mathcal{O}\left(\left(\frac{Pn}{m}\right)^{\frac{2}{3}} \log P\right)$
	# of words	$\mathcal{O}\left(\left(\frac{n^2 m}{P}\right)^{\frac{2}{3}}\right)$
	# of flops	$\mathcal{O}\left(\frac{n^2 m}{P}\right)$
	Memory footprint	$\mathcal{O}\left(\left(\frac{n^2 m}{P}\right)^{\frac{2}{3}}\right)$