communication-avoiding Cholesky-QR2 for rectangular matrices (CA-CQR2)

Edward Hutter and Edgar Solomonik

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

IPDPS 2019

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: $\alpha \gg \beta \gg \gamma$

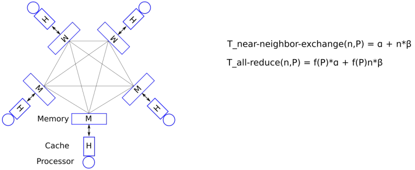


Figure: Horizontal (internode network) communication along critical path

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

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

Contributions: 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 matrices
- requires $\mathcal{O}\left(\left(\frac{\mathbf{Pm^2}}{\mathbf{n^2}}\right)^{\frac{1}{6}}\right)$ less communication than known 2D QR algorithms for $m \times n$ matrices across P processes
- obtains 3x speedups over ScaLAPACK on 1024 nodes
- utilizes first distributed-memory implementation of recursive 3D Cholesky factorization

CA-CQR2's asymptotic communication reduction incurs tradeoffs

- increased computation ($2-4$x more flops than Householder QR)
- constrained applicability (matrix must be sufficiently well-conditioned)
- requires $\mathcal{O}\left(\left(\frac{\mathbf{Pm}}{\mathbf{n}}\right)^{\frac{1}{3}}\right)$ more memory than known 2D QR algorithms for $m \times n$ matrices across P processes

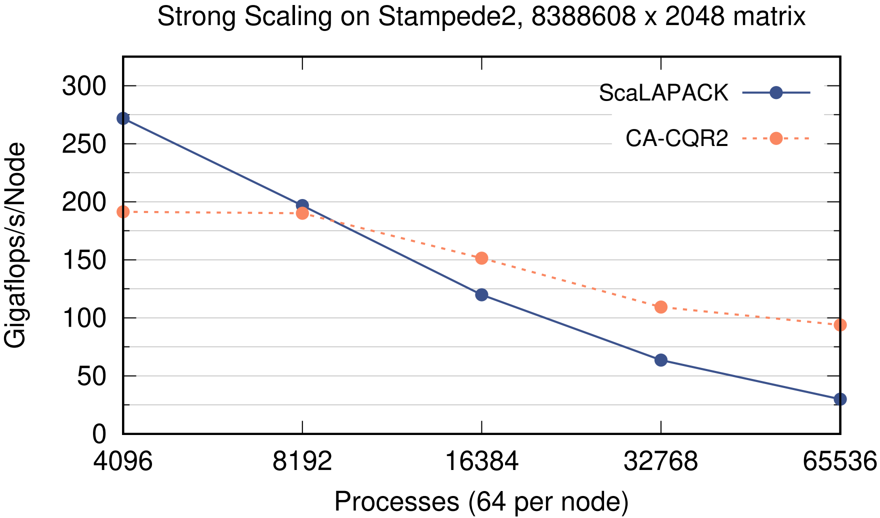


Figure: Strong scaling for $m \times n$ matrices

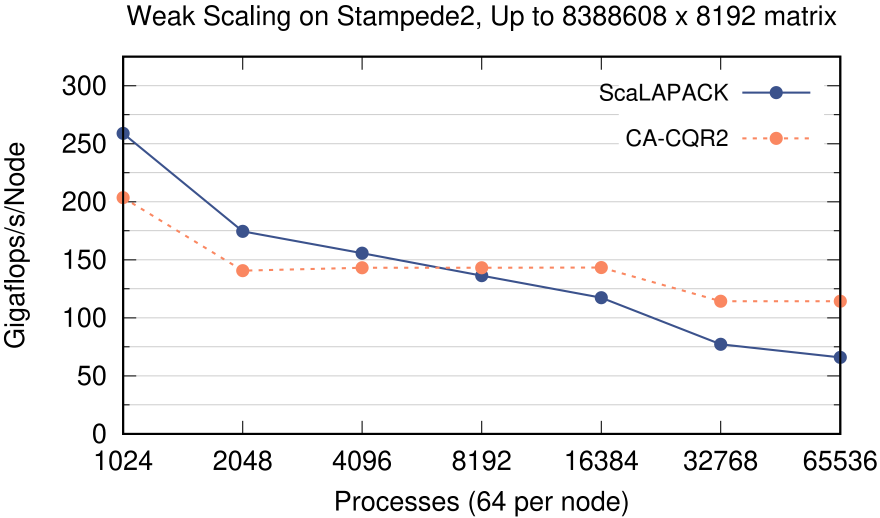


Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count

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

$\alpha - \beta$ model captures communication (β) and synchronization (α) costs over P processors

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)

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

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

[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

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)

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

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 $A^TA = R_1^TZ^TZR_1 = R_1^TR_2^TQ^TQR_2R_1$, 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 $\kappa(A) \leq 1/\epsilon$

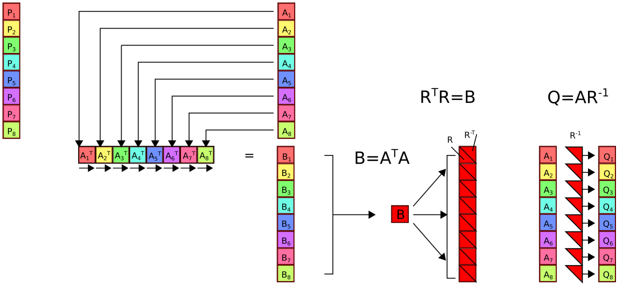
- the eigenvalues of A^TA are shifted to prevent loss of positive definiteness
- three Cholesky-QR iterations required, essentially $3-6$x more flops than Householder approaches

[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

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**

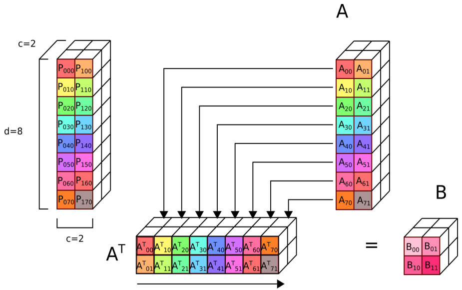
[1] T. Fukaya et al., "CholeskyQR2: A communication-avoiding algorithm", ScalA 2014

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

A recursion tree for recursive Cholesky factorization and triangular inverse yields a tradeoff in communication and synchronization[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 $A^T A$



[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

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

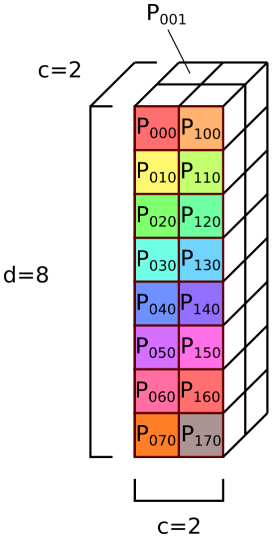
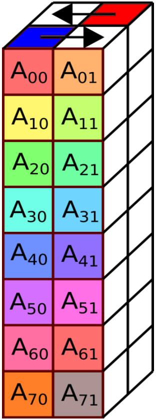
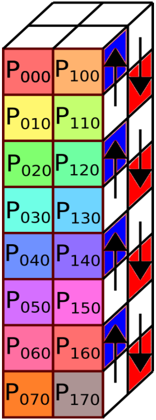


Figure: Broadcast columns of A



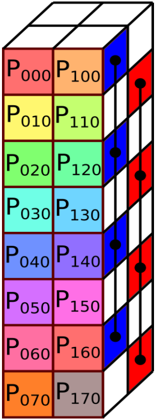
Cost: $2\log_2 c \cdot \alpha + \frac{2mn}{dc} \cdot \beta$

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

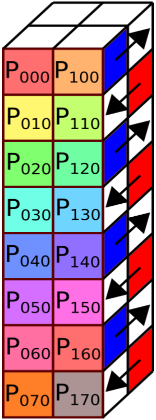
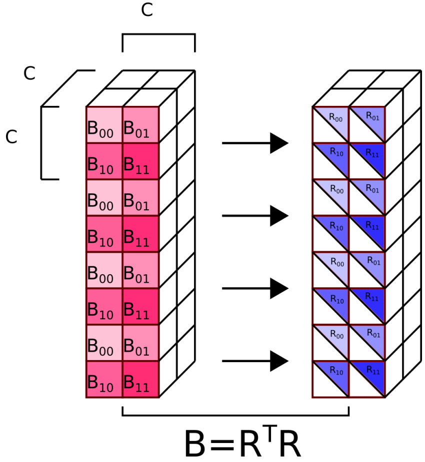


Figure: Broadcast missing pieces of B along depth

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

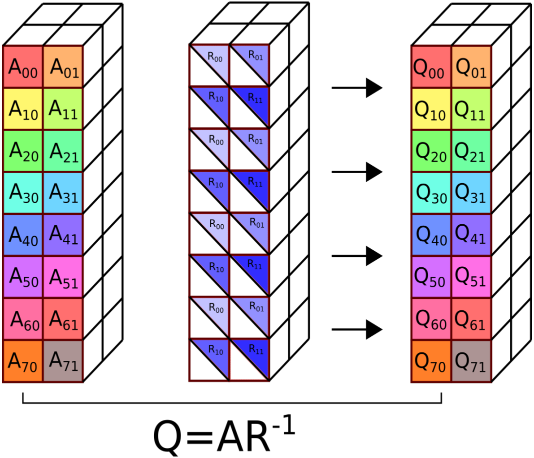
Figure: $\frac{d}{c}$ simultaneous 3D CholeskyInverse on cubes of dimension c



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

CA-CQR2 – Computation of triangular solve

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



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

CA-CQR2's cost expression expresses tunable tradeoffs

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

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

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

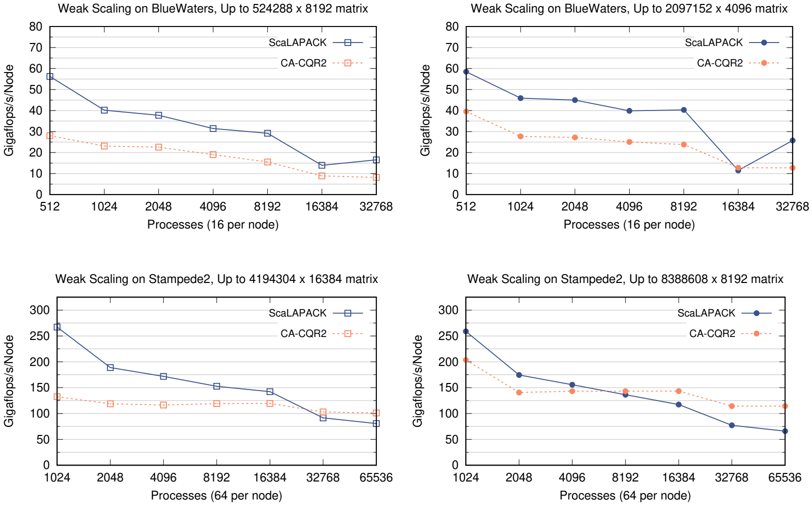
- ratio of peak-flops to network bandwidth is 8x higher on Stampede2[1] than BlueWaters[2]

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

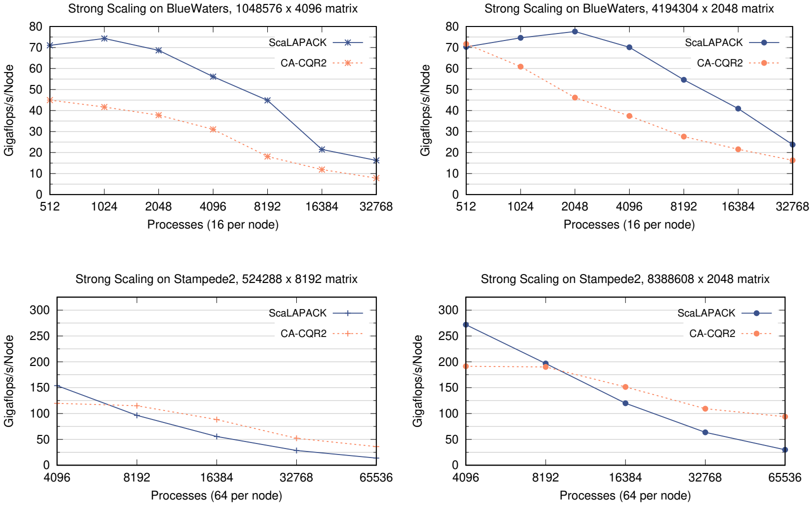
[1] Intel Knights Landing (KNL) cluster at TACC
[2] Cray XE/XK hybrid machine at NCSA

Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count



QR Strong scaling on Stampede2 and Blue Waters

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



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**

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

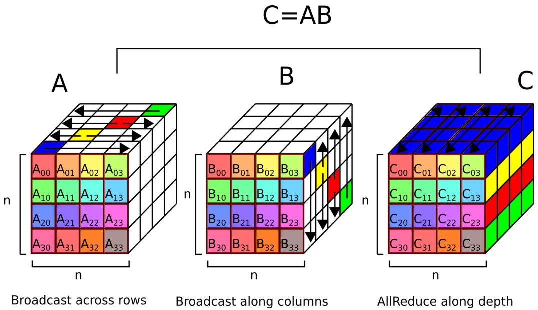
I'd like to acknowledge the **Department of Energy** and **Krell Institute** for supporting this research via awarding me a **DOE Computational Science Graduate Fellowship**[1]

We'd also like to acknowledge a number of computing centers for providing benchmarking resources

- Texas Advanced Computing Center (TACC) via Stampede2[2]
- National Center for Supercomputing Applications (NCSA) via Blue Waters[3]
- Argonne Leadership Computing Facility (Cetus,Mira,Theta) for preliminary benchmarking

[1] Grant number DE-SC0019323
[2] Allocation TG-CCR180006
[3] Awards OCI-0725070 and ACI-1238993

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



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

[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]

$[L, L^{-1}] \leftarrow$ **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 over a 3D processor grid

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

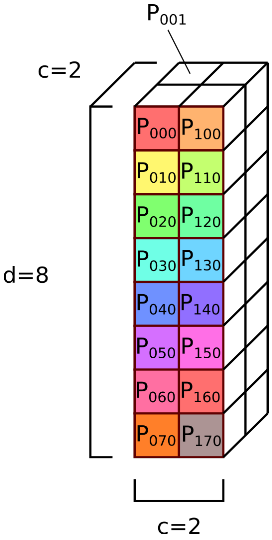


Figure: Compute Gram matrix $A^T A$

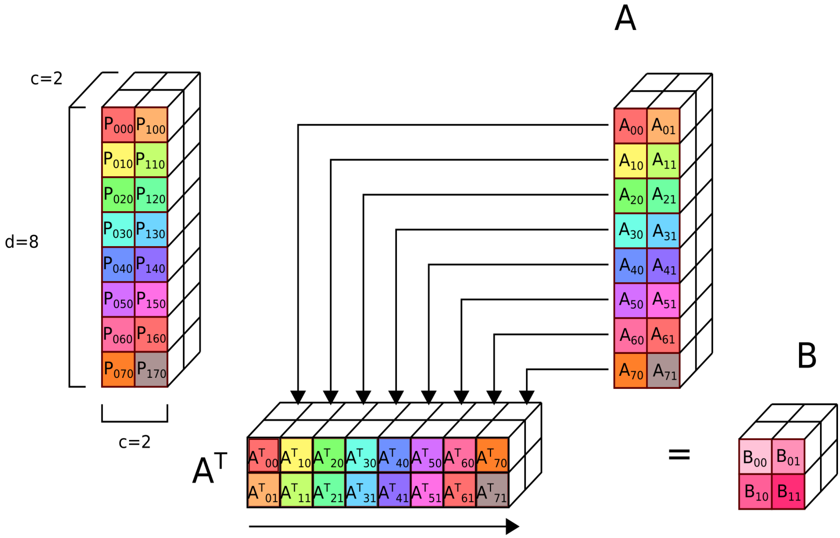


Figure: Compute 3D CholeskyInverse on processor grid cubes of dimension c

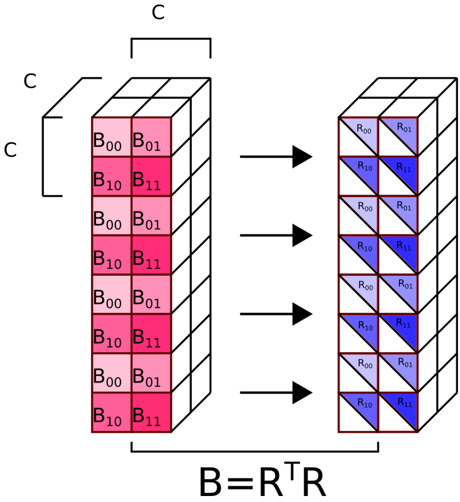
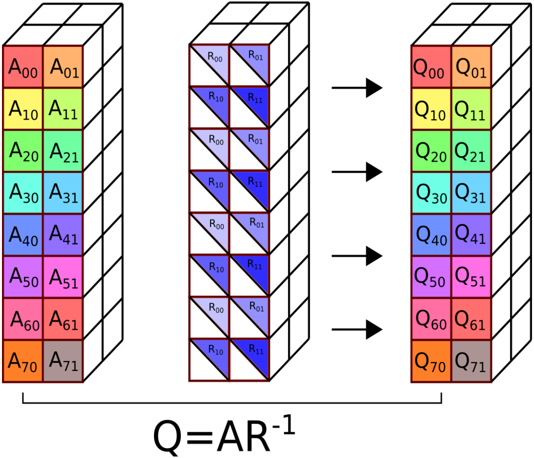


Figure: Compute 3D matrix multiplication or 3D TRSM on processor grid cubes of dimension c



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.

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

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

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	$\mathcal{O}\left(\left(\frac{n^2 m}{P}\right)^{\frac{2}{3}}\right)$

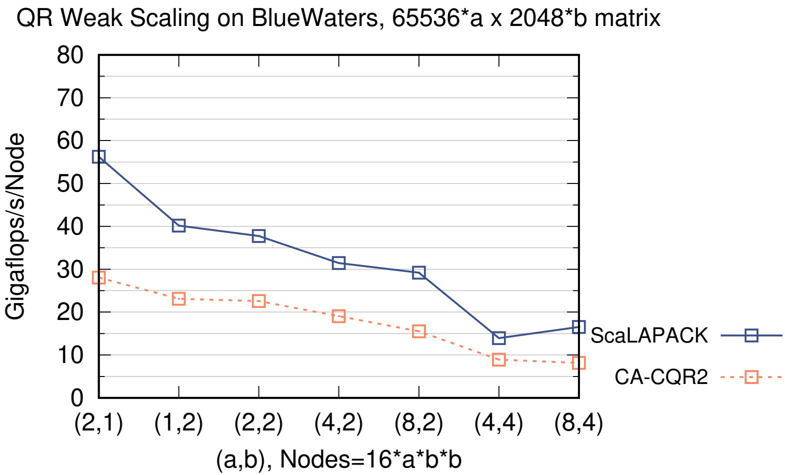


Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count

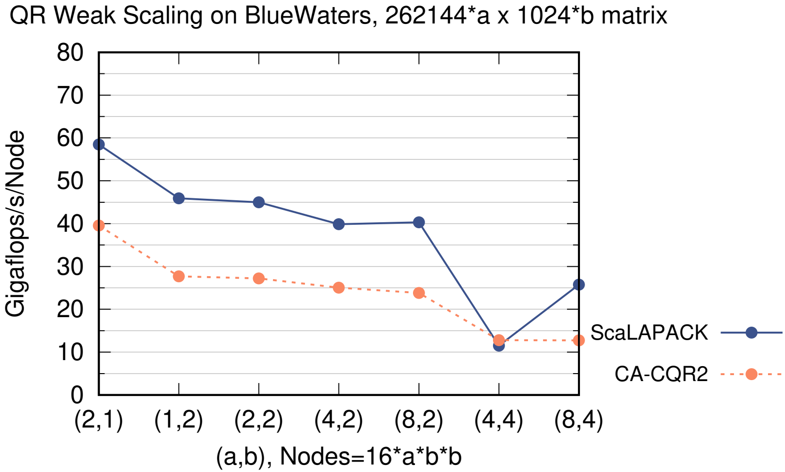


Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count

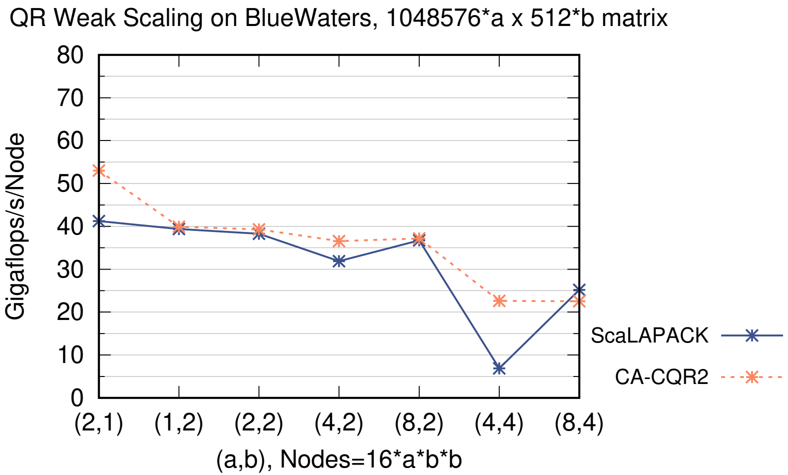


Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count

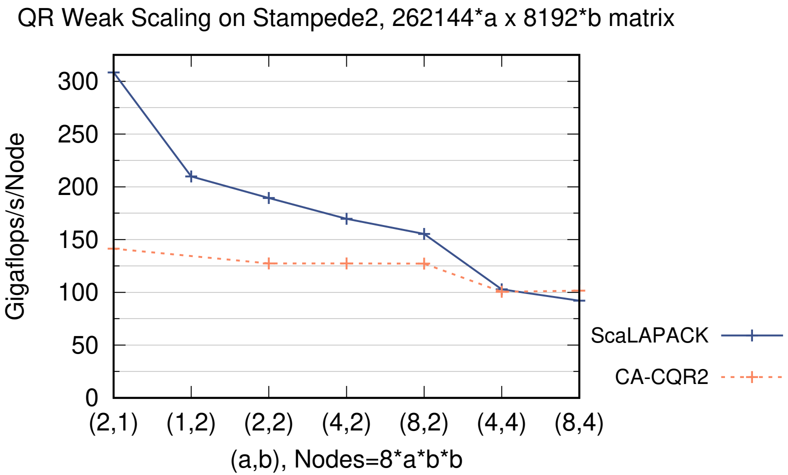


Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count

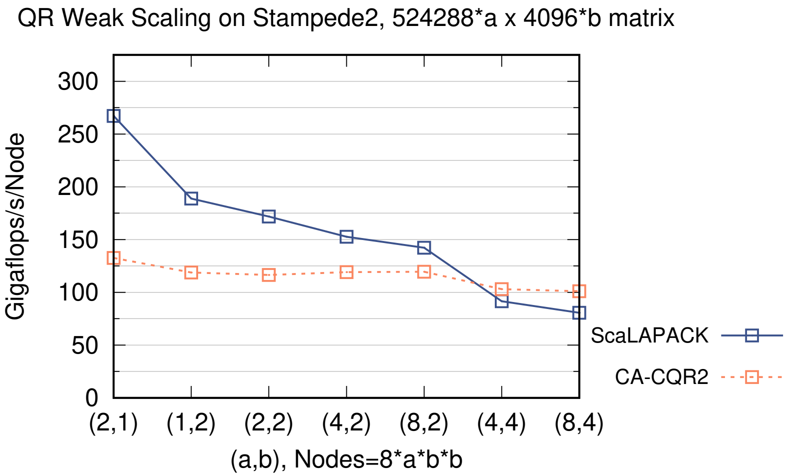


Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count

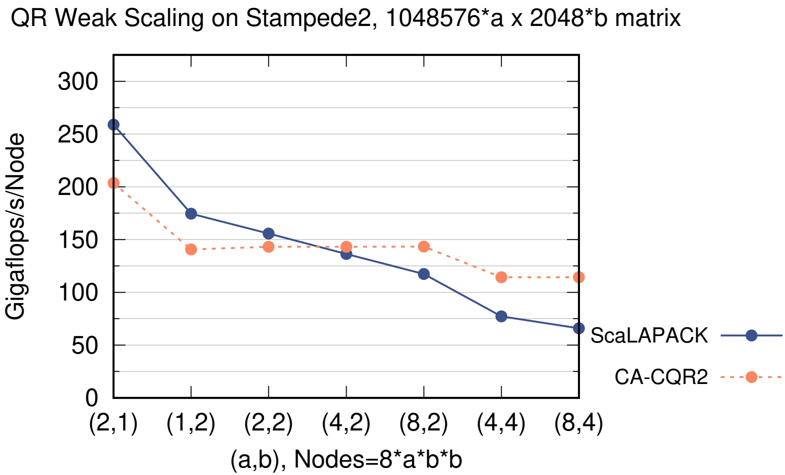


Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count

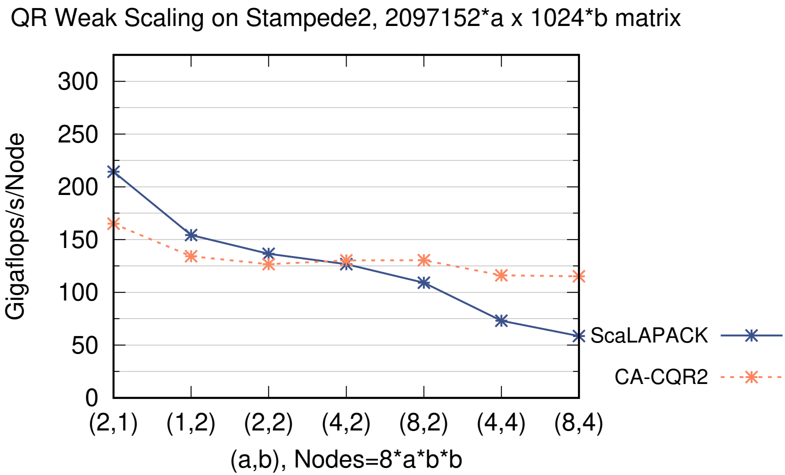


Figure: Weak scaling for $m \times n$ matrices so mn^2 scales linearly with node count

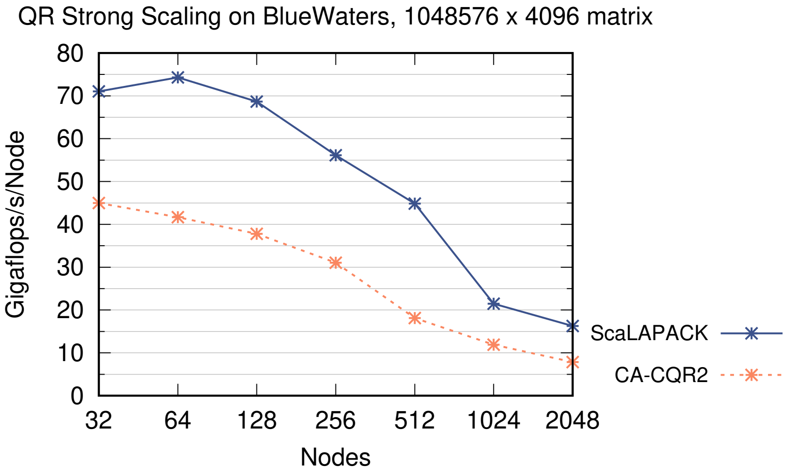


Figure: Strong scaling for QR factorization

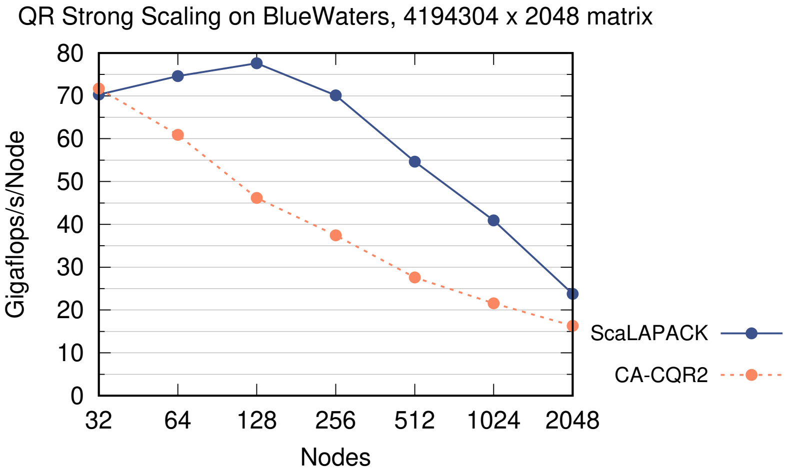


Figure: Strong scaling for QR factorization

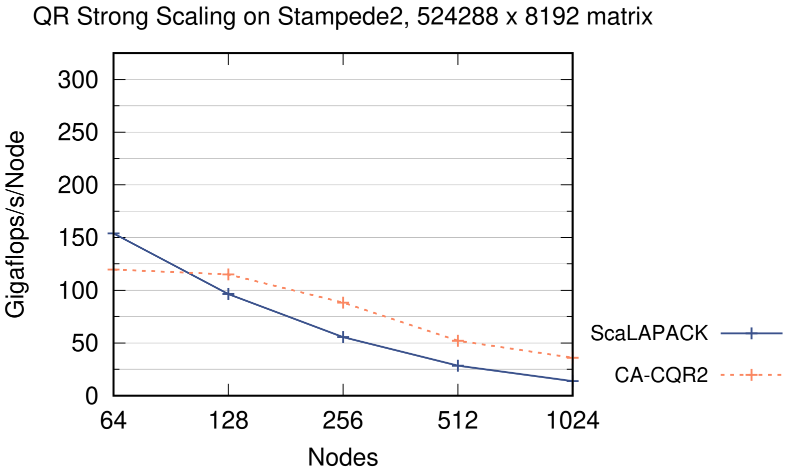


Figure: Strong scaling for QR factorization

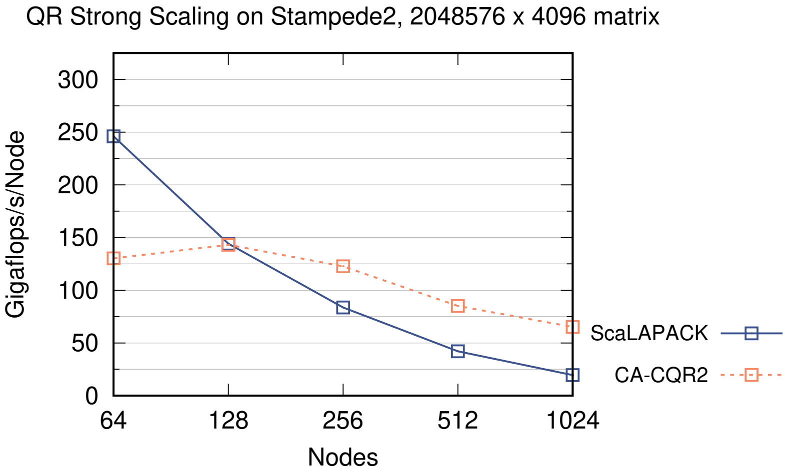


Figure: Strong scaling for QR factorization

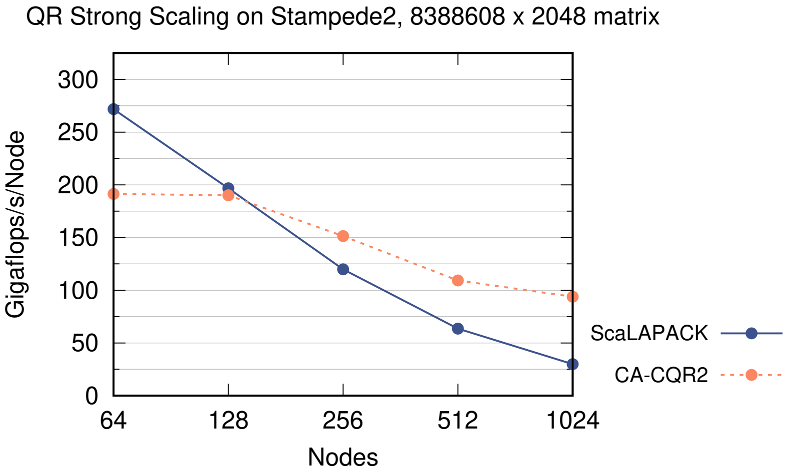


Figure: Strong scaling for QR factorization

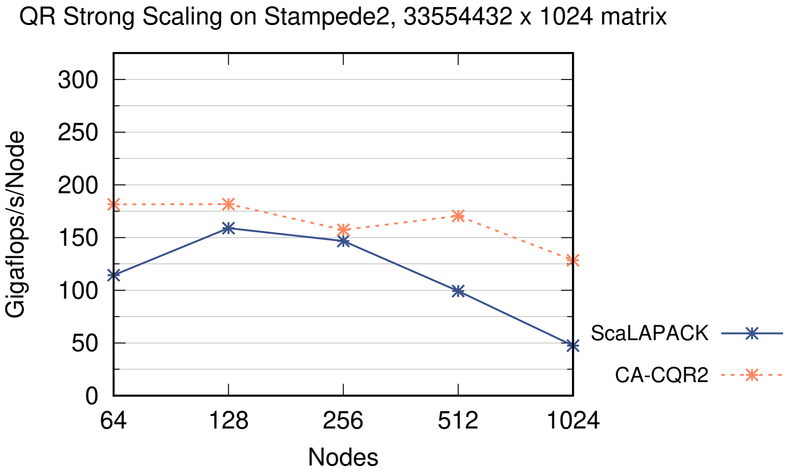


Figure: Strong scaling for QR factorization

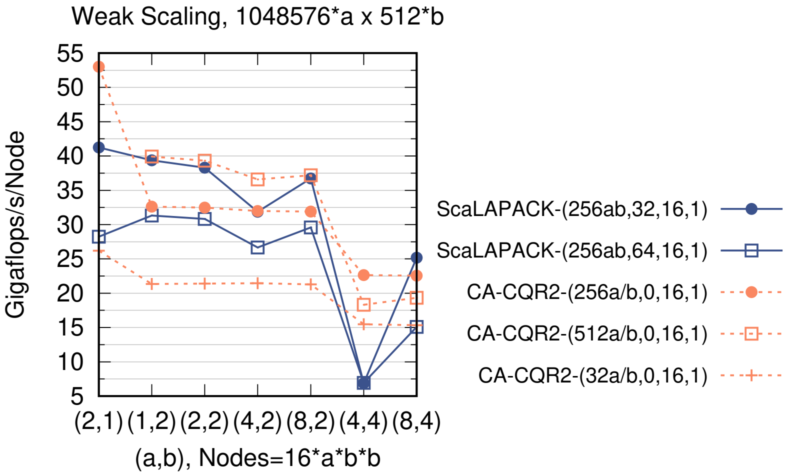


Figure: Weak scaling for matrices with dimensions given in legend

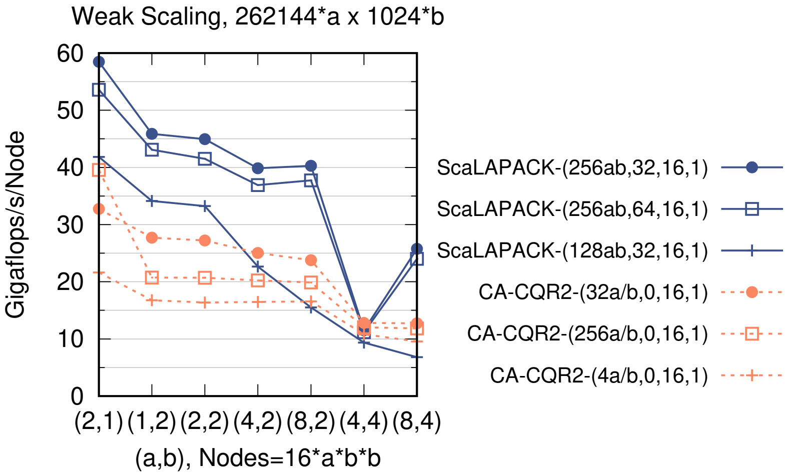


Figure: Weak scaling for matrices with dimensions given in legend

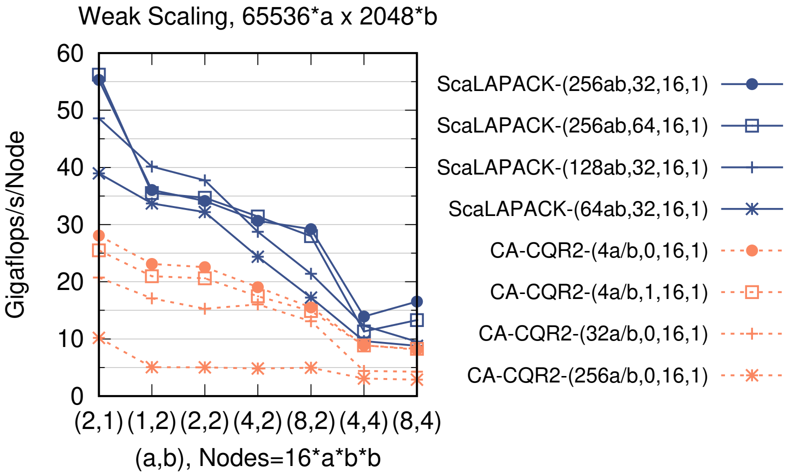


Figure: Weak scaling for matrices with dimensions given in legend

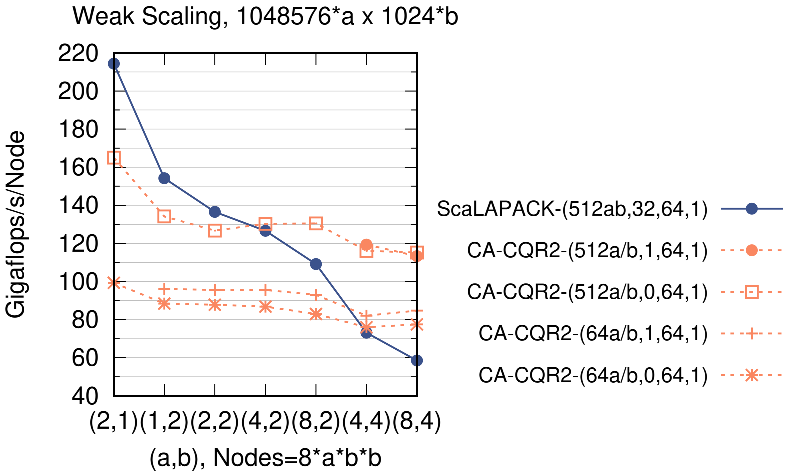


Figure: Weak scaling for matrices with dimensions given in legend

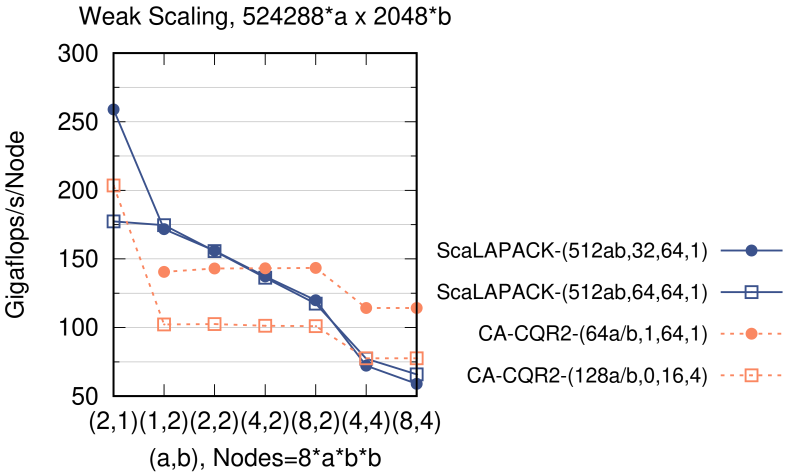


Figure: Weak scaling for matrices with dimensions given in legend

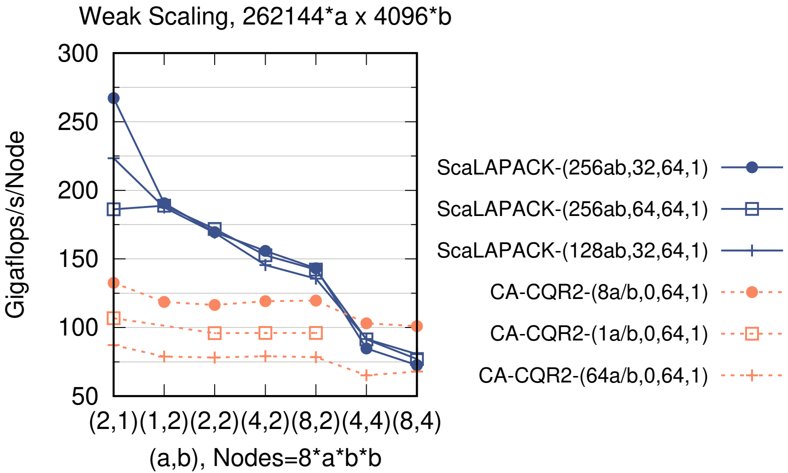


Figure: Weak scaling for matrices with dimensions given in legend

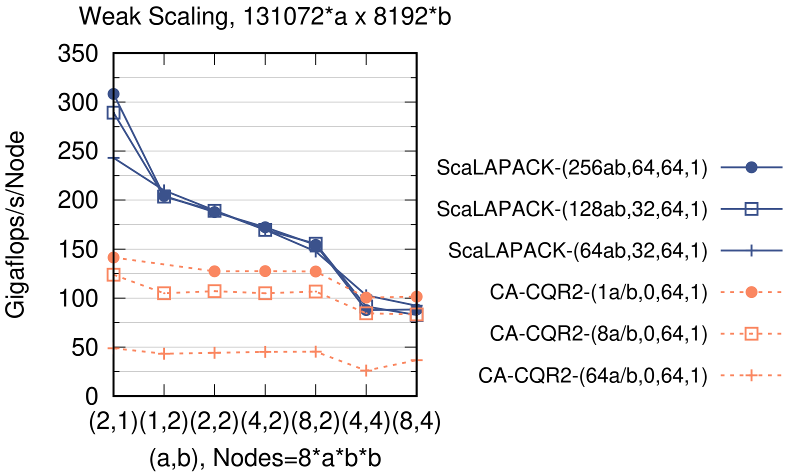


Figure: Weak scaling for matrices with dimensions given in legend

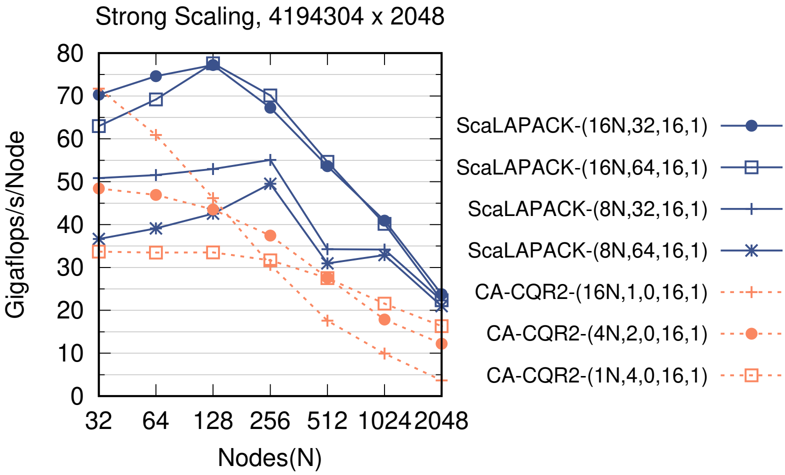


Figure: Strong scaling for matrices with dimensions given in legend

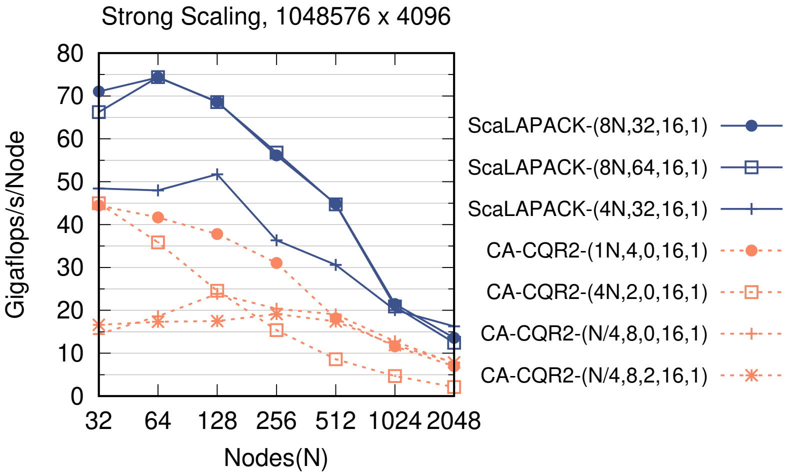


Figure: Strong scaling for matrices with dimensions given in legend

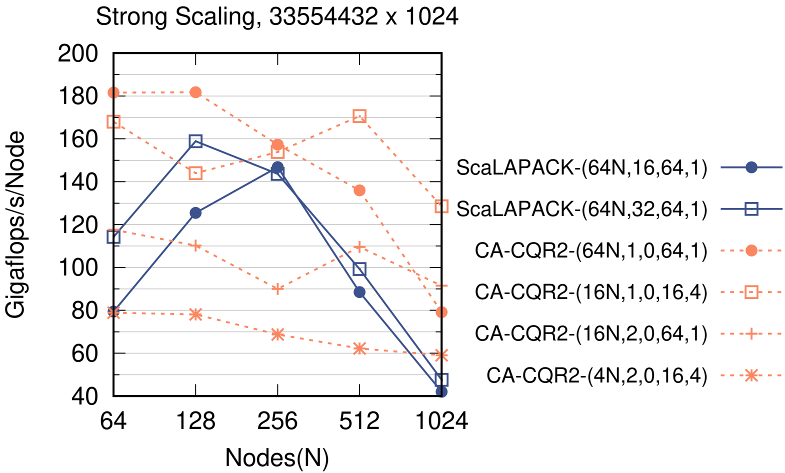


Figure: Strong scaling for matrices with dimensions given in legend

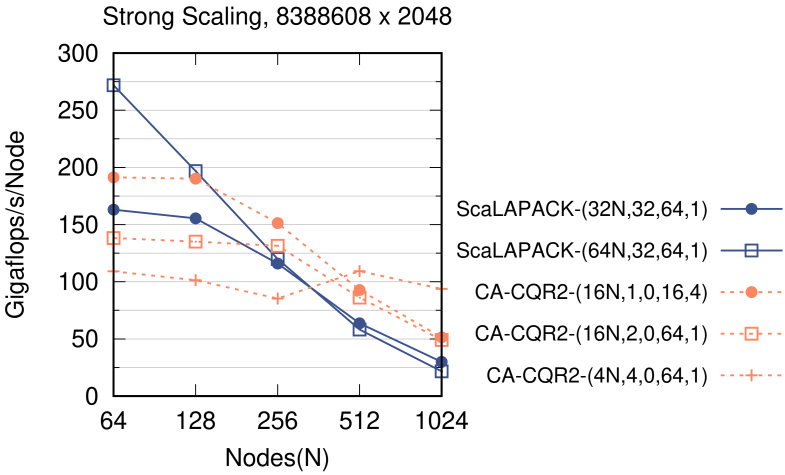


Figure: Strong scaling for matrices with dimensions given in legend

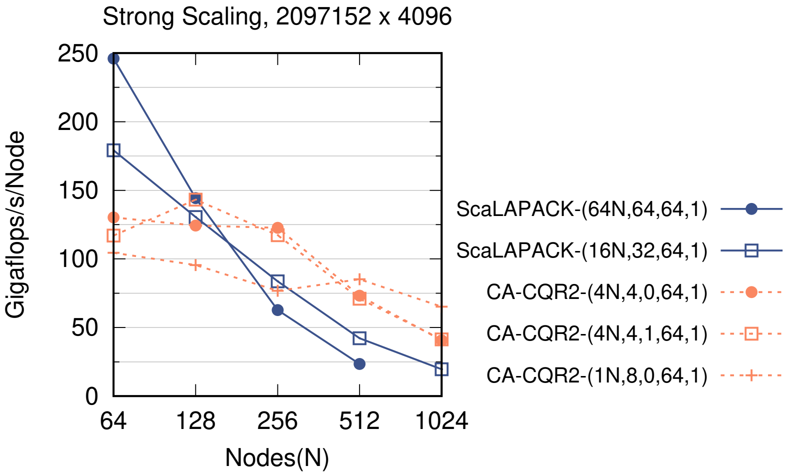


Figure: Strong scaling for matrices with dimensions given in legend

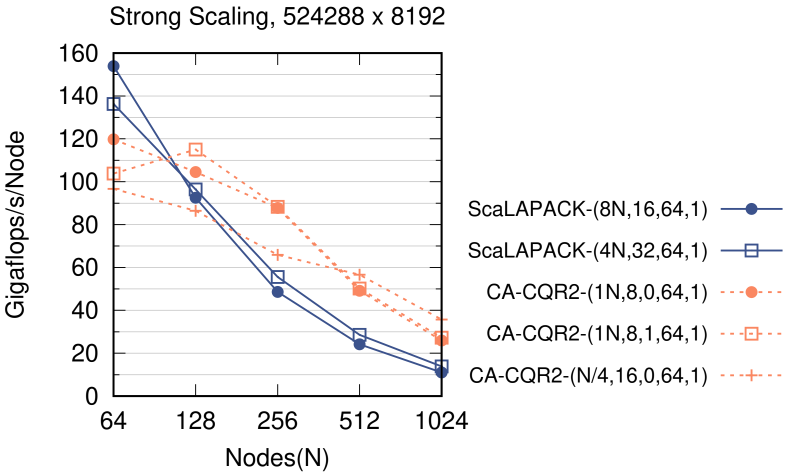


Figure: Strong scaling for matrices with dimensions given in legend