SAND2014-15743PE

Algorithmic Improvements for QR Decomposition

Grey Ballard

Sandia National Laboratories

July 18, 2014

Duke University

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

(SAND2014-XXXX)

Summary

- Communication is Expensive
  - in terms of time and energy
- Avoiding Communication
  - some communication is necessary: we can prove **lower bounds**
  - theoretical analysis identifies suboptimal algorithms and spurs **algorithmic innovation**
  - minimizing communication leads to speedups in practice
- We'll focus on QR decompositions in this talk
  - main new kernel is "Tall-Skinny QR (TSQR)" algorithm
  - we'll discuss some recent improvements based on TSQR

Let's start with dense matrix multiplication...

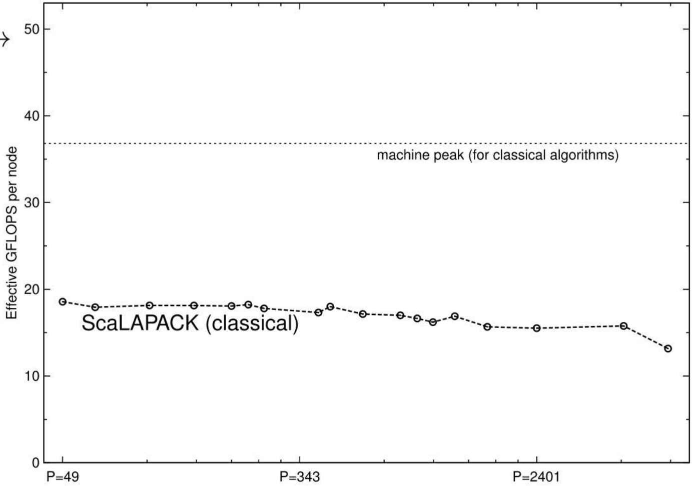
- one of the most fundamental computations
- highly tuned on most architectures
- generally considered to be "compute-bound"

Can we improve performance with better algorithms?

Can we improve dense matrix multiplication?

Here's a strong-scaling plot, for fixed matrix dimension: $n = 94{,}080$

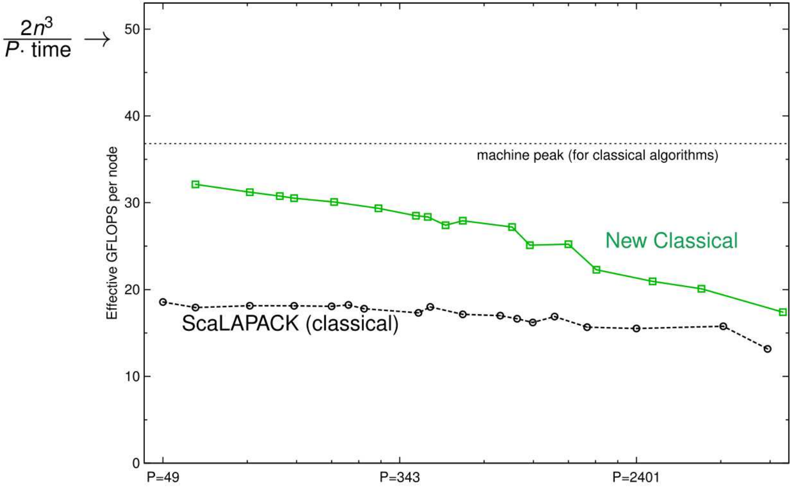
$\frac{2n^3}{P \cdot \text{time}} \rightarrow$



benchmarked on a Cray XT4

Can we improve dense matrix multiplication?

Here's a strong-scaling plot, for fixed matrix dimension: $n = 94{,}080$

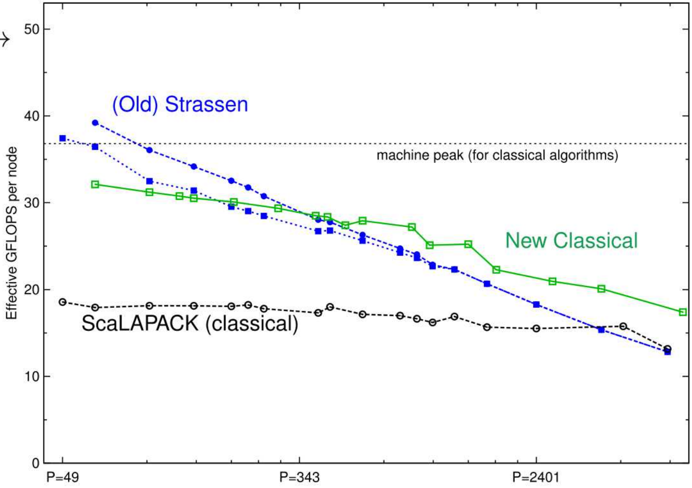


benchmarked on a Cray XT4

Can we improve dense matrix multiplication?

Here's a strong-scaling plot, for fixed matrix dimension: $n = 94{,}080$

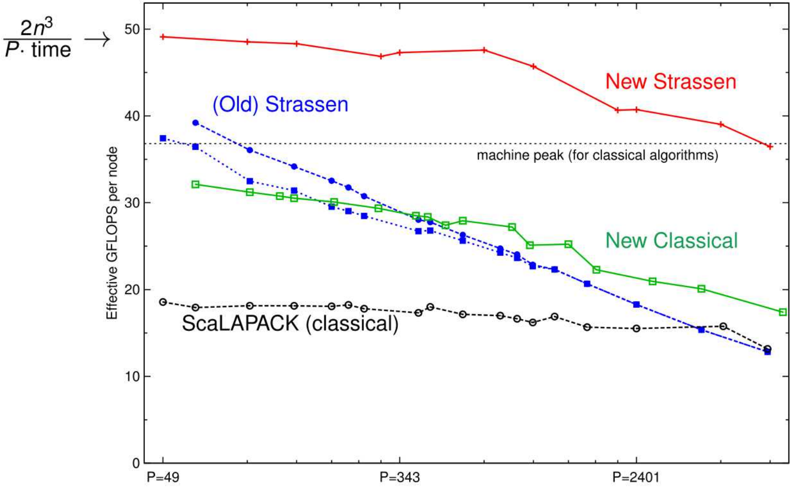
$\frac{2n^3}{P \cdot \text{time}} \rightarrow$



benchmarked on a Cray XT4

Can we improve dense matrix multiplication?

Here's a strong-scaling plot, for fixed matrix dimension: $n = 94{,}080$



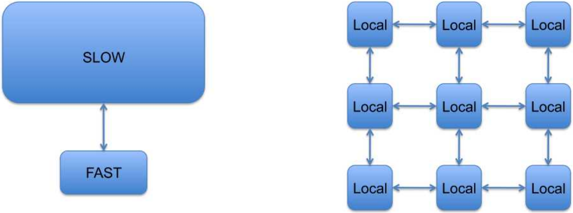
benchmarked on a Cray XT4

We must consider communication

By *communication*, I mean

- moving data within memory hierarchy on a sequential computer
- moving data between processors on a parallel computer

For high-level analysis, we'll use these simple memory models:



Sequential

Parallel

Runtime Model

Measure computation in terms of # *flops* performed

Time per flop: γ

Measure communication in terms of # *words* communicated

Time per word: β

Total running time of an algorithm (ignoring overlap):

$$\gamma \cdot (\text{\# flops}) + \beta \cdot (\text{\# words})$$

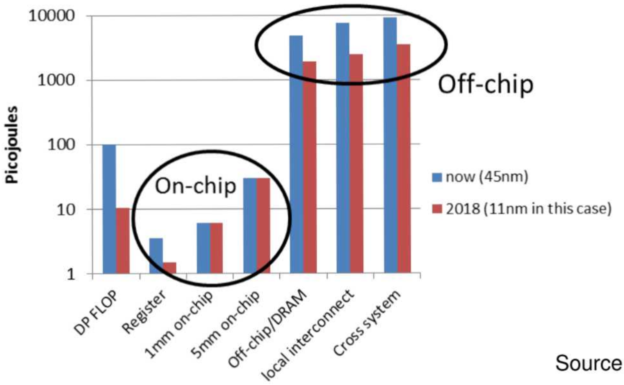
$\beta \gg \gamma$ as measured in time *and* energy, and the relative cost of communication is increasing

Why avoid communication

Annual Improvements in Time

Flop rate γ	DRAM Bandwidth β	Network Bandwidth β
59% per year	23% per year	26% per year

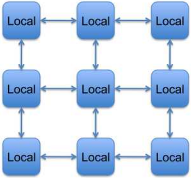
Energy cost comparisons



Source: John Shalf

Costs of matrix multiplication algorithms

$n =$ matrix dimension
$P =$ number of processors
$M =$ size of the local memory

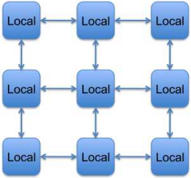


	Computation	Communication
"2D" Algorithm (ScaLAPACK)	$O\left(\frac{n^3}{P}\right)$	$O\left(\frac{n^2}{\sqrt{P}}\right)$
Lower Bound	$\Omega\left(\frac{n^3}{P}\right)$	$\Omega\left(\frac{n^3}{P\sqrt{M}}\right)$

- 2D algorithm is suboptimal if $M \gg \frac{n^2}{P}$ (extra memory available)

Costs of matrix multiplication algorithms

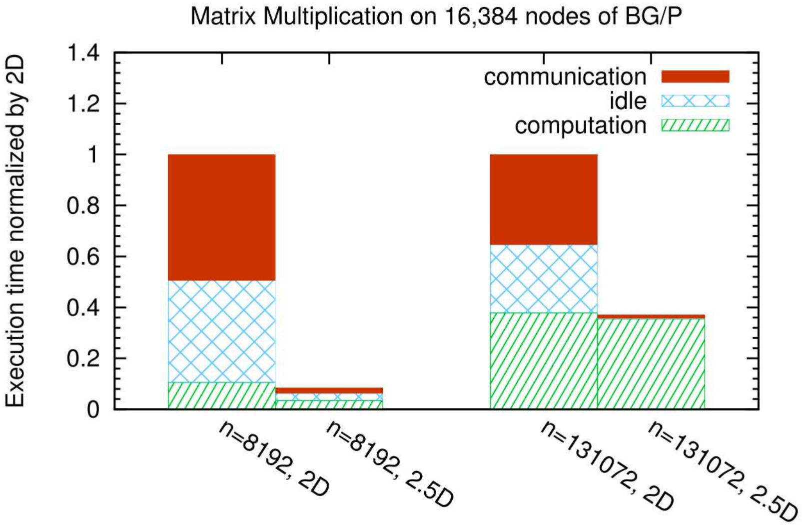
n = matrix dimension
P = number of processors
M = size of the local memory



	Computation	Communication
"2D" Algorithm (ScaLAPACK)	$O\left(\frac{n^3}{P}\right)$	$O\left(\frac{n^2}{\sqrt{P}}\right)$
"2.5D" Algorithm	$O\left(\frac{n^3}{P}\right)$	$O\left(\frac{n^3}{P\sqrt{M}}\right)$
Lower Bound	$\Omega\left(\frac{n^3}{P}\right)$	$\Omega\left(\frac{n^3}{P\sqrt{M}}\right)$

- 2D algorithm is suboptimal if $M \gg \frac{n^2}{P}$ (extra memory available)
- Takeaway: tradeoff extra memory for reduced communication

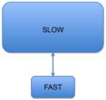
Performance improvement in practice



Lower bounds for classical matrix multiplication

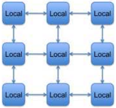
- Assume $\Theta(n^3)$ algorithm
- Sequential case with fast memory of size M
  - lower bound on words moved between fast/slow mem:

$$\Omega\left(\frac{n^3}{\sqrt{M}}\right) \quad \text{[Hong \& Kung 81]}$$



  - attained by blocked algorithm
- Parallel case with P processors (local memory of size M)
  - lower bound on words communicated (along critical path):

$$\Omega\left(\frac{n^3}{P\sqrt{M}}\right) \quad \text{[Toledo et al. 04]}$$



  - attained by 2.5D algorithm

Extensions to the rest of linear algebra

Theorem (Ballard, Demmel, Holtz, Schwartz 11)

If a computation "smells" like 3 nested loops, it must communicate

$$\#\text{ words} = \Omega\left(\frac{\#\text{ flops}}{\sqrt{\text{memory size}}}\right)$$

This result applies to

- dense or sparse problems
- sequential or parallel computers

This work was recognized with the *SIAM Linear Algebra Prize*, given to the best paper from the years 2009-2011

Extensions to the rest of linear algebra

Theorem (Ballard, Demmel, Holtz, Schwartz 11)

If a computation "smells" like 3 nested loops, it must communicate

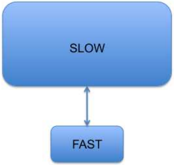
$$\#\,words = \Omega\left(\frac{\#\,flops}{\sqrt{memory\ size}}\right)$$

What smells like 3 nested loops?

- the rest of BLAS 3 (e.g. matrix multiplication, triangular solve)
- Cholesky, LU, LDL^T, LTL^T decompositions
- QR decomposition
- eigenvalue and SVD reductions
- sequences of algorithms (e.g. repeated matrix squaring)
- graph algorithms (e.g. all pairs shortest paths)

This work was recognized with the *SIAM Linear Algebra Prize*, given to the best paper from the years 2009-2011

Optimal Algorithms - Sequential $O(n^3)$ Linear Algebra



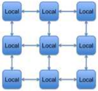
Computation	Optimal Algorithm
BLAS 3	blocked algorithms [Gustavson 97]
Cholesky	LAPACK [Ahmed & Pingali 00] [BDHS10]
Symmetric Indefinite	LAPACK (rarely) [BBD+13a]
LU	LAPACK (rarely) [Toledo 97]* [Grigori et al. 11]
QR	LAPACK (rarely) [Frens & Wise 03] [Elmroth & Gustavson 98]* [Hoemmen et al. 12]*
Eig, SVD	[BDK13], [BDD11]

Algorithms - Parallel $O(n^3)$ Linear Algebra

Algorithm	Reference	Factor exceeding lower bound for # words	Factor exceeding lower bound for # messages
Matrix Multiply	[Cannon 69]	1	1
Cholesky	ScaLAPACK	$\log P$	$\log P$
Symmetric Indefinite	[BBD+13b]	?	?
	ScaLAPACK	$\log P$	$(N/P^{1/2}) \log P$
LU	[Grigori et al. 11]	$\log P$	$\log P$
	ScaLAPACK	$\log P$	$(N/P^{1/2}) \log P$
QR	[Hoemmen et al. 12]*	$\log P$	$\log^3 P$
	ScaLAPACK	$\log P$	$(N/P^{1/2}) \log P$
SymEig, SVD	[BDK13]	?	?
	ScaLAPACK	$\log P$	$N/P^{1/2}$
NonsymEig	[BDD11]	$\log P$	$\log^3 P$
	ScaLAPACK	$P^{1/2} \log P$	$N \log P$

*This table assumes that *one* copy of the data is distributed evenly across processors

Red = not optimal



For more details...

For a more comprehensive (150+ pages) survey, see our

Communication lower bounds and optimal algorithms for numerical linear algebra

in the most recent **Acta Numerica** volume
[BCD+14]

Example Application: Video Background Subtraction

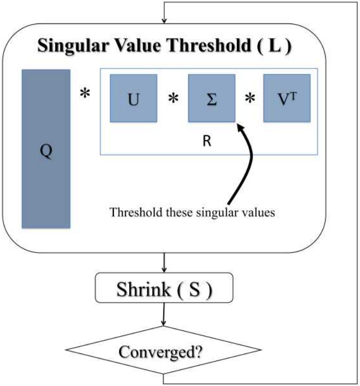
Idea: use Robust PCA algorithm [Candes et al. 09] to subtract constant background from the action of a surveillance video

Given a matrix M whose columns represent frames, compute

$$M = L + S$$

where L is low-rank and S is sparse

Example Application: Video Background Subtraction



Compute:

$$M = L + S$$

where L is low-rank and S is sparse

The algorithm works iteratively, each iteration requires a singular value decomposition (SVD)

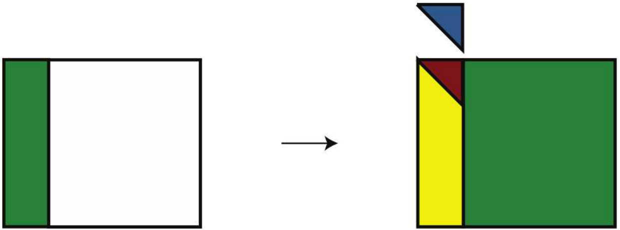
- M is $110{,}000 \times 100$

Communication-avoiding algorithm provided $3\times$ speedup over best GPU implementation [ABDK11]

Householder QR (HhQR)

Blocked Householder QR works by repeating:

1. panel factorization (tall-skinny QR decomposition)
2. trailing matrix update (application of orthogonal factor)



Householder vectors computed and applied one at a time

$$I - \tau y y^T$$

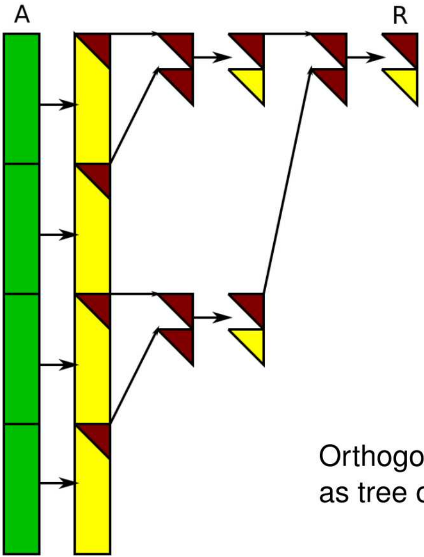
(two parallel reductions per column)

Householder vectors aggregated by computing triangular matrix T

$$I - YTY^T$$

(application $=$ matrix multiplications)

Tall-Skinny QR (TSQR)

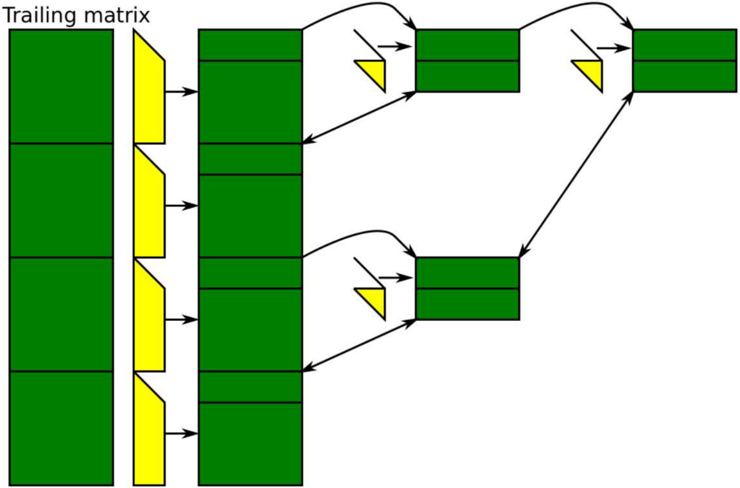


Key benefit of TSQR:
one parallel reduction

Orthogonal factor stored implicitly
as tree of Householder vectors

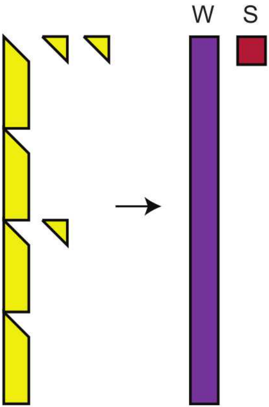
Communication-Avoiding QR (CAQR)

CAQR uses TSQR for panel factorization and applies the update using implicit tree structure



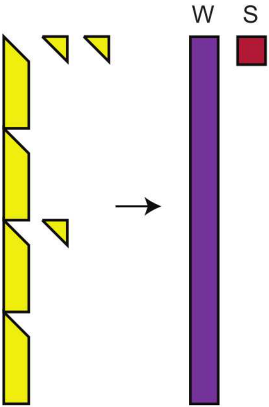
Yamamoto's Idea

- Y. Yamamoto gave a talk at SIAM ALA 2012: he wanted to use TSQR but offload the trailing matrix update to a GPU
- To make CAQR's trailing matrix update more like matrix multiplication, his idea is to convert implicit tree into compact WY-like representation

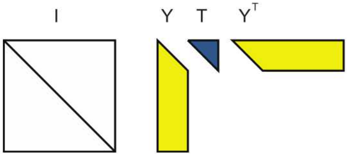


Yamamoto's Idea

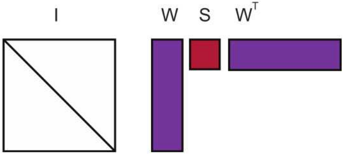
- Y. Yamamoto gave a talk at SIAM ALA 2012: he wanted to use TSQR but offload the trailing matrix update to a GPU
- To make CAQR's trailing matrix update more like matrix multiplication, his idea is to convert implicit tree into compact WY-like representation



Compact WY representation: $I - YTY^T$



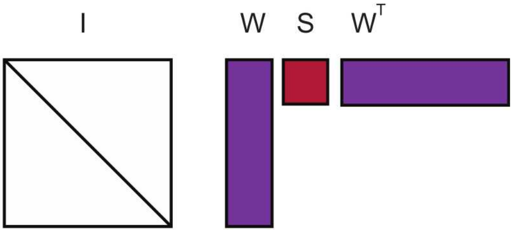
Basis-kernel representation: $I - WSW^T$



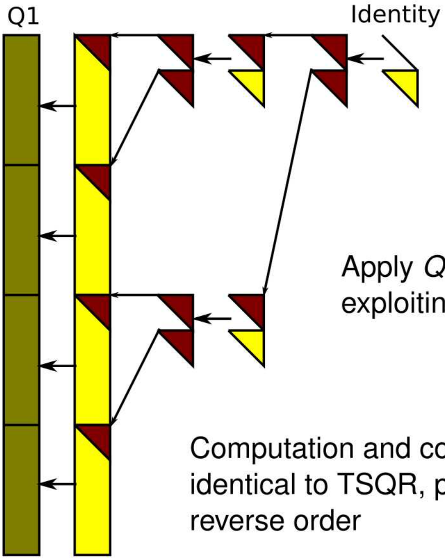
Yamamoto's Algorithm

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Set $W = Q - I$
4. Set $S = (I - Q_1)^{-1}$

$$I - WSW^T = I - \begin{bmatrix} Q_1 - I \\ Q_2 \end{bmatrix} \left[I - Q_1 \right]^{-1} \left[(Q_1 - I)^T \quad Q_2^T \right]$$



How is Q formed?



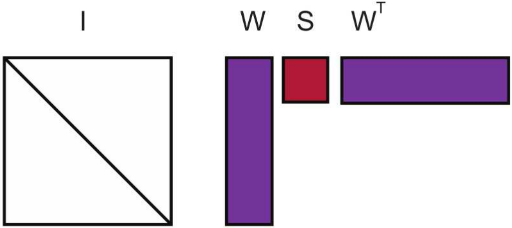
Apply Q to the identity, exploiting sparsity

Computation and communication identical to TSQR, performed in reverse order

Yamamoto's Algorithm

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Set $W = Q - I$
4. Set $S = (I - Q_1)^{-1}$

$$I - WSW^T = I - \begin{bmatrix} Q_1 - I \\ Q_2 \end{bmatrix} \left[I - Q_1\right]^{-1} \left[(Q_1 - I)^T \quad Q_2^T\right]$$

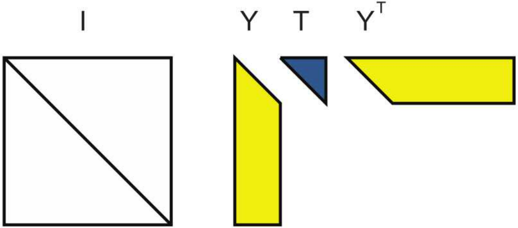


Reconstructing Householder Vectors (TSQR-HR)

With a little more effort, we can obtain the compact WY representation:

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Perform LU decomposition: $Q - I = LU$
4. Set $Y = L$
5. Set $T = -UY_1^{-T}$

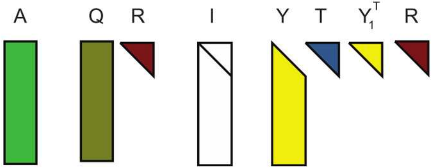
$$I - YTY^T = I - \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \begin{bmatrix} T \end{bmatrix} \begin{bmatrix} Y_1^T & Y_2^T \end{bmatrix}$$



Key Idea

Compute a QR decomposition using Householder vectors*:

$$A = QR = (I - YTY_1^T)R$$

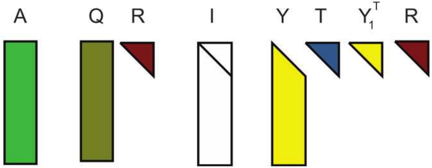


*$I - YTY_1^T$ known as compact WY representation

Key Idea

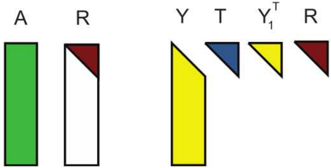
Compute a QR decomposition using Householder vectors*:

$$A = QR = (I - YTY_1^T)R$$



Re-arrange the equation and we have an LU decomposition:

$$A - R = Y \cdot (-TY_1^T R)$$



*$I - YTY_1^T$ known as compact WY representation

Why form Q?

Cheaper approach based on $A - R = Y \cdot (-TY_1^T R)$:

1. Perform TSQR
2. Perform LU decomposition: $A - R = LU$
3. Set $Y = L$
4. Set $T = -UR^{-1}Y_1^{-T}$ (or compute T from Y)

Why form Q?

Cheaper approach based on $A - R = Y \cdot (-TY_1^T R)$:

1. Perform TSQR
2. Perform LU decomposition: $A - R = LU$
3. Set $Y = L$
4. Set $T = -UR^{-1}Y_1^{-T}$ (or compute T from Y)

This approach is similar to computing R using TSQR and Q using Householder QR

- if A is well-conditioned, works fine
- if A is low-rank, QR decomposition is not unique
- if A is ill-conditioned, R matrix is sensitive to roundoff

What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition: $Q - I = LU$
  - what if $Q - I$ is singular?

What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition: $Q - I = LU$
  - what if $Q - I$ is singular?

Actually, we need to make a sign choice:

- Perform LU decomposition: $Q - Sgn = LU$
  - Sgn matrix corresponds to sign choice in Householder QR
  - guarantees $Q - Sgn$ is nonsingular
  - guarantees maximum element on the diagonal (no pivoting)

What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition: $Q - I = LU$
  - what if $Q - I$ is singular?

Actually, we need to make a sign choice:

- Perform LU decomposition: $Q - Sgn = LU$
  - Sgn matrix corresponds to sign choice in Householder QR
  - guarantees $Q - Sgn$ is nonsingular
  - guarantees maximum element on the diagonal (no pivoting)

No pivoting makes LU of tall-skinny matrix very easy

- LU of top block followed by triangular solve for all other rows

Numerical Stability

Theorem

Let $\hat{R}$ be the computed upper triangular factor of $m \times b$ matrix A obtained via the TSQR algorithm with p processors using a binary tree (assuming $m/p \geq b$), and let $\tilde{Q} = I - \tilde{Y}\tilde{T}\tilde{Y}_1^T$ and $\tilde{R} = S\hat{R}$ where $\tilde{Y}$, $\tilde{T}$, and S are the computed factors obtained from Householder reconstruction. Then

$$\|A - \tilde{Q}\tilde{R}\|_F \leq F_1(m, b, p, \varepsilon)\|A\|_F$$

and

$$\|I - \tilde{Q}^T\tilde{Q}\|_F \leq F_2(m, b, p, \varepsilon)$$

where $F_1, F_2 = O\left(\left(b^{3/2}(m/p) + b^{5/2}\log p + b^3\right)\epsilon\right)$ for $b(m/p)\epsilon \ll 1$.

Numerical Experiments for Tall-Skinny Matrices

ρ	κ	$\|A - QR\|_2$	$\|I - Q^T Q\|_2$
1e-01	5.1e+02	2.2e-15	9.3e-15
1e-03	5.0e+04	2.2e-15	8.4e-15
1e-05	5.1e+06	2.3e-15	8.7e-15
1e-07	5.0e+08	2.4e-15	1.1e-14
1e-09	5.0e+10	2.3e-15	9.9e-15
1e-11	4.9e+12	2.5e-15	1.0e-14
1e-13	5.0e+14	2.2e-15	8.8e-15
1e-15	5.0e+15	2.4e-15	9.7e-15

Error of TSQR-HR on tall and skinny matrices ($m = 1000, b = 200$)

Numerical Experiments for Square Matrices

Matrix type	κ	$\|A - QR\|_2$	$\|I - Q^T Q\|_2$
$A = 2 * rand(m) - 1$	$2.1e{+}03$	4.3e-15 (256)	2.8e-14 (2)
Golub-Klema-Stewart	$2.2e{+}20$	0.0e+00 (2)	0.0e+00 (2)
Break 1 distribution	$1.0e{+}09$	1.0e-14 (256)	2.8e-14 (2)
Break 9 distribution	$1.0e{+}09$	9.9e-15 (256)	2.9e-14 (2)
$U \Sigma V^T$ with exponential distribution	$4.2e{+}19$	2.0e-15 (256)	2.8e-14 (2)
The devil's stairs matrix	$2.3e{+}19$	2.4e-15 (256)	2.8e-14 (2)
KAHAN matrix, a trapezoidal matrix	$5.6e{+}56$	0.0e+00 (2)	0.0e+00 (2)
Matrix ARC130 from Matrix Market	$6.0e{+}10$	8.8e-19 (16)	2.1e-15 (2)
Matrix FS_541_1 from Matrix Market	$4.5e{+}03$	5.8e-16 (64)	1.8e-15 (256)
DERIV2: second derivative	$1.2e{+}06$	2.8e-15 (256)	4.6e-14 (2)
FOXGOOD: severely ill-posed problem	$5.7e{+}20$	2.4e-15 (256)	2.8e-14 (2)

Errors of CAQR-HR on square matrices ($m = 1000$)

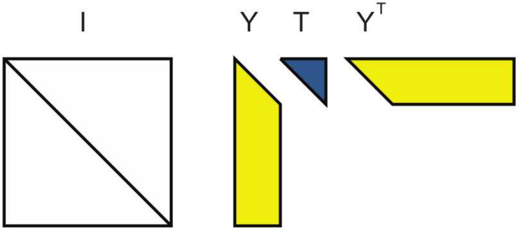
The numbers in parentheses give the panel width yielding largest error

Reconstructing Householder Vectors (TSQR-HR)

With a little more effort, we can obtain the compact WY representation:

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Perform LU decomposition: $Q - I = LU$
4. Set $Y = L$
5. Set $T = -UY_1^{-T}$

$$I - YTY^T = I - \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \begin{bmatrix} T \end{bmatrix} \begin{bmatrix} Y_1^T & Y_2^T \end{bmatrix}$$



Costs of Householder Reconstruction

Householder Reconstruction Let A be $n \times b$

1. Perform TSQR — $2nb^2$ flops, one QR reduction of size $b^2/2$
2. Form Q — $2nb^2$ flops, one QR reduction of size $b^2/2$
3. LU($Q - Sgn$) — nb^2 flops, one broadcast of size $b^2/2$
4. Set $Y = L$
5. Set $T = -U \cdot Sgn \cdot Y_1^{-T}$ — $O(b^3)$ flops

Costs of Householder Reconstruction

Householder Reconstruction — Let A be $n \times b$

1. Perform TSQR — $2nb^2$ flops, one QR reduction of size $b^2/2$
2. Form Q — $2nb^2$ flops, one QR reduction of size $b^2/2$
3. $\text{LU}(Q - Sgn)$ — nb^2 flops, one broadcast of size $b^2/2$
4. Set $Y = L$
5. Set $T = -U \cdot Sgn \cdot Y_1^{-T}$ — $O(b^3)$ flops

Alternative Algorithms

- TSQR — $2nb^2$ flops, one QR reduction of size $b^2/2$
- HhQR (and form T) — $3nb^2$ flops, $2b$ reductions of size $O(b)$
- Yamamoto's — $4nb^2$ flops, two QR reductions of size $b^2/2$

Costs of Householder Reconstruction

Householder Reconstruction Let A be $n \times b$

1. Perform TSQR — $2nb^2$ flops, one QR reduction of size $b^2/2$
2. Form Q — $2nb^2$ flops, one QR reduction of size $b^2/2$
3. LU($Q - Sgn$) — nb^2 flops, one broadcast of size $b^2/2$
4. Set $Y = L$
5. Set $T = -U \cdot Sgn \cdot Y_1^{-T}$ — $O(b^3)$ flops

Alternative Algorithms

- TSQR — $2nb^2$ flops, one QR reduction of size $b^2/2$
- HhQR (and form T) — $3nb^2$ flops, $2b$ reductions of size $O(b)$
- Yamamoto's — $4nb^2$ flops, two QR reductions of size $b^2/2$

For square matrices, flop costs of panel factorization are lower order: $O(n^2 b)$

Recent Improvement*

Improved Householder Reconstruction

1. Perform TSQR — $2nb^2$ flops, one QR reduction
2. Form Q_1 — $O(b^3)$ flops
3. Compute $LU = Q_1 - Sgn$ — $O(b^3)$ flops
4. $Y = \text{Apply } Q \text{ to } \begin{bmatrix} U^{-1} \\ 0 \end{bmatrix}$ — $2nb^2$ flops, one QR reduction
5. $T = -U \cdot Sgn \cdot Y_1^{-T}$ — $O(b^3)$ flops

*Thanks to Nick Knight

Recent Improvement*

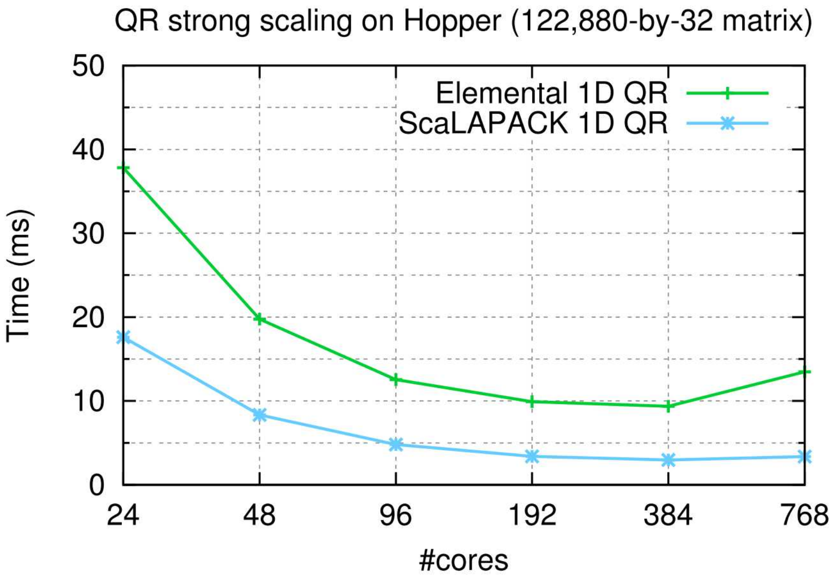
Improved Householder Reconstruction

1. Perform TSQR — $2nb^2$ flops, one QR reduction
2. Form Q_1 — $O(b^3)$ flops
3. Compute $LU = Q_1 - Sgn$ — $O(b^3)$ flops
4. $Y =$ Apply Q to $\begin{bmatrix} U^{-1} \\ 0 \end{bmatrix}$ — $2nb^2$ flops, one QR reduction
5. $T = -U \cdot Sgn \cdot Y_1^{-T}$ — $O(b^3)$ flops

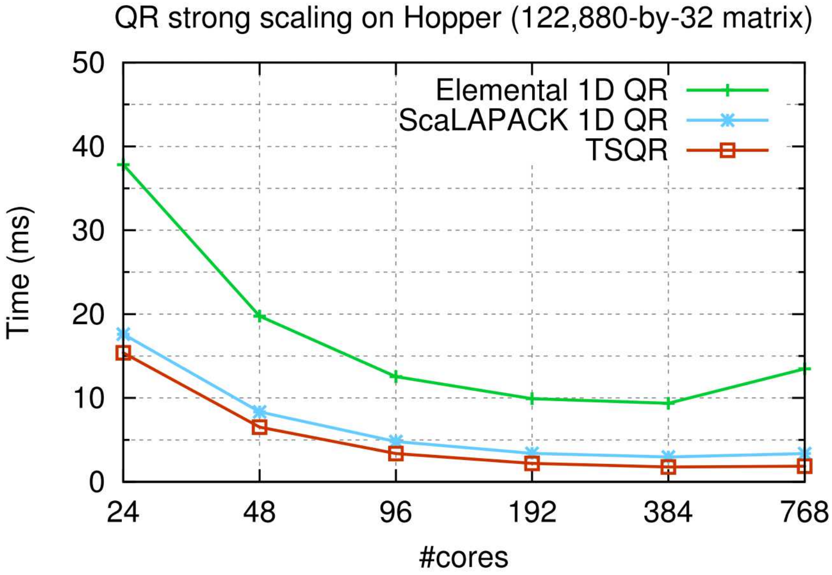
- Intuitively: fold the tall-skinny TRSM into the "Form Q" step
- Achieves same comp/comm costs as Yamamoto's algorithm
- Requires careful choice of TSQR reduction tree
- Implementation underway

*Thanks to Nick Knight

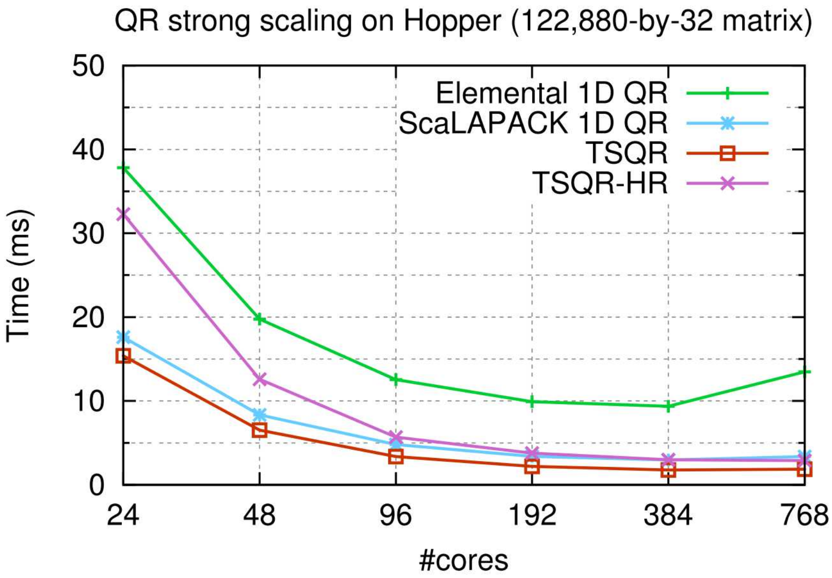
Performance for Tall-Skinny Matrices



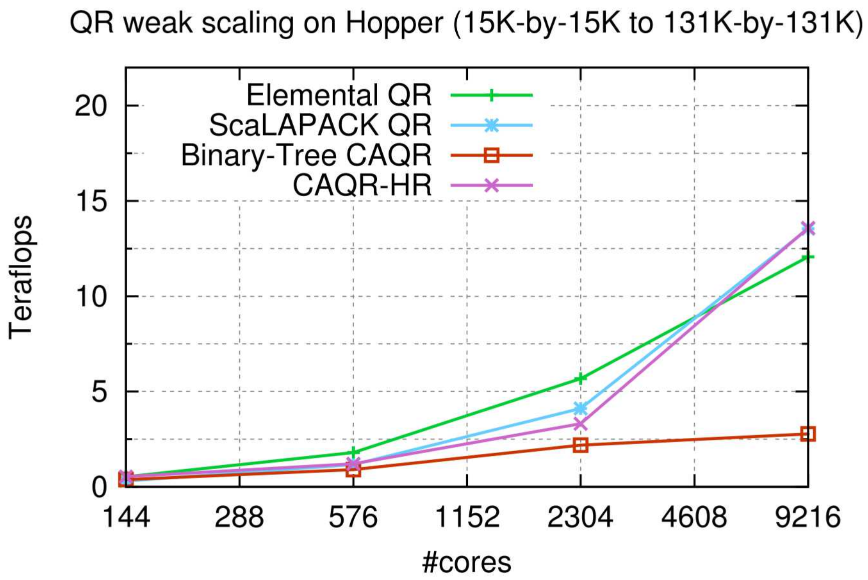
Performance for Tall-Skinny Matrices



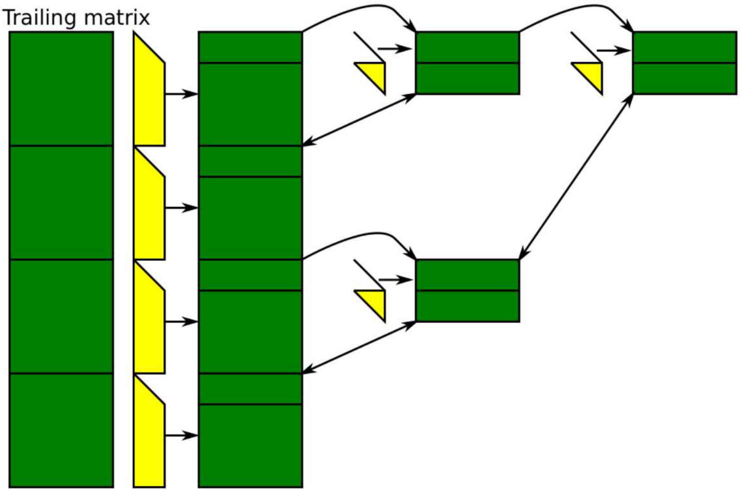
Performance for Tall-Skinny Matrices



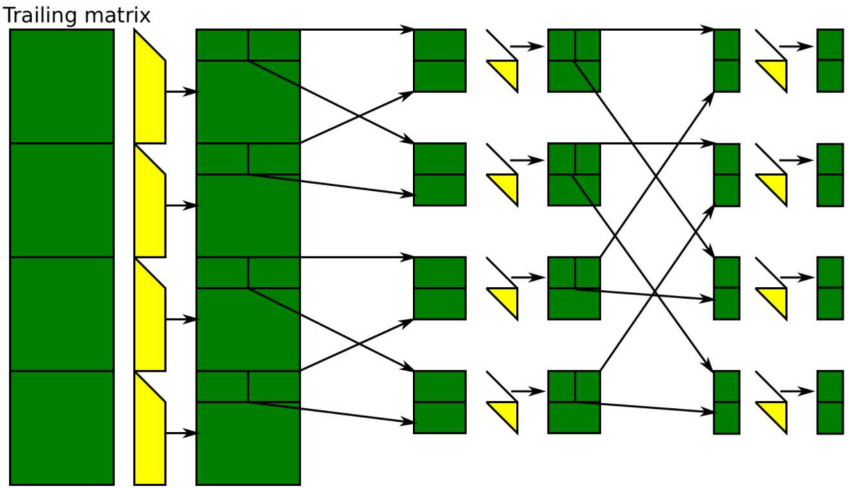
Performance for Square Matrices



Binary-Apply CAQR

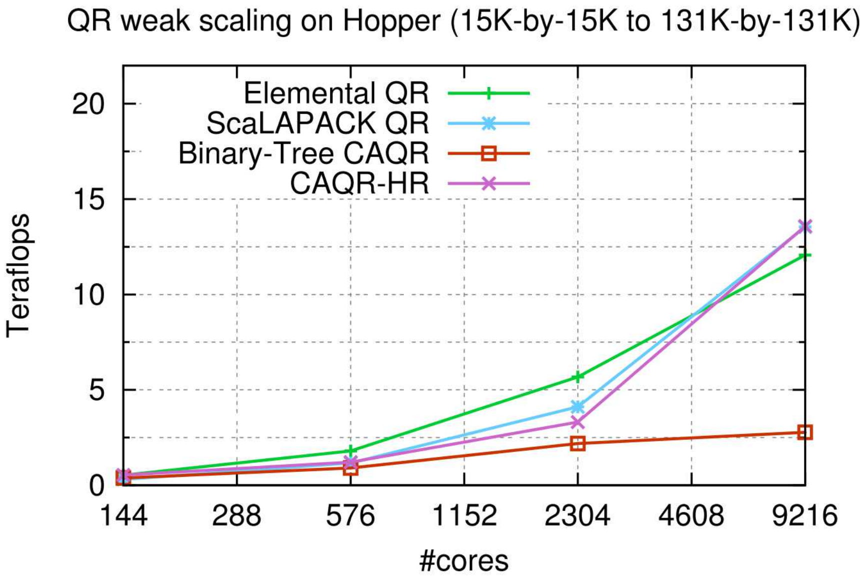


Scatter-Apply CAQR

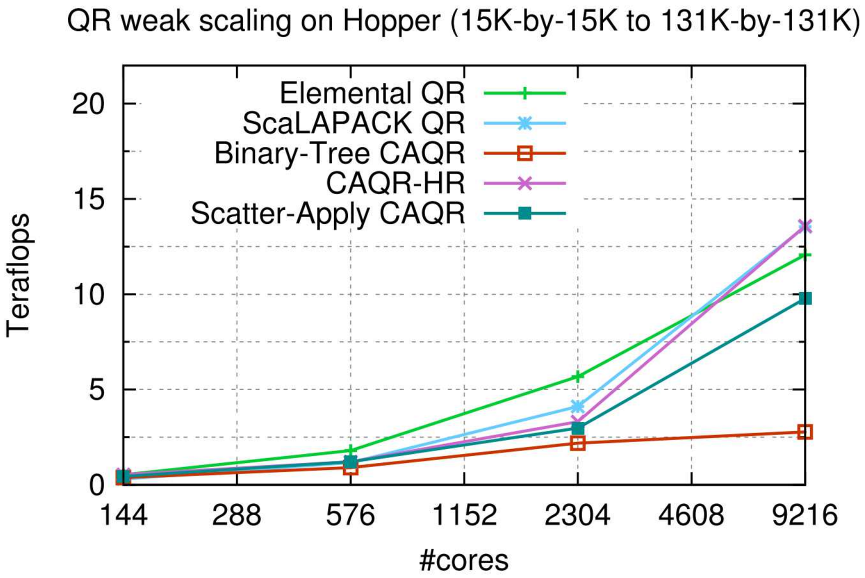


Similar to performing an all-reduce by reduce-scatter followed by all-gather

Performance for Square Matrices



Performance for Square Matrices

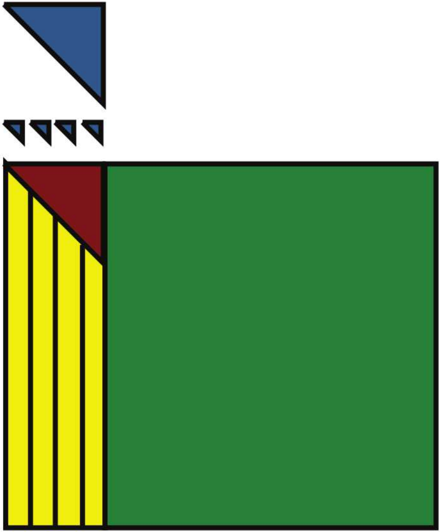


Two-Level Aggregation

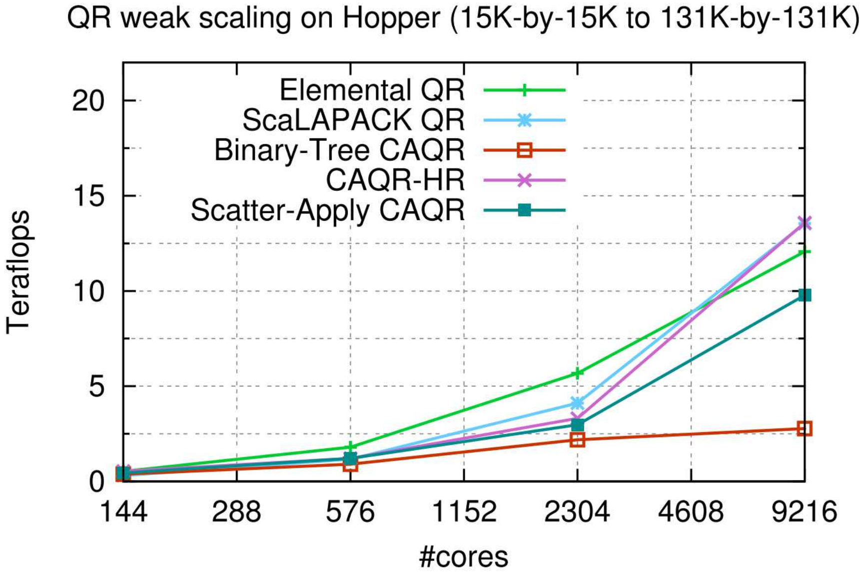
Block size trades off time spent in panel factorizations with efficiency of matrix multiplications

Solution:

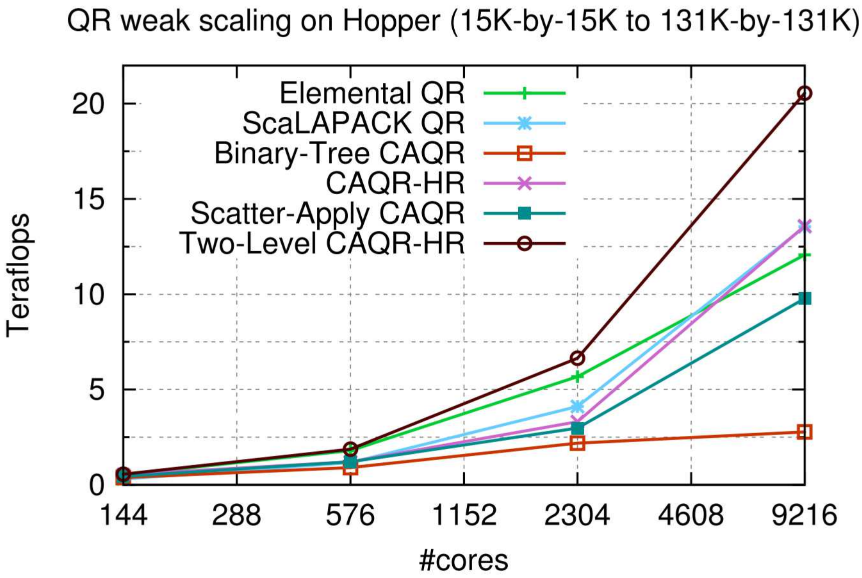
- Use another level of compact WY blocking
- Allow for larger local matrix multiplications
- (Can't use with CAQR)



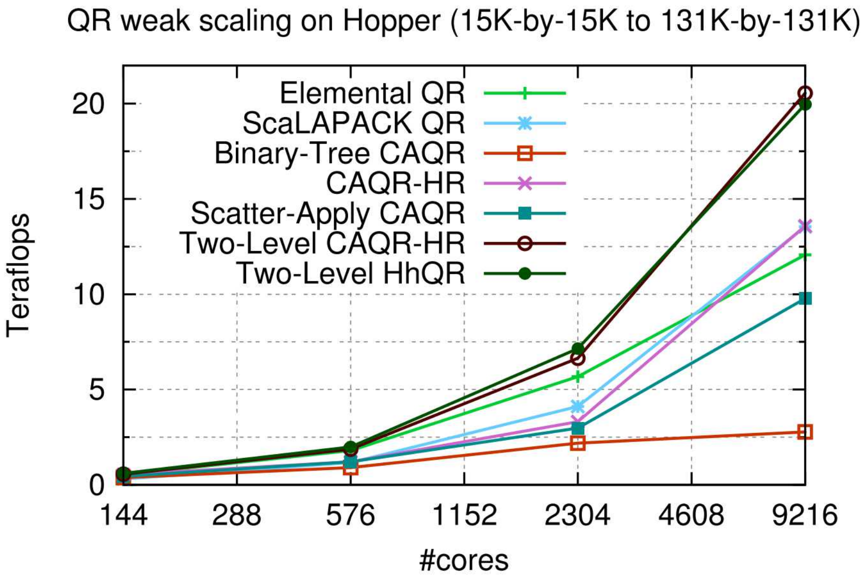
Performance for Square Matrices



Performance for Square Matrices



Performance for Square Matrices



Conclusions

- Communication is costly, even for historically "compute-bound" problems like dense linear algebra
- TSQR reduces communication and runs faster in practice for tall-skinny matrices
- Householder reconstruction provides best of both worlds
  - latency-avoiding panel factorization
  - matrix multiplication for trailing matrix updates
  - backwards compatibility for performance portability

Collaborators

- Michael Anderson (UC Berkeley)
- Austin Benson (Stanford)
- Aydin Buluc (LBNL)
- James Demmel (UC Berkeley)
- Alex Druinsky (Tel-Aviv U)
- Ioana Dumitriu (U Washington)
- Andrew Gearhart (UC Berkeley)
- Laura Grigori (INRIA)
- Olga Holtz (UC Berkeley/TU Berlin)
- Jonathan Hu (Sandia NL)
- Mathias Jacquelin (LBNL)
- Nicholas Knight (UC Berkeley)
- Kurt Keutzer (UC Berkeley)
- Tamara Kolda (Sandia NL)
- Benjamin Lipshitz (Google)
- Inon Peled (Tel-Aviv U)
- Todd Plantenga (Sandia NL)
- Oded Schwartz (UC Berkeley)
- Chris Siefert (Sandia NL)
- Edgar Solomonik (UC Berkeley)
- Sivan Toledo (Tel-Aviv U)
- Ichitaro Yamazaki (UT Knoxville)

Algorithmic Improvements for QR Decomposition

For more details:

Reconstructing Householder Vectors from Tall-Skinny QR

G. Ballard, J. Demmel, L. Grigori, M. Jacquelin, H.D. Nguyen and E. Solomonik

http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-175.html

Thank You!

www.sandia.gov/~gmballa

References I

J. O. Aasen.
On the reduction of a symmetric matrix to tridiagonal form.
BIT Numerical Mathematics, 11(3):233–242, 1971.
10.1007/BF01931804.

M. Anderson, G. Ballard, J. Demmel, and K. Keutzer.
Communication-avoiding QR decomposition for GPUs.
In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, IPDPS '11, pages 48–58, Washington, DC, USA, 2011. IEEE Computer Society.

G. Ballard, D. Becker, J. Demmel, J. Dongarra, A. Druinsky, I. Peled, O. Schwartz, S. Toledo, and I. Yamazaki.
Communication-avoiding symmetric-indefinite factorization.
Technical Report UCB/EECS-2013-127, EECS Department, University of California, Berkeley, July 2013.

References II

G. Ballard, D. Becker, J. Demmel, J. Dongarra, A. Druinsky, I. Peled, O. Schwartz, S. Toledo, and I. Yamazaki.

Implementing a blocked Aasen's algorithm with a dynamic scheduler on multicore architectures.

In *Proceedings of the 27th IEEE International Parallel Distributed Processing Symposium*, IPDPS '13, pages 895–907, May 2013.

L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

ScaLAPACK Users' Guide.

SIAM, Philadelphia, PA, USA, May 1997.

Also available from http://www.netlib.org/scalapack/.

References III

G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz.
Communication lower bounds and optimal algorithms for numerical linear algebra.
Acta Numerica, 23:1–155, 5 2014.

G. Ballard, J. Demmel, and I. Dumitriu.
Communication-optimal parallel and sequential eigenvalue and singular value algorithms.
Technical Report EECS-2011-14, UC Berkeley, February 2011.

G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz.
Communication-optimal parallel algorithm for Strassen's matrix multiplication.
In *Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 193–204, New York, NY, USA, 2012. ACM.

References IV

G. Ballard, J. Demmel, O. Holtz, and O. Schwartz.
Communication-optimal parallel and sequential Cholesky decomposition.
SIAM Journal on Scientific Computing, 32(6):3495–3523, 2010.

G. Ballard, J. Demmel, O. Holtz, and O. Schwartz.
Graph expansion and communication costs of fast matrix multiplication.
In *Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, pages 1–12. ACM, 2011.

G. Ballard, J. Demmel, and N. Knight.
Avoiding communication in successive band reduction.
Technical Report UCB/EECS-2013-131, EECS Department, University of California, Berkeley, July 2013.

References V

L. Cannon.
A cellular computer to implement the Kalman filter algorithm.
PhD thesis, Montana State University, Bozeman, MN, 1969.

J. Demmel, L. Grigori, M. Hoemmen, and J. Langou.
Communication-optimal parallel and sequential QR and LU factorizations.
SIAM Journal on Scientific Computing, 34(1):A206–A239, 2012.

L. Grigori, J. Demmel, and H. Xiang.
CALU: A communication optimal LU factorization algorithm.
SIAM Journal on Matrix Analysis and Applications, 32(4):1317–1350, 2011.

References VI

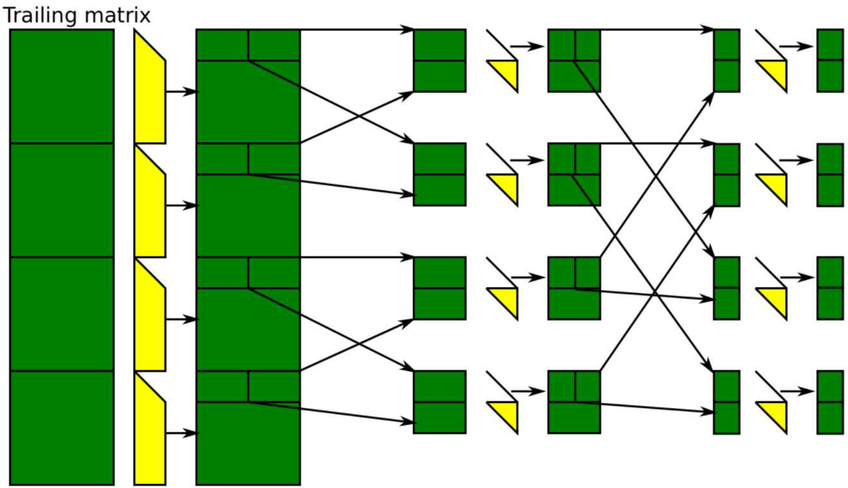
Jack Poulson, Bryan Marker, Robert A. van de Geijn, Jeff R. Hammond, and Nichols A. Romero.
Elemental: A new framework for distributed memory dense matrix computations.
ACM Trans. Math. Softw., 39(2):13:1–13:24, February 2013.

Leading Order Costs for General Matrices

	Flops	Words	Messages
Householder QR	$\frac{2mn^2 - 2n^3/3}{p}$	$\frac{2mn + n^2/2}{\sqrt{p}}$	$n \log p$
Binary-Apply CAQR	$\frac{2mn^2 - 2n^3/3}{p}$	$\frac{2mn + n^2 \log p}{\sqrt{p}}$	$\frac{7}{2}\sqrt{p}\log^3 p$
CAQR-HR	$\frac{2mn^2 - 2n^3/3}{p}$	$\frac{2mn + n^2/2}{\sqrt{p}}$	$6\sqrt{p}\log^2 p$
Scatter-Apply CAQR	$\frac{2mn^2 - 2n^3/3}{p}$	$\frac{2mn + n^2/2}{\sqrt{p}}$	$7\sqrt{p}\log^2 p$

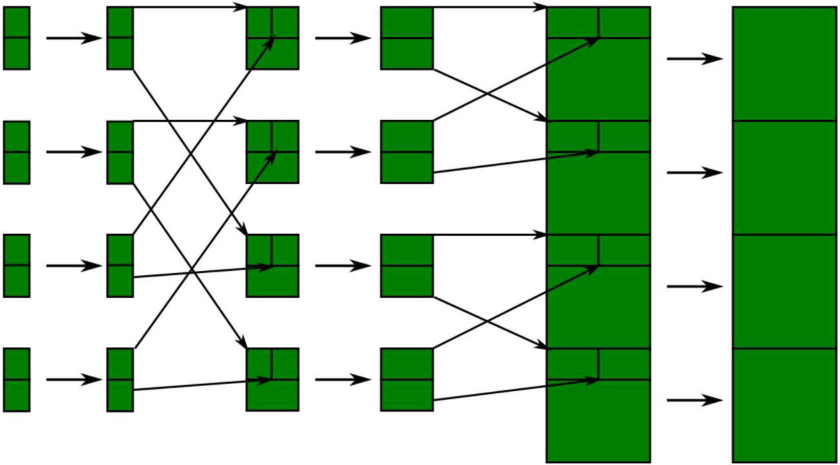
Costs of QR factorization of $m \times n$ matrix distributed over p processors in 2D fashion.

Scatter-Apply CAQR



Similar to performing an all-reduce by reduce-scatter followed by all-gather

Scatter-Apply CAQR



Similar to performing an all-reduce by reduce-scatter followed by all-gather

More Numerical Stability Experiments

		$Q - I$ (T from TSQR-HR)			Yamamoto's approach			$A - R$ (T^{-1} from $Y^T Y$)		
ρ	κ	norm-wise	col-wise	ortho.	norm-wise	col-wise	ortho.	norm-wise	col-wise	ortho.
1e-01	5.1e+02	2.2e-15	2.7e-15	9.3e-15	2.5e-15	3.1e-15	9.2e-15	3.8e-14	1.7e-14	5.5e-15
1e-02	5.0e+03	2.3e-15	2.9e-15	1.0e-14	2.4e-15	3.1e-15	1.1e-14	3.2e-13	1.1e-13	6.2e-15
1e-03	5.0e+04	2.2e-15	2.6e-15	8.4e-15	2.6e-15	3.4e-15	1.1e-14	4.2e-12	1.7e-12	5.6e-15
1e-04	4.9e+05	2.2e-15	2.6e-15	7.7e-15	2.3e-15	2.8e-15	8.7e-15	3.8e-11	1.7e-11	5.4e-15
1e-05	5.1e+06	2.3e-15	2.9e-15	8.7e-15	3.2e-15	4.2e-15	1.0e-14	3.9e-10	1.4e-10	5.3e-15
1e-06	5.0e+07	2.3e-15	3.0e-15	9.1e-15	3.0e-15	3.9e-15	1.0e-14	3.6e-09	1.5e-09	6.1e-15
1e-07	5.0e+08	2.4e-15	3.4e-15	1.1e-14	2.7e-15	3.7e-15	9.9e-15	4.2e-08	2.1e-08	5.0e-15
1e-08	5.1e+09	2.2e-15	2.8e-15	8.6e-15	2.5e-15	3.1e-15	8.9e-15	3.8e-07	1.5e-07	5.8e-15
1e-09	5.0e+10	2.3e-15	3.1e-15	9.9e-15	3.9e-15	5.1e-15	1.3e-14	3.6e-06	2.0e-06	5.4e-15
1e-10	5.0e+11	2.1e-15	2.6e-15	7.1e-15	2.6e-15	3.4e-15	9.9e-15	3.3e-05	1.2e-05	6.3e-15
1e-11	4.9e+12	2.5e-15	3.4e-15	1.0e-14	2.4e-15	3.1e-15	1.0e-14	3.1e-04	1.2e-04	5.9e-15
1e-12	5.1e+13	2.2e-15	2.9e-15	8.5e-15	2.6e-15	3.3e-15	1.2e-14	3.7e-03	1.6e-03	5.8e-15
1e-13	5.0e+14	2.2e-15	2.7e-15	8.8e-15	3.0e-15	3.9e-15	1.0e-14	4.0e-02	1.4e-02	4.7e-15
1e-14	3.5e+15	2.3e-15	3.1e-15	1.0e-14	2.3e-15	2.9e-15	9.4e-15	2.7e-01	9.7e-02	4.9e-15
1e-15	5.0e+15	2.4e-15	3.1e-15	9.7e-15	2.8e-15	3.7e-15	9.4e-15	3.5e-01	1.3e-01	6.3e-15

Error on tall and skinny matrices ($m = 1000, b = 200$) for three approaches. The label "norm-wise" corresponds to $\|A - QR\|_2$, "col-wise" corresponds to $\max_i \|A_i - (QR)_i\|_2$, and "ortho." corresponds to $\|I - Q^T Q\|_2$.

Let's go back to matrix multiplication

Can we do better than the "2.5D" algorithm?

Given the computation involved, it minimized communication. . .

Let's go back to matrix multiplication

Can we do better than the "2.5D" algorithm?

Given the computation involved, it minimized communication. . .

. . . but what if we change the computation?

It's possible to reduce both computation *and* communication

Strassen's Algorithm

Strassen showed how to use 7 multiplies instead of 8 for 2×2 multiplication

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Classical Algorithm

$$\begin{aligned} M_1 &= A_{11} \cdot B_{11} \\ M_2 &= A_{12} \cdot B_{21} \\ M_3 &= A_{11} \cdot B_{12} \\ M_4 &= A_{12} \cdot B_{22} \\ M_5 &= A_{21} \cdot B_{11} \\ M_6 &= A_{22} \cdot B_{21} \\ M_7 &= A_{21} \cdot B_{12} \\ M_8 &= A_{22} \cdot B_{22} \\ C_{11} &= M_1 + M_2 \\ C_{12} &= M_3 + M_4 \\ C_{21} &= M_5 + M_6 \\ C_{22} &= M_7 + M_8 \end{aligned}$$

Strassen's Algorithm

$$\begin{aligned} M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\ M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\ M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\ M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\ M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\ M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\ M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22}) \end{aligned}$$

$$\begin{aligned} C_{11} &= M_1 + M_4 - M_5 + M_7 \\ C_{12} &= M_3 + M_5 \\ C_{21} &= M_2 + M_4 \\ C_{22} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

Strassen's Algorithm

Strassen showed how to use 7 multiplies instead of 8 for 2×2 multiplication

$$\underbrace{\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}}_{n/2,\; n/2} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \bullet \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned}
M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\
M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\
M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\
M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}$$

$$\begin{aligned}
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}$$

Flop count recurrence:

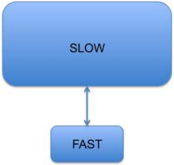
$$F(n) = 7 \cdot F(n/2) + \Theta(n^2)$$

$$F(n) = \Theta\left(n^{\log_2 7}\right)$$

$$\log_2 7 \approx 2.81$$

Sequential Communication Costs

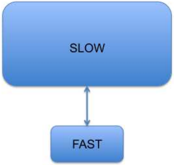
If you implement Strassen's algorithm recursively on a sequential computer:



	Computation	Communication
Classical (blocked)	$O(n^3)$	$O\left(\left(\frac{n}{\sqrt{M}}\right)^3 M\right)$
Strassen	$O(n^{\log_2 7})$	$O\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} M\right)$

Sequential Communication Costs

If you implement Strassen's algorithm recursively on a sequential computer:



	Computation	Communication
Classical (blocked)	$O(n^3)$	$O\left(\left(\frac{n}{\sqrt{M}}\right)^3 M\right)$
Strassen	$O(n^{\log_2 7})$	$O\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} M\right)$

Can we reduce Strassen's communication cost further?

Lower Bounds for Strassen's Algorithm

Theorem (Ballard, Demmel, Holtz, Schwartz 12)

On a sequential machine, Strassen's algorithm must communicate

$$\#\,\text{words} = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} M\right)$$

and on a parallel machine, it must communicate

$$\#\,\text{words} = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} \frac{M}{P}\right)$$

Lower Bounds for Strassen's Algorithm

Theorem (Ballard, Demmel, Holtz, Schwartz 12)

On a sequential machine, Strassen's algorithm must communicate

$$\#\ \text{words} = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} M\right)$$

and on a parallel machine, it must communicate

$$\#\ \text{words} = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} \frac{M}{P}\right)$$

This work received the *SPAA Best Paper Award* [BDHS11] and appeared as a Research Highlight in the *Communications of the ACM*

Optimal Parallel Algorithm?

This lower bound proves that the sequential recursive algorithm is communication-optimal

What about the parallel case?

Optimal Parallel Algorithm?

This lower bound proves that the sequential recursive algorithm is communication-optimal

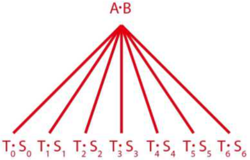
What about the parallel case?

- Earlier attempts to parallelize Strassen had communication costs that exceeded the lower bound
- We developed a new algorithm that is communication-optimal, called Communication-Avoiding Parallel Strassen (CAPS) [BDH+12]

Main idea of CAPS algorithm

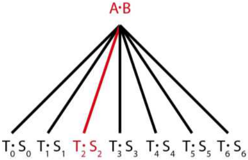
At each level of recursion tree, choose either breadth-first or depth-first traversal of the recursion tree

Breadth-First-Search (BFS)



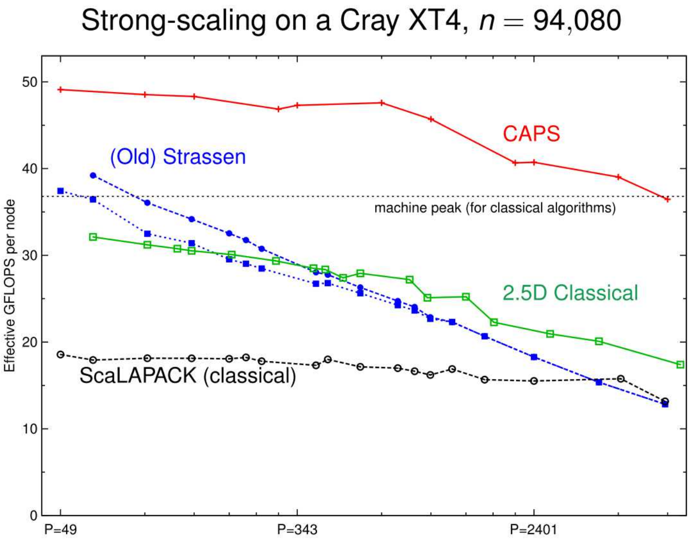
- Runs all 7 multiplies in parallel
  - each uses $P/7$ processors
- Requires $7/4$ as much extra memory
- Requires communication, but minimizes communication in subtrees

Depth-First-Search (DFS)



- Runs all 7 multiplies sequentially
  - each uses all P processors
- Requires $1/4$ as much extra memory
- Increases communication by factor of $7/4$ in subtrees

Performance of CAPS on a large problem



▸ More details

Can we beat Strassen?

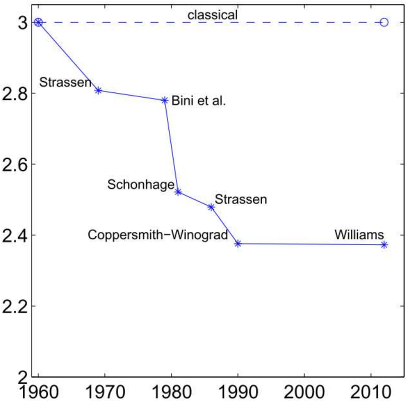
Strassen's algorithm allows for less computation and communication than the classical $O(n^3)$ algorithm

We have algorithms that attain its communication lower bounds and perform well on highly parallel machines

Can we do any better?

Can we beat Strassen?

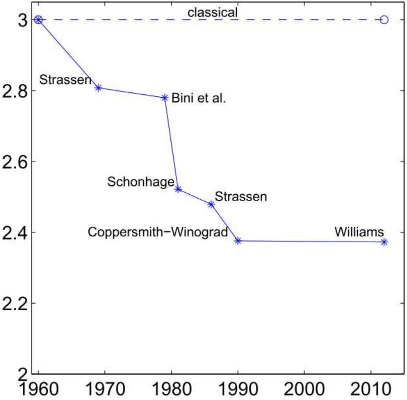
Exponent of matrix multiplication over time



$$F_{MM}(n) = O(n^{?})$$

Can we beat Strassen?

Exponent of matrix multiplication over time



$$F_{\mathrm{MM}}(n) = O(n^{?})$$

Unfortunately, most of these improvements are only theoretical (i.e., not practical) because they

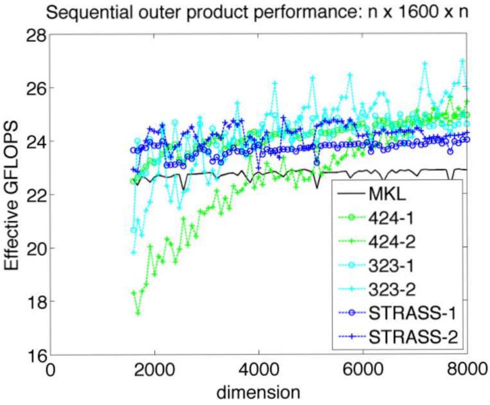
- involve approximations
- are existence proofs
- have large constants

Yes, it's possible!

- Other practical fast algorithms exist (with slightly better exponents)
- Smaller arithmetic exponent means less communication
- Rectangular matrix multiplication prefers rectangular base case

Yes, it's possible!

- Other practical fast algorithms exist (with slightly better exponents)
- Smaller arithmetic exponent means less communication
- Rectangular matrix multiplication prefers rectangular base case



- Parallel implementations underway...