Avoiding Communication in Linear Algebra

Grey Ballard

Microsoft Research Faculty Summit

July 15, 2014

Sandia National Laboratories

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.

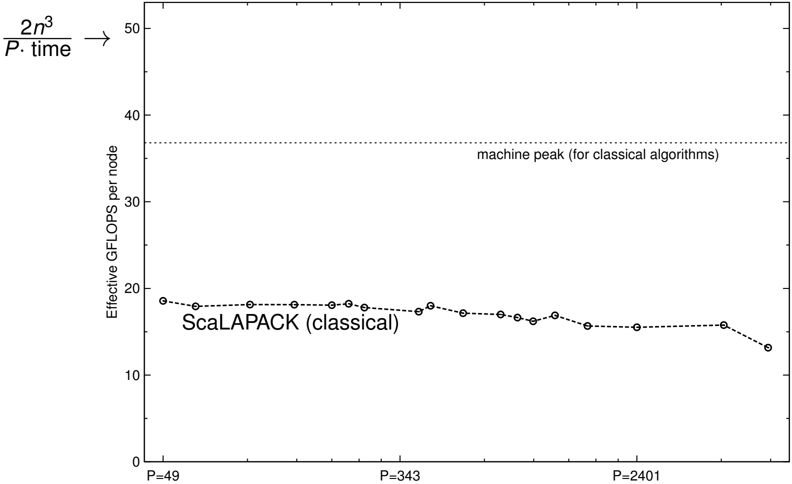
(SAND2014-5649)

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
- New Algorithms
  - (sometimes) require careful implementation to navigate tradeoffs
  - (sometimes) require numerical analysis to ensure correctness

Can we improve dense matrix multiplication?

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

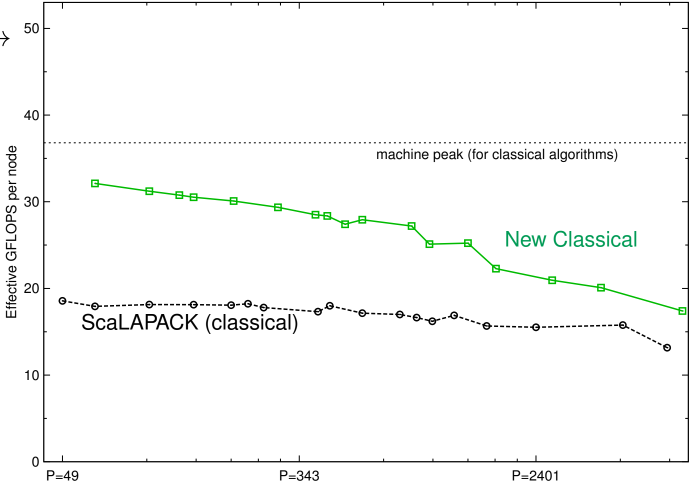


benchmarked on a Cray XT4

Can we improve dense matrix multiplication?

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

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

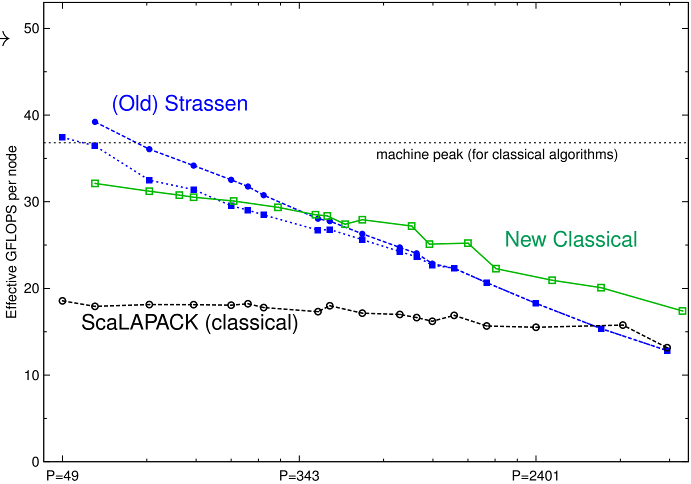


benchmarked on a Cray XT4

Can we improve dense matrix multiplication?

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

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

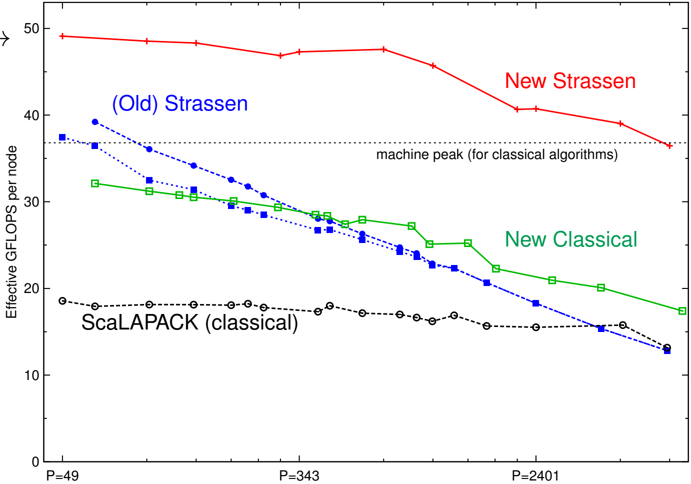


benchmarked on a Cray XT4

Can we improve dense matrix multiplication?

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

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



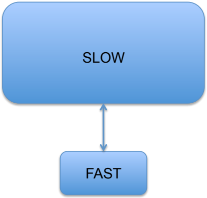
benchmarked on a Cray XT4

We must consider communication

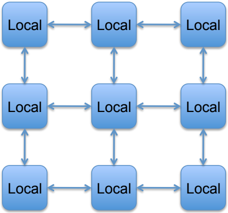
By *communication*, I mean

- 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):

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

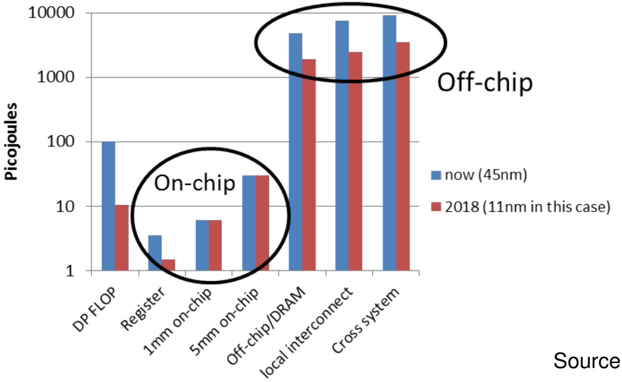
$\beta \gg \gamma$ as measured in time *and* energy, and the relative cost of communication is increasing

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

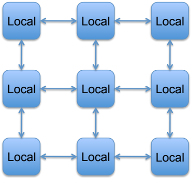
Energy cost comparisons



Source: John Shalf

Costs of matrix multiplication algorithms

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

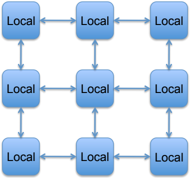


	Computation	Communication
"2D" Algorithm (ScaLAPACK)	$O\left(\frac{n^3}{P}\right)$	$O\left(\frac{n^2}{\sqrt{P}}\right)$
Lower Bound	$\Omega\left(\frac{n^3}{P}\right)$	$\Omega\left(\frac{n^3}{P\sqrt{M}}\right)$

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

Costs of matrix multiplication algorithms

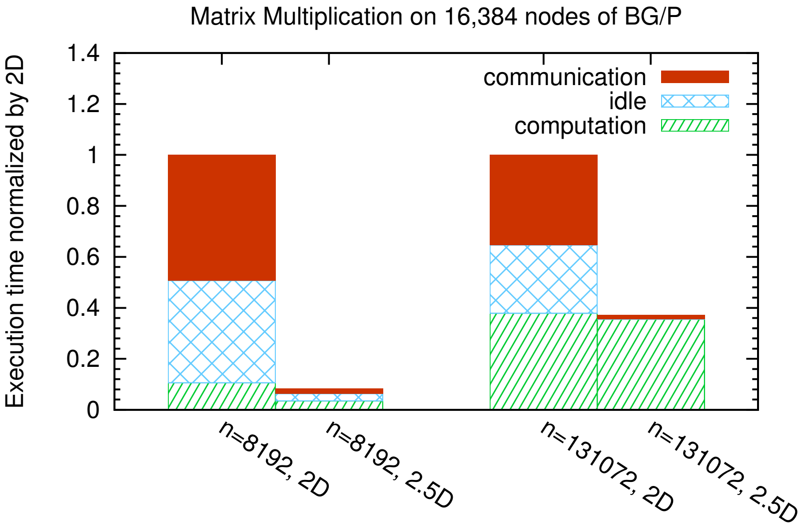
n = matrix dimension
P = number of processors
M = size of the local memory



	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

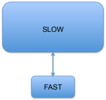
Performance improvement in practice



Lower bounds for classical matrix multiplication

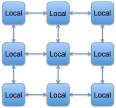
- Assume $\Theta(n^3)$ algorithm
- Sequential case with fast memory of size M
  - lower bound on words moved between fast/slow mem:

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



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

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



  - also attainable

Extensions to the rest of linear algebra

Theorem (Ballard, Demmel, Holtz, Schwartz 11)

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

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

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

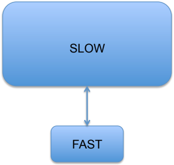
$$\#\ \textit{words} = \Omega\left(\frac{\#\ \textit{flops}}{\sqrt{\textit{memory size}}}\right)$$

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) [BDD$^+$12a]
LU	LAPACK (rarely) [Toledo 97]* [Grigori et al. 11]
QR	LAPACK (rarely) [Frens & Wise 03] [Elmroth & Gustavson 98]* [Hoemmen et al. 12]*
Eig, SVD	[BDK12a], [BDD12b]

Parallel Case

Example: Symmetric Indefinite Linear Solve

Suppose we want to solve $Ax = b$ where A

- is symmetric (save half the storage and flops)
- but indefinite (need to permute rows/cols for numerical stability)

We generally want to compute a factorization

$$PAP^T = LTL^T$$

P is a permutation, L is triangular, and T is symmetric and "simpler"

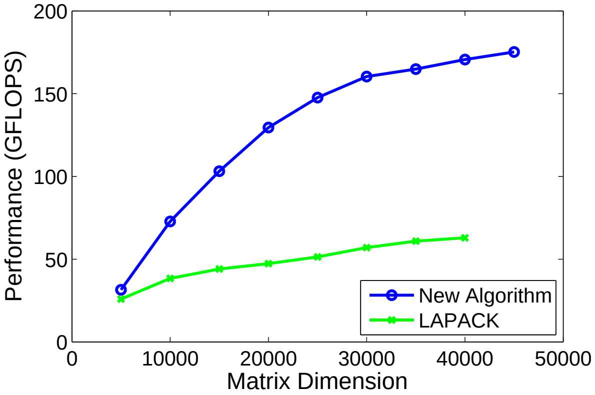
Symmetric Indefinite Factorization

We're solving $Ax = b$ where $A = A^T$ but A is indefinite

- Standard approach is to compute $PAP^T = LDL^T$
  - L is lower triangular and D is block diagonal (1×1 and 2×2 blocks)
  - requires complicated pivoting, harder to do tournament pivoting
- Alternative approach is to compute $PAP^T = LTL^T$ [Aas71]
  - L is lower triangular and T is tridiagonal
  - pivoting is more like LU (nonsymmetric case)

Reducing communication improves performance

Performance of symmetric indefinite linear system solvers on 48-core AMD Opteron node



Implemented within PLASMA library [BBD+13]

This work received a *Best Paper Award* at IPDPS '13

Example Application: Video Background Subtraction

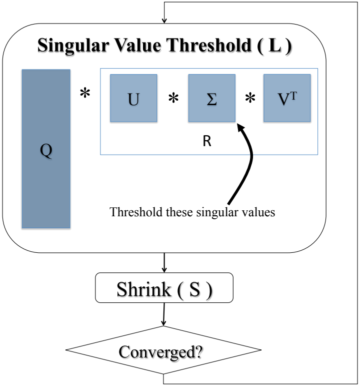
Idea: use Robust PCA algorithm [Candes et al. 09] to subtract constant background from the action of a surveillance video

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]

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

$$\begin{array}{c} n/2\{ \\ n/2\{ \end{array} \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \bullet \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Flop count recurrence:

$$F(n) = 7 \cdot F(n/2) + \Theta(n^2)$$
$$F(n) = \Theta\left(n^{\log_2 7}\right)$$

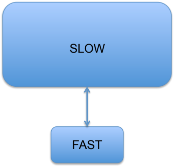
$$\log_2 7 \approx 2.81$$

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

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

Sequential Communication Costs

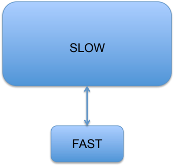
If you implement Strassen's algorithm recursively on a sequential computer:



	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

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

and on a parallel machine, it must communicate

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

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

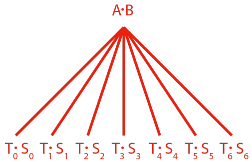
What about the parallel case?

- 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

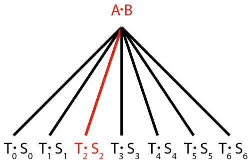
At each level of recursion tree, choose either breadth-first or depth-first traversal of the recursion tree

Breadth-First-Search (BFS)



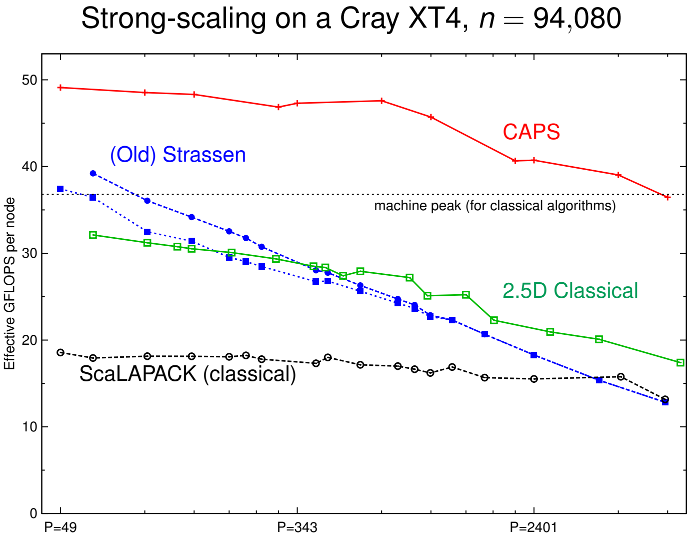
- Runs all 7 multiplies in parallel
  - each uses $P/7$ processors
- Requires $7/4$ as much extra memory
- Requires communication, but minimizes communication in subtrees

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?

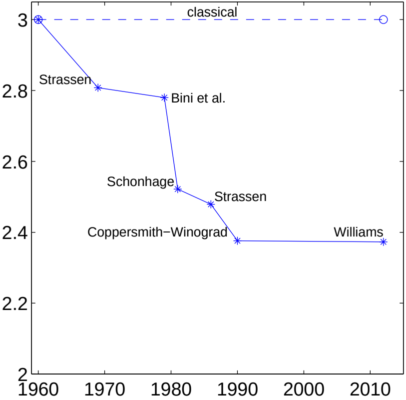
Strassen's algorithm allows for less computation and communication than the classical $O(n^3)$ algorithm

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?

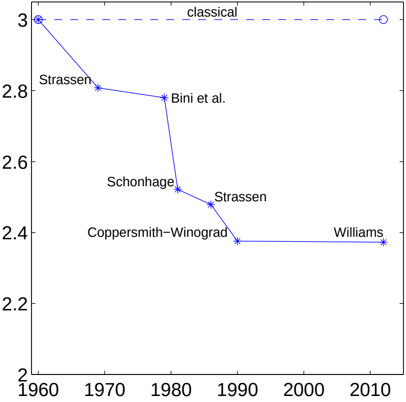
Exponent of matrix multiplication over time



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

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

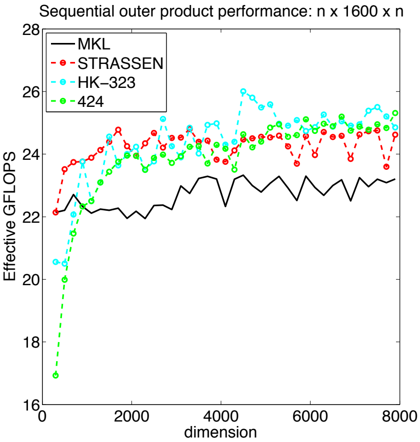
- involve approximations
- are existence proofs
- have large constants

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



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)

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]

Avoiding Communication in Linear Algebra

Grey Ballard

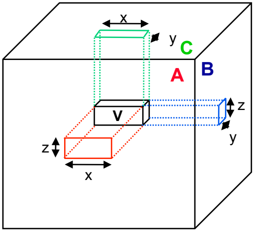
Thank You!

www.sandia.gov/~gmballa

Extra Slides

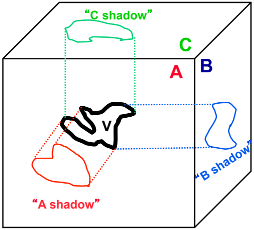
Main Idea of Classical Lower Bound Proof

Crux of proof based on geometric inequality [Loomis & Whitney 49]



Volume of box

$$V = xyz = \sqrt{xz \cdot yz \cdot xy}$$



Volume of a 3D set

$$V \leq \sqrt{\text{area(A shadow)}} \cdot \sqrt{\text{area(B shadow)}} \cdot \sqrt{\text{area(C shadow)}}$$

Given limited set of data, how much useful computation can be done?

Memory-Independent Lower Bounds

Summary for matrix multiplication:

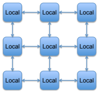
	Classical	Strassen
Memory-dependent lower bound	$\Omega\left(\frac{n^3}{P\sqrt{M}}\right)$	$\Omega\left(\frac{n^\omega}{PM^{\omega/2-1}}\right)$
Memory-independent lower bound	$\Omega\left(\frac{n^2}{P^{2/3}}\right)$	$\Omega\left(\frac{n^2}{P^{2/\omega}}\right)$
Perfect strong scaling range	$P = O\left(\frac{n^3}{M^{3/2}}\right)$	$P = O\left(\frac{n^\omega}{M^{\omega/2}}\right)$
Attaining algorithm	[SD11]	[BDH$^+$12]

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

Algorithm	Reference	Factor exceeding lower bound for # words	Factor exceeding lower bound for # messages
Matrix Multiply	[Can69]	1	1
Cholesky	ScaLAPACK	$\log P$	$\log P$
Symmetric Indefinite	[BDD$^+$12a] ScaLAPACK	? $\log P$	? $(N/P^{1/2}) \log P$
LU	[GDX11] ScaLAPACK	$\log P$ $\log P$	$\log P$ $(N/P^{1/2}) \log P$
QR	[DGHL12] ScaLAPACK	$\log P$ $\log P$	$\log^3 P$ $(N/P^{1/2}) \log P$
SymEig, SVD	[BDK12a] ScaLAPACK	? $\log P$	? $N/P^{1/2}$
NonsymEig	[BDD12b] ScaLAPACK	$\log P$ $P^{1/2} \log P$	$\log^3 P$ $N \log P$

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

Red = not optimal



Back

Example: Compute Eigenvalues of Band Matrix

Suppose we want to solve $Ax = \lambda x$ where A

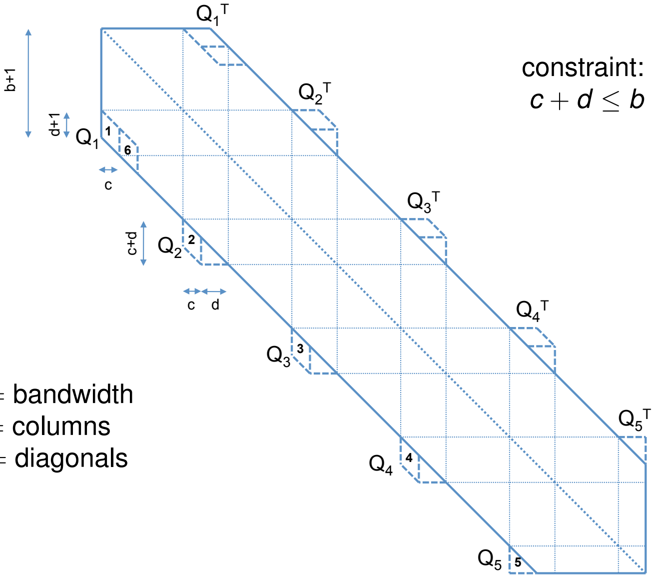
- is symmetric (save half the storage and flops)
- has band structure (exploit sparsity – ignore zeros)

We generally want to compute a factorization

$$A = QTQ^T$$

Q is an orthogonal matrix and T is symmetric tridiagonal

Successive Band Reduction (bulge-chasing)

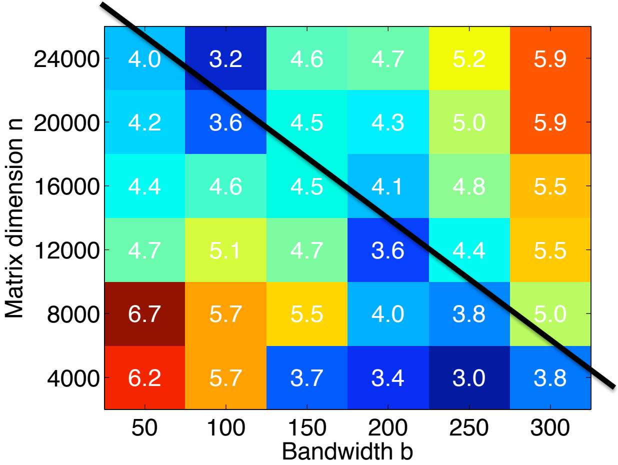


constraint:
$c + d \leq b$

$b =$ bandwidth
$c =$ columns
$d =$ diagonals

Implementation of Band Eigensolver (CASBR)

Speedup of parallel CASBR (10 threads) over PLASMA library



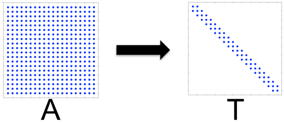
Benchmarked on 10-core Intel Westmere [BDK12a]

▸ Algorithmic Details

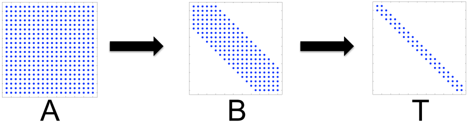
We're solving the symmetric eigenproblem via reduction to tridiagonal form

- Conventional approach (e.g. LAPACK) is direct tridiagonalization
- Two-phase approach reduces first to band, then band to tridiagonal

Direct:



Two-step:



- first phase can be done efficiently
- second phase is trickier, requires successive band reduction (SBR) [BLS00]
  - involves "bulge-chasing"
  - we've improved it to reduce communication [BDK12b]

Back

Communication-Avoiding SBR - theory

	Flops	Words Moved	Data Re-use
Schwarz	$4n^2b$	$O(n^2b)$	$O(1)$
M-H	$6n^2b$	$O(n^2b)$	$O(1)$
B-L-S*	$5n^2b$	$O(n^2 \log b)$	$O\left(\frac{b}{\log b}\right)$
CA-SBR†	$5n^2b$	$O\left(\frac{n^2b^2}{M}\right)$	$O\left(\frac{M}{b}\right)$

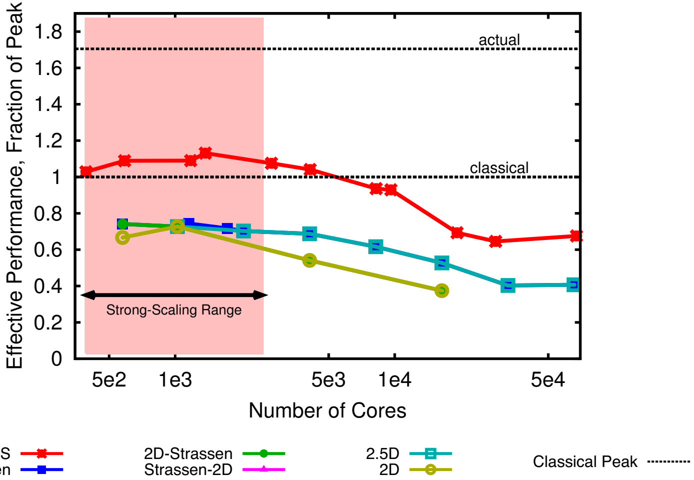
*with optimal parameter choices

†assuming $1 \le b \le \sqrt{M}/3$

Back

Performance of CAPS on large problems

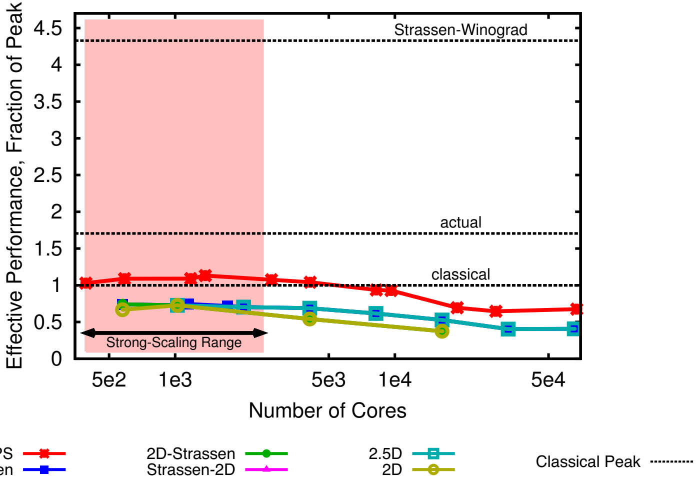
Strong-scaling on Intrepid (IBM BG/P), $n = 65{,}856$.



Back

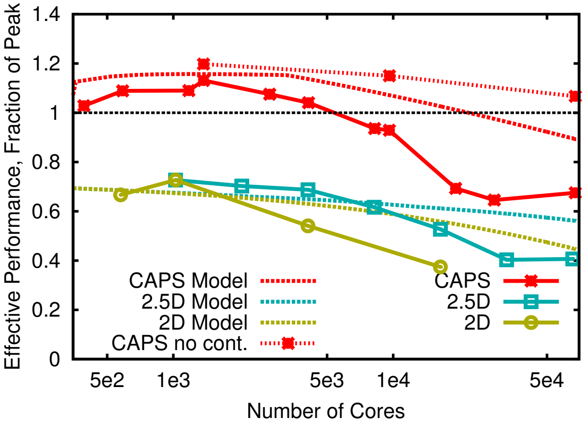
Performance of CAPS on large problems

Strong-scaling on Intrepid (IBM BG/P), $n = 65{,}856$.



Back

Performance: Model vs Actual



Comparison of the parallel models with the algorithms in strong scaling of matrix dimension $n = 65{,}856$ on Intrepid.

Back

Can an $n \times n$ linear system of equations $Ax = b$ be solved in $O(n^{2+\varepsilon})$ operations, where ε is arbitrarily small?

. . . if solved affirmatively, [this] would change the world.

It is an article of faith for some of us that if $O(n^{2+\varepsilon})$ is ever achieved, the big idea that achieves it will correspond to an algorithm that is really practical.

-Nick Trefethen, 2012 SIAM President

Solving the base case...

$2 \times 2 \times 2$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

multiplies	6	7	8
flop count	$O\left(n^{2.58}\right)$	$O\left(n^{2.81}\right)$	$O\left(n^{3}\right)$

Solving the base case. . .

$2 \times 2 \times 2$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

multiplies	~~6~~	7	8
flop count	$\cancel{O\left(n^{2.58}\right)}$	$O\left(n^{2.81}\right)$	$O\left(n^3\right)$

Solving the base case...

$2 \times 2 \times 2$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

multiplies	~~6~~	7	8
flop count	~~$O\left(n^{2.58}\right)$~~	$O\left(n^{2.81}\right)$	$O\left(n^3\right)$

$3 \times 3 \times 3$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

multiplies	19	21	23	27
flop count	$O\left(n^{2.68}\right)$	$O\left(n^{2.77}\right)$	$O\left(n^{2.85}\right)$	$O\left(n^3\right)$

References I

J. O. Aasen.
On the reduction of a symmetric matrix to tridiagonal form.
BIT Numerical Mathematics, 11:233–242, 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.
Implementing a blocked AasenÕs algorithm with a dynamic scheduler on multicore architectures, 2013.
To appear.

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, A. Druinsky, I. Peled, O. Schwartz, and S. Toledo.
Communication avoiding symmetric indefinite factorization, 2012.
In preparation.

G. Ballard, J. Demmel, and I. Dumitriu.
Communication-optimal parallel and sequential nonsymmetric eigenvalue algorithm, 2012.
In preparation.

References II

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.

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: regular submission.
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 the symmetric eigenproblem and SVD, 2012.
In preparation.

G. Ballard, J. Demmel, and N. Knight.
Communication avoiding successive band reduction.
In *Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, PPoPP '12, pages 35–44, New York, NY, USA, 2012. ACM.

C. Bischof, B. Lang, and X. Sun.
A framework for symmetric band reduction.
ACM Trans. Math. Soft., 26(4):581–601, December 2000.

References III

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.

E. Solomonik and J. Demmel.
Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms.
In Emmanuel Jeannot, Raymond Namyst, and Jean Roman, editors, *Euro-Par 2011 Parallel Processing*, volume 6853 of *Lecture Notes in Computer Science*, pages 90–109. Springer Berlin / Heidelberg, 2011.