

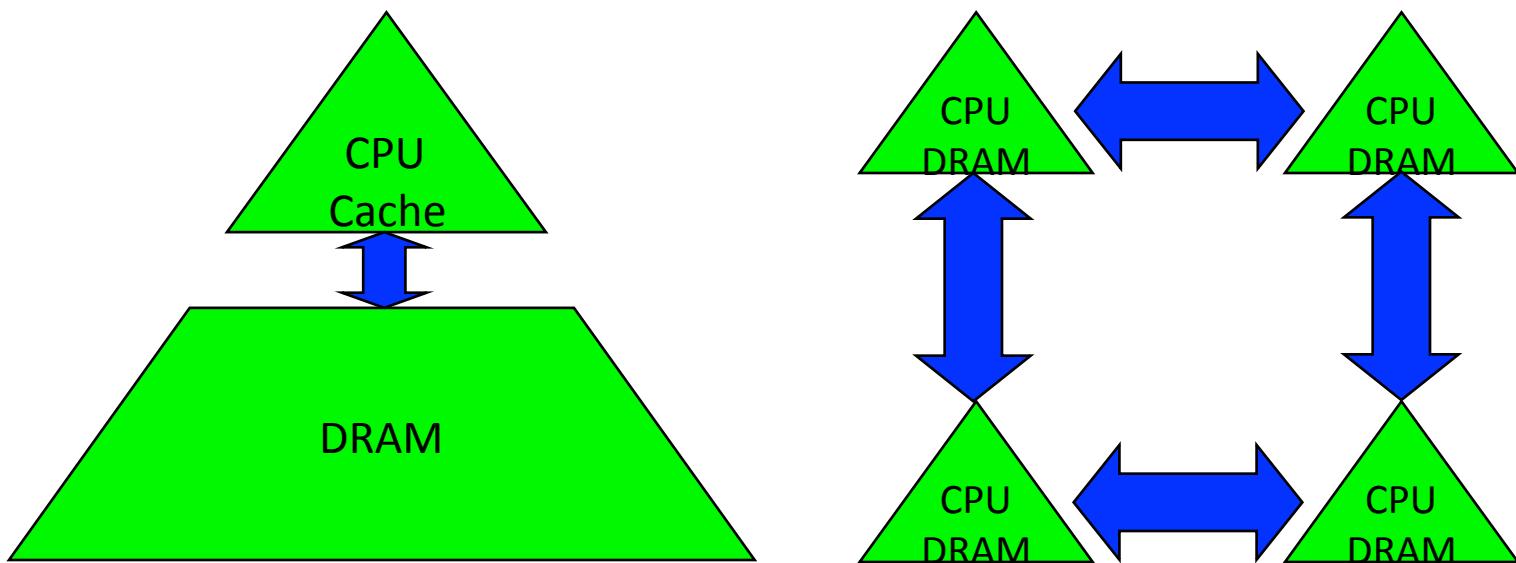
Communication-Avoiding Algorithms for Linear Algebra and Beyond

Jim Demmel,
EECS & Math, UC Berkeley

Why avoid communication? (1/3)

Algorithms have two costs (measured in time or energy):

1. Arithmetic (FLOPS)
2. Communication: moving data between
 - levels of a memory hierarchy (sequential case)
 - processors over a network (parallel case).



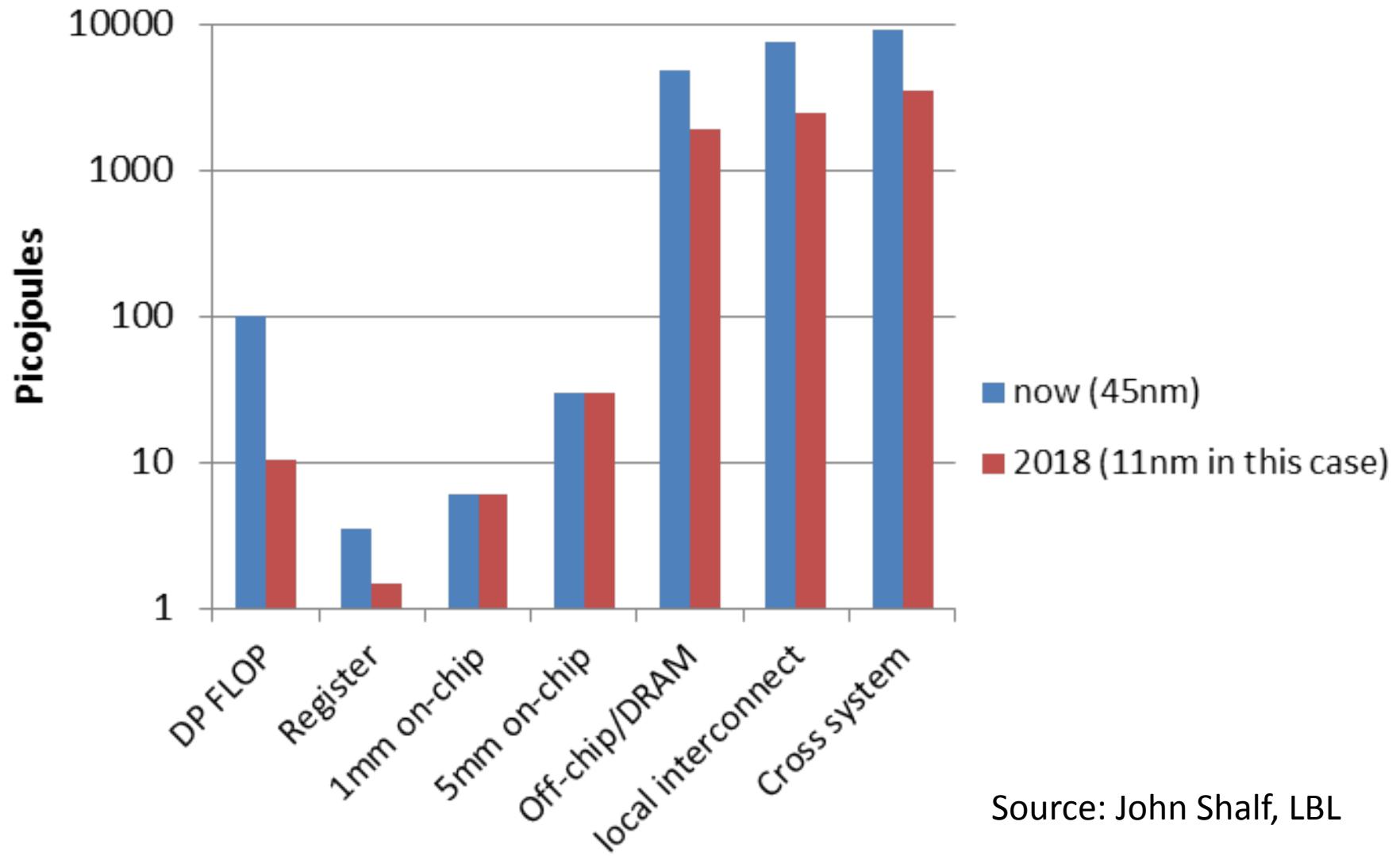
Why avoid communication? (2/3)

- Running time of an algorithm is sum of 3 terms:
 - $\# \text{ flops} * \text{time_per_flop}$
 - $\# \text{ words moved} / \text{bandwidth}$
 - $\# \text{ messages} * \text{latency}$
- $\text{Time_per_flop} \ll 1/\text{bandwidth} \ll \text{latency}$
 - Gaps growing exponentially with time [FOSC]

Annual improvements			
Time_per_flop		Bandwidth	Latency
59%	Network	26%	15%
	DRAM	23%	5%

- Avoid communication to save time
- Goal : reorganize algorithms to *avoid* communication

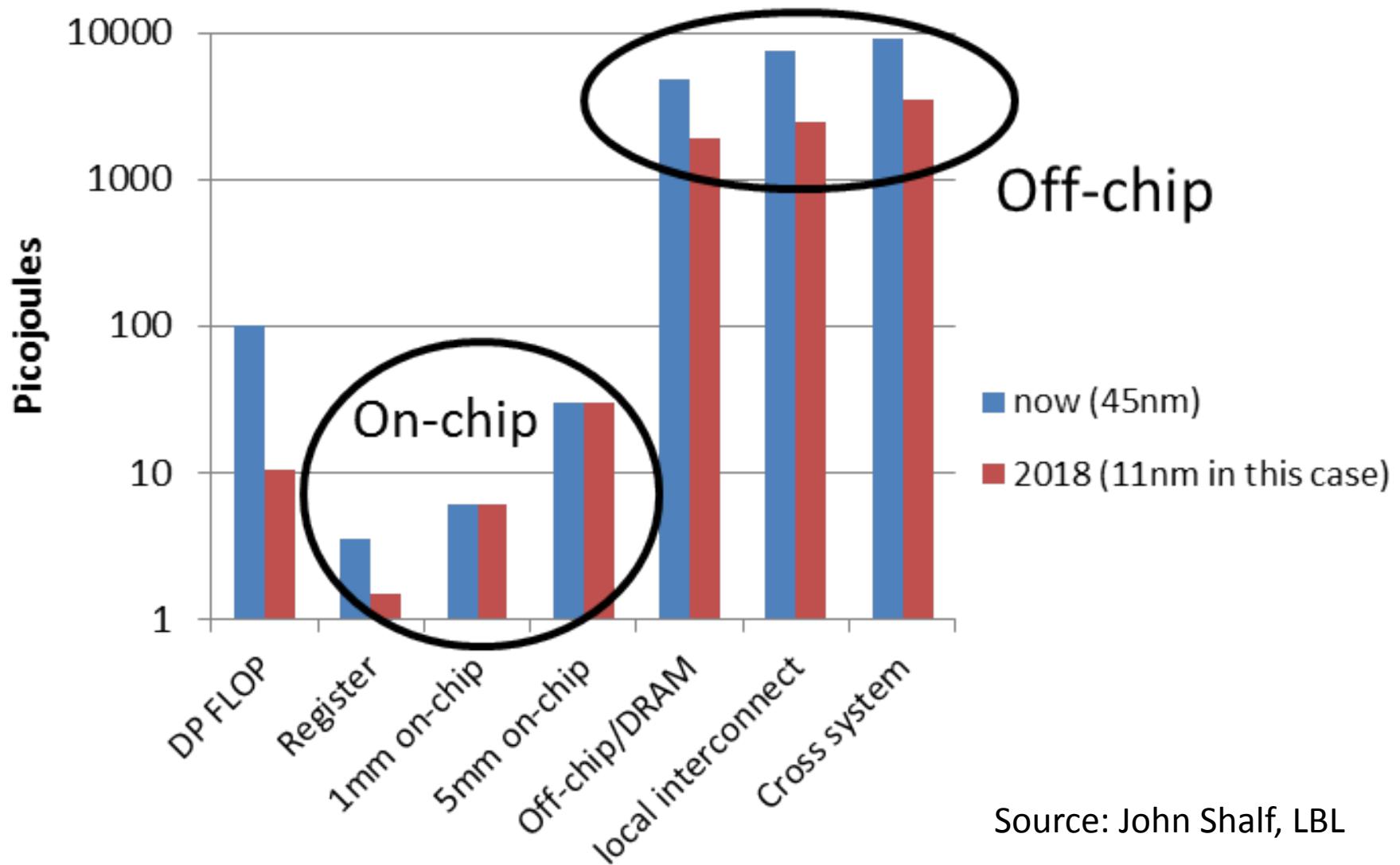
Why Minimize Communication? (3/3)



Source: John Shalf, LBL

Why Minimize Communication? (3/3)

Minimize communication to save energy



Source: John Shalf, LBL

Goals

- Redesign algorithms to *avoid* communication
 - Between all memory hierarchy levels
 - L1 \leftrightarrow L2 \leftrightarrow DRAM \leftrightarrow network, etc
- Attain lower bounds if possible
 - Current algorithms often far from lower bounds
 - Large speedups and energy savings possible

President Obama cites Communication-Avoiding Algorithms in the FY 2012 Department of Energy Budget Request to Congress:

“New Algorithm Improves Performance and Accuracy on Extreme-Scale Computing Systems. **On modern computer architectures, communication between processors takes longer than the performance of a floating point arithmetic operation by a given processor.** ASCR researchers have developed a new method, derived from commonly used linear algebra methods, to **minimize communications between processors and the memory hierarchy, by reformulating the communication patterns specified within the algorithm.** This method has been implemented in the TRILINOS framework, a highly-regarded suite of software, which provides functionality for researchers around the world to solve large scale, complex multi-physics problems.”

FY 2010 Congressional Budget, Volume 4, FY2010 Accomplishments, Advanced Scientific Computing Research (ASCR), pages 65-67.

CA-GMRES (Hoemmen, Mohiyuddin, Yelick, JD)
“Tall-Skinny” QR (Grigori, Hoemmen, Langou, JD)

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
 - TSQR: Tall-Skinny QR
 - CA $O(n^3)$ 2.5D Matmul
 - CA Strassen Matmul
- Beyond linear algebra
 - Extending lower bounds to any algorithm with arrays
 - Communication-optimal N-body algorithm
- CA-Krylov methods

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
 - TSQR: Tall-Skinny QR
 - CA $O(n^3)$ 2.5D Matmul
 - CA Strassen Matmul
- Beyond linear algebra
 - Extending lower bounds to any algorithm with arrays
 - Communication-optimal N-body algorithm
- CA-Krylov methods

Summary of CA Linear Algebra

- “Direct” Linear Algebra
 - Lower bounds on communication for linear algebra problems like $Ax=b$, least squares, $Ax = \lambda x$, SVD, etc
 - Mostly not attained by algorithms in standard libraries
 - New algorithms that attain these lower bounds
 - Being added to libraries: Sca/LAPACK, PLASMA, MAGMA
 - Large speed-ups possible
 - Autotuning to find optimal implementation
- Ditto for “Iterative” Linear Algebra

Lower bound for all “n³-like” linear algebra

- Let M = “fast” memory size (per processor)

$$\text{#words_moved (per processor)} = \Omega(\text{\#flops (per processor)} / M^{1/2})$$

- Parallel case: assume either load or memory balanced
- Holds for
 - Matmul

Lower bound for all “n³-like” linear algebra

- Let M = “fast” memory size (per processor)

$$\# \text{words_moved (per processor)} = \Omega(\# \text{flops (per processor)} / M^{1/2})$$

$$\# \text{messages_sent} \geq \# \text{words_moved} / \text{largest_message_size}$$

- Parallel case: assume either load or memory balanced
- Holds for
 - Matmul, BLAS, LU, QR, eig, SVD, tensor contractions, ...
 - Some whole programs (sequences of these operations, no matter how individual ops are interleaved, eg A^k)
 - Dense and sparse matrices (where #flops << n³)
 - Sequential and parallel algorithms
 - Some graph-theoretic algorithms (eg Floyd-Warshall)

Lower bound for all “n³-like” linear algebra

- Let M = “fast” memory size (per processor)

$$\text{#words_moved (per processor)} = \Omega(\text{\#flops (per processor)} / M^{1/2})$$

$$\text{#messages_sent (per processor)} = \Omega(\text{\#flops (per processor)} / M^{3/2})$$

- Parallel case: assume either load or memory balanced
- Holds for
 - Matmul, BLAS, LU, QR, eig, SVD, tensor contractions, ...
 - Some whole programs (sequences of these operations, no matter how individual ops are interleaved, eg A^k)

SIAM SIAG/Linear Algebra Prize, 2012

Ballard, D., Holtz, Schwartz

Can we attain these lower bounds?

- Do conventional dense algorithms as implemented in LAPACK and ScaLAPACK attain these bounds?
 - Often not
- If not, are there other algorithms that do?
 - Yes, for much of dense linear algebra
 - New algorithms, with new numerical properties, new ways to encode answers, new data structures
 - Not just loop transformations (need those too!)
- Only a few sparse algorithms so far
- Lots of work in progress

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
 - TSQR: Tall-Skinny QR
 - CA $O(n^3)$ 2.5D Matmul
 - CA Strassen Matmul
- Beyond linear algebra
 - Extending lower bounds to any algorithm with arrays
 - Communication-optimal N-body algorithm
- CA-Krylov methods

TSQR: QR of a Tall, Skinny matrix

$$W = \begin{pmatrix} W_0 \\ \hline W_1 \\ \hline W_2 \\ \hline W_3 \end{pmatrix}$$

$$\begin{pmatrix} R_{00} \\ \hline R_{10} \\ \hline R_{20} \\ \hline R_{30} \end{pmatrix} = \begin{pmatrix} Q_{01} & R_{01} \\ \hline Q_{11} & R_{11} \end{pmatrix}$$

$$\begin{pmatrix} R_{01} \\ \hline R_{11} \end{pmatrix} = \begin{pmatrix} Q_{02} & R_{02} \end{pmatrix}$$

TSQR: QR of a Tall, Skinny matrix

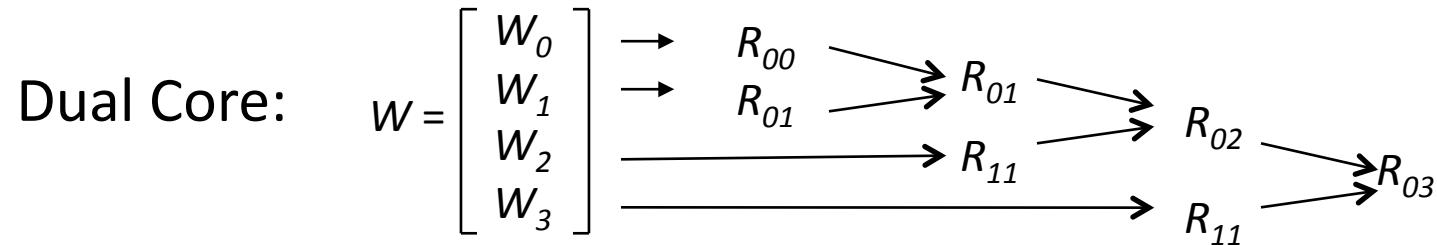
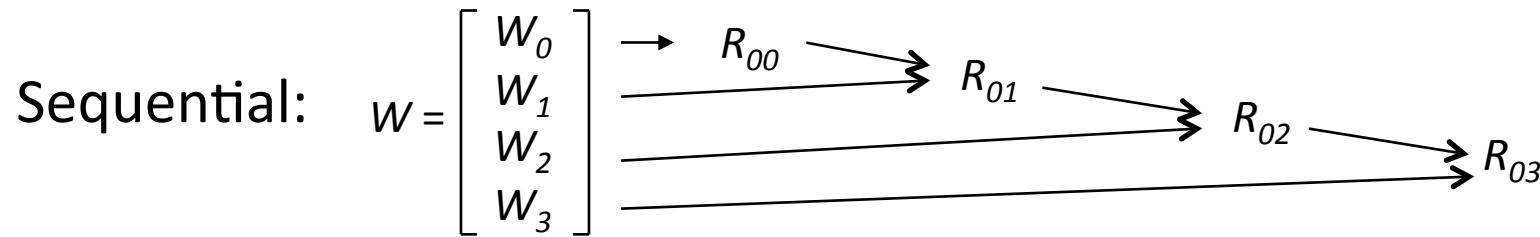
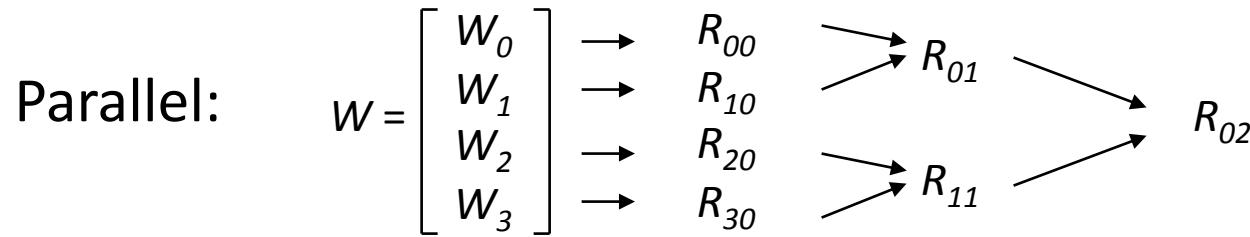
$$W = \begin{pmatrix} W_0 \\ \hline W_1 \\ \hline W_2 \\ \hline W_3 \end{pmatrix} = \begin{pmatrix} Q_{00} & R_{00} \\ \hline Q_{10} & R_{10} \\ \hline Q_{20} & R_{20} \\ \hline Q_{30} & R_{30} \end{pmatrix} = \begin{pmatrix} Q_{00} \\ \hline Q_{10} \\ \hline Q_{20} \\ \hline Q_{30} \end{pmatrix} \cdot \begin{pmatrix} R_{00} \\ \hline R_{10} \\ \hline R_{20} \\ \hline R_{30} \end{pmatrix}$$

$$\begin{pmatrix} R_{00} \\ \hline R_{10} \\ \hline R_{20} \\ \hline R_{30} \end{pmatrix} = \begin{pmatrix} Q_{01} & R_{01} \\ \hline Q_{11} & R_{11} \end{pmatrix} = \begin{pmatrix} Q_{01} \\ \hline Q_{11} \end{pmatrix} \cdot \begin{pmatrix} R_{01} \\ \hline R_{11} \end{pmatrix}$$

$$\begin{pmatrix} R_{01} \\ \hline R_{11} \end{pmatrix} = \begin{pmatrix} Q_{02} & R_{02} \end{pmatrix}$$

$$\text{Output} = \{ Q_{00}, Q_{10}, Q_{20}, Q_{30}, Q_{01}, Q_{11}, Q_{02}, R_{02} \}$$

TSQR: An Architecture-Dependent Algorithm



Multicore / Multisocket / Multirack / Multisite / Out-of-core: ?

Can choose reduction tree dynamically

TSQR Performance Results

- Parallel
 - Intel Clovertown
 - Up to **8x** speedup (8 core, dual socket, 10M x 10)
 - Pentium III cluster, Dolphin Interconnect, MPICH
 - Up to **6.7x** speedup (16 procs, 100K x 200)
 - BlueGene/L
 - Up to **4x** speedup (32 procs, 1M x 50)
 - Tesla C 2050 / Fermi
 - Up to **13x** ($110,592 \times 100$)
 - Grid – **4x** on 4 cities vs 1 city (Dongarra, Langou et al)
 - Cloud – **1.6x slower than accessing data twice** (Gleich and Benson)
- Sequential
 - “**Infinite speedup**” for out-of-core on PowerPC laptop
 - As little as 2x slowdown vs (predicted) infinite DRAM
 - LAPACK with virtual memory never finished
- SVD costs about the same
- Joint work with Grigori, Hoemmen, Langou, Anderson, Ballard, Keutzer, others

Summary of dense *parallel* algorithms attaining communication lower bounds

- Assume $n \times n$ matrices on P processors
- Minimum Memory per processor = $M = O(n^2 / P)$
- Recall lower bounds:
 $\#words_moved = \Omega((n^3 / P) / M^{1/2}) = \Omega(n^2 / P^{1/2})$
 $\#messages = \Omega((n^3 / P) / M^{3/2}) = \Omega(P^{1/2})$
- Does ScaLAPACK attain these bounds?
 - For $\#words_moved$: mostly, except nonsym. Eigenproblem
 - For $\#messages$: asymptotically worse, except Cholesky
- New algorithms attain all bounds, up to polylog(P) factors
 - Cholesky, LU, QR, Sym. and Nonsym eigenproblems, SVD

Can we do Better?

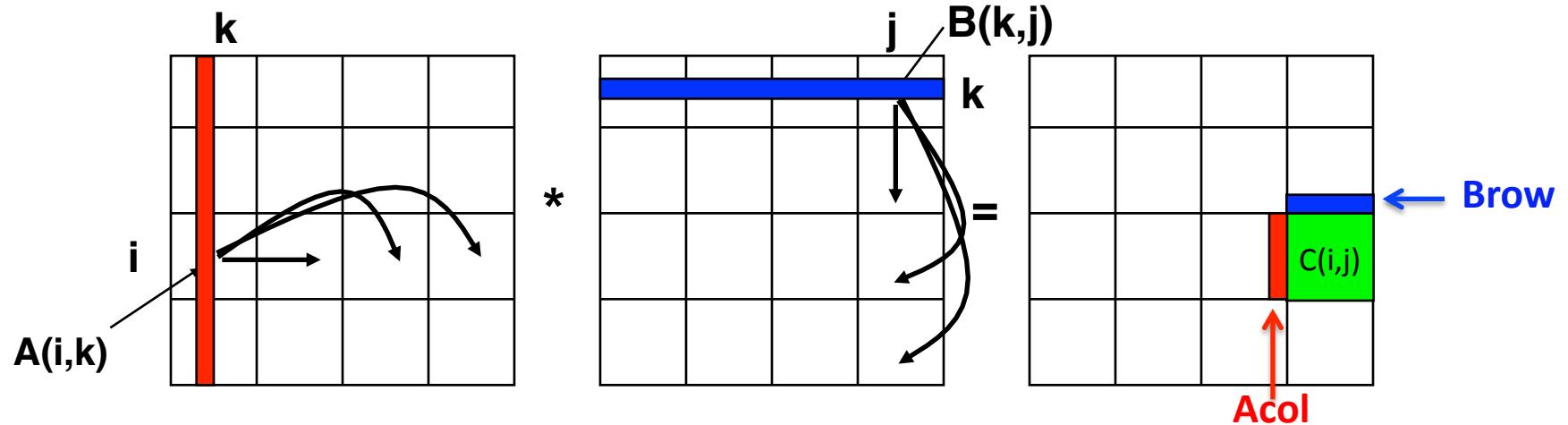
Can we do better?

- Aren't we already optimal?
- Why assume $M = O(n^2/p)$, i.e. minimal?
 - Lower bound still true if more memory
 - Can we attain it?

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
 - TSQR: Tall-Skinny QR
 - CA $O(n^3)$ 2.5D Matmul
 - CA Strassen Matmul
- Beyond linear algebra
 - Extending lower bounds to any algorithm with arrays
 - Communication-optimal N-body algorithm
- CA-Krylov methods

SUMMA– $n \times n$ matmul on $P^{1/2} \times P^{1/2}$ grid (nearly) optimal using minimum memory $M=O(n^2/P)$

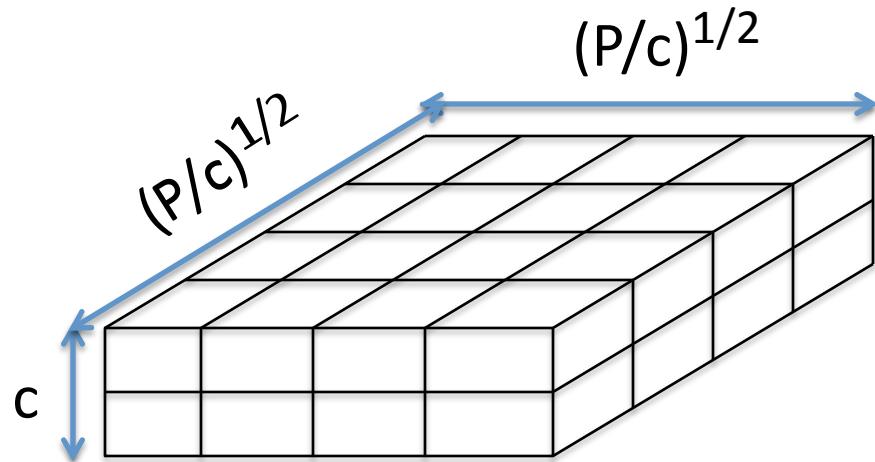


Using more than the minimum memory

- What if matrix small enough to fit $c > 1$ copies, so $M = cn^2/P$?
 - #words_moved = $\Omega(\#flops / M^{1/2}) = \Omega(n^2 / (c^{1/2} P^{1/2}))$
 - #messages = $\Omega(\#flops / M^{3/2}) = \Omega(P^{1/2} / c^{3/2})$
- Can we attain new lower bound?

2.5D Matrix Multiplication

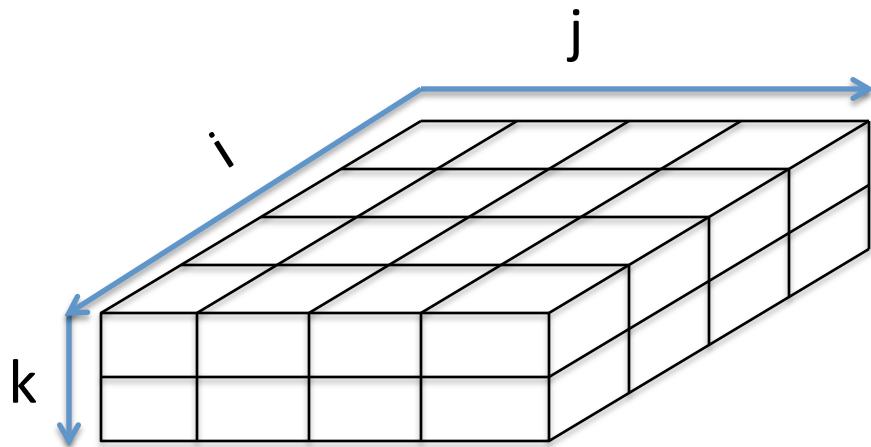
- Assume can fit cn^2/P data per processor, $c > 1$
- Processors form $(P/c)^{1/2} \times (P/c)^{1/2} \times c$ grid



Example: $P = 32, c = 2$

2.5D Matrix Multiplication

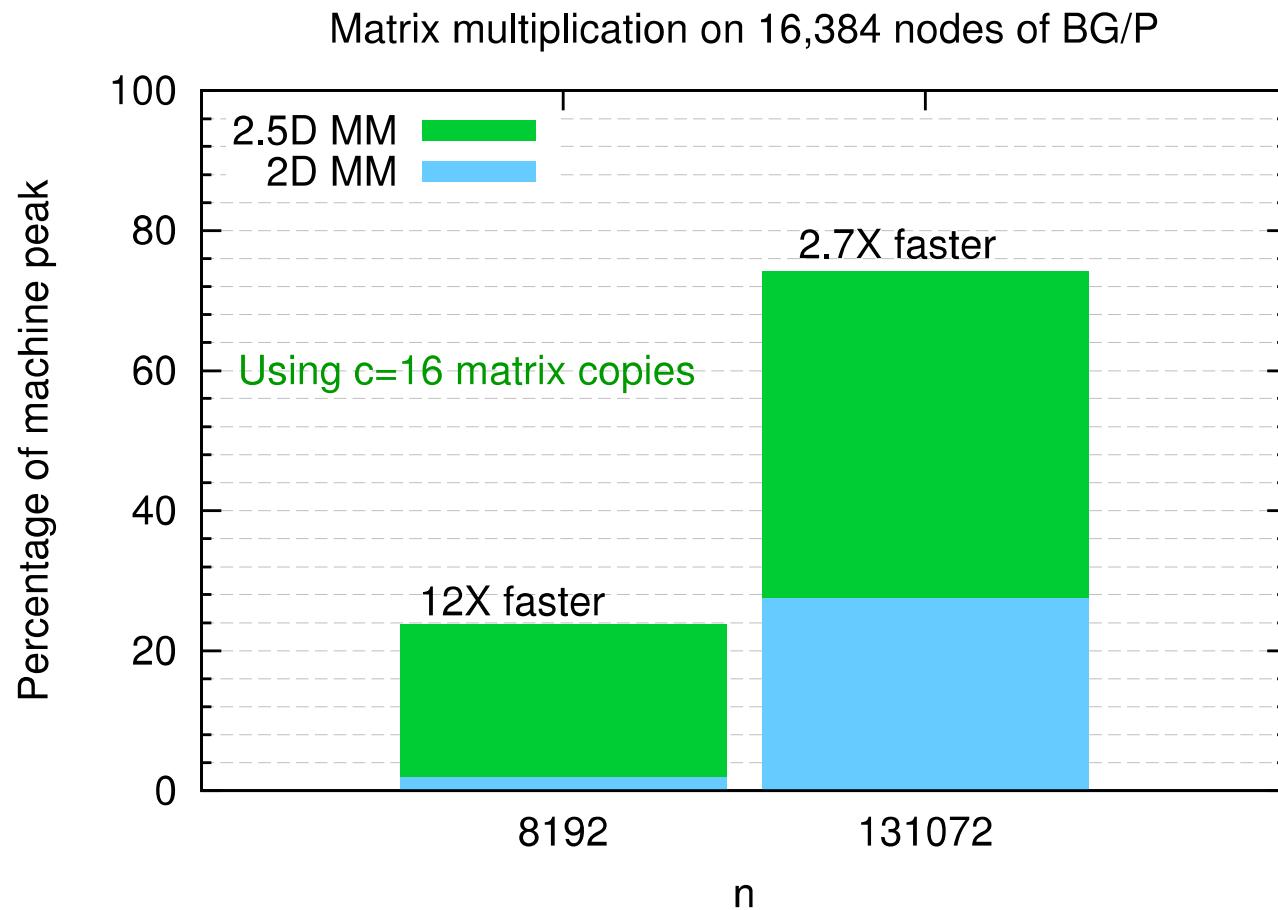
- Assume can fit cn^2/P data per processor, $c > 1$
- Processors form $(P/c)^{1/2} \times (P/c)^{1/2} \times c$ grid



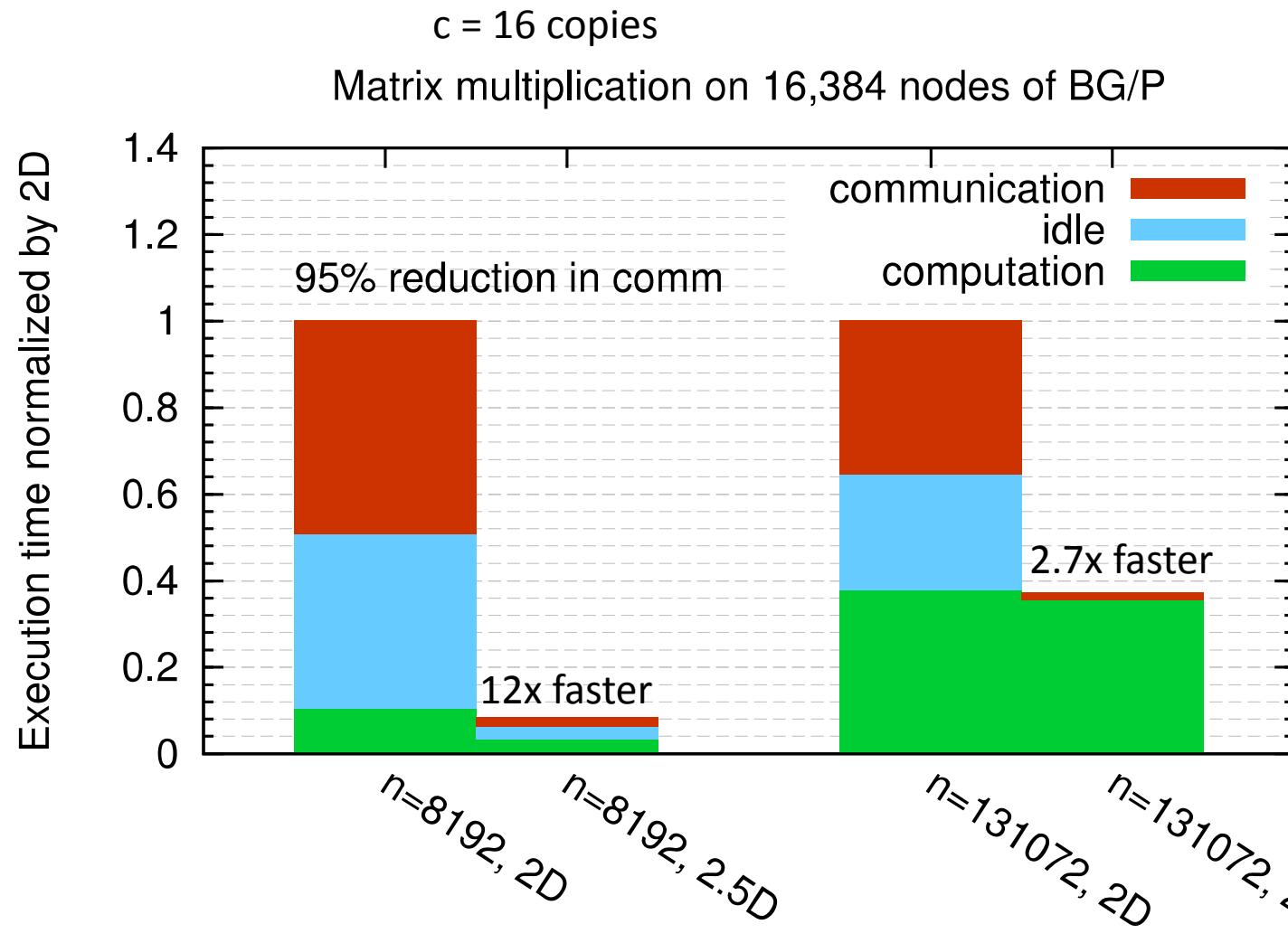
Initially $P(i,j,0)$ owns $A(i,j)$ and $B(i,j)$
each of size $n(c/P)^{1/2} \times n(c/P)^{1/2}$

- (1) $P(i,j,0)$ broadcasts $A(i,j)$ and $B(i,j)$ to $P(i,j,k)$
- (2) Processors at level k perform $1/c$ -th of SUMMA, i.e. $1/c$ -th of $\sum_m A(i,m)*B(m,j)$
- (3) Sum-reduce partial sums $\sum_m A(i,m)*B(m,j)$ along k -axis so $P(i,j,0)$ owns $C(i,j)$

2.5D Matmul on BG/P, 16K nodes / 64K cores

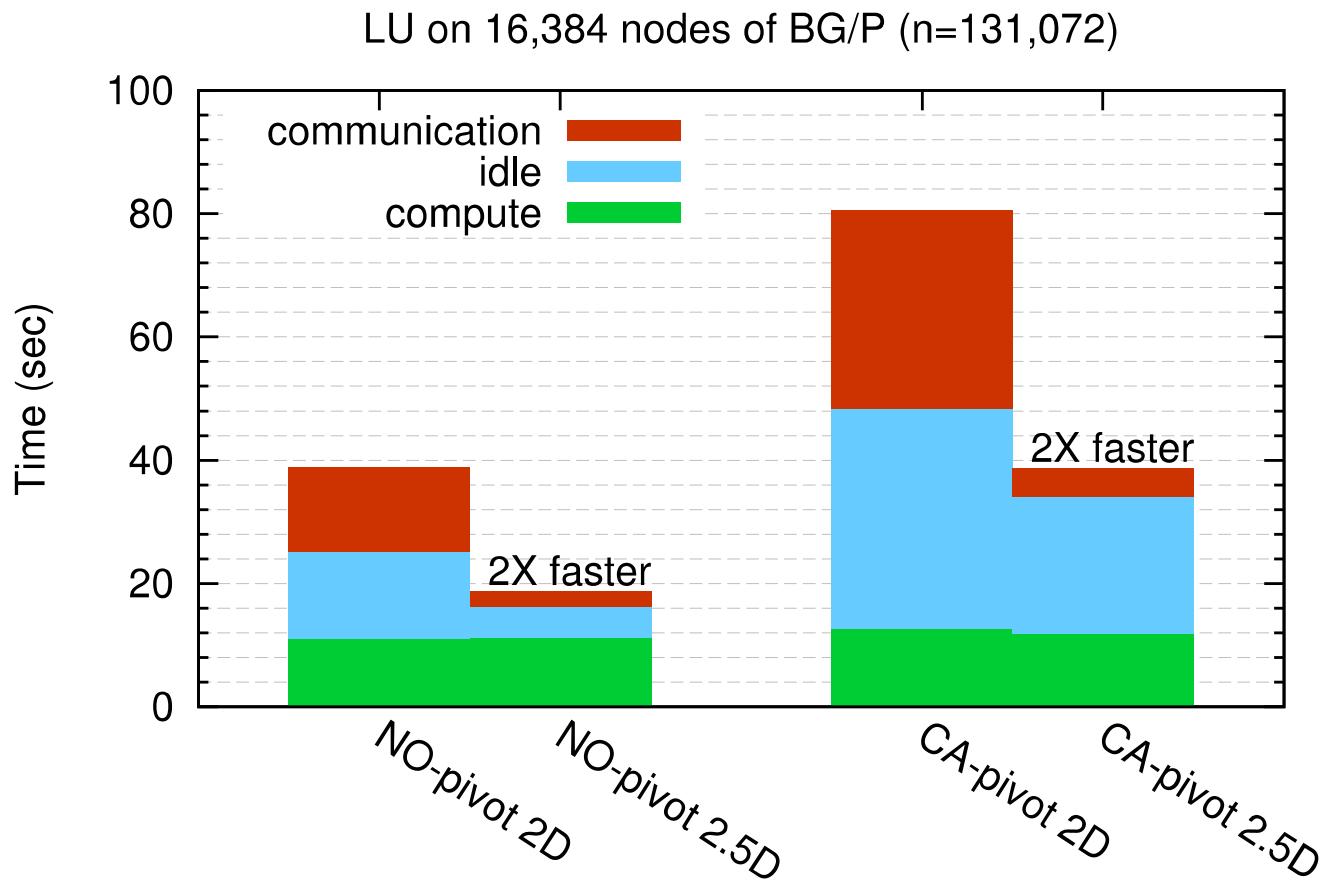


2.5D Matmul on BG/P, 16K nodes / 64K cores



Distinguished Paper Award, EuroPar'11 (Solomonik, D.)
SC'11 paper by Solomonik, Bhatele, D.

2.5D vs 2D LU With and Without Pivoting



Perfect Strong Scaling – in Time and Energy

- Every time you add a processor, you should use its memory M too
- Start with minimal number of procs: $PM = 3n^2$
- Increase P by a factor of $c \rightarrow$ total memory increases by a factor of c
- Notation for timing model:
 - $\gamma_T, \beta_T, \alpha_T$ = secs per flop, per word_moved, per message of size m
- $T(cP) = n^3/(cP) [\gamma_T + \beta_T/M^{1/2} + \alpha_T/(mM^{1/2})]$
 $= T(P)/c$
- Notation for energy model:
 - $\gamma_E, \beta_E, \alpha_E$ = joules for same operations
 - δ_E = joules per word of memory used per sec
 - ε_E = joules per sec for leakage, etc.
- $E(cP) = cP \{ n^3/(cP) [\gamma_E + \beta_E/M^{1/2} + \alpha_E/(mM^{1/2})] + \delta_E MT(cP) + \varepsilon_E T(cP) \}$
 $= E(P)$
- Perfect scaling extends to N-body, Strassen, ...

How to generalize these optimizations to programs that access arrays (with some technical assumptions...)

- Matmul

for $i=1:n$, for $j=1:n$, for $k=1:n$,

$$C(i,j) += A(i,k)*B(k,j)$$

=> for (i,j,k) in $S = \text{subset of } Z^3$

Access locations indexed by (i,j) , (i,k) , (k,j)

- General case

for $i_1=1:n$, for $i_2 = i_1:m$, ... for $i_k = i_3:i_4$... explicit loops not necessary

$$C(i_1+2*i_3-i_7) = \text{func}(A(i_2+3*i_4, i_1, i_2, i_1+i_2, \dots), B(\text{pnt}(3*i_4)), \dots)$$

$$D(\text{something else}) = \text{func}(\text{something else}), \dots$$

=> for (i_1, i_2, \dots, i_k) in $S = \text{subset of } Z^k$

Access locations indexed by group homomorphisms, eg

$$\phi_C(i_1, i_2, \dots, i_k) = (i_1+2*i_3-i_7)$$

$$\phi_A(i_1, i_2, \dots, i_k) = (i_2+3*i_4, i_1, i_2, i_1+i_2, \dots), \dots$$

- For details, see Nick Knight's talk

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
 - TSQR: Tall-Skinny QR
 - CA $O(n^3)$ 2.5D Matmul
 - CA Strassen Matmul
- Beyond linear algebra
 - Extending lower bounds to any algorithm with arrays
 - Communication-optimal N-body algorithm
- CA-Krylov methods

Communication Lower Bounds for Strassen-like matmul algorithms

Classical
 $O(n^3)$ matmul:

$$\#words_moved = \Omega(M(n/M^{1/2})^3/P)$$

Strassen's
 $O(n^{\lg 7})$ matmul:

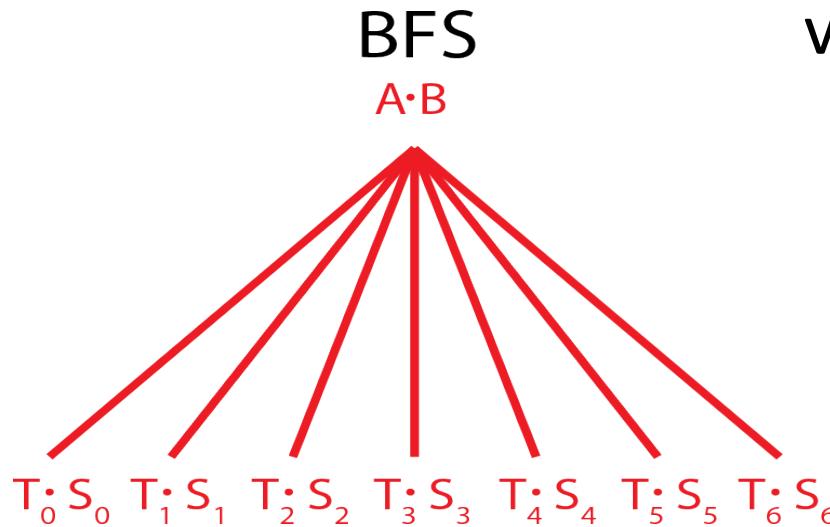
$$\#words_moved = \Omega(M(n/M^{1/2})^{\lg 7}/P)$$

Strassen-like
 $O(n^\omega)$ matmul:

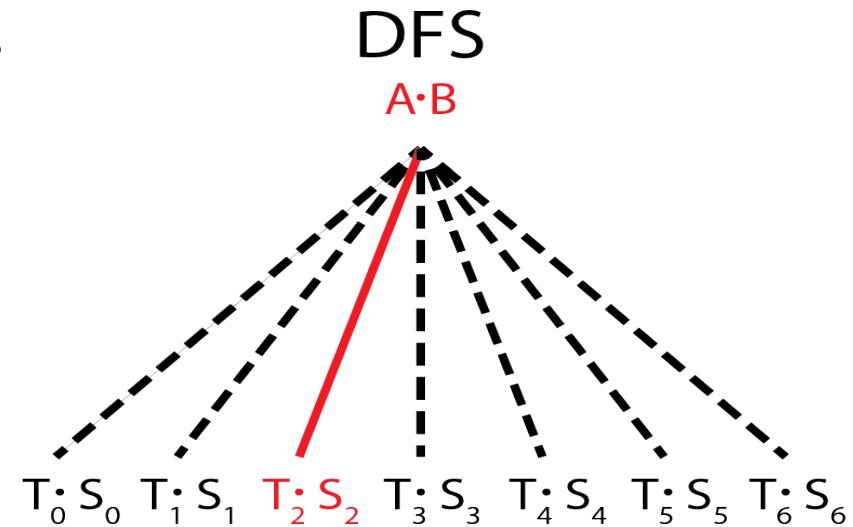
$$\#words_moved = \Omega(M(n/M^{1/2})^\omega/P)$$

- Proof: graph expansion (different from classical matmul)
 - Strassen-like: DAG must be “regular” and connected
- Extends up to $M = n^2 / p^{2/\omega}$
- Best Paper Prize (SPAA'11), Ballard, D., Holtz, Schwartz,
also in JACM
- Is the lower bound attainable?

Communication Avoiding Parallel Strassen (CAPS)



vs.



Runs all 7 multiplies in parallel
Each on P/7 processors
Needs 7/4 as much memory

Runs all 7 multiplies sequentially
Each on all P processors
Needs 1/4 as much memory

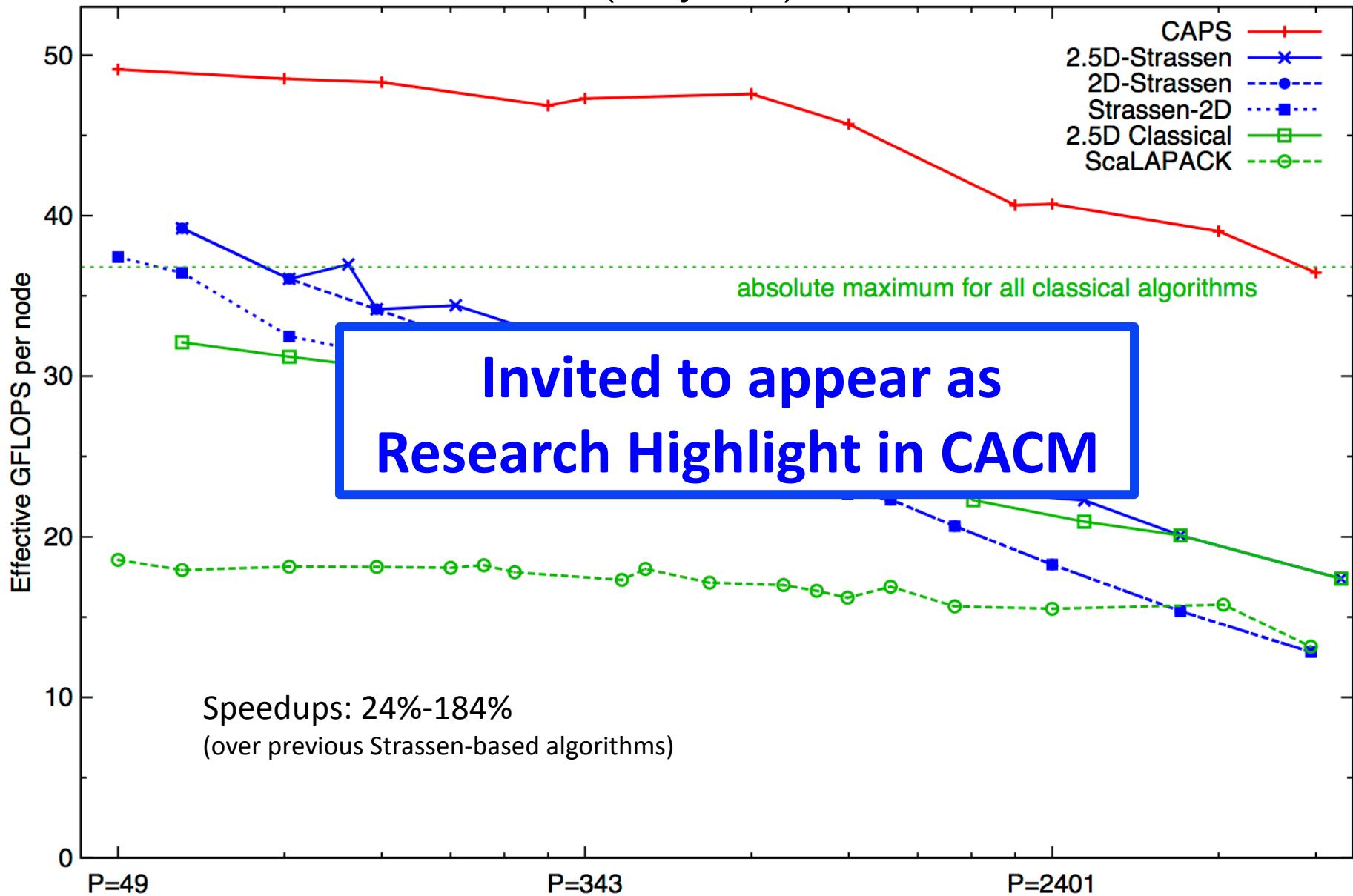
CAPS

```
If EnoughMemory and P ≥ 7
  then BFS step
  else DFS step
end if
```

Best way to interleave
BFS and DFS is an
tuning parameter

Performance Benchmarking, Strong Scaling Plot

Franklin (Cray XT4) $n = 94080$



Some Ongoing Work

- Lots more work on
 - Algorithms:
 - BLAS, LDL^T , QR with pivoting, other pivoting schemes, eigenproblems, ...
 - All-pairs-shortest-path, ...
 - N-body (Penporn Koanantakool)
 - Overlapping and Avoiding Communication (Evangelos Georganas)
 - Both 2D ($c=1$) and 2.5D ($c>1$)
 - But only bandwidth may decrease with $c>1$, not latency
 - Platforms:
 - Multicore, cluster, GPU, cloud, heterogeneous, low-energy, ...
 - Software:
 - Integration into Sca/LAPACK, PLASMA, MAGMA,...
- Integration into applications (on IBM BG/Q)
 - CTF (with ANL): symmetric tensor contractions

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
 - TSQR: Tall-Skinny QR
 - CA $O(n^3)$ 2.5D Matmul
 - CA Strassen Matmul
- Beyond linear algebra
 - Extending lower bounds to any algorithm with arrays
 - Communication-optimal N-body algorithm
- CA-Krylov methods

Recall optimal sequential Matmul

- Naïve code
for $i=1:n$, for $j=1:n$, for $k=1:n$, $C(i,j) += A(i,k)*B(k,j)$
- “Blocked” code
for $i_1 = 1:b:n$, for $j_1 = 1:b:n$, for $k_1 = 1:b:n$
for $i_2 = 0:b-1$, for $j_2 = 0:b-1$, for $k_2 = 0:b-1$
 $i = i_1 + i_2$, $j = j_1 + j_2$, $k = k_1 + k_2$
 $C(i,j) += A(i,k)*B(k,j)$
- Thm: Picking $b = M^{1/2}$ attains lower bound:
 $\#words_moved = \Omega(n^3/M^{1/2})$
- Where does $1/2$ come from?

New Thm applied to Matmul

- for $i=1:n$, for $j=1:n$, for $k=1:n$, $C(i,j) += A(i,k)*B(k,j)$
- Record array indices in matrix Δ

$$\Delta = \begin{pmatrix} i & j & k \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad \begin{matrix} A \\ B \\ C \end{matrix}$$

- Solve LP for $x = [x_i, x_j, x_k]^T$: $\max \mathbf{1}^T x$ s.t. $\Delta x \leq \mathbf{1}$
 - Result: $x = [1/2, 1/2, 1/2]^T$, $\mathbf{1}^T x = 3/2 = S_{HBL}$
- Thm: $\#words_moved = \Omega(n^3/M^{S_{HBL}-1}) = \Omega(n^3/M^{1/2})$
Attained by block sizes $M^{x_i}, M^{x_j}, M^{x_k} = M^{1/2}, M^{1/2}, M^{1/2}$

New Thm applied to Direct N-Body

- for $i=1:n$, for $j=1:n$, $F(i) += \text{force}(P(i), P(j))$
- Record array indices in matrix Δ

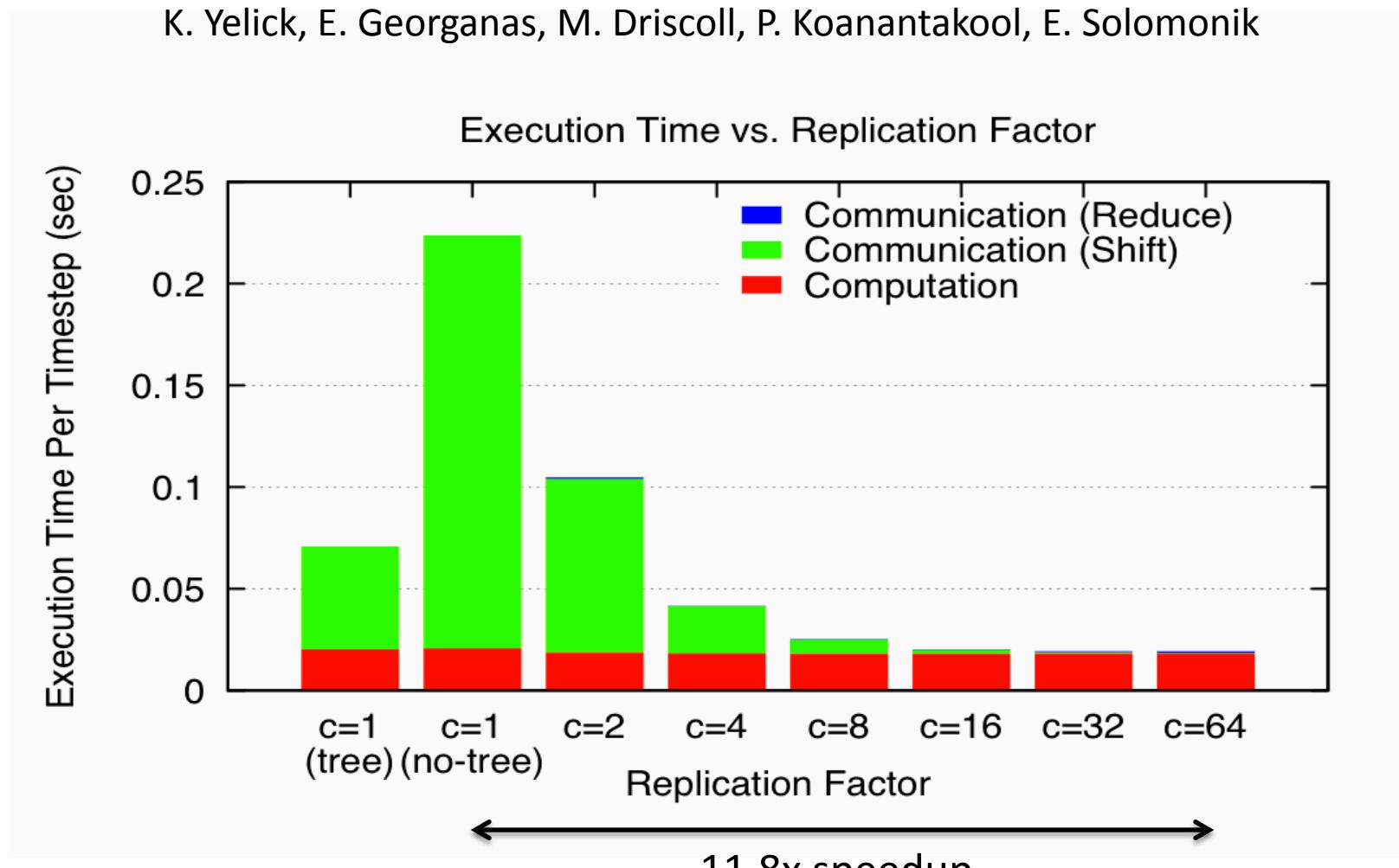
$$\Delta = \begin{pmatrix} & i & j \\ 1 & & 0 \\ 1 & & 0 \\ 0 & & 1 \end{pmatrix} \begin{array}{l} F \\ P(i) \\ P(j) \end{array}$$

- Solve LP for $x = [x_i, x_j]^T$: $\max \mathbf{1}^T x$ s.t. $\Delta x \leq \mathbf{1}$
 - Result: $x = [1, 1]$, $\mathbf{1}^T x = 2 = S_{HBL}$
- Thm: $\#words_moved = \Omega(n^2/M^{S_{HBL}-1}) = \Omega(n^2/M^1)$
Attained by block sizes $M^{x_i}, M^{x_j} = M^1, M^1$

N-Body Speedups on IBM-BG/P (Intrepid)

8K cores, 32K particles

K. Yelick, E. Georganas, M. Driscoll, P. Koanantakool, E. Solomonik



New Thm applied to Random Code

- for $i_1=1:n$, for $i_2=1:n$, ... , for $i_6=1:n$

$$A_1(i_1, i_3, i_6) += \text{func1}(A_2(i_1, i_2, i_4), A_3(i_2, i_3, i_5), A_4(i_3, i_4, i_6))$$

$$A_5(i_2, i_6) += \text{func2}(A_6(i_1, i_4, i_5), A_3(i_3, i_4, i_6))$$

- Record array indices
in matrix Δ

i_1	i_2	i_3	i_4	i_5	i_6	
1	0	1	0	0	1	A1
1	1	0	1	0	0	A2
0	1	1	0	1	0	A3
0	0	1	1	0	1	A3,A4
0	0	1	1	0	1	A5
1	0	0	1	1	0	A6

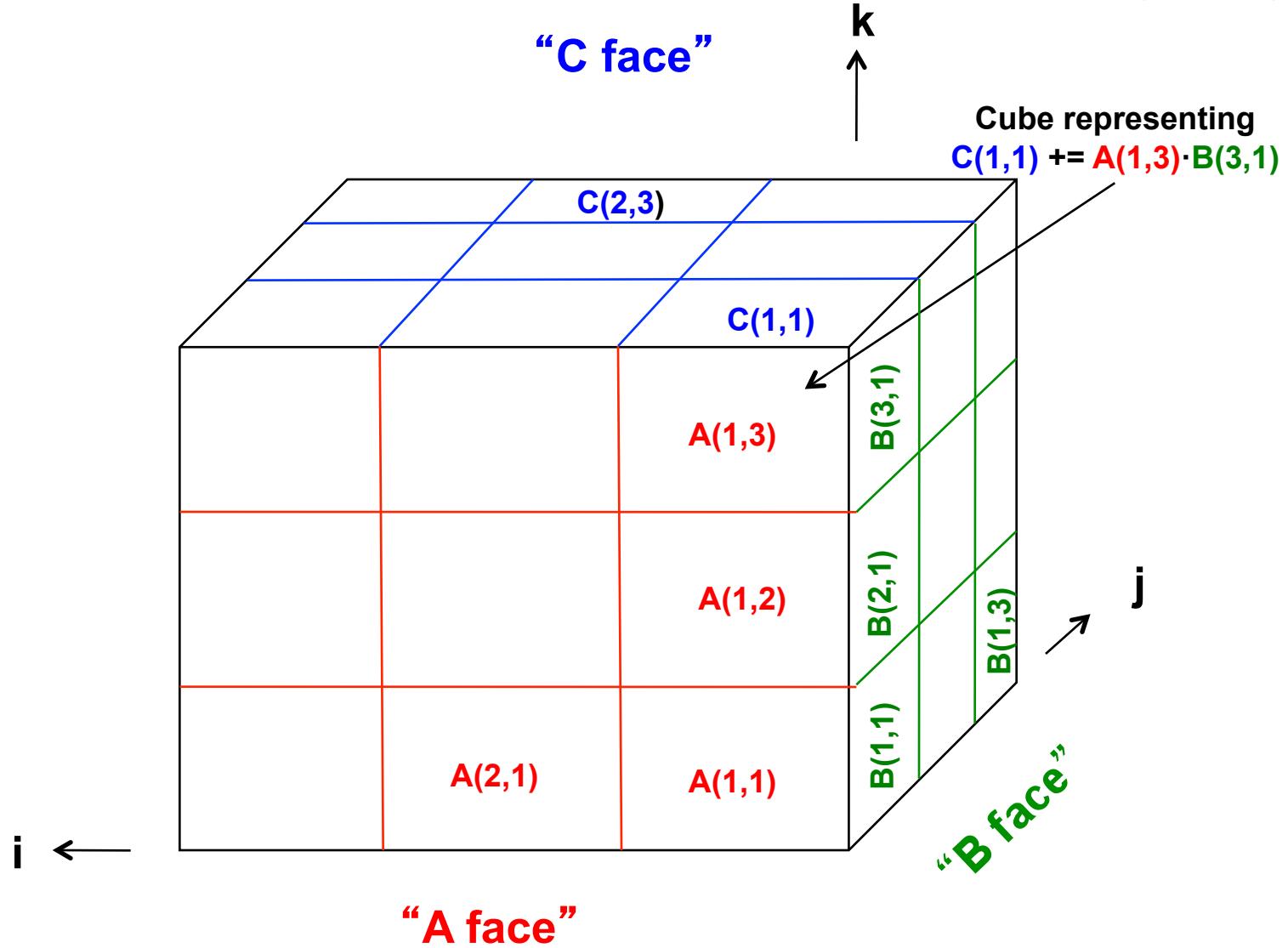
$\Delta =$

- Solve LP for $x = [x_1, \dots, x_7]^T$: $\max \mathbf{1}^T x$ s.t. $\Delta x \leq \mathbf{1}$
 - Result: $x = [2/7, 3/7, 1/7, 2/7, 3/7, 4/7]$, $\mathbf{1}^T x = 15/7 = s_{\text{HBL}}$
- Thm: $\#\text{words_moved} = \Omega(n^6/M^{S_{\text{HBL}}-1}) = \Omega(n^6/M^{8/7})$
Attained by block sizes $M^{2/7}, M^{3/7}, M^{1/7}, M^{2/7}, M^{3/7}, M^{4/7}$

Where do lower and matching upper bounds on communication come from? (1/3)

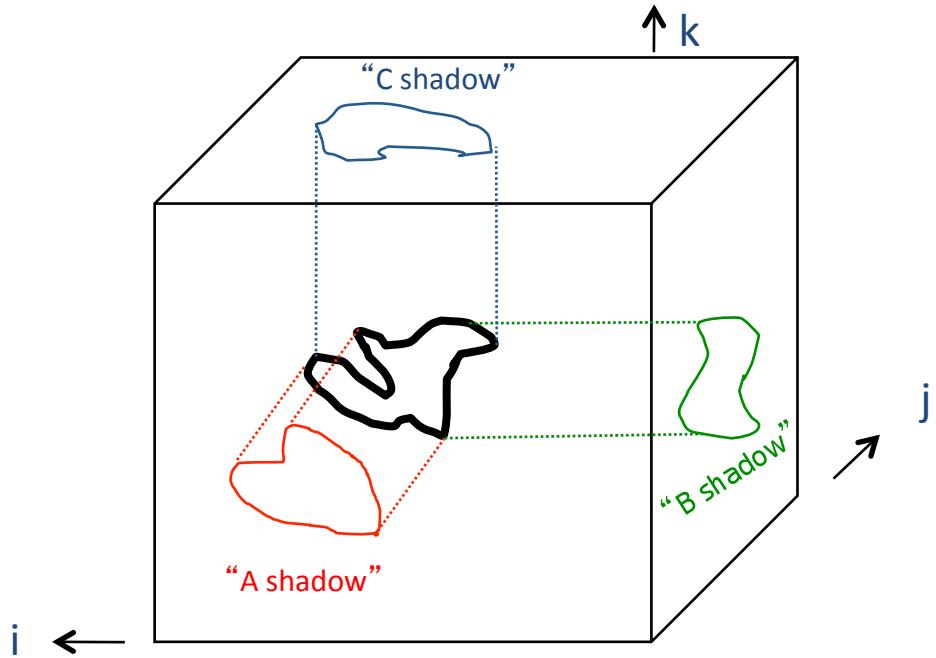
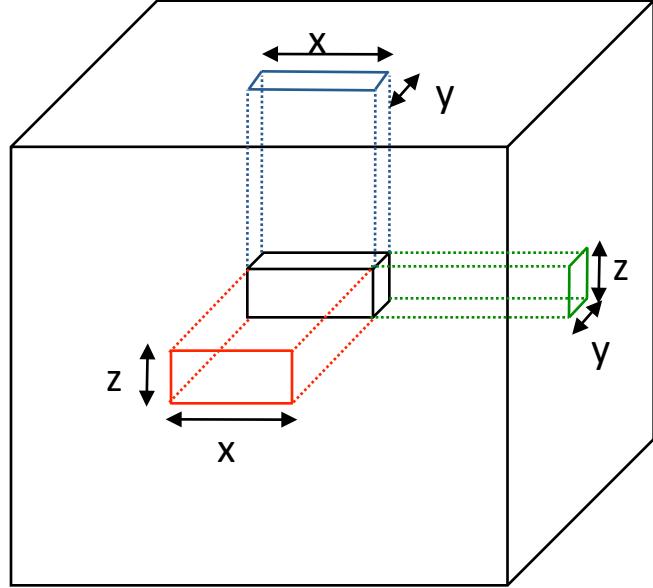
- Originally for $C = A^*B$ by Irony/Tiskin/Toledo (2004)
- Proof idea
 - Suppose we can bound $\#\text{useful_operations} \leq G$ doable with data in fast memory of size M
 - So to do $F = \#\text{total_operations}$, need to fill fast memory F/G times, and so $\#\text{words_moved} \geq MF/G$
- Hard part: finding G
- Attaining lower bound
 - Need to “block” all operations to perform $\sim G$ operations on every chunk of M words of data

Proof of communication lower bound (2/3)



- If we have at most M “A squares”, M “B squares”, and M “C squares”, how many cubes G can we have? 43

Proof of communication lower bound (3/3)



$$\begin{aligned}
 G &= \# \text{ cubes in black box with side lengths } x, y \text{ and } z \\
 &= \text{Volume of black box} \\
 &= x \cdot y \cdot z \\
 &= (xz \cdot zy \cdot yx)^{1/2} \\
 &= (\# \text{A\square}s \cdot \# \text{B\square}s \cdot \# \text{C\square}s)^{1/2} \\
 &\leq M^{3/2}
 \end{aligned}$$

(i, k) is in “A shadow” if (i, j, k) in 3D set
 (j, k) is in “B shadow” if (i, j, k) in 3D set
 (i, j) is in “C shadow” if (i, j, k) in 3D set

Thm (Loomis & Whitney, 1949)

$$\begin{aligned}
 G &= \# \text{ cubes in 3D set} = \text{Volume of 3D set} \\
 &\leq (\text{area(A shadow)} \cdot \text{area(B shadow)} \cdot \text{area(C shadow)})^{1/2} \\
 &\leq M^{3/2}
 \end{aligned}$$

Approach to generalizing lower bounds

- Matmul

for $i=1:n$, for $j=1:n$, for $k=1:n$,

$$C(i,j) += A(i,k)*B(k,j)$$

=> for (i,j,k) in S = subset of Z^3

Access locations indexed by (i,j) , (i,k) , (k,j)

- General case

for $i_1=1:n$, for $i_2 = i_1:m$, ... for $i_k = i_3:i_4$

$$C(i_1+2*i_3-i_7) = \text{func}(A(i_2+3*i_4, i_1, i_2, i_1+i_2, \dots), B(\text{pnt}(3*i_4)), \dots)$$

$$D(\text{something else}) = \text{func}(\text{something else}), \dots$$

=> for (i_1, i_2, \dots, i_k) in S = subset of Z^k

Access locations indexed by group homomorphisms, eg

$$\phi_C(i_1, i_2, \dots, i_k) = (i_1+2*i_3-i_7)$$

$$\phi_A(i_1, i_2, \dots, i_k) = (i_2+3*i_4, i_1, i_2, i_1+i_2, \dots), \dots$$

- Can we bound #loop_iterations ($= |S|$)

given bounds on #points in its images, i.e. bounds on $|\phi_C(S)|$, $|\phi_A(S)|$, ... ?

General Communication Bound

- Given S subset of \mathbb{Z}^k , group homomorphisms ϕ_1, ϕ_2, \dots , bound $|S|$ in terms of $|\phi_1(S)|, |\phi_2(S)|, \dots, |\phi_m(S)|$
- Def: Hölder-Brascamp-Lieb LP (HBL-LP) for s_1, \dots, s_m :
for all subgroups $H < \mathbb{Z}^k$, $\text{rank}(H) \leq \sum_j s_j * \text{rank}(\phi_j(H))$
- Thm (Christ/Tao/Carbery/Bennett): Given s_1, \dots, s_m
$$|S| \leq \prod_j |\phi_j(S)|^{s_j}$$
- Thm: Given a program with array refs given by ϕ_j , choose s_j to minimize $s_{\text{HBL}} = \sum_j s_j$ subject to HBL-LP. Then
$$\#\text{words_moved} = \Omega (\#\text{iterations}/M^{s_{\text{HBL}}-1})$$

Is this bound attainable (1/2)?

- But first: Can we write it down?
- Thm: (bad news) HBL-LP reduces to Hilbert's 10th problem over Q (conjectured to be undecidable)
- Thm: (good news) Another LP with same solution is decidable (but expensive, so far)
- Thm: (better news) Easy to write down LP explicitly in many cases of interest (eg all $\phi_j = \{\text{subset of indices}\}$)
- Thm: (good news) Easy to approximate, i.e. get upper or lower bounds on s_{HBL}

Is this bound attainable (2/2)?

- Depends on loop dependencies
- Best case: none, or reductions (matmul)
- Thm: When all $\phi_j = \{\text{subset of indices}\}$, dual of HBL-LP gives optimal tile sizes:

$$\text{HBL-LP: } \text{minimize } \mathbf{1}^T * \mathbf{s} \text{ s.t. } \mathbf{s}^T * \Delta \geq \mathbf{1}^T$$

$$\text{Dual-HBL-LP: } \text{maximize } \mathbf{1}^T * \mathbf{x} \text{ s.t. } \Delta * \mathbf{x} \leq \mathbf{1}$$

Then for sequential algorithm, tile i_j by M^{x_j}

- Ex: Matmul: $\mathbf{s} = [1/2, 1/2, 1/2]^T = \mathbf{x}$
- Extends to unimodular transforms of indices

Ongoing Work

- Accelerate decision procedure for lower bounds
 - Ex: At most 3 arrays, or 4 loop nests
- Have yet to find a case where we cannot attain lower bound – can we prove this?
- Extend “perfect scaling” results for time and energy by using extra memory
- Incorporate into compilers (DEGAS)

Outline

- Survey state of the art of CA (Comm-Avoiding) algorithms
 - TSQR: Tall-Skinny QR
 - CA $O(n^3)$ 2.5D Matmul
 - CA Strassen Matmul
- Beyond linear algebra
 - Extending lower bounds to any algorithm with arrays
 - Communication-optimal N-body algorithm
- CA-Krylov methods

Avoiding Communication in Iterative Linear Algebra

- k-steps of iterative solver for sparse $Ax=b$ or $Ax=\lambda x$
 - Does k SpMVs with A and starting vector
 - Many such “Krylov Subspace Methods”
 - Conjugate Gradients (CG), GMRES, Lanczos, Arnoldi, ...
- Goal: minimize communication
 - Assume matrix “well-partitioned”
 - Serial implementation
 - Conventional: $O(k)$ moves of data from slow to fast memory
 - **New: $O(1)$ moves of data – optimal**
 - Parallel implementation on p processors
 - Conventional: $O(k \log p)$ messages (k SpMV calls, dot prods)
 - **New: $O(\log p)$ messages - optimal**
- Lots of speed up possible (modeled and measured)
 - Price: some redundant computation
 - Challenges: Poor partitioning, Preconditioning, Num. Stability

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$

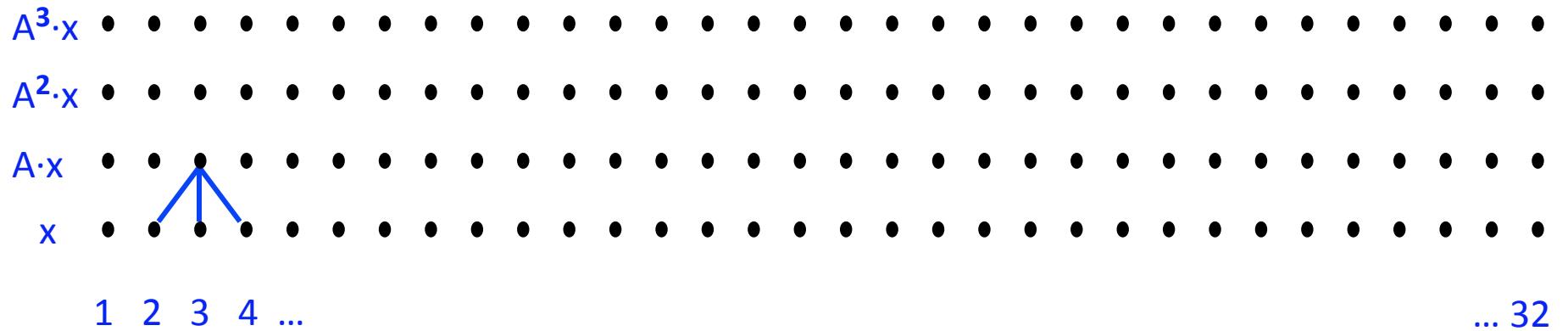
$A^3 \cdot x$ •
 $A^2 \cdot x$ •
 $A \cdot x$ •
 x •
1 2 3 4 32

- Example: A tridiagonal, $n=32$, $k=3$
- Works for any “well-partitioned” A

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$

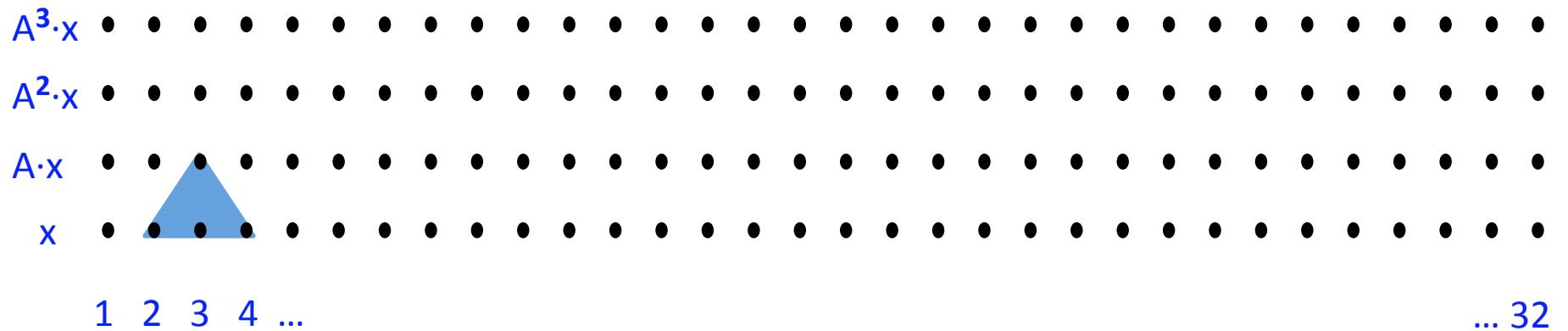


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$

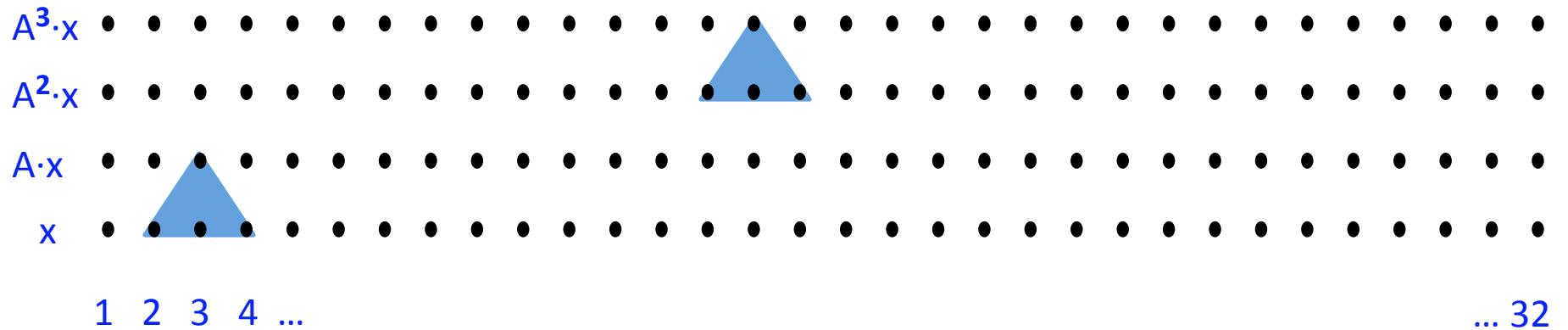


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$

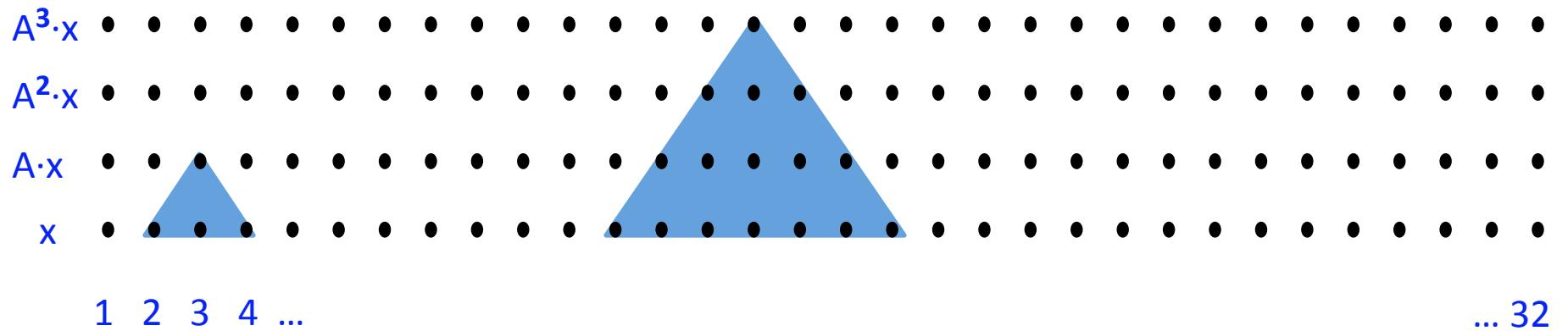


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$

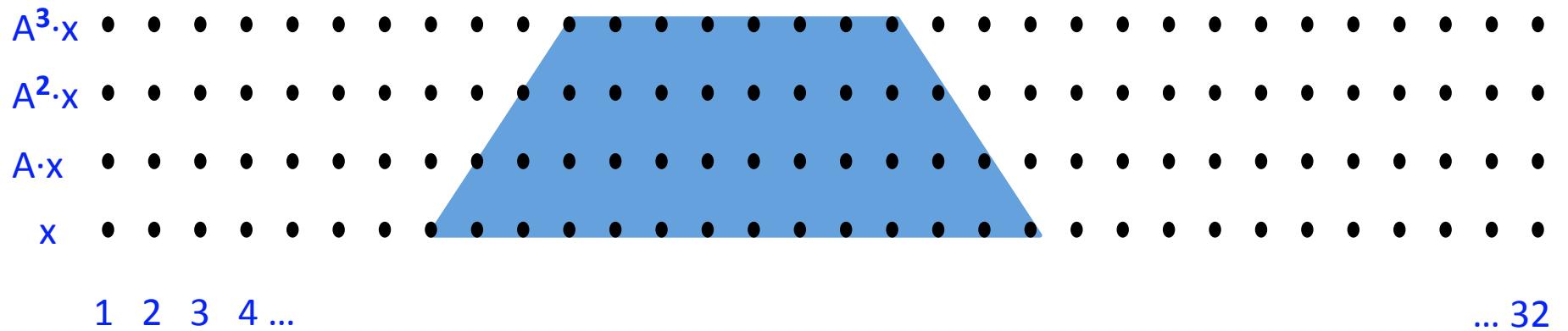


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$

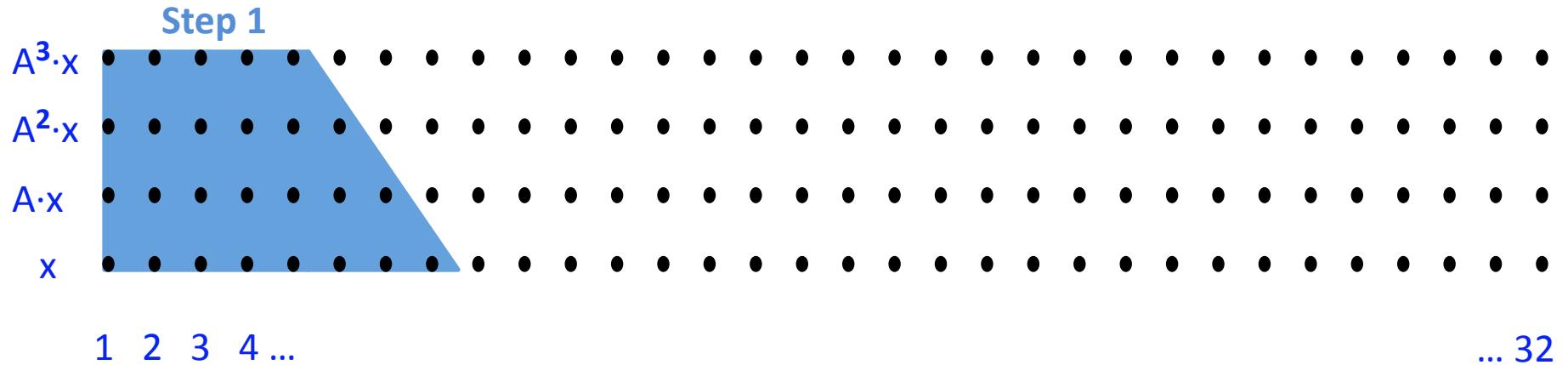


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$
- Sequential Algorithm

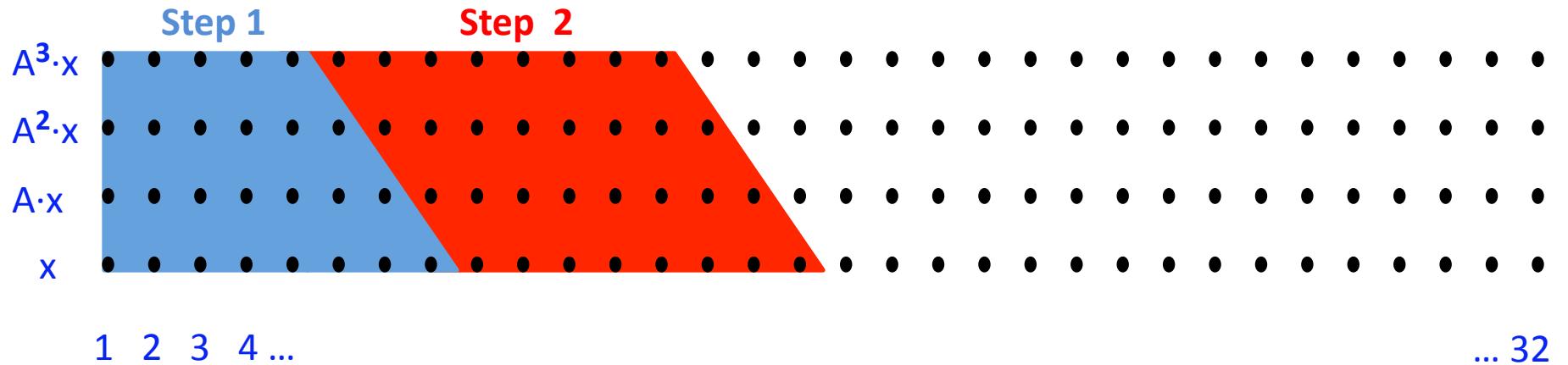


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$
- Sequential Algorithm

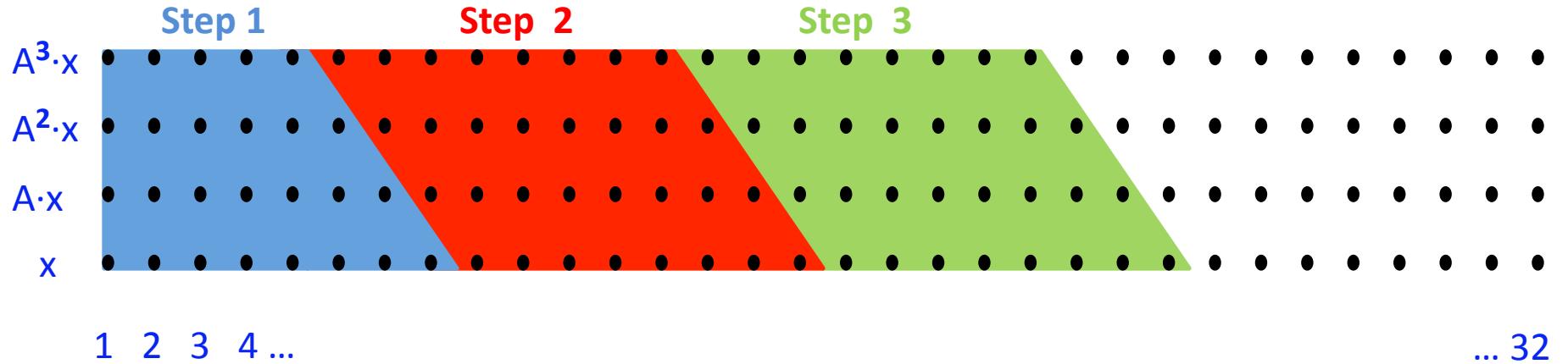


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$
- Sequential Algorithm

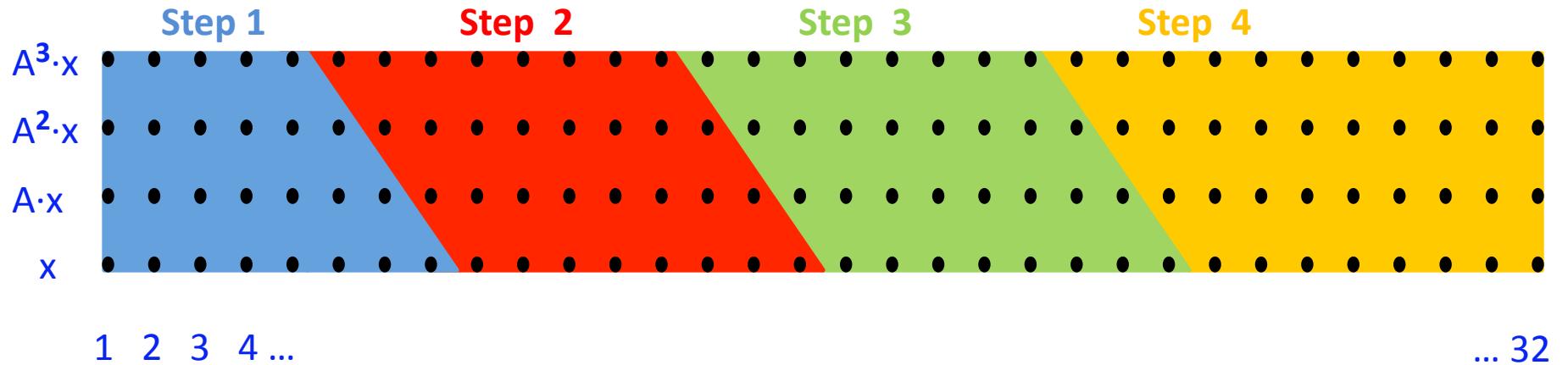


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$
- Sequential Algorithm

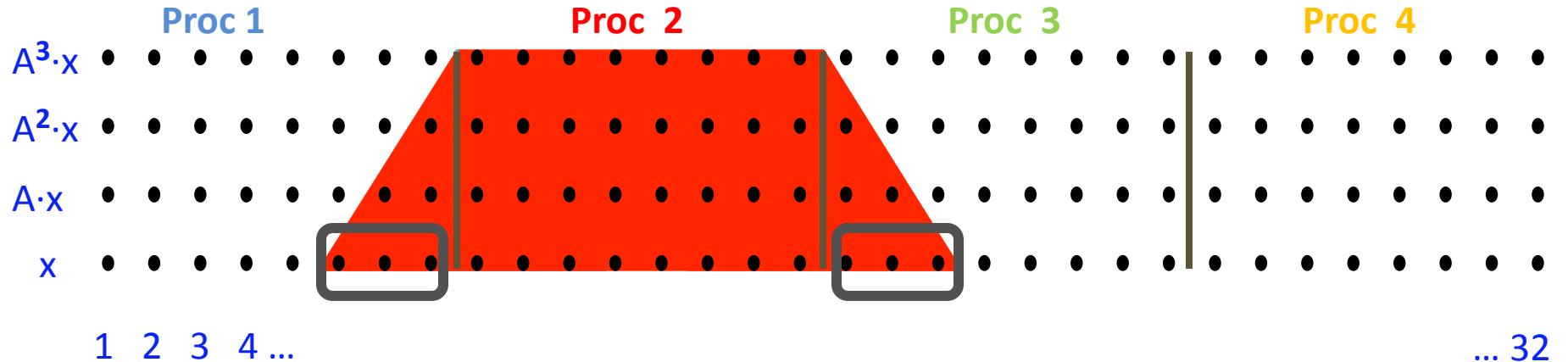


- Example: A tridiagonal, $n=32$, $k=3$

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$
- Parallel Algorithm

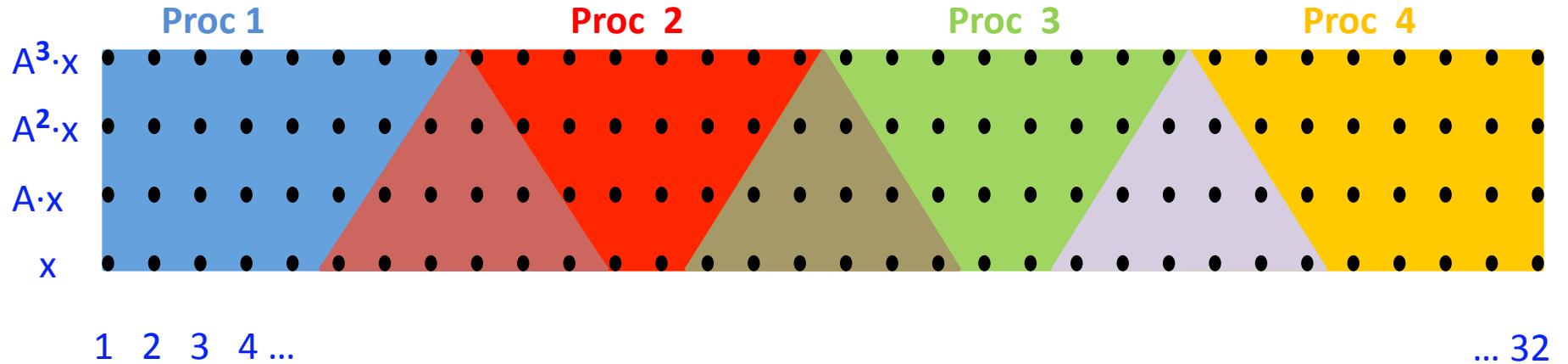


- Example: A tridiagonal, $n=32$, $k=3$
- Each processor communicates once with neighbors

Communication Avoiding Kernels:

The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \dots, A^kx]$
- Parallel Algorithm



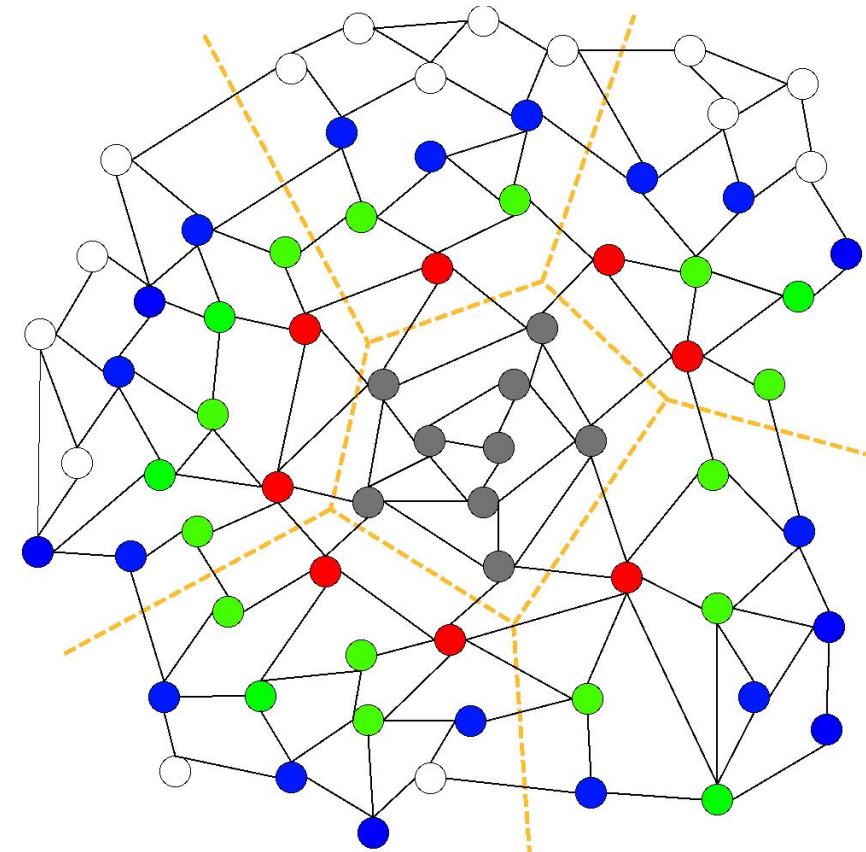
- Example: A tridiagonal, $n=32$, $k=3$
- Each processor works on (overlapping) trapezoid

Communication Avoiding Kernels: The Matrix Powers Kernel : $[Ax, A^2x, \dots, A^kx]$

Same idea works for general sparse matrices

Simple block-row partitioning →
(hyper)graph partitioning

Top-to-bottom processing →
Traveling Salesman Problem



Minimizing Communication of GMRES to solve $Ax=b$

- GMRES: find x in $\text{span}\{b, Ab, \dots, A^k b\}$ minimizing $\| Ax - b \|_2$

Standard GMRES

for $i=1$ to k

$w = A \cdot v(i-1) \dots SpMV$

$MGS(w, v(0), \dots, v(i-1))$

update $v(i)$, H

endfor

solve LSQ problem with H

Communication-avoiding GMRES

$W = [v, Av, A^2v, \dots, A^k v]$

$[Q, R] = TSQR(W)$

... “*Tall Skinny QR*”

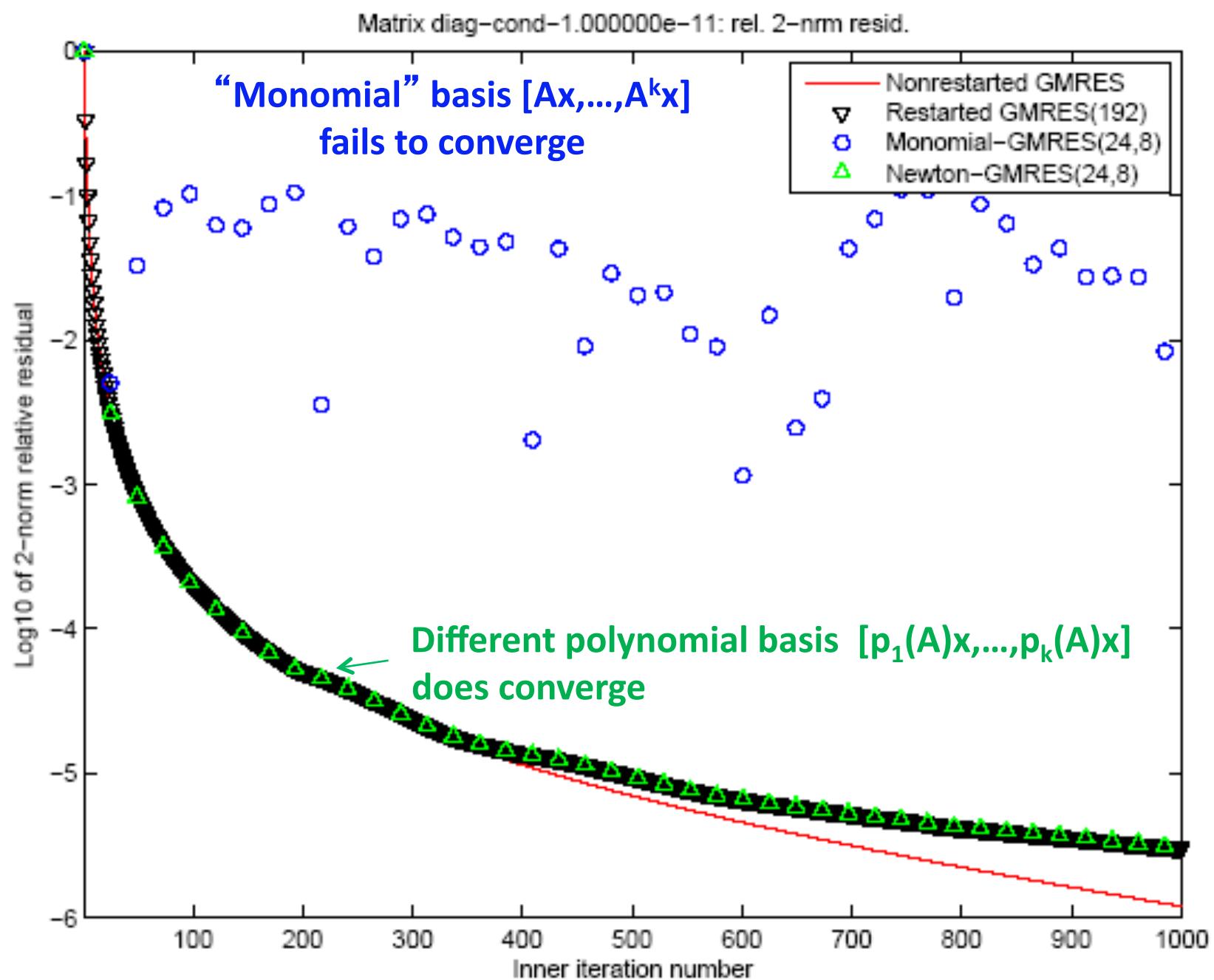
build H from R

solve LSQ problem with H

Sequential case: #words moved decreases by a factor of k

Parallel case: #messages decreases by a factor of k

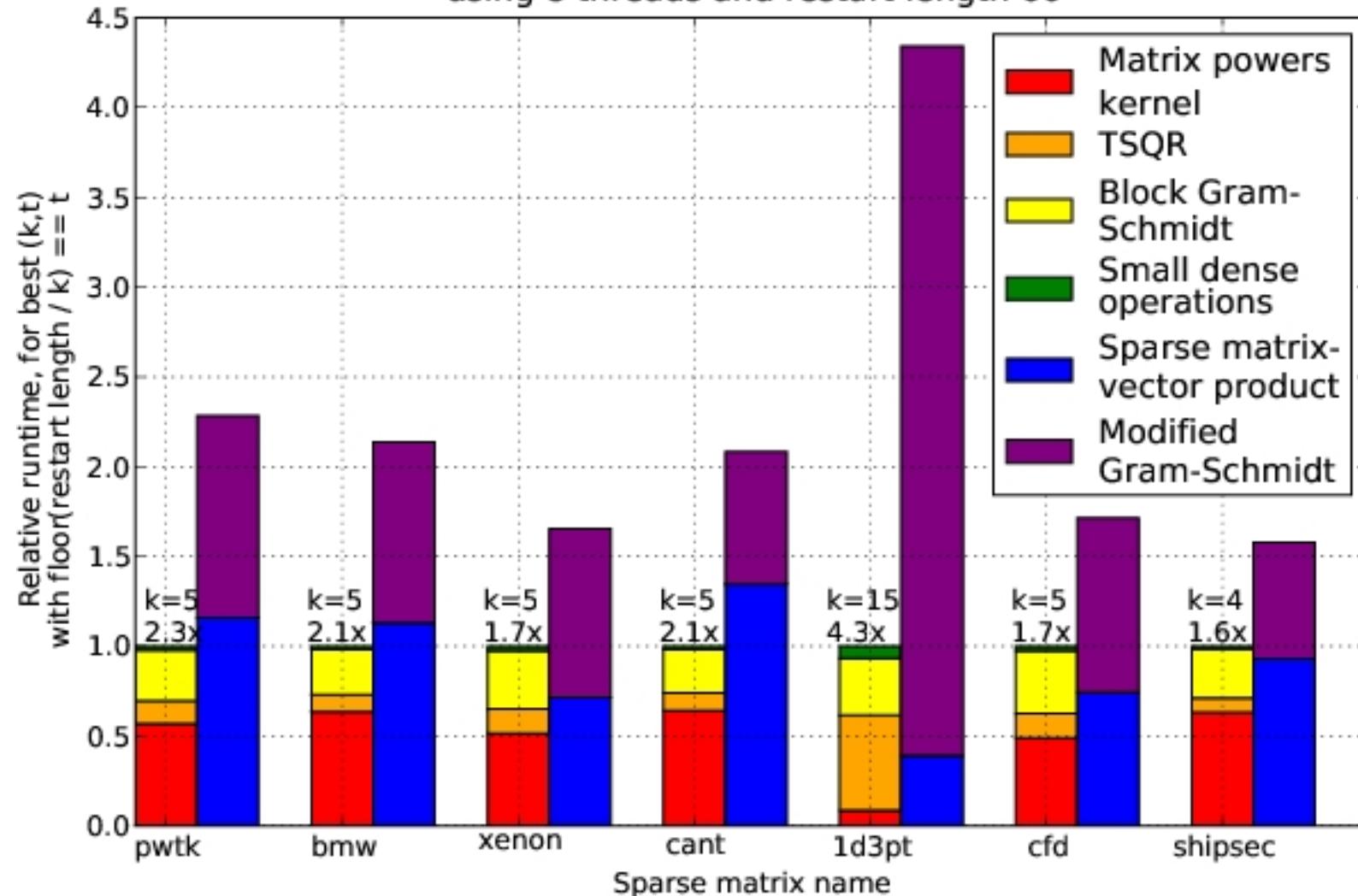
- Oops – W from power method, precision lost!



Speed ups of GMRES on 8-core Intel Clovertown

Requires Co-tuning Kernels
[MHDY09]

Runtime per kernel, relative to CA-GMRES(k,t), for all test matrices,
using 8 threads and restart length 60



Compute $r_0 = b - Ax_0$. Choose r_0^* arbitrary.

Set $p_0 = r_0$, $q_{-1} = 0_{N \times 1}$.

For $k = 0, 1, \dots$, until convergence, Do

$$\begin{aligned} P &= [p_{sk}, Ap_{sk}, \dots, A^s p_{sk}] \\ Q &= [q_{sk-1}, Aq_{sk-1}, \dots, A^s q_{sk-1}] \\ R &= [r_{sk}, Ar_{sk}, \dots, A^s r_{sk}] \end{aligned}$$

//Compute the $1 \times (3s + 3)$ Gram vector.

$$g = (r_0^*)^T [P, Q, R]$$

//Compute the $(3s + 3) \times (3s + 3)$ Gram matrix

$$G = \begin{bmatrix} P^T \\ Q^T \\ R^T \end{bmatrix} \begin{bmatrix} P & Q & R \end{bmatrix}$$

For $\ell = 0$ to s ,

$$b_{sk}^\ell = [B_1(:, \ell)^T, 0_{s+1}^T, 0_{s+1}^T]^T$$

$$c_{sk-1}^\ell = [0_{s+1}^T, B_2(:, \ell)^T, 0_{s+1}^T]^T$$

$$d_{sk}^\ell = [0_{s+1}^T, 0_{s+1}^T, B_3(:, \ell)^T]^T$$

1. Compute $r_0 := b - Ax_0$; r_0^* arbitrary;
2. $p_0 := r_0$.
3. For $j = 0, 1, \dots$, until convergence Do:
4. $\alpha_j := (r_j, r_0^*) / (Ap_j, r_0^*)$
5. $s_j := r_j - \alpha_j Ap_j$
6. $\omega_j := (As_j, s_j) / (As_j, As_j)$
7. $x_{j+1} := x_j + \alpha_j p_j + \omega_j s_j$
8. $r_{j+1} := s_j - \omega_j As_j$
9. $\beta_j := \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \leftarrow \frac{\alpha_j}{\omega_j}$
10. $p_{j+1} := r_{j+1} + \beta_j(p_j - \omega_j Ap_j)$
11. EndDo

CA-BiCGStab

For $j = 0$ to $\lfloor \frac{s}{2} \rfloor - 1$, Do

$$\alpha_{sk+j} = \frac{\langle g, d_{sk+j}^0 \rangle}{\langle g, b_{sk+j}^1 \rangle}$$

$$q_{sk+j} = r_{sk+j} - \alpha_{sk+j}[P, Q, R]b_{sk+j}^1$$

For $\ell = 0$ to $s - 2j + 1$, Do

$$c_{sk+j}^\ell = d_{sk+j}^\ell - \alpha_{sk+j} b_{sk+j-1}^{\ell+1}$$

//such that $[P, Q, R] c_{sk+j}^\ell = A^\ell q_{sk+j}$

$$\omega_{sk+j} = \frac{\langle c_{sk+j+1}^1, Gc_{sk+j+1}^0 \rangle}{\langle c_{sk+j+1}^1, Gc_{sk+j+1}^1 \rangle}$$

$$x_{sk+j+1} = x_{sk+j} + \alpha_{sk+j} p_{sk+j} + \omega_{sk+j} q_{sk+j}$$

$$r_{sk+j+1} = q_{sk+j} - \omega_{sk+j}[P, Q, R]c_{sk+j+1}^1$$

For $\ell = 0$ to $s - 2j$, Do

$$d_{sk+j+1}^\ell = c_{sk+j+1}^\ell - \omega_{sk+j} c_{sk+j+1}^{\ell+1}$$

//such that $[P, Q, R] d_{sk+j+1}^\ell = A^\ell r_{sk+j+1}$

$$\beta_{sk+j} = \frac{\langle g, d_{sk+j+1}^0 \rangle}{\langle g, d_{sk+j}^0 \rangle} \times \frac{\alpha}{\omega}$$

$$p_{sk+j+1} = r_{sk+j+1} + \beta_{sk+j} p_{sk+j} - \beta_{sk+j} \omega_{sk+j}[P, Q, R]b_{sk+j}^1$$

For $\ell = 0$ to $s - 2j$, Do

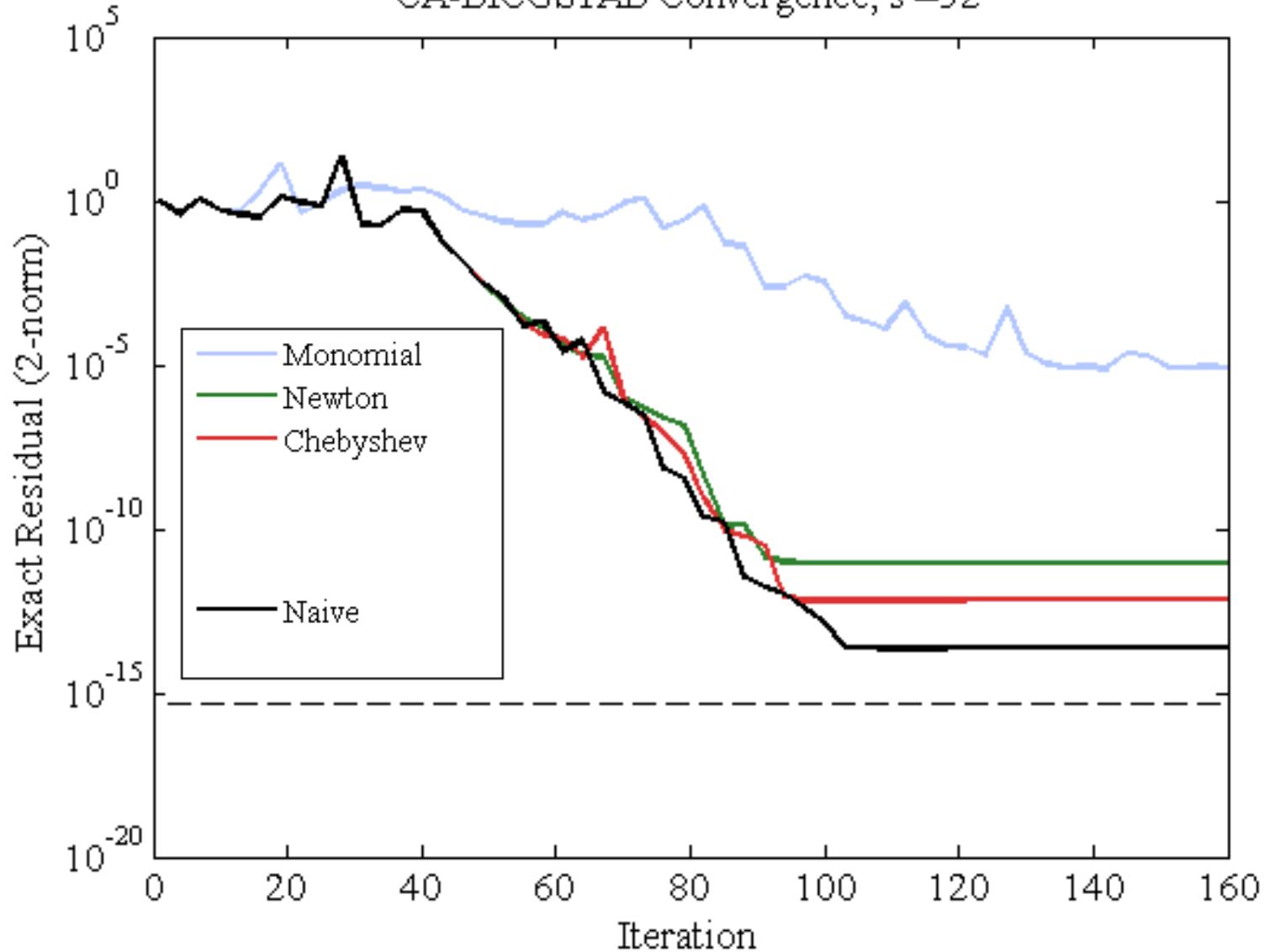
$$b_{sk+j+1}^\ell = d_{sk+j+1}^\ell + \beta_{sk+j} b_{sk+j}^\ell - \beta_{sk+j} \omega_{sk+j} b_{sk+j}^{\ell+1}$$

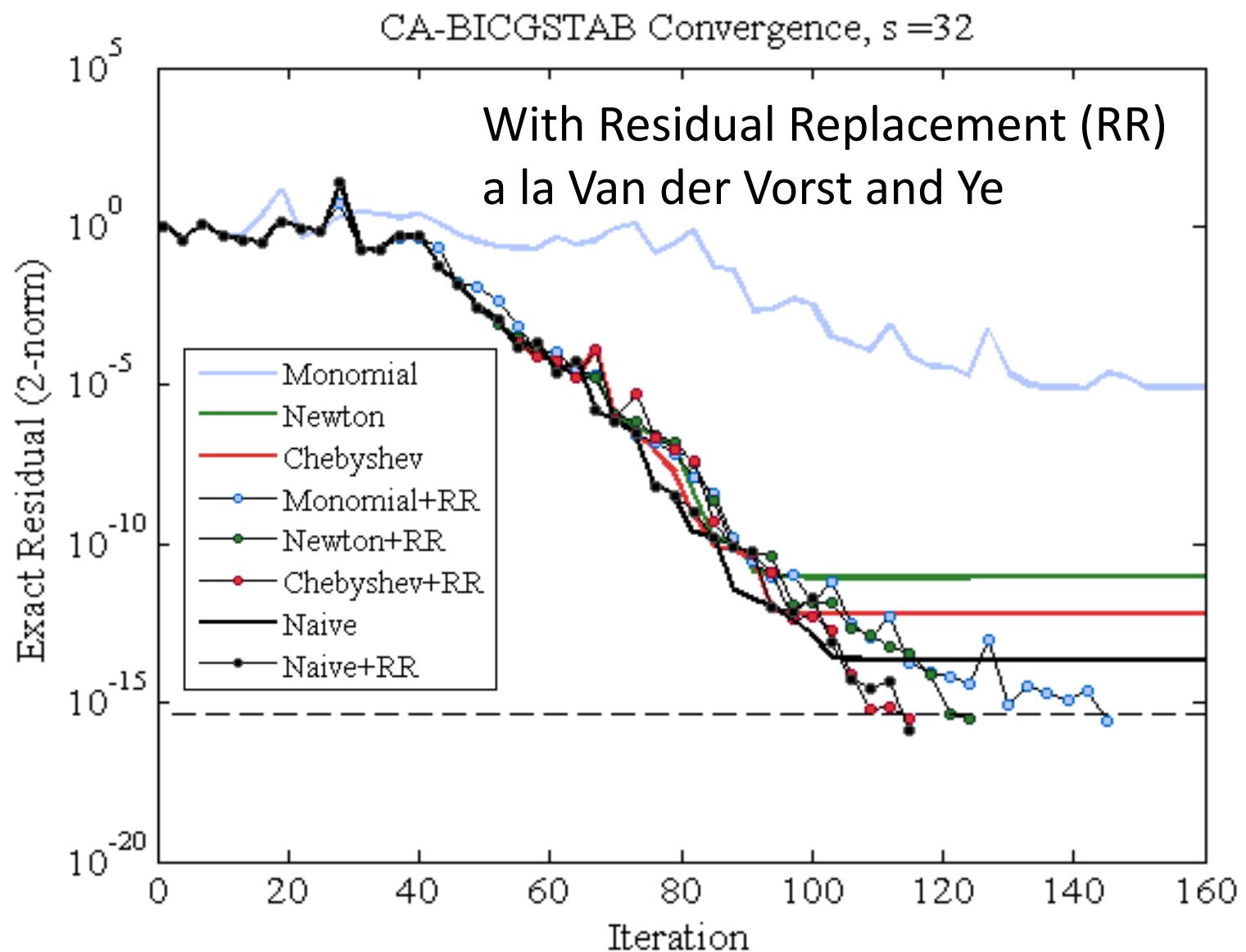
//such that $[P, Q, R] b_{sk+j+1}^\ell = A^\ell p_{sk+j+1}$.

EndDo

EndDo

CA-BICGSTAB Convergence, s = 32





	Naive	Monomial	Newton	Chebyshev
Replacement Its.	74 (1)	[7, 15, 24, 31, ..., 92, 97, 103] (17)	[67, 98] (2)	68 (1)

Summary of Iterative Linear Algebra

- New lower bounds, optimal algorithms, big speedups in theory and practice
- Lots of other progress, open problems
 - Many different algorithms reorganized
 - More underway, more to be done
 - Need to recognize stable variants more easily
 - Preconditioning
 - Hierarchically Semiseparable Matrices
 - Autotuning and synthesis
 - Different kinds of “sparse matrices”

For more details

- Bebop.cs.berkeley.edu
- CS267 – Berkeley’s Parallel Computing Course
 - Live broadcast in Spring 2013
 - www.cs.berkeley.edu/~demmel
 - All slides, video available
 - Prerecorded version broadcast in Spring 2013
 - www.xsede.org
 - Free supercomputer accounts to do homework
 - Free autograding of homework

Related IPDPS'13 Talks, Tuesday

- Communication-Optimal Parallel Recursive Rectangular Matrix Multiplication
 - JD, D. Eliahu, A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, O. Spillinger
 - 1:30-3:30pm, Session 6
- Minimizing communication in all-pairs shortest paths
 - E. Solomonik, A. Buluc, JD
 - 4:00-6:00pm, Session 12

Related IPDPS'13 Talks, Wednesday

- Perfect Strong Scaling Using No Additional Energy
 - JD, A. Gearhart, B. Lipshitz, O. Schwartz
 - 10:00am – 12:00pm, Session 14
- Cyclops Tensor Framework: Reducing communication and eliminating load imbalance in massively parallel contractions
 - E. Solomonik, D. Matthews, J. Hammond, JD
 - 1:30 – 3:30pm, Session 17

Related IPDPS'13 Talks, Thursday

- Implementing a blocked Aasen's algorithm with a dynamic scheduler on multicore architectures
 - I. Yamazaki, G. Ballard, D. Becker, JD, J. Dongarra, A. Druinsky, I. Peled, O. Schwartz, S. Toledo
 - 10:00am – 12:00pm, **Plenary Session: Best Papers**
- A Communication-Optimal N-body Algorithm for Direct Interactions
 - M. Driscoll, E. Georganas, P. Koanantokool, E. Solomonik, K. Yelick
 - 1:30pm – 3:30pm, Session 21

Collaborators and Supporters

- **James Demmel, Kathy Yelick**, Michael Anderson, Grey Ballard, Erin Carson, Aditya Devarakonda, Michael Driscoll, David Eliahu, Andrew Gearhart, Evangelos Georganas, Nicholas Knight, Penporn Koanantakool, Ben Lipshitz, Oded Schwartz, Edgar Solomonik, Omer Spillinger
- Austin Benson, Maryam Dehnavi, Mark Hoemmen, Shoaib Kamil, Marghoob Mohiyuddin
- Abhinav Bhatale, Aydin Buluc, Michael Christ, Ioana Dumitriu, Armando Fox, David Gleich, Ming Gu, Jeff Hammond, Mike Heroux, Olga Holtz, Kurt Keutzer, Julien Langou, Devin Matthews, Tom Scanlon, Michelle Strout, Sam Williams, Hua Xiang
- Jack Dongarra, Dulceneia Becker, Ichitaro Yamazaki
- Sivan Toledo, Alex Druinsky, Inon Peled
- Laura Grigori, Sebastien Cayrols, Simplice Donfack, Mathias Jacquelin, Amal Khabou, Sophie Moufawad, Mikolaj Szydlarski
- Members of DEGAS, ParLab, ASPIRE, BEBOP, CACHE, EASI, FASTMath, MAGMA, PLASMA
- Thanks to DOE, NSF, UC Discovery, INRIA, Intel, Microsoft, Mathworks, National Instruments, NEC, Nokia, NVIDIA, Samsung, Oracle
- bebop.cs.berkeley.edu

Summary

Time to redesign all linear algebra, n-body, ...
algorithms and software
(and compilers)

Don't Communic...

Collaborators and Supporters

- Michael Christ, Jack Dongarra, Ioana Dumitriu, David Gleich, Laura Grigori, Ming Gu, Olga Holtz, Julien Langou, Tom Scanlon, Kathy Yelick
- Grey Ballard, Austin Benson, Abhinav Bhatte, Aydin Buluc, Erin Carson, Maryam Dehnavi, Michael Driscoll, Evangelos Georganas, Nicholas Knight, Penporn Koanantakool, Ben Lipshitz, Oded Schwartz, Edgar Solomonik, Hua Xiang
- Other members of ParLab, BEBOP, CACHE, EASI, FASTMath, MAGMA, PLASMA, TOPS projects
 - bebop.cs.berkeley.edu
- Thanks to NSF, DOE, UC Discovery, Intel, Microsoft, Mathworks, National Instruments, NEC, Nokia, NVIDIA, Samsung, Oracle

Collaborators and Supporters

- Michael Christ, Jack Dongarra, Ioana Dumitriu, Armando Fox, David Gleich, Laura Grigori, Ming Gu, Mike Heroux, Mark Hoemmen, Olga Holtz, Kurt Keutzer, Julien Langou, Tom Scanlon, Michelle Strout, Sam Williams, Hua Xiang, Kathy Yelick
- Michael Anderson, Grey Ballard, Austin Benson, Abhinav Bhatte, Aydin Buluc, Erin Carson, Maryam Dehnavi, Michael Driscoll, Evangelos Georganas, Nicholas Knight, Penporn Koanantakool, Ben Lipshitz, Marghoob Mohiyuddin, Oded Schwartz, Edgar Solomonik
- Other members of ParLab, BEBOP, CACHE, EASI, FASTMath, MAGMA, PLASMA, TOPS projects
- Thanks to NSF, DOE, UC Discovery, Intel, Microsoft, Mathworks, National Instruments, NEC, Nokia, NVIDIA, Samsung, Oracle
- bebop.cs.berkeley.edu