1) Communication-Avoiding Algorithms for Linear Algebra, ML and Beyond
2) Grading the accuracy of the BLAS: What can emulating FP64 guarantee?

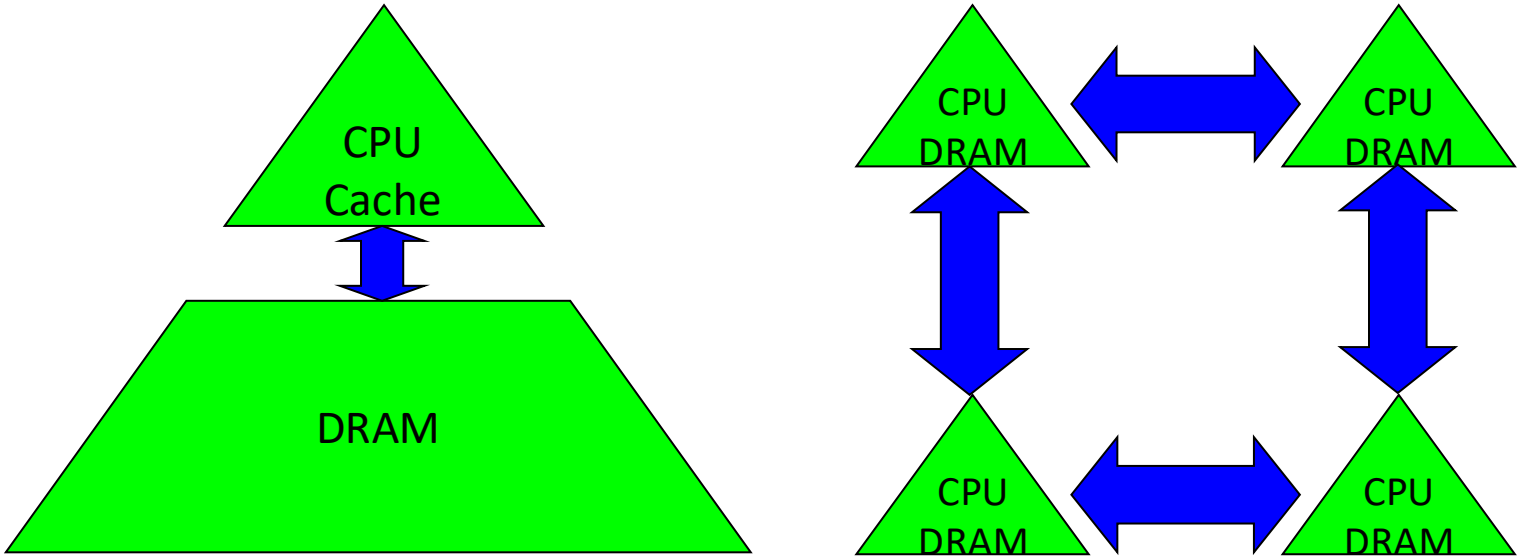
Jim Demmel, EECS & Math Depts., UC Berkeley

And many, many others …

Why avoid communication? (1/2)

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).

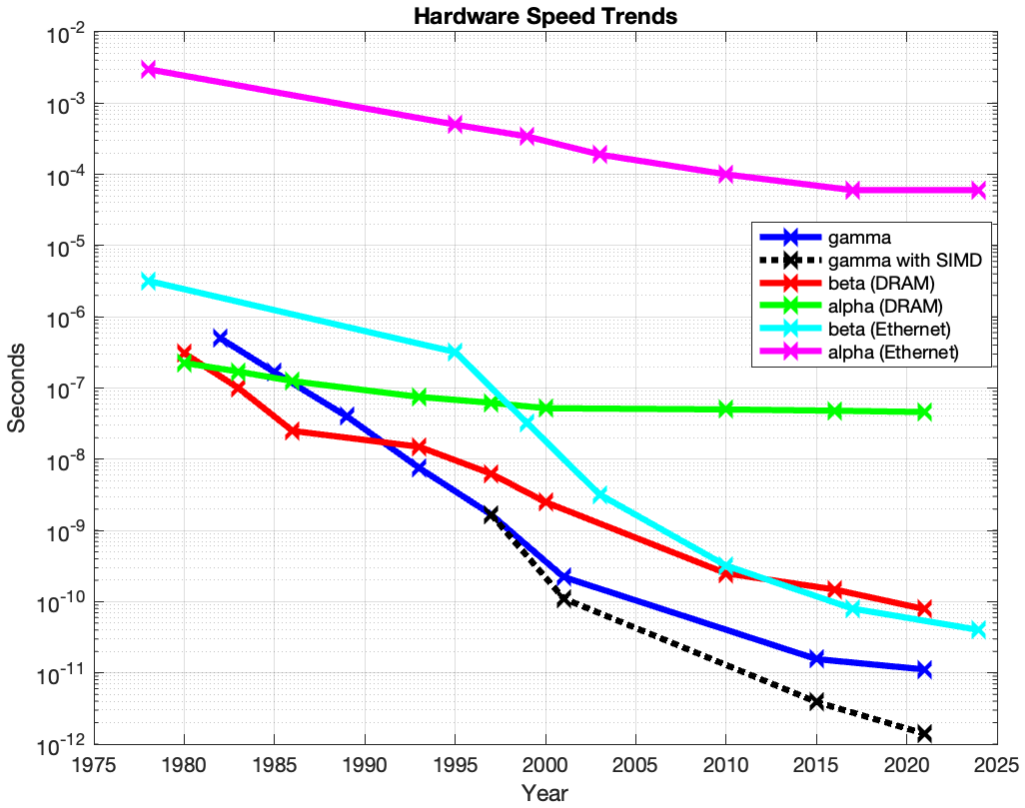


Why avoid communication? (2/2)

- Running time of an algorithm is sum of 3 terms:
  - # flops * time_per_flop
  - # words moved / bandwidth
  - # messages * latency

  } communication

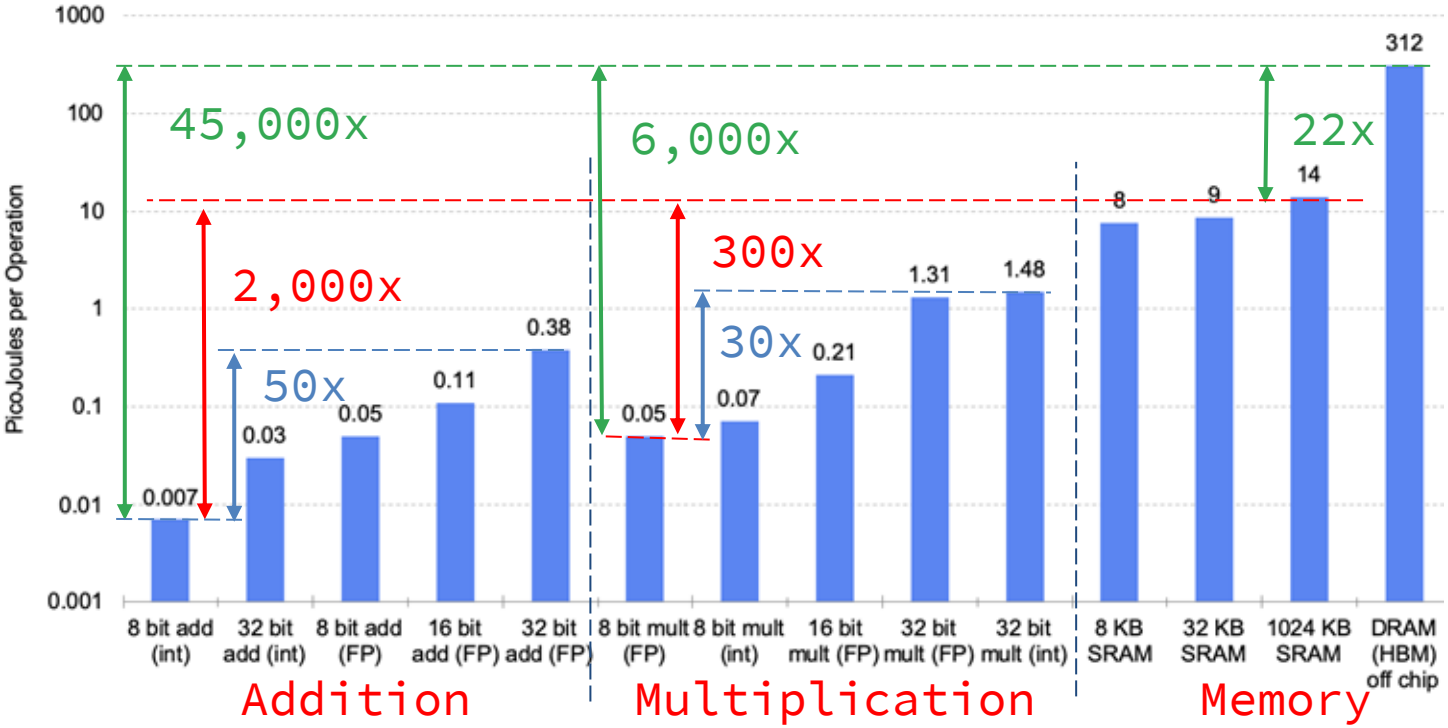
- Time_per_flop (γ) << 1/ bandwidth (β) << latency (α)



Data from Patterson & Hennessey, 2025

Chips limited by Power, Not Transistors (7 nm)

Slide from Dave Patterson



Smaller data =>
Less energy,
Less memory,
Less bandwidth

Goals

- Redesign algorithms to *avoid* communication
  - Between all memory hierarchy levels
    - L1 $\longleftrightarrow$ L2 $\longleftrightarrow$ DRAM $\longleftrightarrow$ network, etc
- Attain lower bounds if possible
  - Classical algorithms often far from lower bounds
  - Large speedups and energy savings possible
- Automate implementation of communication-avoiding (CA) algorithms

Sample Speedups

- Doing same operations, just in a different order
  - Up to **12x** faster for 2.5D dense matmul on 64K core IBM BG/P
  - Up to **100x** faster for 1.5D sparse-dense matmul on 1536 core Cray XC30
  - Up to **6.2x** faster for 2.5D All-Pairs-Shortest-Path on 24K core Cray XE6
  - Up to **11.8x** faster for direct N-body on 32K core IBM BG/P
- Mathematically identical answer, but different algorithm
  - Up to **13x** faster for Tall Skinny QR on Tesla C2050 Fermi NVIDIA GPU
  - Up to **6.7x** faster for symeig(band A) on 10 core Intel Westmere
  - Up to **4.2x** faster for BiCGStab (MiniGMG bottom solver) on 24K core Cray XE6
  - Up to **5.1x** faster for coordinate descent LASSO on 3K core Cray XC30
- Different algorithm, different approximate answer
  - Up to **16x** faster for SVM on a 1536 core Cray XC30
  - Up to **135x** faster for ImageNet training on 2K Intel KNL nodes

Sample Speedups

- Doing same operations, just in a different order

Ideas adopted by Nervana, "deep learning" startup, acquired by Intel in August 2016

Kwasniewski, Hoefler, et al (Best Student Paper, SC'19)

- Mathematically identical answer, but different algorithm

SIAG on Supercomputing Best Paper Prize, 2016

(D., Grigori, Hoemmen, Langou)

Released in LAPACK 3.7, 2016

LAPACK 3.10: Householder Reconstruction, 2021

- Different algorithm, different approximate answer

IPDPS 2015 Best Paper Prize **(You, D. Czechowski, Song, Vuduc)**

ICPP 2018 Best Paper Prize **(You, Zhang, Hsieh, D., Keutzer)**

2019: Idea (LARS) adopted by industry standard benchmark MLPerf

Outline (of avoiding communication)

- Linear Algebra
  - Communication Lower Bounds for classical direct linear algebra
  - CA 2.5D Matmul
  - TSQR - Tall-Skinny QR
  - Iterative Methods for linear algebra
- Machine Learning
  - Training Neural Nets – “ImageNet training in minutes”
- And Beyond
  - Extending communication lower bounds and optimal algorithms to general loop nests

Outline (of avoiding communication)

- **Linear Algebra**
  - **Communication Lower Bounds for classical direct linear algebra**
  - CA 2.5D Matmul
  - TSQR - Tall-Skinny QR
  - Iterative Methods for linear algebra
- Machine Learning
  - Training Neural Nets – “ImageNet training in minutes”
- And Beyond
  - Extending communication lower bounds and optimal algorithms to general loop nests

Summary of CA Linear Algebra

- "Direct" Linear Algebra
  - Lower bounds on communication for linear algebra problems like Ax=b, least squares, Ax = λx, SVD, etc
  - Only some attained by algorithms in standard libraries
    - LAPACK, ScaLAPACK, SLATE, …
  - New algorithms needed to attain these lower bounds
    - New numerical properties, ways to encode answers, data structures, not just loop transformations
  - Autotuning to find optimal implementation (eg GPTune)
  - Sparse matrices: depends on sparsity structure
- Ditto for "Iterative" Linear Algebra

Lower bound for all "n^3-like" linear algebra

- Let M = "fast" memory size (per processor)

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

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

Lower bound for all "n^3-like" linear algebra

- Let M = "fast" memory size (per processor)

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

#messages_sent ≥ #words_moved / 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 << n^3)
  - Sequential and parallel algorithms
  - Some graph-theoretic algorithms (eg Floyd-Warshall)

Lower bound for all "n^3-like" linear algebra

- Let M = "fast" memory size (per processor)

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

#messages_sent (per processor) = Ω(#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

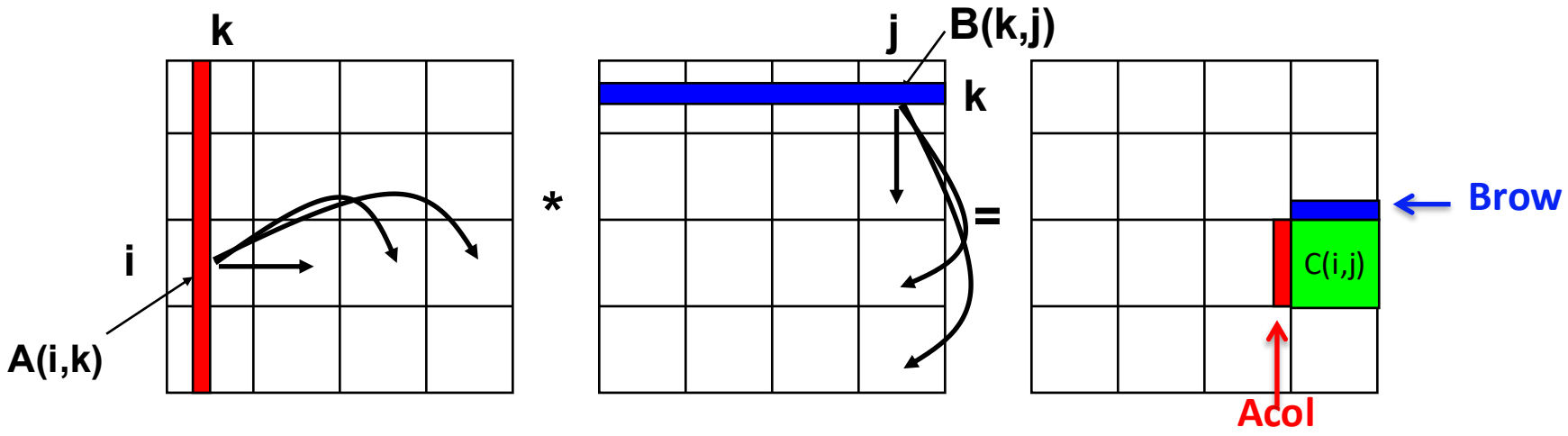
(Ballard, D., Holtz, Schwartz)

Outline (of avoiding communication)

- Linear Algebra
  - Communication Lower Bounds for classical direct linear algebra
  - **CA 2.5D Matmul**
  - TSQR - Tall-Skinny QR
  - Iterative Methods for linear algebra
- Machine Learning
  - Training Neural Nets – “ImageNet training in minutes”
- And Beyond
  - Extending communication lower bounds and optimal algorithms to general loop nests

SUMMA– n x n matmul on $P^{1/2}$ x $P^{1/2}$ grid

(nearly) optimal using minimum memory $M=O(n^2/P)$



For k=0 to n/b-1 … b = block size = #cols in A(i,k) = #rows in B(k,j)

for all i = 1 to $P^{1/2}$

owner of A(i,k) broadcasts it to whole processor row (using binary tree)

for all j = 1 to $P^{1/2}$

owner of B(k,j) broadcasts it to whole processor column (using bin. tree)

Receive A(i,k) into Acol

Receive B(k,j) into Brow

C_myproc = C_myproc + Acol * Brow

Summary of dense *parallel* algorithms attaining communication lower bounds

- Assume nxn 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 = $\Omega((n^3/ P) / M^{3/2}) = \Omega(P^{1/2})$
- SUMMA attains this lower bound
- 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

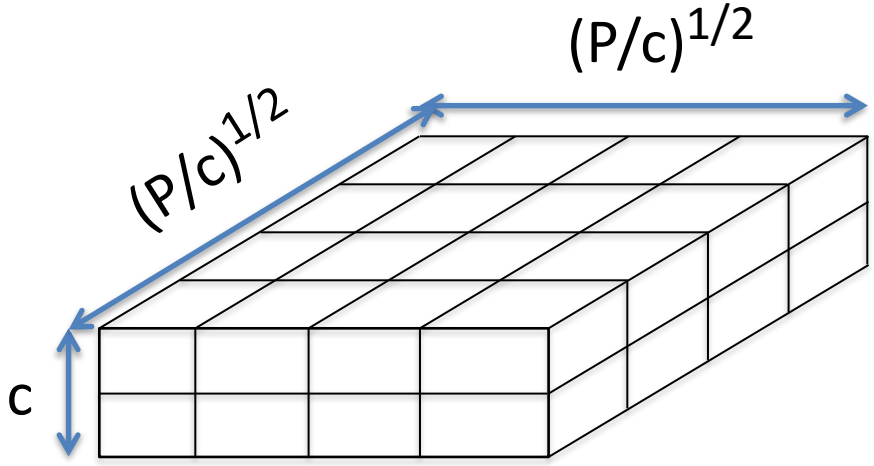
Can we do Better?

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?
- Special case: "3D Matmul"
  - Uses $M = O(n^2/p^{2/3})$
  - Dekel, Nassimi, Sahni [81], Bernsten [89], Agarwal, Chandra, Snir [90], Johnson [93], Agarwal, Balle, Gustavson, Joshi, Palkar [95]
- Not always $p^{1/3}$ times as much memory available…

2.5D Matrix Multiplication

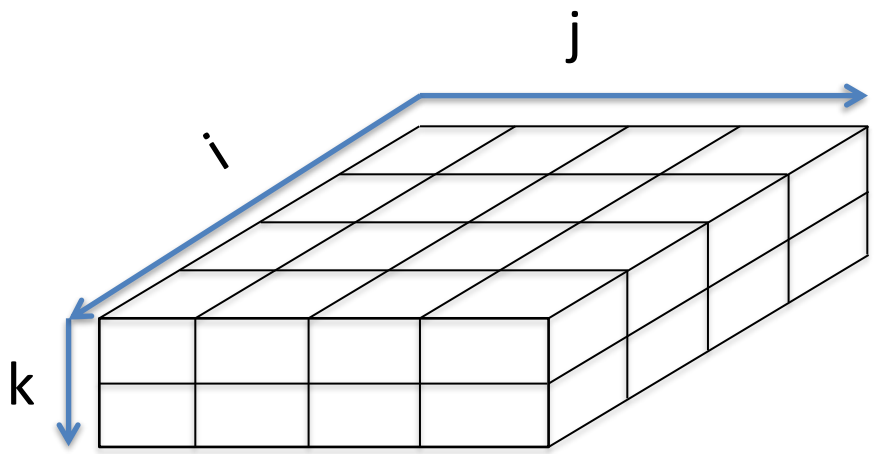
- Assume can fit cn^2/P data per processor, $c > 1$
- Processors form $(P/c)^{1/2}$ x $(P/c)^{1/2}$ x c grid



Example: P = 32, c = 2

2.5D Matrix Multiplication

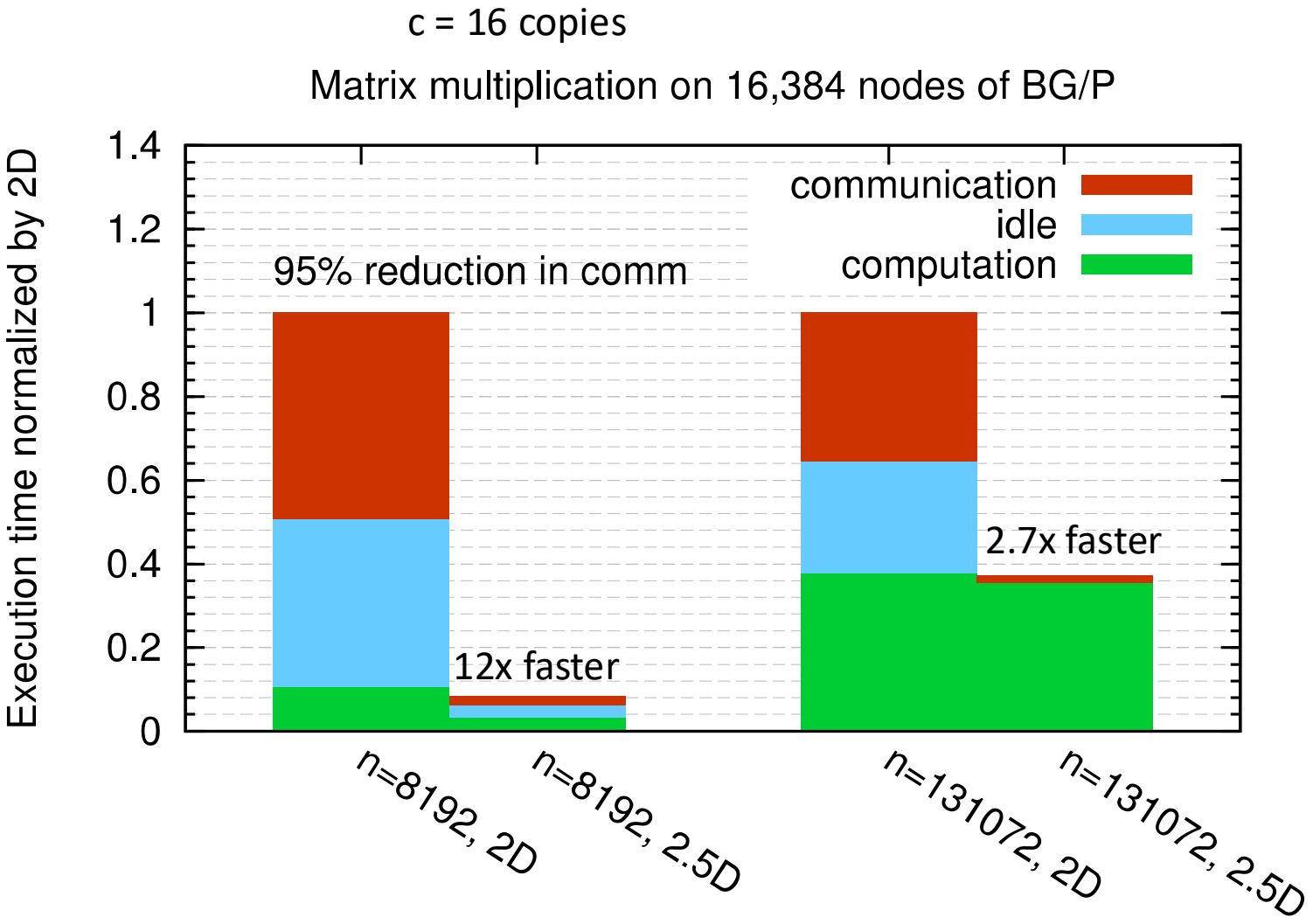
- Assume can fit cn^2/P data per processor, $c > 1$
- Processors form $(P/c)^{1/2}$ x $(P/c)^{1/2}$ x c grid



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

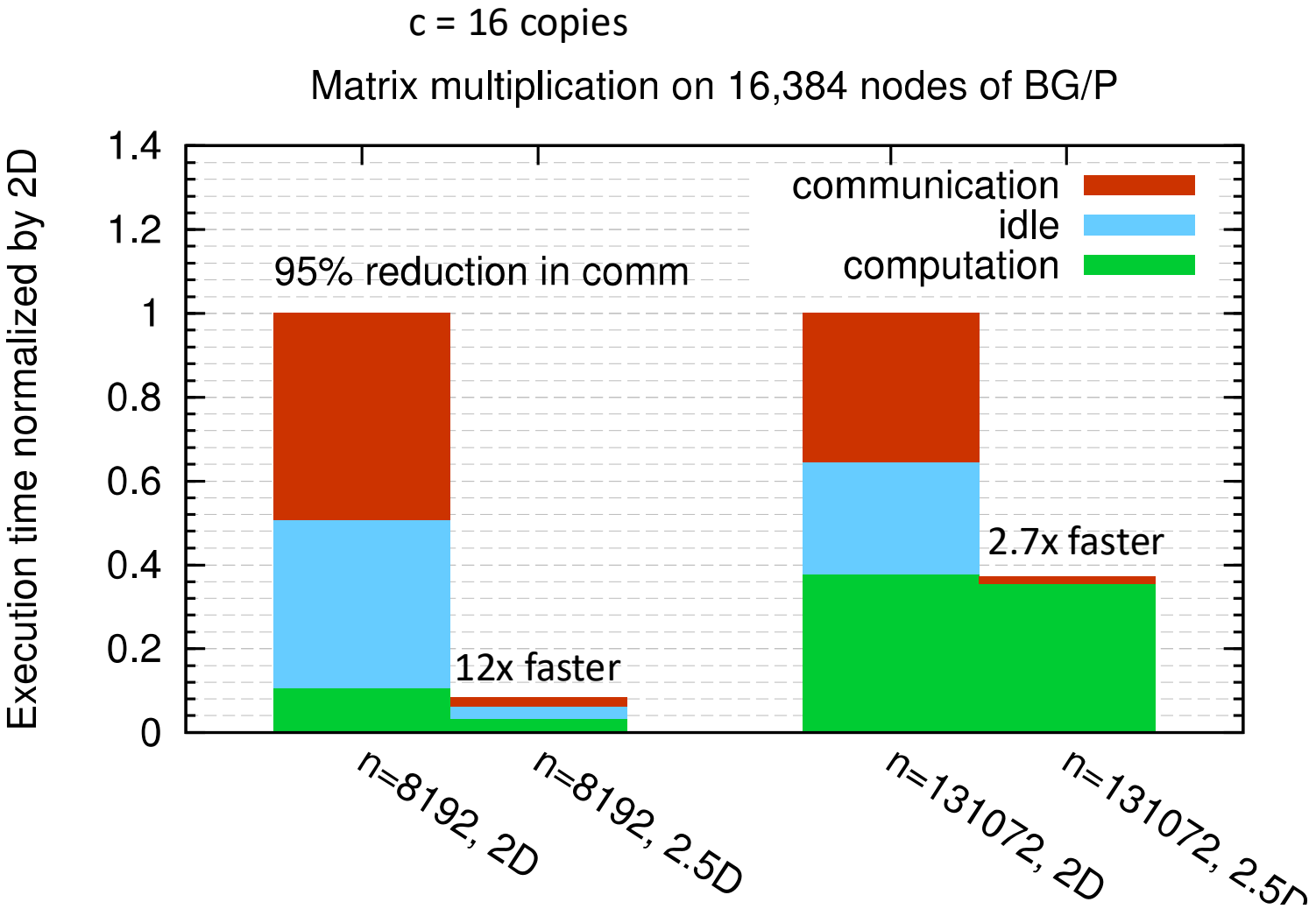
(1) P(i,j,0) broadcasts A(i,j) and B(i,j) to P(i,j,k)
(2) Processors at level k perform 1/c-th of SUMMA, i.e. 1/c-th of Σ_m A(i,m)*B(m,j)
(3) Sum-reduce partial sums Σ_m A(i,m)*B(m,j) along k-axis so P(i,j,0) owns C(i,j)

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



Matmul achieves perfect strong scaling in time and energy

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



Distinguished Paper Award, EuroPar'11 (Solomonik, D.)

Kwasniewski, Hoefler, et al (Best Student Paper, SC'19)

Beyond 2.5D Matmul

- Same idea extends to LU, QR, Jacobi, other algs
  - With a catch: can reduce bandwidth, but not latency
  - Bandwidth * Latency = $\Omega(n^2)$
- N-body problem
  - for i=1:n, f(i)=0, for j=1:n, f(i) = f(i) + force(p(i),p(j))
  - Bandwidth and latency both decrease with extra memory, attainable lower bounds
  - Same idea applies to Transformers
    - arxiv.org/pdf/2407.00611

Outline (of avoiding communication)

- Linear Algebra
  - Communication Lower Bounds for classical direct linear algebra
  - CA 2.5D Matmul
  - **TSQR - Tall-Skinny QR**
  - Iterative Methods for linear algebra
- Machine Learning
  - Training Neural Nets – “ImageNet training in minutes”
- And Beyond
  - Extending communication lower bounds and optimal algorithms to general loop nests

TSQR: QR of a Tall, Skinny matrix

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

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

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

TSQR: QR of a Tall, Skinny matrix

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

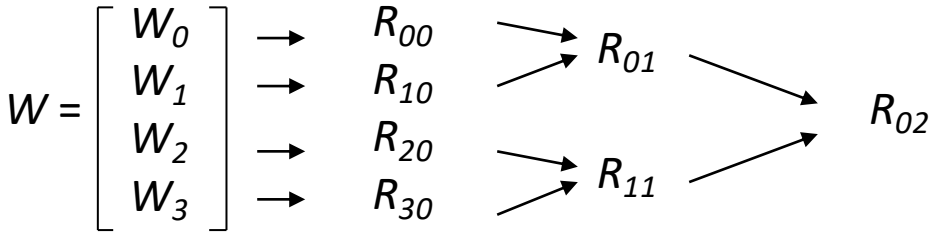
$$\begin{pmatrix} R_{00} \\ R_{10} \\ R_{20} \\ R_{30} \end{pmatrix} = \begin{pmatrix} Q_{01}\, R_{01} \\ Q_{11}\, R_{11} \end{pmatrix} = \begin{pmatrix} Q_{01} & \\ & Q_{11} \end{pmatrix} \cdot \begin{pmatrix} R_{01} \\ R_{11} \end{pmatrix}$$

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

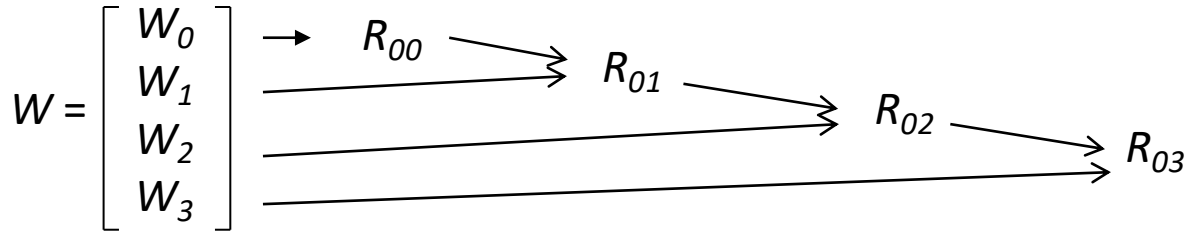
Output = { Q_{00}, Q_{10}, Q_{20}, Q_{30}, Q_{01}, Q_{11}, Q_{02}, R_{02} }

TSQR: An Architecture-Dependent Algorithm

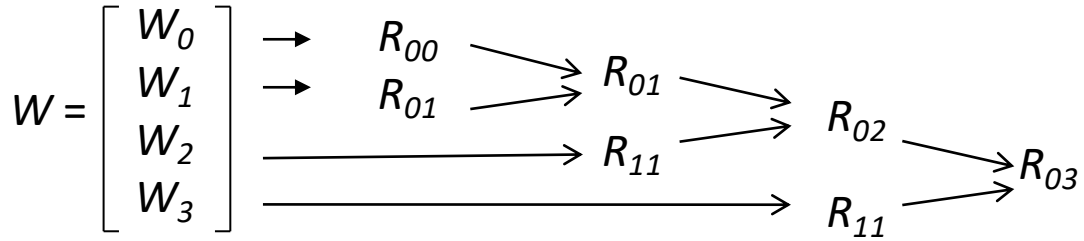
Parallel:



Sequential:



Dual Core:



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 x 100)
  - Grid – **4x** on 4 cities vs 1 city (Dongarra, Langou et al)
  - Cloud – **1.6x slower than just 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

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 x 100)
  - Grid – **4x** on 4 cities vs 1 city (Dongarra, Langou et al)
  - Cloud – **1.6x slower than just accessing data twice** (Gleich and Benson)

SIAG on Supercomputing Best Paper Prize, 2016

(D., Grigori, Hoemmen, Langou)

In LAPACK 3.7.0, 2016

LAPACK 3.10: Householder Reconstruction, 2021

Outline (of avoiding communication)

- Linear Algebra
  - Communication Lower Bounds for classical direct linear algebra
  - CA 2.5D Matmul
  - TSQR - Tall-Skinny QR
  - **Iterative Methods for linear algebra**
- Machine Learning
  - Training Neural Nets – “ImageNet training in minutes”
- And Beyond
  - Extending communication lower bounds and optimal algorithms to general loop nests

Avoiding Communication in Iterative Linear Algebra

- k-steps of iterative solver for sparse Ax=b or Ax=λx
  - 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

Minimizing Communication of GMRES to solve Ax=b

- GMRES: find x in span{b,Ab,…,A^kb} minimizing || Ax-b ||$_2$

Standard GMRES

```
for i=1 to k
   w = A · v(i-1)   … SpMV
   MGS(w, v(0),…,v(i-1))
   update v(i), H
endfor
solve LSQ problem with H
```

Communication-avoiding GMRES

```
W = [ v, Av, A^2v, … , A^kv ]
[Q,R] = TSQR(W)
      … "Tall Skinny QR"
build H from R
solve LSQ problem with H
```

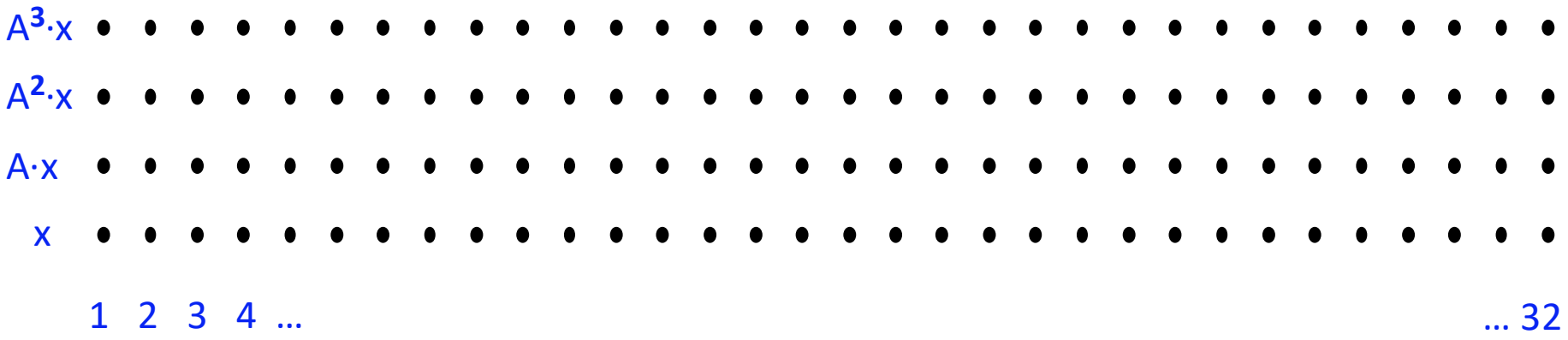
Sequential case: #words moved decreases by a factor of k

Parallel case: #messages decreases by a factor of k

- Oops – W from power method, precision lost!
- Fix: replace W by [v, $p_1(A)v$, $p_2(A)v$, …, $p_k(A)v$] (Hoemmen)
- Up to **2.3x** speedup for GMRES on 8 core Intel Clovertown
- Up to **4.2x** speedup for BiCGStab on 24K core Cray XE6 (Carson)

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

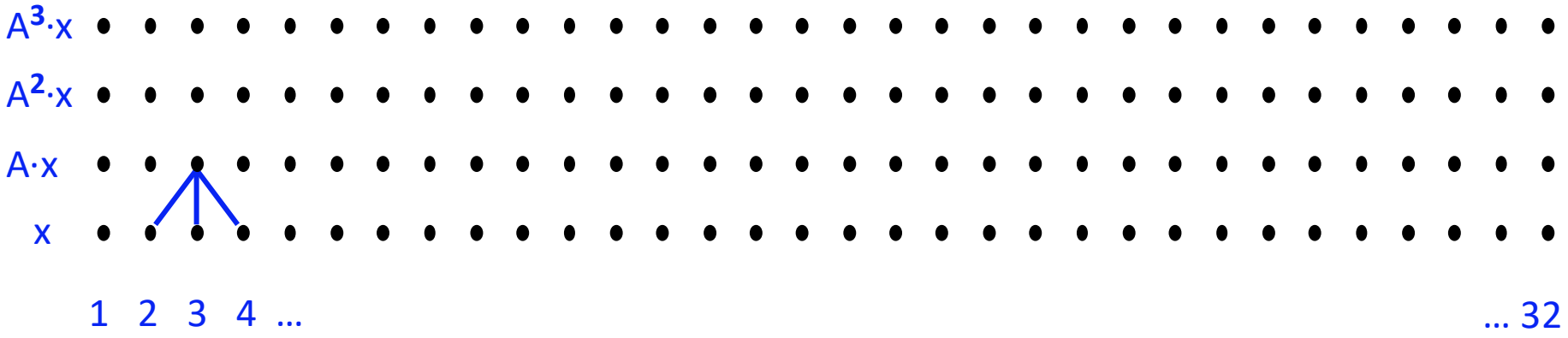
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]



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

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

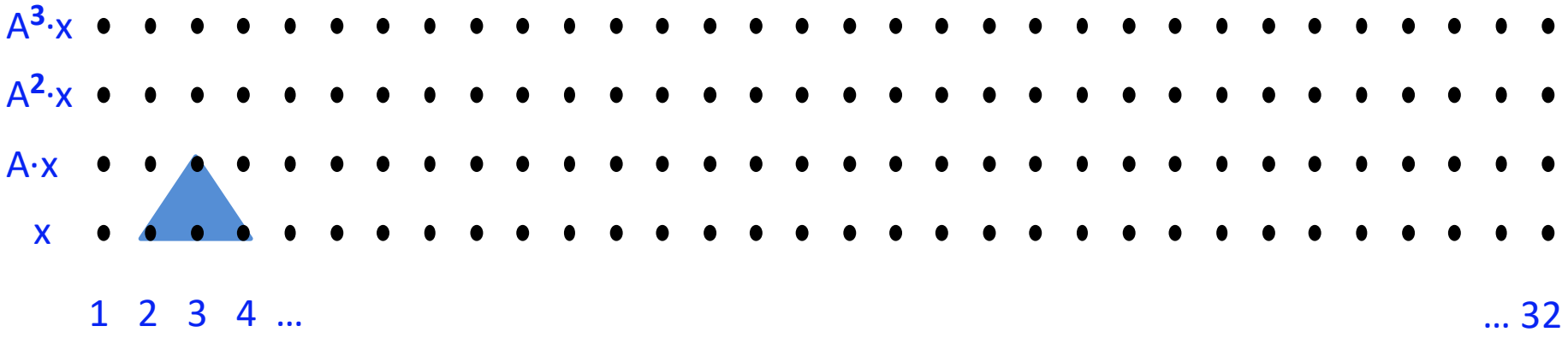
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]



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

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

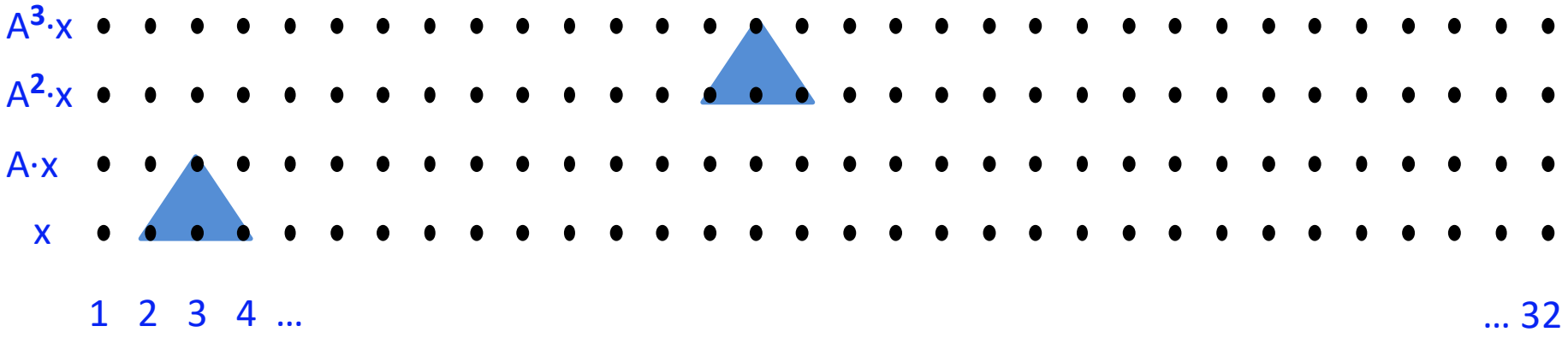
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]



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

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

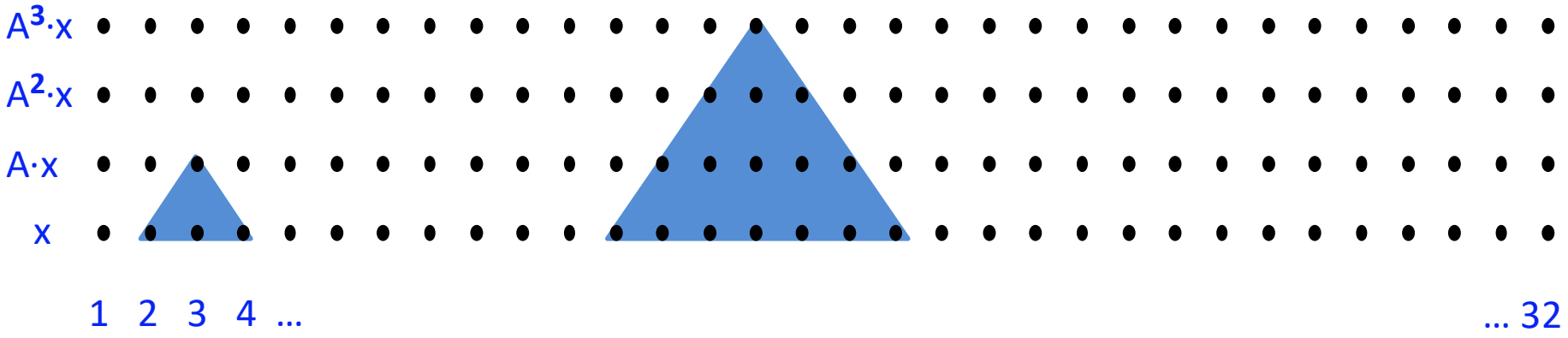
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]



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

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

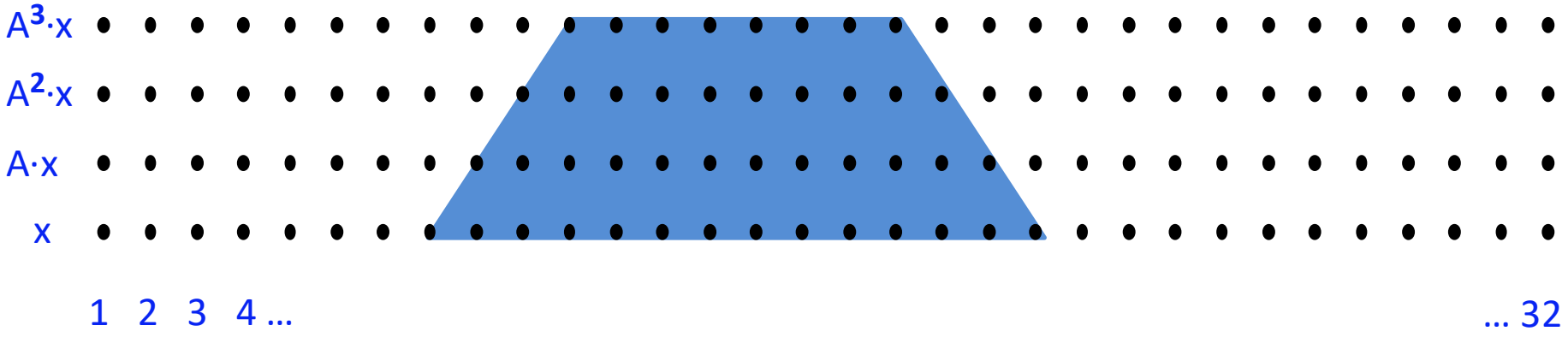
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]



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

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

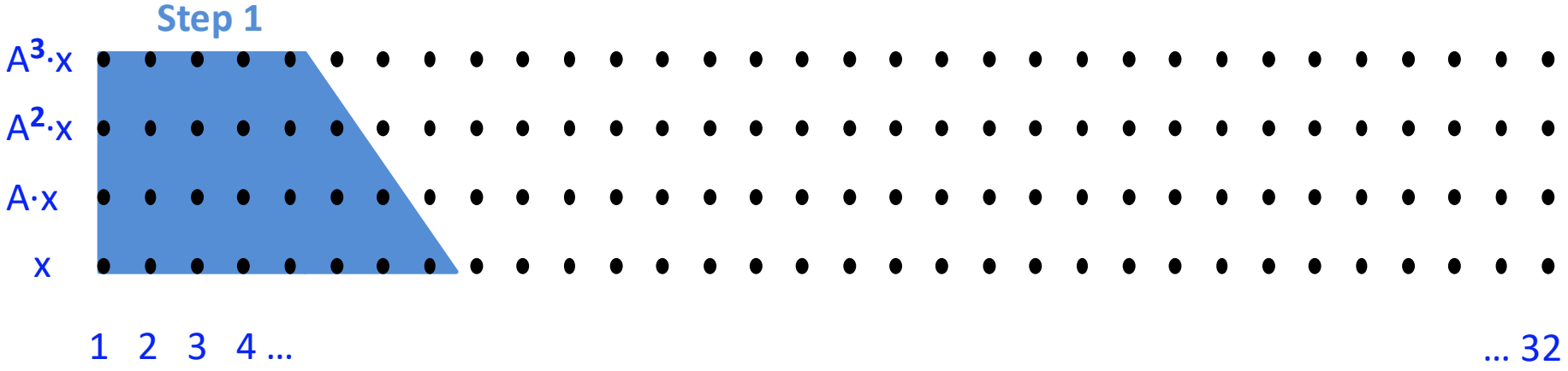
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]



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

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

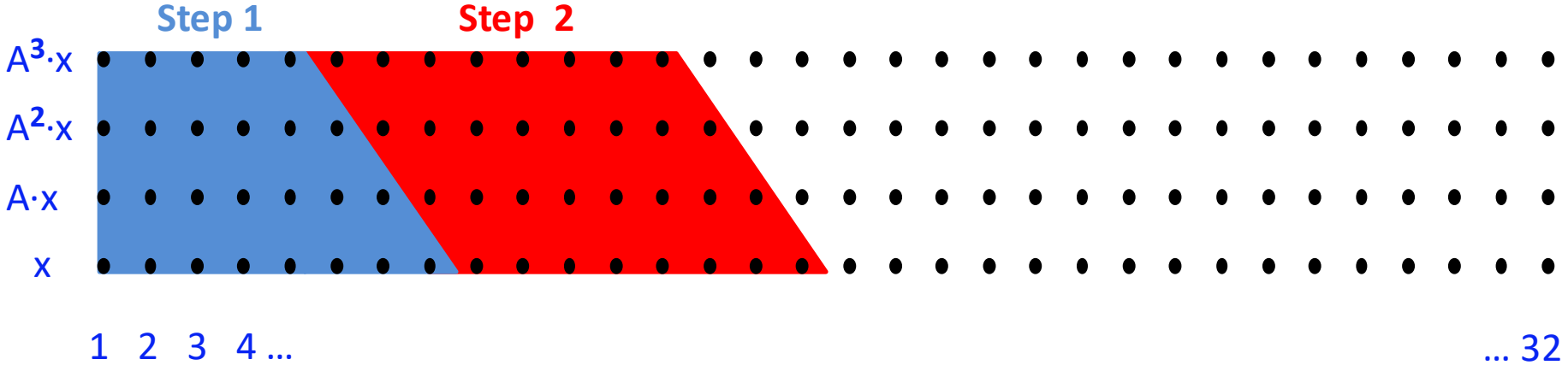
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]
- Sequential Algorithm



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

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

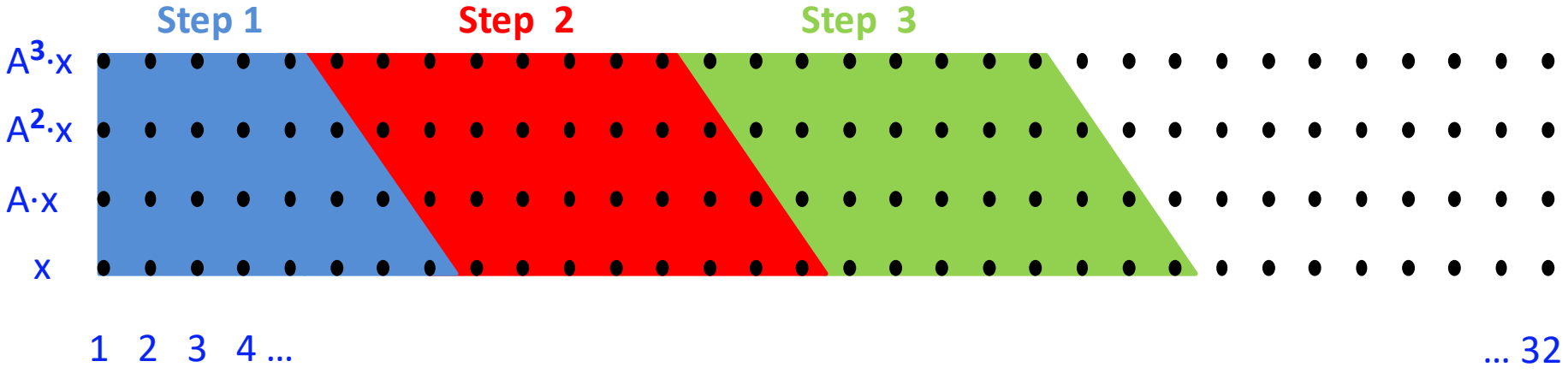
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]
- Sequential Algorithm



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

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

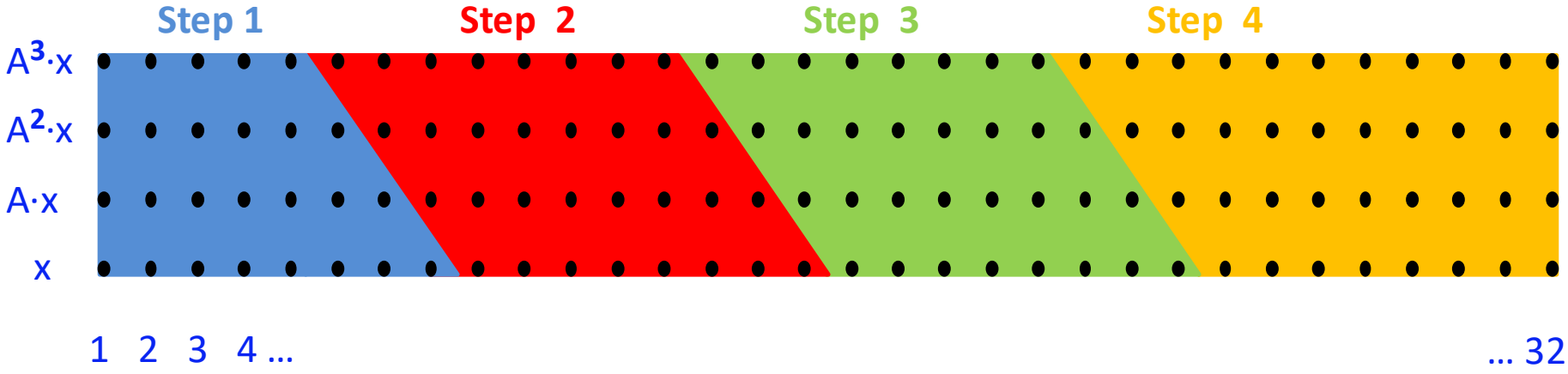
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]
- Sequential Algorithm



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

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

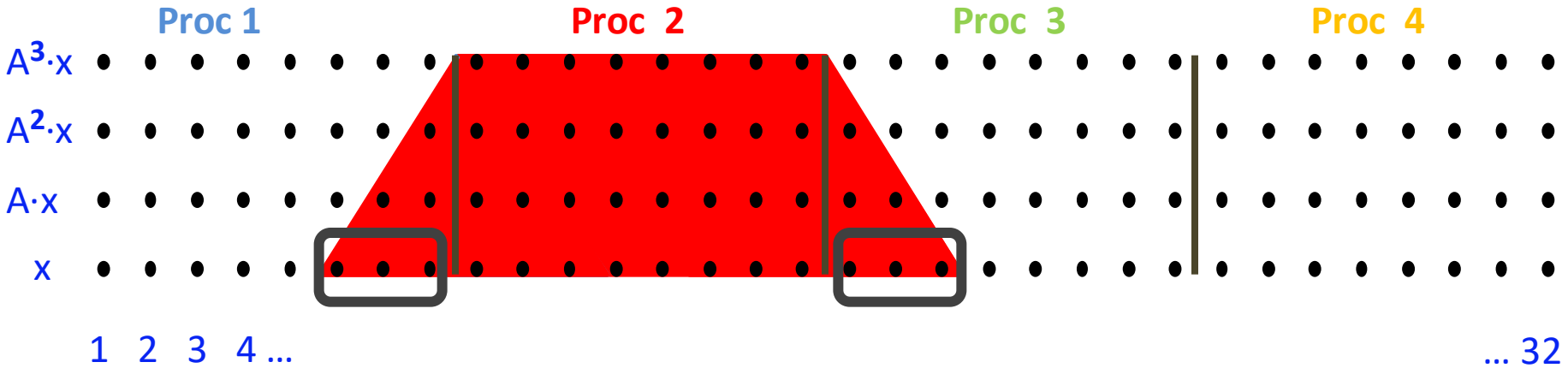
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]
- Sequential Algorithm



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

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

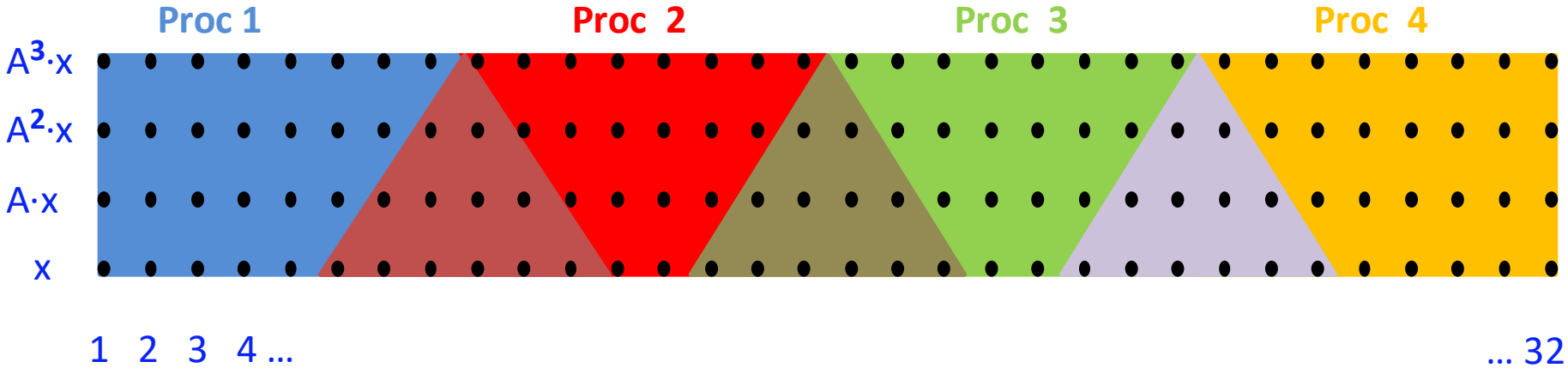
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]
- Parallel Algorithm



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

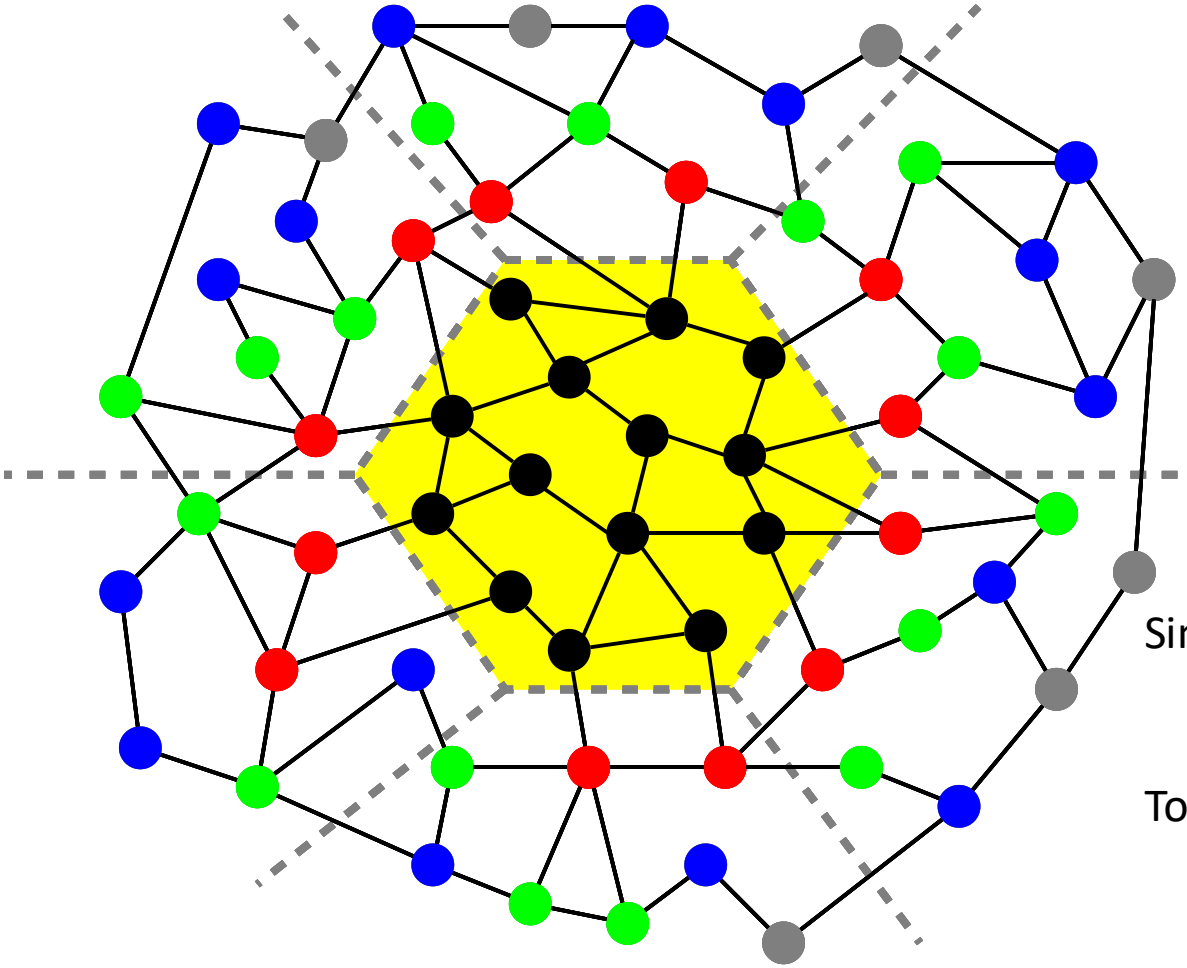
Communication Avoiding Kernels:
The Matrix Powers Kernel : [Ax, A^2x, …, A^kx]

- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]
- Parallel Algorithm



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

The Matrix Powers Kernel : [Ax, A^2x, …, A^kx] on a general matrix (nearest k neighbors on a graph)

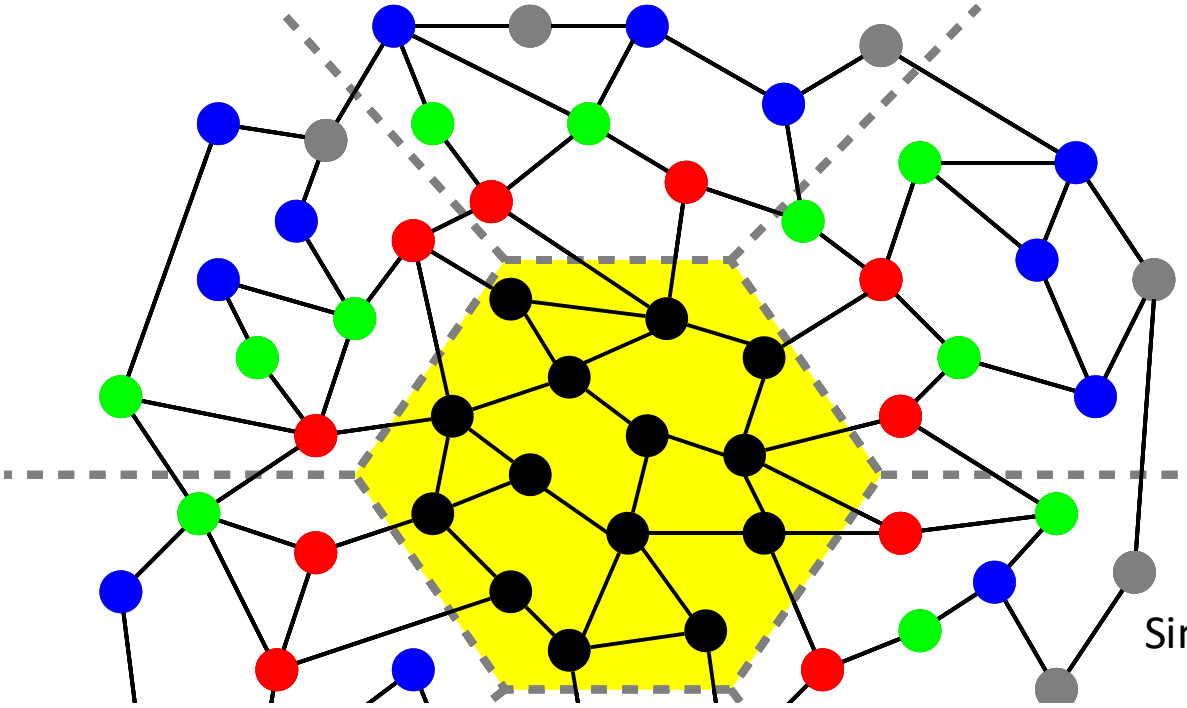


Simple block-row partitioning ➔
(hyper)graph partitioning

Top-to-bottom processing ➔
Traveling Salesman Problem

Same idea for general sparse matrices: k-wide neighboring region

The Matrix Powers Kernel : [Ax, A^2x, …, A^kx] on a general matrix (nearest k neighbors on a graph)



Simple block-row partitioning ➔
(hyper)graph partitioning

Alappat et al, "Level-Based Blocking for Sparse Matrices: Sparse Matrix-Power-Vector Multiplication", 2023

Builds on Alappat et al, SIAG on Supercomputing Best Paper Prize, 2024

CA-BiCGStab

Compute $r_0 = b - Ax_0$. Choose $r_0^\star$ arbitrary.
Set $p_0 = r_0$, $q_{-1} = 0_{N\times 1}$.
For $k = 0, 1, \ldots,$ until convergence, Do

$$P = [p_{sk}, Ap_{sk}, \ldots, A^s p_{sk}]$$
$$Q = [q_{sk-1}, Aq_{sk-1}, \ldots, A^s q_{sk-1}]$$
$$R = [r_{sk}, Ar_{sk}, \ldots, A^s r_{sk}]$$

//Compute the $1 \times (3s+3)$ Gram vector.
$g = (r_0^\star)^T [P, Q, R]$
//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,

$$b_{sk}^\ell = \left[B_1(:,\ell)^T, 0_{s+1}^T, 0_{s+1}^T\right]^T$$
$$c_{sk-1}^\ell = \left[0_{s+1}^T, B_2(:,\ell)^T, 0_{s+1}^T\right]^T$$
$$d_{sk}^\ell = \left[0_{s+1}^T, 0_{s+1}^T, B_3(:,\ell)^T\right]^T$$

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

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

$$\alpha_{sk+j} = \frac{<g, d_{sk+j}^0>}{<g, b_{sk+j}^1>}$$
$$q_{sk+j} = r_{sk+j} - \alpha_{sk+j}[P, Q, R]b_{sk+j}^1$$
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 $[P, Q, R]\, c_{sk+j}^\ell = A^\ell q_{sk+j}$

$$\omega_{sk+j} = \frac{<c_{sk+j+1}^1, Gc_{sk+j+1}^0>}{<c_{sk+j+1}^1, Gc_{sk+j+1}^1>}$$
$$x_{sk+j+1} = x_{sk+j} + \alpha_{sk+j} p_{sk+j} + \omega_{sk+j} q_{sk+j}$$
$$r_{sk+j+1} = q_{sk+j} - \omega_{sk+j}[P, Q, R]c_{sk+j+1}^1$$
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 $[P, Q, R]\, d_{sk+j+1}^\ell = A^\ell r_{sk+j+1}$

$$\beta_{sk+j} = \frac{<g, d_{sk+j+1}^0>}{<g, d_{sk+j}^0>} \times \frac{\alpha}{\omega}$$
$$p_{sk+j+1} = r_{sk+j+1} + \beta_{sk+j} p_{sk+j} - \beta_{sk+j}\omega_{sk+j}[P, Q, R]b_{sk+j}^1$$
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

Outline (of avoiding communication)

- Linear Algebra
  - Communication Lower Bounds for classical direct linear algebra
  - CA 2.5D Matmul
  - TSQR - Tall-Skinny QR
  - Iterative Methods for linear algebra
- **Machine Learning**
  - **Training Neural Nets – "ImageNet training in minutes"**
- And Beyond
  - Extending communication lower bounds and optimal algorithms to general loop nests

Training Neural Nets by Mini-Batch Stochastic Gradient Descent (SGD)

(You, Zhang, Hsieh, D., Keutzer, IPDPS 18)

- Iterate:
  - Pick a mini-batch of B data points
  - Update weights $W = W - \eta \cdot \nabla L(W)$
    - η = learning rate
    - $\nabla L(W)$ = gradient
- Data parallel version on P processors
  - Data partitioned, each processor gets B/P points
  - W_i replicated
  - Each processor computes $\nabla L(W)_i$ wrt its data
  - All-reduce: each processor computes
    $W_i = W_i - (\eta/P) \cdot \Sigma_{i=1}^{P} \nabla L(W)_i$

SGD: $W_i = W_i - (\eta/\mathrm{P}) \cdot \Sigma_{i=1}^{P} \nabla L(W)_i$

- Increase P to go faster: What are the bottlenecks?
- B/P decreases ⇒ less work per processor
  - Small matrix operations ⇒ locally communication bound
- Cost of each reduction $\Sigma_i \nabla L(W)_i$ grows
- Solution: increase B along with P
  - Maintain B/P ⇒ maintain processor efficiency
  - Try to converge in same #epochs (passes over data)
    - Same overall work, fewer reductions
- Oops: Convergence can be much worse
  - Convergence rate, test accuracy

Improving SGD convergence as B grows

- Facebook's strategy: adjust learning rate η
  - Increase B to kB $\Rightarrow$ increase η to $\mathrm{k}\eta$
  - Warmup rule: Start with smaller η, then increase
- Only worked up to B=1K for AlexNet (tried lots of tuning)
- Fix: Add Layer-wise Adaptive Rate Scaling (LARS)
  - $\| W \| / \| \nabla L(W) \|$ can vary by 233x between AlexNet layers
  - Let η be proportional to $\| W \| / \| \nabla L(W) \|$
  - (You, Gitman, Ginsburg, 2017)
  - Also need momentum, weight decay

ImageNet Training in Minutes

Speedup for AlexNet (for batchsize = 32K, changed LRN to BN)

Batch Size	Epochs	Top-1 Accuracy	Platform	Time
256	100	58.7%	8-core + K20 GPU	144 hrs
512	100	58.8%	DGX-1 station	6h 10m
4096	100	58.4%	DGX-1 station	2h 19m
32k	100	58.6%	512 KNLs	24m
32k	100	58.6%	1024 CPUs	**11m**

Speedup for ResNet50

Batch Size	Epochs	Top-1 Accuracy	Platform	Time
32	90	75.3%	CPU + M40 GPU	336h
256	90	75.3%	16 KNLs	45h
32K	90	75.4%	512 KNLs	60m
32K	90	75.4%	1600 CPUs	32m
32K	90	75.4%	2048 KNLs	**20m**

135x

ImageNet Training in Minutes

- Best Paper Prize at ICPP 2018
- Open Source in Caffe, NVIDIA Caffe, Facebook Caffe 2 (PyTorch)
- Media coverage by CACM, EureKalert, Intel, NSF, Science Daily, Science NewsLine, etc.
- Subsequent work at Tencent reached 4 minutes
- LARS adopted by industry standard benchmark MLPerf in 2019

Outline (of avoiding communication)

- Linear Algebra
  - Communication Lower Bounds for classical direct linear algebra
  - CA 2.5D Matmul
  - TSQR - Tall-Skinny QR
  - Iterative Methods for linear algebra
- Machine Learning
  - Training Neural Nets – “ImageNet training in minutes”
- **And Beyond**
  - **Extending communication lower bounds and optimal algorithms to general loop nests**

Communication lower bounds and optimal algorithms for general loop nests

- for i = 1:n, for j=1:n, for k = 1:n
  C(i,j) = C(i,j) + A(i,k)*B(k,j)
- #Words moved between main memory and cache of size M = $\Omega(n^3 / M^{1/2})$, attainable
- For $(i_1,i_2,\ldots i_k) \in S \subseteq \mathbb{Z}^k$, do something with
  - A(i_1), B(i_2, i_3+i_4), C(i_1-i_2, i_2+3*i_3- 5*i_4 +2, …), …
- Thm: #Words moved = $\Omega(|S| / M^{e_{HBL}})$
  (Christ, D., Knight, Scanlon, Yelick)
  - HBL = Hölder / Brascamp / Lieb
  - Uses results by Christ, Tao, others
- Thm: There exists an optimal tiling that attains this lower bound (D., Rusciano)

What's left?

- Dealing with small loop bounds
  - Ex: Matvec special case of Matmul, not optimizable
  - Special cases: CNNs
  - Thm: If all subscripts like (i),(i,j), etc, and S = parallelepiped, ∃ tighter, attainable lower bound (D., Dinh)
- Dealing with dependencies
  - Special cases: Linear algebra outside matmul, Floyd-Warshall, … open problems!
- More realistic performance models than α, β, γ
  - Variable precision
  - Heterogeneous processors, accelerators, network topologies, differing costs of read and writes, …
- Sketching – can beat matmul
- Need to automate! (i.e. compilers)
  - https://iocomplexity.corse.inria.fr/iolb

Collaborators and Supporters

- **James Demmel**, **Kathy Yelick**, Vivek Bharadwaj, Grace Dinh, Tianyu Liang
- Peter Ahrens, Michael Anderson, Grey Ballard, Austin Benson, Erin Carson, Maryam Dehnavi, Aditya Devakonda, Michael Driscoll, David Eliahu, Andrew Gearhart, Evangelos Georganas, Mark Hoemmen, Shoaib Kamil, , Nicholas Knight, Penporn Koanantakool, Ben Lipshitz, Marghoob Mohiyuddin, Hong Diep Nguyen, Jason Riedy, Alex Rusciano, Oded Schwartz, Edgar Solomonik, Omer Spillinger, Yang You
- Abhinav Bhatele, Aydin Buluc, Michael Christ, Ioana Dumitriu, Kimon Fountoulakis, Armando Fox, David Gleich, Ming Gu, Jeff Hammond, Mike Heroux, Olga Holtz, Kurt Keutzer, Julien Langou, Xiaoye Li, Michael Mahoney, Devin Matthews, Tom Scanlon, Michelle Strout, Sam Williams, Hua Xiang, Zhao Zhang, Cho-Jui Hsieh,
- Jack Dongarra, Mark Gates, Jakub Kurzak, Dulceneia Becker, Ichitaro Yamazaki, …
- Sivan Toledo, Alex Druinsky, Inon Peled, Greg Henry, Peter Tang,
- Laura Grigori**,** Sebastien Cayrols, Simplice Donfack, Mathias Jacquelin, Amal Khabou, Sophie Moufawad, Mikolaj Szydlarski
- Members of SLICE, ADEPT, ASPIRE, BEBOP, ParLab, 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

For more details on avoiding communication

- Bebop.cs.berkeley.edu
  - 155 page linear algebra survey in Acta Numerica (2014)
  - Book in progress (with Ballard, Carson, Grigori)
- CS267 – Berkeley's Parallel Computing Course
  - Slides available at https://sites.google.com/lbl.gov/cs267-spr2025
  - Earlier recorded lectures also available at https://sites.google.com/lbl.gov/cs267-spr2022

Summary (of avoiding communication)

Time to redesign all

linear algebra, machine learning, n-body, …
algorithms and software, and compilers

Don’t Communic…

58

Outline (of Grading accuracy of BLAS)

- What accuracy guarantees should BLAS implementations, using low-precision accelerators, or Strassen-like algorithms, promise?
- How can we verify promises made for proprietary, closed-source code?
  - Initial approach: ``reverse engineering'' the algorithm
  - But hard to foresee whatever an LLM may imagine (or hallucinate) tomorrow
- How does this impact higher level algorithms, eg Cholesky?
- Public test code available on github
- Early version adopted by Nvidia [2]

Possible Grades for GEMM: For data and matrix dimensions in "some range," satisfies:

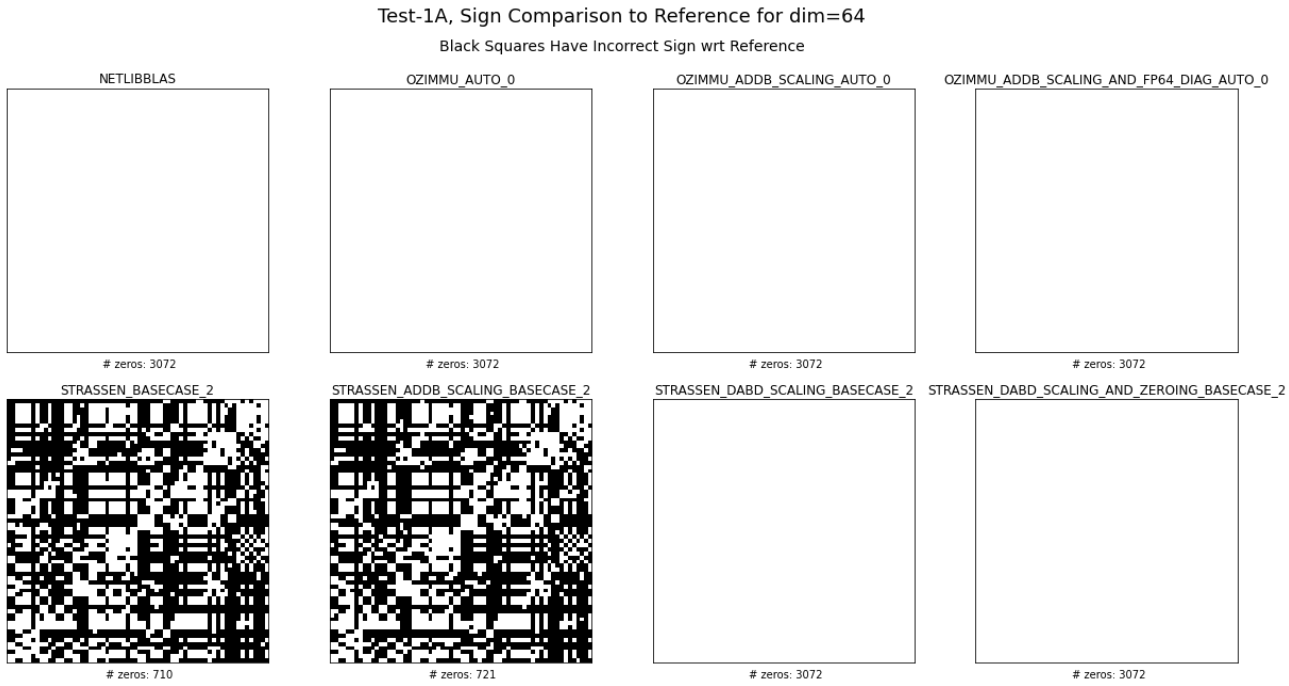
- Grade = "A"
  - $\forall i,j\colon\ |fl(A\cdot B)(i,j)-(A\cdot B)(i,j)|\le f(n)\varepsilon(|A|\cdot|B|)(i,j)$
  - Must do O(n^3) flops, so only classical matmul, not Strassen-like [13]
  - Invariant under diagonal scaling $A\ \rightarrow D_1\ \cdot \mathrm{A}\cdot D_3$, $B\ \rightarrow\ D_3^{-1}\cdot B\cdot D_2$
- Grade = "C"
  - $\|\ fl(A\cdot B)-(A\cdot B)\ \|\le f(n)\varepsilon\ \|\ A\ \|\ \|\ B\ \|$
  - Can be satisfied by Strassen-like algorithms [9,10,11], enough for many backward error analyses [12]
- Grade = "B"
  - $\forall i,j\colon\ |fl(A\cdot B)(i,j)-(A\cdot B)(i,j)|\le f(n)\varepsilon\ \|\ A(i,:)\ \|\ \|\ B(:,j)\ \|$
  - "Between" Bounds 1 and 2; invariant under diagonal scaling with D_1 and D_2, not D_3
  - Satisfied by emulation methods (eg Ozaki 1 and Ozaki 2)
- "A+" also possible (high relative accuracy), but expensive, will not test for it

Test 1: Strassen vs classical

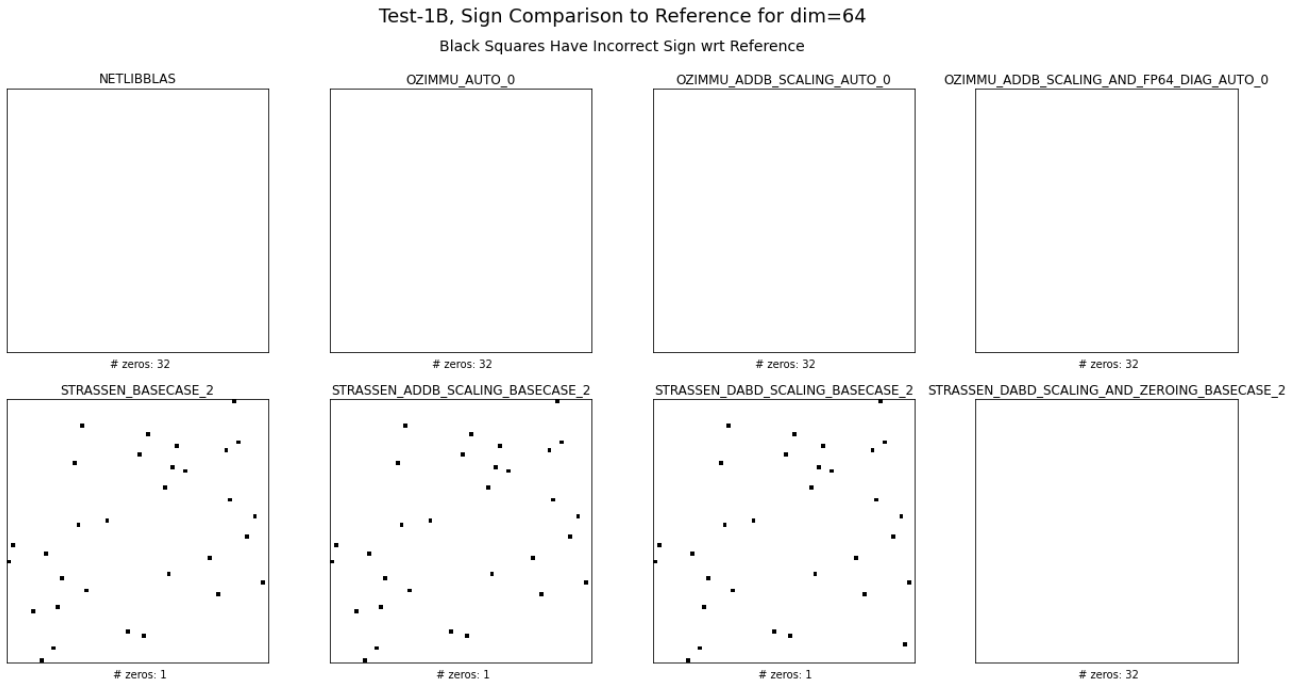
- Test 1a - Gameable
  - A = randn(n, n), B = randn(n, n), set a few randomly chosen rows of A and columns of B to zero, so corresponding rows and columns of $A \cdot B$ are zero
  - Strassen will not compute these as zero (w.h.p. from roundoff)
  - But easy to game by scaling with D_1,D_2 so $\| (D_1 A)(i,:) \| = \| (BD_2)(:,j) \| = 1$ to attain grade="B" for Strassen, can detect and fix zero rows and columns of $A \cdot B$
- Test 1b – Not Gameable
  - Pick some random rows of A, make ~50% randomly sparse, pick equally many random columns of B, make complementarily sparse to a sparse row of A, so corresponding random entries of $A \cdot B$ are exactly sums of zeros
  - Strassen will not compute all these entries of $A \cdot B$ as zero (w.h.p, from roundoff), so not gameable.
  - Can determine size of base case n_0 for Strassen, when switch to $O(n_0^3)$ algorithm
  - Conjecture: Works for Strassen-like algorithms in general
  - Set "tiny" entries of product to 0; errors could result from Strassen or $O(n^3)$ with emulation

Results of Test 1a (see [24] for emulation code)

Results of Test 1a



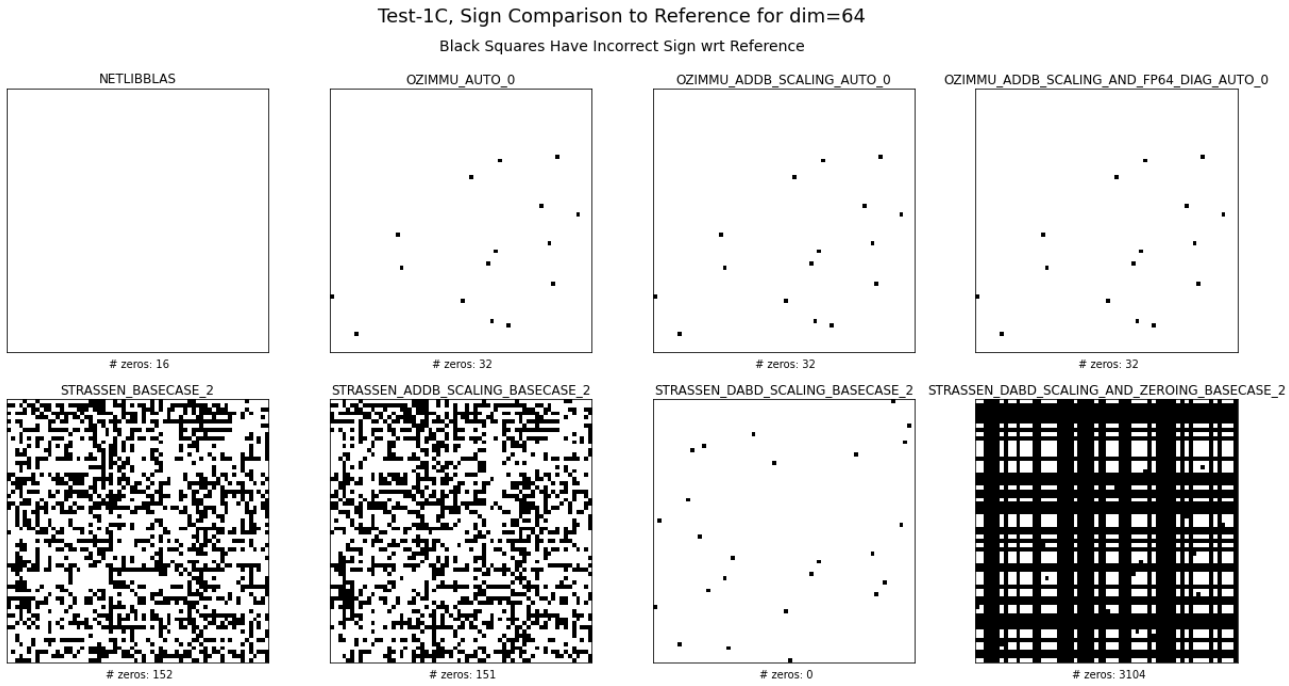
Results of Test 1b



Test 1: Grade ”A” or not

- Test 1c – Not Gameable (so far)
  - Strengthen Test 1b so only an $O(n^3)$ algorithm using conventional floating point can pass
  - In some random subset of sparsified rows/columns of A and B, modify them:
    - For some random k, set $A(i,k) = B(k,j) = x$, so $C(i,j) = x^2$, where $x = O(\text{UN}^{1/2})$
    - Set other nonzero $A(i,*)$ and $B(*,j) = O(\text{OV}^{1/2})$
  - Both Strassen and $O(n^3)$ with emulation will get x^2 wrong
  - Diagonal scaling $D_1 \cdot A \cdot D_3$ and $D_3^{-1} \cdot B \cdot D_2$ doesn’t help

Results of Test 1c



More testing

- Have many more tests
  - Distinguish “A” from “B” from “C”
  - Best not to rely on one!
- Verified on many implementations (with gaming)
  - “A”: Netlib BLAS, OpenBLAS, BLIS, cuBLAS, MKL BLAS
  - “B”: Ozaki 1, Ozaki 2
  - “C”: Strassen
- Publicly available (soon!)

What about solving spd $A \cdot x = b$ with tiled Cholesky?

- GEMM, SYRK and TRSM used as in LAPACK; BLAS2, BLAS1 unchanged
- Let $A = D \cdot H \cdot D$, D diagonal with $D_{ii} = A_{ii}^{1/2}$, $H_{ii} = 1$
  - Possible for cond(H) $\ll$ cond(A) [20,21] if $max_i A_{ii} \gg min_i A_{ii}$
- Let $\hat{x}$ = computed solution, $\hat{L}$ = Cholesky factor, and $E = \hat{L} \cdot \hat{L}^T - A$
- Bound 1 (norm-wise): Grade = "C" [19]
  - $\| E \| \leq f(n)\varepsilon \| A \|$, $\| x - \hat{x} \| / \| x \| = O(\varepsilon)$ cond(A)
- Bound 2 (component-wise): Grade = "A" [17,18]
  $\forall i,j \ |E(i,j)| \leq f(n)\varepsilon(|\hat{L}| \cdot |\hat{L}^T|)(i,j)$, $\| D(x - \hat{x}) \| / \| Dx \| = O(\varepsilon)$ cond(H)
- Bound 3 (mixed) : Grade = "B"
  - $\forall i,j \ |E(i,j)| \leq f(n)\varepsilon D_{ii} D_{jj}$, $\| D(x - \hat{x}) \| / \| Dx \| = O(\varepsilon)$ cond(H)

Future Work

- Similar grades apply to TRSM, other BLAS3
  - Need to implement analogous tests, release
- Expand to complex data (new tricks)
- Error analysis beyond Cholesky
- Better estimates of $f(n)$ factor in error bounds
- Test exception handling

Some Additional References

- Class web page for Matrix Computations at UC Berkeley
  - people.eecs.berkeley.edu/~demmel/ma221_Fall24/
  - Lecture notes, links to many other references
- Class web page for Parallel Computing at UC Berkeley
  - sites.google.com/lbl.gov/cs267-spr2024
  - Lecture notes, links to many other references
  - Earlier recorded lectures also available at sites.google.com/lbl.gov/cs267-spr2022

References related to Avoiding Communication (1/8)

- [1] E. Solomonik, J. Demmel, "Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms," EuroPar'11, **Distinguished Paper Prize**
- [2] E. Solomonik, "Provably efficient algorithms for numerical tensor algebra," PhD Thesis, Computer Science Division, UC Berkeley, 2015, **Householder Prize for Best PhD Thesis**
- [3] P. Koanantakool, A. Azad, A. Buluc, D. Morozov, S-Y. Oh, L. Oliker, K. Yelick, "Communication-Avoiding Parallel Sparse-Dense Matrix-Matrix Multiplication," IPDPS'16
- [4] E. Solomonik, A. Buluc, J. Demmel, "Minimizing communication in all-pairs shortest paths," IPDPS'13
- [5] P. Koanantakool, M. Driscoll, E. Georganas, E. Solomonik, K. Yelick, "A Communication-Optimal N-Body Algorithm for Direct Interactions", IPDPS'13

References related to Avoiding Communication (2/8)

- [6] M. Anderson, G. Ballard, J. Demmel, K. Keutzer, "Communication-Avoiding QR Decomposition for GPUs," IPDPS'11
- [7] J. Demmel, L. Grigori, M. Hoemmen, J. Langou, "Communication-Optimal Parallel and Sequential QR and LU Factorization," SIAM J. Sci. Comp., v. 34, n. 1, 2012, **SIAG on Supercomputing Best Paper Prize, 2016**
- [8] G. Ballard, J. Demmel, N. Knight, "Avoiding Communication in Successive Band Reduction," ACM Trans. Par. Comp., v. 1, i. 2, Jan 2015
- [9] G. Ballard, J. Demmel, N. Knight, "Communication-Avoiding Successive Band Reduction," PPoPP'12
- [10] S. Williams, M. Lijewski, A. Almgren, B. Van Straalen, E. Carson, N. Knight, J. Demmel, "s-step Krylov Subspace Methods as Bottom Solvers for Geometric Multigrid," IPDPS'14

References related to Avoiding Communication (3/8)

- [11] A. Devarakonda, J. Demmel, K. Fountoulakis, M,. Mahoney, "Avoiding Synchronization in First-Order Methods for Sparse Convex Optimization," IPDPS'18
- [12] Y. You, K. Czechowski, L. Song, R. Vuduc, J. Demmel, "CA-SVM: Communication-avoiding support vectors machines on distributed systems," IPDPS'15, **Best Paper Award**
- [13] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, K. Keutzer, "ImageNet Training in Minutes," ICPP'18, **Best Paper Award**
- [14] G. Ballard, J. Demmel, O. Holtz, O. Schwartz, "Minimizing communication in numerical linear algebra," SIAM J. Mat. Anal. Appl., v. 32, n. 3, 2011, **SIAG/LA Best Paper Prize 2012**
- [15] G. Ballard, J. Demmel, O. Holtz, O. Schwartz, "Graph Expansion and Communication Costs of Fast Matrix Multiplication," J. ACM, v. 59, n. 6, 2012, **Invited to appear as CACM Research Highlight**

References related to Avoiding Communication (4/8)

- [16] J. Scott, “An I/O-Complexity Lower Bound for All Recursive Matrix Multiplication Algorithms by Path-Routing”, PhD Thesis, Math Dept., UC Berkeley, 2015
- [17] I. Dumitriu, O. Holtz, J. Demmel, R. Kleinberg, “Fast Matrix Multiplication is Stable,” Num. Math., v. 106, n. 2, 2007, **Cited in 2008 EMS Prize of O. Holtz**
- [18] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, O. Schwartz, “Brief Announcement: Strong Scaling of Matrix Multiplication Algorithms and Memory-Independent Communication Lower Bounds”, SPAA’12
- [19] G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, O. Schwartz, “Communication-Optimal Parallel Algorithm for Strassen’s Matrix Multiplication,” SPAA’12
- [20] G. Ballard, J. Demmel, B. Lipshitz, O. Schwartz, “Communication-Avoiding Parallel Strassen: Implementation and Performance,” Supercomputing’12

References related to Avoiding Communication (5/8)

- [21] A. Gearhart, J. Demmel, B. Lipshitz, O. Schwartz, "Perfect Strong Scaling Using No Additional Energy," A. Gearhart, J. Demmel, B. Lipshitz, O. Schwartz, IPDPS'13
- [22] E. Solomonik, E. Carson, N. Knight, J. Demmel, "Tradeoffs between synchronization, communication and work in parallel linear algebra computations," ACM Trans. Par. Comp, v. 3, i. 1, 2016**, invited**
- [23] L. Grigori, J. Demmel, H. Xiang, "Communication avoiding Gaussian Elimination," Supercomputing 2008
- [24] L. Grigori, J. Demmel, H. Xiang, "CALU: A communication optimal LU factorization algorithm," SIAM J. Mat. Anal. Appl., v. 32, n. 4, 2011
- [25] G. Ballard, J. Demmel, L. Grigori, M.Jacquelin, H. D. Nguyen, N. Knight, "Reconstructing Householder Vectors from Tall-Skinny QR", J. Par. Distr. Comp., v. 85, i. C, 2015

References related to Avoiding Communication (6/8)

- [26] J. Demmel, M. Christ, N. Knight, T. Scanlon, K. Yelick, "Communication Lower Bounds and Optimal Algorithms for Programs that References Arrays – Part 1", UC Berkeley Tech Report UCB/EECS-2013-61, 2013
- [27] J. Demmel, M. Christ, N. Knight, T. Scanlon, K. Yelick, "On multilinear inequalities of Holder-Brascamp-Lieb type for torsion-free discrete abelian groups,"J. Logic and Analysis, v. 16, July, 2024
- [28] A. Rusciano, J. Demmel, "Parallelepipeds obtaining HBL lower bounds," arxiv.org/abs/1612.04003, 2017; UC Berkeley Tech Report UCB/EECS-1016-197, 2016
- [29] J. Demmel, G. Dinh, "Brief Announcement: Communication-optimal tilings for projective nested loops with arbitrary bounds", SPAA'20
- [30] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, O. Schwartz, "Communication lower bounds and optimal algorithms for numerical linear algebra", Acta Numerica, v. 23, 2014

References related to Avoiding Communication (7/8)

- [31] P. Koanantakool, K. Yelick, “A computation and communication-optimal parallel direct 3-body algorithm”, Supercomputing’14
- [32] Z. Liu, S. Wang, S. Cheng, Z. Zhao, K. Wang, X. Zhao, Y. You, J. Demmel, “WallFacer: Harnessing Multi-dimensional Ring Parallelism for Efficient Long Sequence Model Training,” arXiv:2407.00611, 2024
- [33] A. Chen, G. Dinh, M. Haberle, O. Holtz, J. Demmel, “Communication bounds for convolutional neural networks”, PASC’22
- [34] G. Ballard, A. Gearhart, B. Lipshitz, J. Demmel, Y. Oltchik, O. Schwartz, S. Toledo, “Network topologies and inevitable contention”, COM-HPC’16
- [35] E. Carson, J. Demmel, L. Grigori, N. Knight, P. Koanantakool, O. Schwartz, H. V. Simhadro, “Write-Avoiding Algorithms”, IPDPS’16

References related to Avoiding Communication (8/8)

- [36] E. Carson, ”Communication-Avoiding Krylov Subspace Methods in Theory and Practice”, PhD Thesis, Computer Science Division, UC Berkeley, 2015, and many references therein
- [37] M. Hoemmen, “Communication-avoiding Krylov Subspace Methods,” PhD Thesis, Computer Science Division, UC Berkeley, 2010, and many references therein
- [38] J. Demmel, I. Dumitriu, O. Holtz, “Fast Linear Algebra is Stable”, Num. Math., v. 108, n. 1, 2007
- [39] G. Ballard, A. Buluc, J. Demmel, L. Grigori, B> Lipshitz, O. Schwartz, S. Toledo, “Communication-Optimal Parallel Multiplication of Sparse Random Matrices,” SPAA’13

References related to Mixed Precision (1/5)

1. “IEEE Working Group P3109 Interim Report on Binary Floating-point Formats for Machine Learning,’’ 28 June 2026, https://github.com/P3109/Public
2. “Guaranteed DGEMM Accuracy While Using Reduced Precision Tensor Cores Through Extensions of the Ozaki Scheme,” A. Schwartz et al, Proc. SC Asia 2026, 10.1145/3773656.3773670
3. “Proposed Exception Handling for the BLAS and LAPACK,” J. Demmel et.al, Proc. Workshop on Software Correctness for HPC Applications, Nov 2022
4. “OCP 8-bit floating point specification (OFP8)”, P. Micikevisius et. al., https://www.opencompute.org/documents/ocp-8-bit-floating-point-specification-ofp8-revision-1-0-2023-12-01-pdf-1
5. “8-bit numerical formats for deep neural networks”, B. Noune et. al., https://arxiv.org/abs/2206.02915

References related to Mixed Precision (2/5)

6. “Tesla Dojo Technology: A guide to Tesla’s configurable floating point formats and arithmetic”, 2023, https://web.archive.org/web/20230503235751/https://tesla-cdn.thron.com/static/MXMU3S_tesla-dojo-technology_1WDVZN.pdf
7. “On Stochastic Rounding with few random bits”, A. Fitzgibbon and S. Felix, ARITH 2025
8. LoFloat: https://github.com/SudhanvaKulkarni123/LoFloat
9. “Fast Matrix Multiplication is Stable,” J. Demmel, I. Dumitriu, O. Holtz, R. Kleinberg, Num. Math, v. 106, n. 2, 2007, arxiv.org/abs/math/0603207

References related to Mixed Precision (3/5)

10. “Fast Linear Algebra is Stable,” J. Demmel, I. Dumitriu, O. Holtz, Num. Math., v. 108, n. 1, 2007, arxiv.org/abs/math/0612264
11. “Improving the numerical stability of fast matrix multiplication”, G. Ballard, A. Benson, A. Druinsky, B. Lipshitz, O. Schwartz, SIAM J. Mat. Anal. Appl., v. 37, n. 4, 2016
12. “Stability of Block Algorithms with Fast Level-3 BLAS”, J. Demmel, N. Higham, ACM TOMS, v. 18, n. 3, 1992
13. “Computational complexity and numerical stability,” W. Miller, SIAM J. Comput. vol. 4, n. 2, 1975

References related to Mixed Precision (4/5)

14. “Reproducible BLAS: Make Addition Associative Again!”, J. Demmel, J. Riedy, W. Ahrens, SIAM News, Oct 1, 2018
15. “A New IEEE 754 Standard for Floating-Point Arithmetic in an Ever-Changing World”, J. Demmel, J. Riedy, SIAM News, July 6, 2021
16. “Algorithms for Efficient Reproducible Floating Point Summation,” J. Demmel, W. Ahrens, J. D. Nguyen, ACM TOMS, v. 46, July 2020
17. “On floating point errors in Cholesky,” J. Demmel, LAPACK Working Note #14, 1989
18. “Accuracy and Stability of Numerical Algorithms”, N. Higham, 2002, SIAM

References related to Mixed Precision (5/5)

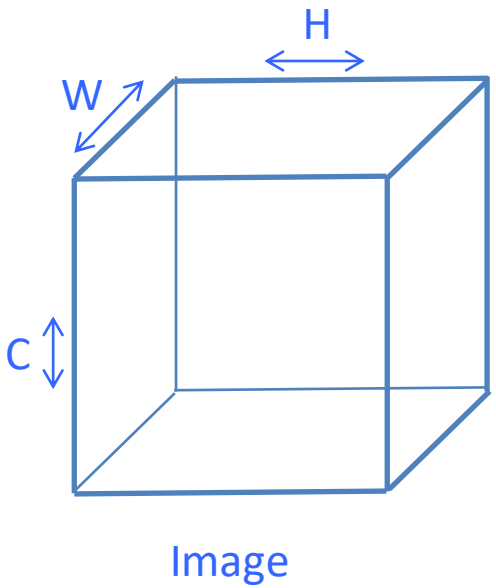
19. “Stability of block algorithms with fast level-3 BLAS”, J. Demmel, N. Higham, ACM TOMS, v. 18, n. 3, 1992
20. “Condition numbers and equilibration of matrices”, A. Van Der Sluis, Num. Math, v. 14, 1969
21. “Nearly optimal block Jacobi preconditioning”, J. Demmel, SIAM J. Mat. Anal. Appl., v. 44, 2023
22. “Proposed Consistent Exception Handling for the BLAS and LAPACK”, J. Demmel, J. Dongarra, M. Gates, G. Henry, J. Langou, X. S. Li, P. Luszczek, W. Pereira, J. Riedy, C. Rubio-Gonzalez; arxiv.org/abs/2207.09281, ~90 pages
23. “EXCVATE: Spoofing Exceptions and Solving Constraints to Test Exception Handling in Numerical Libraries,” J. Vanover, X. S. Li, J. Demmel, C. Rubio-Gonzalez, ARITH 2025, **Best Paper Award**
24. ”DGEMM on Integer Matrix Multiplication Unit”, H. Ootomo, K. Ozaki, R. Yokota, arxiv2306:11975v4, Mar 2024

Backup slides

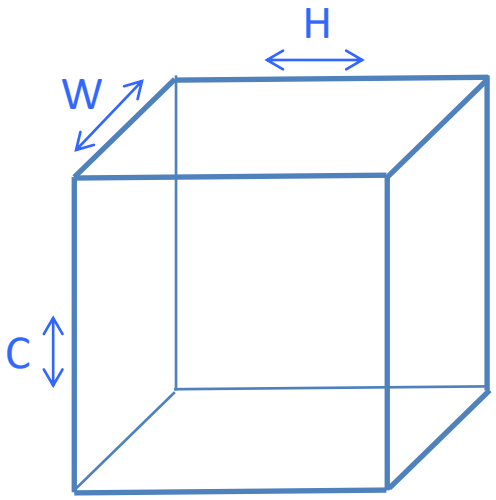
Outline

- Linear Algebra
  - Communication Lower Bounds for classical direct linear algebra
  - CA 2.5D Matmul
  - TSQR - Tall-Skinny QR
  - Iterative Methods for linear algebra
- Machine Learning
  - Training Neural Nets – "ImageNet training in minutes"
  - **Convolutional Neural Nets**
- And Beyond
  - Extending communication lower bounds and optimal algorithms to general loop nests

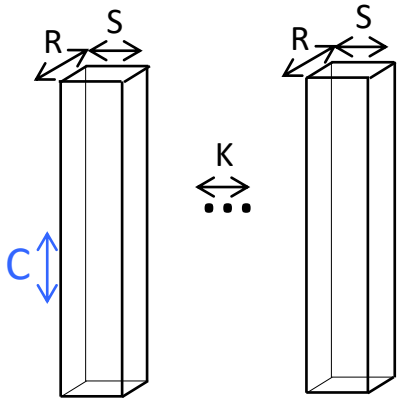
What CNNs compute



What CNNs compute

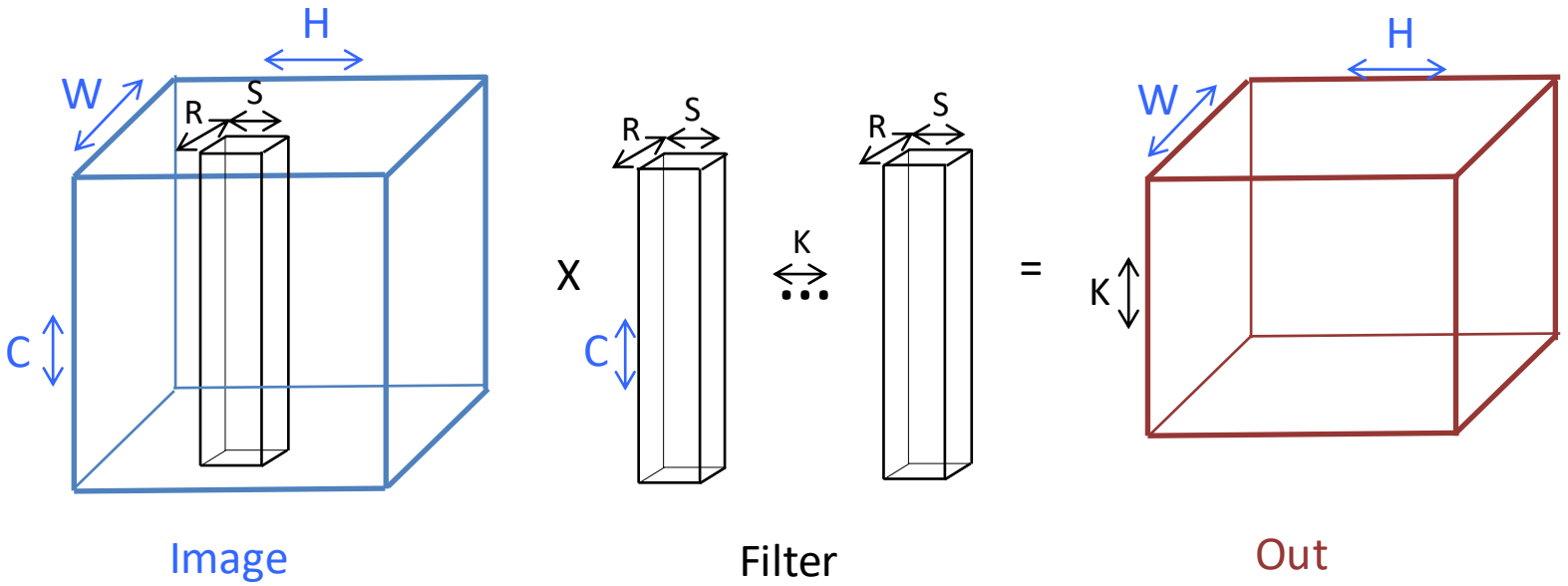


Image

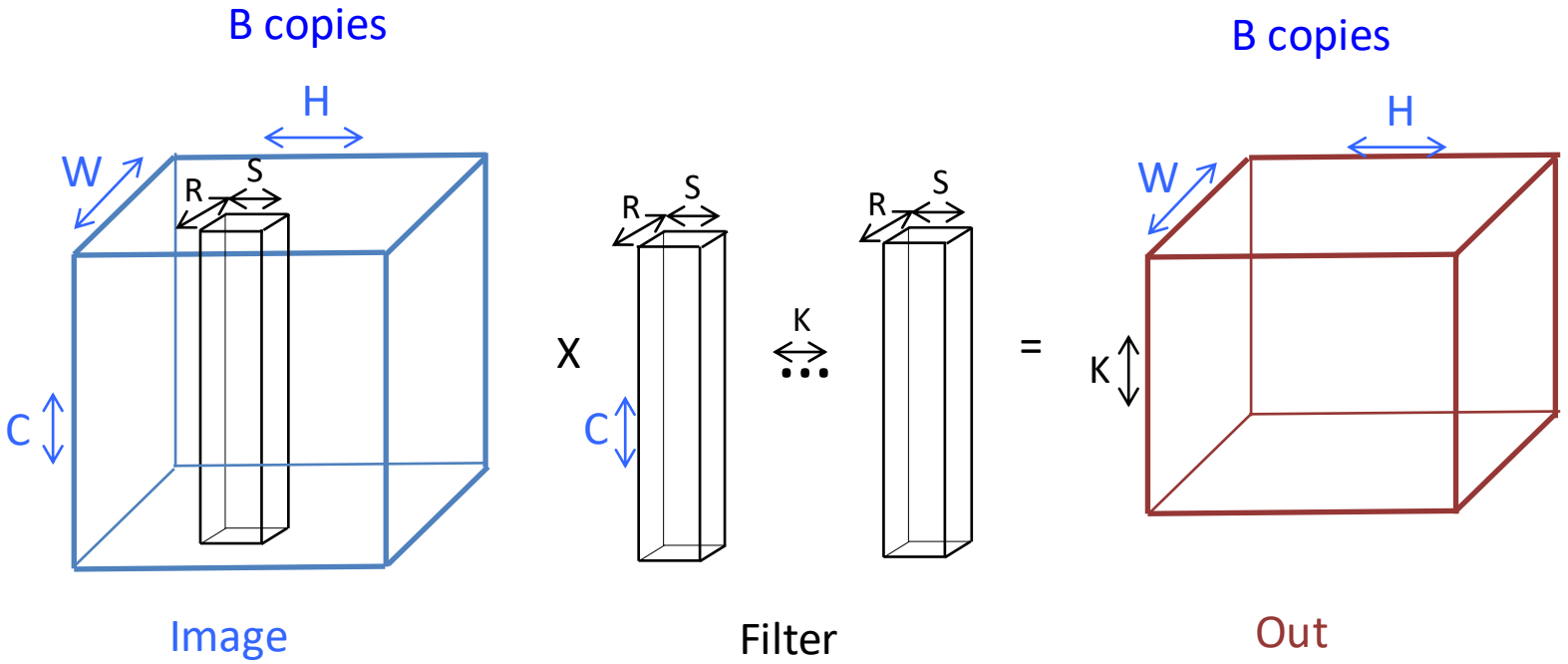


Filter

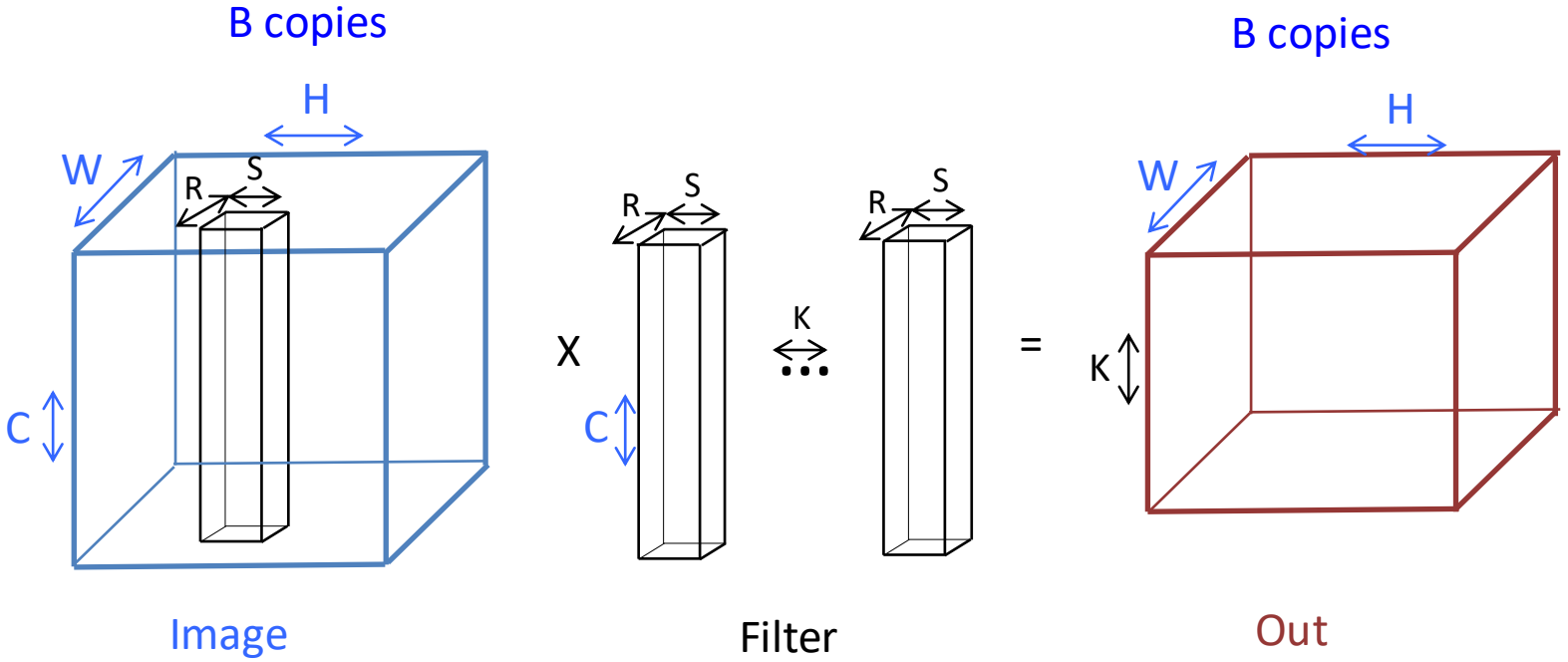
What CNNs compute



What CNNs compute



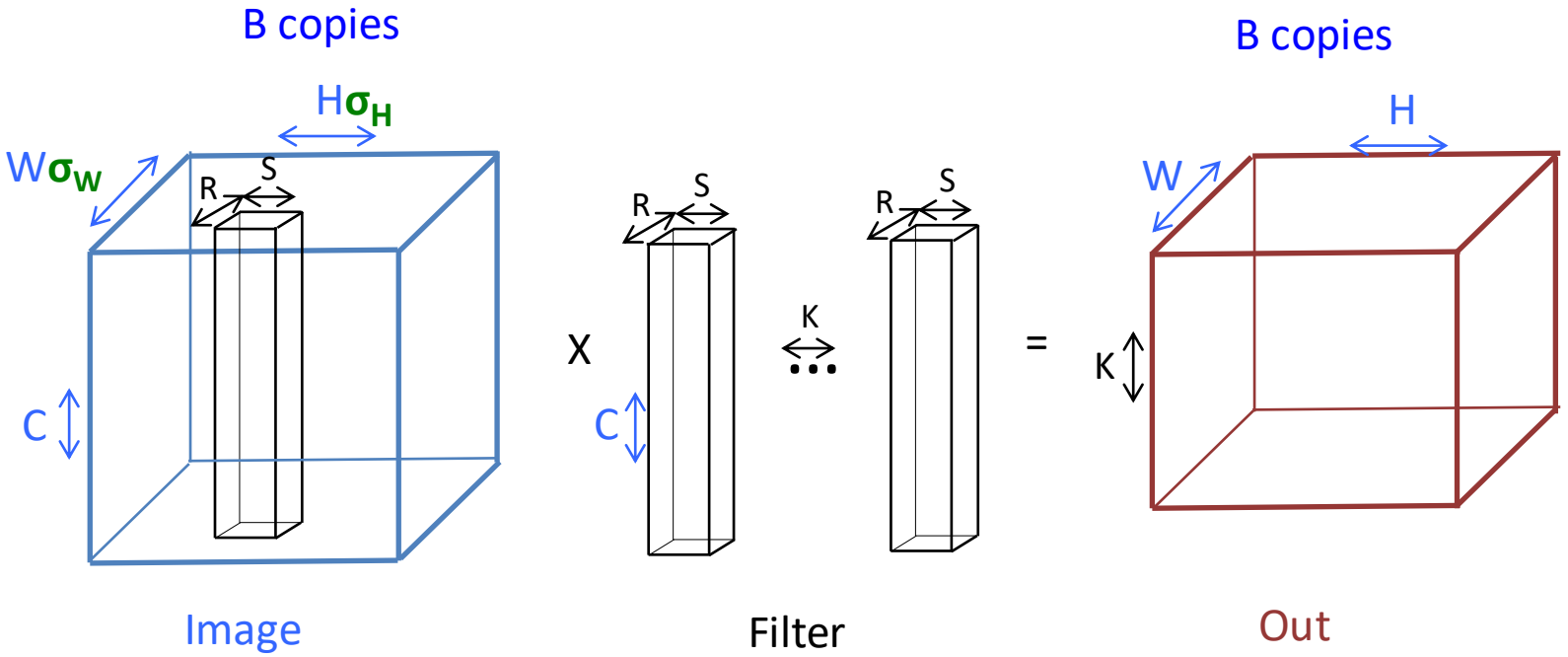
What CNNs compute



for k=1:K, for h=1:H, for w=1:W, for r=1:R,
for s=1:S, for c=1:C, for b=1:B

Out(k, h, w, b) += Image(r+w, s+h, c, b) * Filter(k, r, s, c)

What CNNs compute



for k=1:K, for h=1:H, for w=1:W, for r=1:R,
for s=1:S, for c=1:C, for b=1:B

Out(k, h, w, b) += Image(r+σ_Ww, s+σ_Hh, c, b) * Filter(k, r, s, c)

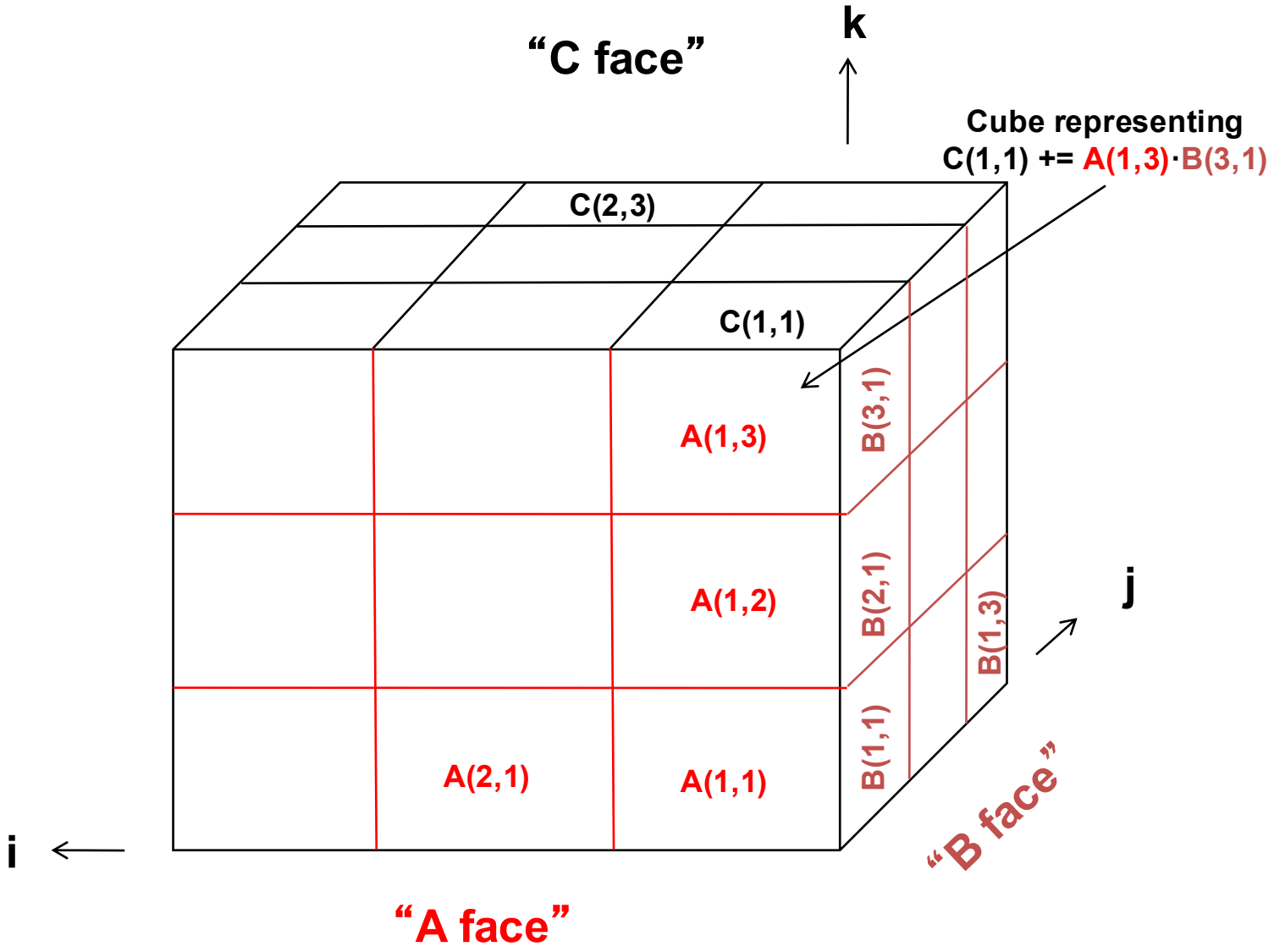
Communication Lower Bound for CNNs

- Let N = #iterations = KHWRSCB, M = cache size
- #words moved = Ω(max(… 5 terms
  - BKHW, … size of Out
  - $\sigma_H\sigma_W$BCWH, … size of Image
  - CKRS, … size of Filter
  - N/M, … lower bound for large loop bounds
  - $N/(M^{1/2} (RS/(\sigma_H\sigma_W))^{1/2})$ ... lower bound for small filters)
- Any one of 5 terms may be largest
- Bottommost bound beats matmul by factor $(RS/(\sigma_H\sigma_W))^{1/2}$
  - Applies in common case when data does not fit in cache, but one RxS filter does
  - Tile needed to attain N/M too big to fit in loop bounds
- Thm: Always attainable! (computer generated proof)
  - Beats im2col in data movement for various practical sizes
- Improved constants in PASC‘22
- Chen/Han/Wang (arxiv:1911.05662v3): HW accelerator

Proof of Communication Lower Bound on C = A·B (1/4)

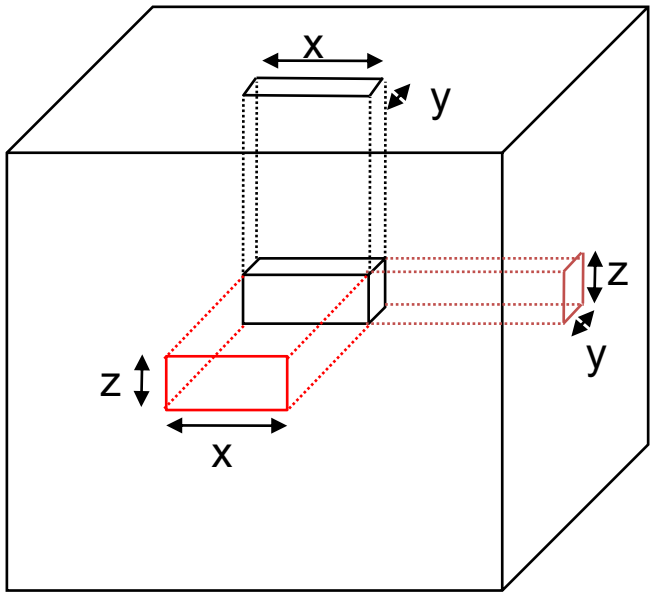
- Basic Counting Argument:
  - Only M entries of A, B and C are available in cache
  - Find an upper bound F on the number of different iterations C(i,j) = C(i,j) + A(i,k)*B(k,j) we can perform
  - Need to refill cache n^3/F times to complete algorithm
  - Need to read/write at least $M\ n^3/\ F$ words to/from cache
- Represent iterations and data geometrically

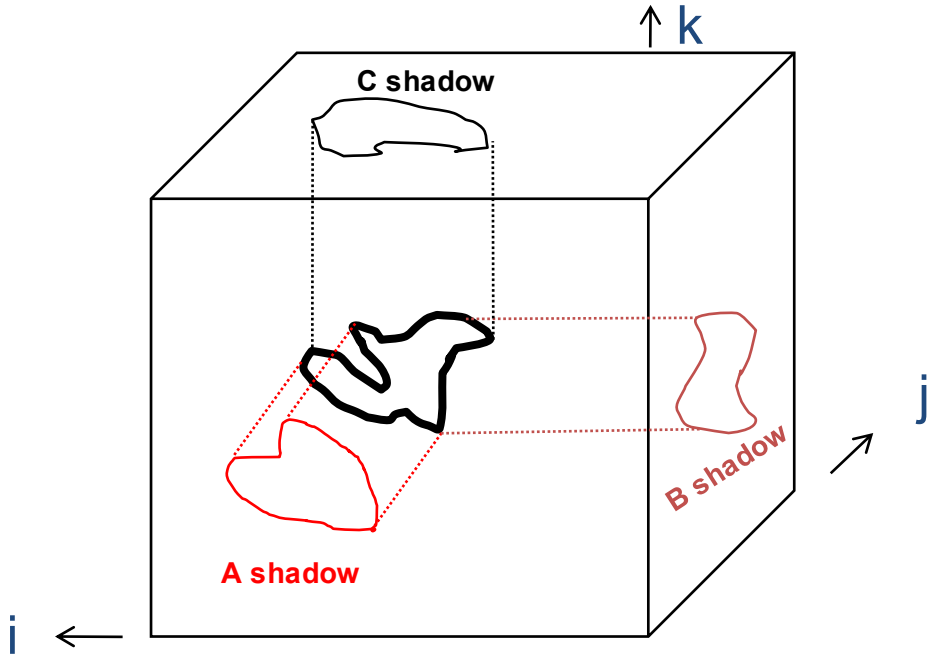
Proof of Communication Lower Bound on C = A·B (2/4)



- If we have at most M "A squares", "B squares", and "C squares" on faces, how many cubes can we have?

Proof of Communication Lower Bound on C = A·B (3/4)





 # cubes in black box with side lengths x, y and z
= Volume of black box
= x·y·z
= (xz · zy · yx)$^{1/2}$
= (#A□s · #B□s · #C□s)$^{1/2}$

(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)
 # cubes in 3D set = Volume of 3D set
≤ (area(A shadow) · area(B shadow) · area(C shadow))$^{1/2}$

Proof of Communication Lower Bound on C = A·B (4/4)

- # loop iterations doable with M words of data = #cubes

  $\leq$ (area(A shadow) · area(B shadow) ·area(C shadow)) $^{1/2}$

  $\leq (M \cdot M \cdot M)^{1/2} = M^{3/2} = F$

- Need to read/write at least $M\, n^3/F = \Omega(n^3/M^{1/2}) = \Omega(\#\text{loop iterations} / M^{1/2})$ words to/from cache