Communication-Avoiding Algorithms for Linear Algebra, ML and Beyond

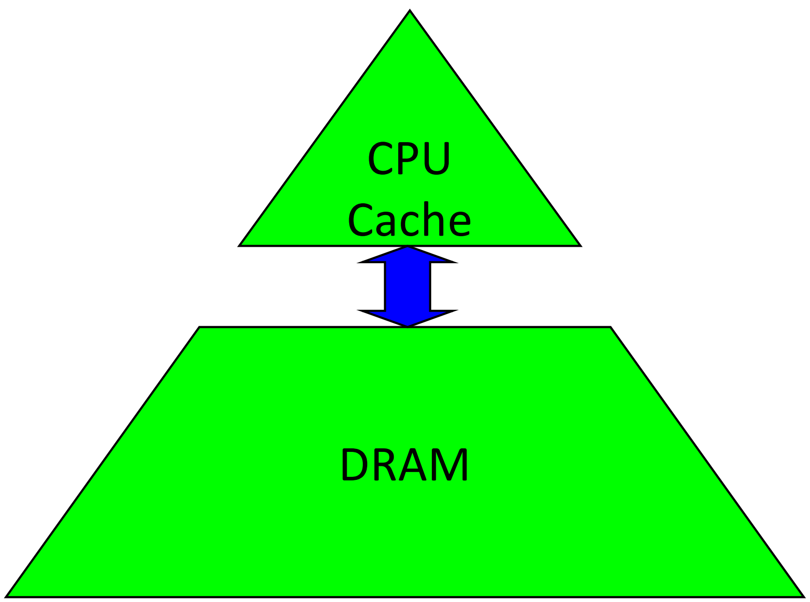
Jim Demmel, EECS & Math Depts., UC Berkeley

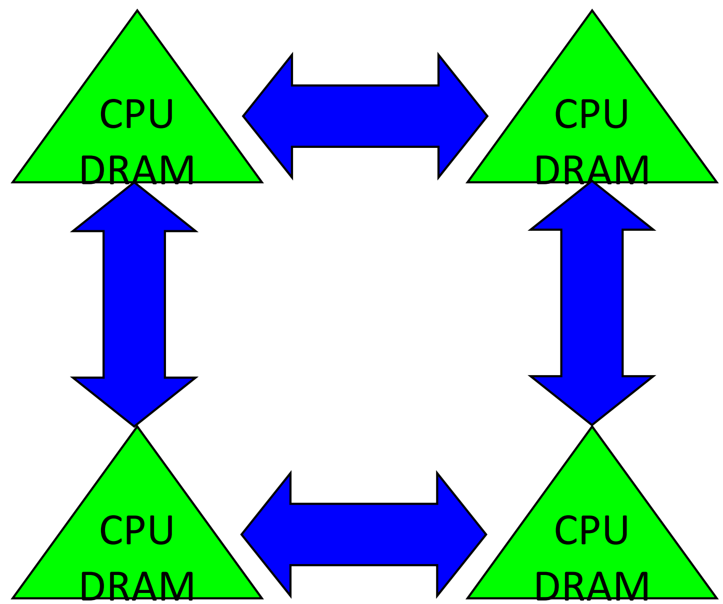
And many, many others …

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).



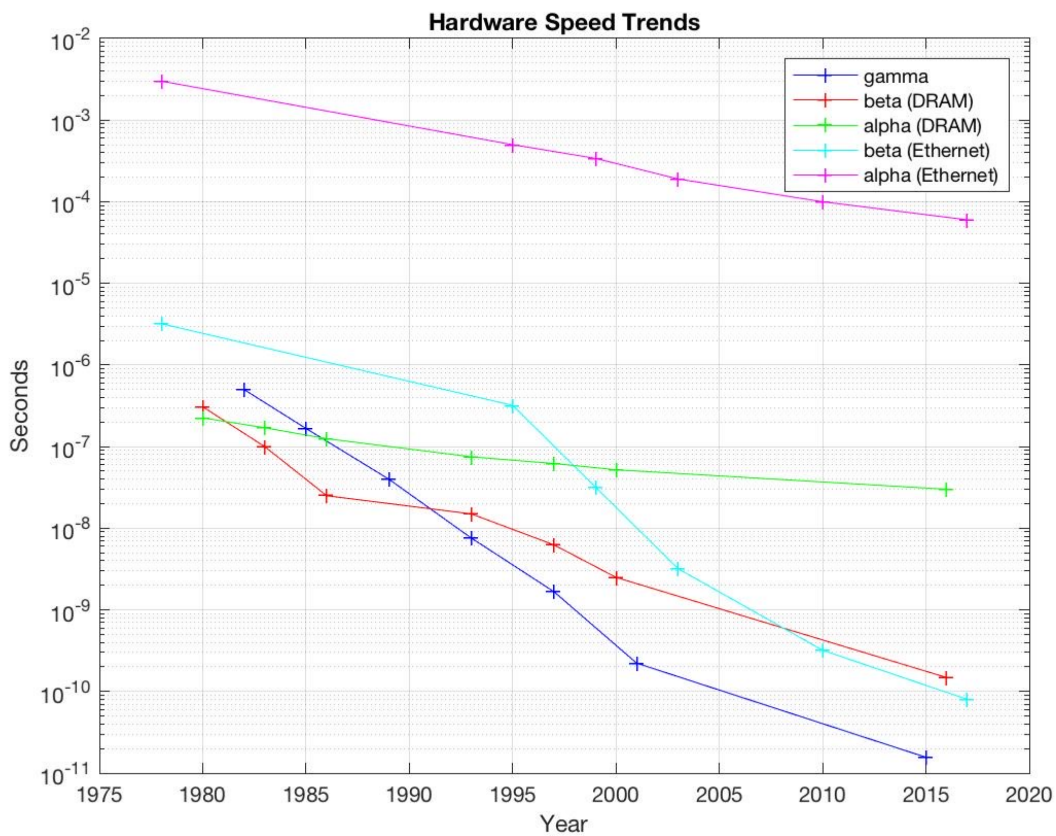


Why avoid communication? (2/3)

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

  (the last two terms: communication)

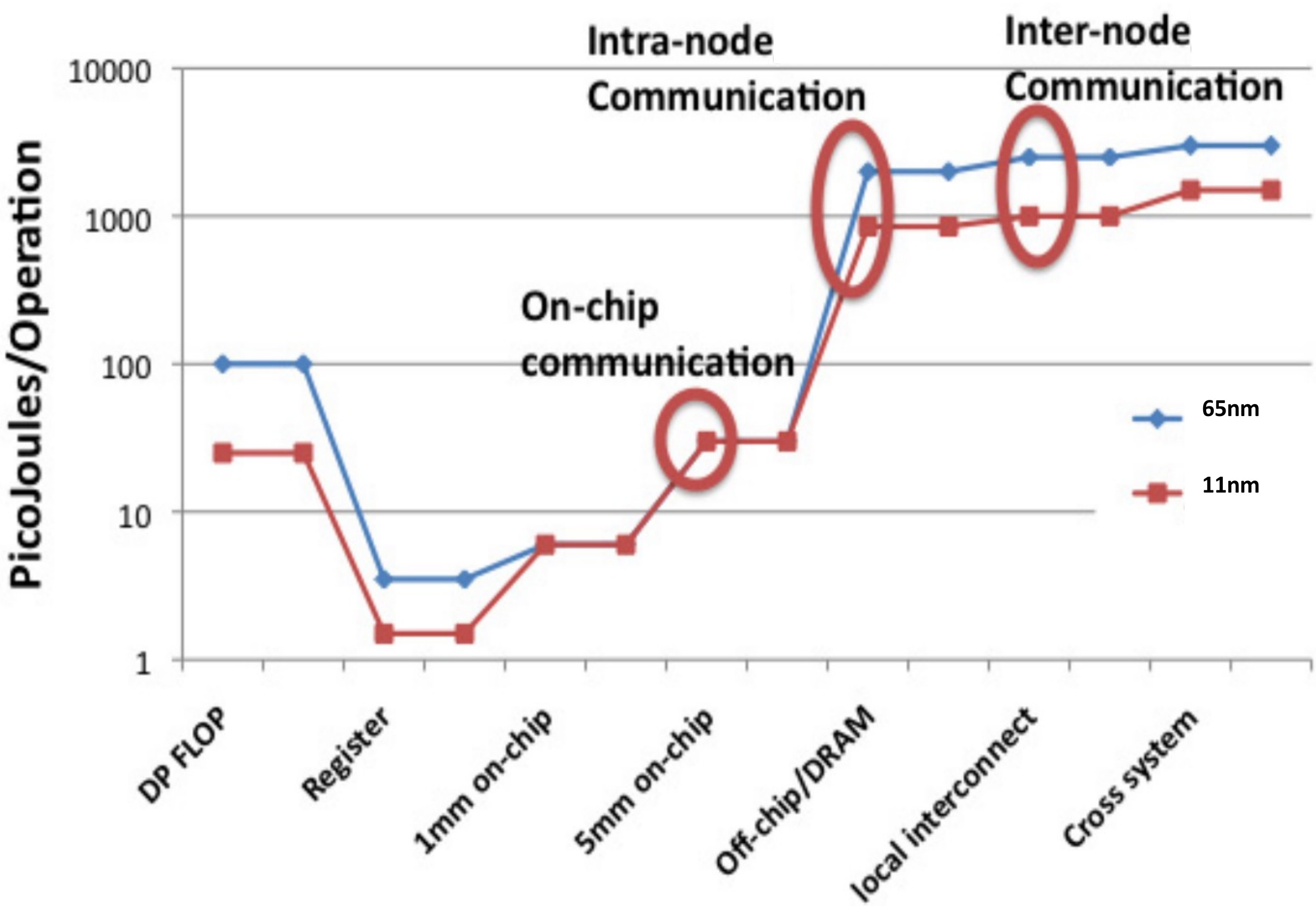
- Time_per_flop (γ) << 1/ bandwidth (β) << latency (α)



Data from Patterson & Hennessey, 2019

Why avoid communication? (3/3)

Same story for saving energy



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

- 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

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

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, …
  - 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

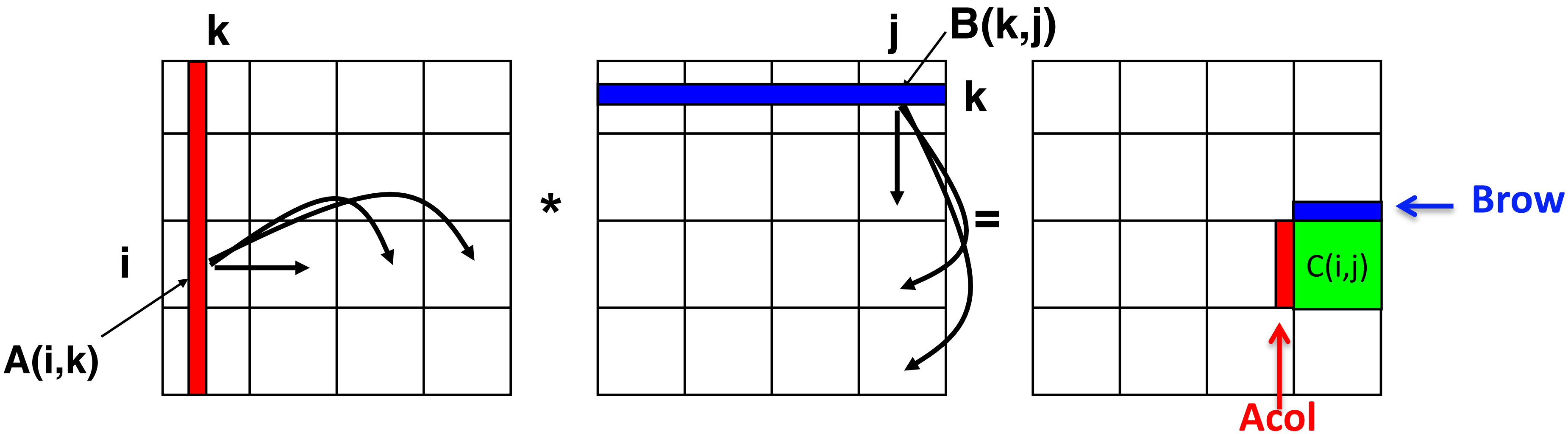
(Ballard, D., Holtz, Schwartz)

Outline

- 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

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

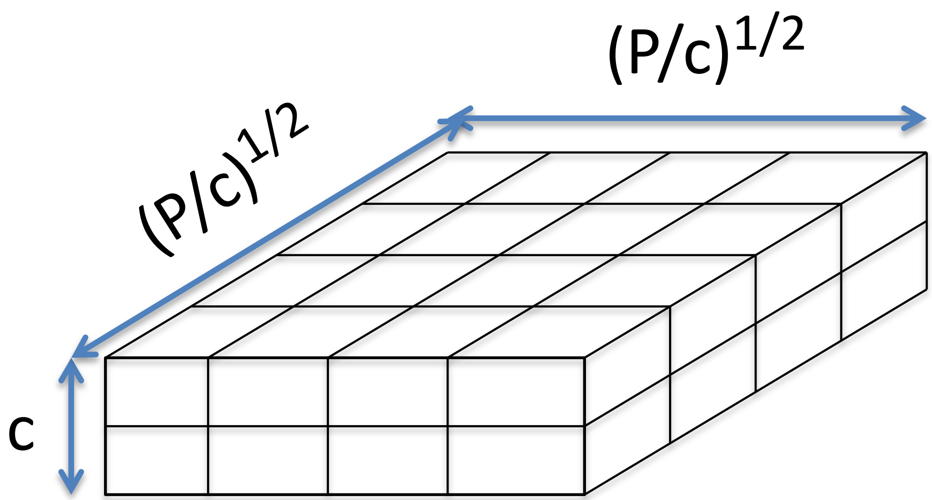
Can we do Better?

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

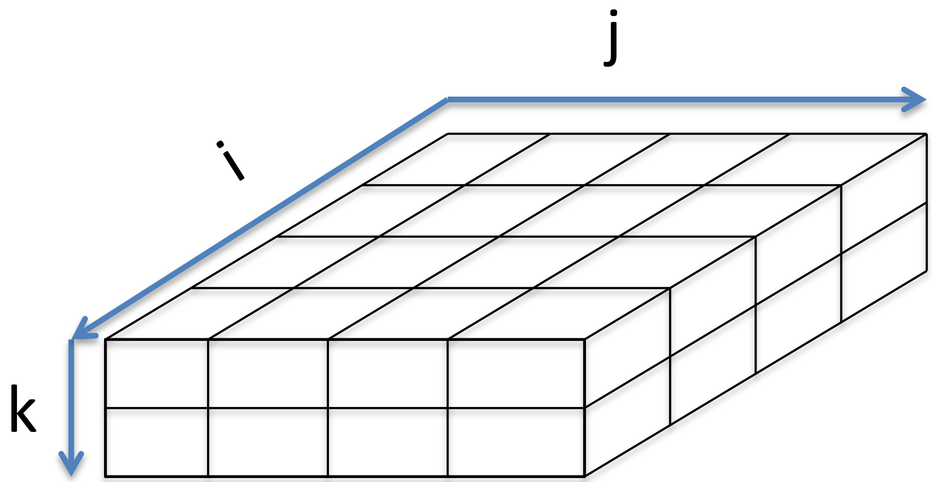
- Assume can fit cn^2/P data per processor, $c > 1$
- Processors form $(P/c)^{1/2}$ x $(P/c)^{1/2}$ x c grid



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

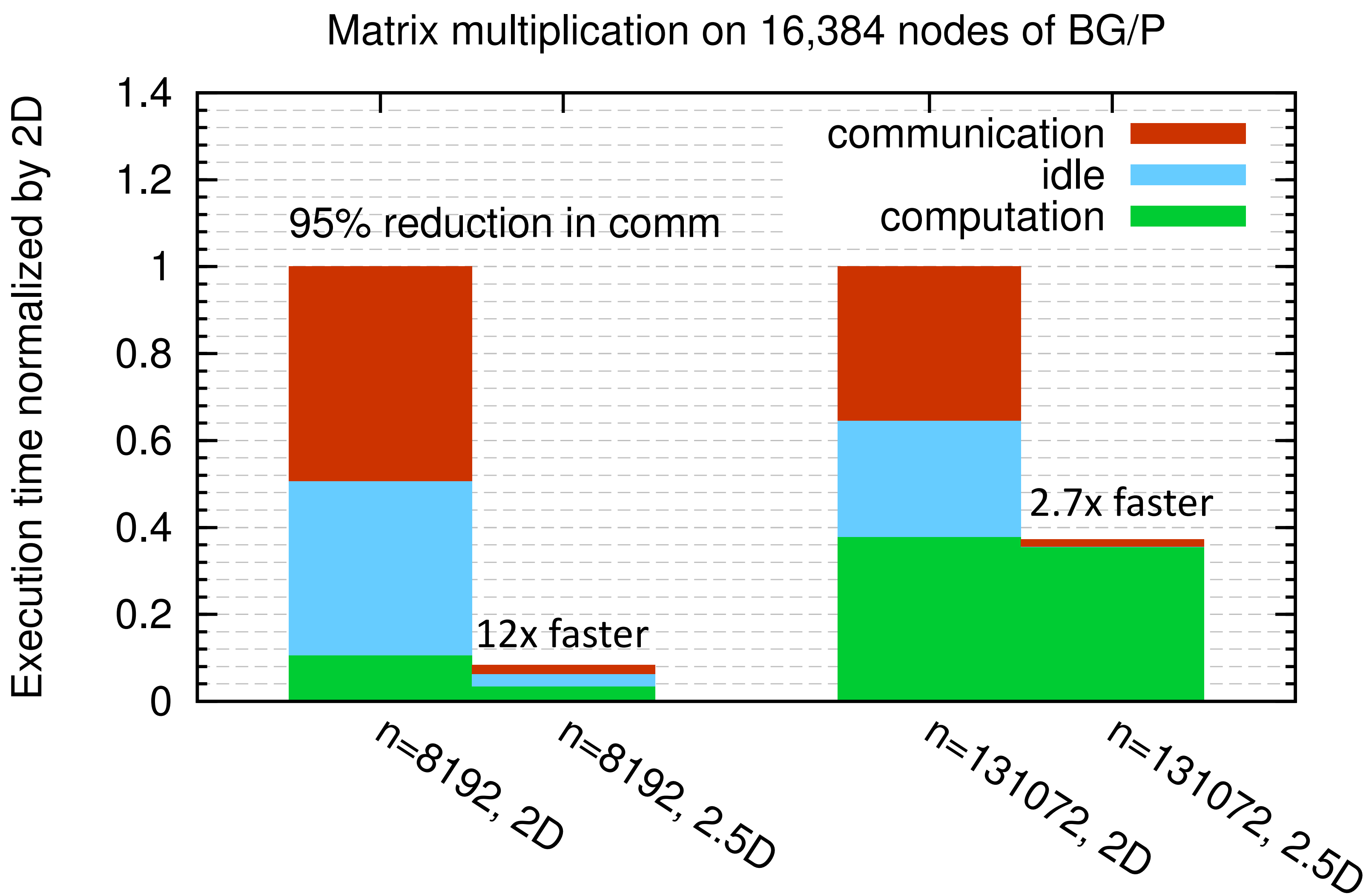


Initially P(i,j,0) owns A(i,j) and B(i,j) each of size $n(c/P)^{1/2}$ x $n(c/P)^{1/2}$

(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)

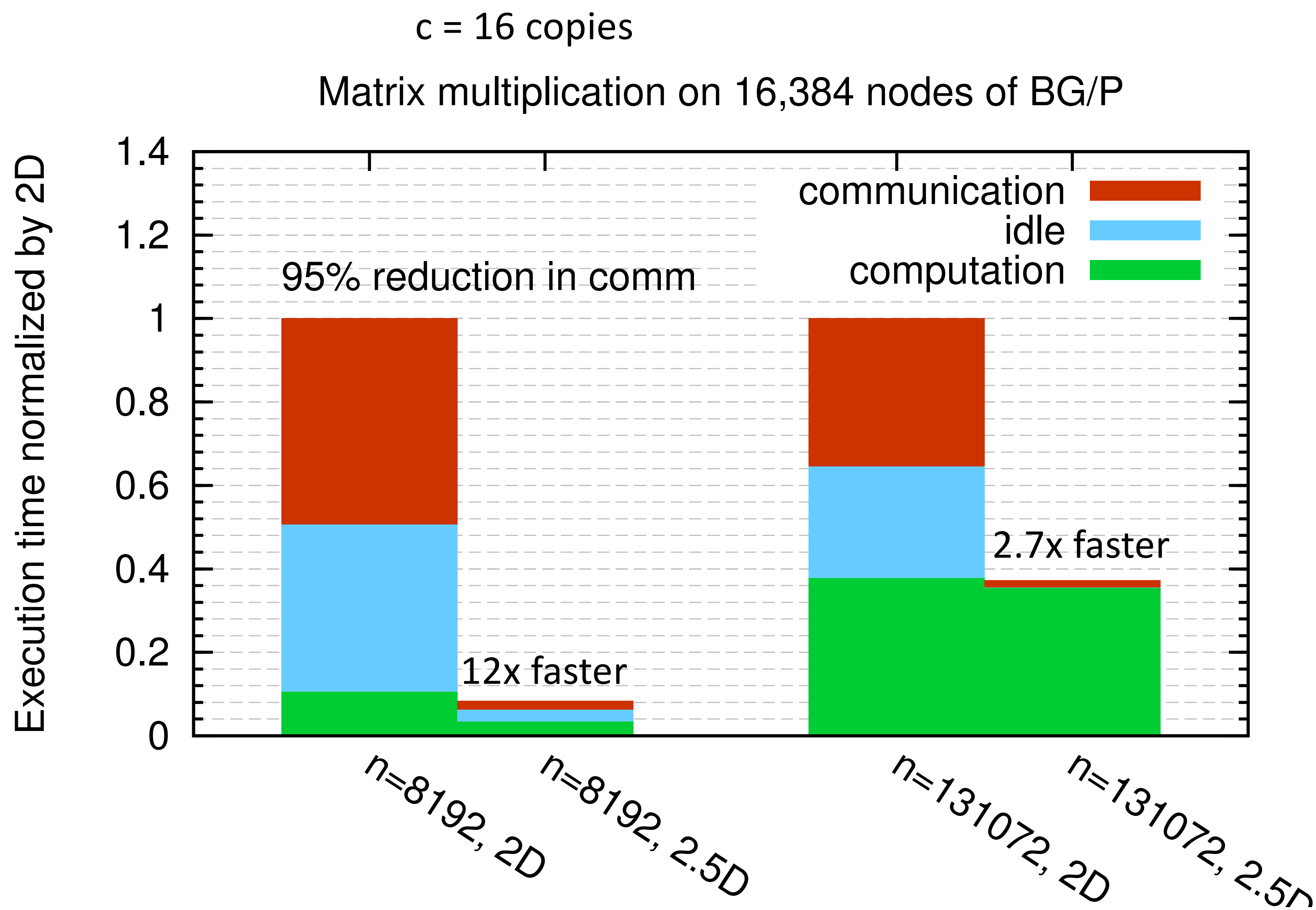
2.5D Matmul on BG/P, 16K nodes / 64K cores

c = 16 copies



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

- 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

TSQR: QR of a Tall, Skinny matrix

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

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

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

TSQR: QR of a Tall, Skinny matrix

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

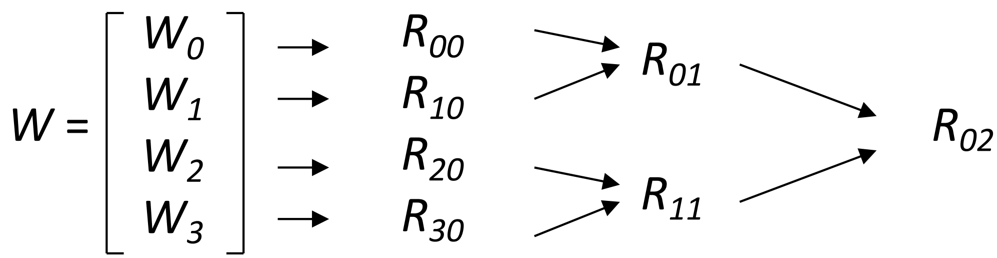
$$\begin{pmatrix} R_{00} \\ R_{10} \\ \hline R_{20} \\ R_{30} \end{pmatrix} = \begin{pmatrix} Q_{01} R_{01} \\ \hline Q_{11} R_{11} \end{pmatrix} = \begin{pmatrix} Q_{01} & \\ \hline & Q_{11} \end{pmatrix} \cdot \begin{pmatrix} R_{01} \\ \hline R_{11} \end{pmatrix}$$

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

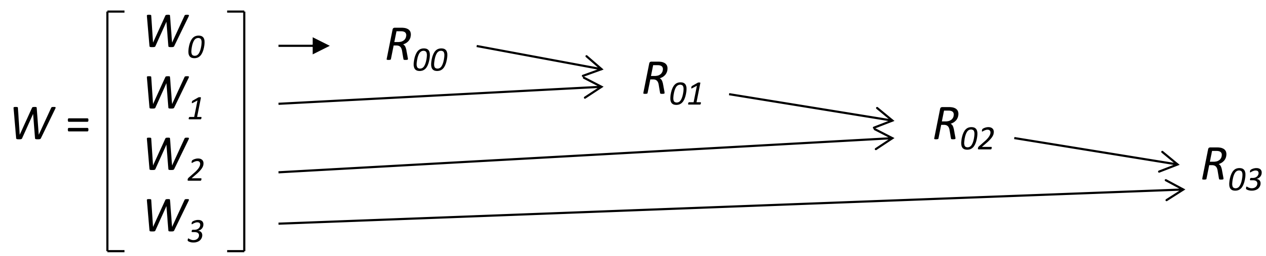
Output = $\{ Q_{00}, Q_{10}, Q_{20}, Q_{30}, Q_{01}, Q_{11}, Q_{02}, R_{02} \}$

TSQR: An Architecture-Dependent Algorithm

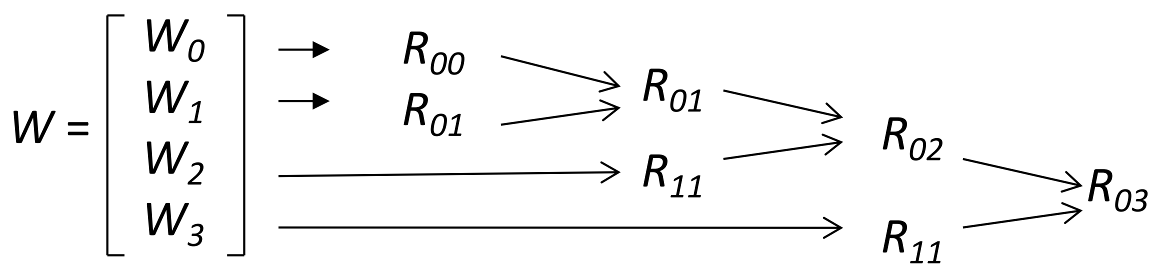
Parallel:



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

- 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

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

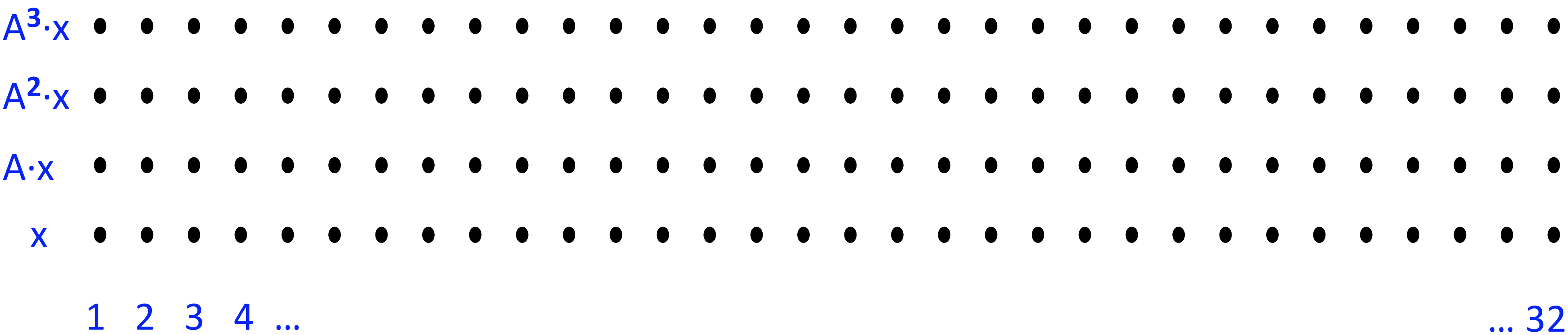
```
W = [ v, Av, A^2v, … , A^kv ]
[Q,R] = TSQR(W)
     … "Tall Skinny QR"
build H from R
solve LSQ problem with H
```

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]
- Up to **2.3x** speedup for GMRES on 8 core Intel Clovertown (Hoemmen)
- Up to **4.2x** speedup for BiCGStab on 24K core Cray XE6 (Carson)

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

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

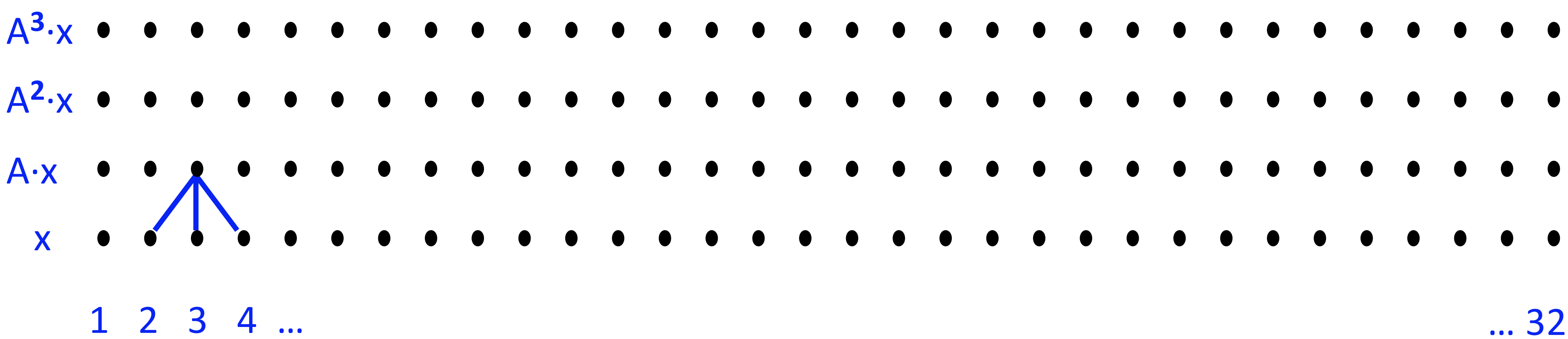


- Example: A tridiagonal, n=32, k=3
- Works for any "well-partitioned" A

Communication Avoiding Kernels:

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

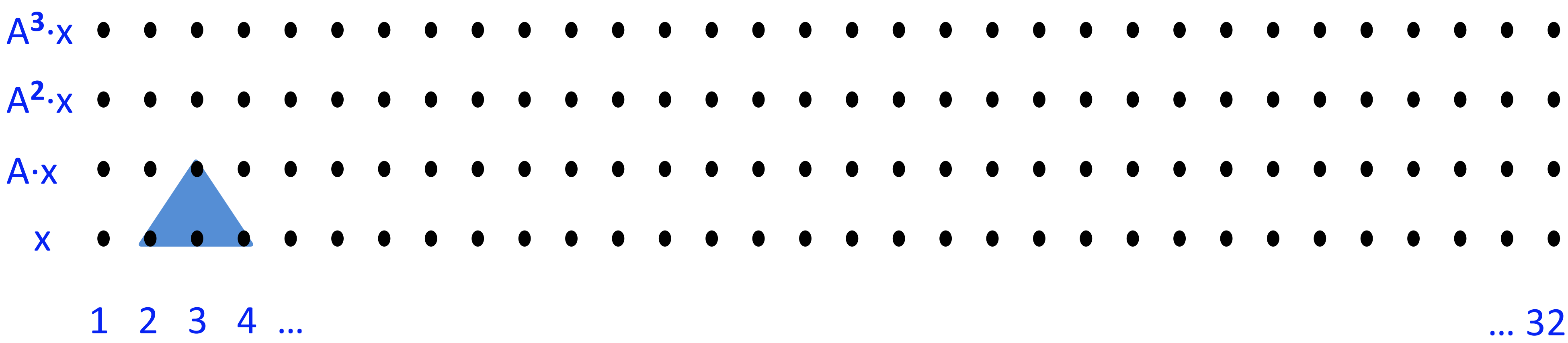
- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \ldots, A^kx]$



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

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

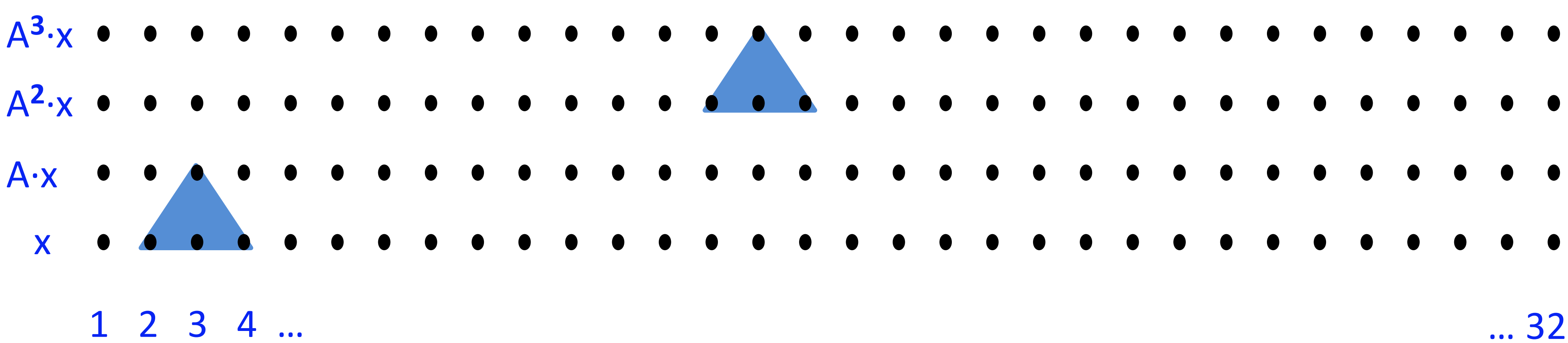
- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \ldots, A^kx]$



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

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

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

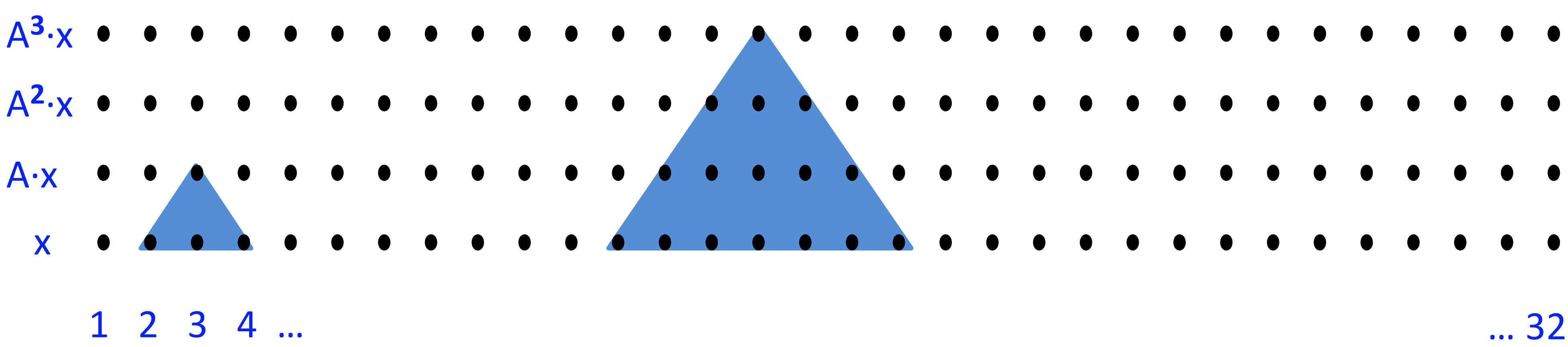


- Example: A tridiagonal, n=32, k=3

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]

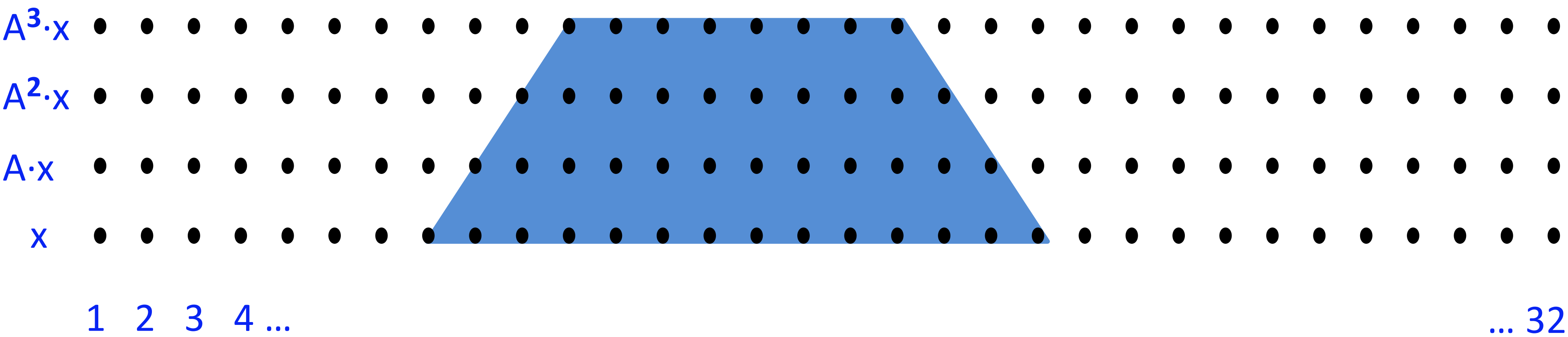


- Example: A tridiagonal, n=32, k=3

Communication Avoiding Kernels:

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

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

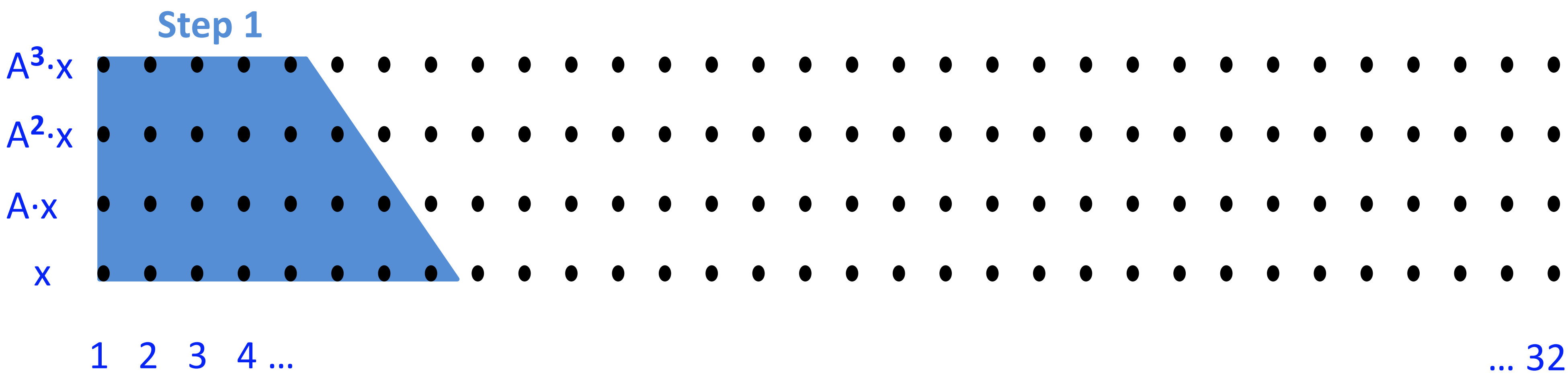


- Example: A tridiagonal, n=32, k=3

Communication Avoiding Kernels:

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

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

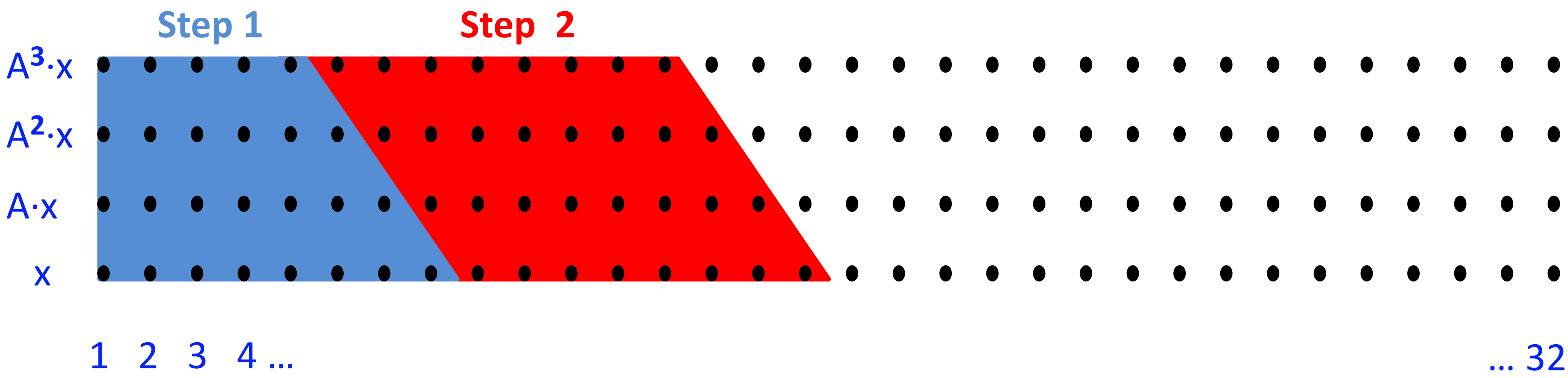


- Example: A tridiagonal, n=32, k=3

Communication Avoiding Kernels:

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

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

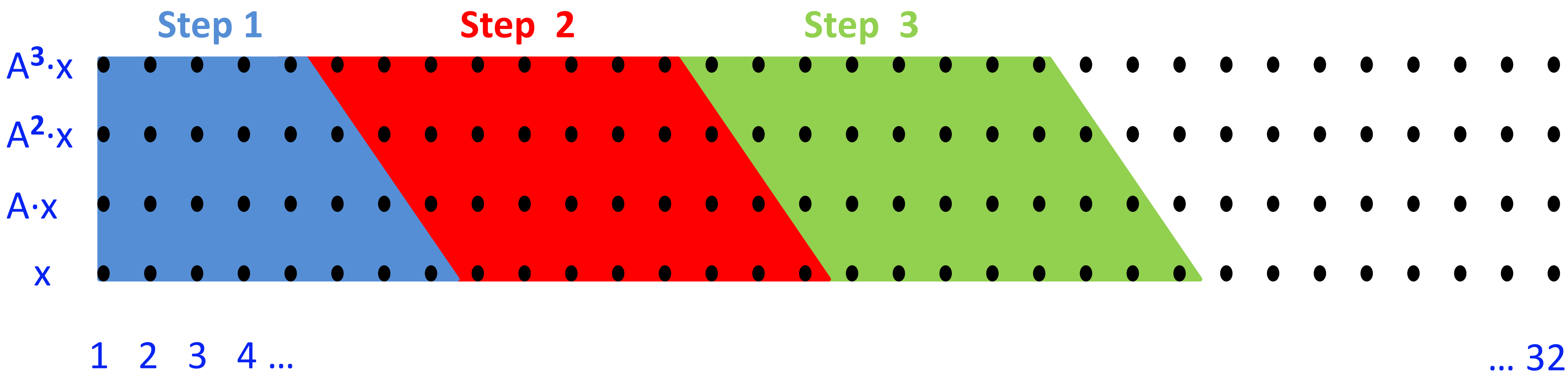


- Example: A tridiagonal, n=32, k=3

Communication Avoiding Kernels:

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

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

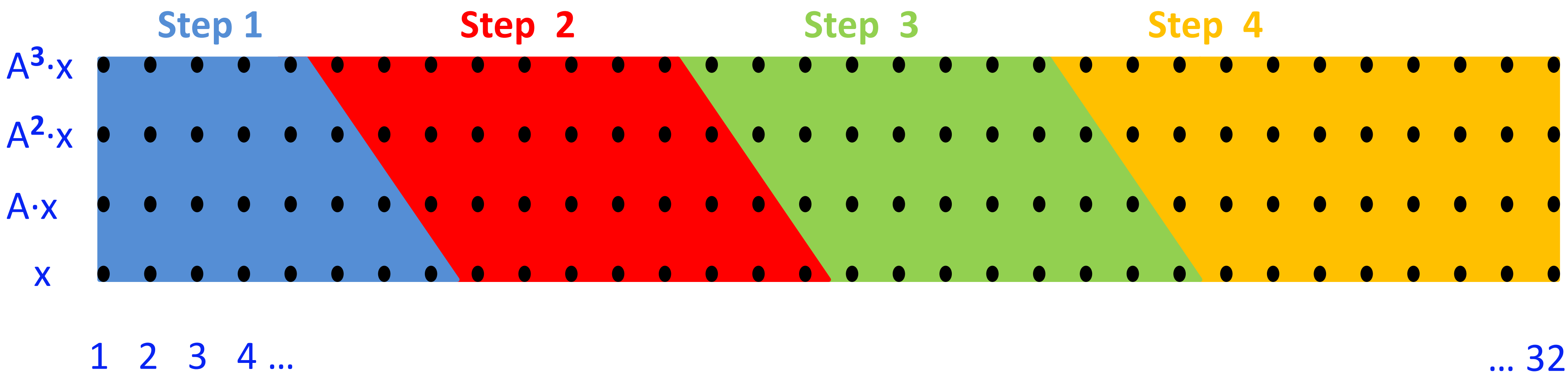


- Example: A tridiagonal, n=32, k=3

Communication Avoiding Kernels:

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

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

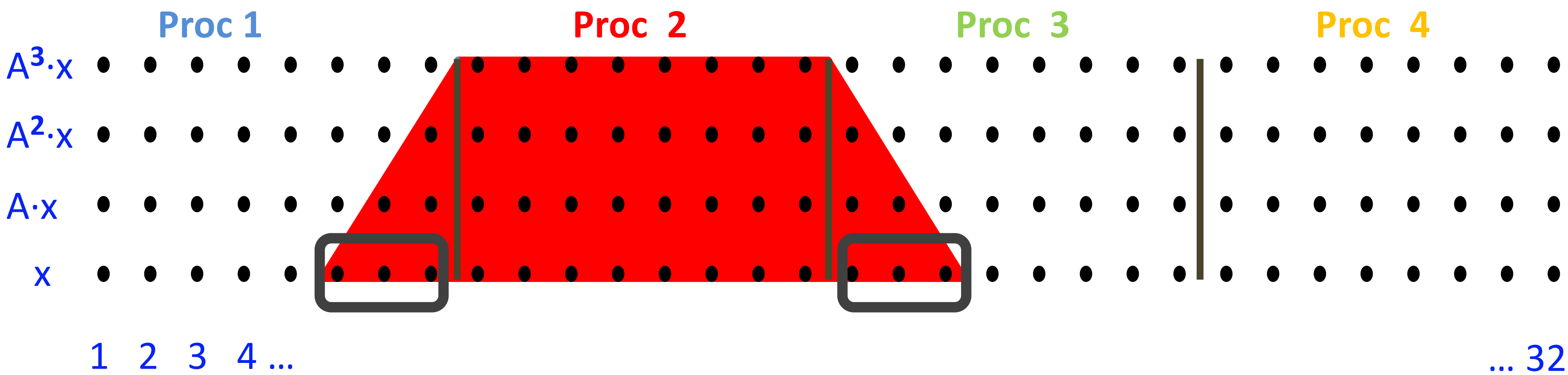


- Example: A tridiagonal, n=32, k=3

Communication Avoiding Kernels:

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

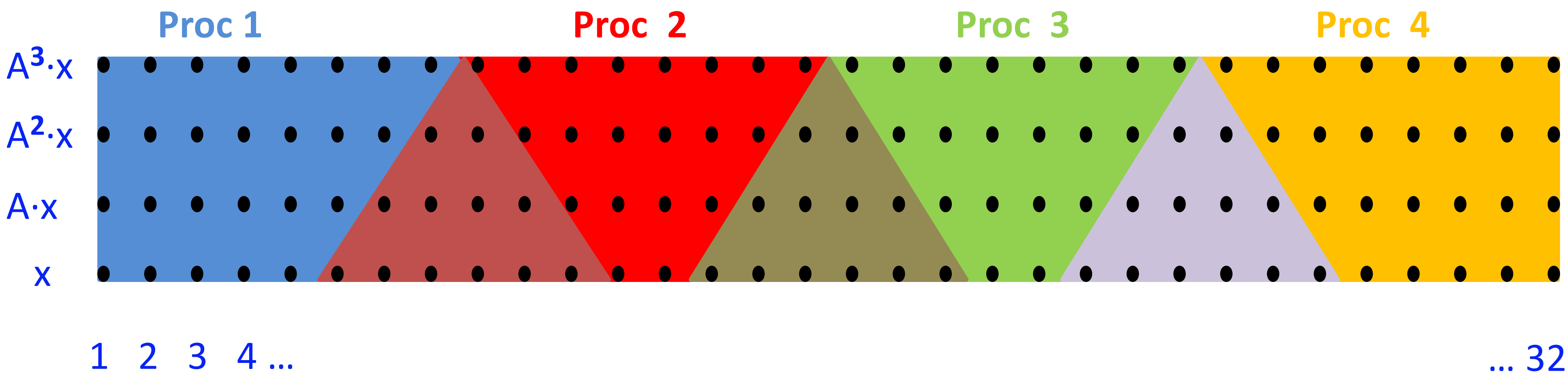
- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, ..., A^kx]$
- Parallel Algorithm



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

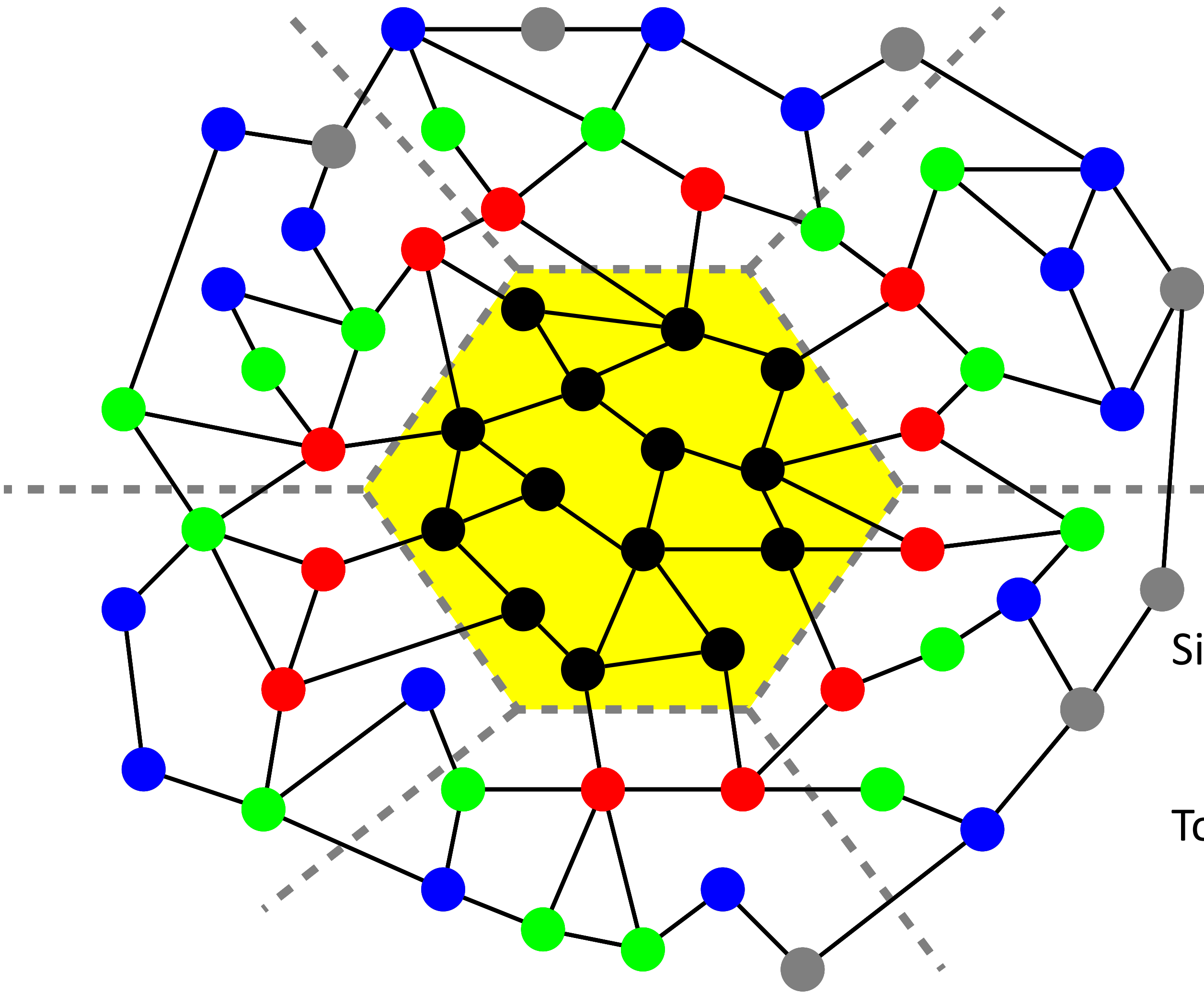
Communication Avoiding Kernels:
The Matrix Powers Kernel : $[Ax, A^2x, \ldots, A^kx]$

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



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

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

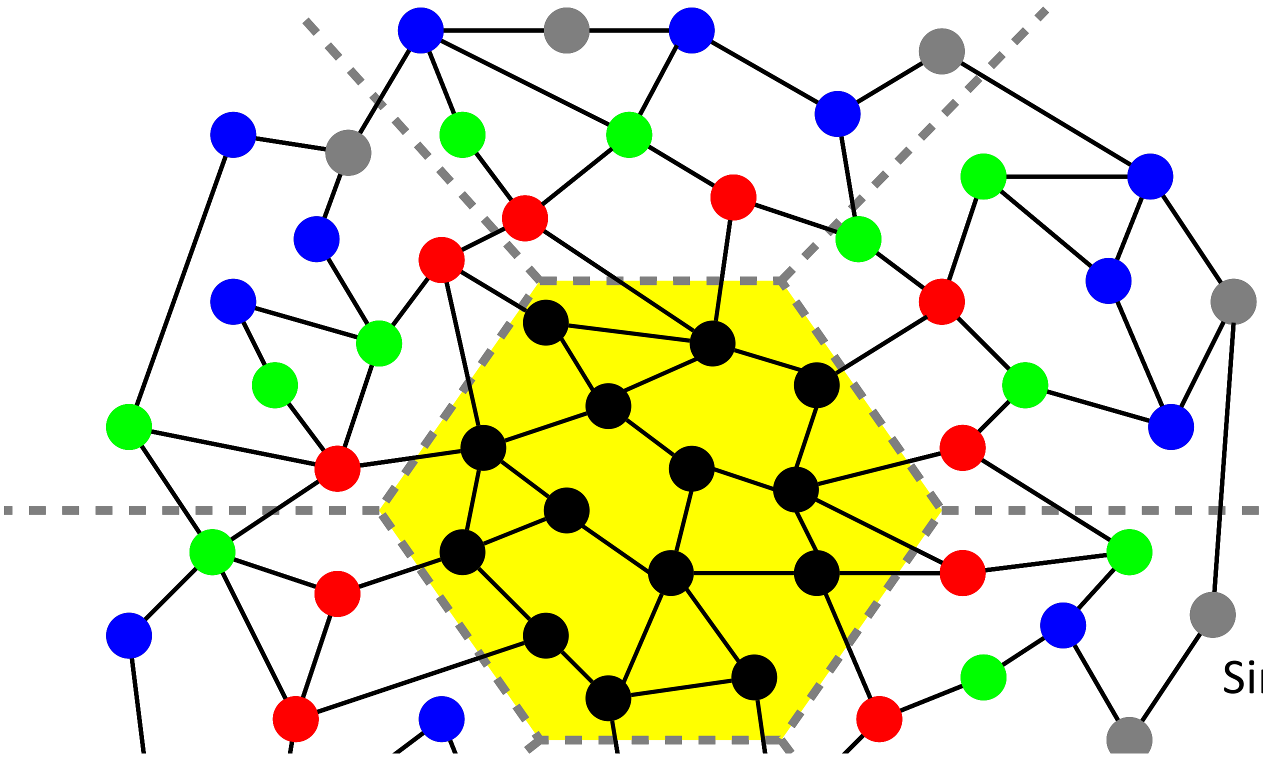


Simple block-row partitioning → (hyper)graph partitioning

Top-to-bottom processing → Traveling Salesman Problem

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

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

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

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

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

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

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. $\alpha_j := (r_j, r_0^*)/(Ap_j, r_0^*)$
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. $\beta_j := \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \times \frac{\alpha_j}{\omega_j}$
10. $p_{j+1} := r_{j+1} + \beta_j(p_j - \omega_j Ap_j)$
11. EndDo

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

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

- 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 (45h → 20m)

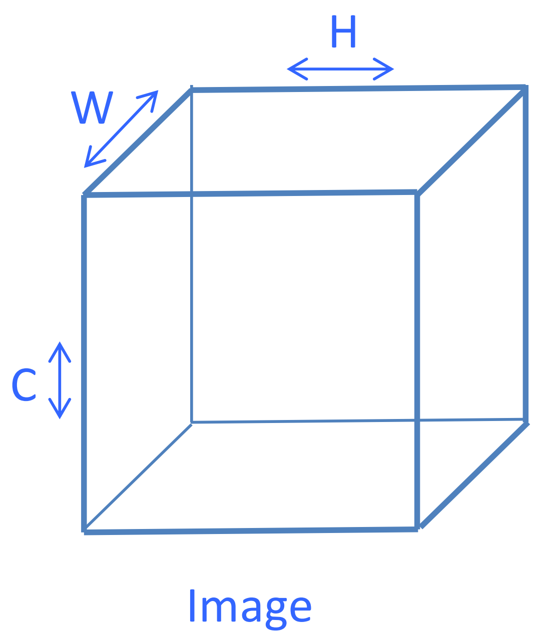
ImageNet Training in Minutes

- 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

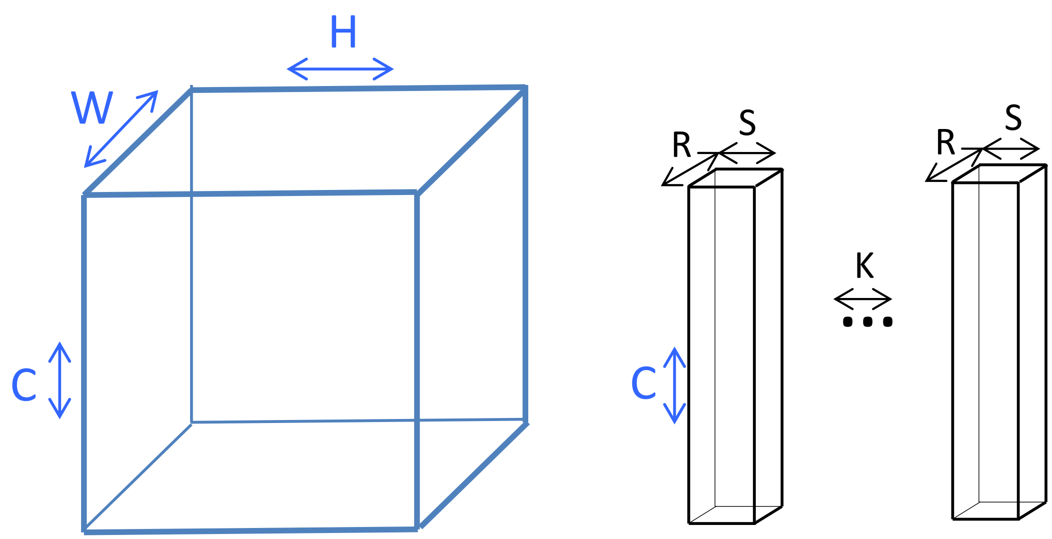
Outline

- 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



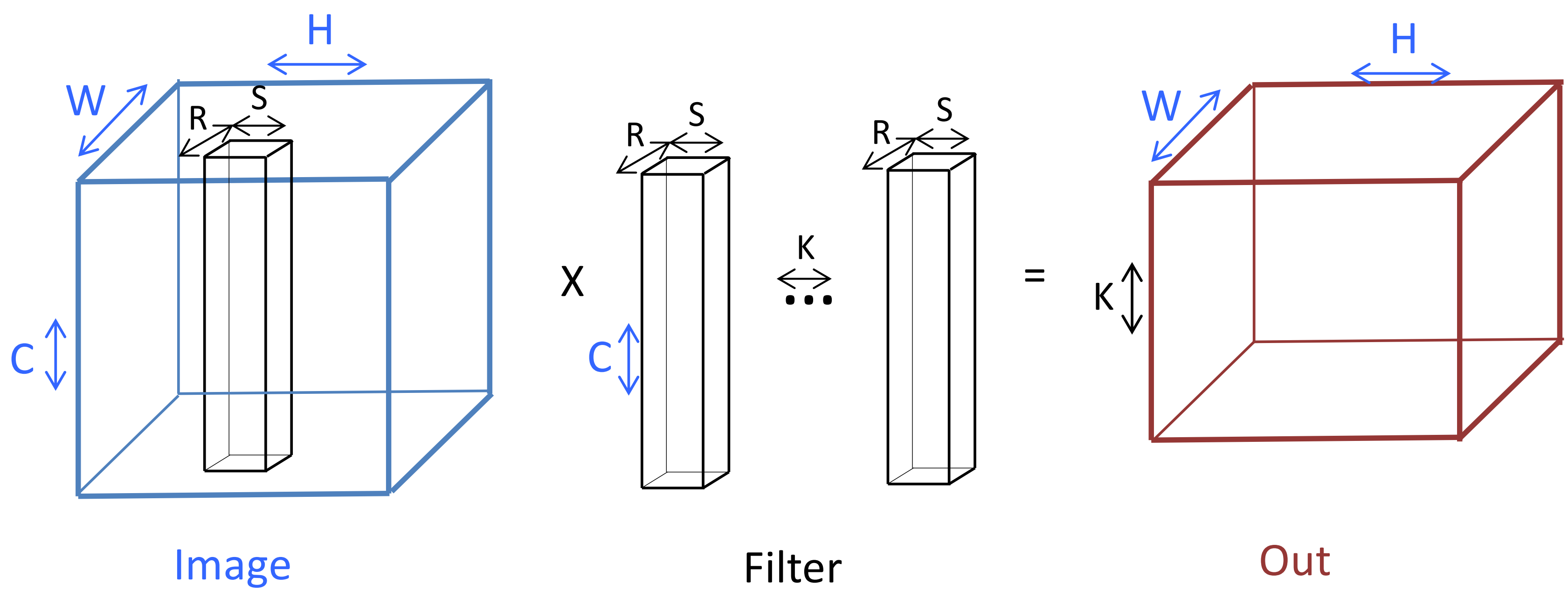
What CNNs compute



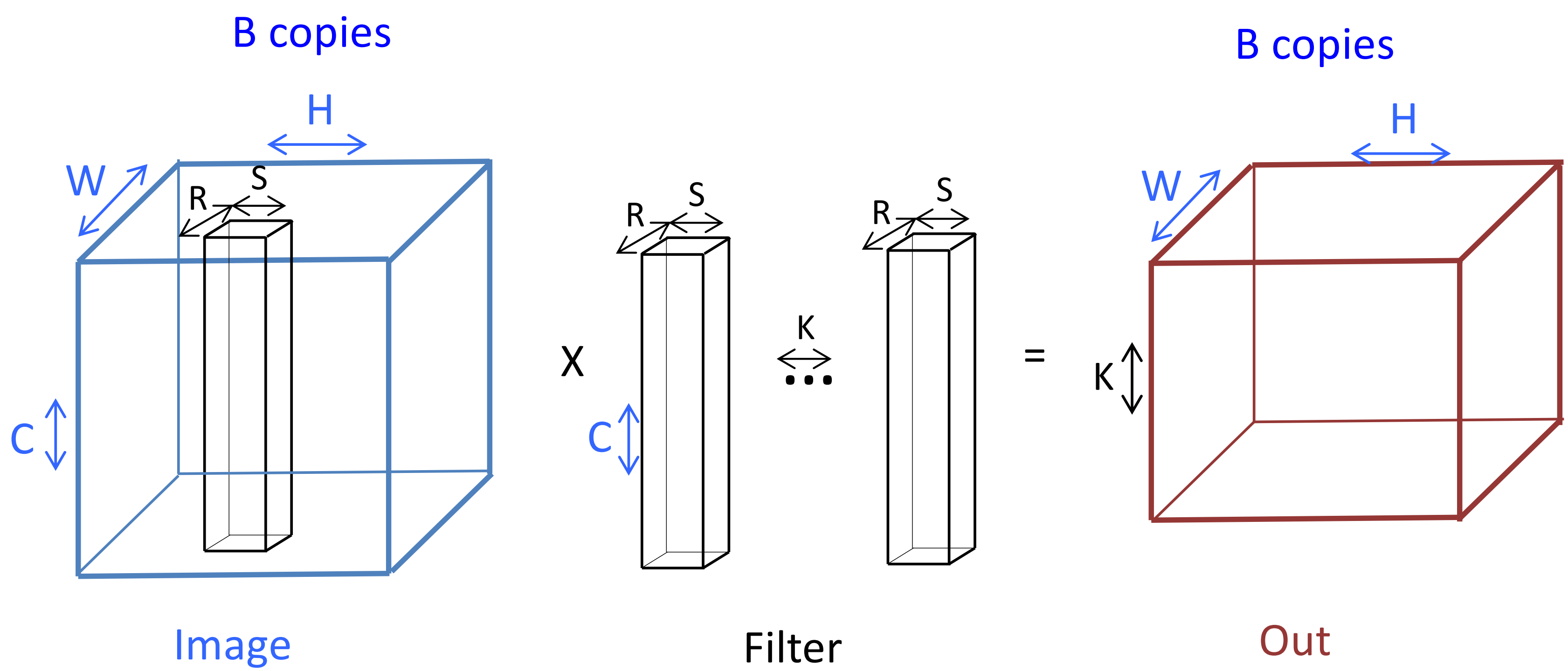
Image

Filter

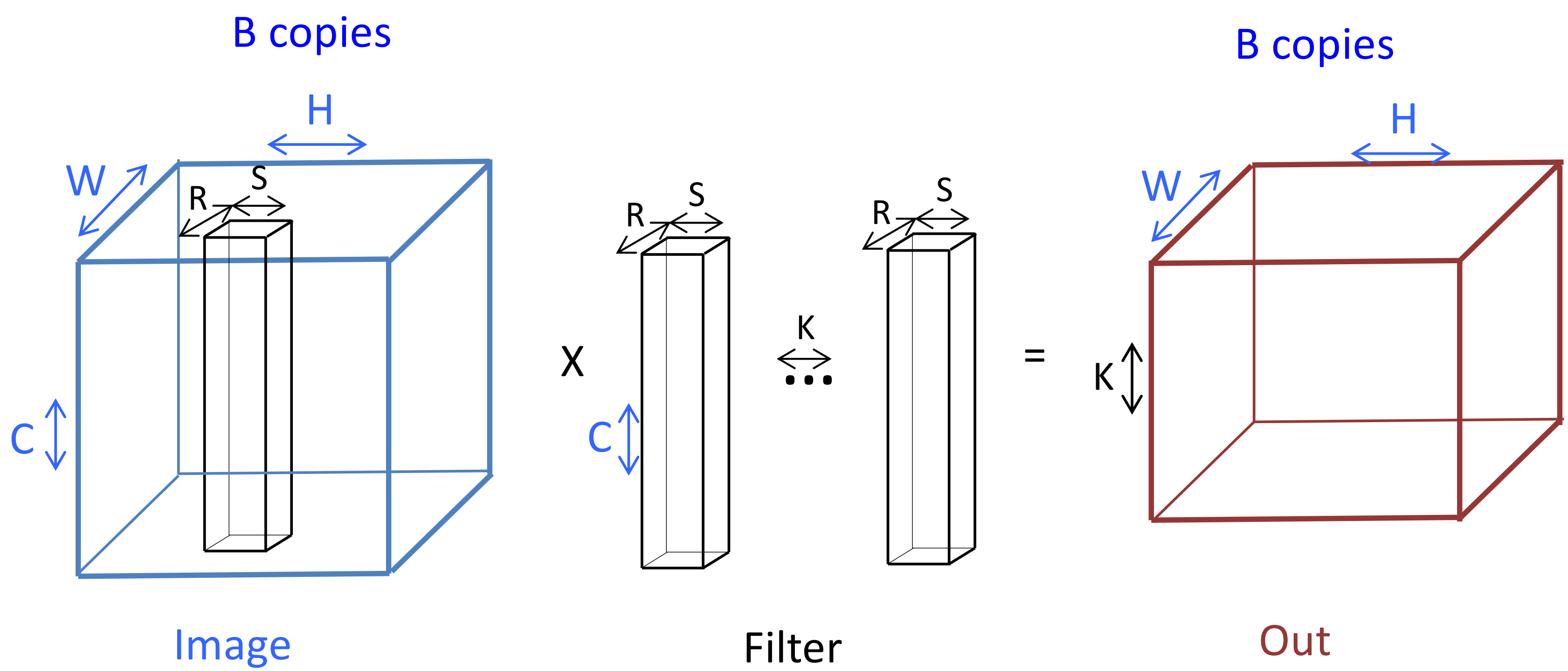
What CNNs compute



What CNNs compute

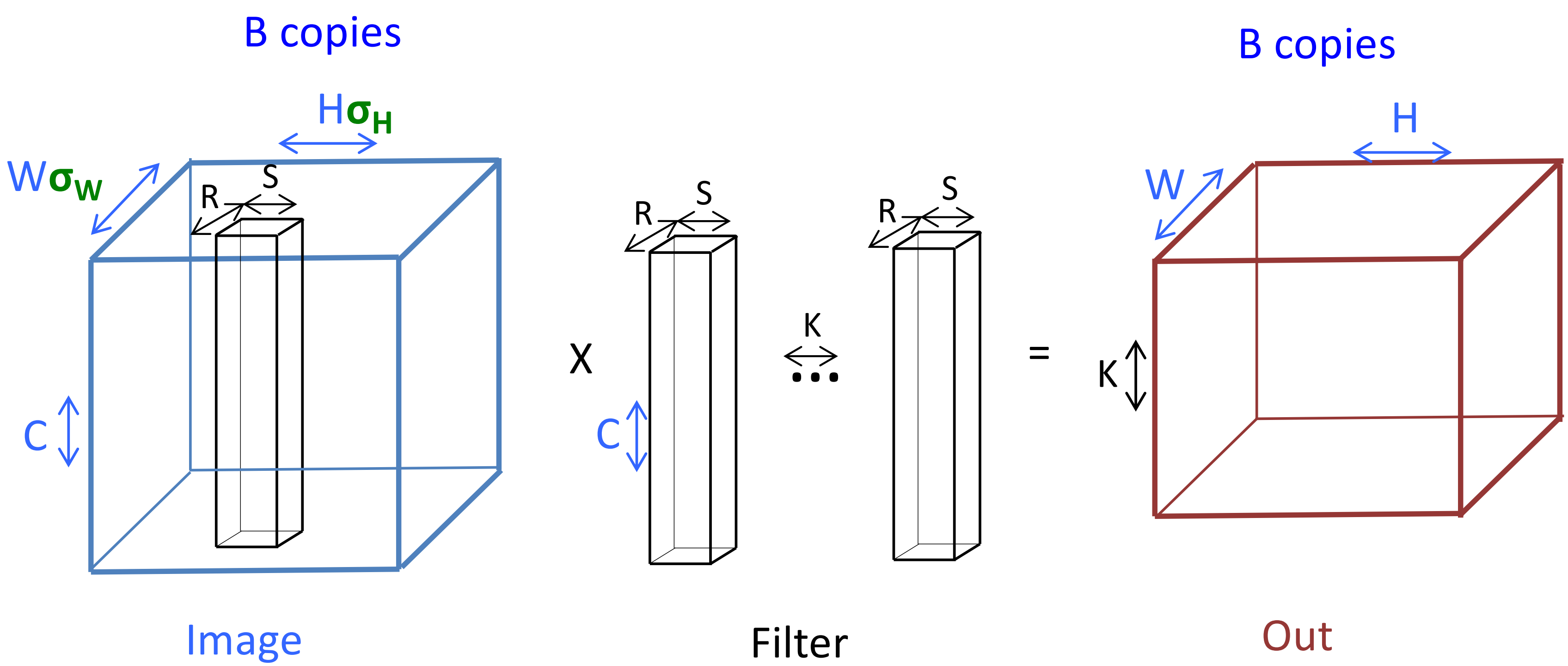


What CNNs compute



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

$$\text{Out}(k, h, w, b) \mathrel{+}= \text{Image}(r+\sigma_W w, s+\sigma_H h, c, b) * \text{Filter}(k, r, s, c)$$

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

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**

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
  - $A(i_1)$, $B(i_2, i_3+i_4)$, $C(i_1-i_2, i_2+3*i_3- 5*i_4 +2, \ldots)$, …
- 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

- 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

Time to redesign all

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

Don't Communic…

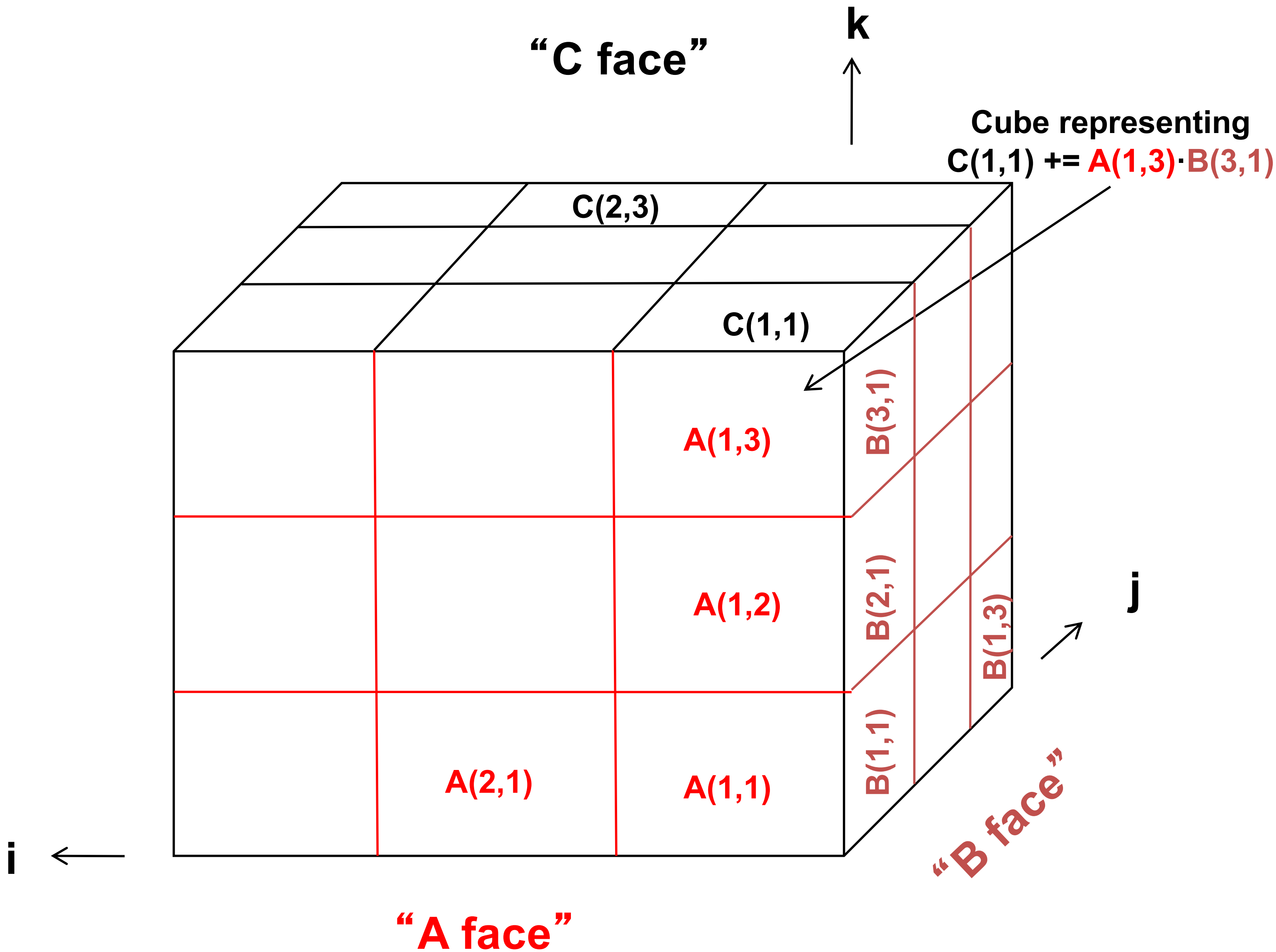
77

Backup slides

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

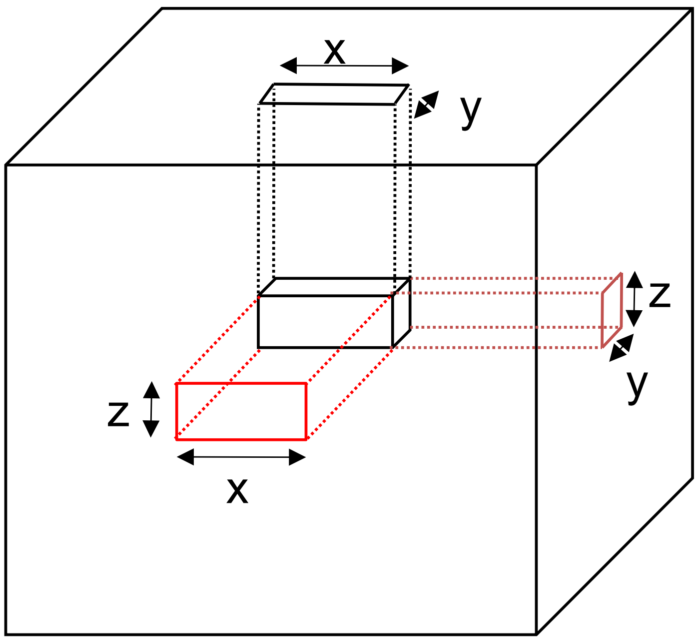
- Basic Counting Argument:
  - Only M entries of A, B and C are available in cache
  - Find an upper bound F on the number of different iterations C(i,j) = C(i,j) + A(i,k)*B(k,j) we can perform
  - Need to refill cache n^3/F times to complete algorithm
  - Need to read/write at least $M\ n^3/\ F$ words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

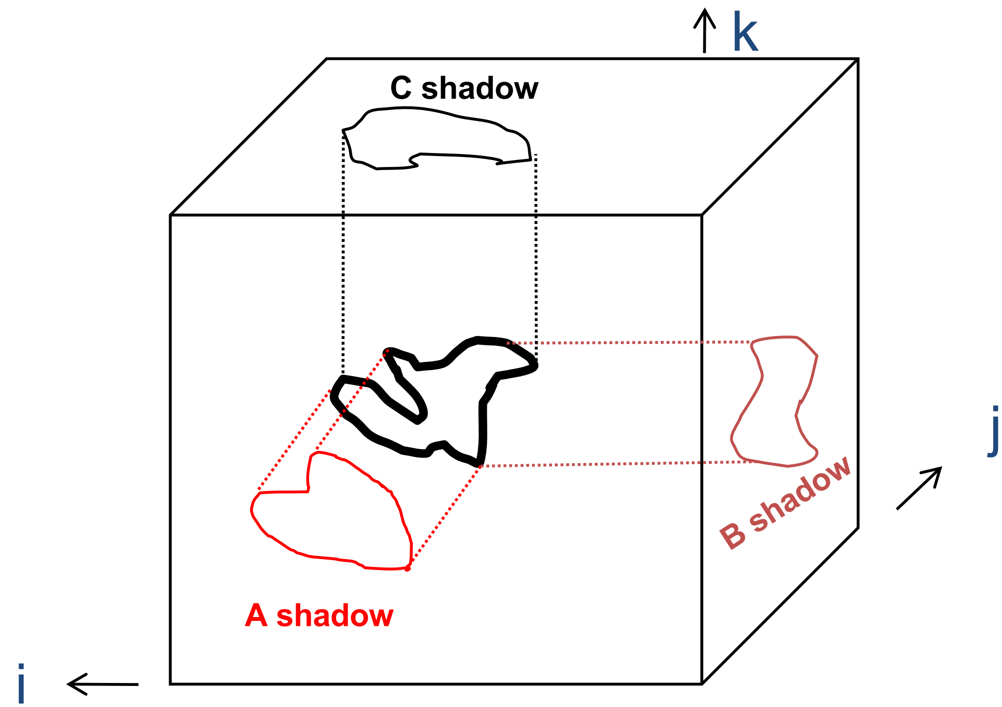
Proof of Communication Lower Bound on C = A·B (2/4)



- 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)





(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

 # 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)$^{1/2}$

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 $C = A \cdot B$ (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