MS76: Communication-Avoiding Algorithms - Part I of II

4:00-4:25 Communication-Avoiding Algorithms: Challenges and New Results
Erin C. Carson

4:30-4:55 Communication-Optimal Loop Nests
Nicholas Knight

5:00-5:25 Communication Lower Bounds for Matricized-Tensor Times Khatri-Rao Product
Grey Ballard, Nicholas Knight, Kathryn Rouse

5:30-5:55 Matrix Multiplication, a Little Faster
Oded Schwartz and *Elaye E. Karstadt*

MS93: Communication-Avoiding Algorithms - Part II of II

Friday, July 14, 4:00 PM - 6:00 PM, *Spirit of Pittsburgh B*

4:00-4:25 Communication-Avoiding Primal and Dual Methods for Regularized Least-Squares
Aditya Devarakonda, Kimon Fountoulakis, James Demmel, and Michael Mahoney

4:30-4:55 Communication-Avoiding Sparse Inverse Covariance Matrix Estimation
Penporn Koanantakool, Sang-Yun Oh, Dmitriy Morozov, Aydin Buluc, and Leonid Oliker, Katherine Yelick

5:00-5:25 Black-box Communication Optimal Low Rank Approximations
Alan Ayala, Laura Grigori

5:30-5:55 Performance of S-step and Pipelined Krylov Methods
Ichitaro Yamazaki, Mark Hoemmen, Jack J. Dongarra, Piotr Luszczek

Communication-Avoiding Algorithms: Challenges and New Results

Erin Carson

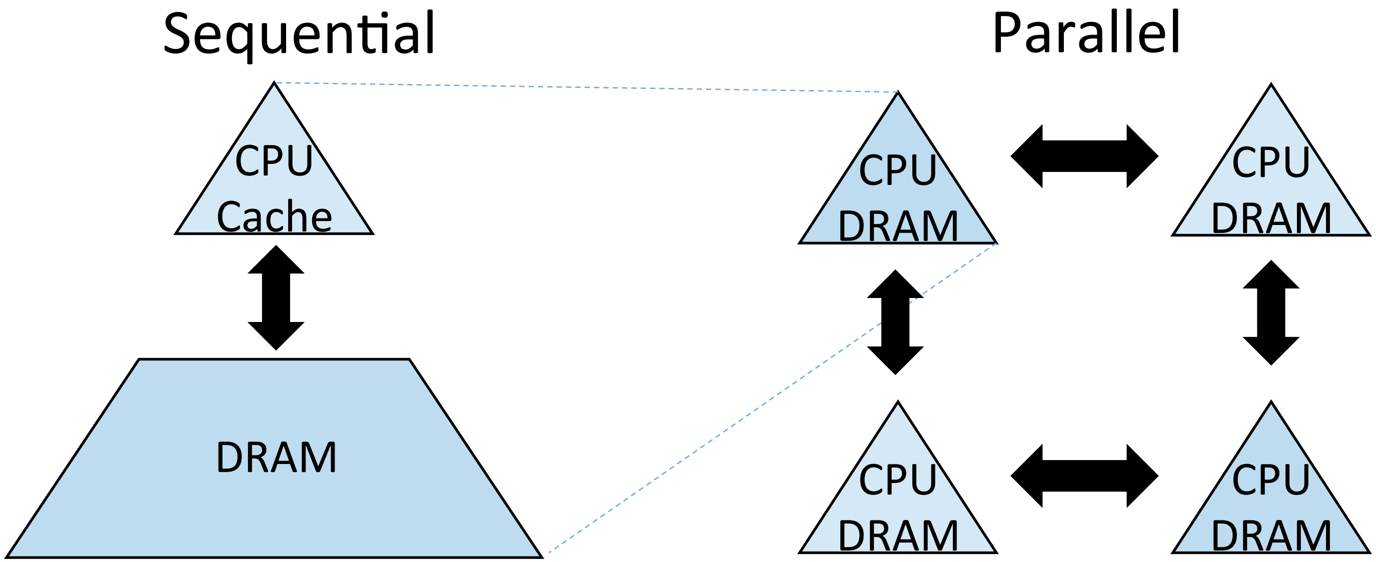
New York University

SIAM Annual Meeting 2017

Pittsburgh, PA, USA

What is communication?

- Runtime of an algorithm is the sum of
  - #flops x (time/flop) → **computation**
  - #words moved x (1/bandwidth)
  - #messages x latency → **communication**



- Communication is expensive, computation is cheap
  - in terms of both time and energy! (time → joules)

Future exascale systems

	Petascale Systems (2009)
System Peak	$2 \cdot 10 \uparrow 15$ flops/s
Node Memory Bandwidth	25 GB/s
Total Node Interconnect Bandwidth	3.5 GB/s
Memory Latency	100 ns
Interconnect Latency	1 μs

*Sources: from P. Beckman (ANL), J. Shalf (LBL), and D. Unat (LBL)

- Gaps between communication/computation cost only growing larger in future systems
- **Avoiding communication will be essential for applications at exascale!**

Motivation for CA Algorithms

- This problem will not be solved (in the foreseeable future) in hardware
- Needs to be addressed higher in the computing stack - at the ***algorithm*** level
- A paradigm shift in the way the numerical algorithms are designed is required
  - **We can't only think about computational complexity, we must also think about *communication complexity***
- Communication-avoiding algorithms:
  - Minimize communication volume (total words moved)
  - Minimize number of messages
  - Minimize over multiple levels of memory/parallelism
  - Allow redundant computations

Work in CA algorithms

- For numerical linear algebra computations…

1. Prove lower bounds on communication cost

2. Design new (stable) algorithms that attain these lower bounds

3. Implement algorithms on various machines, for various applications

Brief History: Dense Linear Algebra

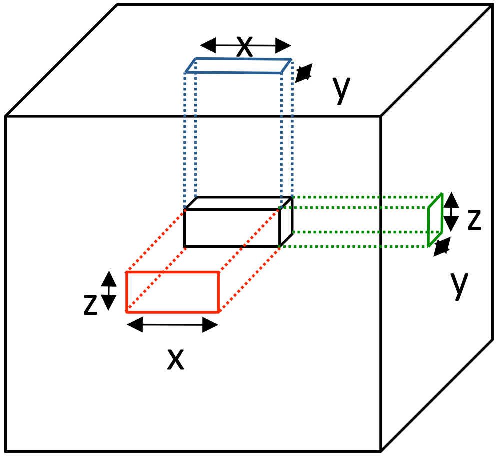
- Communication long recognized as a bottleneck
  - BLAS1 → BLAS2 → BLAS3
  - EISPACK, LINPACK → LAPACK, ScaLAPACK

- Lower bounds for (dense) matrix multiply
  - Hong and Kung (1981): for sequential matmul, must move $\Omega(n^3 / M^{1/2})$ words (M is size of fast memory)
  - Irony, Tiskin, Toledo (2004): Generalized to parallel case: when each of P processors stores $M = O(n^2 / P)$ words, $\Omega(n^3 / P^{1/2})$ words moved.
    - Attainable by Cannon's Algorithm (Cannon 1969), SUMMA (van de Geijn and Watts 1997)
  - Demmel, Grigori, Hoemmen, Langou (2008): same lower bounds apply to LU, QR factorization

Generalization: 3 nested loops

- Generalized by Ballard, Demmel, Holtz, Schwartz (2011) to any "3-nested loops" algorithm, using Loomis Whitney inequality
    - matmul, Cholesky, LU, LDL^T, QR, Floyd-Warshall, etc.; for dense and sparse matrices; sequential, parallel, hybrid
    - words moved = $\Omega(\#flops/M^{1/2})$,

      messages = $\Omega(\#flops/M^{3/2})$
    - (assuming $M=O(n^2/P)$)

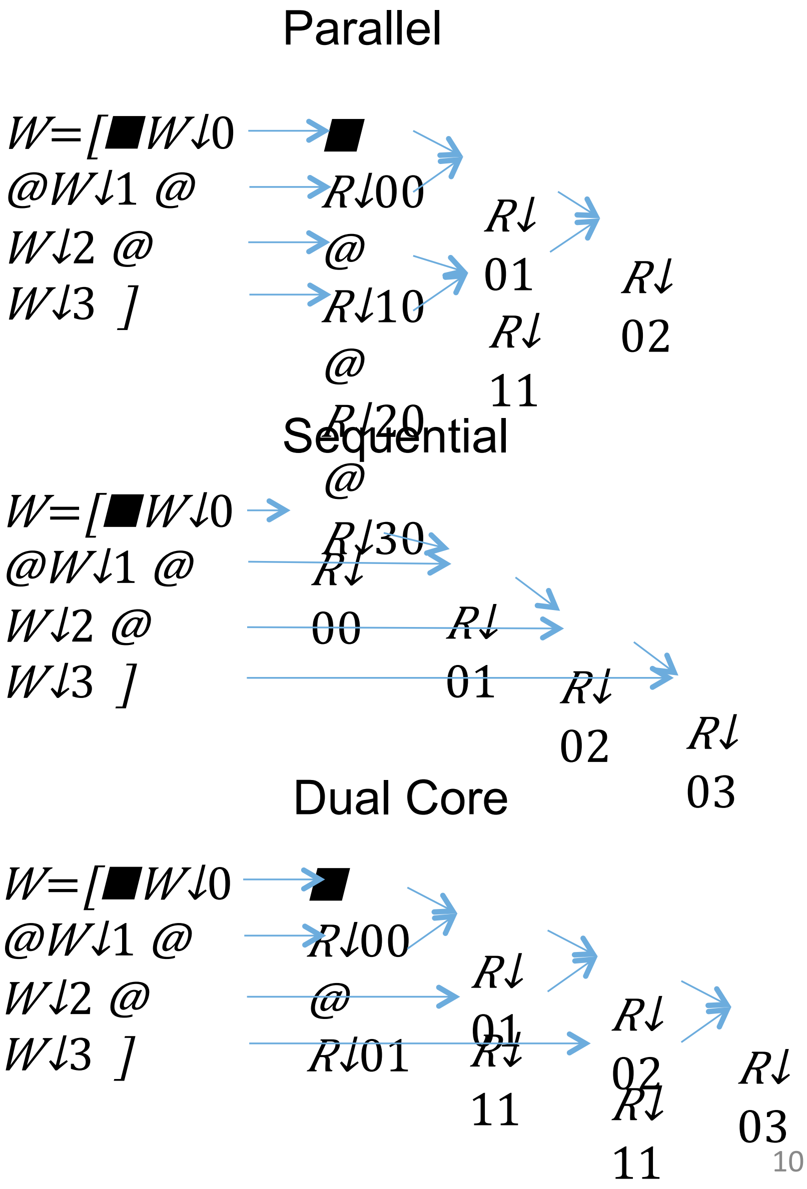


Algorithm	Minimizes # words moved	Minimizes # words moved and # messages
Cholesky	ScaLAPACK	ScaLAPACK
LU	ScaLAPACK	**CALU** **(Grigori, Demmel, Xiang, 2008), (Khabou, Demmel, Grigori, Gu, 2012)**
QR	ScaLAPACK	**CAQR** **(Demmel, Grigori, Hoemmen, Langou, 2008), (Ballard et al., 2014)**

- 3D matmul: $M=O(n^2/P^{2/3})$, 2.5D matmul: $M=cn^2/P$

Tall-Skinny QR

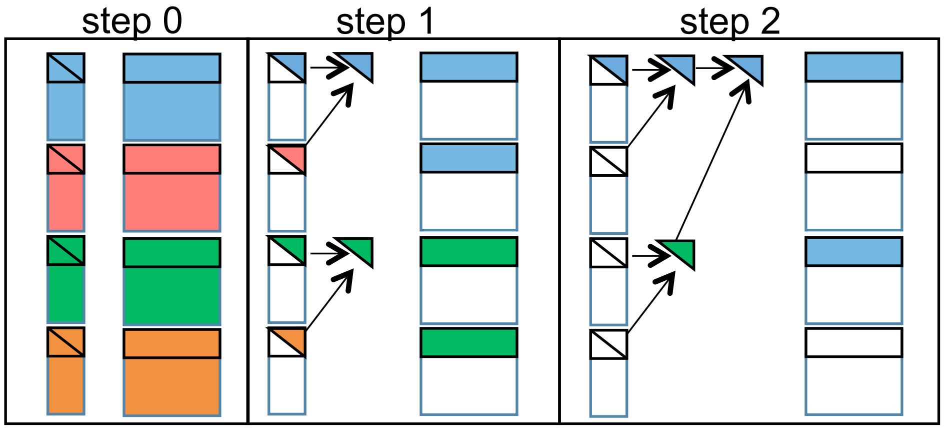
- TSQR: QR factorization of a tall skinny matrix using Householder transformations
- QR decomposition of m x b matrix W, m >> b
  - P processors, block row layout
- **Classic Parallel Algorithm**
  - Compute Householder vector for each column
  - Number of messages ∝ b log P
- **Communication Avoiding Algorithm**
  - Reduction operation, with QR as operator
  - Number of messages ∝ log P


Parallel
W=[■W↓0 @W↓1 @ W↓2 @ W↓3]
■ R↓00 @ R↓10 @ R↓20 @ R↓30
R↓ 01
R↓ 11
R↓ 02
Sequential
W=[■W↓0 @W↓1 @ W↓2 @ W↓3]
R↓ 00
R↓ 01
R↓ 02
R↓ 03
Dual Core
W=[■W↓0 @W↓1 @ W↓2 @ W↓3]
■ R↓00 @ R↓01
R↓ 01
R↓ 11
R↓ 02
R↓ 11
R↓ 03

CAQR

- Demmel, Grigori, Hoemmen, Langou (2008)
- TSQR on panels
- Performing trailing matrix update is complicated - Q factor represented implicitly
    - Can reconstruct Householder vectors from Tall-Skinny QR (TSQR-HR) (Ballard, Demmel, Grigori, Jacquelin, Knight, Nyugen, 2015)

- Close to optimal
    - Assume $M=O(n^2/P)$
    - Choose b near $n/P^{1/2}$
- Words moved: $O(n^2 \log P / P^{1/2})$
    - Lower bound: $\Omega(n^2/P^{1/2})$
- Messages: $O(n \log P / b)$
    - Lower bound: $\Omega(P^{1/2})$



- As stable as Householder QR

Recent work (*Ayala,* Claeys, Grigori)

- Talk tomorrow in Part II (MS93)
- Comm.-optimal low-rank approximation of matrices arising from boundary element problems
- Uses CAQR factorization

TSQR/CAQR implemented in

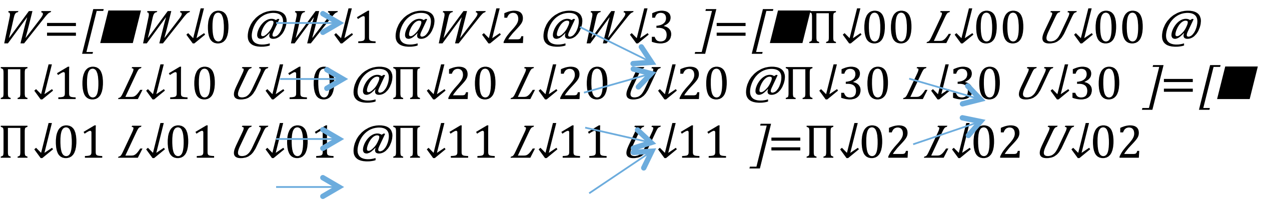
- Intel MKL library
- GNU Scientific Library
- ScaLAPACK
- Spark for data mining

Tall-Skinny LU

- Similar idea as in TSQR; uses tournament pivoting

- Step 1: preprocessing: select a set of b good pivot rows
  - Reduction operation with GEPP being the operator used to select pivot rows at each node of the reduction tree
  - Like in TSQR, shape of reduction tree depends on underlying architecture

- Step 2: permute the b pivot rows into the first b positions in the panel; perform LU factorization with no pivoting



𝑊=[■𝑊↓0 @𝑊↓1 @𝑊↓2 @𝑊↓3]=[■Π↓00 𝐿↓00 𝑈↓00 @
Π↓10 𝐿↓10 𝑈↓10 @Π↓20 𝐿↓20 𝑈↓20 @Π↓30 𝐿↓30 𝑈↓30]=[■
Π↓01 𝐿↓01 𝑈↓01 @Π↓11 𝐿↓11 𝑈↓11]=Π↓02 𝐿↓02 𝑈↓02

Π↓2↑𝑇 Π↓1↑𝑇 Π↓0↑𝑇 𝑊=𝐿

- Stability caveat: upper bound on the growth factor is worse than for GEPP
  - For GEPP, growth factor bounded by 2^{n-1} ; for CALU, $2^{n\ell}$, where ℓ is the depth of the reduction tree

CALU

- (Grigori, Demmel, Xiang, 2008), (Khabou, Demmel, Grigori, Gu, 2012)
- Assume 2D grid of P processors, using a 2D block cyclic layout with square blocks of size b

```
For ib = 1 to n-1 step b
        1. Find permutation for current panel using TSLU
        2. Apply all row permutations (pdlaswp)
                - broadcast pivot information along the rows of the grid
        3. Compute panel factorization (dtrsm)
        4. Compute block row of U (pdtrsm)
                - broadcast right diagonal part of L of current panel
        5. Update trailing matrix (pdgemm)
                - broadcast right block column of L
                - broadcast down block row of U
```

$M=O(n^2/P)$, choose b near $n/P^{1/2}$

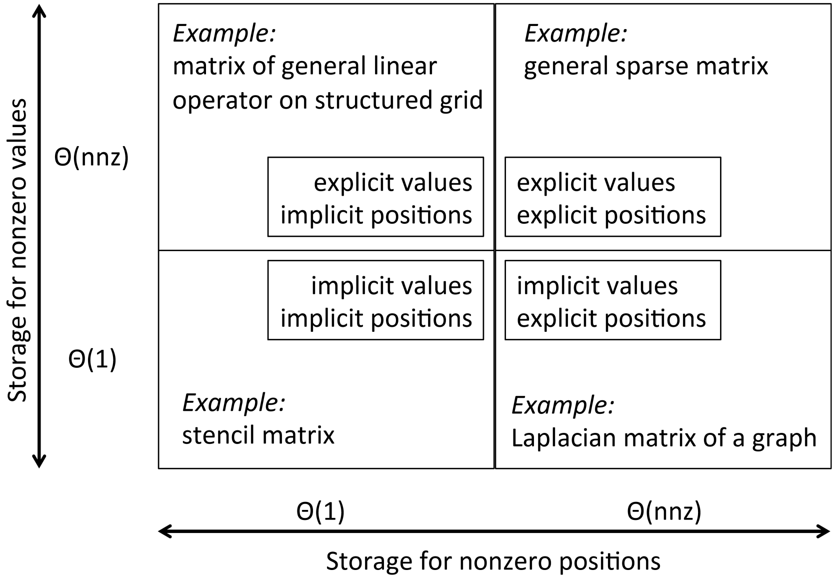
- Words: $O(n^2 \log P/P^{1/2})$; Lower bound: $\Omega(n^2/P^{1/2})$
- Messages: $O(n\log P/b)$; Lower bound: $\Omega(P^{1/2})$

CALU implemented in

- Cray's libsci
- To be implemented in LAPACK/ScaLAPACK

Sparse Matrix Computations

- Sparse Matrix x Vector (SpMV) ($y = Ax$)
  - Very communication-bound; no reuse
  - Lower bound depends on sparsity structure, algorithm used (1D rowwise/ colwise, 2D, etc.)
  - Communication cost depends on partition
  - Hypergraph models capture communication dependencies (Catalyurek, Aykanat, 1999)
    - minimize hypergraph cut = minimize words moved

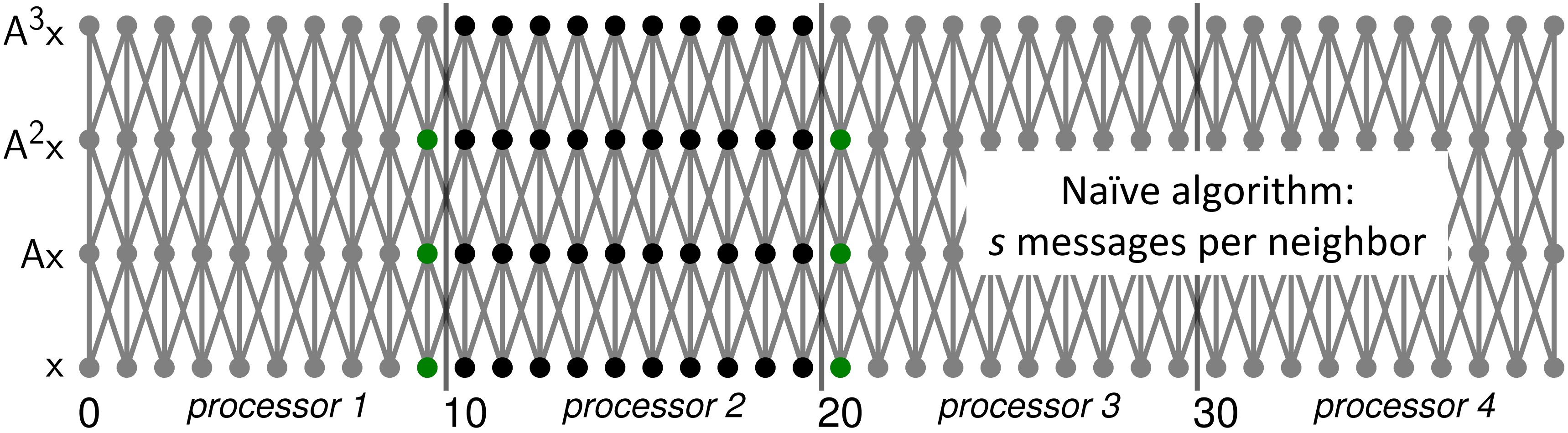


Storage for nonzero values \ Storage for nonzero positions	Θ(1)	Θ(nnz)
Θ(nnz)	*Example:* matrix of general linear operator on structured grid explicit values implicit positions	*Example:* general sparse matrix explicit values explicit positions
Θ(1)	implicit values implicit positions *Example:* stencil matrix	implicit values explicit positions *Example:* Laplacian matrix of a graph

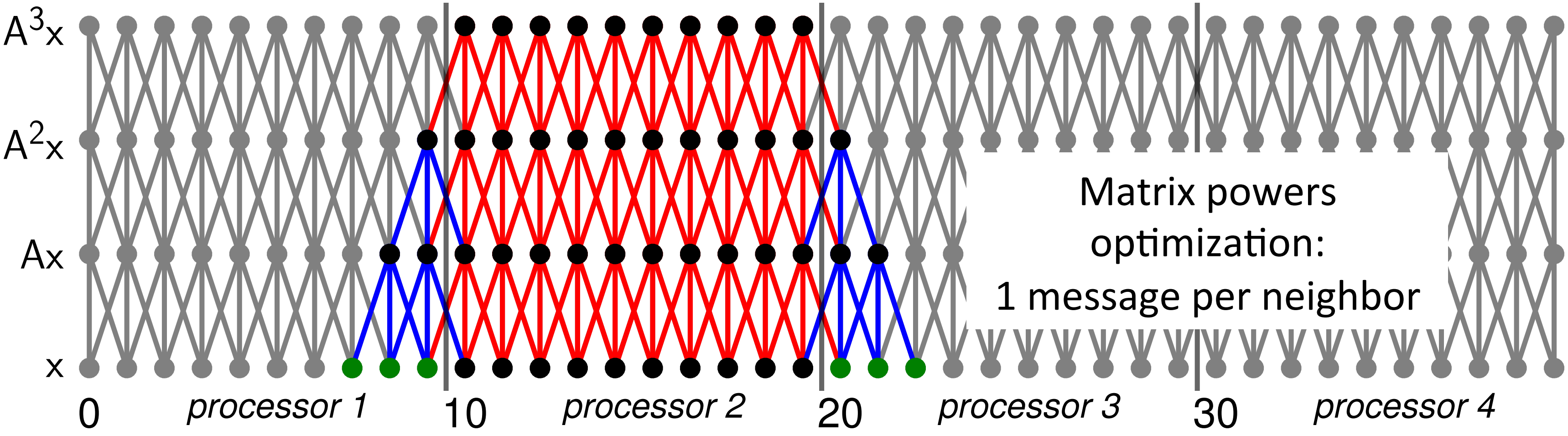
- Repeated SpMVs ($Y=[Ax,A\uparrow 2\ x,\ldots,A\uparrow k\ x]$)
  - Naive approach: k repeated SpMVs
  - Communication-avoiding approach: "matrix powers kernel"
    - see, e.g., (Demmel, Hoemmen, Mohiyuddin, Yelick, 2008)
    - Avoids communication:
      - In serial, by exploiting temporal locality in reading A, reading vectors
      - In parallel, by doing only 1 'expand' phase (instead of k).

Parallel Matrix Powers Kernel

Example: tridiagonal matrix, *s* = 3, *n* = 40, *p* = 4



Naïve algorithm:
s messages per neighbor



Matrix powers optimization:
1 message per neighbor

Sparse Matrix Computations

- **Sparse matrix x sparse matrix**
  - (Ballard, Druinsky, Knight, Schwartz, 2016)
  - Fine-grained and coarse-grained hypergraph models for sparse matrix-matrix multiplication
  - Identifying a communication-optimal algorithm for particular input matrices is equivalent to solving a hypergraph partitioning problem

- **Sparse matrix x dense matrix**
  - Building block of an increasing number of applications in many areas such as machine learning and graph algorithms
  - (Koanantakool, Azad, Buluc, Morozov, Oh, Oliker, Yelick, 2016)
  - Communication lower bounds
  - New communication-avoiding algorithms based on 1.5D decomposition
    - Outperform 2D and 3D variants in both theory and practice

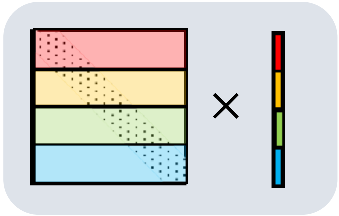
Krylov subspace methods

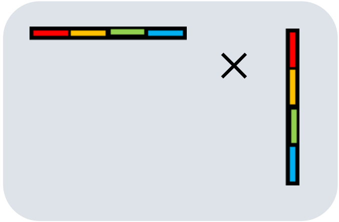
- **Krylov Subspace Method** is a projection process onto the Krylov subspace

$$\mathcal{K}_i(A, r_0) = span\{r_0, Ar_0, A^2 r_0, \ldots, A^{i-1} r_0\}$$

where A is an $N \times N$ matrix and $r_0 = b - Ax_0$ is a length-N vector

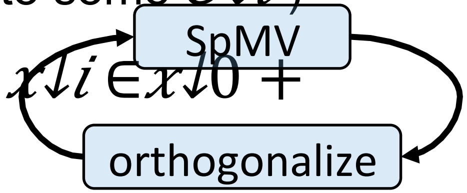
- Linear systems $Ax = b$, eigenvalue problems, singular value problems, least squares, etc.
- Ex: Lanczos/Conjugate Gradient (CG), Arnoldi/Generalized Minimum Residual (GMRES), Biconjugate Gradient (BICG), BICGSTAB, GKL, LSQR, etc.
- In each iteration,
  - Add a dimension to the Krylov subspace
    - Forms nested sequence of Krylov subspaces





$$\mathcal{K}_1(A, r_0) \subset \mathcal{K}_2(A, r_0) \subset \cdots \subset \mathcal{K}_i(A, r_0)$$

  - Orthogonalize (with respect to some $\mathcal{C}_i$)
  - Select approximate solution $x_i \in x_0 + \mathcal{K}_i(A, r_0)$



Communication-avoiding (s-step) KSMs

- Idea: Compute blocks of s iterations at once
  - Compute updates in a different basis (using matrix powers kernel)
  - Communicate every s iterations instead of every iteration
  - Reduces number of synchronizations per iteration by a factor of s

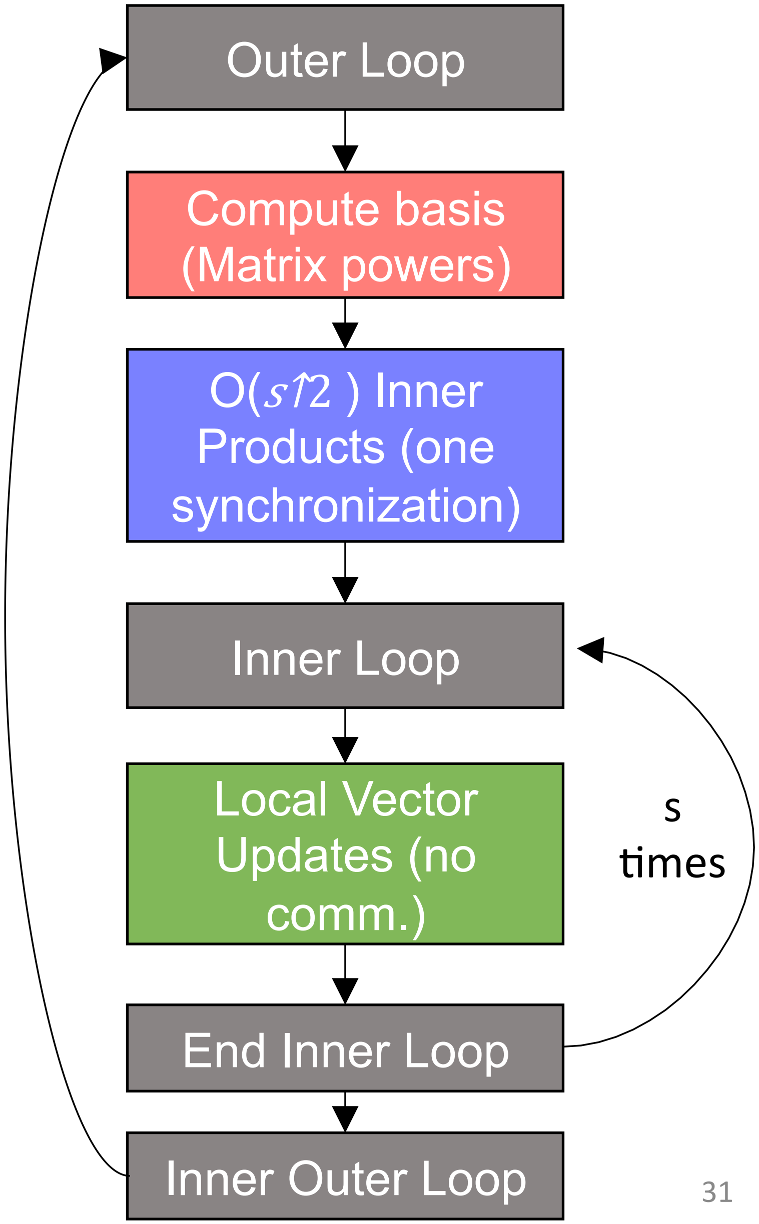
- An idea rediscovered many times…
- First related work: s-dimensional steepest descent, least squares
  - Khabaza (‘63), Forsythe (‘68), Marchuk and Kuznecov (‘68)
- Flurry of work on s-step Krylov methods in ‘80s/early ‘90s: see, e.g., Van Rosendale (1983); Chronopoulos and Gear (1989)

- Another approach: "pipelined" Krylov subspace methods
  - Reduce the cost of synchronization by enabling overlapping of All-Reduces and other computations

Recent work (*Yamazaki*, Hoemmen, Dongarra, Luszczek)
- Talk tomorrow in Part II (MS93)
- Performance comparison of s-step and pipelined Krylov subspace methods
- New approach that combines these two techniques

CA-CG (s-step CG)

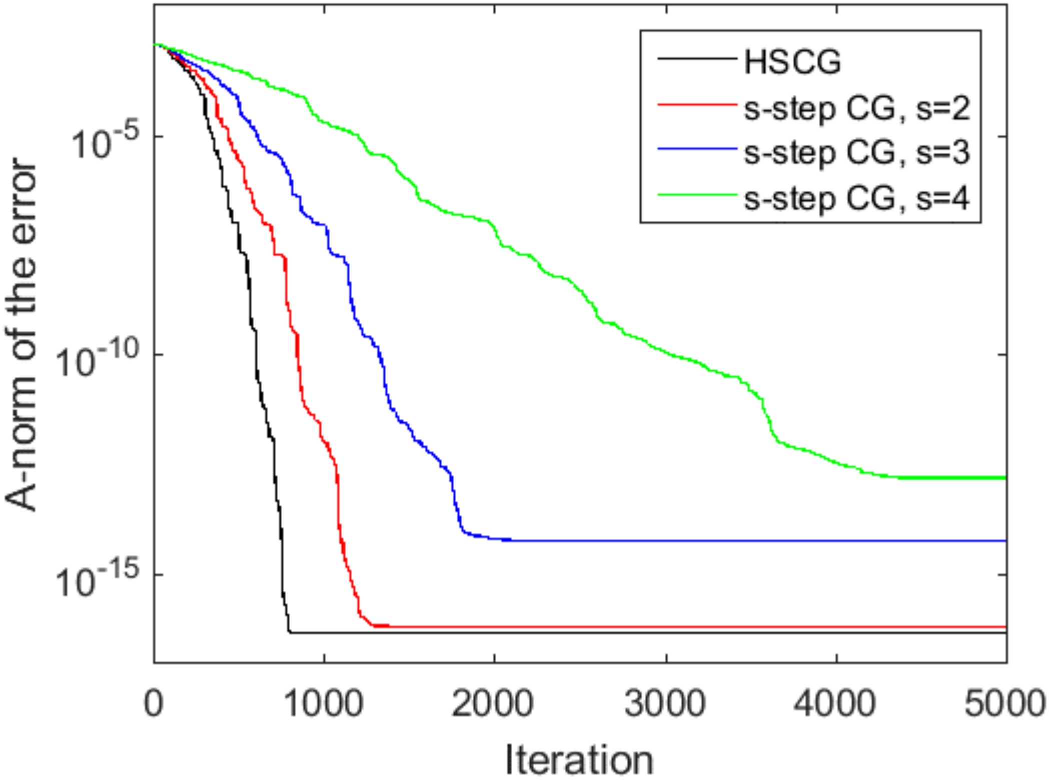
```
$r_0 = b - A x_0$, $p_0 = r_0$
for $k=0$:nmax/$s$
    Compute $\mathcal{Y}_k$ and $\mathcal{B}_k$ such that $A\mathcal{Y}_k = \mathcal{Y}_k \mathcal{B}_k$ and
        span($\mathcal{Y}_k$) = $\mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$
    $\mathcal{G}_k = \mathcal{Y}_k^T \mathcal{Y}_k$
    $x_0' = 0$, $r_0' = e_{s+2}$, $p_0' = e_1$
    for $j=1:s$
        $\alpha_{sk+j-1} = r_{j-1}'^T \mathcal{G}_k r_{j-1}' / p_{j-1}'^T \mathcal{G}_k \mathcal{B}_k p_{j-1}'$
```



Outer Loop
Compute basis (Matrix powers)
O(s^2) Inner Products (one synchronization)
Inner Loop
Local Vector Updates (no comm.)
End Inner Loop
Inner Outer Loop
s times

CA-CG (s-step CG)

s-step CA-CG with monomial basis ($\mathcal{Y}=[p_i, Ap_i, \ldots, A^s p_i, r_i, Ar_i, \ldots A^{s-1} r_i]$)



Changes to how the recurrences are computed can exacerbate finite precision effects of **convergence delay** and **loss of accuracy**!

A: bcsstk03 from UFSMC, b: equal components in the eigenbasis of A and $\|b\|=1$

$N=112$, $\kappa(A)\approx 7e6$

Optimizing CA iterative solvers

- CA variants are designed to reduce the time/iteration
- But what we really want to minimize is the **runtime, subject to some constraint on accuracy,**

runtime = (time/iteration) x (# iterations)

- Known that attainable accuracy and convergence rate in CA Krylov subspace methods depends on conditioning of the generated s-step bases
  - (C. and Demmel, 2014), (C. and Demmel, 2015), (Philippe and Reichel, 2012)

- Using finite precision analysis, can develop ways to improve numerical behavior while still avoiding communication in each iteration
  - e.g., residual replacement (C. and Demmel, 2014), adaptive basis size (C. 2016)

- To reduce communication, can also focus on reducing the number of iterations
  - Preconditioning
    - Challenging to implement in a CA way, but sometimes possible, e.g., ILU (Grigori, Moufawad, 2015), SPAI (Dehnavi, Demmel, Fernández, 2014), DD (Yamazaki, Rajamanickam, Boman, Hoemmen, Heroux, Tomov, 2014)

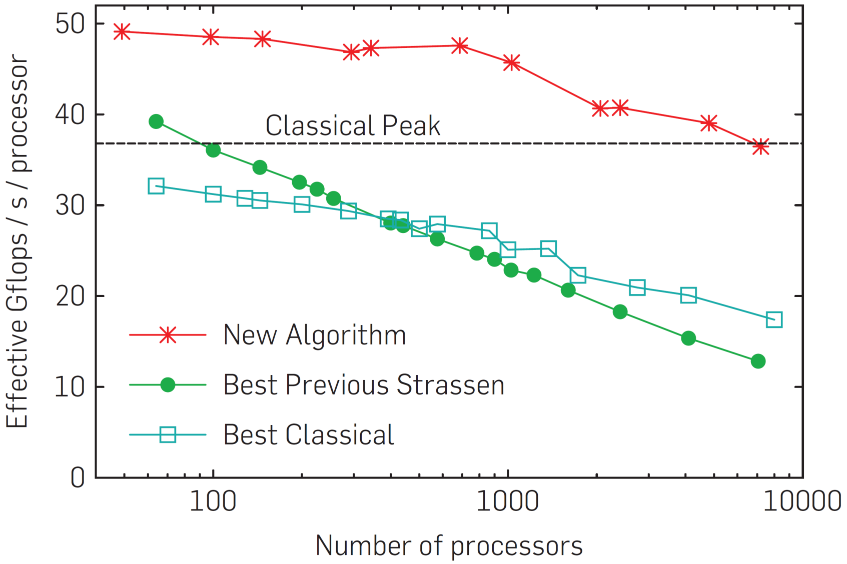
Extensions: Strassen's Matmul

- Ballard, Demmel, Holtz, Schwartz (2011 SPAA, 2012 JACM, 2014 CACM)

	Classical $O(n^{3})$ matmul	Strassen's $O(n^{\lg 7})$ matmul	Strassen-like $O(n^{\omega})$ matmul
words moved:	$\Omega(M/P\cdot(n/\sqrt{M})^{3})$	$\Omega(M/P\cdot(n/\sqrt{M})^{\lg 7})$	$\Omega(M/P\cdot(n/\sqrt{M})^{\omega})$

- New communication-optimal parallelization of Strassen’s algorithm
- Outperforms all previous matrix multiplication algorithms

Recent work: (Schwartz and *Karstadt*)
- 4th talk in this MS
- New faster version of Strassen's alg.
- Communication costs, parallelization, extension to other Strassen-like algorithms



(Strong scaling on a Cray XT4)

Extensions: Tensor Computations

- For tensor contractions, generalized lower bounds apply
  - Cyclops Tensor Framework (Solomonik, Matthews, Hammond, Stanton, Demmel, 2014)
    - A massively parallel tensor contraction framework for coupled-cluster computations
    - Avoids communication by exploiting replication, dynamically selecting a parallel decomposition with the least communication cost

Recent work (*Ballard*, Knight, Rouse)

- 3rd talk in this MS!
- Matricized-tensor times Khatri-Rao product (MTTKRP)
  - Bottleneck in CANDECOMP/PARAFAC decomposition of a tensor
- Communication lower bounds for sequential and parallel memory models, communication costs of existing algorithms

Extension: Machine Learning Algorithms

- CA-SVMs (You, Demmel, Czechowski, Song, Vuduc, 2015)

Recent work (*Devarakonda*, Fountoulakis, Demmel, Mahoney)

- Talk tomorrow in Part II (MS93)
- Communication-avoiding primal and dual block coordinate descent methods for regularized least-squares
- Extends results on CA-KSMs

Recent work (*Koanantakool*, Oh, Morozov, Buluc, Oliker, Yelick)

- Talk tomorrow in Part II (MS93)
- Communication-Avoiding Sparse Inverse Covariance Matrix Estimation
- Main bottleneck: iterative sparse-dense matmul
- For the first time, enables analyzing high-dimensional data sets with millions of variables and arbitrary underlying graph structures
- Experimental results using 3D brain fMRI data sets

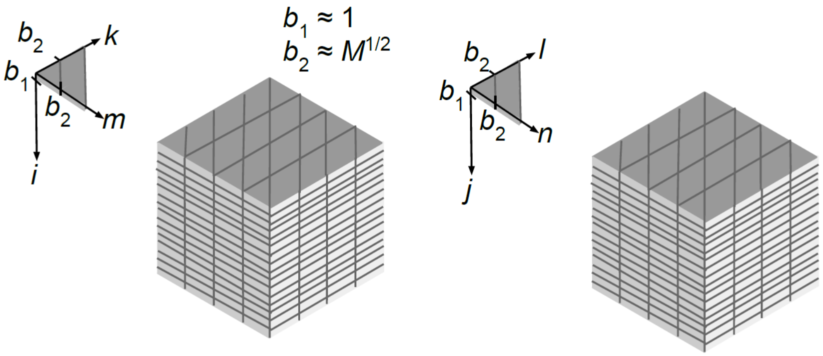
Generalization: Arbitrary Nested Loops

- Beyond Linear Algebra: Affine Array References
  - e.g., A(i + 2j; 3k + 4)
- (Christ, Demmel, Knight, Scanlon, Yelick, 2013); (Knight, 2015)

- Extend previous lower bounds to larger class of programs
- Lower bounds are computable - proof via Hölder-Brascamp-Lieb (HBL) theory
- Matching upper bounds (i.e., optimal algorithms) in special cases: linear algebra, tensor contraction, direct N–body, database join, etc. (when array references pick a subset of the loop indices)
- Ongoing work addresses attainability in the general case

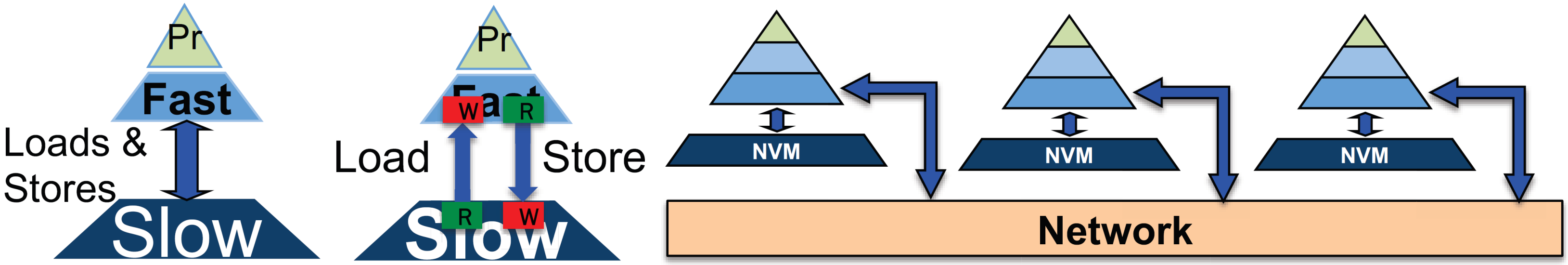
Recent work (*Knight*)

- 2nd talk in this MS!
- Application of HBL theory to Convolutional Neural Networks (CNNs)



Generalization: Write-Avoiding Algorithms

- (C., Demmel, Grigori, Knight, Koanantakool, Schwartz, Simhadri, 2016)
- Writes can be much more expensive than reads in some current and emerging storage devices such as nonvolatile memories.



(a) Symmetric model for sequential algorithms

(b) Asymmetric model for accounting reads/writes in sequential algorithms

(c) Distributed memory model with NVM disks on each node for parallel algorithms. Interprocessor communication occurs between second lowest level of the memories of each node.

“communication-avoiding” (CA): minimize the sum of reads and writes
"write-avoiding" (WA): CA and also do asymptotically fewer writes than reads

- Some CA algorithms are also WA
    - Matrix multiply, Cholesky, TRSM, direct N-body
- For some algorithms, WA algorithm cannot exist (#writes must be within a constant factor of the total # of reads and writes; "bounded reuse")
    - Strassen matmul, Cooley-Tukey FFT, cache oblivious (CO) algorithms for classical linear algebra (no WACO algs!)

Lots of speedups…

- Up to **12x** faster for 2.5D matmul on 64K core IBM BG/P
- Up to **3x** faster for tensor contractions on 2K core Cray XE/6
- Up to **6.2x** faster for All-Pairs-Shortest-Path on 24K core Cray CE6
- Up to **2.1x** faster for 2.5D LU on 64K core IBM BG/P
- Up to **11.8x** faster for direct N-body on 32K core IBM BG/P
- 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 **2x** faster for 2.5D Strassen on 38K core Cray XT4
- Up to **4.2x** faster for MiniGMG benchmark bottom solver, using CA-BICGSTAB (**2.5x** for overall solve), **2.5x** / **1.5x** for combustion simulation code
- Up to **42x** for Parallel Direct 3-Body

These and many more recent papers available at bebop.cs.berkeley.edu

Thank you!

erinc@cims.nyu.edu

math.nyu.edu/~erinc