Communication-Avoiding Iterative Methods

Erin Carson
UC Berkeley Parallel Computing Lab
BeBop Group

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

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

CA-GMRES (Hoemmen, Mohiyuddin, et al.)

Talk Outline

• What is communication?

• What are Krylov Subspace Methods?

• Communication-Avoiding Krylov Subspace Methods
  – New Communication-Avoiding Kernels

• Challenges in Communication-Avoiding Krylov Subspace Methods
  – Stability and Convergence
  – Performance

• Preconditioning

• Related Work: “s-step methods”

• Future Work
Talk Outline

- **What is communication?**
- What are Krylov Subspace Methods?
- Communication-Avoiding Krylov Subspace Methods
  - New Communication-Avoiding Kernels
- Challenges in Communication-Avoiding Krylov Subspace Methods
  - Stability and Convergence
  - Performance
- Preconditioning
- Related Work: “s-step methods”
- Future Work
What is “Communication”?

• Algorithms have 2 costs:
  – Arithmetic (FLOPS)
  – Movement of data

  • Two parameters: $\alpha$ – Latency, $\beta$ – Reciprocal Bandwidth
    – Time to move $n$ words of data is $\alpha + n\beta$
Communication in the future…

• Gaps growing exponentially…
  
  • Floating point time $\ll 1/\text{Network BW} \ll \text{Network Latency}$
    • Improving 59%/year vs. 26%/year vs. 15%/year
  
  • Floating point time $\ll 1/\text{Memory BW} \ll \text{Memory Latency}$
    • Improving 59%/year vs. 23%/year vs. 5.5%/year
  
• We want more than just “hiding” communication
  – Arbitrary speedups possible, vs. at most 2x speedup
Talk Outline

• What is communication?

• **What are Krylov Subspace Methods?**
  - Communication-Avoiding Krylov Subspace Methods
    - New Communication-Avoiding Kernels
  - Challenges in Communication-Avoiding Krylov Subspace Methods
    - Stability and Convergence
    - Performance

• Preconditioning

• Related Work: “s-step methods”

• Future Work
Motivation: Sparse Matrices

- Many algorithms for scientific applications require solving linear systems of equations: \( Ax = b \)

- In many cases, the matrix \( A \) is sparse
  - Sparse matrix: a matrix with enough zero entries to be worth taking advantage of
    - This means that information is “local” instead of “global”. A given variable only depends on some of the other variables.
  - Example: Simulating Pressure around Airfoil

Figure: Simulating Pressure over Airfoil. Source: http://www.nada.kth.se
Solving a Sparse Linear System

- **Direct** methods solve a linear system in a finite sequence of operations
  - Often used to solve *dense* problems
  - Ex: Gaussian Elimination

- **Iterative** methods iteratively refine an approximate solution to the system
  - Used when
    - System is large and sparse – direct method too expensive
    - We only need an approximation – don’t need to solve exactly, so less operations needed
    - A is not explicitly stored
  - Ex: Krylov Subspace Methods (KSMs)

Iterative Method for Solving $Ax = b$

Initial guess

Convergence? → Yes → Return solution

Convergence? → No → Refine Solution
How do Krylov Subspace Methods Work?

• A Krylov Subspace is defined as:
  \[ \mathcal{K}_k(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{k-1}v\} \]

• In each iteration,
  – Sparse matrix-vector multiplication (SpMV) with A to create new basis vector
    • Adds a dimension to the Krylov Subspace
  – Use vector operations to choose the “best” approximation of the solution in the expanding Krylov Subspace (projection of a vector onto a subspace)
    • How “best” is defined distinguishes different methods

• Examples: Conjugate Gradient (CG), Generalized Minimum Residual Methods (GMRES), Biconjugate Gradient (BiCG)
A Few Applications of Krylov Subspace Methods

- **Image Processing Applications**
  - Ex: Image segmentation, Contour detection

- **Physical simulations**
  - Solving PDEs
    - Often used in combination with Multigrid as bottom-solve
    - Ex: Simulating blood flow (Parlab’s Health App)

- **Mobile/Cloud applications**
  - Even more important where bandwidth is very limited, latency is long (or if this parameters are variable between machines!)
    - Auto-tuning becomes more important if we don’t know our hardware
Krylov Subspace Methods are Communication-Bound

• Problem: Calls to communication-bound kernels every iteration
  – SpMV (computing A*v)
    • Parallel: share/communicate source vector with neighbors
    • Sequential: read A (and vectors) from slow memory
  – Vector operations
    – Orthogonalization
      » Dot products
      » Vector addition and scalar multiplication
• Solution:
  – Replace Communication-bound kernels by Communication-Avoiding ones
  – Reformulate KSMs to use these kernels
Example: GMRES

Pseudocode to perform \( s \) steps of original algorithm:

1. \textbf{for} \( k = 1 \) to \( s \) \textbf{do}
2. \hspace{1em} \( w = Av_{k-1} \)
3. \hspace{1em} Orthogonalize \( w \) against \( v_0, \ldots, v_{k-1} \) using Modified Gram-Schmidt
4. \hspace{1em} \textbf{end for}
5. Compute solution using \( H \)

\textbf{SpMV operation in every iteration:} requires communication of current entries of \( v \) (parallel) / reading \( A \) and vectors from slow memory (sequential)

\textbf{Vector operations in every iteration:} requires global communication (parallel) / reading \( \mathcal{O}(n) \) words from slow memory (sequential)
Talk Outline

• What is communication?

• What are Krylov Subspace Methods?

• **Communication-Avoiding Krylov Subspace Methods**
  – New Communication-Avoiding Kernels

• Challenges in Communication-Avoiding Krylov Subspace Methods
  – Stability and Convergence
  – Performance

• Preconditioning

• Related Work: “s-step methods”

• Future Work
Communication-Avoiding KSMs

• We need to break the dependency between communication bound kernels and KSM iterations

• Idea: Expand the subspace $s$ dimensions ($s$ SpMVs with $A$), then do $s$ steps of refinement
  – unrolling the loop $s$ times

• To do this we need two new Communication-Avoiding kernels
  – “Matrix Powers Kernel” replaces SpMV
  – “Tall Skinny QR” (TSQR) replaces orthogonalization operations
The Matrix Powers Kernel

• Given $A$, $v$, and $s$, Matrix powers kernel computes
  \[
  \{v, Av, A^2v, \ldots, A^{s-1}v\}
  \]

• If we figure out dependencies beforehand, we can do all the
  communication for $s$ steps of the algorithm only
  reading/communicating $A$ $o(1)$ times!
    – Parallel case: Reduces latency by a factor of $s$ at the cost of
      redundant computations
    – Sequential case: reduces latency and bandwidth by a factor of $s$, no
      redundant computation

• Simple example: a tridiagonal matrix

```
Sequential

Parallel
```
Communication Avoiding Kernels: TSQR

• TSQR = Tall Skinny QR (#rows >> #cols)
  - QR: factors a matrix A into the product of
    • An orthogonal matrix (Q)
    • An upper triangular matrix (R)
  - Here, A is the matrix of the Krylov Subspace Basis Vectors
    • output of the matrix powers kernel
  - Q and R allow us to easily expand the dimension of the Krylov Subspace

• Usual Algorithm
  • Compute Householder vector for each column O(n log P) messages

• Communication Avoiding Algorithm
  • Reduction operation, with QR as operator O(log P) messages

• Shape of reduction tree depends on architecture
  - Parallel: use “deep” tree, saves messages/latency
  - Sequential: use flat tree, saves words/bandwidth
  - Multicore: use mixture

Figure: [ABDK10]
Example: CA-GMRES

s steps of original algorithm:

1: \textbf{for} \ k = 1 \textbf{ to } s \ \textbf{do} \\
2: \hspace{1em} w = A v_{k-1} \\
3: \hspace{1em} \text{Orthogonalize } w \text{ against } v_0, \ldots, v_{k-1} \text{ using Modified Gram-Schmidt} \\
4: \hspace{1em} \textbf{end for} \\
5: \hspace{1em} \text{Compute solution using } H

s steps of CA algorithm:

1: \ W = [v_0, Av_0, A^2v_0, \ldots, A^sv_0] \\
2: \ [Q, R] = TSQR(W) \\
3: \ \text{Compute } H \text{ using } R \\
4: \ \text{Compute solution using } H

s powers of A for no extra latency cost
s steps of QR for one step of latency
Runtime per kernel, relative to CA-GMRES(k,t), for all test matrices, using 8 threads and restart length 60

Platform: Intel Clovertown, 8 cores

[MHDY09]
Current CA Krylov Subspace Methods

• CG, Lanczos, Arnoldi (Hoemmen, 2010),
• GMRES (Hoemmen, Mohiyuddin, Demmel, Yelick, 2009)
• BiCG, CGS, BiCGStab (Carson, Knight, Demmel, 2011).

• Factor of s less communication than standard version.

• General approach for CG-like methods:
  – In each outer loop, compute s basis vectors from previous iteration’s residual vectors
  – Perform s inner loop iterations
    • Compute current recurrence coefficients
    • Replace SpMVs with local basis vector operations
    • Replace dot products with shorter, local dot products
  – continue until convergence…..
Talk Outline

• What is communication?

• What are Krylov Subspace Methods?

• Communication-Avoiding Krylov Subspace Methods
  – New Communication-Avoiding Kernels

• **Challenges in Communication-Avoiding Krylov Subspace Methods**
  – Stability and Convergence
  – Performance

• Preconditioning

• Related Work: “s-step methods”

• Future Work
Challenges: Stability and Convergence

• **Stability** of Communication-Avoiding Krylov Subspace Methods **depends on** $s$

• Does $v, Av, A^2v, \ldots$ look familiar?
  – Power Method! Converges to principle eigenvector
  – Expected linear dependence of basis vectors
    • Means the Krylov Subspace can’t expand any more – method breaks down, convergence stalls

• Can we remedy this problem to remain stable for larger $s$ values?
  – Yes! Other possible basis choices:
    • Newton $[v, (A - \theta_1 I)v, (A - \theta_2 I)(A - \theta_1 I)v, \ldots]$
    • Chebyshev $[v, T_1(v), T_2(v), \ldots]$
CABiCG Convergence, $s = 4$, Cond# $\sim 1e4$
CABiCG Convergence, $s = 10$, Cond# $\sim 1e4$
CABiCG Convergence, $s = 10$, Cond# $\sim 1e4$
CABiCG Convergence, $s = 10$, Cond# $\sim 1e4$
Summary of Preliminary Results

• Our CA variants (generally) maintain stability for s in between 2 and 10
  – Which basis (Monomial, Newton, or Chebyshev) is most effective depends on the specific Krylov method we use and the condition number of A (and other spectral properties of A)
  – Reduces communication costs by a factor of s
    • So, if s = 10, possible speedup is 10x!

• In general, as s increases, the number of iterations needed to converge increases, and after a certain point, the method breaks down
  – Could be remedied by preconditioning, extended precision, etc.

• Must choose s to maintain stability
Talk Outline

• What is communication?

• What are Krylov Subspace Methods?

• Communication-Avoiding Krylov Subspace Methods
  – New Communication-Avoiding Kernels

• Challenges in Communication-Avoiding Krylov Subspace Methods
  – Stability and Convergence
  – Performance

• Preconditioning

• Related Work: “s-step methods”

• Future Work
Challenges: Performance

• How to choose s?
  – Assuming that stability is not an issue…
  – After some value of s, the matrix is too dense to avoid communication using the Matrix Powers Kernel
  – But exactly computing this value of s requires computing the matrix powers!

• How to partition the matrix for $A^s x$?
  – As above, computing dependencies requires computing matrix powers
  – The redundant work ("ghost zones") are induced by the partition. So how can we achieve load balance?
Partitioning for CA-KSMs

Minimizing communication in matrix powers reduces to hypergraph partitioning \(s\)-level column-nets.

**Problem:** Computational and storage cost:
- \(s \times \) Boolean sparse matrix \-matrix multiplies!
Partitioning for CA KSMs

• Solution: Use reachability estimation [Cohen ’94]
  – $O(\text{nnz})$ time randomized algorithm for estimating size of transitive closure.
    • Calculating transitive closure costs $O(n^{*}\text{nnz})$

• Can be used to estimate nnz-per-column in matrix product $A^s$ in $O(\text{nnz})$ time
  – Can be used to **sparsify** the hypergraph – Drop large nets during construction
  – Reduces size of data structure and computational cost, while still providing a good partition

• Can be used to estimate **overlap** between columns – the number of nonzero rows two column have in common
  – This could allow us to heuristically load balance
Challenges: Performance for Stencil-like Matrices

- What if A is stencil-like (in general, $o(n)$ cost to read)?
  - In the sequential algorithm…
    - Not communication-bound due to reading A, but…
    - Communication bottleneck is now reading Krylov vectors
      - $O(kn)$ cost to read Krylov basis vectors every $k$ steps
- Can we reduce the communication cost of $k$ steps from $O(kn)$ to $O(n)$?

Figure: 2D 5-point stencil. Each grid-point is updated at each time-step using only nearest neighbor values
Streaming Matrix Powers Computation

- Idea: Don’t explicitly store basis vectors
  - Streaming Matrix Powers: Interleave matrix powers computation and construction of the Gram Matrix G
  - Part i computes $G^+=V_i^T V_i$, discards $V_i$

- Tradeoff: requires two matrix powers invocations, but bandwidth reduced by a factor of k
  - OK if reading and applying A is inexpensive (e.g., stencil, AMR base case, others?)
  - Overall communication reduced from $O(kn)$ to $O(n)$!
Auto-tuning for CA-KSMs

• Auto-tuning for stability
  – Choice of basis to use
    • Depends on s, condition number of A, method, etc.

• Auto-tuning for performance
  – Partitioning A amongst parallel processors to minimize communication
  – Partitioning for cache blocking to maximize cache reuse
  – Determine which variant of the matrix powers kernel to use
    • E.g., “streaming” if A is stencil-like
  – Many other standard parallel and sequential optimizations…

• Eventually will be built into pOSKI (Parallel Optimized Sparse Kernel Interface), an auto-tuning library for sparse matrix computations
Talk Outline

• What is communication?

• What are Krylov Subspace Methods?

• Communication-Avoiding Krylov Subspace Methods
  – New Communication-Avoiding Kernels

• Challenges in Communication-Avoiding Krylov Subspace Methods
  – Stability and Convergence
  – Performance

• **Preconditioning**

• Related Work: “s-step methods”

• Future Work
What is preconditioning?

• The number of iterations a KSM takes to converge depends on the “condition number” of $A$
  – Condition number is a property of a matrix/system (not of the algorithm or precision used)
    • For $Ax=b$, roughly denotes how error in $b$ affects error in $x$
  – The lower the condition number, the fewer iterations needed for convergence

• Preconditioning: Instead of solving $Ax=b$, solve $(MA)x = Mb$, where the matrix $MA$ has a lower condition number than $A$
  – Many methods exist for finding a matrix $M$ which has this property
    • “Sparse Approximate Inverse”, “Incomplete LU”, “Polynomial Preconditioning”, etc.

• This technique is used in almost all practical applications of KSMs
What About Preconditioning in CA-KSMs?

• Problem: CA preconditioning approach requires a different approach/implementation for each type of preconditioner!

• Existing algorithms
  • Polynomial preconditioners (Saad, Toledo)
    • M is polynomial in A – easily incorporated into Matrix Powers Kernel
  • CA-Left and Right preconditioning (Hoemmen, 2010)
    • For 2 non-trivial classes of preconditioners
    • 1 + o(1) more messages than single SpMV, 1 preconditioner solve
      • Tradeoff: computation cost increases significantly
        • Can require twice as many flops as s SPMVs!
Talk Outline

• What is communication?

• What are Krylov Subspace Methods?

• Communication-Avoiding Krylov Subspace Methods
  – New Communication-Avoiding Kernels

• Challenges in Communication-Avoiding Krylov Subspace Methods
  – Stability and Convergence
  – Performance

• Preconditioning

• Related Work: “s-step methods”

• Future Work
## Related Work: s-step Methods

<table>
<thead>
<tr>
<th>Author</th>
<th>Algorithm</th>
<th>Basis</th>
<th>Preconditioning</th>
<th>Matrix Powers?</th>
<th>TSQR?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Van Rosendale, 1983</td>
<td>CG</td>
<td>Monomial</td>
<td>Polynomial</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>Leland, 1989</td>
<td>CG</td>
<td>Monomial</td>
<td>Polynomial</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>Walker, 1988</td>
<td>GMRES</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulos and Gear, 1989</td>
<td>CG</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>Chronopoulos and Kim, 1990</td>
<td>Orthomin, GMRES</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulos, 1991</td>
<td>MINRES</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Kim and Chronopoulos, 1991</td>
<td>Symm. Lanczos, Arnoldi</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Sturler, 1991</td>
<td>GMRES</td>
<td>Chebyshev</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
## Related Work, contd.

<table>
<thead>
<tr>
<th>Author</th>
<th>Algorithm</th>
<th>Basis</th>
<th>Preconditioning</th>
<th>Matrix Powers?</th>
<th>TSQR?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joubert and Carey, 1992</td>
<td>GMRES</td>
<td>Chebyshev</td>
<td>None</td>
<td>Yes (stencil only)</td>
<td>No</td>
</tr>
<tr>
<td>Chronopoulous and Kim, 1992</td>
<td>Nonsymm. Lanczos</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>Bai, Hu, and Reichel, 1991</td>
<td>GMRES</td>
<td>Newton</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Erhel, 1995</td>
<td>GMRES</td>
<td>Newton</td>
<td>None</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>De Sturler and van der Vorst, 2005</td>
<td>GMRES</td>
<td>Chebyshev</td>
<td>General</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Toledo, 1995</td>
<td>CG</td>
<td>Monomial</td>
<td>Polynomial</td>
<td>Yes (stencil only)</td>
<td>-</td>
</tr>
<tr>
<td>Chronopoulous and Swanson, 1990</td>
<td>CGR, Orthomin</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>Chronopoulous and Kinkaid, 2001</td>
<td>Orthodir</td>
<td>Monomial</td>
<td>None</td>
<td>No</td>
<td>-</td>
</tr>
</tbody>
</table>
Talk Outline

• What is communication?

• What are Krylov Subspace Methods?

• Communication-Avoiding Krylov Subspace Methods
  – New Communication-Avoiding Kernels

• Challenges in Communication-Avoiding Krylov Subspace Methods
  – Stability and Convergence
  – Performance

• Preconditioning

• Related Work: “s-step methods”

• Future Work
Future Work

• Other CA Krylov Subspace methods?

• Evaluate current preconditioning methods
  – and extend CA approach to other classes of preconditioners

• Parallel Implementations
  – Performance tests

• Improving stability
  – Extended precision

• Auto-tuning work
  – Incorporation of Matrix Powers into pOSKI (Jong-Ho Byun, et al., UCB)
  – Code generation for Matrix Powers (collaborating with Ras Bodik, Michelle Strout)
  – Exploring co-tuning for CA-KSMS (i.e., Matrix Powers and TSQR)

• Looking forward: how do Communication-Avoiding algorithms relate to energy efficiency?
Thank you!

Questions?
Extra Slides
CA-BiCG

For $j = 0, 1, \ldots$, until convergence, Do

\[ \alpha_j = \frac{\langle r_j, r_j^* \rangle}{\langle A p_j, r_j \rangle} \]
\[ x_{j+1} = x_j + \alpha_j p_j \]
\[ r_{j+1} = r_j - \alpha_j A p_j \]
\[ r_j^* = r_j^* - \alpha_j A^T p_j^* \]
\[ \beta_j = \frac{\langle r_{j+1}, r_{j+1}^* \rangle}{\langle r_j, r_j^* \rangle} \]
\[ p_{j+1} = r_{j+1} + \beta_j p_j \]
\[ p_j^* = r_j^* + \beta_j p_j^* \]

EndDo

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

Compute four $N \times (s+1)$ matrices via $2 \times 2$ matrix powers kernel, and change-of-basis matrix $B$

\[ P = [p_{sk}, Ap_{sk}, \ldots, A^s p_{sk}] \]
\[ R = [r_{sk}, Ar_{sk}, \ldots, A^s r_{sk}] \]
\[ P^* = [p_{sk}^*, A^T p_{sk}^*, \ldots, (A^T)^s p_{sk}^*] \]
\[ R^* = [r_{sk}^*, A^T r_{sk}^*, \ldots, (A^T)^s r_{sk}^*] \]

Compute the $(2s+2) \times (2s+2)$ Gram matrix

\[ G = \begin{bmatrix} (P^*)^T & (R^*)^T \end{bmatrix} \begin{bmatrix} P & R \end{bmatrix} \]

Compute the $(2s+2) \times (s+1)$ coefficient matrices

\[ c_{sk} = \begin{bmatrix} 1 & B \\ 0_{s \times 1} & 0_{(s+1) \times (s+1)} \end{bmatrix}, \quad d_{sk} = \begin{bmatrix} 0_{(s+1) \times (s+1)} & B \\ 1 & 0_{s \times 1} \end{bmatrix} \]

For $j = 0$ to $s-1$, Do

\[ \alpha_{sk+j} = \frac{(d_{sk+j}^0)^T G d_{sk+j}^0}{(c_{sk+j}^0)^T G c_{sk+j}^1} \]
\[ x_{sk+j+1} = x_{sk+j} + \alpha_{sk+j} p_{sk+j} \]
\[ r_{sk+j+1} = r_{sk+j} - \alpha_{sk+j} [P, R] c_{sk+j}^1 \]
\[ r_{sk+j+1}^* = r_{sk+j}^* - \alpha_{sk+j} [P^*, R^*] c_{sk+j}^1 \]
\[ d_{sk+j+1} = d_{sk+j}^{0:-(s-j-1)} - \alpha_{sk+j} c_{sk+j}^{1:(s-j)} \]
\[ \beta_{sk+j} = \frac{(d_{sk+j+1}^0)^T G d_{sk+j+1}^0}{(d_{sk+j}^0)^T G d_{sk+j}^0} \]
\[ p_{sk+j+1} = r_{sk+j+1} + \beta_{sk+j} p_{sk+j} \]
\[ p_{sk+j+1}^* = r_{sk+j+1}^* + \beta_{sk+j} p_{sk+j}^* \]
\[ c_{sk+j+1} = d_{sk+j+1} + \beta_{sk+j} c_{sk+j}^{0:-(s-j-1)} \]

EndDo

EndDo
ALGORITHM 7.3: Biconjugate Gradient (BCG)

1. Compute $r_0 := b - Ax_0$. Choose $r_0^*$ such that $(r_0, r_0^*) \neq 0$.
2. Set, $p_0 := r_0$, $p_0^* := r_0^*$
3. For $j = 0, 1, \ldots$, until convergence Do:
   4. $\alpha_j := (r_j, r_j^*)/(Ap_j, p_j^*)$
   5. $x_{j+1} := x_j + \alpha_j p_j$
   6. $r_{j+1} := r_j - \alpha_j Ap_j$
   7. $r_{j+1}^* := r_j^* - \alpha_j A^T p_j^*$
   8. $\beta_j := (r_{j+1}, r_{j+1}^*)/(r_j, r_j^*)$
   9. $p_{j+1} := r_{j+1} + \beta_j p_j$
10. $p_{j+1}^* := r_{j+1}^* + \beta_j p_j^*$
11. EndDo

(Saad, 2000)
ALGORITHM 7.5: Conjugate Gradient Squared

1. Compute $r_0 := b - Ax_0; r_0^*$ arbitrary.
2. Set $p_0 := u_0 := r_0$.
3. For $j = 0, 1, 2 \ldots$, until convergence Do:
4. \[ \alpha_j = (r_j, r_0^*) / (Ap_j, r_0^*) \]
5. \[ q_j = u_j - \alpha_j Ap_j \]
6. \[ x_{j+1} = x_j + \alpha_j (u_j + q_j) \]
7. \[ r_{j+1} = r_j - \alpha_j A(u_j + q_j) \]
8. \[ \beta_j = (r_{j+1}, r_0^*) / (r_j, r_0^*) \]
9. \[ u_{j+1} = r_{j+1} + \beta_j q_j \]
10. \[ p_{j+1} = u_{j+1} + \beta_j (q_j + \beta_j p_j) \]
11. EndDo

(Saad, 2000)
ALGORITHM 7.6: BICGSTAB

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

(Saad, 2000)
Communication-Avoiding Krylov Subspace Methods

- Tall Skinny QR
- Matrix Powers Kernel
- Choice of Basis
- Stability and Roundoff Error
- Preconditioning

(Algorithms)

(Numerical Analysis)
Algorithm Overview

• Initially assign $r$-vector of rankings $(a_1,\ldots, a_r)$, (sampled from exponential R.V., $\lambda = 1$) to each vertex $v$

• In each iteration (up to $s$), for each vertex $v$, take the coordinate-wise minima of the $r$-vectors reachable from $v$ (denoted $S(v)$, non-zeros in column of $A$ corresponding to $v$)

• Apply estimator:

$$\frac{r - 1}{\sum_{i=1}^{r} a_i}$$

• Intuition: lowest-ranked node in $S(v)$ is highly correlated with $|S(v)|$
  – Example: If $S(v)$ contains half the nodes, we expect the lowest rank of nodes in $S(v)$ is very small.

$$\text{Prob}(|\hat{T} - T| \geq \epsilon T) = O\left(\frac{1}{\epsilon \sqrt{r}}\right)$$

where $T$ is the actual size of the transitive closure, $r$ is the number of randomized rankings per vector
\[ r_2(i) = \min (r_1(i), r_2(i), r_3(i)) \]
\[ r_4(i) = \min( r_3(i), r_4(i), r_5(i) ) \]
$r_3(i) = \min(r_2(i), r_3(i), r_4(i))$
Preliminary Experiments

• Set of small test matrices from UFSMC [Davis ‘94]

• $tol = 0.5$ (half-dense), 4 parts, $s \in \{2, 3, 4\}$ depending on fill in $A^s$

• Comparison of hypergraph size and communication volume for four strategies:
  – $s$-level column nets
  – Sparsified column nets (somewhere between $s$- and 1-level)
  – 1-level column nets
  – Graph partitioning ($A+A^T$)

• Software: PaToH [Catalyurek, Aykanat, ‘99] and Metis [Karypis, Kumar ‘98]
<table>
<thead>
<tr>
<th>Matrix</th>
<th>Application</th>
<th>( n )</th>
<th>( \text{nnz} )</th>
<th>Spy plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>arc130</td>
<td>Materials Science</td>
<td>130</td>
<td>1037</td>
<td></td>
</tr>
<tr>
<td>west0132</td>
<td>Chemical Engineering</td>
<td>132</td>
<td>413</td>
<td></td>
</tr>
<tr>
<td>str_0</td>
<td>LP</td>
<td>363</td>
<td>2454</td>
<td></td>
</tr>
<tr>
<td>gre_343</td>
<td>Directed graph</td>
<td>343</td>
<td>1032</td>
<td></td>
</tr>
<tr>
<td>mcca</td>
<td>Astrophysics</td>
<td>180</td>
<td>2659</td>
<td></td>
</tr>
<tr>
<td>rw496</td>
<td>Markov Chain Model</td>
<td>496</td>
<td>1859</td>
<td></td>
</tr>
<tr>
<td>str_200</td>
<td>LP</td>
<td>363</td>
<td>3068</td>
<td></td>
</tr>
</tbody>
</table>
nnz (workload) for Various Partitioning Strategies

- $A^s$
- $A$ Sparse
- $A$
- $A + A^T$

Test Matrices:
- arc130
- west0132
- str_0
- gre_343
- mcca
- rw496
- str_200

Y-axis: nnz
X-axis: Test Matrices
Normalized Communication Volume for Various Partitioning Strategies

- $A^s$
- $A$ Sparse
- $A$
- $A + A^T$

Test Matrices:
- arc130
- west0132
- str_0
- gre_343
- mcca
- rw496
- str_200

Normalized Communication Volume (PaToH)
Results and Observations

• Sparsified nets lead to comparable partition quality for significantly reduced hypergraph size

• Tuning parameter $tol$ gives flexibility to trade off:
  – Quality of partition
  – Computation and storage costs