Communication avoiding algorithms in linear algebra

Laura Grigori
ALPINES
INRIA Rocquencourt - LJLL, UPMC
France

Plan

• Motivation
• Selected past work on reducing communication
• Communication complexity of linear algebra operations
• Communication avoiding for dense linear algebra
  • LU, QR, Rank Revealing QR factorizations
  • Often not in ScalAPACK or LAPACK (YET !)
  • Algorithms for multicore processors
• Communication avoiding for sparse linear algebra
  • Iterative methods and preconditioning
• Conclusions

The role of numerical linear algebra

• Challenging applications often rely on solving linear algebra problems
• Linear systems of equations
  Solve $Ax = b$, where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^n$
  • Direct methods
    $PA = LU$, then solve $P^T L U x = b$
    Backward stability of LU factorization depends on $g_W$:
    \[ \|L \|_\infty \leq (1 + 2(n^2 - n)g_W) \|A\|_\infty, \]
    where $g_W = \max_{i,j,k} |a_{ij}^k| / \|a_{ij}\|_\infty$
  • Iterative methods
    • Find a solution $x_k$ from $x_0 + K_k (A, r_o)$, where $K_k (A, r_o) = \text{span} \{ r_o, A r_o, ..., A^{k-1} r_o \}$ such that the Petrov-Galerkin condition $b - A x_k \perp L_k$ is satisfied, where $L_k$ is a subspace of dimension $k$ and $r_o = A x_0 - b$.
    • Convergence depends on $\kappa(A)$ and the eigenvalue distribution (for SPD matrices).

Data driven science

Numerical simulations require increasingly computing power as data sets grow exponentially

Figures from astrophysics:
• Produce and analyze multi-frequency 2D images of the universe when it was 5% of its current age.
• COBE (1989) collected 10 gigabytes of data, required 1 Teraflop per image analysis.
• PLANCK (2010) produced 1 terabyte of data, requires 100 Petaflops per image analysis.
• Future experiment (2020) estimated to collect .5 petabytes, require 100 Exaflops per image analysis.
  Source: J. Borrill, LBNL, R. Stompor, Paris 7
Least Square (LS) Problems

- Given $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, solve $\min_{x} \|Ax - b\|_2$.
- Any solution of the LS problem satisfies the normal equations: $A^T A x = A^T b$.
- Given the QR factorization of $A$
  
  $A = QR$

  where $R$ is $n \times n$ upper triangular matrix

  $Q$ is $m \times m$ orthogonal matrix

  if $\text{rank}(A) = \text{rank}(R) = n$, then the LS solution is given by $x = (Q^T b)[:, 1:n]$.
- The QR factorization is column-wise backward stable
  
  \[
  \left\| A - \hat{Q}\hat{R} \right\|_F \leq \sqrt{n} \sigma_n \left( A \right) \left\| A(:, j) \right\|_2
  \]

  where $0 \leq j \leq n$.

Motivation - the communication wall

- Runtime of an algorithm is the sum of:
  - $\#\text{flops} \times \text{time per flop}$
  - $\#\text{words moved} / \text{bandwidth}$
  - $\#\text{messages} \times \text{latency}$
- Time to move data $>>$ time per flop
  - Gap steadily and exponentially growing over time

<table>
<thead>
<tr>
<th>Annual improvements</th>
<th>Bandwidth</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time/flop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>26%</td>
<td>15%</td>
</tr>
<tr>
<td>DRAM</td>
<td>23%</td>
<td>5%</td>
</tr>
</tbody>
</table>

- Performance of an application is less than 10% of the peak performance

  “We are going to hit the memory wall, unless something basic changes”  
  
  [W. Wulf, S. McKee, 95]

Rank revealing factorizations

- A rank revealing QR (RRQR) factorization is given as

  $A \Pi = QR \left( \begin{array}{cc} R_{11} & R_{12} \\ R_{21} & R_{22} \end{array} \right)$, $R_{11}$ is $k \times p(k,n)$

  with $\sigma_{\text{min}}(R_{11}) \geq \frac{\sigma_n(A)}{p(k,n)}$, $\sigma_{\text{min}}(R_{22}) \leq \sigma_{\text{min}}(A)$ $p(k,n)$

  $p(k,n)$ is a low degree polynomial in $n$ and $k$, $R_{11}$ is well conditioned, $\left\| R_{22} \right\|_2$ is small.

- Since $\sigma_{\text{min}}(A) \leq \sigma_{\text{min}}(R_{22}) = \left\| R_{22} \right\|_2$, the numerical rank of $A$ is $k$.

- $Q(:,1:k)$ forms an approximate orthogonal basis for the range of $A$.

- $R_{22}$ forms approximate null vectors.

- Applications: subset selection and linear dependency analysis, rank determination, low rank approximation - solve $\min_{\text{rank}(X) = k} \| A - X \|_2$.

- The communication problem needs to be taken into account higher in the computing stack

- A paradigm shift in the way the numerical algorithms are devised is required

- Communication avoiding algorithms - a novel perspective for numerical linear algebra
  - Minimize volume of communication
  - Minimize number of messages
  - Minimize over multiple levels of memory/parallelism
  - Allow redundant computations (preferably as a low order term)
Previous work on reducing communication

• **Tuning**
  - Overlap communication and computation, at most a factor of 2 speedup

• **Ghosting**
  - Store redundantly data from neighboring processors for future computations

• **Scheduling**
  - Block algorithms for linear algebra
    - Barron and Swinnerton-Dyer, 1960
    - ScaLAPACK, Blackford et al 97
  - Cache oblivious algorithms for linear algebra
    - Gustavson 97, Toledo 97, Frens and Wise 03, Ahmed and Pingali 00

Communication Complexity of Dense Linear Algebra

• Matrix multiply, using $2n^3$ flops (sequential or parallel)
  - Lower bound on Bandwidth = $\Omega \left( \frac{\#\text{flops}}{M^{1/2}} \right)$
  - Lower bound on Latency = $\Omega \left( \frac{\#\text{flops}}{M^{3/2}} \right)$

• Same lower bounds apply to LU using reduction
  - Demmel, LG, Hoemmen, Langou 2008

$$
\begin{pmatrix}
  I & -B \\
  A & I \\
  I & I
\end{pmatrix} =
\begin{pmatrix}
  I & -B \\
  A & I \\
  I & I
\end{pmatrix}
\begin{pmatrix}
  I & I \\
  A & I \\
  I & I
\end{pmatrix}
$$

• And to almost all direct linear algebra [Ballard, Demmel, Holtz, Schwartz, 09]

Communication in CMB data analysis

• **Map-making problem**
  - Find the best map $x$ from observations $d$, scanning strategy $A$, and noise $N^{-1}$
  - Solve generalized least squares problem involving sparse matrices of size $10^{12}$-by-$10^{7}$

• **Spherical harmonic transform (SHT)**
  - Synthesize a sky image from its harmonic representation
  - Computation over rows of a 2D object (summation of spherical harmonics)
  - Communication to transpose the 2D object
  - Computation over columns of the 2D object (FFTs)

2D Parallel algorithms and communication bounds

• If memory per processor = $n^2 / P$, the lower bounds become
  \[
  \#\text{words}_\text{moved} \geq \Omega \left( \frac{n^2}{P^{1/2}} \right), \quad \#\text{messages} \geq \Omega \left( \frac{P}{P^{1/2}} \right)
  \]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Minimizing #words (not #messages)</th>
<th>Minimizing #words and #messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesky</td>
<td>ScaLAPACK</td>
<td>ScaLAPACK</td>
</tr>
<tr>
<td>LU</td>
<td>ScaLAPACK</td>
<td>[LG, Demmel, Xiang, 08]</td>
</tr>
<tr>
<td></td>
<td>uses partial pivoting</td>
<td>[Khabou, Demmel, LG, Gu, 12]</td>
</tr>
<tr>
<td>QR</td>
<td>ScaLAPACK</td>
<td>[Demmel, LG, Hoemmen, Langou, 08]</td>
</tr>
<tr>
<td>RRQR</td>
<td>ScaLAPACK</td>
<td>[Demmel, LG, Gu, Xiang 13]</td>
</tr>
</tbody>
</table>

• Only several references shown, block algorithms (ScaLAPACK) and communication avoiding algorithms

Page 9

Page 11
LU factorization (as in ScaLAPACK pdgetrf)

LU factorization on a P = P_r x P_c grid of processors

For ib = 1 to n-1 step b

\[ A^{(ib)}_n = A_{ib:n, ib:n} \]

1. Compute panel factorization
   - find pivot in each column, swap rows
2. Apply all row permutations
   - broadcast pivot information along the rows
   - swap rows at left and right
3. Compute block row of U
   - broadcast right diagonal block of L of current panel
4. Update trailing matrix
   - broadcast right block column of L
   - broadcast down block row of U

#messages

\[ O(n \log_2 P_r) \]

\[ O(n/b(\log_2 P_c + \log_2 P_r)) \]

\[ O(n/b \log_2 P_c) \]

\[ O(n/b(\log_2 P_c + \log_2 P_r)) \]

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 \( \propto b \log P \)

Communication Avoiding Algorithm
- Reduction operation, with QR as operator
- Number of messages \( \propto \log P \)

\[ W = \begin{bmatrix} W_0 & W_1 & W_2 & W_3 \end{bmatrix} \rightarrow \begin{bmatrix} R_{00} & R_{10} & R_{20} & R_{30} \end{bmatrix} \]

\[ \rightarrow \begin{bmatrix} R_{01} & R_{02} \end{bmatrix} \]

Dual Core: W = \[
\begin{bmatrix} W_0 & W_1 & W_2 & W_3 \end{bmatrix} \]

Reduction tree will depend on the underlying architecture, could be chosen dynamically

Page 13

Flexibility of TSQR and CAQR algorithms

Parallel: \[ W = \begin{bmatrix} W_0 & W_1 & W_2 & W_3 \end{bmatrix} \rightarrow \begin{bmatrix} R_{00} & R_{10} & R_{20} & R_{30} \end{bmatrix} \rightarrow \begin{bmatrix} R_{01} & R_{11} \end{bmatrix} \rightarrow \begin{bmatrix} R_{02} \end{bmatrix} \]

Sequential: \[ W = \begin{bmatrix} W_0 & W_1 & W_2 & W_3 \end{bmatrix} \rightarrow \begin{bmatrix} R_{00} & R_{10} & R_{20} & R_{30} \end{bmatrix} \rightarrow \begin{bmatrix} R_{01} & R_{11} \end{bmatrix} \rightarrow \begin{bmatrix} R_{02} \end{bmatrix} \]

Modeled Speedups of CAQR vs ScaLAPACK

Petascale up to 22.9x
IBM Power 5 up to 9.7x
“Grid” up to 11x

Petascale machine with 8192 procs, each at 500 GFlops/s, a bandwidth of 4 GB/s.

\[ \gamma = 2 \cdot 10^{-5} s, \alpha = 10^{-5} s, \beta = 2 \cdot 10^{-5} s/word. \]
Obvious generalization of TSQR to LU

• Block parallel pivoting:
  • uses a binary tree and is optimal in the parallel case
  \[
  W = \begin{bmatrix}
  W_0 & U_{00} & U_{01} & U_{02} \\
  W_1 & U_{10} & & \\
  W_2 & U_{20} & U_{11} & \\
  W_3 & & & \\
  \end{bmatrix}
  \]

• Block pairwise pivoting:
  • uses a flat tree and is optimal in the sequential case
  • introduced by Barron and Swinnerton-Dyer, 1960: block LU factorization used to solve a system with 100 equations on EDSAC 2 computer using an auxiliary magnetic-tape
  • used in PLASMA for multicore architectures and FLAME for out-of-core algorithms and for multicore architectures

Page 17

Stability of the LU factorization

• The backward stability of the LU factorization of a matrix \( A \) of size \( n \)-by-\( n \)
  \[
  \|L\| \|U\| \leq (1 + 2(n^2 - n)g_{max}) \|A\|
  \]
  depends on the growth factor
  \[
  g_w = \max_{i,j,k} \left| a_{ij}^k \right| \quad \text{where} \quad a_{ij}^k \text{are the values at the k-th step.}
  \]

• \( g_w \leq 2^{n-1} \), but in practice it is on the order of \( n^{2/3} \) -- \( n^{1/2} \)

• Two reasons considered to be important for the average case stability [Trefethen and Schreiber, 90] :
  - the multipliers in \( L \) are small,
  - the correction introduced at each elimination step is of rank 1.

Page 18

Block parallel pivoting

• Results shown for random matrices
• Will become unstable for large matrices

\[
W = \begin{bmatrix}
W_0 & U_{00} & U_{01} & U_{02} \\
W_1 & U_{10} & & \\
W_2 & U_{20} & U_{11} & \\
W_3 & & & \\
\end{bmatrix}
\]

Page 19

Block pairwise pivoting

• Unstable for large number of processors \( P \)
• When \( P=\text{number rows} \), it corresponds to parallel pivoting, known to be unstable (Trefethen and Schreiber, 90)

Page 20
Tournament pivoting - the overall idea

- At each iteration of a block algorithm
  \[ A = \begin{pmatrix} A_{11} & A_{21} \\ A_{21} & A_{22} \end{pmatrix} \]
  where \( W = \begin{pmatrix} A_{11} \end{pmatrix} \)

- Preprocess \( W \) to find at low communication cost good pivots for the LU factorization of \( W \), return a permutation matrix \( P \).
- Permute the pivots to top, ie compute \( PA \).
- Compute LU with no pivoting of \( W \), update trailing matrix.

\[ PA = \begin{pmatrix} L_{11} & U_{11} \\ L_{21} & I_{n-b} \end{pmatrix} \begin{pmatrix} U_{12} & \end{pmatrix} = A_{22} - L_{21}U_{12} \]

Growth factor for binary tree based CALU

- Random matrices from a normal distribution
- Same behaviour for all matrices in our test, and \( |L| \leq 4.2 \)

Stability of CALU (experimental results)

- Results show \( ||PA-LU||/||A|| \), normwise and componentwise backward errors, for random matrices and special ones
  - See [LG, Demmel, Xiang, SIMAX 2011] for details
  - BCALU denotes binary tree based CALU and FCALU denotes flat tree based CALU
Our “proof of stability” for CALU

- CALU as stable as GEPP in following sense: In exact arithmetic, CALU process on a matrix A is equivalent to GEPP process on a larger matrix G whose entries are blocks of A and zeros.

- Example of one step of tournament pivoting:

\[
A = \begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22} \\
A_{31} & A_{32}
\end{pmatrix}
\]

\[
G = \begin{pmatrix}
\tilde{A}_{11} & \tilde{A}_{12} \\
\tilde{A}_{21} & \tilde{A}_{22} \\
-\tilde{A}_{31} & \tilde{A}_{32}
\end{pmatrix}
\]

- Proof possible by using original rows of A during tournament pivoting (not the computed rows of U).

LU factorization and low rank matrices

- For low rank matrices, the factorization of A₁ can be computed as following might not be stable:

  Compute PA=LU by using GEPP
  L(k+1:end,k) = A(k+1:end,k)/A(k,k)
  Permute the matrix A
  Compute LU with no pivoting A₁=L₁U₁
  L(k+1:end,k) = L(k+1:end,k)* (1/A(k,k))

- Example A = randn(6,3)*randn(3,5), max(abs(L₁)) = 1, max(abs(L₁))₁⁻¹ = 10⁻¹⁵

After 4 steps of factorization of PA we obtain:

\[
P4₁ = \begin{pmatrix}
1.0000 & 0.1729 & 0.0661 & 0.5776 & 0.4789 & -0.3264 & 0.0000 & 0.8068 & 0.0543 & -0.7514 & 0.0000 & -0.3264 & 1.0000 & 2.3333 & 1.0000 & 0.0000 & 0.1545 & 2.3300 & 1.7778
\end{pmatrix}
\]

\[
P₄₁^{-1} = \begin{pmatrix}
-4.4766 & 3.0163 & -4.7290 & 4.2180 & -0.8164 & -1.5439 & -0.4703 & 1.9267 & 1.0925 & 1.6149 & 2.3623 & 0.3167 & 9.9e-16 & 1.6e-16 & 1 & 3.3e-16 & 8.3e-17
\end{pmatrix}
\]

Schur complement after 4 elimination steps:

\[
A₄₁^{-1} = \begin{pmatrix}
1.0000 & 0.1729 & 0.0661 & 0.5776 & 0.4789 & -0.3264 & 0.0000 & 0.8068 & 0.0543 & -0.7514 & 0.0000 & -0.3264 & 1.0000 & 2.3333 & 1.0000 & 0.0000 & 0.1545 & 2.3300 & 1.7778
\end{pmatrix}
\]

After 4 steps of factorization of A we obtain:

\[
A₄₁ = \begin{pmatrix}
-4.4766 & 3.0163 & -4.7290 & 4.2180 & -0.8164 & -1.5439 & -0.4703 & 1.9267 & 1.0925 & 1.6149 & 2.3623 & 0.3167 & 9.9e-16 & 1.6e-16 & 1 & 3.3e-16 & 8.3e-17
\end{pmatrix}
\]

\[
P₄₄₁ = \begin{pmatrix}
1.0000 & 0.1729 & 0.0661 & 0.5776 & 0.4789 & -0.3264 & 0.0000 & 0.8068 & 0.0543 & -0.7514 & 0.0000 & -0.3264 & 1.0000 & 2.3333 & 1.0000 & 0.0000 & 0.1545 & 2.3300 & 1.7778
\end{pmatrix}
\]

\[
P₄₄₁^{-1} = \begin{pmatrix}
-4.4766 & 3.0163 & -4.7290 & 4.2180 & -0.8164 & -1.5439 & -0.4703 & 1.9267 & 1.0925 & 1.6149 & 2.3623 & 0.3167 & 9.9e-16 & 1.6e-16 & 1 & 3.3e-16 & 8.3e-17
\end{pmatrix}
\]

Bound for F

\[
2^n \text{ for } O((m + \log n)n^2) \text{ extra flops}
\]

- f is a small constant
- b is the block size of the block RRQR

Growth factor in exact arithmetic

- Matrix of size m-by-n, reduction tree of height H=log(P).
- (CA)LU_PRRP select pivots using strong rank revealing QR (A. Khabou, J. Demmel, LG, M. Gu, SIMAX 2013)
- “In practice” means observed/expected/conjectured values.

### Comparison of Growth Factors

<table>
<thead>
<tr>
<th>DAU - PRRP</th>
<th>GEPP</th>
<th>CALU</th>
<th>CALU_PRRP</th>
<th>LU_PRRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper bound</td>
<td>$2^{n(\log(P)+1)-1}$</td>
<td>$2^{n-1}$</td>
<td>$(1+2b)^{(n/b)\log(P)}$</td>
<td>$(1+2b)^{(n/b)}$</td>
</tr>
<tr>
<td>In practice</td>
<td>$n^{2/3} - n^{1/2}$</td>
<td>$n^{3/2} - n^{1/2}$</td>
<td>$(n/b)^{2/3} - (n/b)^{1/2}$</td>
<td>$(n/b)^{2/3} - (n/b)^{1/2}$</td>
</tr>
</tbody>
</table>

Better bounds

- For a matrix of size $10^7$-by-$10^7$ (using petabytes of memory) $n^{1/2} = 10^{3.5}$
- When will Linpack have to use the QR factorization for solving linear systems?

Rank revealing QR factorization (RRQR)

- A RRQR factorization $A\Pi = QR = \begin{pmatrix} R_{i1} & R_{i2} \\ R_{j1} & R_{j2} \end{pmatrix}$ $R_{i1}$ is $k \times k$

satisfies

$$1 \leq \frac{\sigma_i(A)}{\sigma_i(R_{i1})}, \quad \frac{\sigma_j(R_{j2})}{\sigma_j(A)} \leq 1 + F^2(n-k),$$

for any $1 \leq i \leq k$ and $1 \leq j \leq \min(m,n) - k$

### Better Bounds

<table>
<thead>
<tr>
<th>QR with Column Pivoting</th>
<th>Strong RRQR (Gu, Eisenstat)</th>
<th>CA-RRQR Tournament pivoting</th>
</tr>
</thead>
<tbody>
<tr>
<td>2^n</td>
<td>$f$</td>
<td>$\sim (b+n)^{1/2}$ $(fb)^{(n/b)\log(n/b)}$</td>
</tr>
</tbody>
</table>
Communication avoiding RRQR

- Tournament pivoting used during panel factorization to select b columns.

- One step of CA-RRQR, $A=(A_0, A_1)$.

- Perform (strong) RRQR on each column block.
  \[ A_0 \Pi_{00} = Q_{00} R_{00} \] pick first b columns, form $A_{00}$
  \[ A_1 \Pi_{10} = Q_{10} R_{10} \] same for $A_{10}$

- Perform (strong) RRQR on $(A_{00}, A_{10})$.
  \[ (A_{00}, A_{10}) \Pi_{10} = Q_{10} R_{10} \] pick b columns, form $A_{01}$

- Permute $A_{01}$ in front, compute QR with no pivoting

Demmel, LG, Gu, Xiang 2013.

CA-RRQR experimental results

- Performed in the context of QLP decomposition (Stewart)
  \[ A \Pi = ca - rrqr(Q, R) \]
  \[ R^T \Pi L = cagqr(P, L^T) \]

- Devil’s stairs matrix (P. Stewart) with multiple gaps in the singular values
- Singular values computed using DGESVJ (Z. Drmac and K. Veselic, 08)

---

Performance vs ScaLAPACK

- Parallel TSLU (LU on tall-skinny matrix)
  - IBM Power 5
    - Up to 4.37x faster (16 procs, 1M x 150)
  - Cray XT4
    - Up to 5.52x faster (8 procs, 1M x 150)

- Parallel CALU (LU on general matrices)
  - Intel Xeon (two socket, quad core)
    - Up to 2.33x faster (8 cores, 10^6 x 500)
  - IBM Power 5
    - Up to 2.29x faster (64 procs, 1000 x 1000)
  - Cray XT4
    - Up to 1.81x faster (64 procs, 1000 x 1000)

- Details in SC08 (LG, Demmel, Xiang), IPDPS’10 (S. Donfack, LG).

Lightweight scheduling for CALU

- Static scheduling
- Static + 10% dynamic scheduling
- 100% dynamic scheduling

Donfack, LG, Gropp, Kale, IPDPS 2012
Plan

- Motivation
- Selected past work on reducing communication
- Communication complexity of linear algebra operations
- Communication avoiding for dense linear algebra
  - LU, LU_PRRP, OR, Rank Revealing QR factorizations
  - Often not in ScalAPACK or LAPACK
  - Algorithms for multicore processors
- Communication avoiding for sparse linear algebra
  - Iterative methods and preconditioning
- Conclusions

Minimizing communication in iterative solvers

- To minimize communication
  - Generate a set of s vectors \((Ab, A^2b, \ldots, A^sb)\)
  - Orthogonalize the s vectors, check convergence
  - \(O(\log P)\) messages, optimal

However

- Important instability problem to address (monomial basis)
- CA-preconditioners to further decrease the number of iterations

Communication in Krylov subspace methods

**Iterative methods to solve \(Ax = b\)**

- Find a solution \(x_k\) from \(x_0 + K_k(A, r_0)\), where \(K_k(A, r_0) = \text{span} \{r_0, Ar_0, \ldots, A^{k-1}r_0\}\) such that the Petrov-Galerkin condition \(b - Ax_k \perp L_k\) is satisfied.

- For numerical stability, an orthonormal basis \(\{q_1, q_2, \ldots, q_k\}\) for \(K_k(A, r_0)\) is computed (CG, GMRES, BiCGstab,…)

- Each iteration requires
  - Sparse matrix vector product
  - Dot products for the orthogonalization process

- **S-step Krylov subspace methods**
  - Unroll s iterations, orthogonalize every s steps

Research opportunities and limitations

**Length of the basis “s” is limited by**

- Size of ghost data
- Loss of precision

Here is a cost table for a 3D regular grid, 7 pt stencil:

<table>
<thead>
<tr>
<th>Method</th>
<th>Memory Cost</th>
<th>Flops Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMRES</td>
<td>(O(s n/P))</td>
<td>(O(s n/P))</td>
</tr>
<tr>
<td>CA-GMRES</td>
<td>(O(s n/P) + O(s^2 (n/P)^{2/3}))</td>
<td>(O(s n/P) + O(s^2 (n/P)^{2/3}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Preconditioners: few identified so far to work with s-step methods

- Highly decoupled preconditioners: Block Jacobi
- Hierarchical, semiseparable matrices (M. Hoemmen, J. Demmel)

A look at three classes of preconditioners

- Incomplete LU factorizations (joint work with S. Moufawad)
- Two level preconditioners in DDM
- Deflation techniques through preconditioning
ILU0 with nested dissection and ghosting

Let \( a_j \) be the set of equations to be solved by one processor.
For \( j = 1 \) to \( s \) do
Find \( f_j = \text{ReachableVertices} \ (G(U), \ a_j) \)
Find \( \gamma_j = \text{ReachableVertices} \ (G(L), \ f_j) \)
Find \( \delta_j = \text{Adj} \ (G(A), \ \gamma_j) \)
Set \( a_j = \delta_j \)
end

Ghost data required:
\[ x(\gamma_j), A(\gamma_j, \ \delta_j), L(\delta_j, \ \gamma_j), U(\delta_j, \ \delta_j) \]

\( \Rightarrow \) Half of the work performed on one processor.

CA-ILU0 with AMML reordering and ghosting

- Reduce volume of ghost data by reordering the vertices using Alternating Min-Max Layers (AMML) reordering:
  - First number the vertices at odd distance from the separators
  - Then number the vertices at even distance from the separators
- CA-ILU0 computes a standard ILU0 factorization

Comparison with Block Jacobi

- Block Jacobi is another preconditioner which does not require communication
- Tests for a boundary value problem (Achdou, Nataf), 40x40x40 grid

\[ -\text{div}(\kappa(x)\nabla u) = f \quad \text{in} \Omega \]
\[ u = 0 \quad \text{on} \partial D \]
\[ \frac{\partial u}{\partial n} = 0 \quad \text{on} \partial N \]
\[ \Omega = [0,1]^3, \ \partial D = \partial D^0 \bigcup \partial D^1 \]
\[ \kappa \text{ jumps from } 1 \text{ to } 10^3 \]

Methods tested:
- Natural ordering NO+ILU0
- CA-ILU0 - k-way+AMML(1)+ILU0
- Block Jacobi using LU - BJ+ILU0
- Block Jacobi using ILU0 - BJ-ILU0

Challenge in getting scalable preconditioners

Many preconditioners (as ILU) have plateaus in the convergence, often due to the presence of few low eigenvalues

Direction preserving factorization

- Preconditioner \( M \) satisfies a filtering property
  \[ MT = AT \quad \text{or} \quad T^T M = T^T A \]
- Filtering vectors \( T \) are chosen to improve the convergence

Block Filtering (BFD) and Nested Filtering (NFF) Preconditioners
R. Fezzani, LG, P. Kumar, R. Lacroix, F. Nataf, L. Qu, K. Wang
- Algebraic preconditioners based on nested dissection and block/nested factorization
- Every Schur complement is approximated to satisfy the filtering property:
  \[ L_{ik} D_{ij} U_{kj} t = L_{ik} F_{kj} U_{kj} t, \text{ e.g. } F_{kj} = \text{Diag}(D_{ik} U_{kj} t) (U_{kj} t) \]
Preserving directions of interest

- **Pointwise approximate factorization satisfying a row-sum criteria**, Dupont, Kendall, and Rachford (1968), Gustafsson (1978)
  - Improves the condition number of the preconditioned matrix for matrices arising from finite difference approximation of second order elliptic equations
- **Nested factorization**, Appleyard, Cheshire (1983)
  - If \( t^r \mathbf{r}_0 = 0 \), then at any iteration \( t^r \mathbf{r}_k = 0 \), this ensures a mass conservation property
- **Direction preserving semiseparable approximation of SPD matrices**, Gu, Li, Vassilevski (2010)
  - If the near null-space of the original fine grid matrix is preserved, then view the preconditioner as a coarse discretization matrix
  - Conditioning analysis performed by Napov, components dropped are orthogonal to components preserved
- **Multigrid methods**
  - Bootstrap AMG (Brandt, Brannick, Kahl, and Livshits)

Results for a boundary value problem

- SKY (provided by Achdou, Nataf), discretized on a 400x400x400 grid (64 millions unknowns, 447 millions nonzeros)
- \[-\text{div}(\mathbf{x}(x)\nabla u) = f \text{ in } \Omega\]
- \( u = 0 \text{ on } \partial \Omega_0 \)
- \( \frac{\partial u}{\partial n} = 0 \text{ on } \partial \Omega_N \)
- \( \Omega = [0,1]^3, \Omega_N = \partial \Omega \setminus \partial \Omega_0 \)
- Tests use GMRES (PETSc), tolerance = 10^{-8}

Comparison with Restricted Additive Schwarz (RAS)

Settings:
- Curie supercomputer based on Bullx system, nodes composed of two eight-core Intel Sandy Bridge.
- Subdomains solved using Pardiso, separators solved using MUMPS.
- GMRES and RAS from PETSc.

Conclusions

- Introduced a new class of communication avoiding algorithms that minimize communication
  - Attain theoretical lower bounds on communication
  - Minimize communication at the cost of redundant computation
  - Are often faster than conventional algorithms in practice
- Remains a lot to do for sparse linear algebra
  - Communication bounds, communication optimal algorithms
  - Numerical stability of s-step methods
  - Alternatives as block iterative methods, pipelined iterative methods
  - Preconditioners - limited by memory and communication, not flops
- And BEYOND

Best student paper finalist, Qu, LG, Nataf, SC’13
Conclusions

- Many previous results
  - Only several cited, many references given in the papers
  - Flat trees algorithms for QR factorization, called tiled algorithms used in the context of
    - Out of core - Gunter, van de Geijn 2005

Collaborators, funding

Collaborators:
- A. Branescu, INRIA, S. Donfack, INRIA, A. Khabou, INRIA, M. Jacquelin, INRIA, S. Moufawad, INRIA, H. Xiang, University Paris 6
- J. Demmel, UC Berkeley, B. Gropp, UIUC, M. Gu, UC Berkeley, M. Hoemmen, UC Berkeley, J. Langou, CU Denver, V. Kale, UIUC

Funding: ANR Petal and Petalh projects, ANR Midas, Digiteo Xscale NL, COALA INRIA funding

Further information:
http://www-rocq.inria.fr/who/Laura.Grigori/

References

Results presented from:

Parallel TSQR

References: Golub, Plemmons, Sameh 88, Pothen, Raghavan, 89, Da Cunha, Becker, Patterson, 02
LU_PRRP: LU with panel rank revealing pivoting

- Pivots are selected by using strong rank revealing QR on each panel.
- The factorization after one panel elimination is written as

\[ PA = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I_b \\ A_{21}A_{11}^{-1} \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{22} - A_{21}A_{11}^{-1}A_{12} \end{pmatrix} \]

\( A_{21}A_{11}^{-1} \) is computed through strong rank revealing QR and
\[ \max(|A_{21}A_{11}^{-1}|) \leq \frac{\gamma}{g} \]

- LU_PRRP and CALU_PRRP stable for pathological cases and matrices from two real applications (Volterra integral equation - Foster, a boundary value problem - Wright) on which GEPP fails.

A. Khabou, J. Demmel, LG, M. Gu, 2012

CA-ILU0: numerical experiments

- CA-ILU0 computes a standard ILU0 preconditioner, but with a different ordering.
- Convergence is similar to ILU0 with nested dissection ordering.
  - Results presented for a 3D Matrix issued from a triphasic Black Oil model (elliptic behavior of the pressure block).

<table>
<thead>
<tr>
<th>Ordering</th>
<th>Rel. residual</th>
<th>Error</th>
<th>No of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>E-8</td>
<td>E-9</td>
<td>55</td>
</tr>
<tr>
<td>ND 16</td>
<td>E-8</td>
<td>E-9</td>
<td>83</td>
</tr>
<tr>
<td>ND 64</td>
<td>E-7</td>
<td>E-9</td>
<td>85</td>
</tr>
<tr>
<td>ND 128</td>
<td>E-8</td>
<td>E-9</td>
<td>86</td>
</tr>
<tr>
<td>ND 256</td>
<td>E-8</td>
<td>E-9</td>
<td>78</td>
</tr>
<tr>
<td>AND 16</td>
<td>E-7</td>
<td>E-9</td>
<td>75</td>
</tr>
<tr>
<td>AND 64</td>
<td>E-8</td>
<td>E-9</td>
<td>82</td>
</tr>
<tr>
<td>AND 128</td>
<td>E-8</td>
<td>E-9</td>
<td>83</td>
</tr>
<tr>
<td>AND 256</td>
<td>E-8</td>
<td>E-9</td>
<td>85</td>
</tr>
</tbody>
</table>

Tournament pivoting for a tall skinny matrix

1) Compute GEPP factorization of each \( W_i \), find permutation \( \Pi_i \)

\[
W = \begin{pmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \end{pmatrix} = \begin{pmatrix} \Pi_{i0} \Pi_{10} L_{00} U_{00} \\ \Pi_{i1} \Pi_{11} L_{10} U_{10} \\ \Pi_{i2} \Pi_{12} L_{20} U_{20} \\ \Pi_{i3} \Pi_{13} L_{30} U_{30} \end{pmatrix}
\]

Pick b pivot rows, form \( A_{00} \)

Same for \( A_{10} \)

Same for \( A_{20} \)

Same for \( A_{30} \)

2) Perform \( \log_2(P) \) times GEPP factorizations of 2b-by-b rows, find permutations \( \Pi_i, \Pi_j \)

\[
\begin{pmatrix} A_{00} \\ A_{10} \\ A_{20} \end{pmatrix} = \begin{pmatrix} \Pi_{i0} \Pi_{10} L_{00} U_{00} \\ \Pi_{i0} \Pi_{10} L_{10} U_{10} \\ \Pi_{i0} \Pi_{10} L_{20} U_{20} \end{pmatrix}
\]

Pick b pivot rows, form \( A_{01} \)

Same for \( A_{11} \)

\[
\begin{pmatrix} A_{01} \\ A_{11} \end{pmatrix} = \begin{pmatrix} \Pi_{i0} \Pi_{10} L_{01} U_{01} \\ \Pi_{i0} \Pi_{10} L_{11} U_{11} \end{pmatrix}
\]

3) Compute LU factorization with no pivoting of the permuted matrix:

\[
\Pi_i^T \Pi_j^T \Pi_k W = LU
\]

Two level preconditioners

In the unified framework of (Tang et al. 09), let:

\[
P := I - A Q, \quad Q := Z E^{-1} Z^T, \quad E := Z^T A Z
\]

where
- \( M \) is the first level preconditioner (eg based on additive Schwarz)
- \( Z \) is the deflation subspace matrix of full rank
- \( E \) is the coarse grid correction, a small dense invertible matrix
- \( P \) is the deflation matrix

Examples of preconditioners:

\[ P_{\text{ADD}} = M^{-1} + Z E^{-1} Z^T, \quad P_{\text{ADEF2}} = P^T M^{-1} + Z E^{-1} Z^T \] (Mandel 1993)

- DDM - \( Z \) and \( Z^T \) are the restriction and prolongation operators based on subdomains, \( E \) is a coarse grid, \( P \) is a subspace correction
- Deflation - \( Z \) contains the vectors to be deflated
- Multigrid - interpretation possible
Two level preconditioners

\[ P_{ADD} \] for a Poisson-like problem, using \( Z \) defined as in (Nicolaiides 1987):

\[
Z = \begin{bmatrix}
1_{n_0} & 0 & \ldots & 0 \\
0 & 1_{n_1} & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1_{n_2}
\end{bmatrix}
\]

\( Z = \begin{bmatrix} Z \end{bmatrix} (A \chi) = \begin{bmatrix} Z \end{bmatrix} E^{-1} \begin{bmatrix} Z \end{bmatrix} ^T (A \chi) \]

Evolution of high performance architectures

- Computers get faster, but their architecture gets more complex
- First petascale system 2008, 1.33 Pflop/s
  - RoadRunner, IBM, LANL
  - A TriBlade formed by
    - Two dual-core Opterons with 16 GB of memory
    - Four PowerXCell 8i CPUs with 16 GB Cell RAM
    - A total of 13,824 Opteron cores + 116,640 Cell cores
  - Fastest supercomputer today, 17.59 Pflop/s
    - Titan, Cray XK7, ORNL, 18,688 compute nodes
    - 16 cores AMD Opteron, 32 GB of RAM memory per node
    - 560,640 processors including 261,632 NVIDIA K20x accelerator cores.

Lower bounds for linear algebra

- Computation modelled as an n-by-n-by-n set of lattice points \((i,j,k)\) represents the operation \( c(i,j) = f(g(a(i,k)*b(k,j))) \)
- The computation is divided in S phases
- Each phase contains exactly M (the fast memory size) load and store instructions
- Determine how many flops the algorithm can compute in each phase, by applying discrete Loomis-Whitney inequality:

\[
w^2 \leq N_A N_B N_C
\]

Lower bounds for matrix multiplication (contd)

- Discrete Loomis-Whitney inequality:

\[
w^2 \leq N_A N_B N_C
\]

- Since there are at most 2M elements of A, B, C in a phase, the bound is:

\[
w \leq 2\sqrt{2} M^{3/2}
\]

- The number of phases \( S \) is \#flops/w, and hence the lower bound on communication is:

\[
\frac{\#flops}{w} = \Omega \left( \frac{\#flops}{M^{1/2}} \right)
\]

\[
\frac{\#loads/stores}{w} = \Omega \left( \frac{\#flops}{M^{1/2}} \right)
\]