Communication Bounds for Sequential and Parallel Eigenvalue Problems

Grey Ballard¹, Jim Demmel¹, Ioana Dumitriu²

¹UC Berkeley
²University of Washington

April 19, 2010

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (CAREER Award #DIG07-10227), as well as the National Science Foundation (Award #DMS-0847661)
1. Communication-Avoiding Linear Algebra

2. Randomized Spectral Divide and Conquer
   - Divide and Conquer Algorithm
   - Randomized Bisection
   - Communication Costs
   - Numerical Experiments

3. Other Optimal (Deterministic) Algorithms

4. Conclusions
Motivation

By *communication* we mean

- moving data within memory hierarchy on a sequential computer
- moving data between processors on a parallel computer

Communication is expensive, so our goal is to minimize it

- some algorithms for one-sided factorizations attain lower bounds
- need new algorithms to solve eigenvalue problems optimally
Communication Cost Model

- Time required to perform one flop: $\gamma$

- Time required to transfer one message of $w$ words:
  \[ \alpha + \beta w \]
  - in the sequential case, a message is a contiguous block of memory

- $\alpha$ is latency+overhead cost, $\beta w$ is bandwidth cost

- Total running time of an algorithm (ignoring overlap):
  \[ \alpha \cdot (\# \text{ messages}) + \beta \cdot (\# \text{ words}) + \gamma \cdot (\# \text{ flops}) \]
Matrix Multiplication Lower Bounds

- Assume $O(n^3)$ algorithm (i.e. not Strassen-like)
- Sequential case with fast memory of size $M$
  - lower bound on words moved between fast/slow mem:
    \[ \Omega \left( \frac{n^3}{\sqrt{M}} \right) \]  
    \[ \text{[HK81]} \]
    - attained by blocked algorithm
- Parallel case with $P$ processors (local memory of size $M$)
  - lower bound on words communicated (along critical path):
    \[ \Omega \left( \frac{n^3}{P\sqrt{M}} \right) \]  
    \[ \text{[ITT04]} \]
- “2D” and “3D” algorithms:

<table>
<thead>
<tr>
<th></th>
<th>$M$</th>
<th>lower bound</th>
<th>attained by</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>$O\left(\frac{n^2}{P}\right)$</td>
<td>$\Omega\left(\frac{n^2}{\sqrt{P}}\right)$</td>
<td>[Can69]</td>
</tr>
<tr>
<td>3D</td>
<td>$O\left(\frac{n^2}{P^{2/3}}\right)$</td>
<td>$\Omega\left(\frac{n^2}{P^{2/3}}\right)$</td>
<td>[Joh93]</td>
</tr>
</tbody>
</table>
We extended the approach of [ITT04] to other algorithms:
- the rest of BLAS
- Cholesky, LU, QR factorizations
- Eigenvalue and SVD reductions
- sequences of algorithms (e.g. repeated matrix squaring)
- graph algorithms (e.g. all pairs shortest paths)
- to dense or sparse problems
- to sequential, hierarchical, or parallel cases

see [BDHS10] for details and proof

general lower bound:

\[
\text{# words} = \Omega \left( \frac{\text{# flops}}{\sqrt{\text{fast/local memory size}}} \right)
\]
Optimal Algorithms

Existing implementations in (Sca)LAPACK don’t attain lower bounds, but new algorithms do:

- Communication-Avoiding LU [GDX08]
  - Need to replace partial pivoting (but still stable)
  - Does $O(n^2)$ extra work

- Communication-Avoiding QR [DGHL08]
  - Need to represent output ($Q$) differently (architecture dependent)
  - Does $O(n^2)$ extra work
  - “Tall-skinny QR” used to factor panel

- Communication-Avoiding Rank-Revealing QR
  - Column-pivoting, but different pivot order than standard algorithm
1 Communication-Avoiding Linear Algebra

2 Randomized Spectral Divide and Conquer
   - Divide and Conquer Algorithm
   - Randomized Bisection
   - Communication Costs
   - Numerical Experiments

3 Other Optimal (Deterministic) Algorithms

4 Conclusions
Conventional algorithms for solving eigenvalue problems communicate too much
  - costs lie in reduction to condensed form and HessQR
  - these algorithms could be improved to reduce communication

Alternative: spectral divide-and-conquer approach
  - requires matrix multiplication, QR, and rank-revealing QR
  - deterministic rank-revealing QR algs also communicate too much

Randomized rank-revealing QR algorithm communicates less!

We also use randomization to choose how to divide the spectrum
Eigenvalue Algorithms on Sequential Machines

- Communication lower bounds

\[
\text{# words} = \Omega \left( \frac{n^3}{\sqrt{M}} \right) \quad \text{# messages} = \Omega \left( \frac{n^3}{M^{3/2}} \right)
\]

- Factors by which algorithms exceed lower bounds:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>New Algorithms</th>
<th>LAPACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># words</td>
<td># messages</td>
</tr>
<tr>
<td>Symm Eig/SVD</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Nonsymm Eig</td>
<td></td>
<td>Optimal!</td>
</tr>
</tbody>
</table>

- $M$ is fast memory size, $n$ is problem size
- New algorithms have same communication complexity as QR decomposition and achieve lower bound
Communication lower bounds

\[ \text{# words} = \Omega \left( \frac{n^2}{\sqrt{P}} \right) \quad \text{# messages} = \Omega \left( \sqrt{P} \right) \]

Factors by which algorithms exceed lower bounds:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>New Algorithms</th>
<th>ScaLAPACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># words</td>
<td># messages</td>
</tr>
<tr>
<td>Symm Eig/SVD</td>
<td>( O(\log P) )</td>
<td>( O(\log P) )</td>
</tr>
<tr>
<td>Nonsymm Eig</td>
<td>Nearly optimal!</td>
<td></td>
</tr>
</tbody>
</table>

\( P \) is number of processors, \( n \) is problem size

New algorithms have same communication complexity as QR decomposition and achieve lower bound (up to polylog factors)
History of Spectral Divide and Conquer

- Ideas go back to Bulgakov, Godunov, Malyshev [BG88], [Mal89]
- Bai, Demmel, Gu [BDG97]
  - reduced to matmul, QR, generalized QR with pivoting (bug)
- Demmel, Dumitriu, Holtz [DDH07]
  - instead of QR with pivoting, use RURV (randomized URV) (no bug)
  - requires matmul and QR, no column pivoting
- Demmel, Grigori, Hoemmen, Langou [DGHL08]
  - communication-optimal QR decomposition ("CAQR")
- New communication-optimal algorithm
  - use generalized RURV for better rank-detection than [DDH07]
  - use communication-optimal implementations for matrix multiplication and QR as subroutines
  - use randomization in divide and conquer
Overview of Algorithm

One step of divide and conquer:

1. Compute \( \left( I + (A^{-1})^{2^k} \right)^{-1} \) implicitly
   - maps eigenvalues of \( A \) to 0 and 1 (roughly)

2. Compute rank-revealing decomposition to find invariant subspace

3. Output block-triangular matrix

\[
A_{\text{new}} = U^* A U = \begin{bmatrix}
A_{11} & A_{12} \\
\varepsilon & A_{22}
\end{bmatrix}
\]

- block sizes chosen to minimize norm of \( \varepsilon \)
- eigenvalues of \( A_{11} \) all lie outside unit circle, eigenvalues of \( A_{22} \) lie inside unit circle, subproblems solved recursively
- stable, but progress guaranteed only with high probability
Implicit Repeated Squaring

\[ A_0 = A, \quad B_0 = I \]

Repeat

1. \[ \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \cdot \begin{bmatrix} R_j \\ 0 \end{bmatrix} = qr \left( \begin{bmatrix} B_j \\ -A_j \end{bmatrix} \right) \]
2. \[ A_{j+1} = Q_{12}^* \cdot A_j \]
3. \[ B_{j+1} = Q_{22}^* \cdot B_j \]

until \( R_j \) converges

Output is \( A_k, B_k \) such that

\[ A_k^{-1} B_k = \left( A^{-1} \right)^{2^k} \]
Implicit Repeated Squaring

\[ A_0 = A, \quad B_0 = I \]

Repeat

1. \[
\begin{bmatrix}
Q_{11} & Q_{12} \\
Q_{21} & Q_{22}
\end{bmatrix}
\cdot
\begin{bmatrix}
R_j \\
0
\end{bmatrix}
= \text{qr}\left(\begin{bmatrix}
B_j \\
-A_j
\end{bmatrix}\right)
\]

2. \[
A_{j+1} = Q_{12}^\ast \cdot A_j
\]

3. \[
B_{j+1} = Q_{22}^\ast \cdot B_j
\]

until \( R_j \) converges

Output is \( A_k, B_k \) such that

\[
A_k^{-1} B_k = \left(A^{-1}\right)^{2^k}
\]

Next step is to compute a rank-revealing decomposition of

\[
\left(I + (A^{-1})^{2^k}\right)^{-1} = \left(I + A_k^{-1} B_k\right)^{-1} = (A_k + B_k)^{-1} A_k
\]
Randomized Rank-Revealing QR (RURV)

Use a Haar-distributed random matrix:

1. generate random matrix $B$ with i.i.d. $N(0, 1)$ entries
2. $V \cdot R_1 = \text{qr}(B)$
3. $U \cdot R = \text{qr}(A \cdot V^*)$

so that

$$A = URV$$

where $U$ and $V$ are orthogonal and $R$ is upper triangular
If $\sigma_r \sim \sigma_1$ and $\sigma_{r+1} \sim \frac{1}{\text{poly}(n)} \sigma_r$, then with high probability

\[
\sigma_{\min}(R_{11}) \geq O\left(\frac{1}{\sqrt{rn}}\right) \sigma_r
\]
\[
\sigma_{\max}(R_{22}) \leq O\left((rn)^2\right) \sigma_{r+1}
\]

- first inequality matches best deterministic URV algorithms
- second inequality is much weaker, but proof is lax (actual bound may be linear)
- repeated squaring will drive $\sigma_r$ and $\sigma_{r+1}$ very far apart
Generalized RURV (GRURV)

We want to compute RURV of matrices of the form $C^{-1}D$:

$$(A_k + B_k)^{-1}A_k$$

We can do it implicitly:

1. $U_2 \cdot R_2 \cdot V = \text{rurv}(D)$
2. $R_1 \cdot U_1 = \text{rq}(U_2^* \cdot C)$

so that

$$C^{-1}D = (U_2R_1U_1)^{-1}(U_2R_2V) = U_1^*(R_1^{-1}R_2)V$$

- No inverses computed (we only need the orthogonal matrix $U_1$)
- Computing $U_1 \cdot A \cdot U_1^*$ completes one step of divide and conquer
Generalized RURV works for arbitrary products of matrices:

\[ A_1 \pm 1 \cdot A_2 \pm 1 \cdots A_k \pm 1 \]

- requires one RURV (or RULV) and \( k - 1 \) QR’s (or RQ’s)
- output is \( U(R_1 \pm 1 \cdot R_2 \pm 1 \cdots R_k \pm 1) V \)
- rank-revealing properties same as for RURV (on one matrix)

Deterministic rank-revealing QR (for one matrix) doesn’t suffice in generalized case
Choosing splitting lines

- Computing \( \left( I + (A^{-1})^{2^k} \right)^{-1} \) splits spectrum along unit circle

- Use Moebius transformation to split along any circle or line in complex plane
  - set \( A_0 = \alpha A + \beta I, B_0 = \gamma A + \delta I \)

- Continue splitting until subproblem fits
  - on one processor or
  - in fast memory

and use standard algorithms (no extra communication costs)
Randomized Bisection

Pick inner circle around center

Gershgorin bounding disc
Randomized Bisection

Choose random angle

θ
Randomized Bisection
Randomized Bisection

Choose random perpendicular in range
Randomized Bisection
“Success” means iterative process converges
- either we split the spectrum, or
- we narrow down the region containing all the eigenvalues

If the splitting line does not intersect the $(\epsilon \cdot \|A\|)$-pseudospectrum, then convergence occurs within a constant number of iterations
- number of iterations depends on smallest relative perturbation that moves an eigenvalue onto splitting line (it does not depend on $n$)

For the case of normal matrices, the probability of not intersecting the pseudospectrum with randomized bisection is

$$1 - O(n \cdot \epsilon)$$
Communication Upper Bound (sequential case)

- $M =$ memory size, $\gamma =$ cost of flop, $\beta =$ inverse bandwidth, $\alpha =$ latency

Assuming constant number of iterations, cost of one step of divide-and-conquer is

$$C_{D+C}(n) = \alpha \cdot O \left( \frac{n^3}{M^{3/2}} \right) + \beta \cdot O \left( \frac{n^3}{\sqrt{M}} \right) + \gamma \cdot O(n^3)$$

Assuming we split the spectrum by some fraction each time, the total cost of the entire algorithm is asymptotically the same

- same communication complexity as matrix multiplication and QR
- attains lower bound
Communication Upper Bound (parallel case)

- $P =$ number of processors, $\gamma =$ cost of flop, $\beta =$ inverse bandwidth, $\alpha =$ latency

Assuming constant number of iterations, cost of one step of divide-and-conquer is

$$C_{D+C}(n, P) = \alpha \cdot O\left(\sqrt{P \log P}\right) + \beta \cdot O\left(\frac{n^2}{\sqrt{P}} \log P\right) + \gamma \cdot O\left(\frac{n^3}{P} \log P\right)$$

By assigning disjoint subsets of processors to two subproblems after each split, subproblems can be solved in parallel yielding the same asymptotic cost for the entire algorithm

- same communication complexity as QR
- attains lower bound (to within logarithmic factors)
Numerical Experiments

One step of divide-and-conquer to split spectrum of $A$ about unit circle:

$$A_0 = A, \quad B_0 = I$$

Repeat

1. $$\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \cdot \begin{bmatrix} R_j \\ 0 \end{bmatrix} = \text{qr} \left( \begin{bmatrix} B_j \\ -A_j \end{bmatrix} \right)$$

2. $$A_{j+1} = Q_{12}^* \cdot A_j$$

3. $$B_{j+1} = Q_{22}^* \cdot B_j$$

4. $$U = \text{GRURV}(A_j + B_j, A_j)$$

5. $$A_{\text{new}} = U \cdot A \cdot U^* = \begin{bmatrix} A_{11} & A_{12} \\ E_{21} & A_{22} \end{bmatrix}$$

until $$\frac{\|E_{21}\|}{\|A\|}$$ is small

$$\frac{\|E_{21}\|}{\|A\|}$$ measures backward error, shown in convergence plots
Random matrix $A = QDQ^*$
Try a tougher matrix

- Half the eigenvalues lie at distance $10^{-5}$ outside unit circle
- Other half of eigenvalues lie at distance $10^{-5}$ inside unit circle

Unit circle worst choice for splitting curve!
Try a different splitting curve

- Half the eigenvalues lie at distance $10^{-5}$ outside unit circle
- Other half of eigenvalues lie at distance $10^{-5}$ inside unit circle
1 Communication-Avoiding Linear Algebra

2 Randomized Spectral Divide and Conquer
   - Divide and Conquer Algorithm
   - Randomized Bisection
   - Communication Costs
   - Numerical Experiments

3 Other Optimal (Deterministic) Algorithms

4 Conclusions
Another approach to Symmetric/SVD case using reduction to tri-/bidiagonal form
- minimizes words moved in both parallel and sequential cases
- minimizes messages moved in parallel case (up to log factor)

SBR: Two-step reduction
1. reduce the full matrix to banded form
   - uses QR factorization and BLAS 3 - communication-optimal
2. reduce the banded matrix to tri-/bidiagonal form
   - not optimal alone, but communication dominated by first step
Initial bandwidth $b = \Theta \left( \sqrt{M} \right)$

Choose $d = c = b/2$ (at each pass) to minimize communication
Eigenvector matrix $X$ of triangular matrix $T$ can be computed by

$$X_{ij} = \frac{T_{ij}X_{jj} + \sum_{k=i+1}^{j-1} T_{ik}X_{kj}}{T_{jj} - T_{ii}}$$

for $i < j$ where $X_{jj}$’s can be arbitrary nonzeros.

- Lower bounds given by results in [BDHS10]
- Current (Sca)LAPACK algorithms solve for one eigenvector at a time (pessimal communication costs)
- New blocked algorithms (for sequential and parallel) attain lower bounds
Conclusions

- New divide-and-conquer approach communication-optimal
  - Symm/SVD, Nonsymm, and generalized problems, words and messages, sequential and parallel
  - possible large constant factor more flops than standard algorithms
  - requires randomization

- Successive Band Reduction approach communication-optimal
  - Symm/SVD cases
  - similar flop count to standard algs for values only, more for vectors

- Standard libraries not optimal - time for new algorithms
  - we have asymptotically faster (optimal) algorithms for nearly all direct linear algebra

- Current work
  - Prove lower bounds for computing Schur form
  - Develop optimal SBR parallel algorithm
  - Develop deterministic communication-optimal rank-revealing QR
Thank you!
Sequential Algorithm for **TREVC**

**Algorithm 1** Blocked Iterative Algorithm

```
for j = 1 to n/b do
    for $i = j - 1$ down to 1 do
        $S = 0$
        for $k = i + 1$ to j do
            $S = S + T[i, k] \cdot X[k, j]$
        end for
        solve $T[i, i] \cdot X[i, j] + S = X[i, j] \cdot D[j, j]$ for $X[i, j]$
    end for
end for
```

- notation: $T[i, j]$ is a $b \times b$ block
- use blocksize $b = \Theta(\sqrt{M})$ and block-contiguous DS for optimality
- this algorithm ignores need for scaling to prevent under/overflow
- a recursive, cache-oblivious algorithm also achieves optimality
- LAPACK’s **TREVC** solves for one eigenvector at a time
Parallel Algorithm for **PTREVC**

- Using 2D blocked layout for $T$ on square grid of processors, compute $X$ with same layout
- Iterate over block diagonals, updating trailing matrix each step
  - Local computation occurs in gray: (a) and (d)
  - Communication occurs along arrows: (b) is a broadcast of $X$ block, (c) is a nearest-neighbor pass of $T$ block

Communication costs within log $P$ of optimality

**ScaLAPACK's PTREVC** solves for one eigenvector at a time
Subproblem assignment

- Assign number of processors proportional to size of subproblem

- assuming 2D blocked layout, at most one processor owns pieces of both subproblems
- use one of the idle processors to help out
- cost of larger subproblem dominates cost of smaller subproblem
Alternate Convergence Criterion

One step of divide-and-conquer to split spectrum of $A$ about unit circle:

$A_0 = A$, $B_0 = I$

Repeat

1. \[
\begin{bmatrix}
  Q_{11} & Q_{12} \\
  Q_{21} & Q_{22}
\end{bmatrix} \cdot \begin{bmatrix}
  R_j \\
  0
\end{bmatrix} = qr \left( \begin{bmatrix}
  B_j \\
  -A_j
\end{bmatrix} \right)
\]

2. $A_{j+1} = Q_{12}^* \cdot A_j$

3. $B_{j+1} = Q_{22}^* \cdot B_j$

until $\frac{||R_j - R_{j-1}||}{||R_{j-1}||}$ is small

4. $U = GRURV(A_j + B_j, A_j)$

5. $A_{\text{new}} = U \cdot A \cdot U^*$

cheap convergence test, we’ll refer to this as “$R\ conv$”
Normal Matrix

- Half the eigenvalues lie at distance $10^{-5}$ outside unit circle
- Other half of eigenvalues $< .5$ in absolute value
Non-normal Matrix with Jordan block

- Half the eigenvalues form Jordan block centered at 1.3
- Other half of eigenvalues < .5 in absolute value
Try restarting

- Half the eigenvalues form Jordan block centered at 1.3
- Other half of eigenvalues < .5 in absolute value

- Restart iteration with nearly block triangular matrix
Create a set of bulges and chase into second half of columns
Parallel SBR

- Chase a set of bulges from first half to second half of columns
Parallel SBR

- Clear a set of bulges off the end
Proof of Communication Lower Bound on $C = A \ast B$ (5/6)

# cubes in black box with side lengths $x$, $y$ and $z$
$= \text{Volume of black box}$
$= x \cdot y \cdot z$
$= (\#A\square \ast \#B\square \ast \#C\square )^{1/2}$
$= (xz \cdot zy \cdot yx)^{1/2}$

(i,k) is in “A shadow” if (i,j,k) in 3D set
(j,k) is in “B shadow” if (i,j,k) in 3D set
(i,j) is in “C shadow” if (i,j,k) in 3D set

Thm (Loomis & Whitney, 1949)
$\# \text{cubes in 3D set} = \text{Volume of 3D set}$
$\leq (\text{area}(A\text{ shadow}) \ast \text{area}(B\text{ shadow}) \ast \text{area}(C\text{ shadow})))^{1/2}$
Z. Bai, J. Demmel, and M. Gu.
An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems.

Minimizing communication in linear algebra.

Circular dichotomy of a matrix spectrum.

_ScaLAPACK Users’ Guide_.
Also available from http://www.netlib.org/scalapack/.

C. Bischof, B. Lang, and X. Sun.
A framework for symmetric band reduction.

L. Cannon.
_A cellular computer to implement the Kalman filter algorithm._
J. Demmel, I. Dumitriu, and O. Holtz.
Fast linear algebra is stable.

J. Demmel, L. Grigori, M. Hoemmen, and J. Langou.
Communication-optimal parallel and sequential QR and LU factorizations.

L. Grigori, J. Demmel, and H. Xiang.
Communication-avoiding Gaussian elimination.
Supercomputing 08, 2008.

J. W. Hong and H. T. Kung.
I/O complexity: The red-blue pebble game.

D. Irony, S. Toledo, and A. Tiskin.
Communication lower bounds for distributed-memory matrix multiplication.

S. Lennart Johnsson.
Minimizing the communication time for matrix multiplication on multiprocessors.

A.N. Malyshev.
Computing invariant subspaces of a regular linear pencil of matrices.