Avoiding Communication in Parallel Bidiagonalization of Band Matrices

Grey Ballard, James Demmel, Nicholas Knight

UC Berkeley

SIAM CSE 13

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from Par Lab affiliates National Instruments, NEC, Nokia, NVIDIA, and Samsung.
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

- in many cases we need new algorithms
- in many cases we can prove lower bounds and optimality
$\gamma = \text{time per flop}$

$\beta = \text{time per word moved}$

$\alpha = \text{time per message}$

$F = \text{#flops}$

$BW = \text{#words moved}$

$L = \text{#messages}$

Running time $= \gamma \cdot F + \beta \cdot BW + \alpha \cdot L$
Direct vs Two-Step Bidiagonalization

Application: computing the dense SVD via reduction to bidiagonal form (bidiagonalization)

- Conventional approach (e.g. LAPACK) is direct bidiagonalization
- Two-step approach reduces first to band, then band to bidiagonal

Direct:

Two-step:
Direct vs Two-Step Bidiagonalization

Application: computing the dense SVD via reduction to bidiagonal form (bidiagonalization)

- Conventional approach (e.g. LAPACK) is direct bidiagonalization
- Two-step approach reduces first to band, then band to bidiagonal

**Direct:**

- A

**Two-step:**

- A

---

![Graph showing MFLOPS vs n for MatMul, Direct, and Two-step methods](image)

- \[ \text{MatMul} \]
- \[ \text{Direct} \]
- \[ \text{Two-step} \]
Why is direct bidiagonalization slow?

Communication costs!

<table>
<thead>
<tr>
<th>Approach</th>
<th>Flops</th>
<th>Words Moved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>$\frac{8}{3} n^3$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>Two-step</td>
<td>(1) $\frac{8}{3} n^3$</td>
<td>$O\left(\frac{n^3}{\sqrt{M}}\right)$</td>
</tr>
<tr>
<td></td>
<td>(2) $O\left(n^2 \sqrt{M}\right)$</td>
<td>$O\left(n^2 \sqrt{M}\right)$</td>
</tr>
</tbody>
</table>

$M = \text{fast memory size}$

- Direct approach achieves $O(1)$ data re-use
- Two-step approach moves fewer words than direct approach
  - using intermediate bandwidth $b = \Theta(\sqrt{M})$
- Full-to-banded step (1) achieves $O(\sqrt{M})$ data re-use
  - this is optimal
- Band reduction step (2) achieves $O(1)$ data re-use
  - Can we do better?
We want to compute and apply orthogonal matrices $Q$ and $W$ to transform a band matrix $B$ to a bidiagonal matrix $C$:

$$Q^T BW = C$$

The basic procedure for band reduction is known as “bulge chasing”
- main idea is to annihilate entries with orthogonal transformations but maintain band sparsity structure
- there’s a big design space, many different approaches
- same ideas work for symmetric band eigenproblem
Successive Band Reduction (bulge-chasing)

\[ c + d \leq b \]

\[ b = \text{bandwidth} \]
\[ c = \text{columns} \]
\[ d = \text{diagonals} \]

[In Bischof, Lang, Sun 2000]
SBR - 1 Sweep Approach

eliminate one column at a time
bidiagonal after one sweep

\[ b = \text{bandwidth} \]
\[ c = 1 \]
\[ d = b - 1 \]
Several Different Scenarios. . .

- starting with dense matrix OR starting with band matrix
- seeking singular values only OR seeking also singular vectors
  - left AND/OR right singular vectors (some OR all of them)
- sequential machine OR parallel machine
- singular value decomposition OR symmetric eigenproblem
Several Different Scenarios...

- starting with dense matrix OR starting with band matrix
- seeking singular values only OR seeking also singular vectors
  - left AND/OR right singular vectors (some OR all of them)
- sequential machine OR parallel machine
- singular value decomposition OR symmetric eigenproblem

We’ll focus on bidiagonalization of (lower triangular) band matrices for the rest of the talk, considering

- sequential and parallel cases
- values only and values and (left and right) vectors cases

Our main goal will be to find ways to re-use data in band reduction process
Accumulating Orthogonal Transformations

Band reduction:

\[ B = QCW^T \]

Bidiagonal SVD:

\[ C = U\Sigma V^T \]

Full SVD:

\[ B = (QU)\Sigma(WV)^T \]

To compute left singular vectors of band matrix \( B \), either

1. form \( Q \) explicitly and apply \( U \) to \( Q \) from right, or
2. store \( Q \) implicitly and apply \( Q \) to \( U \) from left
How do we get data re-use?

1. Increase number of columns in parallelogram \((c)\)
   - permits blocking Householder updates: \(O(c)\) re-use
   - constraint \(c + d \leq b \implies\) trade-off between re-use and progress
   - requires multiple “sweeps”

2. Chase multiple bulges at a time \((\omega)\)
   - apply several updates to band while it’s in local memory: \(O(\omega)\) re-use
   - bulges cannot overlap, need working set to fit in local memory
How do we get data re-use?

1. Increase number of columns in parallelogram \((c)\)
   - permits blocking Householder updates: \(O(c)\) re-use
   - constraint \(c + d \leq b \implies\) trade-off between re-use and progress
   - requires multiple “sweeps”

2. Chase multiple bulges at a time \((\omega)\)
   - apply several updates to band while it’s in local memory: \(O(\omega)\) re-use
   - bulges cannot overlap, need working set to fit in local memory
How do we get data re-use?

1. Increase number of columns in parallelogram (c)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple “sweeps”

2. Chase multiple bulges at a time ($\omega$)
   - apply several updates to band while it’s in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory
How do we get data re-use?

1. Increase number of columns in parallelogram \((c)\)
   - permits blocking Householder updates: \(O(c)\) re-use
   - constraint \(c + d \leq b \implies\) trade-off between re-use and progress
   - requires multiple “sweeps”

2. Chase multiple bulges at a time \((\omega)\)
   - apply several updates to band while it’s in local memory: \(O(\omega)\) re-use
   - bulges cannot overlap, need working set to fit in local memory
How do we get data re-use?

1. Increase number of columns in parallelogram ($c$)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple “sweeps”

2. Chase multiple bulges at a time ($\omega$)
   - apply several updates to band while it’s in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory
How do we get data re-use?

1. Increase number of columns in parallelogram \((c)\)
   - permits blocking Householder updates: \(O(c)\) re-use
   - constraint \(c + d \leq b \implies\) trade-off between re-use and progress
   - requires multiple "sweeps"

2. Chase multiple bulges at a time \((\omega)\)
   - apply several updates to band while it’s in local memory: \(O(\omega)\) re-use
   - bulges cannot overlap, need working set to fit in local memory
How do we get data re-use?

1. Increase number of columns in parallelogram \((c)\)
   - permits blocking Householder updates: \(O(c)\) re-use
   - constraint \(c + d \leq b \implies\) trade-off between re-use and progress
   - requires multiple “sweeps”

2. Chase multiple bulges at a time \((\omega)\)
   - apply several updates to band while it’s in local memory: \(O(\omega)\) re-use
   - bulges cannot overlap, need working set to fit in local memory
How do we get data re-use?

1. Increase number of columns in parallelogram \((c)\)
   - permits blocking Householder updates: \(O(c)\) re-use
   - constraint \(c + d \leq b \implies\) trade-off between re-use and progress
   - requires multiple “sweeps”

2. Chase multiple bulges at a time \((\omega)\)
   - apply several updates to band while it’s in local memory: \(O(\omega)\) re-use
   - bulges cannot overlap, need working set to fit in local memory
How do we get data re-use?

1. Increase number of columns in parallelogram ($c$)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple “sweeps”

2. Chase multiple bulges at a time ($\omega$)
   - apply several updates to band while it’s in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory
Data access patterns

One bulge at a time

Four bulges at a time
Asymptotics - singular values only - sequential case

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK</td>
<td>$4n^2b$</td>
<td>$O(n^2b)$</td>
<td>$O(n^2b)$</td>
</tr>
<tr>
<td>1 Sweep SBR</td>
<td>$8n^2b$</td>
<td>$O(n^2b)$</td>
<td>$O\left(\frac{n^2b}{M}\right)$</td>
</tr>
</tbody>
</table>

CA-SBR cuts remaining bandwidth in half at each sweep.
Asymptotics - singular values only - sequential case

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK</td>
<td>$4n^2b$</td>
<td>$O(n^2b)$</td>
<td>$O(n^2b)$</td>
</tr>
<tr>
<td>1 Sweep SBR</td>
<td>$8n^2b$</td>
<td>$O(n^2b)$</td>
<td>$O\left(\frac{n^2b}{M}\right)$</td>
</tr>
<tr>
<td>Improved 1 Sweep SBR\n</td>
<td>$8n^2b$</td>
<td>$O\left(\frac{n^2b^3}{M}\right)$</td>
<td>$O\left(\frac{n^2b^3}{M^2}\right)$</td>
</tr>
</tbody>
</table>

\[\dagger\text{assuming } 1 \leq b \leq \sqrt{M}/3\]
Asymptotics - singular values only - sequential case

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK</td>
<td>$4n^2b$</td>
<td>$O(n^2b)$</td>
<td>$O(n^2b)$</td>
</tr>
<tr>
<td>1 Sweep SBR</td>
<td>$8n^2b$</td>
<td>$O(n^2b)$</td>
<td>$O\left(\frac{n^2b}{M}\right)$</td>
</tr>
<tr>
<td>Improved 1 Sweep SBR†</td>
<td>$8n^2b$</td>
<td>$O\left(\frac{n^2b^3}{M}\right)$</td>
<td>$O\left(\frac{n^2b^3}{M^2}\right)$</td>
</tr>
<tr>
<td>CA-SBR†</td>
<td>$6n^2b$</td>
<td>$O\left(\frac{n^2b^2}{M}\right)$</td>
<td>$O\left(\frac{n^2b^2}{M^2}\right)$</td>
</tr>
</tbody>
</table>

†assuming $1 \leq b \leq \sqrt{M}/3$

CA-SBR cuts remaining bandwidth in half at each sweep

- starts with big $c$ and decreases by half at each sweep
- starts with small $\omega$ and doubles at each sweep
What if you want singular vectors too? - sequential case

We’ve used two optimizations:

1. Chase multiple bulges (increase $\omega$)
2. Take multiple sweeps (increase $c$)
   - Accumulating orthogonal transformations costs $O(n^3)$ flops per sweep

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK</td>
<td>$4n^3$</td>
<td>$O(n^2b + n^3)$</td>
<td>$O(n^2b + n^3M)$</td>
</tr>
<tr>
<td>1 Sweep SBR</td>
<td>$4n^3$</td>
<td>$O(n^2b + n^3\sqrt{M})$</td>
<td>$O(n^2bM + n^3M)$</td>
</tr>
<tr>
<td>Improved 1 Sweep SBR†</td>
<td>$4n^3$</td>
<td>$O(n^2b^3M + n^3\sqrt{M})$</td>
<td>$O(n^2b^3M^2 + n^3M^3/2)$</td>
</tr>
<tr>
<td>CA-SBR†</td>
<td>$2n^3\log b$</td>
<td>$O(n^2b\sqrt{M} + n^3\log b\sqrt{M})$</td>
<td>$O(n^2b\log bM + n^3\log bM^3/2)$</td>
</tr>
</tbody>
</table>
What if you want singular vectors too? - sequential case

We’ve used two optimizations:

1. chase multiple bulges (increase $\omega$)
2. take multiple sweeps (increase $c$)

- accumulating orthogonal transformations costs $O(n^3)$ flops per sweep

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK</td>
<td>$4n^3$</td>
<td>$O(n^2 b + n^3)$</td>
<td>$O \left( n^2 b + \frac{n^3}{M} \right)$</td>
</tr>
<tr>
<td>1 Sweep SBR</td>
<td>$4n^3$</td>
<td>$O \left( n^2 b + \frac{n^3}{\sqrt{M}} \right)$</td>
<td>$O \left( \frac{n^2 b}{M} + \frac{n^3}{M} \right)$</td>
</tr>
</tbody>
</table>

communication costs: band reduction + orthogonal updates
What if you want singular vectors too? - sequential case

We’ve used two optimizations:
1. chase multiple bulges (increase $\omega$)
2. take multiple sweeps (increase $c$)
   - accumulating orthogonal transformations costs $O(n^3)$ flops per sweep

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK</td>
<td>$4n^3$</td>
<td>$O(n^2 b + n^3)$</td>
<td>$O\left( n^2 b + \frac{n^3}{M} \right)$</td>
</tr>
<tr>
<td>1 Sweep SBR</td>
<td>$4n^3$</td>
<td>$O\left( \frac{n^2 b}{M} + \frac{n^3}{\sqrt{M}} \right)$</td>
<td>$O\left( \frac{n^2 b}{M^2} + \frac{n^3}{M^{3/2}} \right)$</td>
</tr>
<tr>
<td>Improved 1 Sweep SBR†</td>
<td>$4n^3$</td>
<td>$O\left( \frac{n^2 b^3}{M} + \frac{n^3}{\sqrt{M}} \right)$</td>
<td>$O\left( \frac{n^2 b^3}{M^2} + \frac{n^3}{M^{3/2}} \right)$</td>
</tr>
</tbody>
</table>

communication costs: band reduction + orthogonal updates
†assuming $1 \leq b \leq \sqrt{M}/3$
What if you want singular vectors too? - sequential case

We’ve used two optimizations:

1. chase multiple bulges (increase $\omega$)
2. take multiple sweeps (increase $c$)
   - accumulating orthogonal transformations costs $O(n^3)$ flops per sweep

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK</td>
<td>$4n^3$</td>
<td>$O(n^2 b + n^3)$</td>
<td>$O\left(\frac{n^2 b}{M} + \frac{n^3}{M^{3/2}}\right)$</td>
</tr>
<tr>
<td>1 Sweep SBR</td>
<td>$4n^3$</td>
<td>$O\left(\frac{n^2 b^3}{M} + \frac{n^3}{\sqrt{M}}\right)$</td>
<td>$O\left(\frac{n^2 b^3}{M^2} + \frac{n^3}{M^{3/2}}\right)$</td>
</tr>
<tr>
<td>Improved 1 Sweep SBR$^\dagger$</td>
<td>$4n^3$</td>
<td>$O\left(\frac{n^2 b^3}{M} + \frac{n^3}{\sqrt{M}}\right)$</td>
<td>$O\left(\frac{n^2 b^3}{M^2} + \frac{n^3}{M^{3/2}}\right)$</td>
</tr>
<tr>
<td>CA-SBR$^\dagger$</td>
<td>$2n^3 \log b$</td>
<td>$O\left(\frac{n^2 b}{\sqrt{M}} + \frac{n^3 \log b}{\sqrt{M}}\right)$</td>
<td>$O\left(\frac{n^2 \log b}{M} + \frac{n^3 \log b}{M^{3/2}}\right)$</td>
</tr>
</tbody>
</table>

communication costs: band reduction + orthogonal updates

$^\dagger$assuming $1 \leq b \leq \sqrt{M}/3$
Parallel 1 Sweep SBR

eliminate one column at a time; bidiagonal after one sweep

works like a bandsaw: columns move left; Householder vectors move right; $O(1)$ messages per column

[Lang 1993]
Parallel CA-SBR

works like a sandbag relay:
each processor passes bulges along
$O(p)$ messages per sweep

cut bandwidth in half each sweep;
requires multiple sweeps
Multiple sweeps and chasing multiple bulges reduces latency cost

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Sweep SBR</td>
<td>$O\left(\frac{n^2 b}{p}\right)$</td>
<td>$O(nb)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>CA-SBR\textsuperscript{†}</td>
<td>$O\left(\frac{n^2 b}{p}\right)$</td>
<td>$O(nb)$</td>
<td>$O(p \log b)$</td>
</tr>
</tbody>
</table>

\textsuperscript{†}assuming $1 \leq b \leq n/(3p)$
What if you want singular vectors too? - parallel case

Run band reduction on $\sqrt{p}$ processors, orthogonal updates on all $p$
- broadcasting band reduction updates, or
- redundantly computing band reduction

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Flops</th>
<th>Words</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Sweep SBR*</td>
<td>$\frac{4n^3}{p}$</td>
<td>$O\left(nb + \frac{n^2}{\sqrt{p}}\right)$</td>
<td>$O(n + \log p)$</td>
</tr>
<tr>
<td>CA-SBR†</td>
<td>$\frac{2n^3 \log b}{p}$</td>
<td>$O\left(nb + \frac{n^2}{\sqrt{p}} \log b\right)$</td>
<td>$O(\sqrt{p} \log b)$</td>
</tr>
</tbody>
</table>

* [Auckenthaler et al. 2011]
† assuming $1 \leq b \leq n/(3\sqrt{p})$

Again, latency is reduced at the cost of extra computation
Conclusions and Future Work

We’ve used two means to improve data re-use in band reduction schemes:

1. taking multiple sweeps (re-using data within a bulge chase)
2. chasing multiple bulges (re-using data among bulge chases)

Asymptotic communication improvements:

1. in sequential case, we can reduce both bandwidth and latency costs
2. in parallel case, we can reduce latency cost

For singular vectors, multiple sweeps results in extra computation

- for subset of vectors, extra computation decreases
- to navigate tradeoff, take $1 \leq \# \text{sweeps} \leq \log b$

These ideas can also benefit full SVD case (starting with dense matrix) and symmetric eigenproblem (with different constant factors)
Thank you!

Grey Ballard, Jim Demmel, Nick Knight
{ballard,demmel,knight}@cs.berkeley.edu
References

Aggarwal, A., and Vitter, J. S.
The input/output complexity of sorting and related problems.

http://icl.cs.utk.edu/plasma/.

Ballard, G., Demmel, J., Holtz, O., and Schwartz, O.
Minimizing communication in linear algebra.

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

Bischof, C. H., Lang, B., and Sun, X.
Algorithm 807: The SBR Toolbox—software for successive band reduction.

Demmel, J., Grigori, L., Hoemmen, M., and Langou, J.
Communication-optimal parallel and sequential QR and LU factorizations.
Dongarra, J., Hammarling, S., and Sorensen, D.
Block reduction of matrices to condensed forms for eigenvalue computations.

Fuller, S. H., and Millett, L. I., Eds.
The Future of Computing Performance: Game Over or Next Level?

Haidar, A., Ltaief, H., and Dongarra, J.
Parallel reduction to condensed forms for symmetric eigenvalue problems using aggregated fine-grained and memory-aware kernels.

Howell, G., Demmel, J., Fulton, C., Hammarling, S., and Marmol, K.
Cache efficient bidiagonalization using BLAS 2.5 operators.

Kaufman, L.
Banded eigenvalue solvers on vector machines.

Kaufman, L.
Band reduction algorithms revisited.
Lang, B.
A parallel algorithm for reducing symmetric banded matrices to tridiagonal form.

Lang, B.
Efficient eigenvalue and singular value computations on shared memory machines.

Ltaief, H., Luszczek, P., and Dongarra, J.
High performance bidiagonal reduction using tile algorithms on homogeneous multicore architectures.
Submitted to ACM TOMS.

Luszczek, P., Ltaief, H., and Dongarra, J.
Two-stage tridiagonal reduction for dense symmetric matrices using tile algorithms on multicore architectures.

Murata, K., and Horikoshi, K.
A new method for the tridiagonalization of the symmetric band matrix.
References IV

Rajamanickam, S.
Efficient Algorithms for Sparse Singular Value Decomposition.

Rutishauser, H.
On Jacobi rotation patterns.

Schwarz, H.
Algorithm 183: Reduction of a symmetric bandmatrix to triple diagonal form.
Comm. ACM 6, 6 (June 1963), 315–316.

Schwarz, H.
Tridiagonalization of a symmetric band matrix.
Anatomy of a symmetric bulge-chase

**QR**: create zeros

**PRE**: $A \leftarrow Q^T A$

**SYM**: $A \leftarrow Q^T AQ$

**POST**: $A \leftarrow AQ$
Shared-Memory Parallel Implementation

lots of dependencies:
use pipelining

threads maintain working sets which never overlap