Avoiding Communication in Parallel Bidiagonalization of Band Matrices

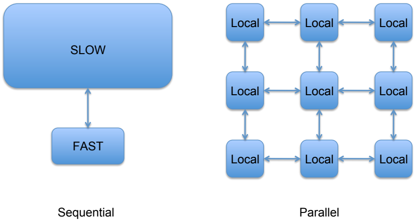
Grey Ballard, James Demmel, Nicholas Knight

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.

Motivation



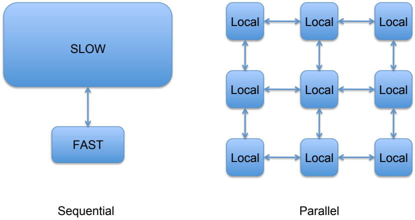
By *communication* we mean

- 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

- in many cases we need new algorithms
- in many cases we can prove lower bounds and optimality

Motivation



γ = time per flop

β = time per word moved

α = time per message

F = #flops

BW = #words moved

L = #messages

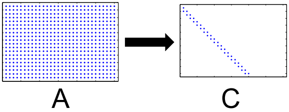
$$\text{Running time } = \gamma \cdot F + \beta \cdot BW + \alpha \cdot L$$

Direct vs Two-Step Bidiagonalization

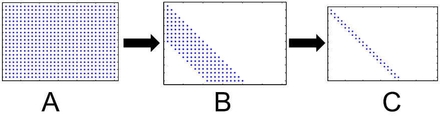
Application: computing the dense SVD via reduction to bidiagonal form (bidiagonalization)

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

Direct:



Two-step:

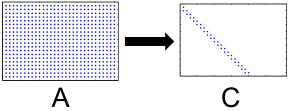


Direct vs Two-Step Bidiagonalization

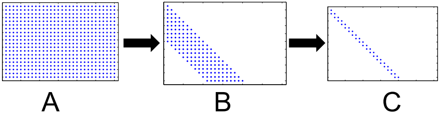
Application: computing the dense SVD via reduction to bidiagonal form (bidiagonalization)

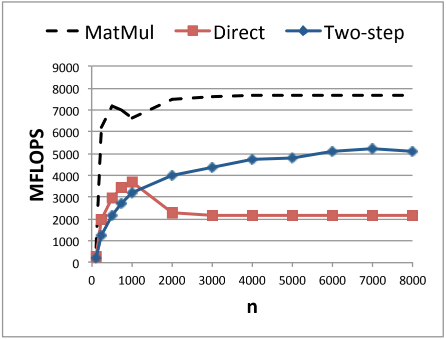
- Conventional approach (e.g. LAPACK) is direct bidiagonalization
- Two-step approach reduces first to band, then band to bidiagonal

Direct:



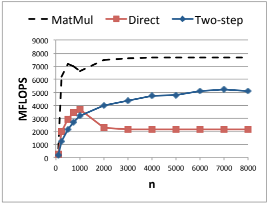
Two-step:





Why is direct bidiagonalization slow?

Communication costs!



Approach		Flops	Words Moved
Direct		$\frac{8}{3}n^3$	$O\left(n^3\right)$
Two-step	(1)	$\frac{8}{3}n^3$	$O\left(\frac{n^3}{\sqrt{M}}\right)$
	(2)	$O\left(n^2\sqrt{M}\right)$	$O\left(n^2\sqrt{M}\right)$

M = 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?**

Band Reduction via Bulge Chasing

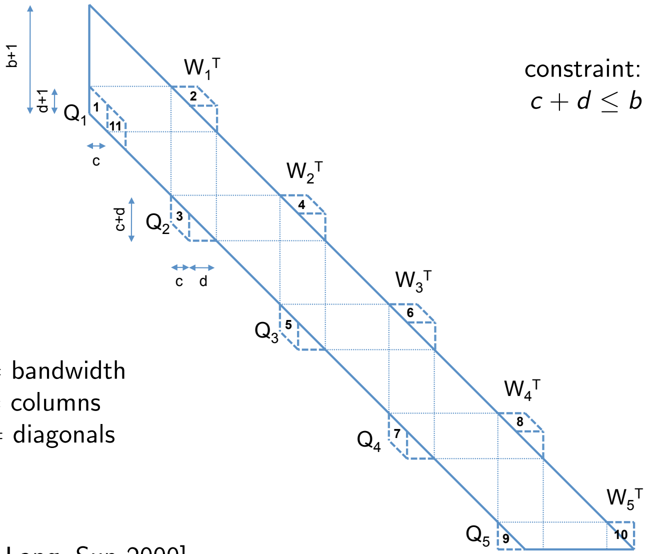
We want to compute and apply orthogonal matrices Q and W to transform a band matrix B to a bidiagonal matrix C:

$$Q^T BW = C$$

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)

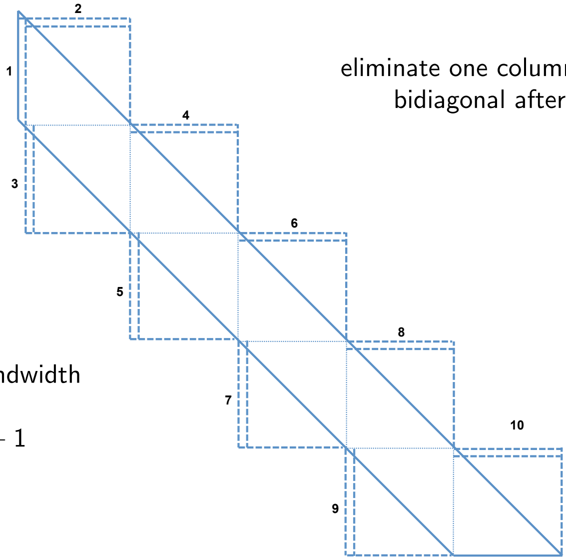


constraint:
$c + d \leq b$

b = bandwidth
c = columns
d = diagonals

[Bischof, Lang, Sun 2000]

SBR - 1 Sweep Approach



eliminate one column at a time
bidiagonal after one sweep

$b =$ 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:

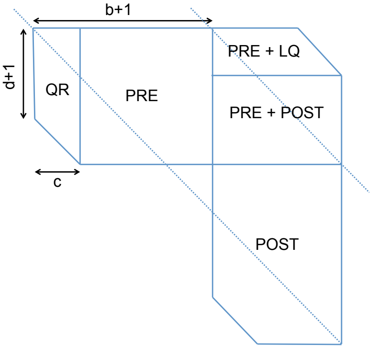
$$B = (QU)\Sigma(WV)^T$$

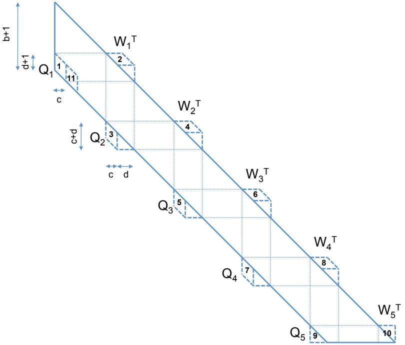
To compute left singular vectors of band matrix B, either

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?

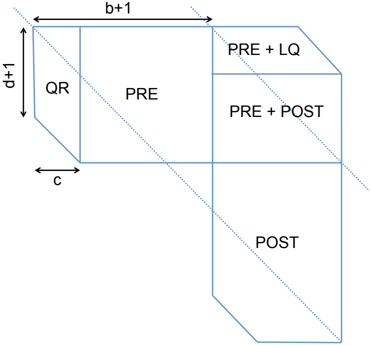
1. Increase number of columns in parallelogram (c)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple "sweeps"
2. Chase multiple bulges at a time (ω)
   - apply several updates to band while it's in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory

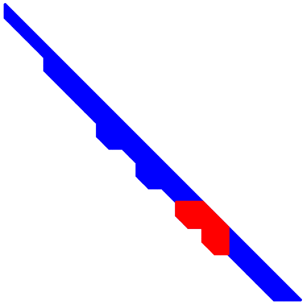




How do we get data re-use?

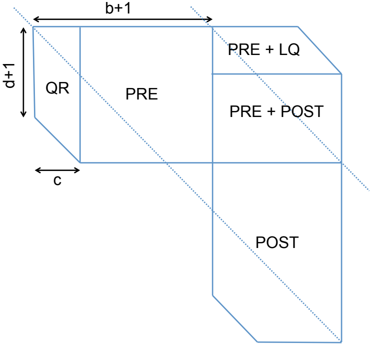
1. Increase number of columns in parallelogram (c)
    - permits blocking Householder updates: $O(c)$ re-use
    - constraint $c + d \leq b \implies$ trade-off between re-use and progress
    - requires multiple "sweeps"
2. Chase multiple bulges at a time (ω)
    - apply several updates to band while it's in local memory: $O(\omega)$ re-use
    - bulges cannot overlap, need working set to fit in local memory

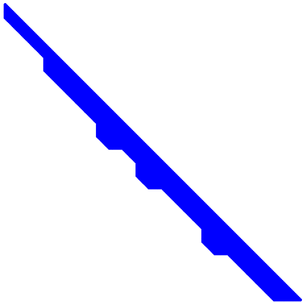




How do we get data re-use?

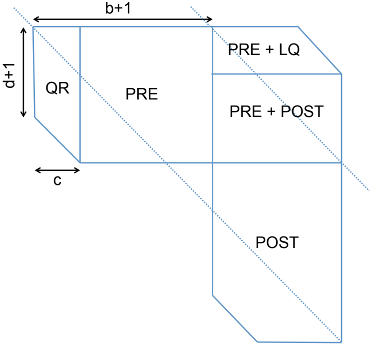
1. Increase number of columns in parallelogram (c)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple "sweeps"
2. Chase multiple bulges at a time (ω)
   - apply several updates to band while it's in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory

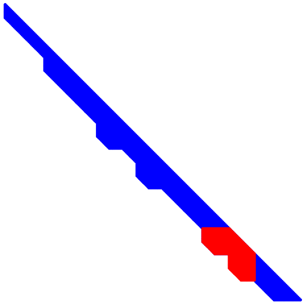




How do we get data re-use?

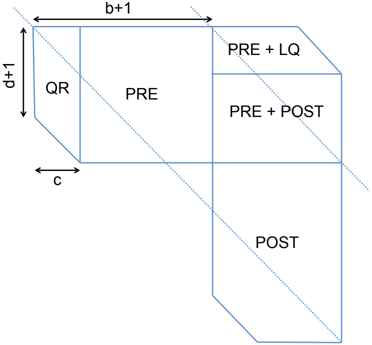
1. Increase number of columns in parallelogram (c)
    - permits blocking Householder updates: $O(c)$ re-use
    - constraint $c + d \leq b \implies$ trade-off between re-use and progress
    - requires multiple "sweeps"
2. Chase multiple bulges at a time (ω)
    - apply several updates to band while it's in local memory: $O(\omega)$ re-use
    - bulges cannot overlap, need working set to fit in local memory

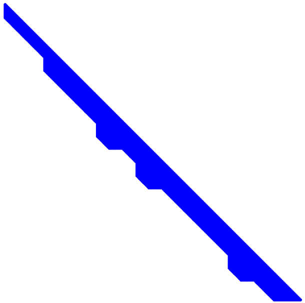




How do we get data re-use?

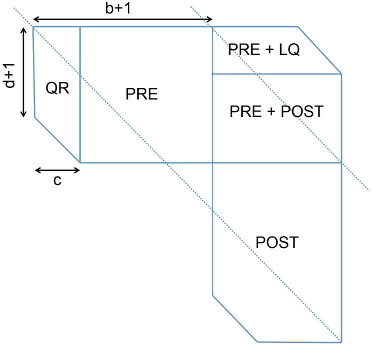
1. Increase number of columns in parallelogram (c)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple "sweeps"
2. Chase multiple bulges at a time (ω)
   - apply several updates to band while it's in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory

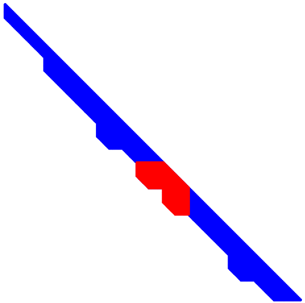




How do we get data re-use?

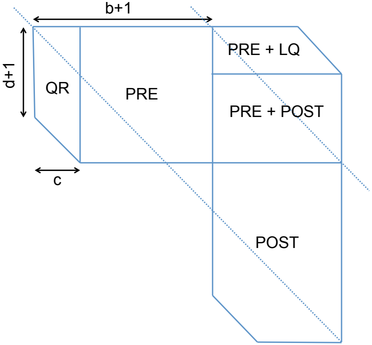
1. Increase number of columns in parallelogram (c)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple "sweeps"
2. Chase multiple bulges at a time (ω)
   - apply several updates to band while it's in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory

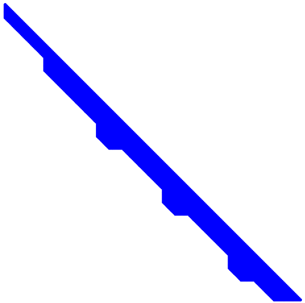




How do we get data re-use?

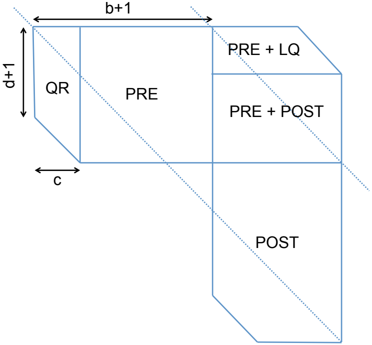
1. Increase number of columns in parallelogram (c)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple "sweeps"
2. Chase multiple bulges at a time (ω)
   - apply several updates to band while it's in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory

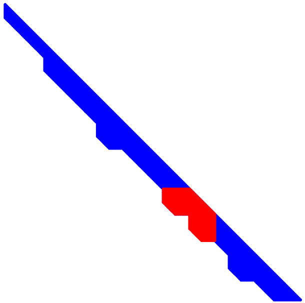




How do we get data re-use?

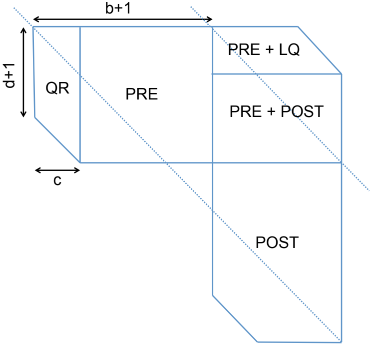
1. Increase number of columns in parallelogram (c)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple "sweeps"
2. Chase multiple bulges at a time (ω)
   - apply several updates to band while it's in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory

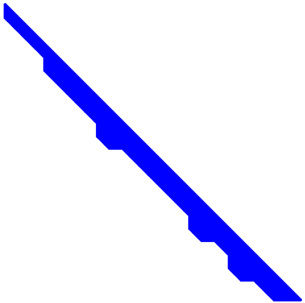




How do we get data re-use?

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 (ω)
   - 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

Algorithm	Flops	Words	Messages
LAPACK	$4n^2b$	$O(n^2b)$	$O\left(n^2b\right)$
1 Sweep SBR	$8n^2b$	$O(n^2b)$	$O\left(\frac{n^2b}{M}\right)$

Asymptotics - singular values only - sequential case

Algorithm	Flops	Words	Messages
LAPACK	$4n^2b$	$O(n^2b)$	$O\left(n^2b\right)$
1 Sweep SBR	$8n^2b$	$O(n^2b)$	$O\left(\frac{n^2b}{M}\right)$
Improved 1 Sweep SBR†	$8n^2b$	$O\left(\frac{n^2b^3}{M}\right)$	$O\left(\frac{n^2b^3}{M^2}\right)$

†assuming $1 \leq b \leq \sqrt{M}/3$

Asymptotics - singular values only - sequential case

Algorithm	Flops	Words	Messages
LAPACK	$4n^2b$	$O(n^2b)$	$O\left(n^2b\right)$
1 Sweep SBR	$8n^2b$	$O(n^2b)$	$O\left(\frac{n^2b}{M}\right)$
Improved 1 Sweep SBR†	$8n^2b$	$O\left(\frac{n^2b^3}{M}\right)$	$O\left(\frac{n^2b^3}{M^2}\right)$
CA-SBR†	$6n^2b$	$O\left(\frac{n^2b^2}{M}\right)$	$O\left(\frac{n^2b^2}{M^2}\right)$

†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 ω and doubles at each sweep

What if you want singular vectors too? - sequential case

We've used two optimizations:

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

What if you want singular vectors too? - sequential case

We've used two optimizations:

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

Algorithm	**Flops**	**Words**	**Messages**
LAPACK	$4n^3$	$O(n^2 b + n^3)$	$O\left(n^2 b + \frac{n^3}{M}\right)$
1 Sweep SBR	$4n^3$	$O\left(n^2 b + \frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^2 b}{M} + \frac{n^3}{M}\right)$

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 ω)
2. take multiple sweeps (increase c)
   - accumulating orthogonal transformations costs $O(n^3)$ flops per sweep

Algorithm	Flops	Words	Messages
LAPACK	$4n^3$	$O(n^2b + n^3)$	$O\left(n^2b + \frac{n^3}{M}\right)$
1 Sweep SBR	$4n^3$	$O\left(n^2b + \frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^2b}{M} + \frac{n^3}{M}\right)$
Improved 1 Sweep SBR†	$4n^3$	$O\left(\frac{n^2b^3}{M} + \frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^2b^3}{M^2} + \frac{n^3}{M^{3/2}}\right)$

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 ω)
2. take multiple sweeps (increase c)
   - accumulating orthogonal transformations costs $O(n^3)$ flops per sweep

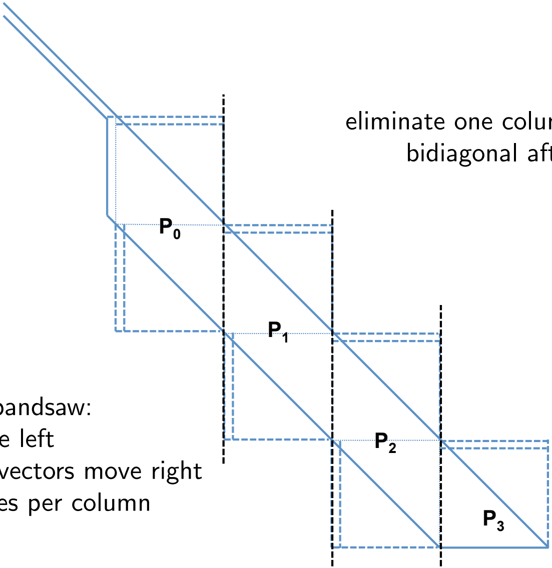
Algorithm	Flops	Words	Messages
LAPACK	$4n^3$	$O(n^2 b + n^3)$	$O\left(n^2 b + \frac{n^3}{M}\right)$
1 Sweep SBR	$4n^3$	$O\left(n^2 b + \frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^2 b}{M} + \frac{n^3}{M}\right)$
Improved 1 Sweep SBR†	$4n^3$	$O\left(\frac{n^2 b^3}{M} + \frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^2 b^3}{M^2} + \frac{n^3}{M^{3/2}}\right)$
CA-SBR†	$2n^3 \log b$	$O\left(\frac{n^2 b}{\sqrt{M}} + \frac{n^3 \log b}{\sqrt{M}}\right)$	$O\left(\frac{n^2 \log b}{M} + \frac{n^3 \log b}{M^{3/2}}\right)$

communication costs: band reduction + orthogonal updates

†assuming $1 \leq b \leq \sqrt{M}/3$

Parallel 1 Sweep SBR

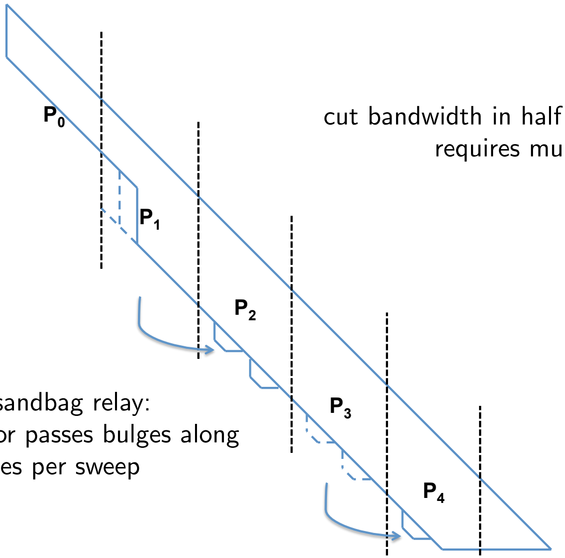
eliminate one column at a time;
bidiagonal after one sweep



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

[Lang 1993]

Parallel CA-SBR



cut bandwidth in half each sweep;
requires multiple sweeps

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

Asymptotics - singular values only - parallel case

Multiple sweeps and chasing multiple bulges reduces latency cost

Algorithm	Flops	Words	Messages
1 Sweep SBR	$O\left(\frac{n^2 b}{p}\right)$	$O(nb)$	$O(n)$
CA-SBR†	$O\left(\frac{n^2 b}{p}\right)$	$O(nb)$	$O(p \log b)$

†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

Algorithm	**Flops**	**Words**	**Messages**
1 Sweep SBR*	$\frac{4n^3}{p}$	$O\left(nb + \frac{n^2}{\sqrt{p}}\right)$	$O(n + \log p)$
CA-SBR†	$\frac{2n^3 \log b}{p}$	$O\left(nb + \frac{n^2}{\sqrt{p}} \log b\right)$	$O(\sqrt{p} \log b)$

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

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 $1 \leq \#$ 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 I

Aggarwal, A., and Vitter, J. S.
The input/output complexity of sorting and related problems.
Comm. ACM 31, 9 (1988), 1116–1127.

Agullo, E., Dongarra, J., Hadri, B., Kurzak, J., Langou, J., Langou, J., Ltaief, H., Luszczek, P., and YarKhan, A.
PLASMA users' guide, 2009.
http://icl.cs.utk.edu/plasma/.

Ballard, G., Demmel, J., Holtz, O., and Schwartz, O.
Minimizing communication in linear algebra.
SIAM Journal on Matrix Analysis and Applications 32, 3 (2011), 866-901.

Bischof, C., Lang, B., and Sun, X.
A framework for symmetric band reduction.
ACM Trans. Math. Soft. 26, 4 (2000), 581–601.

Bischof, C. H., Lang, B., and Sun, X.
Algorithm 807: The SBR Toolbox—software for successive band reduction.
ACM Trans. Math. Soft. 26, 4 (2000), 602–616.

Demmel, J., Grigori, L., Hoemmen, M., and Langou, J.
Communication-optimal parallel and sequential QR and LU factorizations.
SIAM J. Sci. Comput. (2011). To appear.

References II

Dongarra, J., Hammarling, S., and Sorensen, D.
Block reduction of matrices to condensed forms for eigenvalue computations.
Journal of Computational and Applied Mathematics 27 (1989).

Fuller, S. H., and Millett, L. I., Eds.
The Future of Computing Performance: Game Over or Next Level?
The National Academies Press, Washington, D.C., 2011.

Haidar, A., Ltaief, H., and Dongarra, J.
Parallel reduction to condensed forms for symmetric eigenvalue problems using aggregated fine-grained and memory-aware kernels.
Proceedings of the ACM/IEEE Conference on Supercomputing (2011).

Howell, G., Demmel, J., Fulton, C., Hammarling, S., and Marmol, K.
Cache efficient bidiagonalization using BLAS 2.5 operators.
ACM Trans. Math. Softw. 34, 3 (2008), 14:1-14:33.

Kaufman, L.
Banded eigenvalue solvers on vector machines.
ACM Trans. Math. Softw. 10 (1984), 73–86.

Kaufman, L.
Band reduction algorithms revisited.
ACM Trans. Math. Softw. 26 (December 2000), 551–567.

References III

Lang, B.
A parallel algorithm for reducing symmetric banded matrices to tridiagonal form.
SIAM J. Sci. Comput. 14, 6 (1993), 1320–1338.

Lang, B.
Efficient eigenvalue and singular value computations on shared memory machines.
Par. Comp. 25, 7 (1999), 845 – 860.

Ltaief, H., Luszczek, P., and Dongarra, J.
High performance bidiagonal reduction using tile algorithms on homogeneous multicore architectures.
Tech. Rep. 247, LAPACK Working Note, May 2011.
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.
In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium* (2011).

Murata, K., and Horikoshi, K.
A new method for the tridiagonalization of the symmetric band matrix.
Information Processing in Japan 15 (1975), 108–112.

References IV

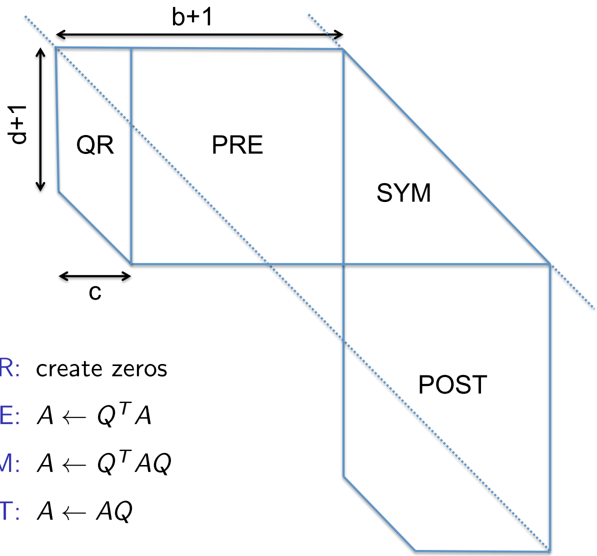
RAJAMANICKAM, S.
Efficient Algorithms for Sparse Singular Value Decomposition.
PhD thesis, University of Florida, 2009.

RUTISHAUSER, H.
On Jacobi rotation patterns.
In *Proceedings of Symposia in Applied Mathematics* (1963), vol. 15, pp. 219–239.

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.
Numerische Mathematik 12 (1968), 231–241.

Anatomy of a symmetric bulge-chase



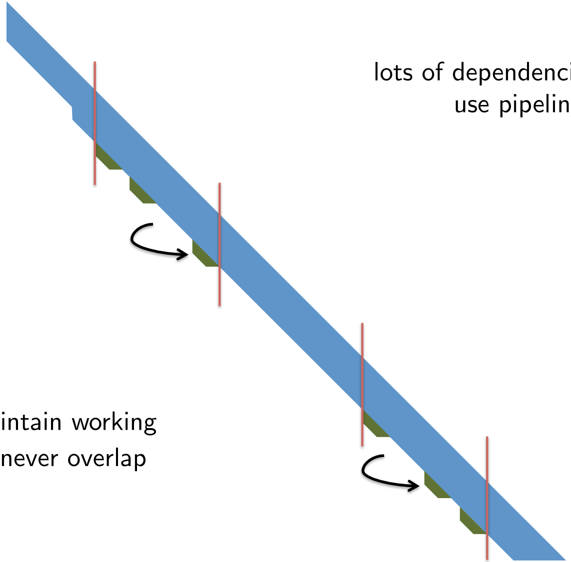
QR: create zeros

PRE: $A \leftarrow Q^T A$

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

POST: $A \leftarrow AQ$

Shared-Memory Parallel Implementation



lots of dependencies:
use pipelining

threads maintain working
sets which never overlap