Communication-Avoiding Successive Band Reduction

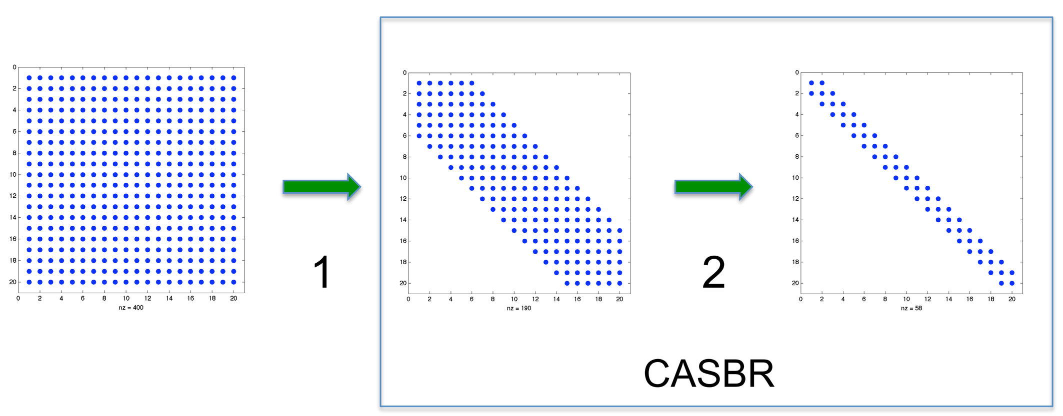
Grey Ballard and Nicholas Knight

{ballard,knight}@cs.berkeley.edu

Symmetric Eigenproblem

Standard approaches to computing the eigenvalues and eigenvectors of a symmetric matrix use orthogonal similarity transformations to reduce the matrix to tridiagonal form

- (Sca)LAPACK directly reduces full matrix to tridiagonal form
- We consider a two-stage approach
  - 1^{st} stage reduces full matrix to band form
  - 2^{nd} stage reduces band to tridiagonal using *successive band reduction*

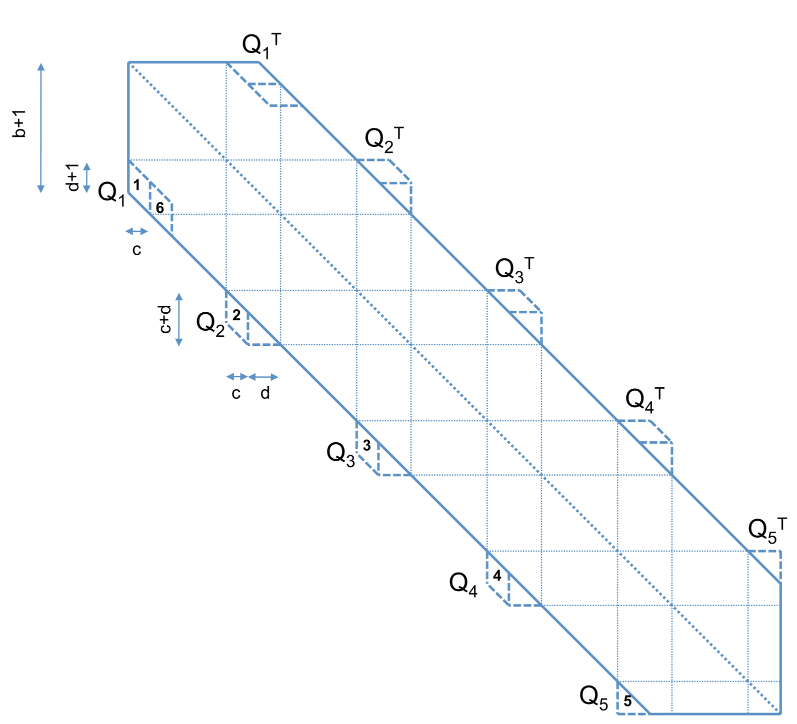


Successive Band Reduction

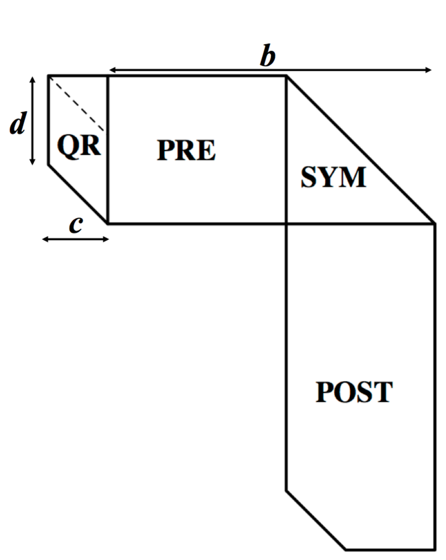
In successive band reduction, we reduce the bandwidth of the matrix by zeroing out parallelograms and chasing the trapezoidal-shaped fill or "bulge" (multiple times) off the band

Bulge-chasing

The picture below shows the annihilation of one parallelogram and its bulges through the bulge-chasing process



Anatomy of a bulge-chase



QR: compute Q to annihilate parallelogram and update triangle

PRE: apply Q from left to columns of rectangle

SYM: apply Q from left and Q^T from right to lower triangle of symmetric square

POST: apply Q^T from right to rows of rectangle

Asymptotic Analysis

Communication-avoiding approaches require asymptotically less data movement then existing algorithms in the sequential two-level memory model

$n =$ matrix dimension $b =$ matrix bandwidth $M =$ fast memory size

Full reduction

Direct tridiagonalization suffers from high communication costs, whereas reducing to banded form can be done efficiently with blocked algorithms

	Flops	Words	Messages
LAPACK	$\frac{4}{3}n^3$	$O(n^3)$	$O\left(\frac{n^3}{M}\right)$
Full-to-banded	$\frac{4}{3}n^3$	$O\left(\frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^3}{M^{3/2}}\right)$
CASBR	$5n^2\sqrt{M}$	$O(n^2)$	$O\left(\frac{n^2}{M}\right)$

Band reduction

Most band reduction algorithms achieve only $O(1)$ data re-use, whereas CASBR achieves $O(b)$ re-use when $b \leq \sqrt{M}$

	Flops	Words	Messages
LAPACK	$4n^2b$	$O(n^2b)$	$O(n^2b)$
CASBR	$5n^2b$	$O(n^2)$	$O\left(\frac{n^2}{M}\right)$

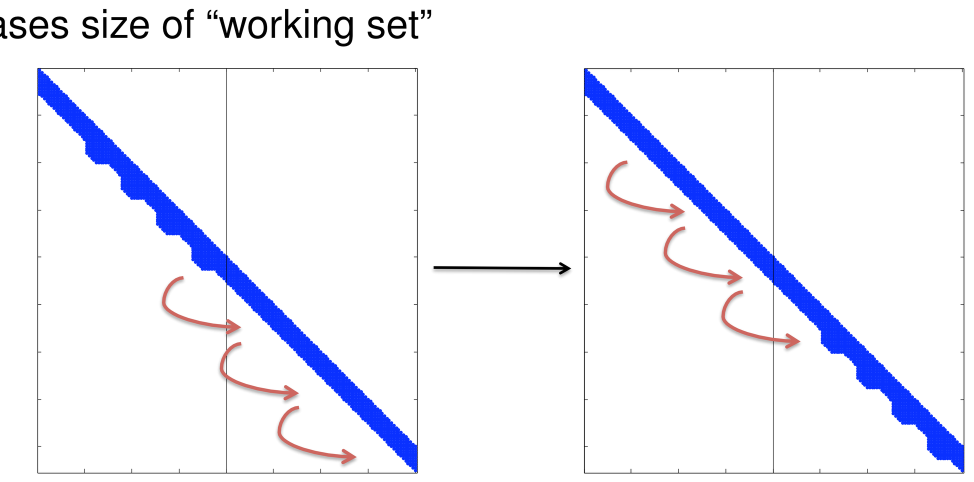
Avoiding Communication

Obtaining data locality

There are two main approaches to avoiding communication (i.e., obtaining locality) with SBR:

(a) Increase the number of columns (c) in each parallelogram
- permits use of BLAS-3 kernels which attain O(c) data re-use
- reduces number of diagonals (d) eliminated in current sweep

(b) Increase the number of bulges (`mult`) chased at a time
- decreases number of times band is read from slow memory
- increases size of "working set"

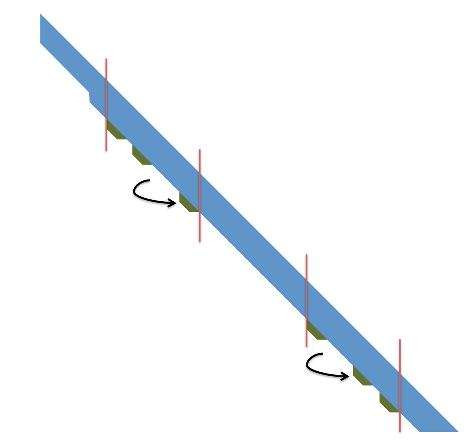


Tuning Parameters

1. Number of sweeps and diagonals per sweep: $\{d_i\}$ (such that $b = \sum d_i$)
2. Bulge parameters for i^{th} sweep
   (a) number of threads: p_i
   (b) the number of columns in each parallelogram: c_i (such that $c_i + d_i \leq b_i$)
   (c) the number of bulges chased at a time: $\texttt{mult}_i$
   (d) the number of times each bulge is chased at a time: $\texttt{hops}_i$
3. Implementation of single bulge-chase (choice of subroutines, data structure)

Shared-Memory Parallel Implementation

We have extended our sequential implementation to shared-memory parallel machines by exploiting pipeline parallelism

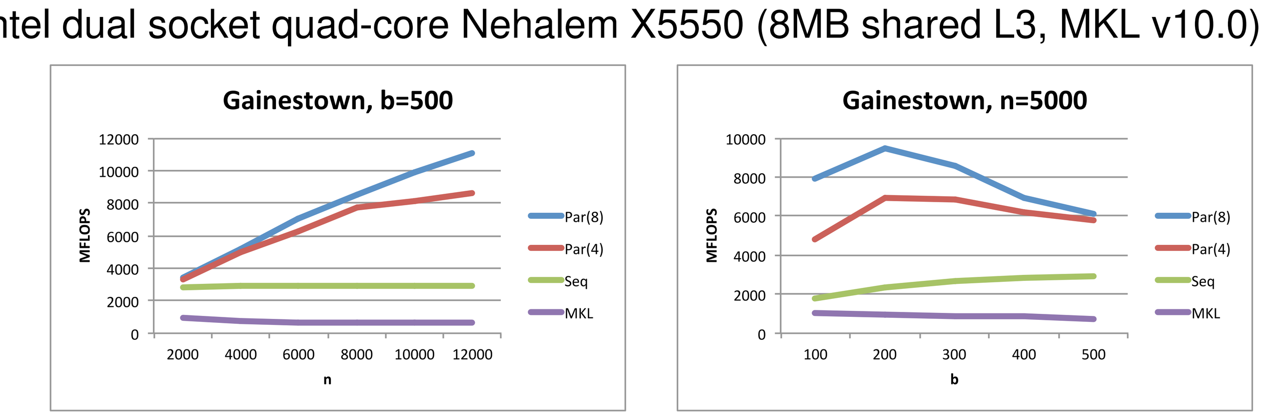


Performance Results

We show performance results of CASBR against LAPACK's DSBTRD
- CASBR has not been fully tuned; parameters were heuristically chosen
  - 2 sweeps ($b_2 = 48$), $c_i = b_i - d_i$, $\texttt{mult}_i = \texttt{hops}_i = 1$

Gainestown

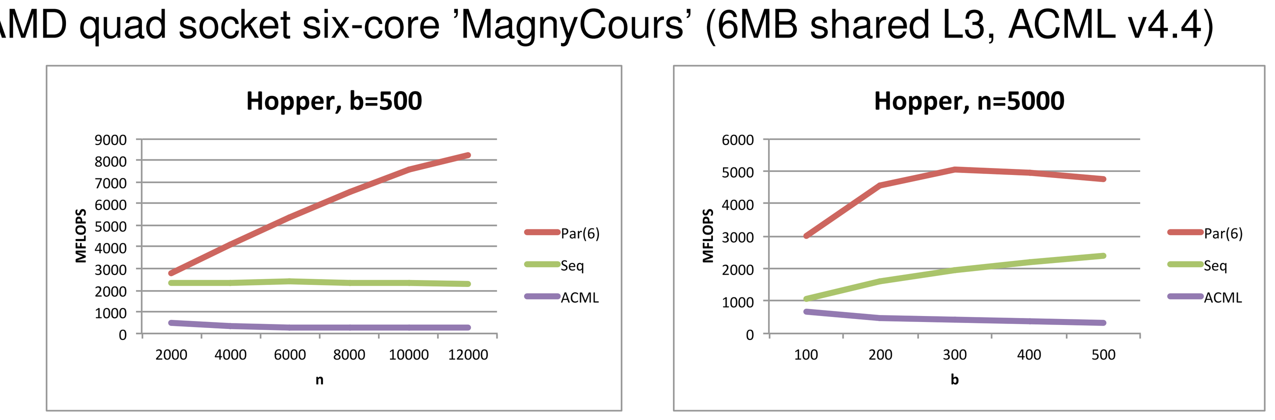
Intel dual socket quad-core Nehalem X5550 (8MB shared L3, MKL v10.0)



Best parallel speedup over MKL: $17\times$ ($n = 12000$, $b = 500$, 8 threads)
Best sequential speedup over MKL: $4.5\times$ ($n = 12000$, $b = 500$)
Best parallel efficiency: $3\times$ over sequential ($n = 12000$, $b = 500$, 4 threads)

Hopper

AMD quad socket six-core 'MagnyCours' (6MB shared L3, ACML v4.4)



Best parallel speedup over ACML: $30\times$ ($n = 12000$, $b = 500$, 6 threads)
Best sequential speedup over ACML: $8\times$ ($n = 8000$, $b = 500$)
Best parallel efficiency: $3.6\times$ over sequential ($n = 12000$, $b = 500$, 6 threads)

Future Work

- Use autotuning framework to optimize CASBR across several platforms
- Implement distributed-memory parallel algorithm (MPI and NUMA-aware)
- Handle eigenvector updates (results shown here are for eigenvalues only)
- Prove a lower bound to show that CASBR is asymptotically optimal

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.