Symmetric Eigenvalue Problem: Reduction to Tridiagonal

Grey Ballard, Mehrzad Tartibi

CS267 Spring 2009

Symmetric Eigenvalue Problem

- Eigenvalues and eigenvectors are usually computed via two steps [Dem97]
  - Reduction of the matrix to tridiagonal form using two-sided orthogonal transformations
  - Solve eigenproblem on tridiagonal matrix (e.g. QR iteration, divide-and-conquer)
    - Eigenvalues of tridiagonal matrix are the same as the original matrix
    - Eigenvectors of tridiagonal matrix are transformations of those of the original matrix
- The bulk of the work (and communication) of the entire algorithm occurs in the reduction to tridiagonal step
- In this project, we assume that only the eigenvalues of the matrix are desired

Single Step vs. Multistep Reduction Algorithms

Utilizing BLAS 3

- Naive approach (column-by-column) is completely BLAS 2
- Single step reduction: LAPACK approach does multiple trailing matrix updates at once
  - column panel reduction still BLAS 2
  - trailing matrix update becomes BLAS 3
- Multistep reduction: first reduce full matrix to banded (phase 1), then reduce banded matrix to tridiagonal (phase 2)
  - phase 1 performs QR on column panels (entirely BLAS 3)
  - phase 2 requires "chasing the bulge"

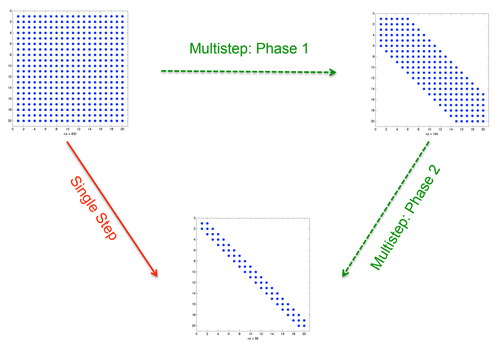

Figure 1: *Single step and multistep reduction paths.*

Algorithm 1: *full→tridiagonal*

Input: A is symmetric $n \times n$ matrix
1: **for** $i = 1$ to n **step** b **do**
2: **for** $j = i$ to $i + b - 1$ **do**
3: Reduce column j to tridiagonal
4: Update columns $j + 1$ to $i + b - 1$
▷ store Householder vector info
5: **end for**
6: Update trailing submatrix
7: **end for**
Output: A is symmetric tridiagonal with same eigenvalues

Algorithm 2: *full→banded*

Input: A is symmetric $n \times n$ matrix
1: **for** $i = 1$ to $n - b$ **step** b **do**
2: Reduce $A(i + b : n, i : i + b)$ to upper triangular
▷ Use TSQR
3: Update trailing submatrix
4: **end for**
Output: A is symmetric banded (bandwidth b) with same eigenvalues

"Chasing the Bulge" [BLS00b]

- Different data-structures and algorithms are used to reduce banded matrix to tridiagonal
- Large design space, but general idea consists of repeatedly
  - annihilating one or several elements, and
  - bulge chasing to restore the banded form
- Parameters include number of elements/bands to annihilate at a time and how much of the bulge to chase
  - tradeoffs between extra flops/storage and performance

Current Performance

Sequential Performance

- Multistep reduction outperforms single step reduction for $N \geq 1200$
  - BLAS 2 computation in single step kills performance when the problem no longer fits in L2 cache (6 MB)
  - Multistep incurs overhead in extra computation and repacking the matrix into banded form
- Intermediate bandwidth for multistep is tunable parameter ($b = 64$ generally performs well for large matrices)

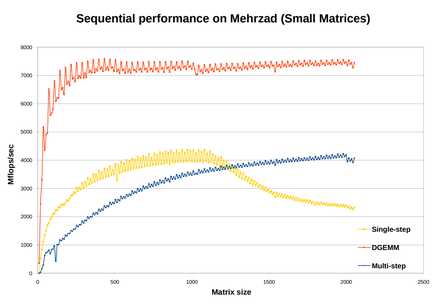

Figure 2: *Sequential performance for small matrices.*

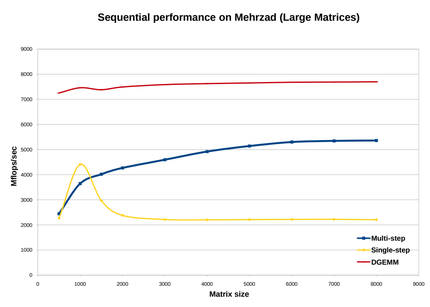

Figure 3: *Sequential performance for large matrices.*

Multithreaded Performance

- Multistep outperforms single step for $N \geq 3000$
- More L2 cache is available in multi-threaded mode
- All measurements were taken using 8 threads
- Large performance gap compared to DGEMM

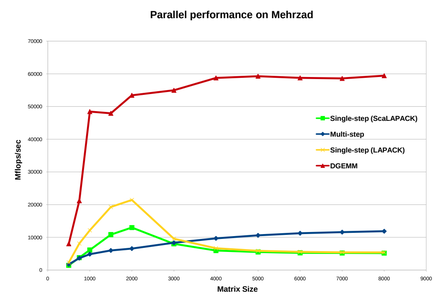

Figure 4: *Parallel performance for large matrices.*

Processor Architecture

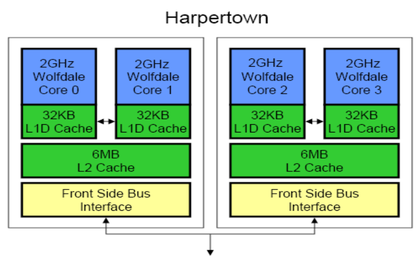

Figure 5: *Processor architecture.*

- All performance data was gathered on "Mehrzad"
- 2 quad-core Intel Xeon E5405 (Harpertown) processors
  - 2x6 MB L2 cache on each chip
  - 32 KB L1 cache for each core
  - 2.0 GHz sequential peak performance

Our Goal: Optimizing Reduction to Banded

Coarse-grain data flow model

- We will use the coarse-grain approach of "algorithms-by-tiles" or "algorithms-by-blocks" (see [CVZB+08], [LKD09])
  - matrix blocks become fundamental units of data, operations on blocks become fundamental (sequential) computations (matrix is stored in contiguous blocks)
  - static or dynamic scheduling and out-of-order execution determined by block computation DAG

Successive Band Reduction (SBR) toolbox [BLS00a]

- The SBR toolbox contains Fortran routines which supplement LAPACK and include
  - full to banded, banded to narrower banded, and banded to tridiagonal reductions
  - repacking routines to convert between full and banded storage schemes
- Most of the running time is spent in the full to banded reduction routine
  - for $N = 2048$, $b = 32$, 83% of time is spent in reduction to banded (in sequential case)
- Optimizing banded to tridiagonal reduction for multi-core is an open problem

SMP Superscalar (SMPSs) framework [Bar08]

- We have chosen to use the SMPSs framework in which
  - the programmer identifies atomic tasks and specifies input/output dependencies
  - a runtime library schedules tasks to cores based on computation DAG
- SMPSs scheduling performed almost as well as handwritten static scheduling for LU, QR, Cholesky [KLD+09]
- SMPSs scheduling performed as well as handwritten dynamic and handwritten static scheduling of Hessenberg and bidiagonal reductions (just to banded form) [LKD09]
  - Hessenberg reduction becomes tridiagonal reduction in the symmetric case

References

[Bar08] Barcelona Supercomputing Center. *SMP Superscalar (SMPSs) User's Manual*, May 2008.

[BLS00a] Christian H. Bischof, Bruno Lang, and Xiaobai Sun. Algorithm 807: The SBR toolbox—software for successive band reduction. *ACM Trans. Math. Softw.*, 26(4):602–616, 2000.

[BLS00b] Christian H. Bischof, Bruno Lang, and Xiaobai Sun. A framework for symmetric band reduction. *ACM Trans. Math. Softw.*, 26(4):581–601, 2000.

[CVZB+08] Ernie Chan, Field G. Van Zee, Paolo Bientinesi, Enrique S. Quintana-Orti, Gregorio Quintana-Orti, and Robert van de Geijn. Supermatrix: a multithreaded runtime scheduling system for algorithms-by-blocks. In *PPoPP '08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 123–132, New York, NY, USA, 2008. ACM.

[Dem97] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1st edition, August 1997.

[KLD+09] Jakub Kurzak, Hatem Ltaief, Jack Dongarra, , and Rosa M. Badia. Scheduling linear algebra operations on multicore processors. Technical Report 213, LAPACK Working Note, February 2009.

[LKD09] Hatem Ltaief, Jakub Kurzak, and Jack Dongarra. Scheduling two-sided transformations using algorithms-by-tiles on multicore architectures. Technical Report 214, LAPACK Working Note, February 2009.