Parallel Block Hessenberg Reduction using Algorithms-By-Tiles for Multicore Architectures Revisited

Hatem Ltaief

ICL FRIDAY LUNCH
August 15th, 2008
1 Motivations

2 Orthogonal Transformations

3 Sequential HR

4 Parallel Block HR

5 Results

6 Future Work
Solving Dense Linear Algebra Problems.

Target: standard (non-symmetric) EVP

\[ Ax = \lambda x, \]
\[ A \in \mathbb{R}^{n \times n}, \quad x \in \mathbb{R}^{n}, \quad \lambda \in \mathbb{R}. \]

Transform the original problem into an easier one (pre-processing)

\[ H = Q \times A \times Q^T, \]
\[ A \in \mathbb{R}^{n \times n}, \quad Q \in \mathbb{R}^{n \times n}, \quad H \in \mathbb{R}^{n \times n}. \]
Evolution of Chips:

- Single-core (1950)
- Dual-core (2004)
- Quad-core (2006)
- Nine-core (2005) → Multicore

Important fact: 98% of the fastest parallel systems in the world are based on multicores (Top500, June08).
Evolution of Chips:

- **Single-core** (1950)
- **Dual-core** (2004)
- **Quad-core** (2006)
- **Nine-core** (2005) → **Multicore**
- **Many core!??**

**Important fact:** 98% of the fastest parallel systems in the world are based on multicore technologies (Top500, June08).
Evolution of Chips:

- Single-core (1950)
- Dual-core (2004)
- Quad-core (2006)
- Nine-core (2005)
- Many core!?

Important fact: 98% of the fastest parallel systems in the world are based on multicores (Top500, June08).
Evolution of Chips:

- **Single-core** (1950)
- **Dual-core** (2004)
- **Quad-core** (2006)
- **Nine-core** (2005)
- **Multicore**

Many core!?

**Important fact:** 98% of the fastest parallel systems in the world are based on multicores (Top500, June08).
Evolution of Chips:

- Single-core (1950)
- Dual-core (2004)
- Quad-core (2006)
- Nine-core (2005)
- Multicore

Important fact: 98% of the fastest parallel systems in the world are based on multicores (Top500, June08).
Evolution of Chips:

**Single-core** (1950)

**Dual-core** (2004)

**Quad-core** (2006)

- - - - - - >

**Many core!?**

Important fact: 98% of the fastest parallel systems in the world are based on multicores (Top500, June08).
Evolution of Chips:

- Single-core (1950)
- Nine-core (2005)
- Dual-core (2004)
- Quad-core (2006)
- Multicore

Important fact: 98% of the fastest parallel systems in the world are based on multicores (Top500, June08).
Evolution of Chips:

- Single-core (1950)
- Dual-core (2004)
- Quad-core (2006)
- Nine-core (2005)
- Many core!? (2005)

Important fact: 98% of the fastest parallel systems in the world are based on multicore architectures (Top500, June08).
Algorithms-By-Tiles for one-sided transformations (Buttari et al [2007]):

1. Fine Granularity
2. Asynchronicity
3. Block Data Layout
4. Dynamic Data Driven Scheduler

Extend to two-sided transformations.

Operation counts: 1-sided < 2-sided, e.g. QR ($4/3n^3$) and HR ($10/3n^3$).
Investigations:

1. Householder reflectors
2. Givens rotations
3. Fast Givens rotations
Quick Hessenberg demo
Quick Hessenberg demo
Quick Hessenberg demo
Quick Hessenberg demo
BLAS-2 Algorithm

**Algorithm 1** Hessenberg Reduction with Householder reflectors

1: for $j = 1$ to $n - 2$ do
2:     $x = A_{j+1:n,j}$
3:     $v_j = \text{sign}(x_1) \|x\|_2 \, e_1 + x$
4:     $v_j = v_j / \|v_j\|_2$
5:     $A_{j+1:n,j:n} = A_{j+1:n,j:n} - 2 \, v_j (v_j^* A_{j+1:n,j:n})$
6:     $A_{1:n,j+1:n} = A_{1:n,j+1:n} - 2 \,(A_{j+1:n,j:n} \, v_j) \, v_j^*$
7: end for

- Cons: quite challenging to implement in parallel.
- Pros: 4 flops per annihilated element, efficient reflector accumulation procedure (WY compact).
BLAS-1 Algorithm

Algorithm 2: Hessenberg Reduction with Givens rotations

1: $G \leftarrow \text{Id}_n$
2: for $j = 1, 2$ to $n - 2$
3:   for $i = n, n - 1$ to $j + 2$
4:     Build the local $g(i - 1, i)$ such that $A_{i,j} = 0$
5:     Accumulate $G = g(i - 1, i) \times G$
6:     Update $A = g^T(i - 1, i) \times A$
7:     Update $A = A \times g(i, i - 1)$
8:   end for
9: end for

- Cons: 6 flops per annihilated element, expensive rotation accumulation procedure (Stewart [1974]).
- Pros: easy to implement in parallel.
Issues with Fast Givens rotations:

1. Stability: risk of underflow/overflow
2. Structure more complicated
3. Need a bookkeeping procedure
   \[\Rightarrow\] No efficient way to automatically accumulate
4. Example: Chained Fast Givens rotations (Anda and Park [1994])
Orthogonal transformations considered for the parallel implementation:

1. Householder reflectors
2. Givens rotations
6 similar kernels for HH and GIV:

1. Factorization of a sub-diagonal block.
2. Application of the left transformation of a sub-diagonal block.
3. Application of the right transformation of a sub-diagonal block.
4. Factorization of a block composed by a sub-diagonal block on top of a square block.
5. Application of the left transformation of a sub-diagonal block.
6. Application of the right transformation of a sub-diagonal block.

Figure: Givens rotation matrix produced by DTSQRG used during the update procedures.
Figure: Scheduling of the Left Orthogonal Transformation.
Figure: Scheduling of the Right Orthogonal Transformation.
Figure: Tracing of Dynamic Data Driven Execution with 8 cores.
Why a block HR and not a full HR?

Operation counts: \( b \) is the block size

\[
\text{block HR } 10/3 n (n - b) (n - b) \approx \text{full HR } 10/3 n^3
\]

\( O(n^2 b) \) less flops

Assumed \( b << n \).

QR Algorithm on a BH matrix.
Figure: Elapsed time in seconds for the Block Hessenberg Reduction on a dual-socket quad-core Intel Itanium2 1.6 GHz with MKL BLAS V10.0.1.
Figure: Parallel Performance of the Block Hessenberg Reduction on a dual-socket quad-core Intel Itanium2 1.6 GHz processors with MKL BLAS V10.0.1.
Figure: Elapsed time in seconds for the Block Hessenberg Reduction on a quad-socket quad-core Intel Xeon 2.4 GHz processors with MKL BLAS V10.0.1.
Figure: Parallel Performance of the Block Hessenberg Reduction on a quad-socket quad-core Intel Xeon 2.4 GHz processors with MKL BLAS V10.0.1.
QZ Algorithm (Kagstrom et. al, U of Umea).
Band Tri-Diagonalization reduction (symmetric EVP).
Band Bi-Diagonalization reduction (SVD).
Exploit any natural redundancy within the DAG for FT purposes.
Thank You!