Extreme-Scale Eigenvalue Reordering in the Real Schur Form

Lars Karlsson, Mirko Myllykoski, Björn Adlerborn, Carl Christian Kjelgaard Mikkelsen, and Bo Kågström

Department of Computing Science and HPC2N
Umeå University

SIAM-CSE-2017
The problem
The kernel

- Solve tiny Sylvester equation
  \[ AX - XB = \gamma C. \]
- Compute small QR factorization
  \[
  \begin{bmatrix}
  -X \\
  \gamma I
  \end{bmatrix}
  = QR
  \]
- Apply orthogonal transformation
Scalar code (DTRSEN, LAPACK)

- Slide each eigenvalue up the diagonal as in bubble-sort
- Low arithmetic intensity
- Poor cache utilization
Block code (BDTRSEN)

- Move eigenvalues within small window
- Apply transform to block rows and columns
- Slide window up and repeat
Parallel blocked code (PBDTRSEN, SCALAPACK)
Task based programming

Figure: Move a single block to the top
### Table of operations

<table>
<thead>
<tr>
<th>symbol</th>
<th>operation</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Swap adjacent blocks</td>
<td>$S(3, 4)$</td>
</tr>
<tr>
<td>R</td>
<td>Right update of block column</td>
<td>$R(i &lt; 3, 3 : 4)$</td>
</tr>
<tr>
<td>L</td>
<td>Left update of block row</td>
<td>$L(3 : 4, 3 &lt; j)$</td>
</tr>
</tbody>
</table>
Compressed DAG

\[ R(i < 4, 4 : 5) \]
\[ R(i < 3, 3 : 4) \]
\[ R(i < 2, 2 : 3) \]
\[ S(4,5) \]
\[ S(3,4) \]
\[ S(2,3) \]
\[ L(3 : 4, 4 < j) \]
\[ L(2 : 3, 3 < j) \]
\[ L(1 : 2, 2 < j) \]

start

\[ \text{end} \]
Expanded DAG
An advantage of task based programming ...
BDTRSEN courtesy of Kressner - Thank you very much!

- Blocked code
- Move eigenvalues within windows
- Delays update of matrix

PBDTRSEN (ScaLAPACK) (Granat, Kressner, Kågström)

- Parallel block code
- Uses multiple windows
- Global synchronisation after row/column updates

Task based code running under StarPU (Mirko Myllykoski)

- Uses multiple windows
- Uses BDTRSEN to process windows
- Low level synchronisation among threads
Real Schur forms are built from a seed and parameters

- $n$, the dimension of the problem
  
  $n \in \{10000, 20000, 30000, 40000\}$

- $k$, the number of 2 by 2 real blocks
  
  $2k = \frac{n}{2}$

- $p$, the probability of choosing any diagonal block
  
  $p \in \{0.05, 0.15, 0.35, 0.50\}$

Eigenvalues are drawn from a grid of well separated points

- Ensures that “all” Sylvester equations well conditioned
The machine

<table>
<thead>
<tr>
<th>Machine K* (operational 2016)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon E5-2690v4</td>
</tr>
<tr>
<td>2 NUMA nodes per node</td>
</tr>
<tr>
<td>14 cores per NUMA node</td>
</tr>
<tr>
<td>Each core has its own FPU</td>
</tr>
<tr>
<td>Cores drawn from list 0:1:27</td>
</tr>
</tbody>
</table>

*Standard compute node on Kebnekaise at High Performance Computing Center North (HPC2N)*
For all $\lambda \in \lambda(A)$, we have

$$\frac{|\lambda - \tilde{\lambda}|}{|\lambda|} \lesssim 900u.$$ 

In addition, we have

$$\frac{\|Q^T AQ - \tilde{A}\|_F}{\|A\|_F} \lesssim 190u$$

and

$$\frac{\|Q^T Q - I\|_F}{\|I\|_F} \lesssim 315u.$$
Time to solve

MPI versus StarPU / Kebnekaise / 5% selected

- MPI (28 ranks)
- StarPU (28 workers)
Time to solve

MPI versus StarPU / Kebnekaise / 15% selected

- MPI (28 ranks)
- StarPU (28 workers)

Matrix dimension vs Runtime [s]
Time to solve

MPI versus StarPU / Kebnekaise / 35% selected

MPI (28 ranks)  
StarPU (28 workers)

Matrix dimension

Runtime [s]
Time to solve

MPI versus StarPU / Kebnekaise / 50% selected

- MPI (28 ranks)
- StarPU (28 workers)
Best serial code?

StarPU runtime / Kebnekaise / 5% selected

Runtime [s]
Matrix dimension
BDTRSEN
1 worker
4 workers
12 workers
20 workers
28 workers
Best serial code?
Best serial code?

StarPU runtime / Kebnekaise / 35% selected

Runtime [s]
Matrix dimension
StarPU runtime / Kebnekaise / 35% selected
BDTRSEN
1 worker
4 workers
12 workers
20 workers
28 workers

BDTRSEN / 44
1 worker / 22
Best serial code?

StarPU runtime / Kebnekaise / 50% selected

- BDTRSEN
- 1 worker
- 4 workers
- 12 workers
- 20 workers
- 28 workers

Runtime [s]
Matrix dimension
Strong scalability, StarPU

Strong scalability / StarPU / Kebnekaise / 5% selected

Efficiency

CPU cores / StarPU workers

N = 10000
N = 20000
N = 30000
N = 40000
Strong scalability, StarPU

![Graph showing strong scalability with StarPU, Kebnekaise, 15% selected, and various N values.]
Strong scalability, StarPU

Strong scalability / StarPU / Kebnekaise / 35% selected

Efficiency

CPU cores / StarPU workers

N = 10000
N = 20000
N = 30000
N = 40000
Strong scalability, StarPU

The diagram illustrates the strong scalability of StarPU on the Kebnekaise system with 50% selected. The x-axis represents the number of CPU cores/StarPU workers, and the y-axis represents the efficiency. The graph shows four lines corresponding to different values of N: 10000, 20000, 30000, and 40000.
Weak scalability

Weak scalability / StarPU / Kebnekaise

Efficiency

CPU cores / StarPU workers

N = 40,000 for 28 cores

5% selected

15% selected

35% selected

50% selected

50% selected

35% selected

15% selected

5% selected
Flop rate

Floating-point performance / Kebnekaise / 5% selected

Efficiency vs. CPU cores / StarPU workers for different values of N (10000, 20000, 30000, 40000).
Flop rate

Floating-point performance / Kebnekaise / 15% selected

Efficiency

CPU cores / StarPU workers

Floating-point performance / Kebnekaise / 15% selected

N = 40000
N = 30000
N = 20000
N = 10000

Efficiency

CPU cores / StarPU workers

0
0.2
0.4
0.6
0.8
1

N = 10000
N = 20000
N = 30000
N = 40000
Flop rate

Floating-point performance / Kebnekaise / 35% selected

Efficiency

CPU cores / StarPU workers

Floating-point performance / Kebnekaise / 35% selected

N = 40000
N = 30000
N = 20000
N = 10000
Flop rate

Floating-point performance / Kebnekaise / 50% selected

Efficiency

CPU cores / StarPU workers

N = 10000
N = 20000
N = 30000
N = 40000

Efficiency vs. CPU cores / StarPU workers for different values of N.
Idle time

Runtime distribution / Kebnekaise / 5% selected

Portion of execution time

Matrix dimension : StarPU workers

Overhead
Idle
Executing
Iden time

Runtime distribution / Kebnekaise / 15% selected

Portion of execution time

Matrix dimension : StarPU workers

Overhead
Idle
Executing
Idle time

Runtime distribution / Kebnekaise / 35% selected

Overhead
Idle
Executing

Portion of execution time

Matrix dimension : StarPU workers
Idle time

![Chart showing runtime distribution over different matrix dimensions and StarPU workers. The chart indicates the portion of execution time spent in idle, overhead, and executing phases.]
The end is near!

Thank you for your attention
Additional material and figures
Strong scalability, MPI

Strong scalability / MPI / Kebnekaise / 5% selected

Efficiency vs. CPU cores / MPI ranks for different values of $N$: $N = 10000$, $N = 20000$, $N = 30000$, $N = 40000$. The graph shows a decrease in efficiency as the number of CPU cores / MPI ranks increases.
Strong scalability, MPI

Efficiency vs. CPU cores / MPI ranks for different values of N (10000, 20000, 30000, 40000) on the Kebnekaise system with 15% of the selected cases.
Strong scalability, MPI

The diagram illustrates the efficiency of strong scalability with MPI on the Kebnekaise system. The graph shows the efficiency plotted against the number of CPU cores or MPI ranks for different data set sizes (N = 10000, N = 20000, N = 30000, N = 40000). As the number of CPU cores increases, the efficiency decreases, indicating diminishing returns on performance with additional resources.
Strong scalability, MPI

![Graph showing strong scalability with MPI on Kebnekaise with 50% selected.

- Efficiency vs. CPU cores / MPI ranks
- Lines for different N values: N = 10000 (purple), N = 20000 (green), N = 30000 (blue), N = 40000 (orange)

Data points suggest a decrease in efficiency as the number of CPU cores increases, with a notable drop around 20-24 cores for all N values.
Additional figures