Exploring OpenMP Task Priorities on the MR$^3$ Eigensolver

Jan Winkelmann    Paolo Bientinesi
{winkelmann,pauldj}@aices.rwth-aachen.de

Aachen Institute for Advanced Studies in Computational Engineering Science

12.04.2016
Symmetric Tridiagonal Eigensolver
Powers Lapack’s dense hermitian solver ZHEEVR
$O(n^2)$ complexity, Subset of Eigenpairs at reduced cost
Explicit parallelization required; no higher level BLAS
Control flow depends highly on spectrum of matrix
⇒ Spectrum influences performance, memory requirements
Symmetric Tridiagonal Eigensolver
Powers Lapack’s dense hermitian solver ZHEEVR
$O(n^2)$ complexity, Subset of Eigenpairs at reduced cost
Explicit parallelization required; no higher level BLAS
Control flow depends highly on spectrum of matrix
⇒ Spectrum influences performance, memory requirements
MR3SMP: A Parallel MR³ Solver

**MR3SMP**

- Shared memory parallel MR³ implementation
- MPI extension powers Elemental’s HermitianEig
- Available at: https://github.com/HPAC/mr3smp

---

*\textsuperscript{a}Petschow, Bientinesi. Parallel Computing, Volume 37(12), December 2011*

*\textsuperscript{b}http://libelemental.org/

**Parallelism**

- Uses Pthreads-based priority queue
- Scales even for large number of threads
- Does not rely on task execution order for correctness
- Previous Work shows potential for OpenMP
MR³ Parallelism: Tasks

Scan
Classify λ’s

Singleton
Calculate Eigenpair

clustered

Refine λ’s

separated

Cluster
Shift into new RRR
MR³ Parallelism: Tasks

\[ L_3 D_3 L_3^T = L_2 D_2 L_2^T - \sigma_3 I \]

\[ L_2 D_2 L_2^T = L_0 D_0 L_0^T - \sigma_2 I \]

\[ L_0 D_0 L_0^T = T - \sigma_0 I \]
MR³ “naturally” decomposes into tasks

- Naive implementations do not scale well
  \[ \Rightarrow \] Many small optimization necessary

- Trade-off: Parallelism and Memory requirement
```c
int fibonacci(int n) {
    int res1, res2;
    if ( n == 1 || n == 0 ) return n;
    #pragma omp task shared(res1)
        res1 = fib(n-1)
    #pragma omp task shared(res2)
        res2 = fib(n-2)
    #pragma omp taskwait
    return res1 + res2;
}
```

**Definition**

A specific instance of executable code and its *data environment*, generated when a *thread* encounters a *task* [...] construct.
#pragma omp task [...] priority(prio-val)

- Introduced recently in OpenMP 4.5
- Hint for execution order of tasks
- Higher values are recommended for earlier execution
- Priorities may not be relied upon for correctness
- Scheduling of OpenMP Tasks are implementation defined
OpenMP Task Priorities

```c
#pragma omp task [...] priority(prio-val)
```

- Introduced recently in OpenMP 4.5
- Hint for execution order of tasks
- Higher values are recommended for earlier execution
- Priorities may not be relied upon for correctness
- Scheduling of OpenMP Tasks are implementation defined

Support for Task Priorities in Runtimes

<table>
<thead>
<tr>
<th>Name</th>
<th>Supports Priorities?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel</td>
<td>No</td>
</tr>
<tr>
<td>GCC</td>
<td>No (Scheduled for 6.1)</td>
</tr>
<tr>
<td>Clang</td>
<td>Forthcoming</td>
</tr>
<tr>
<td>Mercurium</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\[\text{a}https://pm.bsc.es/mcxx\]
Porting MR3SMP to OpenMP

- Previous Work: Promising results even without priorities
- Small porting effort (2 days)
- No use of OpenMP dependencies
- Significant reduction of code complexity

Experiment Setup

- 48 Threads Intel Haswell
- Intel 16.0.2
- GCC 4.8.5
- Mercurium nightly, BF Scheduler \(^a\)

\(^a\)Version 1.99.9-2016-01-28
Timing Results

Speedup of OpenMP Runtimes vs Pthreads

- Gnu (no priorities)
- Intel (no priorities)
- Mercurium (priorities)

Application Matrix

- #1
  - n=4k
- #2
  - n=7k
- #3
  - n=8k
- #4
  - n=12k
- #5
  - n=13k
- #6
  - n=16k
- #7
  - n=24k
- #8
  - n=32k

Figure: Speedup of OpenMP Runtimes vs Pthreads
Do Priorities actually matter?

Differences between Priority Assignments (Matrix #7)

- Time (s)
- Memory (RRR)

- FIFO
- RSC
- RCS
- CSR
- SRC
- CRS
- SCR

Jan Winkelmann (AICES)
Special cases: Easier Coding and Testing

- Easier experimentation through simpler, more flexible code
  ⇒ Revealed complicated Performance Bug
- For some matrices memory footprint larger than expected
- Fix: different priority for small cluster tasks
- OpenMP: 2 lines of code, Pthreads: major code change
Special cases: Easier Coding and Testing

- Easier experimentation through simpler, more flexible code
- Revealed complicated Performance Bug
- For some matrices memory footprint larger than expected
- Fix: different priority for small cluster tasks
- OpenMP: 2 lines of code, Pthreads: major code change
Special cases: Easier Coding and Testing

Improvement by fixing "SCAS" Bug in OpenMP

<table>
<thead>
<tr>
<th>Application Matrix</th>
<th>Ratio of Old Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 n=4k</td>
<td></td>
</tr>
<tr>
<td>#2 n=7k</td>
<td></td>
</tr>
<tr>
<td>#3 n=8k</td>
<td></td>
</tr>
<tr>
<td>#4 n=12k</td>
<td></td>
</tr>
<tr>
<td>#5 n=13k</td>
<td></td>
</tr>
<tr>
<td>#6 n=16k</td>
<td></td>
</tr>
<tr>
<td>#7 n=24k</td>
<td></td>
</tr>
<tr>
<td>#8 n=32k</td>
<td></td>
</tr>
</tbody>
</table>

Figure: Improvement by fixing "SCAS" Bug in OpenMP
Parallel MR$^3$ using a complicated priority queue
- Easy port to OpenMP yields overall better performance
- More flexible code aids in finding Performance Bugs
Conclusion

- Parallel MR$^3$ using a complicated priority queue
- Easy port to OpenMP yields overall better performance
- More flexible code aids in finding Performance Bugs

OpenMP

- Even without priorities often performs well
- With priorities always performs better
- Better performance through
  - Code that is easier to read
  - Code that is more flexible to test
Conclusion

- Parallel MR$^3$ using a complicated priority queue
- Easy port to OpenMP yields overall better performance
- More flexible code aids in finding Performance Bugs

OpenMP

- Even without priorities often performs well
- With priorities always performs better
- Better performance through
  - Code that is easier to read
  - Code that is more flexible to test

Pthreads

- Strictly more expressive than OpenMP
- Harder to program and test
Financial support from the Deutsche Forschungsgemeinschaft (German Research Association) through grant GSC 111 is gratefully acknowledged.
Priorities: What we would expect

Differences between Priority Assignments (Matrix #7)

- **Runtime**
- **Memory**

- FIFO
- RSC
- RCS
- CSR
- SRC
- CRS

Time (s)

Memory (RRR)

Jan Winkelmann (AICES)
Relative distances

\[ \sigma(T) = \{\lambda_1, \ldots, \lambda_n\} \]

- Shifting \( T \) by \( \tau \):
  
  \[ \sigma(T - \tau) = \{\lambda_1 - \tau, \ldots, \lambda_n - \tau\} \]

- Shifting changes the magnitude of eigenvalues, not absolute distances

  \[ \text{reldist}(\lambda_i, \lambda_{i+1}) = \frac{|\lambda_i - \lambda_{i+1}|}{\max\{|\lambda_i|, |\lambda_{i+1}|\}} \]