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

MR^3

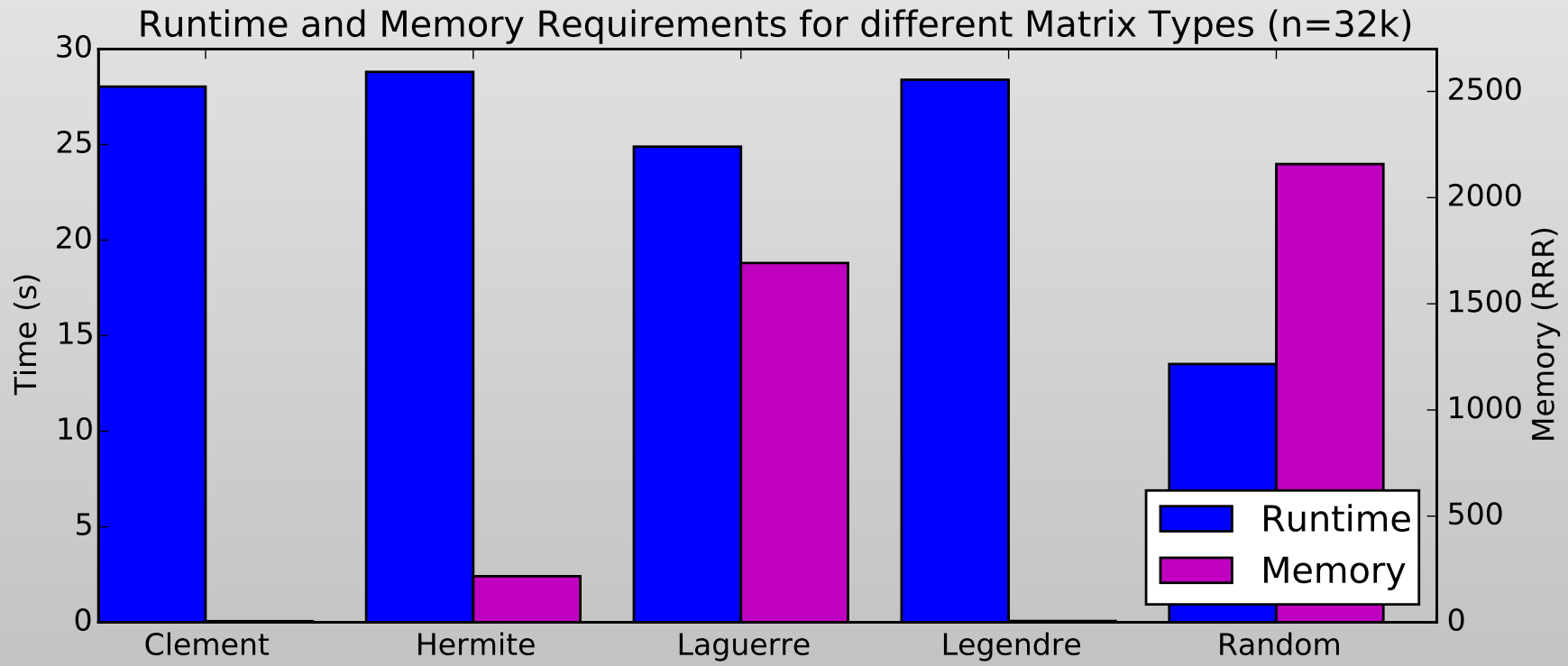
- Symmetric Tridiagonal Eigensolver
- Powers Lapack's dense hermitian solver `ZHEEVR`
- $O(n^2)$ complexity, Subset of Eigenpairs at reduced cost
- Explicit parallelization required; no higher level BLAS
- Control flow depends highly on spectrum of matrix

$\Rightarrow$ Spectrum influences performance, memory requirements

MR^3

- 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

$\Rightarrow$ Spectrum influences performance, memory requirements



MR3SMP: A Parallel MR^3 Solver

MR3SMP

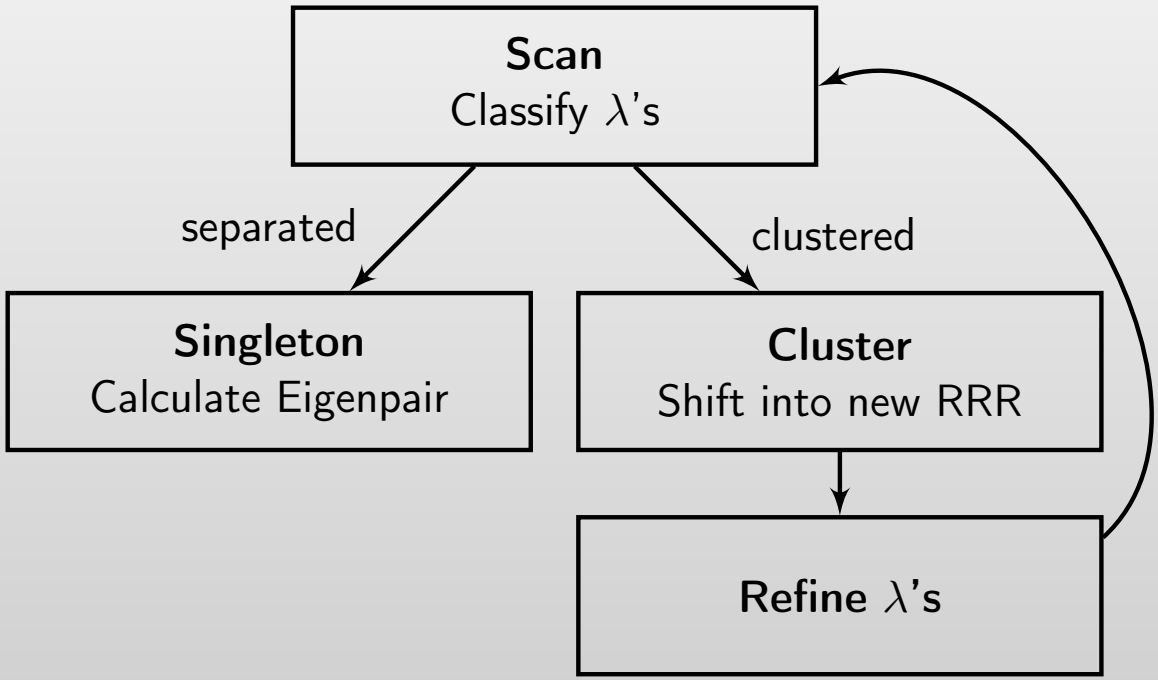
- Shared memory parallel MR^3 implementation [a]
- MPI extension powers Elemental's `HermitianEig`[b]
- Available at: `https://github.com/HPAC/mr3smp`

[a] Petschow, Bientinesi. Parallel Computing, Volume 37(12), December 2011
[b] http://libelemental.org/

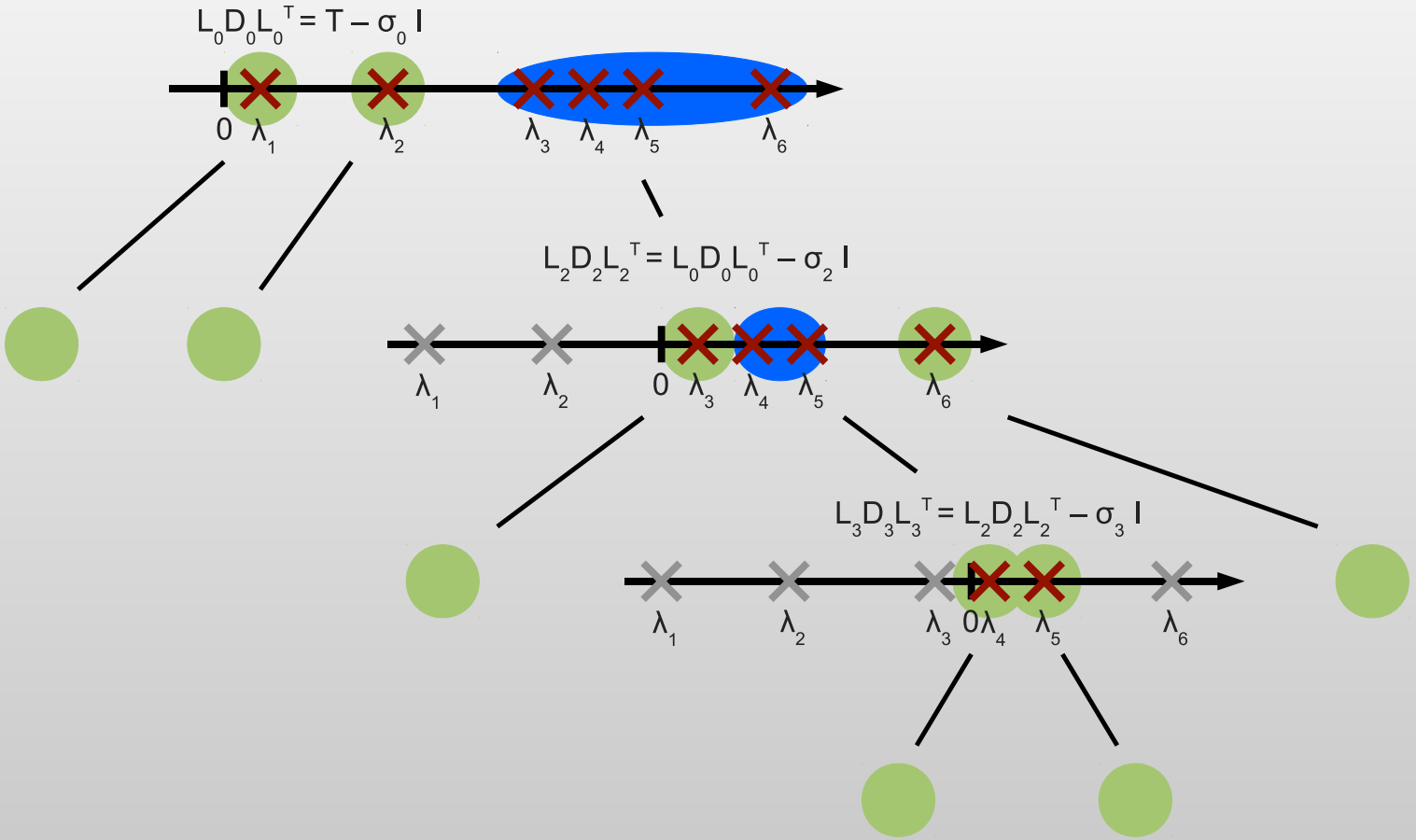
Parallelism

- 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

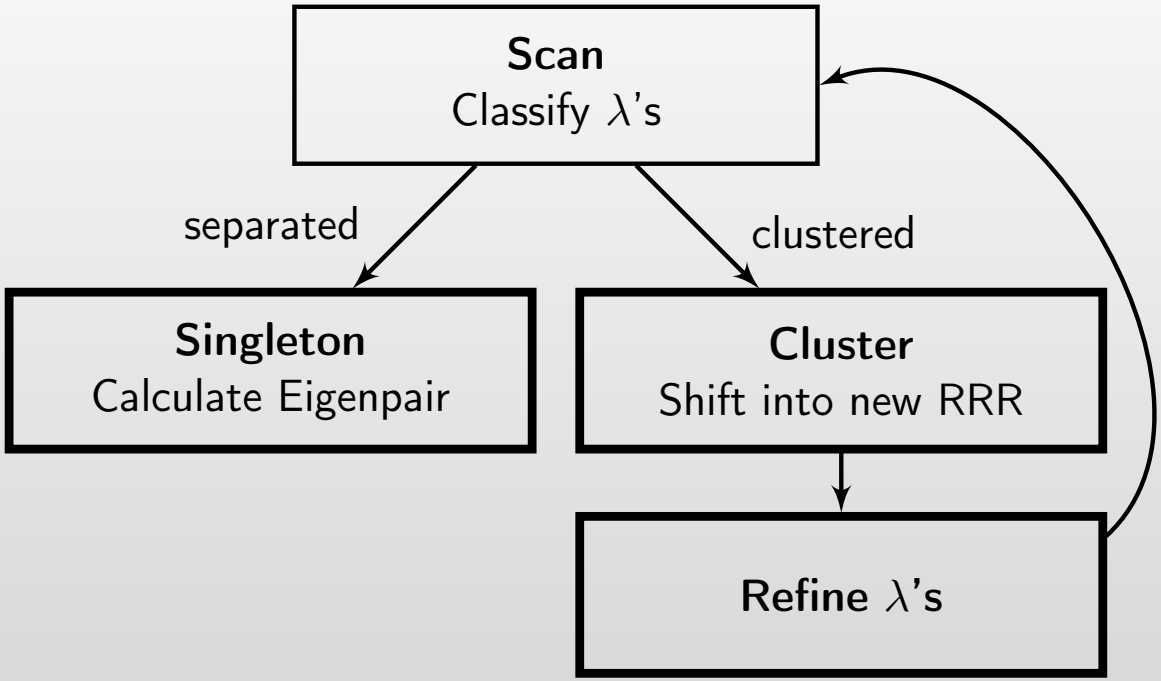
MR^3 Parallelism: Tasks



MR^3 Parallelism: Tasks



MR^3 Parallelism: Tasks



- MR^3 "naturally" decomposes into tasks
- Naive implementations do not scale well

$\Rightarrow$ Many small optimization necessary

- Trade-off: Parallelism and Memory requirement

OpenMP Tasks

```
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

OpenMP Task Priorities

```
#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

```
#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

Name	Supports Priorities?
Intel	No
GCC	No (Scheduled for 6.1)
Clang	Forthcoming
Mercurium [a]	Yes

[a] https://pm.bsc.es/mcxx

Experiment Setup

Porting MR3SMP to OpenMP

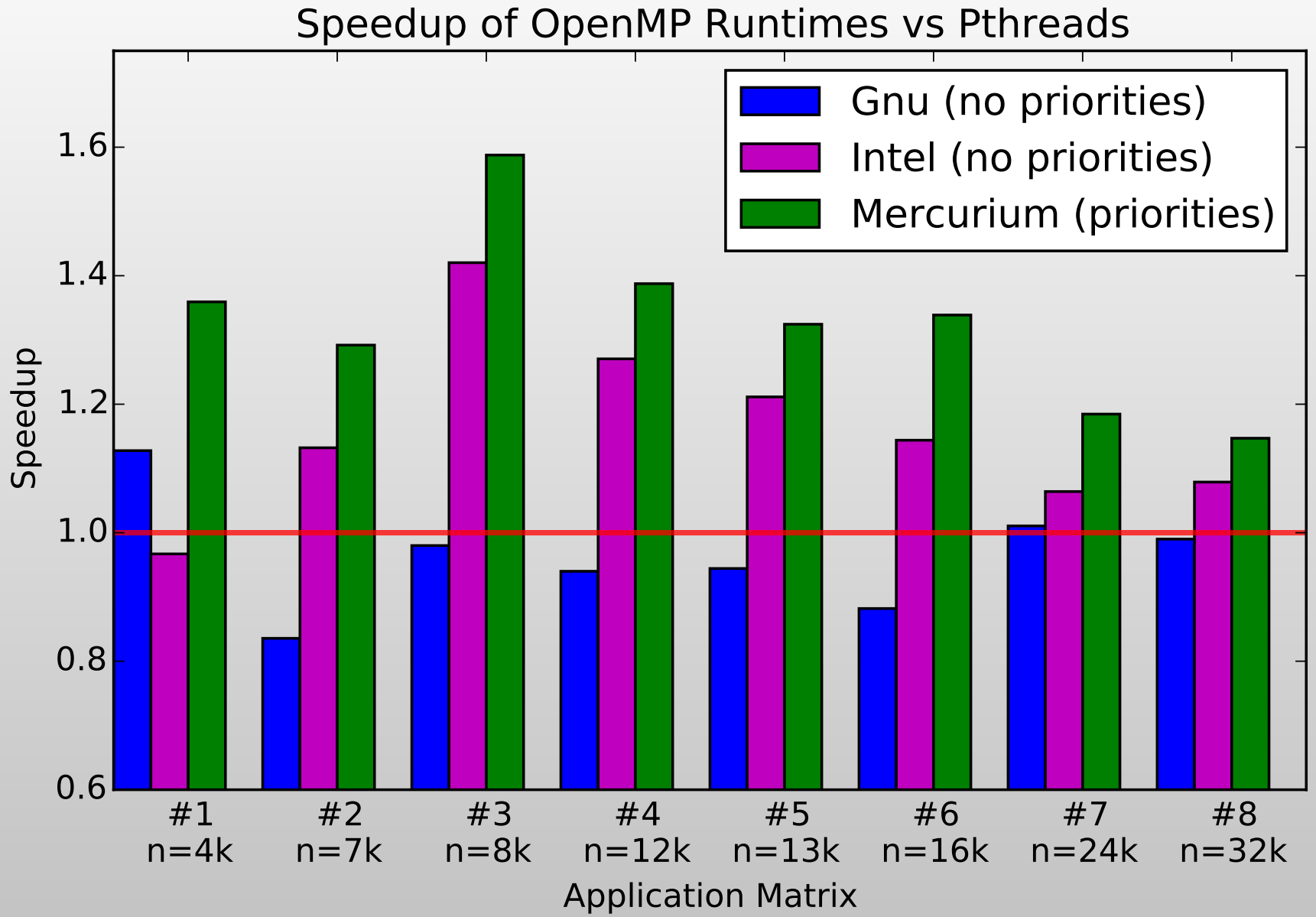
- Previous Work: Promising results even without priorites
- Small porting effort (2 days)
- No use of OpenMP dependencies
- Significant reduction of code complexity

Experiment Setup

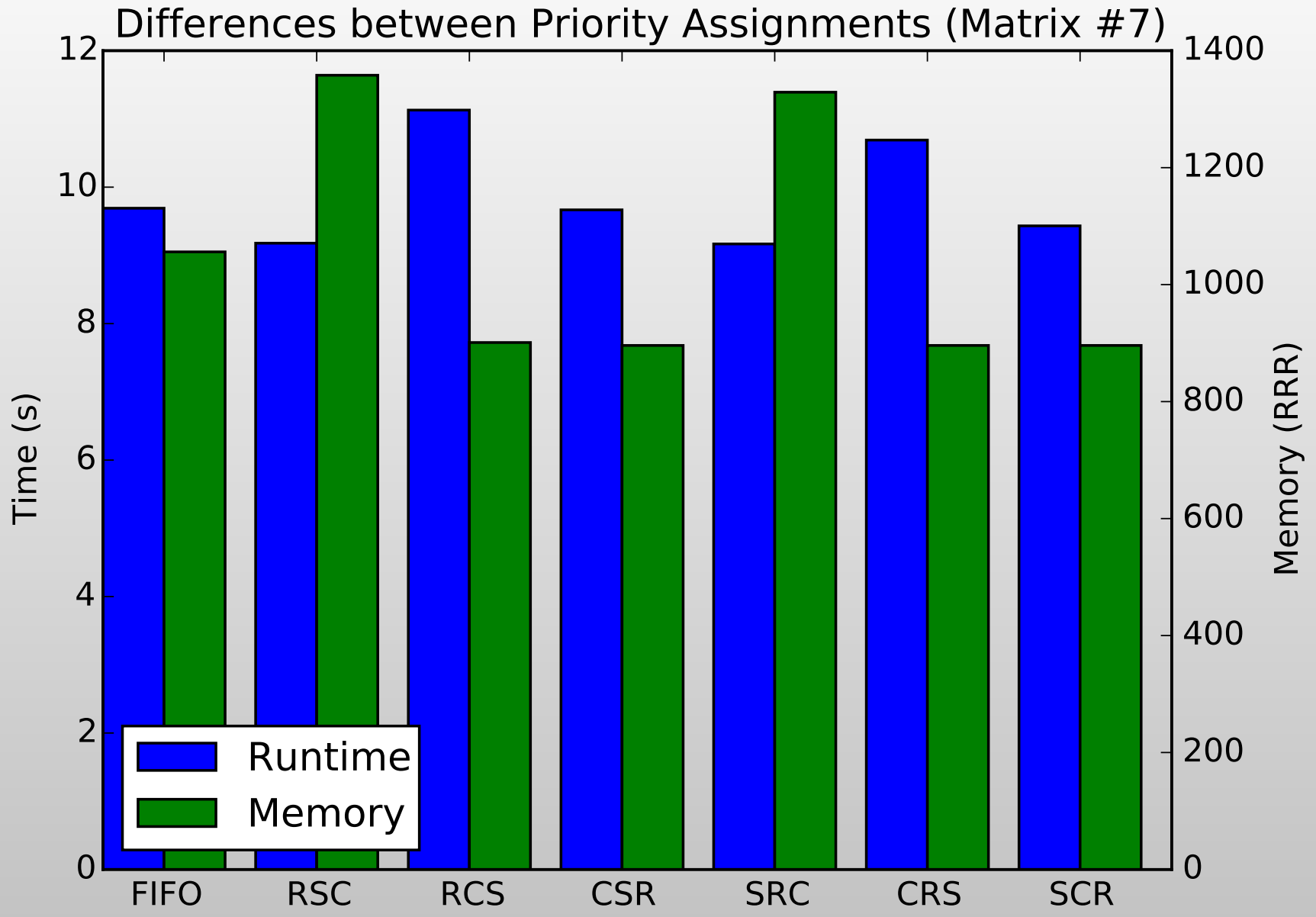
- 48 Threads Intel Haswell
- Intel 16.0.2
- GCC 4.8.5
- Mercurium nightly, BF Scheduler [a]

[a] Version 1.99.9-2016-01-28

Timing Results



Do Priorities actually matter?



Special cases: Easier Coding and Testing

- Easier experimentation through simpler, more flexible code

$\Rightarrow$ Revealed complicated Performance Bug

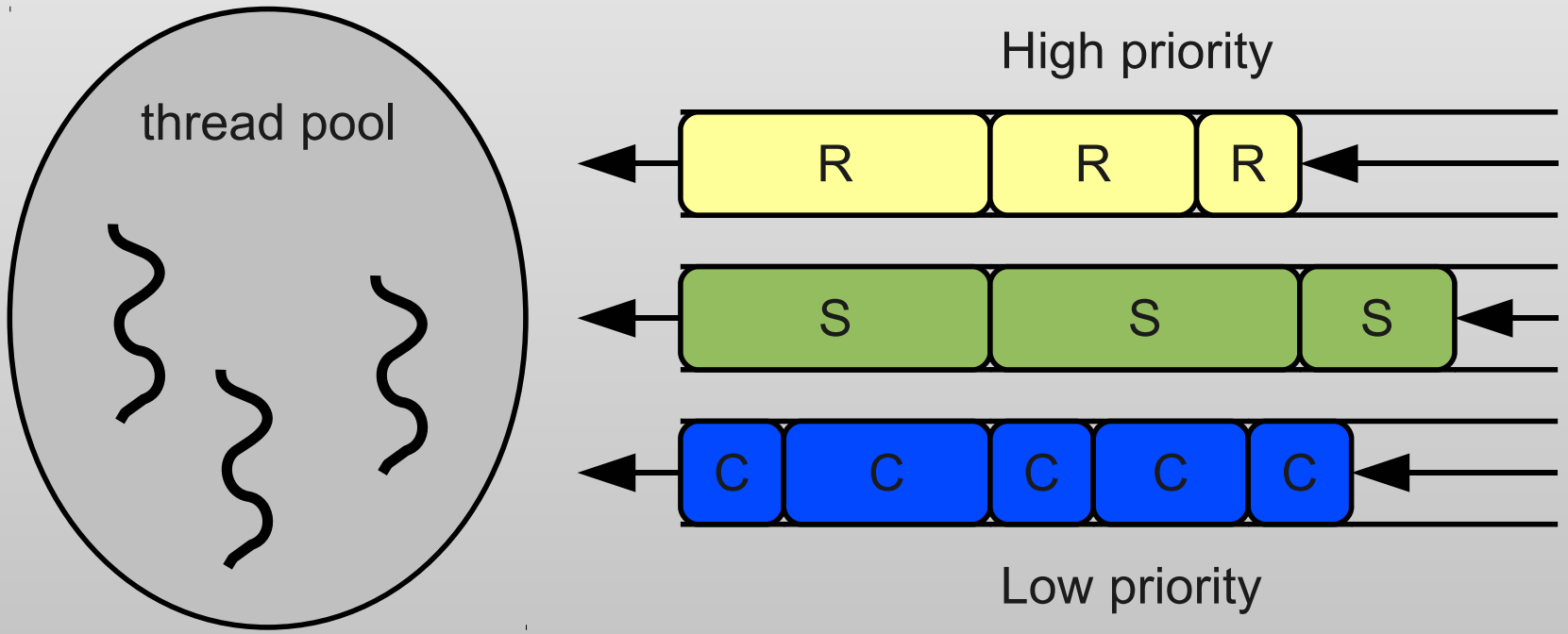
- For some matrices memory footprint larger than expected
- Fix: different priority for small cluster tasks
- OpenMP: 2 lines of code, Pthreads: major code change

Special cases: Easier Coding and Testing

- Easier experimentation through simpler, more flexible code

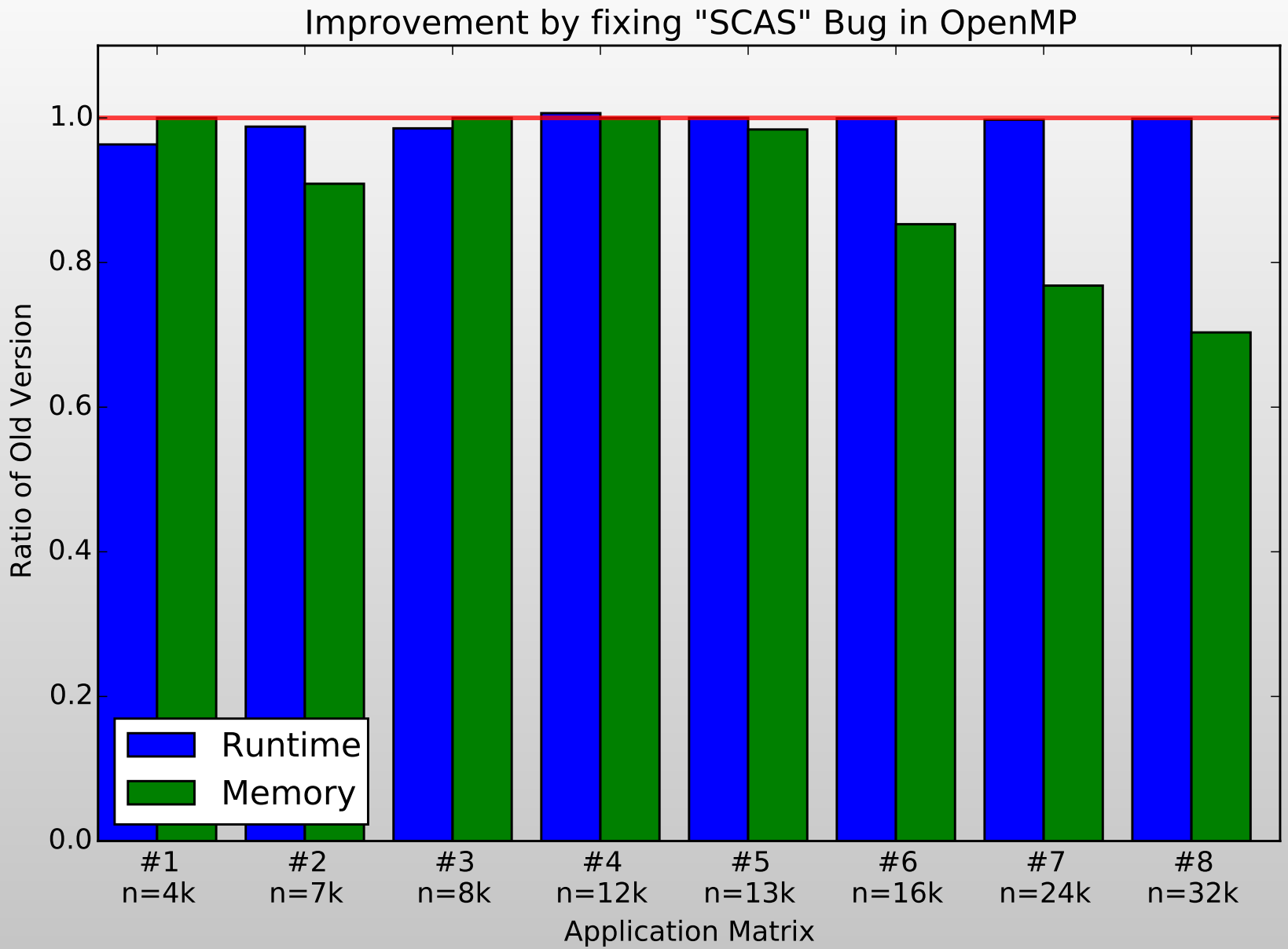
$\Rightarrow$ Revealed complicated Performance Bug

- 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



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

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

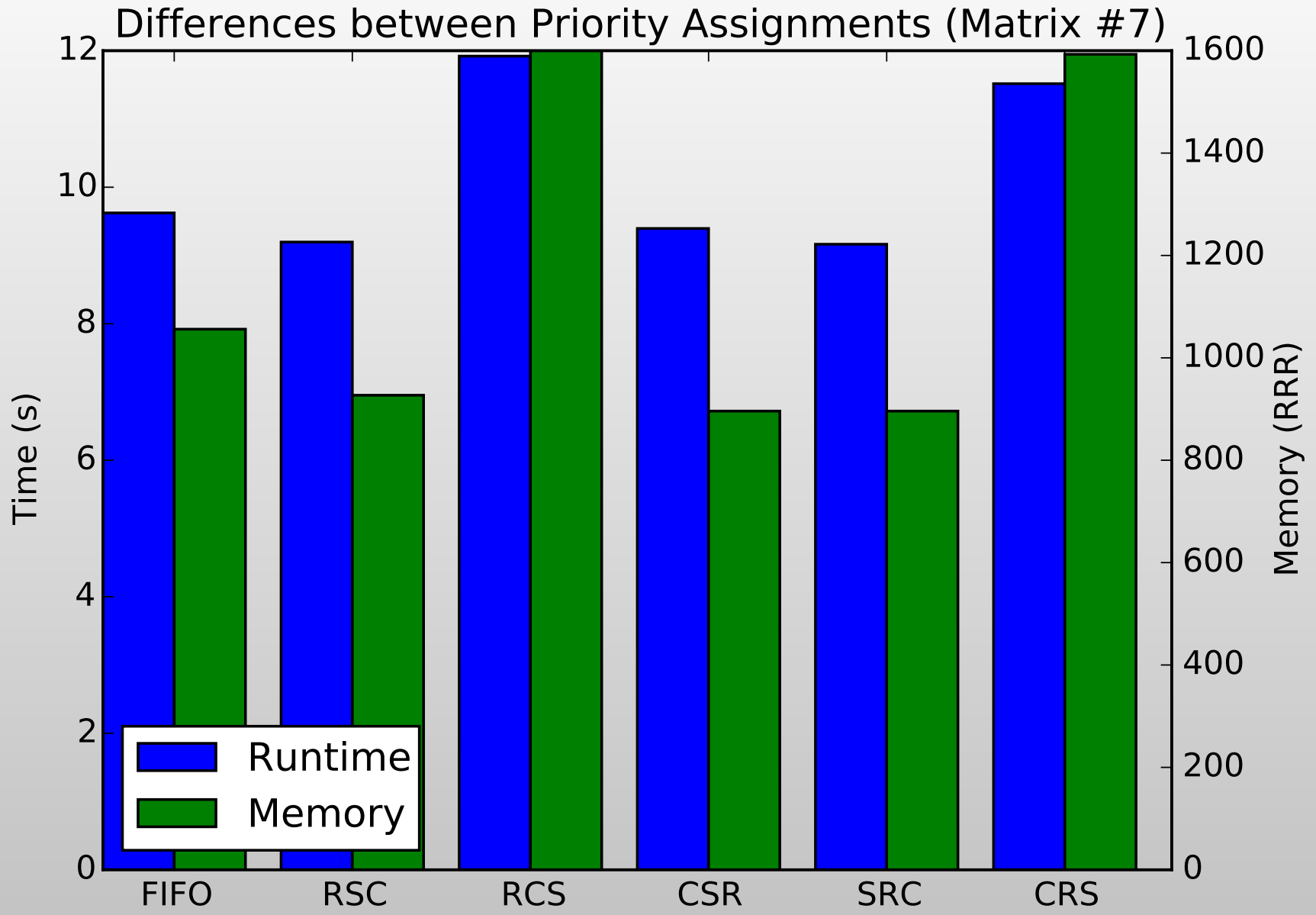
The End

Questions?

Deutsche Forschungsgemeinschaft DFG

Financial support from the Deutsche Forschungsgemeinschaft (German Research Association) through grant GSC 111 is gratefully acknowledged.

Priorities: What we would expect



Relative distances

$$\sigma(T) = \{\lambda_1, ..., \lambda_n\}$$

- Shifting T by τ:

$$\sigma(T - \tau) = \{\lambda_1 - \tau, ..., \lambda_n - \tau\}$$

- Shifting changes the magnitude of eigenvalues
  not absolute distances

$$\mathit{reldist}(\lambda_i, \lambda_{i+1}) = \frac{|\lambda_i - \lambda_{i+1}|}{\max\{|\lambda_i|, |\lambda_{i+1}|\}}$$