Extreme-Scale Eigenvalue Reordering in the Real Schur Form

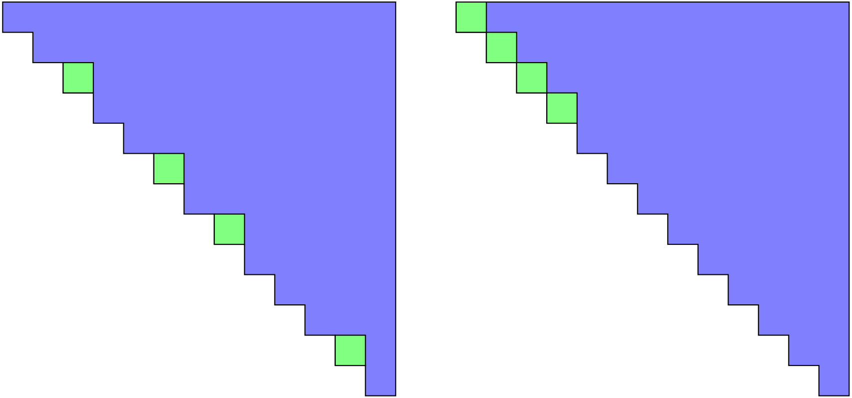
Lars Karlsson, Mirko Myllykoski, Björn Adlerborn,
Carl Christian Kjelgaard Mikkelsen, and Bo Kågström

Department of Computing Science and HPC2N
Umeå University

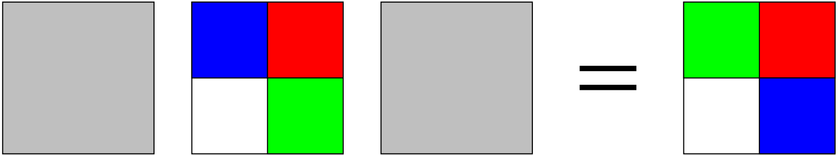
SIAM-CSE-2017

NLAFET
Parallel Numerical Linear Algebra for Future Extreme Scale Systems

The problem



The kernel



- Solve tiny Sylvester equation

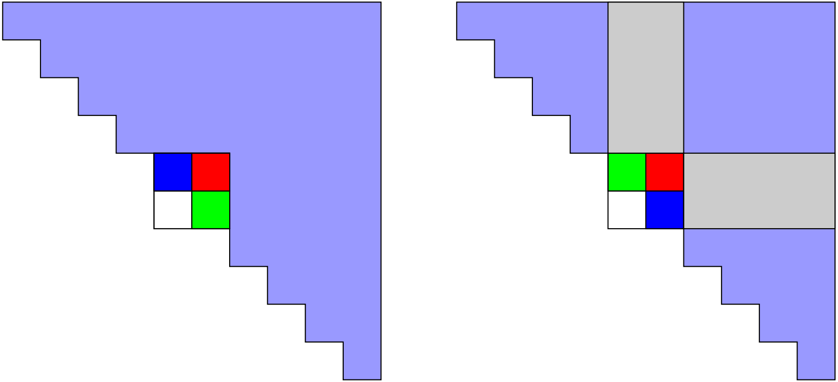
$$AX - XB = \gamma C.$$

- Compute small QR factorization

$$\begin{bmatrix} -X \\ \gamma I \end{bmatrix} = QR$$

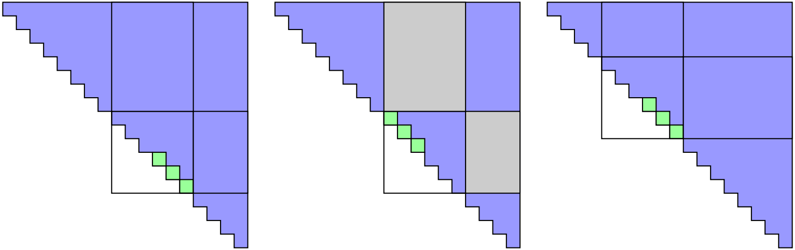
- Apply orthogonal transformation

Scalar code (DTRSEN, LAPACK)



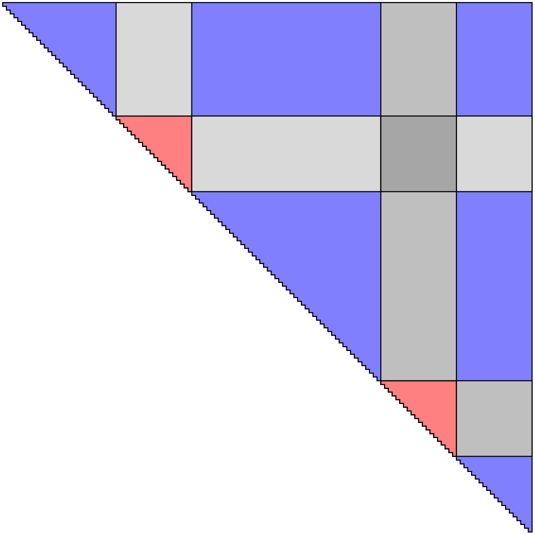
- Slide each eigenvalue up the diagonal as in bubble-sort
- Low arithmetic intensity
- Poor cache utilization

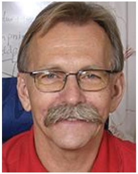
Blocked code (BDTRSEN)



- 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

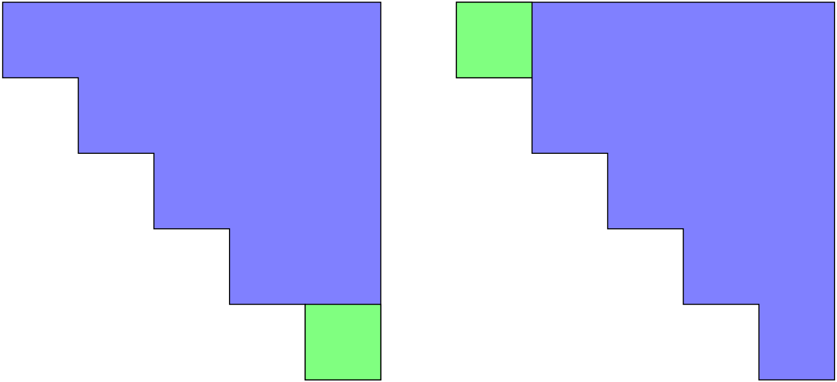
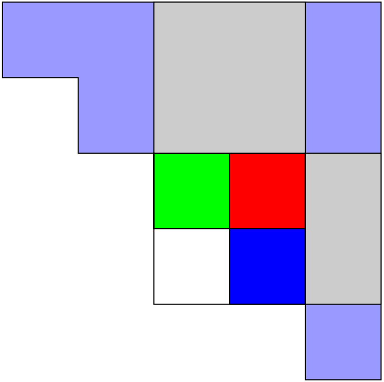


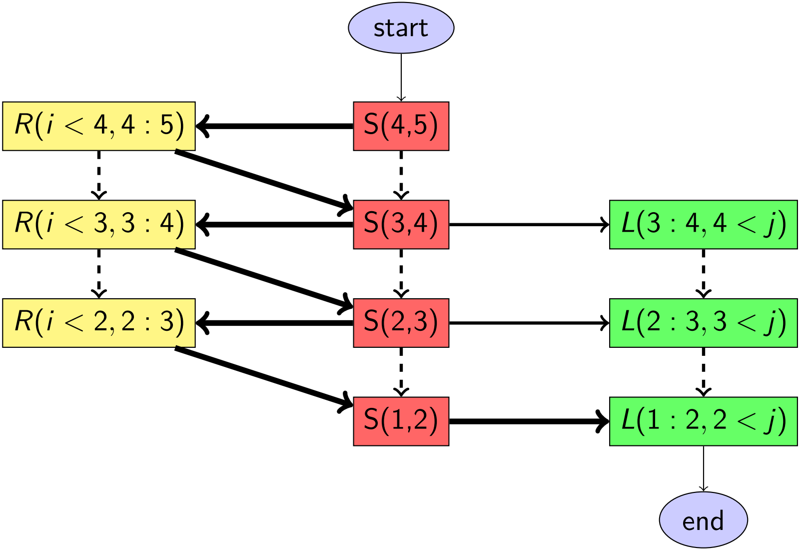
Figure : Move a single block to the top

Table of operations

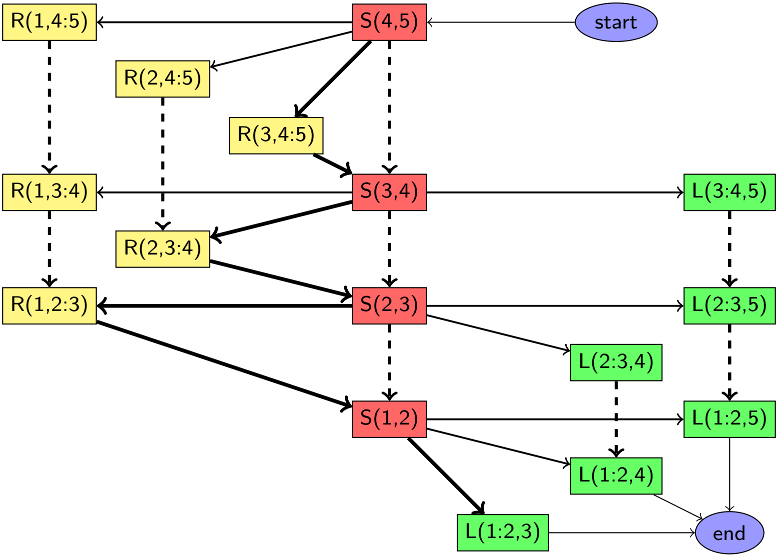


symbol	operation	example
S	Swap adjacent blocks	$S(3,4)$
R	Right update of block column	$R(i<3,3:4)$
L	Left update of block row	$L(3:4,3<j)$

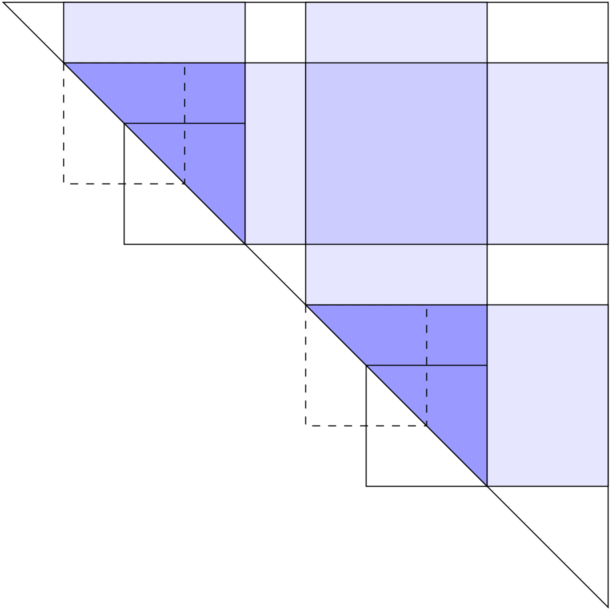
Compressed DAG



Expanded DAG



An advantage of task based programming ...



Software

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 synchonization after row/column updates

Task based code running under StarPU (Mirko Myllykoski)

- Uses multiple windows
- Uses BDTRSEN to process windows
- Low level synchronization among threads

Test problems

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

Machine K* (operational 2016)

- Intel Xeon E5-2690v4
- 2 NUMA nodes per node
- 14 cores per NUMA node
- Each core has its own FPU
- Cores drawn from list 0:1:27

*Standard compute node on Kebnekaise at High Performance Computing Center North (HPC2N)

Accuracy

- For all $\lambda \in \lambda(A)$, we have

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

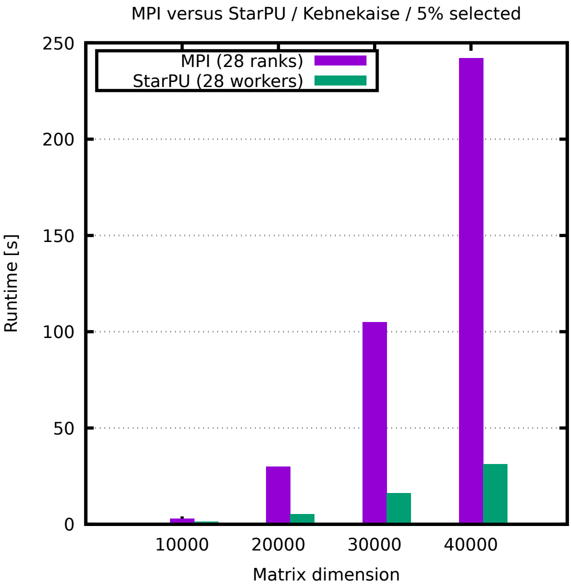
- In addition, we have

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

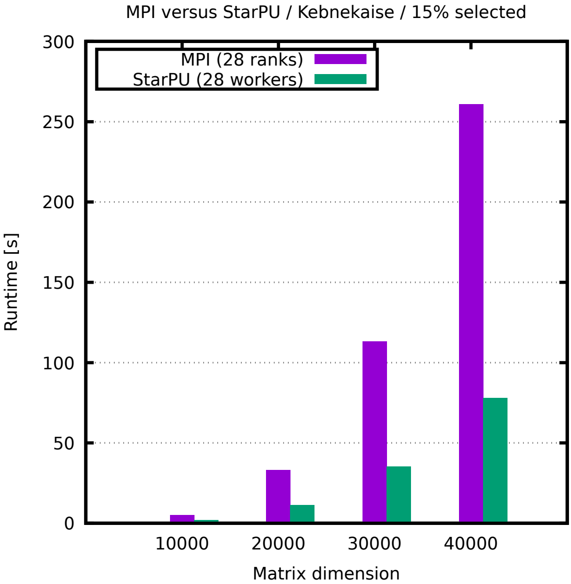
and

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

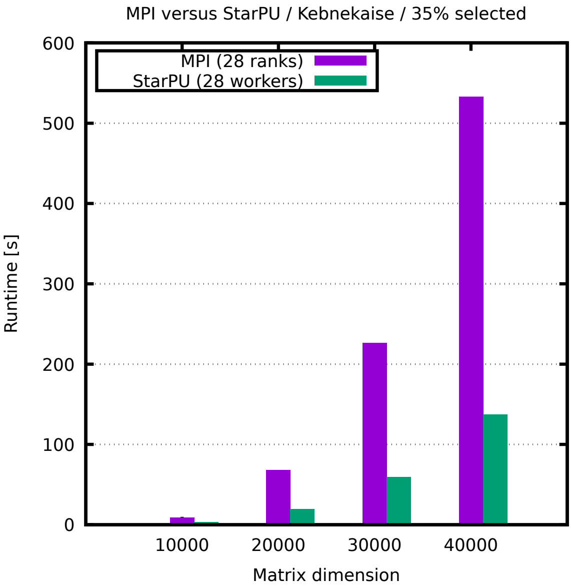
Time to solve



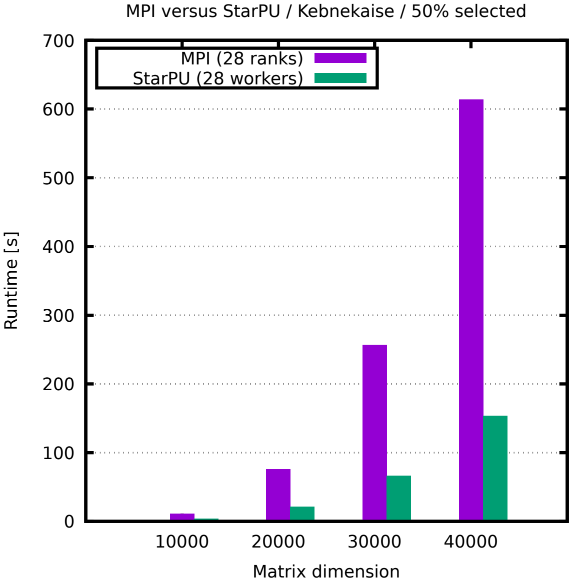
Time to solve



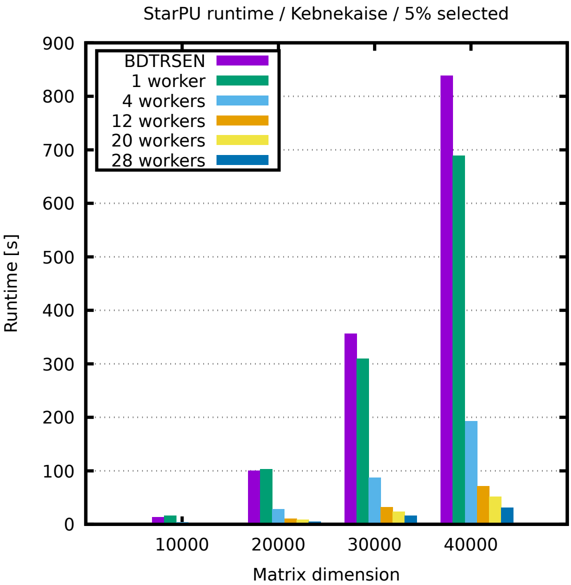
Time to solve



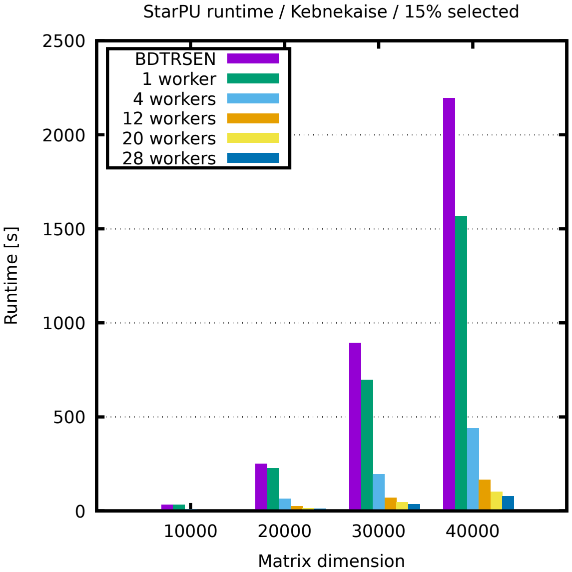
Time to solve



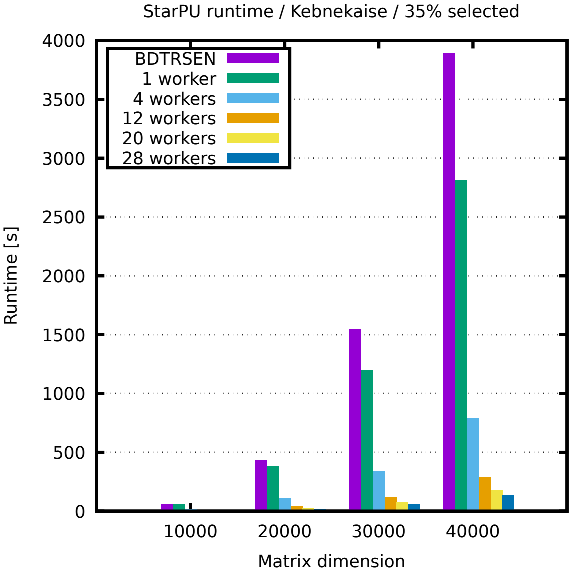
Best serial code?



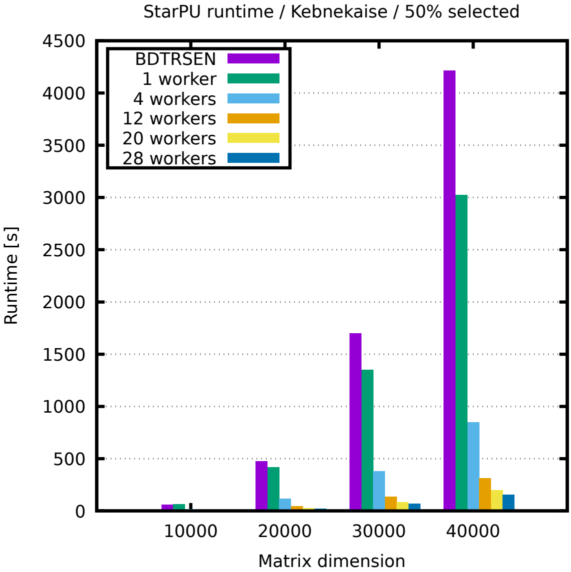
Best serial code?



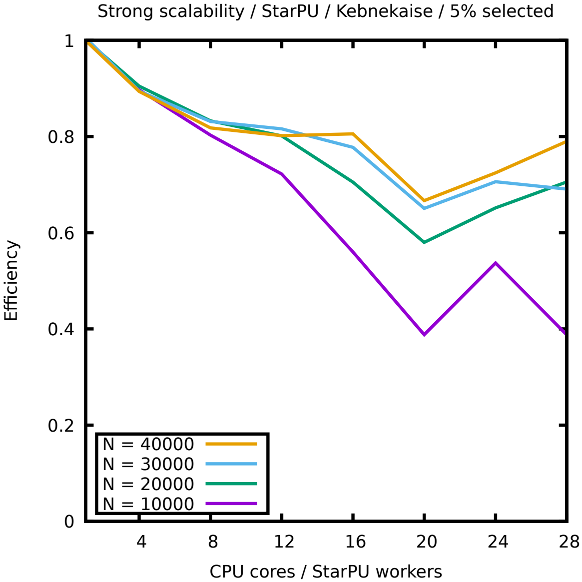
Best serial code?



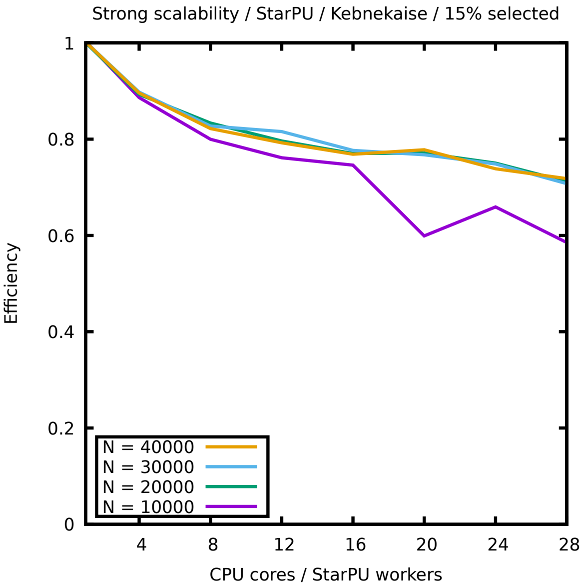
Best serial code?



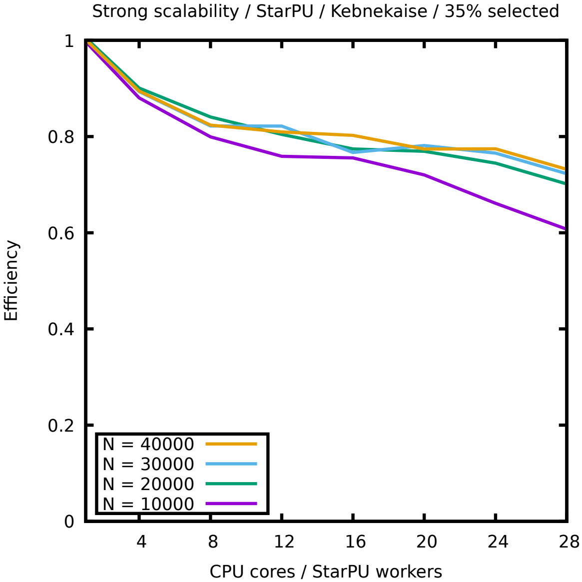
Strong scalability, StarPU



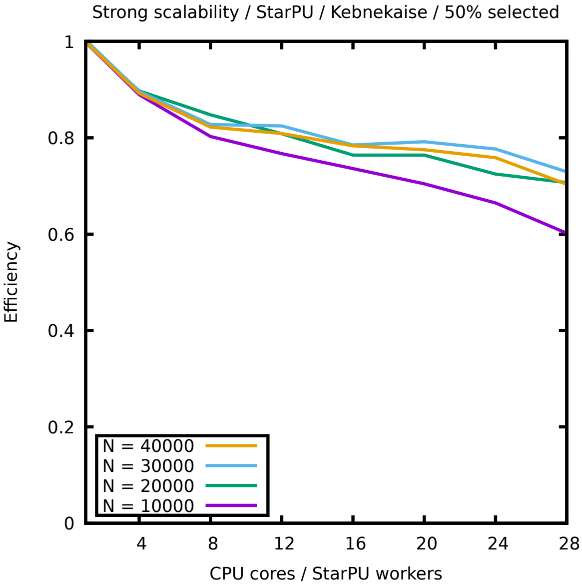
Strong scalability, StarPU



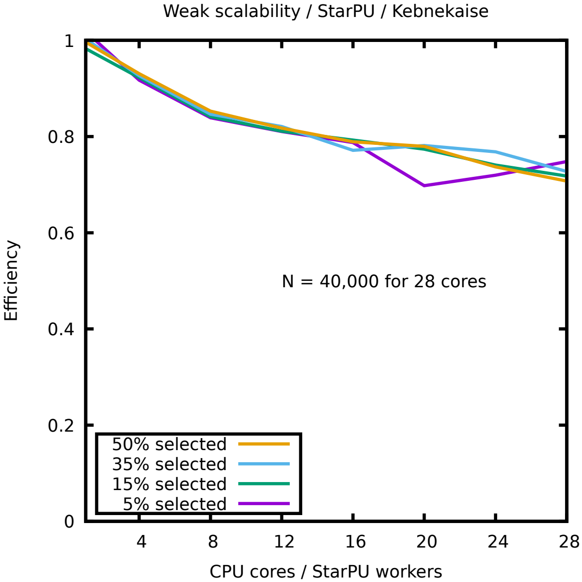
Strong scalability, StarPU



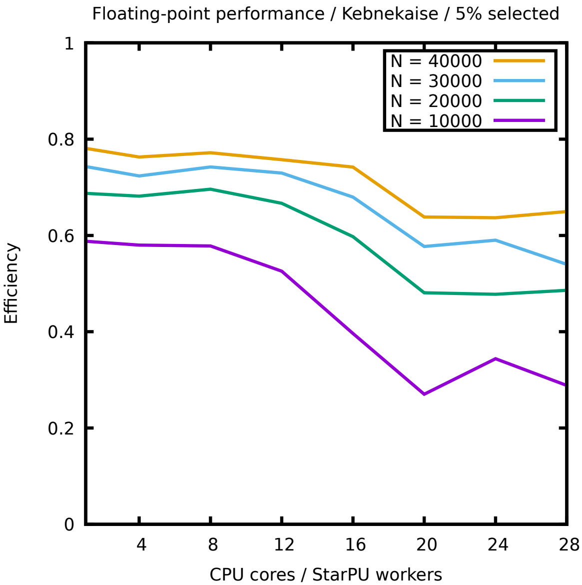
Strong scalability, StarPU



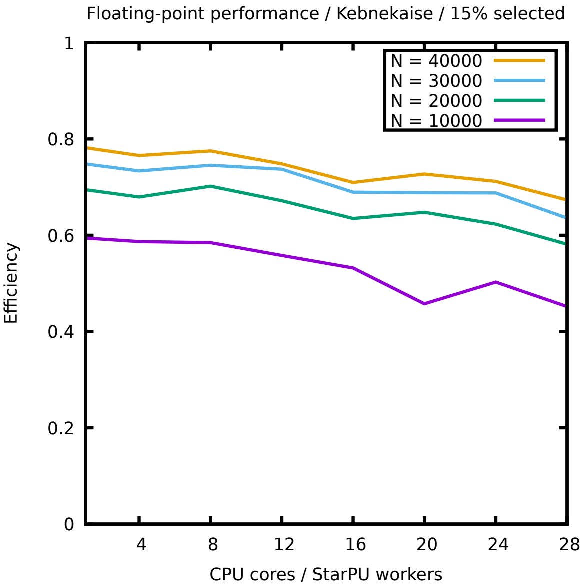
Weak scalability



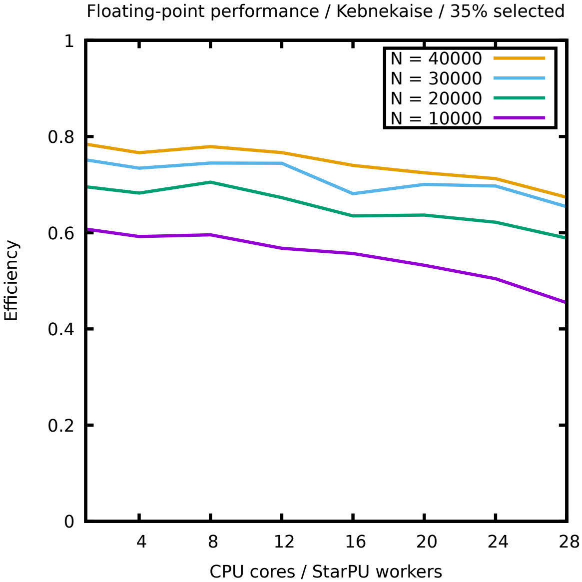
Flop rate


Floating-point performance / Kebnekaise / 5% selected
Efficiency
CPU cores / StarPU workers
N = 40000
N = 30000
N = 20000
N = 10000
0
0.2
0.4
0.6
0.8
1
4
8
12
16
20
24
28

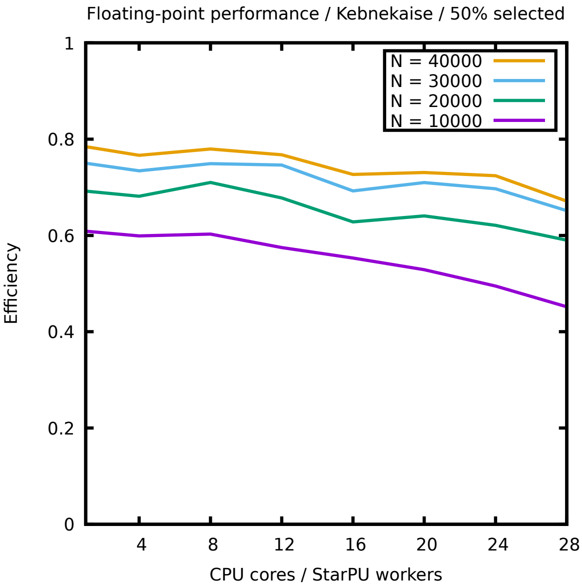
Flop rate


Floating-point performance / Kebnekaise / 15% selected
N = 40000
N = 30000
N = 20000
N = 10000
Efficiency
1
0.8
0.6
0.4
0.2
0
4
8
12
16
20
24
28
CPU cores / StarPU workers

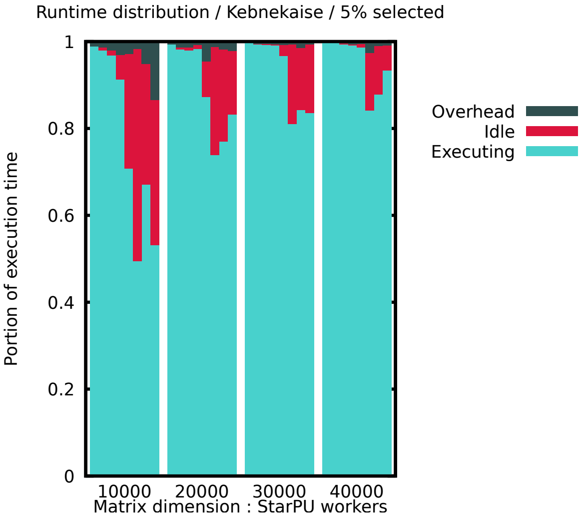
Flop rate


Floating-point performance / Kebnekaise / 35% selected
Efficiency
CPU cores / StarPU workers
0
0.2
0.4
0.6
0.8
1
4
8
12
16
20
24
28
N = 40000
N = 30000
N = 20000
N = 10000

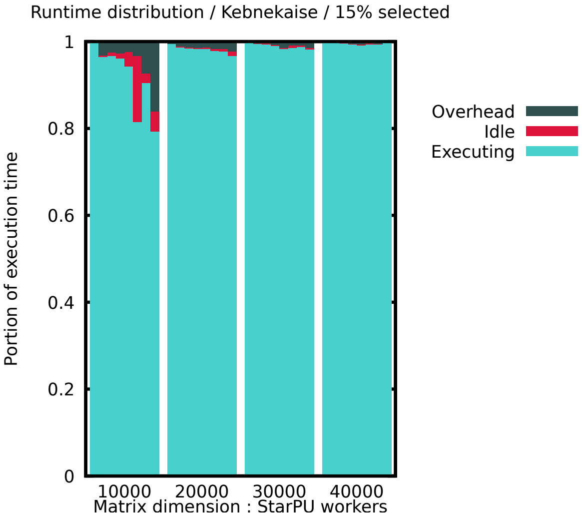
Flop rate


Floating-point performance / Kebnekaise / 50% selected
Efficiency
CPU cores / StarPU workers
N = 40000
N = 30000
N = 20000
N = 10000
0
0.2
0.4
0.6
0.8
1
4
8
12
16
20
24
28

Idle time

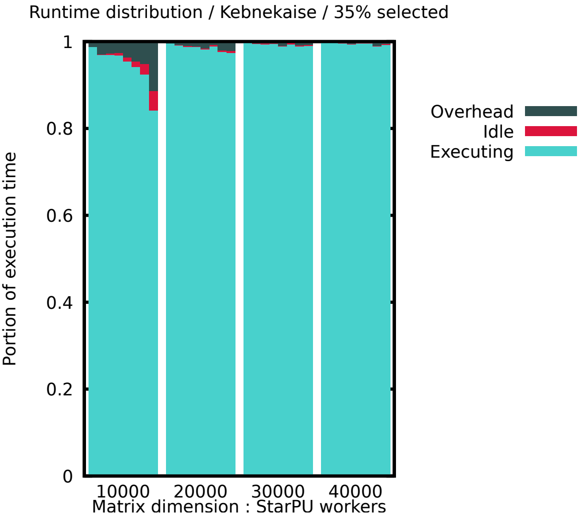

Runtime distribution / Kebnekaise / 5% selected
Portion of execution time
1
0.8
0.6
0.4
0.2
0
10000
20000
30000
40000
Matrix dimension : StarPU workers
Overhead
Idle
Executing

Idle time

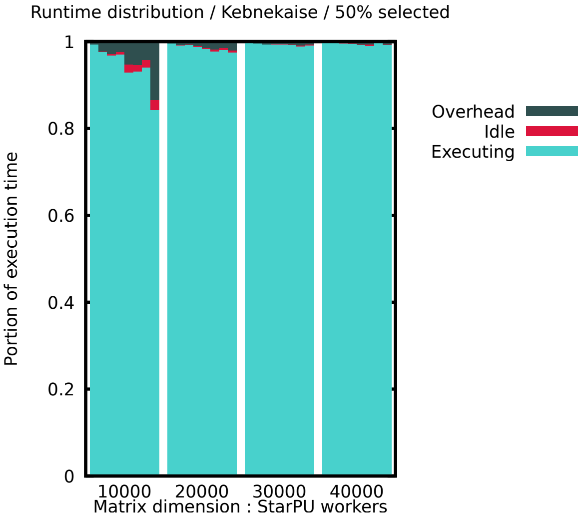


Runtime distribution / Kebnekaise / 15% selected
1
0.8
0.6
0.4
0.2
0
Portion of execution time
10000
20000
30000
40000
Matrix dimension : StarPU workers
Overhead
Idle
Executing

Idle time


Runtime distribution / Kebnekaise / 35% selected
Portion of execution time
1
0.8
0.6
0.4
0.2
0
10000
20000
30000
40000
Matrix dimension : StarPU workers
Overhead
Idle
Executing

Idle time

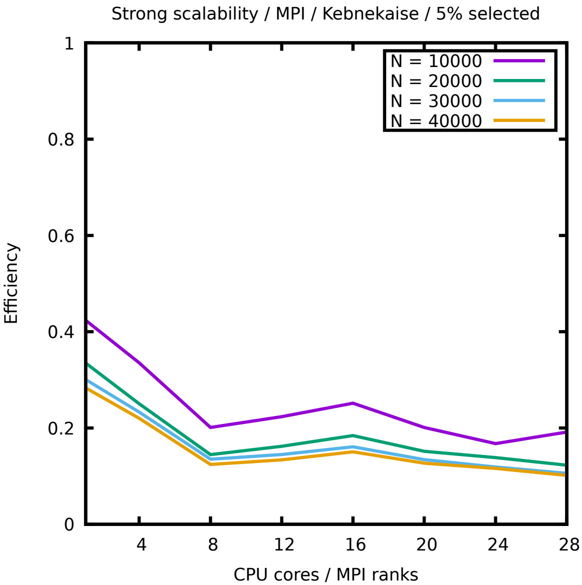

Runtime distribution / Kebnekaise / 50% selected
Portion of execution time
1
0.8
0.6
0.4
0.2
0
10000
20000
30000
40000
Matrix dimension : StarPU workers
Overhead
Idle
Executing

The end is near!

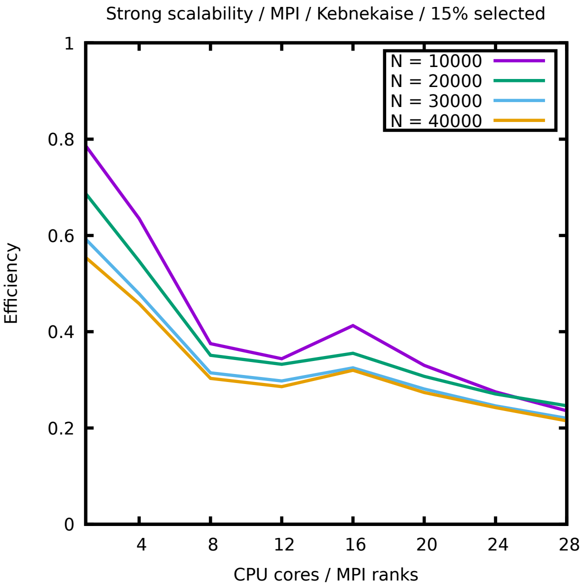
Thank you for your attention

Additional material and figures

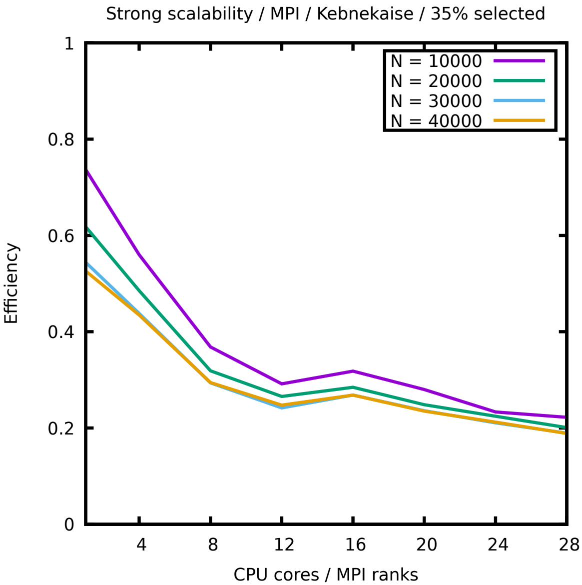
Strong scalability, MPI



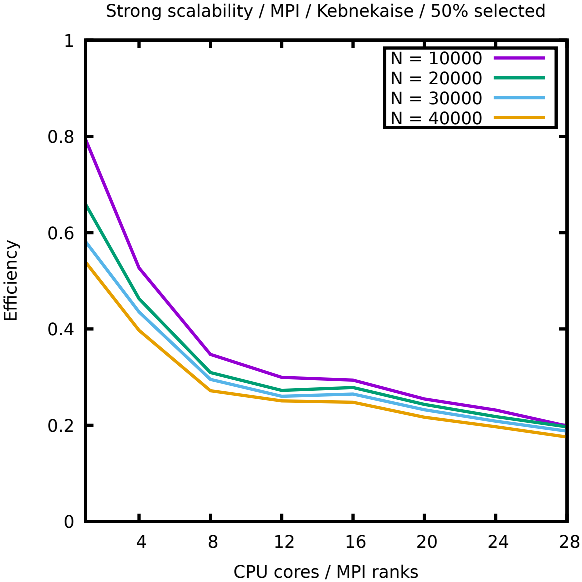
Strong scalability, MPI



Strong scalability, MPI



Strong scalability, MPI



Additional figures

Multiwindow DAG