Smarter technology for all

Current status of Hybrid Density Functionals and ELPA

Florian Merz | 02.08.2023

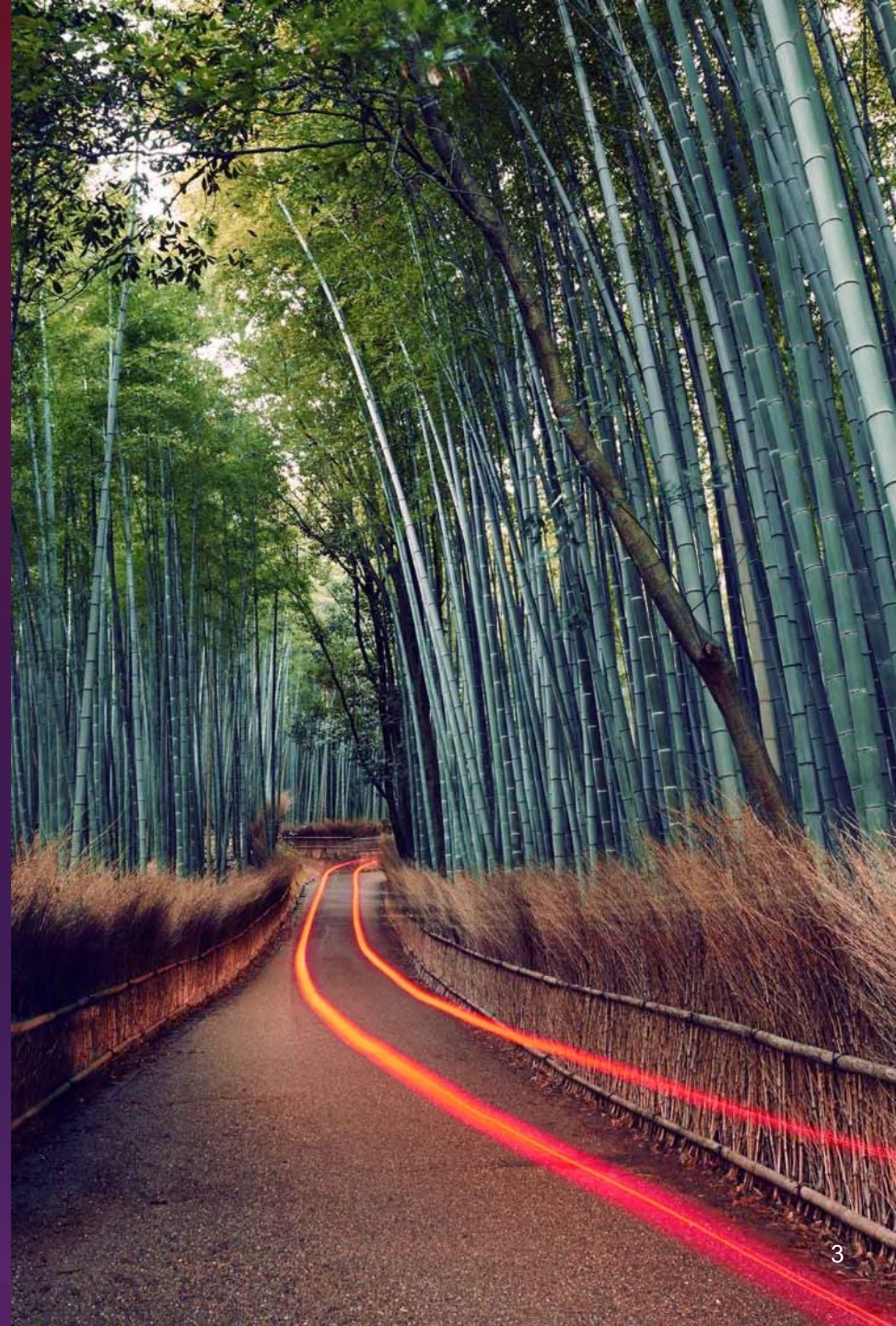
Disclaimer + Contributors

- I’m not an expert in materials sciences / DFT, my focus is on HPC application performance

- This work is part of a long-standing collaboration between FHI and MPCDF (Markus Rampp, Andreas Marek) / Lenovo to boost FHI-aims performance and scalability
- On the FHI-aims side, the work has been supported by **Sebastian Kokott**, Volker Blum, Christian Carbogno, Mariana Rossi, Ville Havu, Min-Ye Zhang, Xinguo Ren, Uthpala Herath, Matthias Scheffler
- ELPA GPU work coordinated/done by MPCDF (**Andreas Marek** et al.)

Hybrid functionals update

Includes results by Sebastian Kokott



Exact exchange calculations with hybrid-density functionals

Significant overhaul of the 2015 version

- MPI3 shared memory to reduce memory consumption
- Exploit more sparsity – memory compression for RI coefficients
- Loop level optimization, improved memory accesses and cache usage
- Optimized communication patterns (one-sided MPI instead of mpi_allreduce)
- Blocking of exchange matrix computation to reduce load imbalance
- Significantly improved scalability due to additional parallelization layer

Other areas that were tuned:

- Hartree multipole summation for periodic case (factor 5-30)
- Density computation with forces (factor 5)
- Fourier transforms in band output routines (Sebastian)

Old parallelization scheme

- Parallelization scheme is fixed in the beginning, optimized for actual Fock matrix computation
  - Not optimal for all other routines

Coulomb matrix initialization

RI coefficient initialization

do DFT loop

Density computation

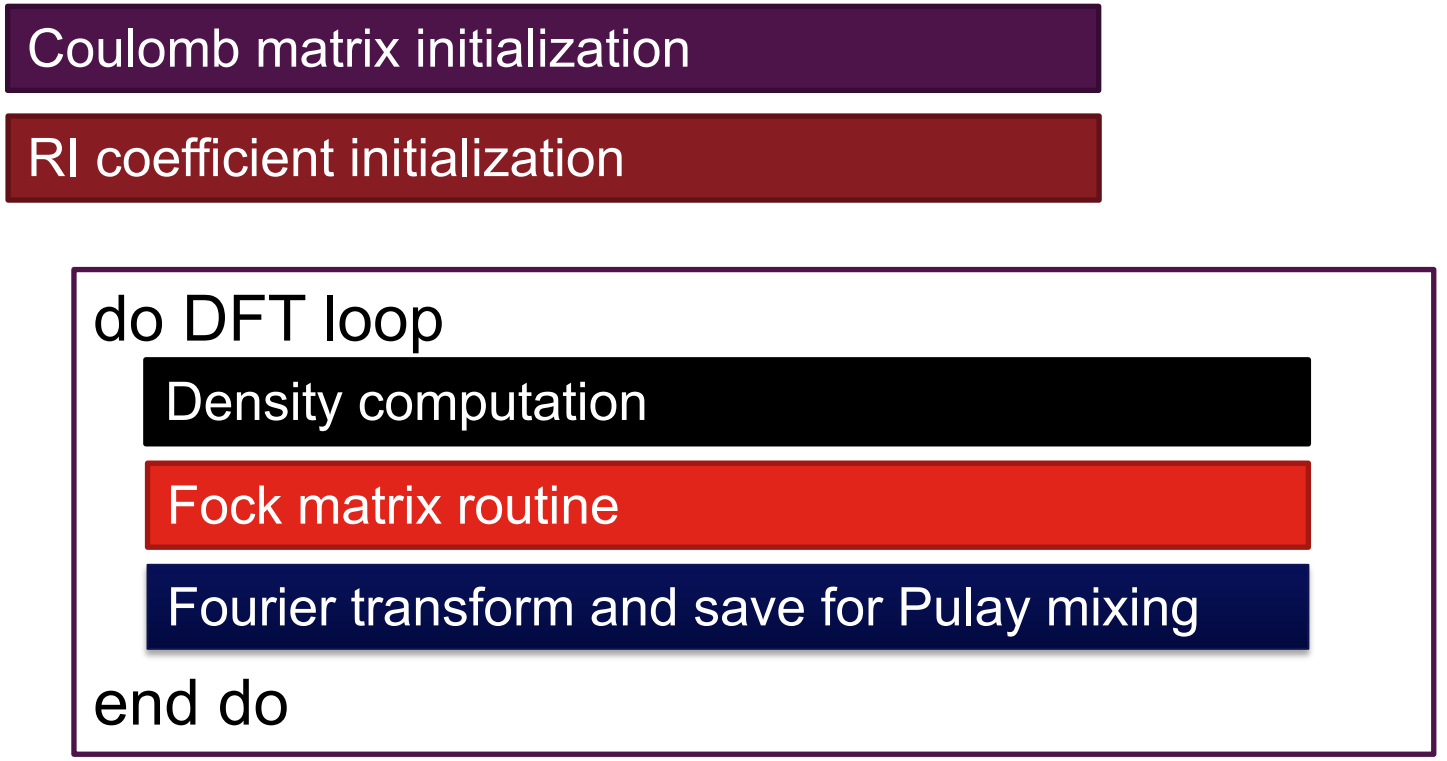
Fock matrix routine

Fourier transform and save for Pulay mixing

end do

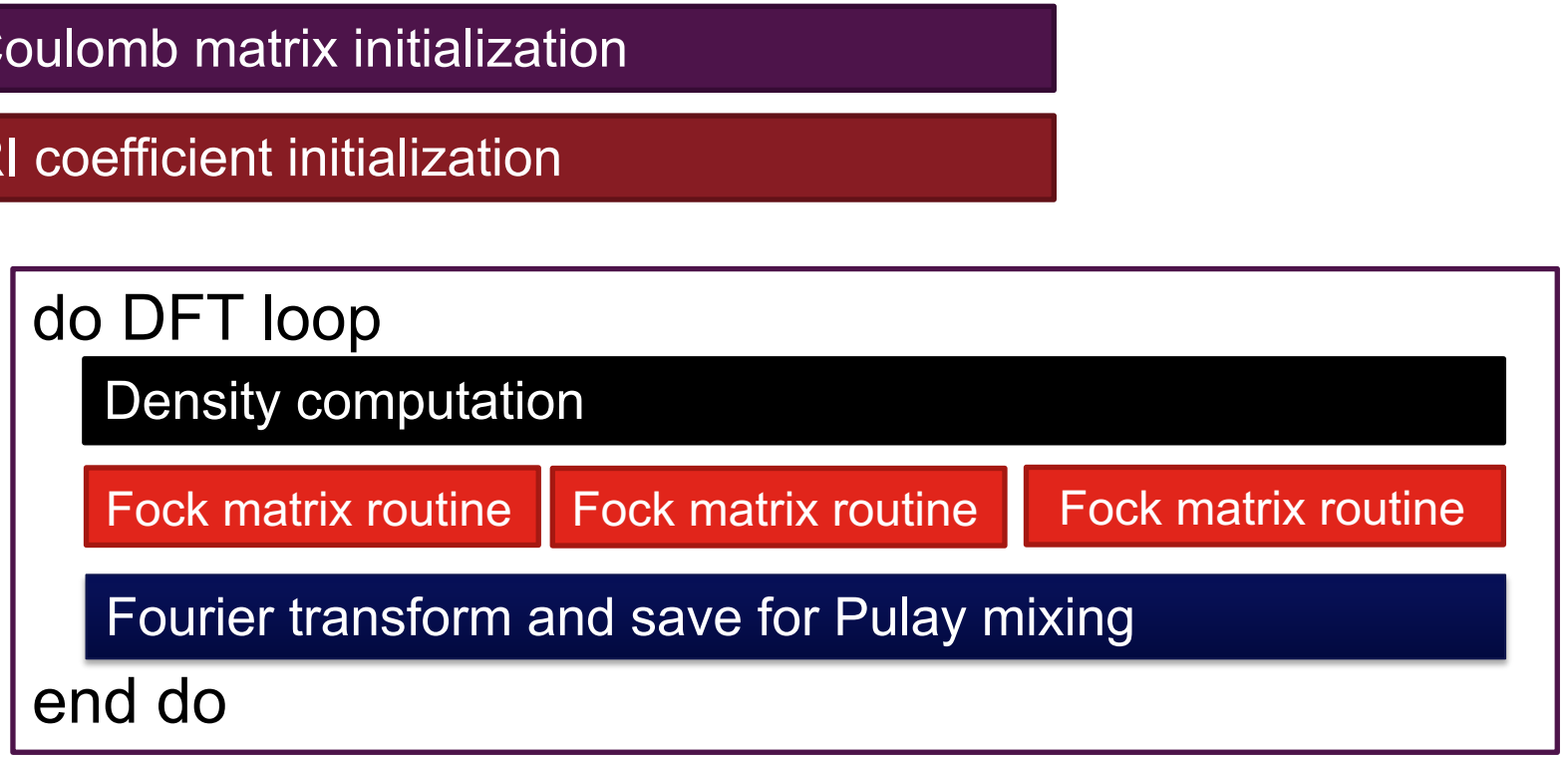
New parallelization scheme, part 1

- Optimized parallelization / data layout for all parts!
  - Requires redistribution of data between steps



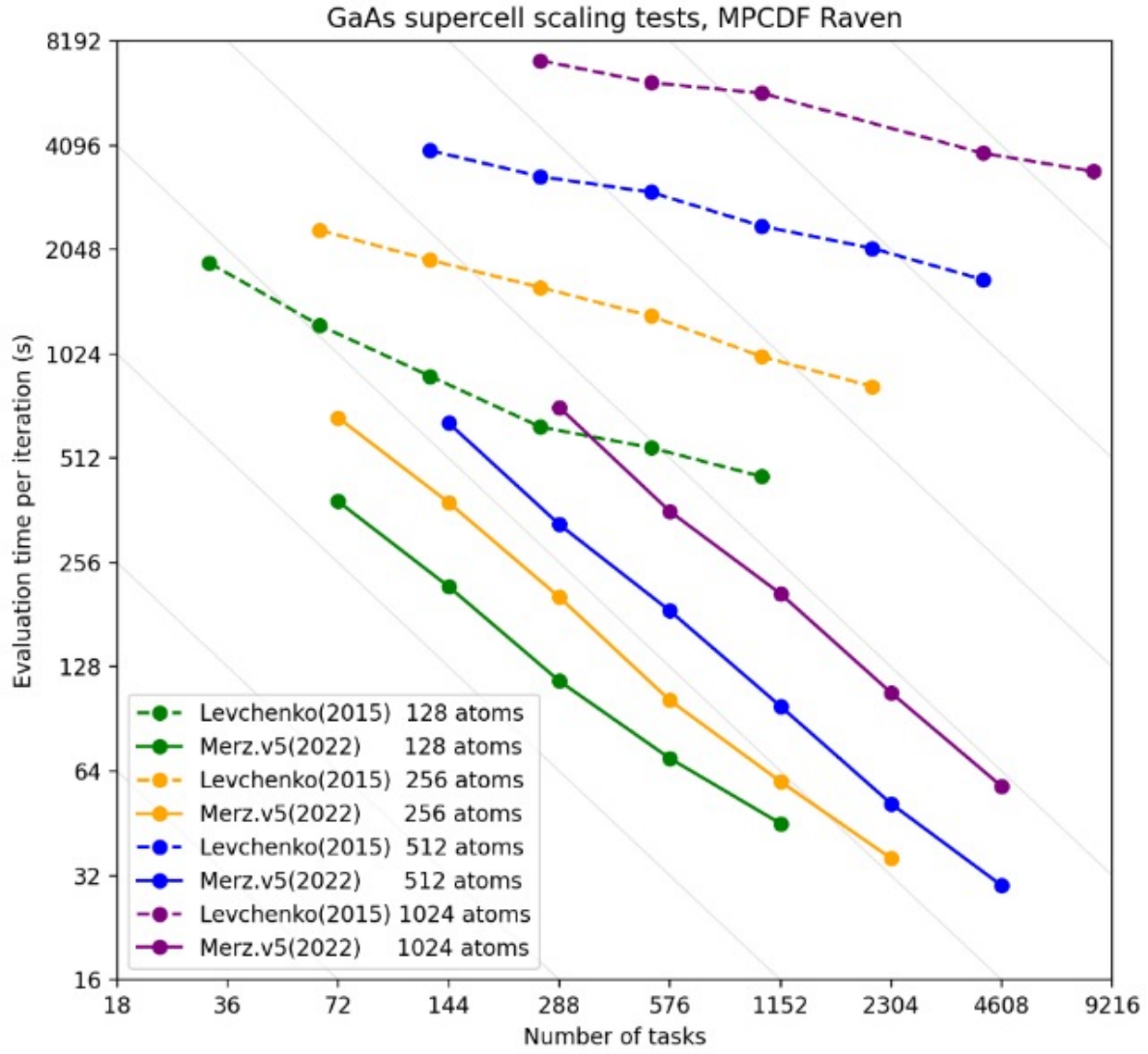
New parallelization scheme, part 2

- Initialize multiple instances of the Fock matrix routine on separate subcommunicators when enough memory is available
  - Additional parallelization layer - compute different blocks (column index chunks) of the FM on different instances



Improved strong scaling

- With increasing node count, the additional memory is used to increase the number of instances
- Number of nodes per instance stays the same, i.e. the communication intensive parallelization atom pairs and cells does not increase
- Almost ideal strong scaling for a wide range of nodes



Larger systems

- We optimized memory consumption and runtimes for large number of atoms
  - Exact exchange computations (initialization and computation)
  - Bottlenecks in other parts of initialization removed with shared memory arrays (Sebastian!)

- FHI-aims can now run hybrid computations with > 10.000 atoms
- Successfully simulated ice (H2O) with 30K atoms and light settings

New infrastructure

- Intra/inter node communicators are available throughout the FHI-aims run (e.g. to aggregate memory estimates of unevenly distributed arrays in **`get_max_mem_node`** routine)

- Routines to allocate/deallocate shared memory arrays (see mpi_shm.f90)

- **`call show_free(info_str)`** routine to print minimum free memory per node at a certain stage of the computation
  - Queries the OS of the node
  - Now used after many initialization routines
  - (has to be called by all MPI tasks)

Advertisement: MPI3 Shared memory

- The MPI 3 standard introduced shared memory support for tasks within a node
  - All tasks in a node can directly read and write to the same array without communication! Synchronization calls necessary to make sure the data is there when needed.
- Number of cores per node is still increasing!
- Scenarios where shared memory may help:
  - Arrays that are identical on all MPI tasks – one copy per node instead of one copy per task reduces memory by factor 2 orders of magnitude on current CPUs
  - Global broadcasts will be much faster when only 1 array per node is filled

- Approach myself or Sebastian or Volker if you would like to discuss / need guidance if this approach makes sense for your project / how to extend the mpi_shm to make it more useful

- Good introduction: https://software.intel.com/en-us/articles/an-introduction-to-mpi-3-shared-memory-programming

User Flags

- In general, FHI-aims tries to choose optimal parameters based on available memory, available nodes, problem size – printed to output
- Automatic selection is not perfect, there are knobs to tune it in case it‘s needed

- In case of out-of-memory crashes it is possible to
  - Reduce blocksize with **`fock_matrix_blocking`** keyword in control.in
  - Reduce number of instances with **`fock_matrix_nodes_per_instance`**

- To get the last bit of performance for a series of runs (e.g. for MD) it is possible to
  - Try to increase number of instances (may lead to out-of-memory)

Outlook

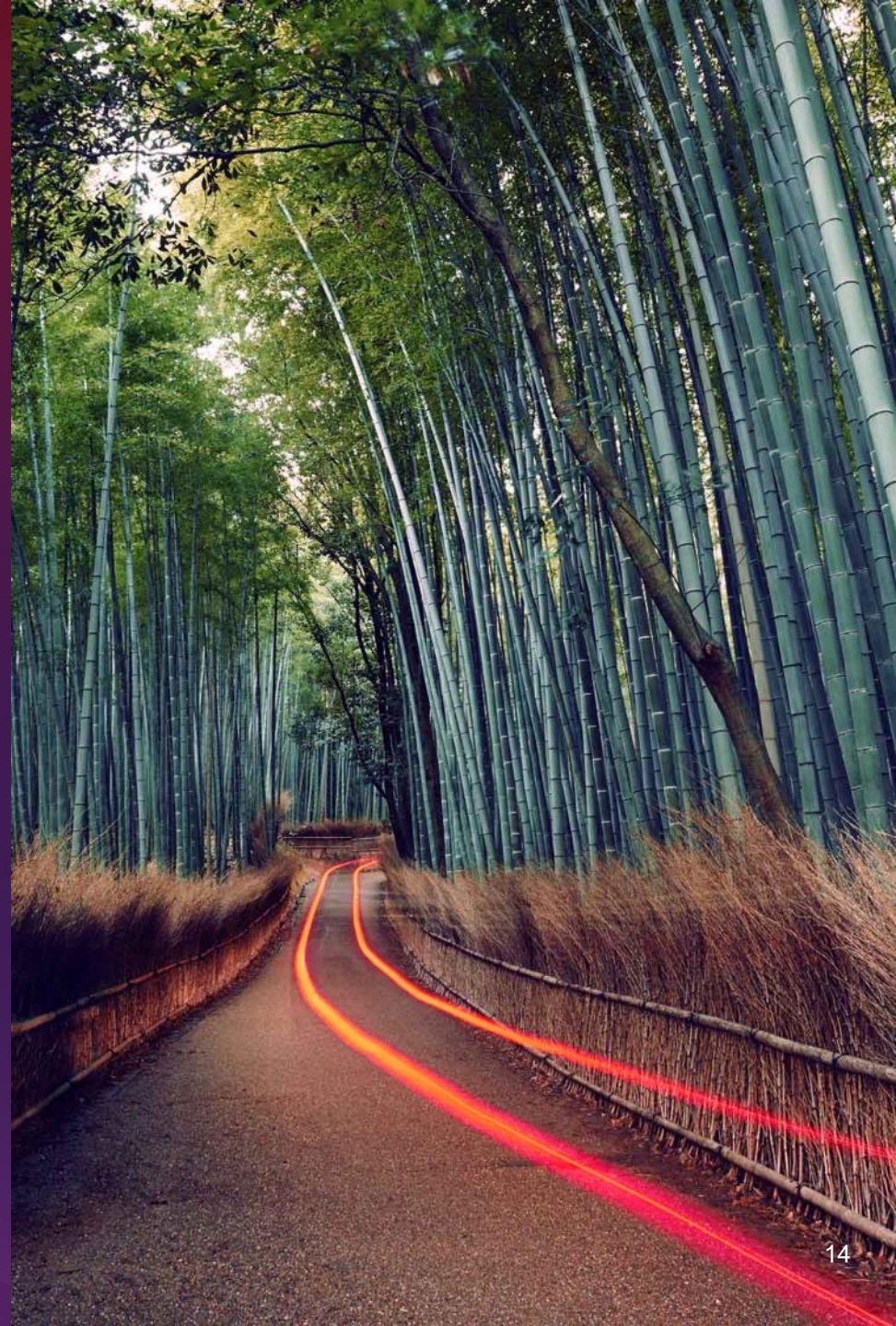
- Paper about improvements almost complete (Sebastian)

- Still working on minor improvements for memory estimates / automatic parameter selection

- Started to look into GW
  - Reusing / transferring the tunings for RI coefficient initialization

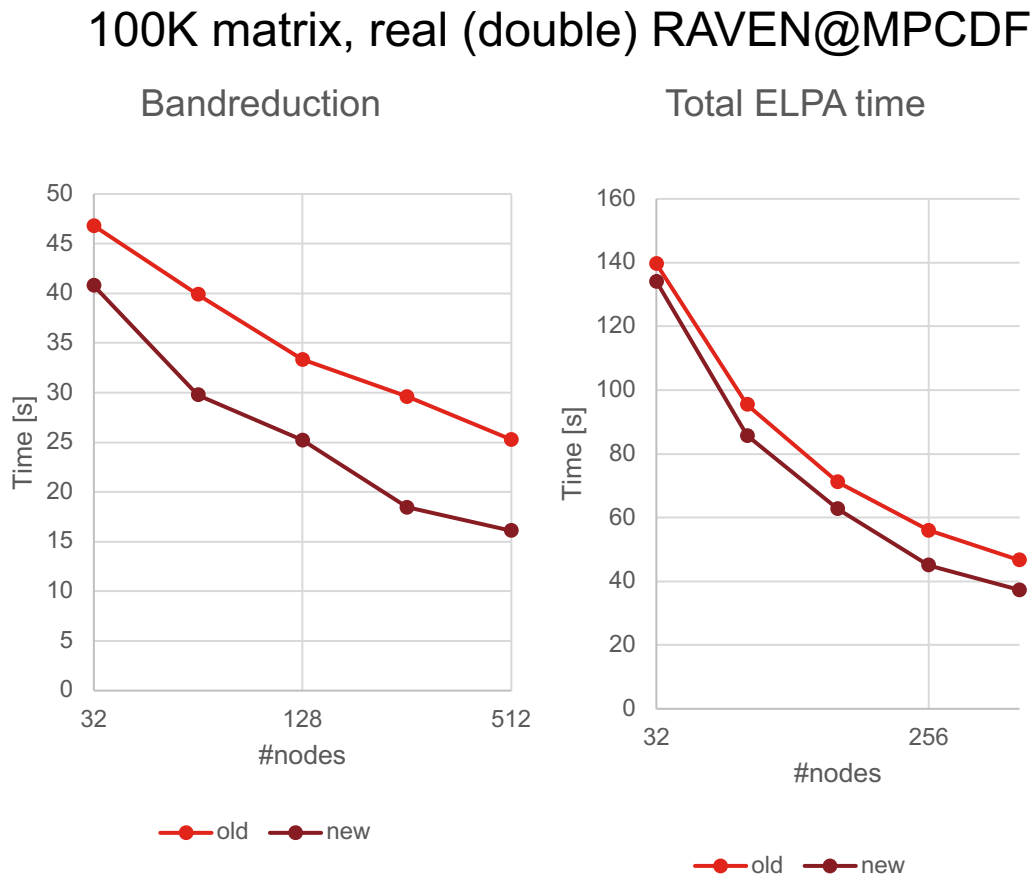
ELPA update

Includes results by A. Marek



ELPA 2stage improvements

- FHI-aims uses ELPA to solve the generalized eigenvalue problem – often large fraction of computation time
- Eigenvalue problem often not very big compared to number of cores – scaling limit
- The bottleneck for ELPA 2stage solver in the scaling limit is the band reduction. In 11/2022 version, we
  - Introduced blocking
  - Optimized communication patterns
  - Obtained up to 45% speedup on band reduction
  - Up to 25% speedup for total time
- Old implementation still available, ELPA will automatically select the best implementation

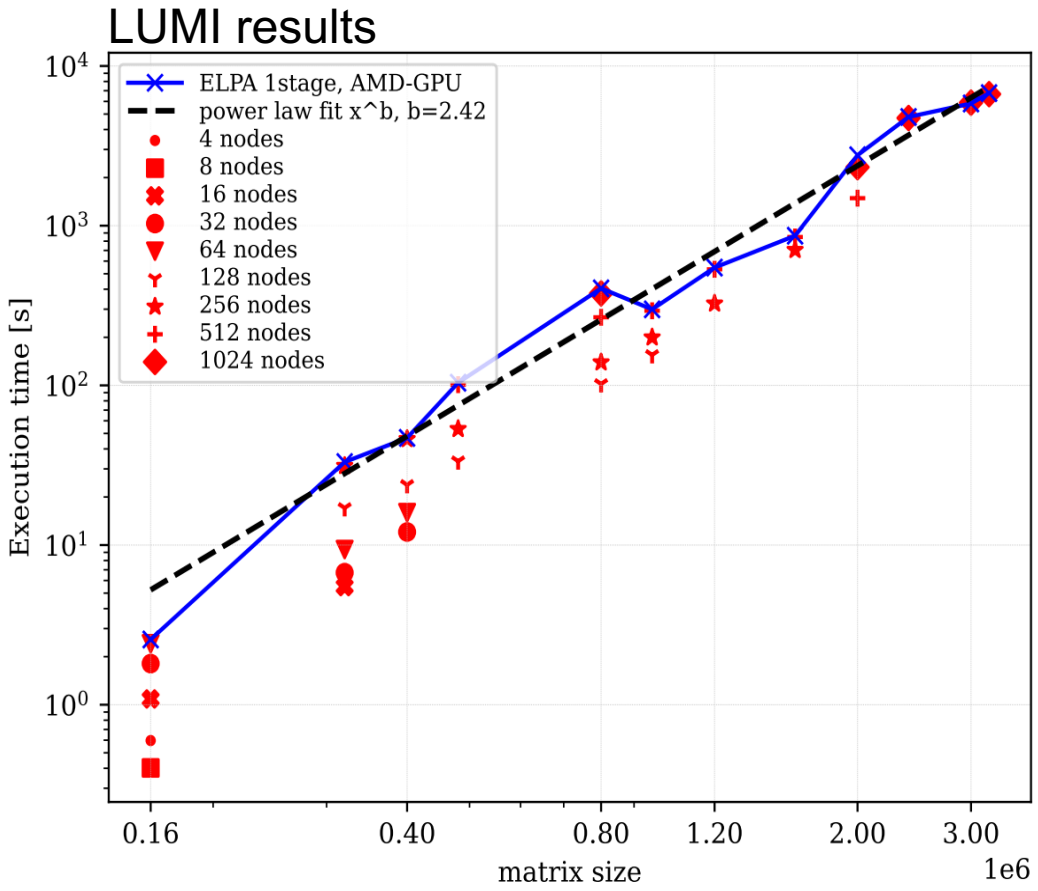


ELPA GPU

- ELPA on NVIDIA GPUs

- ELPA now offers the full support on AMD GPUs
  - Successful demonstration on LUMI (CSCS), Finland
  - No MPS equivalent so far, number of tasks per GPU may have to be limited

- ELPA now offers the full support on Intel GPUs
  - Demonstration runs on „Ponte Vecchio“ starting soon
  - At the moment: for experienced users only



ELPA development plan

- Further improving GPU performance
  - Extended scaling limit
  - Mitigating memory transfer bottlenecks

- GPU to GPU communication
  - Implementation of communication with NCCL and alike

- Iterative refinement of results
  - Compute in lower, refine to higher precision

- Preparation for upcoming architectures
  - Nvidia H100, AMD Mi300
  - First porting to RISCV (EPI)