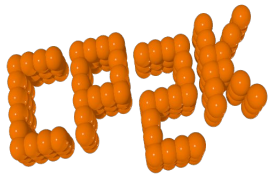
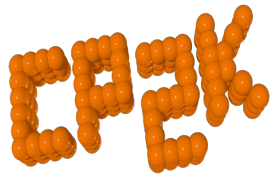
CP2K Developers Meeting

February 6th, 14:00-16:00 2023

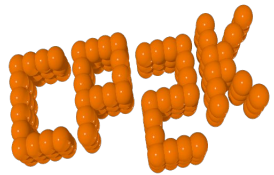


CP2K Developers Meeting

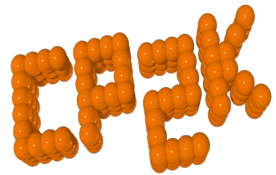
1. Introduction
2. Current Development Efforts
3. CP2K on Intel Xeon Max
4. DBM and DBT
5. ERI on FPGAs (UPB)
6. Modernization of the MPI wrapper (Frederick Stein)
7. …
8. Current Issues when running CP2K
9. New member (GLB-UZH)
10. CP2K Release (all)
11. CP2K-related Events (all)



Introduction



Current Development Efforts



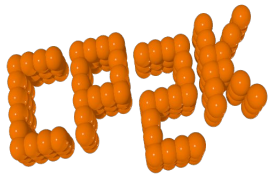
Development Efforts: CP2K on Intel Xeon Max

There are “several” workloads benefitting from memory bandwidth, and “a few” that benefit from compute (FLOPS)*

- GPUs can be beneficial in both cases with global memory exposing high mem. B/W at the expense of cost/capacity (compared to typical CPUs).
- Recent GPU generations grew FLOPS faster than mem. B/W, and “machine balance” shifted to FLOPS (at least when considering specialized CUs).

For scientist, above is acknowledged but often remains abstract even when argumenting with the Roofline model, etc.

⟶ Intel made its 2nd attempt with high-B/W memory on CPUs and the generational speedup for CP2K can be up to 3x

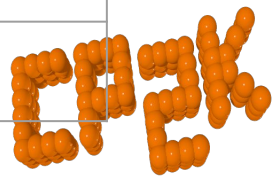


* Meant to be a fair rather than a scientific statement.

Development Efforts: CP2K on Intel Xeon Max (cont.)

Intel 4th Gen. Xeon 9480 w/ 2x64 GB HBM2e and 2x56 cores (8480 w/ DDR5)

Workload	Speedup*	Comment
diag_cu144_broy, bench_dftb, RI-MP2_ammonia, H2O-gga	TTS < 20 seconds	Too small since a few generations
32-H2O/RPA/MP2	1.8x (1.6x)	more compute-bound
QS_ot_ls/H2O-256	1.8x (1.5x)	
QS/H2O-512	1.8x (1.5x)	
QMMM/CIC-19	1.9x (1.7x)	
QMMM/CBD_PHY	3.1x (2.8x)	
QS_DM_LS/H2O-DFT-LS (NREP=3, MAX_SCF=20)	2.2x (1.8x)	LS regularly shows GPU acceleration (DBCSR)



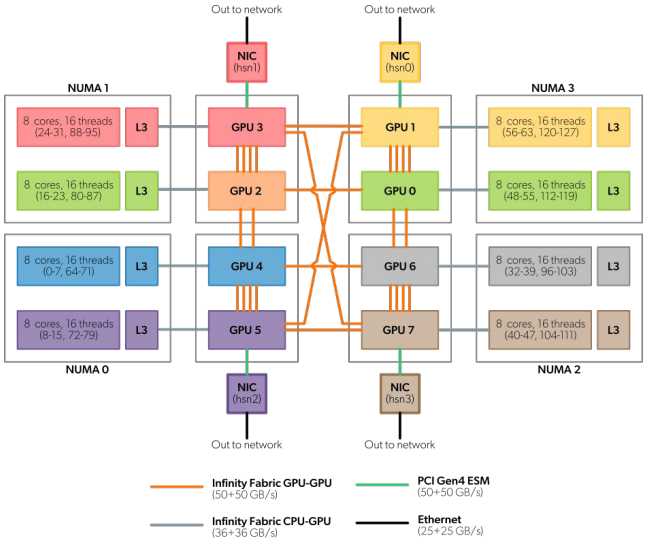
* Comparison with previous gen. Intel Xeon (8360Y) using same binary built with GNU Compiler Collection, Intel MKL, and LIBXSMM.

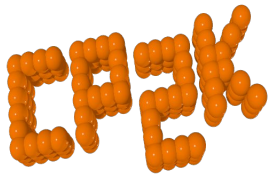
Development Efforts: DBM and DBT

- Nice 3x GPU speed up on LUMI
- Limited by Host-to-Device communication:
  - Remove optimization for square proc grids in `dbt_contract`.
  - GPU-to-GPU communication could unlock further 2x speedup.
  - For `dbm_multiply` rather straightforward.
  - For `dbt_reshape` requires major refactoring (not planed ATM).

- Multi-GPU is essentially a new architecture:
  - Data has to remain on GPU.
  - GPU-to-GPU communication is key.
  - More workloads onto GPU (Amdahl's law).

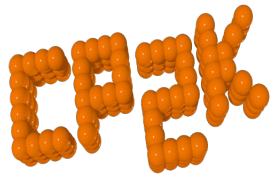
LUMI GPU Node





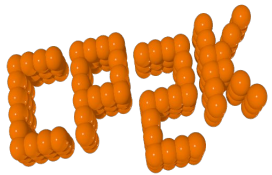
Development Efforts: cuSOLVERMp

- New Eigensolver from Nvidia (documentation, code example).
- Supports multi-node and multi-GPU.
- Faster than ELPA.
- Show case with VASP.
- **Nvidia is looking for a large science case with CP2K.**
  - Good opportunity to finally fix grid code for large basis sets (#1785).
- For AMD GPUs there will soon also be DLAF from CSCS.



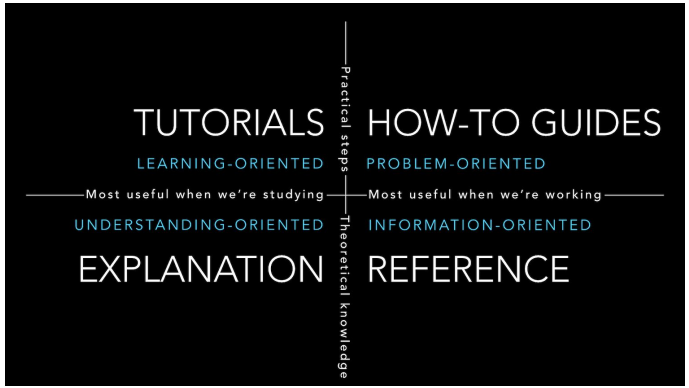
Development Efforts: Revive PAO-ML

- Apply Equivariant Neural Networks to PAO-ML.
- Use pyTorch for the ML.
- Enable Linear Scaling DFT via DBM:
  - Introduce intermediate API layer (remember `cp_dbcsr_ ?`).
  - Reduce API surface: Support only `REAL(dp)` and avoid `dbcsr_get_data_p`.
  - Allow switching between DBCSR and DBM via input keyword.



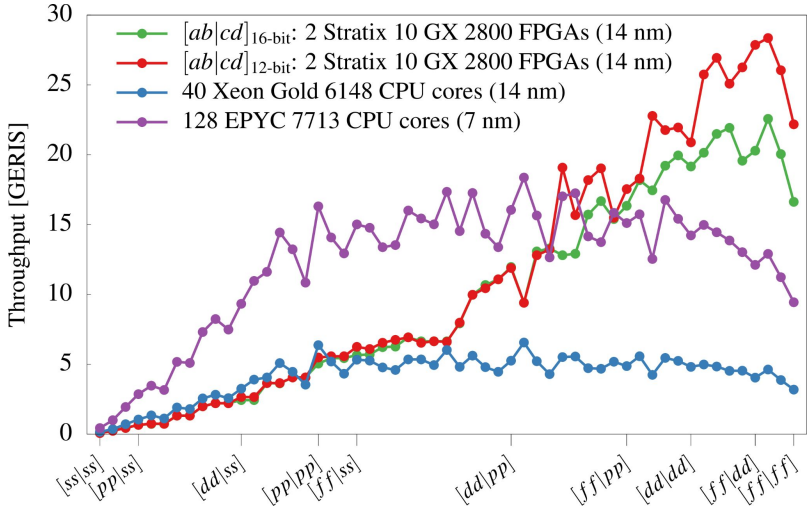
Development Efforts: Revamp Documentation

- Move user docs out of the wiki...
- Use git and pull requests instead.
- Use Markdown and Sphinx for formatting.
- Use Algolia for search.
- Use *The documentation system*:
  - **Tutorials:** Migrate cp2k.org/exercises:common.
  - **Howtos:** Migrate cp2k.org/howto.
  - **Reference:** Port manual generator from XSLT to Python.
  - **Explanation:** Write / curate textbook style articles.

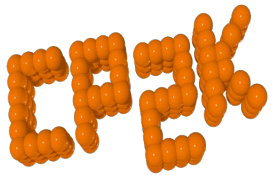


Development Efforts: ERIs on FPGA (UPB)

- Compute and compress Cartesian ERIs
  - Rys quadrature and arbitrary bitwidth compression
  - reach 10 GERIS (10 x 10^9 ERIs per second) on one Intel Stratix 10 GX 2800 FPGA
  - on two Stratix 10 GX 2800 cards outperforms libint
    - on 40 Xeon Gold 6148 CPU cores by up to 6.0x
    - on 128 EPYC 7713 CPU cores by up to 1.9x
    - PCIe (6 GB/s) is a practical bottleneck
- Ongoing work for FPGA kernels: compute and compress spherical ERIs
  - preliminary results for [ff|ff]:
    - 7.27 GERIS on one Stratix 10 GX 2800
    - 2.16 GERIS for libint on 128 EPYC 7713 CPU cores



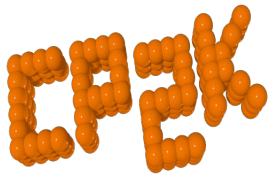
from [ss|ss] to [ff|ff]



Development Efforts: ERIs on FPGA (UPB)

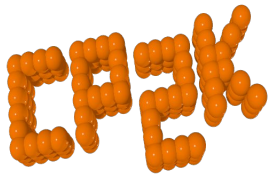
Current Work: integration into CP2K

- ERIs on FPGAs require large batches of ERI-classes
- support for multiple FPGAs per node
- both CPUs and FPGAs need to be used for FPGAs
- → currently replacing atom-based HFX-distribution code by a batched-ERI-class code with global load balancing



Modernization of the MPI wrapper (Frederick Stein)

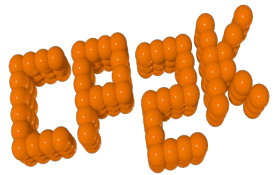
- Currently/Before: CP2K employs the Fortran-90 based MPI wrapper
  - drawbacks: integers as handles, unclear what interfaces the libraries actually provide (rank combinations, derived types), non-blocking communication, …
  - Solution: mpi_f08 (fully compatible with Fortran 2008+TS29113)
  - Currently tested with OpenMPI and Intel oneAPI
  - Current issues: Not available with Gfortran version <9, compiler bugs with MPICH+Gfortran 11 (our standard build at the dashboard)
- Latest developments:
  - Drop support for MPI 2 (already applied to DBCSR, too)
  - Wrap integer handles in derived types (serial mode!)
  - Switch to mpifort and mpiexec in favor of mpif90 and mpirun
  - Apply OOP to handles (to be merged today)
  - Drawback: Finalization lacks compiler support
  - Currently: cp_para_env_types, cp_para_cart_types, cp_blacs_env



Current Issues when Running CP2K

Using the CP2K API to launch multiple instances of CP2K on modern supercomputers with GPU support.

- How are the GPUs allocated to the individual CP2K processes?
- Is it necessary to change to code to enable correct allocation of GPU hardware to each CP2K call?



CP2K-Release

CP2K v2023.1:

- Add gradients for SOS-MP2 and RPA incl. benchmarks (#2208,#2271,#2473)
- TDDFT/Linear Response: Add GAPW/GAPW_XC and ADMM/GAPW options (#2200)
- TDDFT: Add excited state forces as property (#2363)
- RI-RPA: Allow for XC correction in ADMM RI-RPA (#2216)
- RTP: Velocity gauge and magnetic delta pulse (#2343)
- GW: Automatically extrapolate k-point mesh (#2229)
- xTB: Add vdW options (#2431)
- xTB: Fix electronic energy dependence on EPS_DEFAULT (#2287)
- Vibrational analysis: Raman Intensities (#2263)
- New pseudopotentials and basis sets (#2472, #2193)
- Improve NewtonX interface (#2443)
- Fist: Add LAMMPS style tabulated pair potentials (#2313)
- EC: Variational Density-Corrected DFT (DC-DFT) (#2322)
- Update active space interface (#2346)
- Helium: Add missing xyz output format (#2432)
- SIRIUS: Add support for libvdwxc (#2270)
- ELPA: Fix block size issue on GPU (#2407)
- Drop Support for MPI 2.0 (#2438)
- Add experimental CMake build system (#2364)
- Fix regtests on ARM64 (#1855)
- Start testing with Address Sanitizer (#2306)
- Start testing on macOS Apple M1 (sponsored by MacStadium)

CP2K-related events:

Ideas: Paderborn

- QM/MM together with Gromacs
- Post-HF in CP2K (EXX, RPA, GW,...)

- M. Watkins
  - Regular CP2K Workshop/School