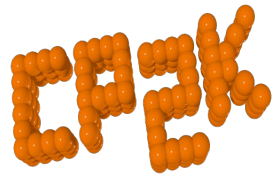
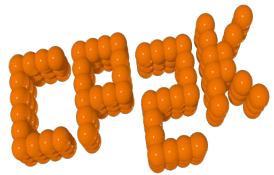
CP2K-GPU Meeting

2022-07-07 14:00-16:00



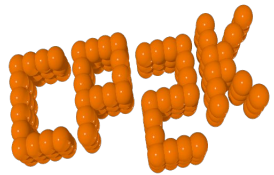
CP2K-GPU Meeting

1. Introduction
2. State of GPU Acceleration in CP2K
3. Current or Planned Projects
4. Contacts/Collaborations to external people
5. CPU Architectures in LIBXSMM 2.0 (5m)
6. CPU Memory Placement for HBM+DDR (5m)
7. DBCSR-GPU



Introduction

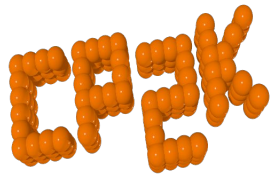
Short round of Introduction and Background (GPU experience)



State of GPU Acceleration in CP2K

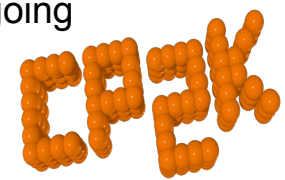
Status of GPU Support in CP2K

Library	Status	Accelerates	Backends	NGC Container
DBCSR	Ready	LS-SCF	CUDA, HIP, OpenCL	Included
grid	Ready	GPW	CUDA, HIP	Included
pw	Ready	SCCS	CUDA, HIP	Included
COSMA	Ready	RPA	CUDA, HIP	Included
SIRIUS	Ready	PW DFT	CUDA, HIP	Included
ELPA	Ready (kinda)	Diagonalization	CUDA	-
Alternative eigensolver	In progress	Diagonalization	CUDA	-
SpFFT	In progress	GPW, SCCS	CUDA, HIP	-
DBM	In progress	RI methods	CUDA, HIP	-
SPLA	In progress	MP2	CUDA, HIP	-
Two-electron integrals	In progress	HFX	-	-
GEEP	Planned	QM/MM	-	-
libxc	Planned	GPW	-	-
One-electron integrals	Planned	GPW	-	-



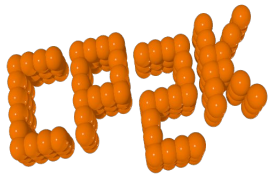
State of GPU Acceleration in CP2K

Notes
- dbcsr
  * libxsmm (intel), cuda backend, hip backend (CSCS,AMD,..), OpenCL (Hans Pabst)
- grid (collocate and integrate)
  * Ole has rewritten in C, then CUDA
  * performance issues with large basis sets (plan to work on it at the end of the year)
- pw
  * original Implementation from Andreas Glöss
  * Mathieu und Ole moved it to HIP
- COSMA
  * mostly finished, for NVidia and AMD
  * improved communication schema implementations recently (esp. for RPA)
  * GPU-to-GPU communication,
- SIRIUS
  * some functionality noch available in CP2K, option passing from CP2K missing, work ongoing
  * recent work on LDA+U
- ELPA
  * no real owner, can pick GPU kernel, but doesn't work well



State of GPU Acceleration in CP2K

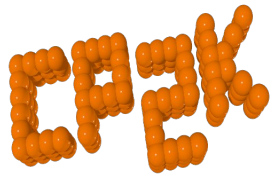
Notes
- COSMA, SIRUS, ELPA
  * Intel progress and support for other GPU vendors ongoing
  * will be upstreamed when done
  * SIRIUS in completion stage
- Alternative eigensolver
  * at CSCS, distributed eigensolver on GPUs: DLA-future (finished by the end of the year), integration into C
- SpFFT, CSCS
  * GPU support finished
  * challenge to integrate into CP2K
- SPLA, CSCS
  * GPU support finished
  * challenge to integrate into CP2K
- DBM
  * reimplementation of DBCSR with BSD license, mainly for tensors
  * ongoing work for OpenMP and CUDA
- ERI
  * libint has GPU acceleration for 3-center matrix elements will be released soon
  * TODO: get together for ERIs on accelerators



State of GPU Acceleration in CP2K

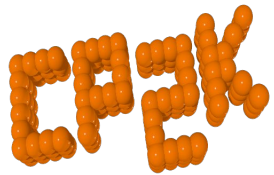
Notes

- GEEP
  * expensive part of QM/MM
  * need to do it, not many developers
  * TODO:
- libxc:
  * has GPU support
  * integration task
- one-electron integrals
  * in gpw application especially the force is relevant



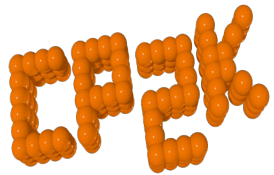
Other Current and Planned Projects

- Submatrix with HIP
- ERI on GPUs? (UPB/PC2 does FPGA)



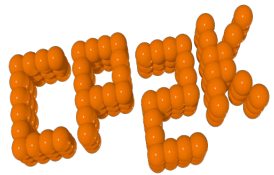
Contact (or Collaboration) with External People

- Nvidia:
  - CSCS in general, Ben Cumming (access, contact to library development team)
  - Ole Schütt, Matt Watkins
  - PC2 (Nvidia US)
- AMD:
  - CSCS (access)
  - Lumi (Alfio Lazzaro, Hutter, Mathieu,...)
- Intel:
  - Abhishek Bagusetty (SYCL, dpcpp, access)
  - Hans Pabst (connections)



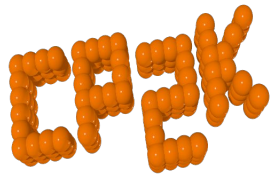
CPU Architectures in LIBXSMM 2.0 (5m)

- ARM cpu support in libxsmm (SVE Fujitsu (512 bit) but also 256 bit)
- maybe GPU support in libxsmm in the future
- maybe kernels for Intel GPUs
- broader support for other hardwares in use in HPC
- API refactoring ongoing
- release 2.0 end of this year or begin of next year

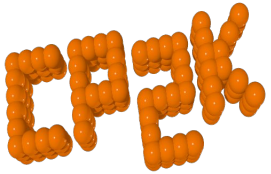


CPU Memory Placement for HBM+DDR (5m)

- 64 GB HBM2 per socket



DBCSR-GPU



- DBCSR v2.3 is out
- Current backends: CUDA (NVIDIA), HIP (NVIDIA, AMD), OpenCL (*)
- Things to do for the next major CP2K release:
  - Autotuning kernels for A100 and Mi200
  - Provide a generic kernel for GPUs (BLAS?) when a tuned kernel is not available
    - currently it fallbacks to CPU, maybe call cuBLAS? instead of fallblack
  - Enable G2G MPI communication (with contribution by AMD)
    - probably in next release, already good performance
- Question:
  - Can the DBCSR-tensor part be replaced by the new DBT (aka. DBM) library?
    - DBT=fork of DBCSR-Tensor code with removals and changes
    - unclear if CP2K still uses DBCSR-tensor code somewhere
    - DBM currently can’t completely replace DBCSR (example: symmetric matrices)
    - DBM currently not working for Apple M1 (OpenMP locks?)
    - Todo Ole: list of required and available features
    - Plan Ole: move complex/special functionality out of DBCSR into CP2K
    - Plan Alfio: maybe a common set of interfaces and runtime/compile time switch?
  - Would it be offload OpenMP a better solution for Fortran applications?
    - direct offloading in Fortran without glue-layer
    - performance is challenging, specialization for targets is required

PRACE Benchmark Suite

Should we try to swap the LS-DFT benchmark from the PRACE Benchmark Suite for one that targets the tensor code?

That code powers many of our recent additions, including GW and the various RI methods. As a bonus, it would also give vendors a new opportunity for contributing optimizations.

Differences:

- block sizes (skinny and tall and much more different)
- MPI communication in tensor lib instead of in matrix multiplication

Todo Ole/Hans: confirm maximal number of benchmarks

Other options: QM/MM (GEEP, FIST), SIRIUS?, COSMA

Proposal:

- make preliminary benchmark like above @Ole
- gather experience and insight into scaling and behaviour @all

Topics for next CP2K DEV Meeting:

Topics for next CP2K DEV Meeting:

- DeePMD integration (https://github.com/cp2k/cp2k/pull/1620)

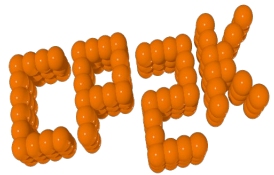
Current issues when running cp2k

- CP2K on CSCS systems seems to trigger some bug in mpi. It is difficult to fully exclude bugs in cp2k either. It affects users running the latest version of cp2k as well as the official version.
- DBCSR was particularly unstable when compiled with GPU support on ORNL and NERSC systems (both cray systems).

Questions

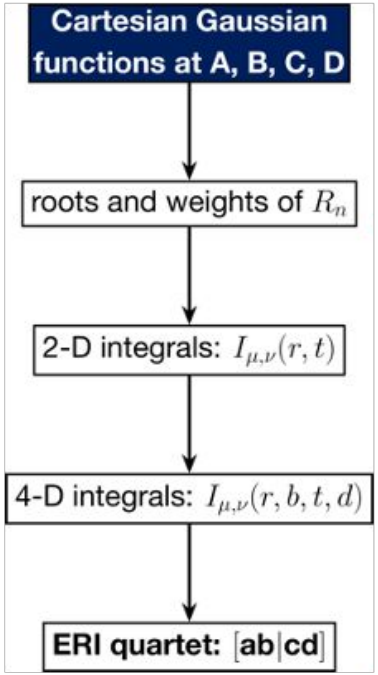
Q: Are there any plans to allow assignment/utilization of GPU acceleration through the Fortran/C++ interface of CP2K?

Q: Support for Apple and other platforms and OSes?



ERI on FPGA

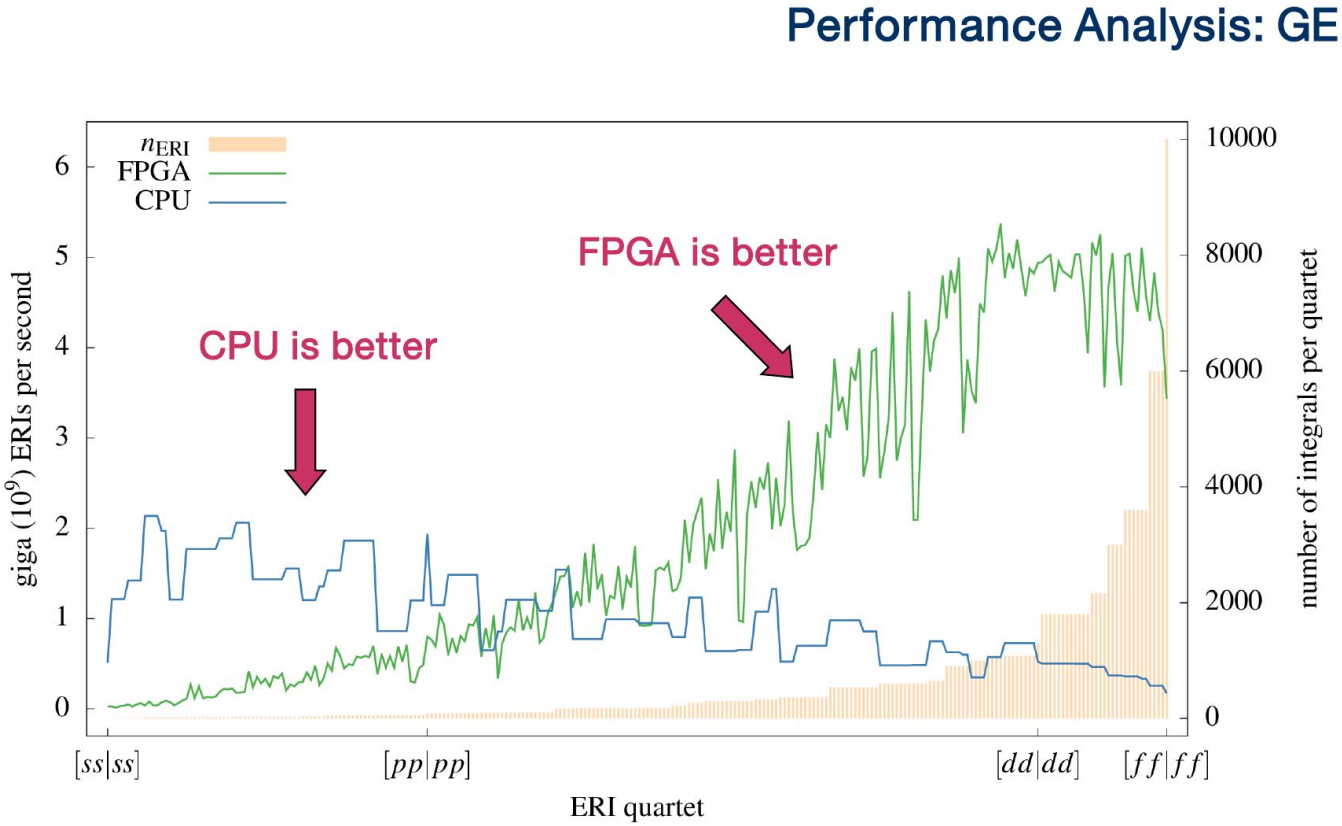
- **Three major algorithms**
  - the McMurchie-Davidson algorithm
    → in `SHARK` of ORCA 5
  - the Head-Gordon-Pople algorithm
    → in `libint` and used in CP2K
  - the Rys quadrature algorithm
    → in `libcint`
- **Rys quadrature**
  - compute $[ab|cd]$ via Gaussian quadrature using a set of orthogonal Rys polynomials
  - features:
    - efficient for ERI quartet of higher angular momentum
    - low memory requirement for intermediate results
      → use of fast on-chip memories in FPGA
    - numerically very stable in computation
      → allows single-precision floating-point arithmetic



J. Rys et. al: *Computation of Electron Repulsion Integrals Using the Rys Quadrature Method*. J. Comput. Chem. 1983.

T. Helgaker et. al: *Molecular Electronic-Structure Theory*. John Wiley & Sons, Ltd. 2000.

ERI on FPGA


Performance Analysis: GERIS
n_{ERI}
FPGA
CPU
CPU is better
FPGA is better
giga (10^9) ERIs per second
number of integrals per quartet
0
1
2
3
4
5
6
0
2000
4000
6000
8000
10000
[ss|ss]
[pp|pp]
[dd|dd]
[ff|ff]
ERI quartet