SIMULATeQCD
A Simple multi-GPU lattice code for QCD calculations

Christian Schmidt

UNIVERSITÄT BIELEFELD
Faculty of Physics

CRC-TR 211

PUNCH 4 NFDI

Sajid Ali, Luis Altenkort, Dennis Bollweg, David A. Clarke, Henrik Dick, Jishnu Goswami, Olaf Kaczmarek, Rasmus Larsen, Lukas Mazur, Swagato Mukherjee, Marius Neumann, Hauke Sandmeyer, CS, Philipp Scior, Hai-Tao Shu

(HotQCD Collaboration)

PoS LATTICE2021 (2022) 196, arXiv:2111.10354
Github: https://github.com/LatticeQCD/SIMULATeQCD

MITP, March 6-10, 2023, Mainz, Germany

Introduction: HPC in Europ

- Lattice QCD is very compute intensive
- EuroHPC JU develops, runs and maintains HPC infrastructure in the EU with a budget of 7 billion for 2021-2027
- New HPC systems have just been installed, a ExaFlop System in in development.
- Compute time is distributed by EuroHPC JU through Calls for Academia and Industry
- Proposals require demonstration of efficiency and scaling of the used codes

→ **GPUs are ubiquitous in HPC**

EuroHPC Joint Undertaking

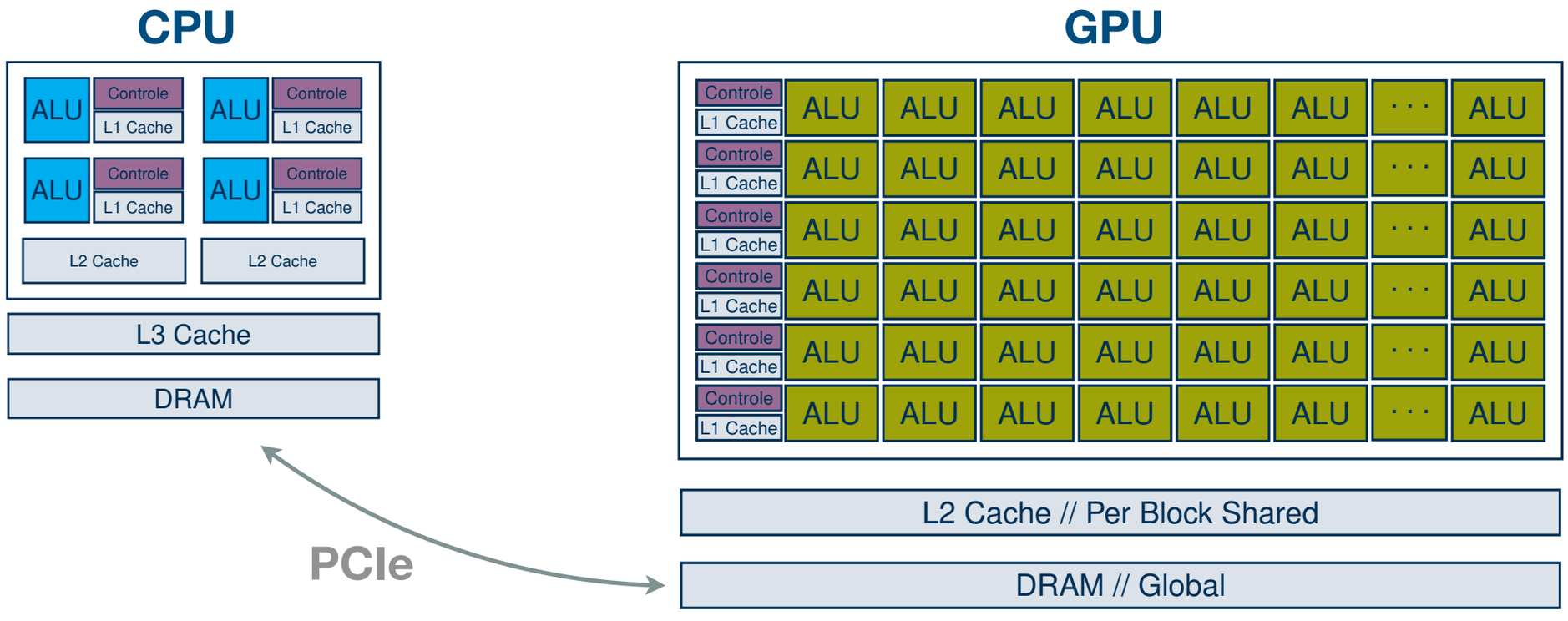
429 PFlop/s (Peak)
#3 on top 500

255 PFlop/s (Peak)
#4 on top 500

Introduction: Some Public Lattice Codes

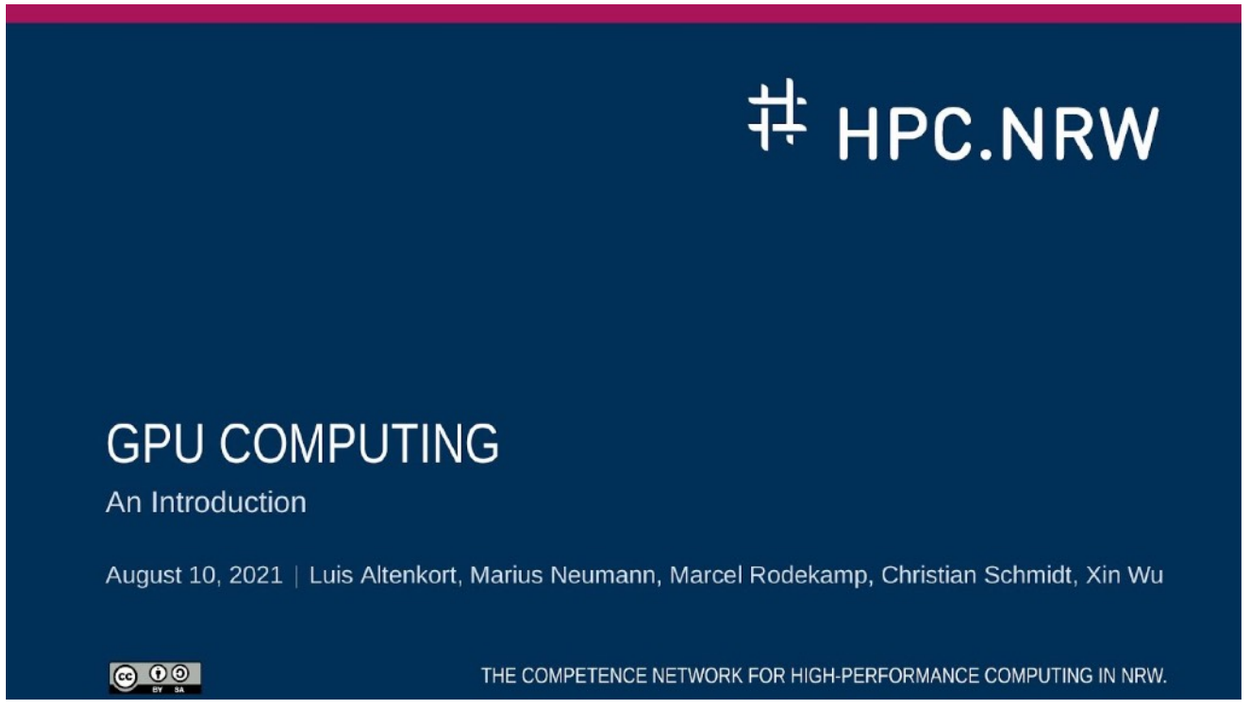
- Bridge++, on arXiv
  Bridge++ is a code set for numerical simulations of lattice gauge theories including QCD (Quantum Chromodynamics), written in C++. It currently focuses on Wilson (clover) fermions and implements a number of Dirac solvers, HMC algorithms, and measurement functions. Support for other fermion formulations is in development.
- Chroma, on GitHub
  Chroma is a software package for lattice field theory and in particular lattice QCD. It supports data-parallel programming constructs relying on the SciDAC QDP++ library written in C++. The QDP++ library presents a single high-level code image to the user, but can generate highly optimized code for many architectures including single-node workstations, multi-core and many-core nodes, clusters of nodes via the QMP library, classic vector computers, and GPUs.
- CL2QCD on arXiv on gitlab.itp.uni-frankfurt.de on Zenodo
  CL2QCD is a Lattice QCD application based on OpenCL, applicable to CPUs and GPUs. It provides the possibility of producing gauge configurations using different algorithms as well as measuring observables on given configurations.
- Grid, on arXiv, on GitHub
  Grid is an open-source lattice QCD framework written in C++ 11 with support for a multitude of architectures, including all Intel x86 SIMD extensions, Arm NEON and 512-bit SVE, and AMD/NVIDIA GPGPU. The framework is designed for performance portability targeting a large variety of current and future supercomputer architectures. The portability issue is resolved by implementation of low-level functions that hide the details of the underlying hardware architecture from the user, e.g., complex arithmetics. Grid achieves 100% SIMD efficiency on all architectures by combining template meta-programming and intrinsics where available. The framework supports MPI for communication. OpenMP is used for on-chip parallelization on CPUs.
- GPT - Grid Python Toolkit, on GitHub
  The Grid Python Toolkit (GPT) is an open-source Python measurement toolkit providing a physics library for lattice QCD and related theories, a module including a digital quantum computing simulator, and an experimental machine learning module. The toolkit is built on the data parallelism (MPI, OpenMP, SIMD, and SIMT) of the Grid lattice QCD framework. GPT is designed for performance portability targeting current and future supercomputer architectures.
- MILC, on GitHub
- openQCD, openQCD-FASTSUM
  The openQCD software package contains simulation programs for lattice QCD based on the HMC (Hybrid Monte Carlo) or the SMD (Stochastic Molecular Dynamics) algorithm offering state-of the-art simulation techniques such as nested hierarchical integrators for the molecular-dynamics equations, twisted-mass Hasenbusch frequency splitting, even-odd preconditioning, twisted-mass determinant reweighting, and deflation acceleration. Several different solvers (CGNE, MSCG, SAP+GCR, deflated SAP+GCR) for the Dirac equation are available. The programs are parallelized and highly optimized for machines with current Intel and AMD processors, but should run correctly on any system compliant to the IEEE 754, ISO C89, MPI 1.2 and (since openQCD 2.4) OpenMP 4.5 standards.
- QUDA, on GitHub
  QUDA is a library for performing calculations in lattice QCD on graphics processing units (GPUs), leveraging NVIDIA's CUDA platform.
- SIMULATeQCD, on arXiv, on GitHub
  a SImple MUlti-GPU LATtice code for QCD calculations. SIMULATeQCD is a multi-GPU Lattice QCD framework that makes it simple and easy for physicists to implement lattice QCD formulas while still providing the best possible performance.

GPU computing 101

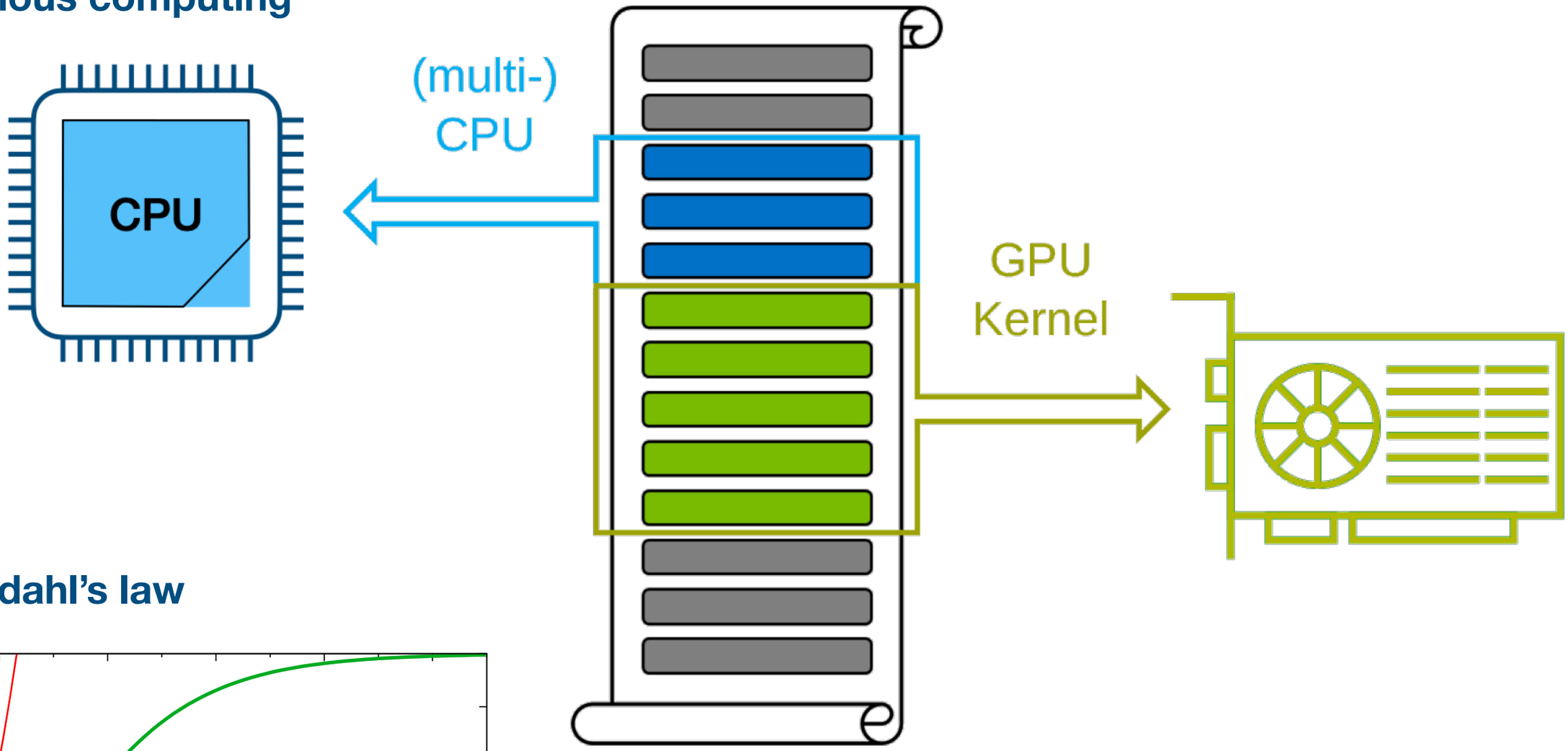


- Need to know the hardware
- Lattice QCD codes are bandwidth limited (Dslash has low arithmetic intensity)
  - ➡ Avoid transfers via PCIe
  - ➡ Have optimal access to GPU memory (access adjusted to SIMD units → structure of arrays instead of array of structures)
  - ➡ compress your data

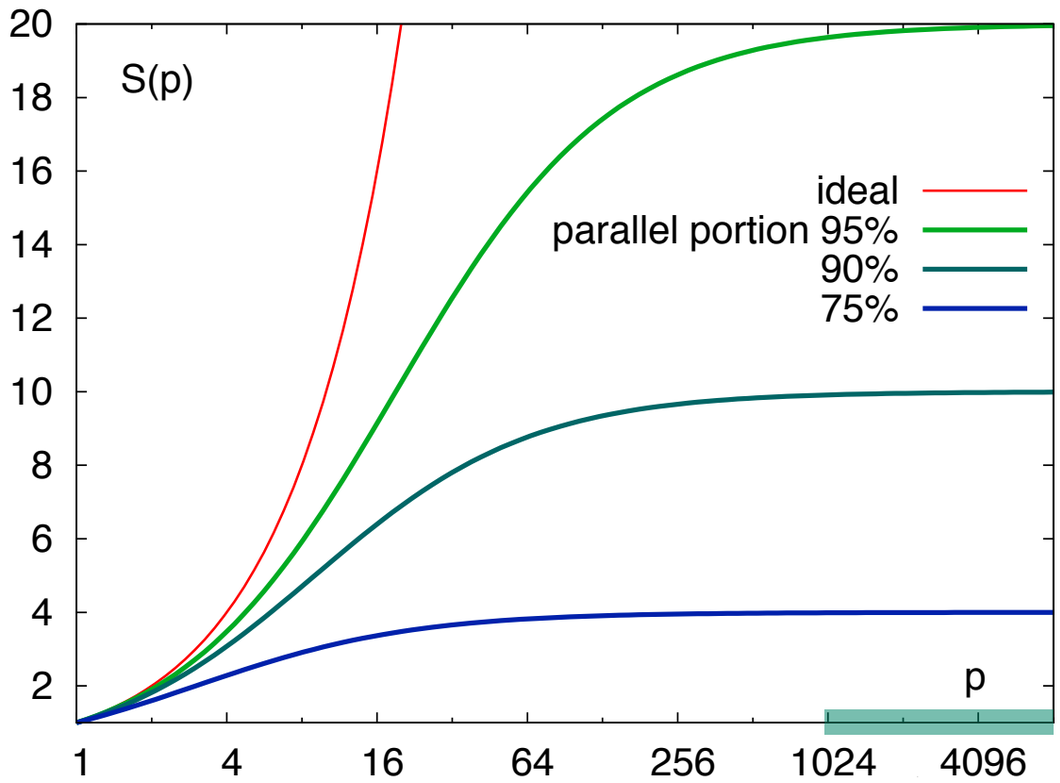
https://hpc-wiki.info/hpc/GPU_Tutorial



Heterogenous computing



Problem: Amdahl's law



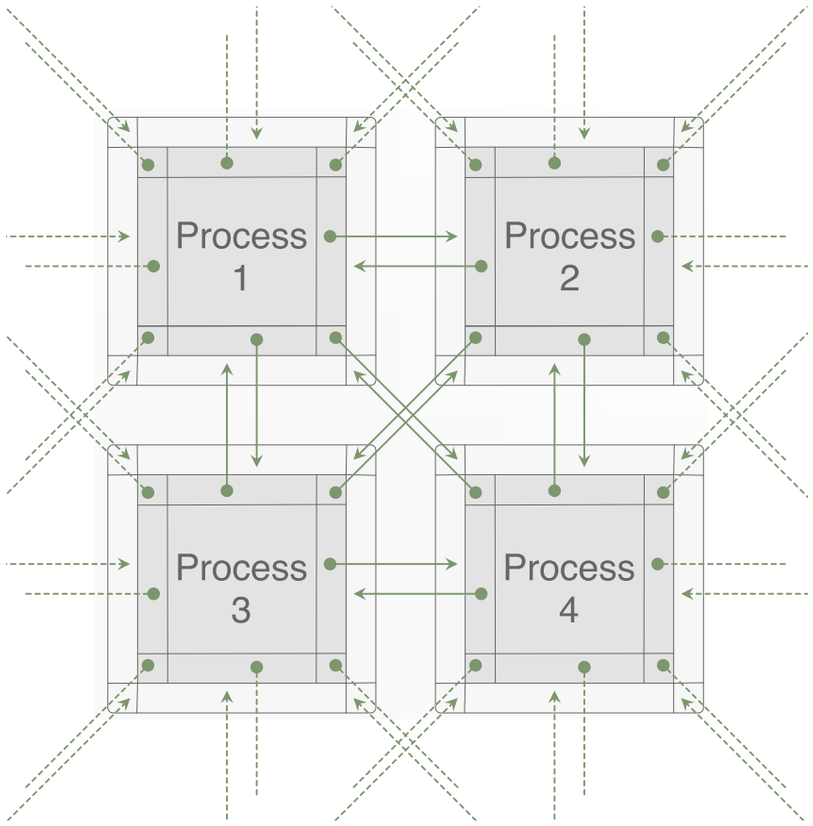
- Speed-up is finite in the limit of large compute cores
- The limit is set by the fraction of the parallelizable part
- Speed-up levels off surprisingly early

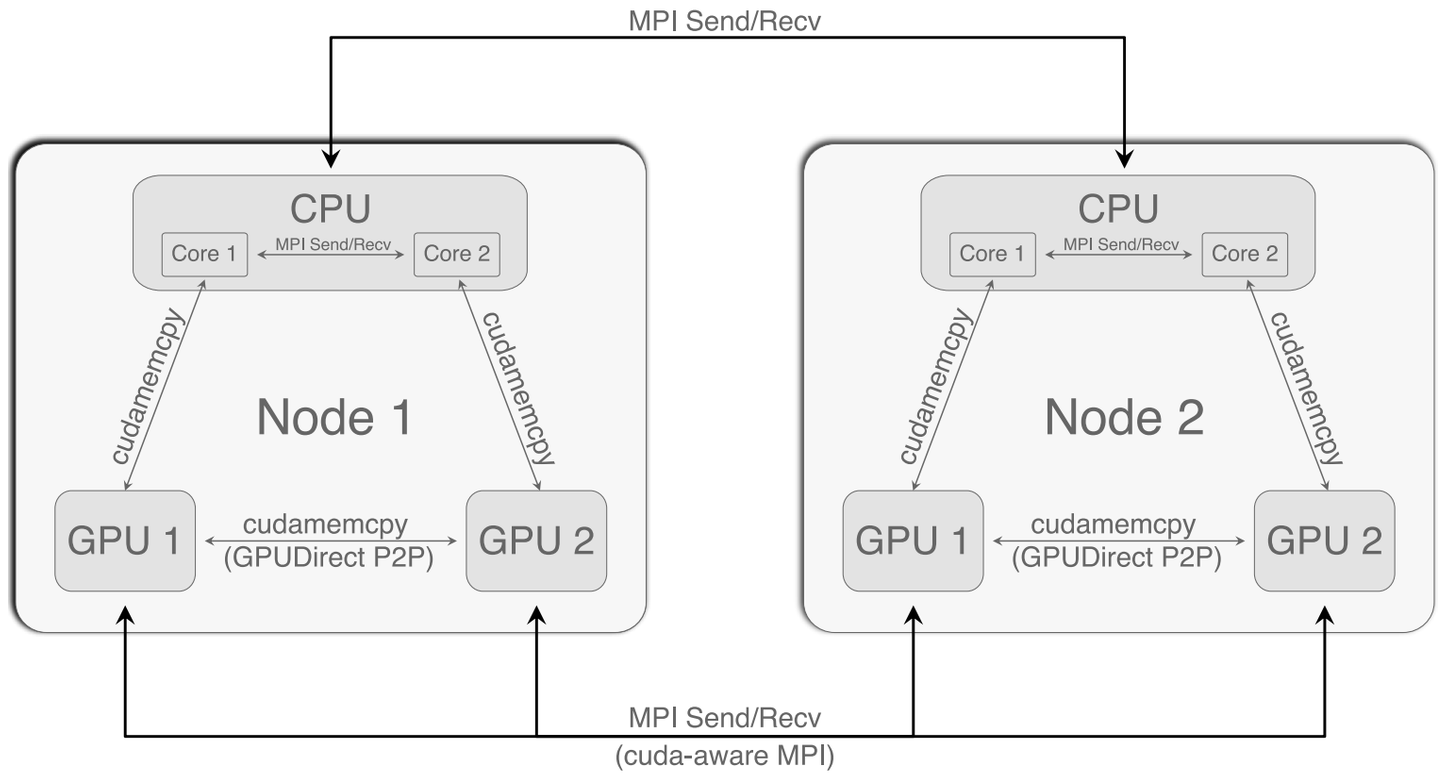
SIMULATeQCD is a multi-GPU, multi-node lattice code written using C++17, utilizing the OOP paradigm and modern C++ features. It currently supports HISQ and Pure Gauge calculations. Wilson is planed. We aim on

1. High-performance
2. Efficiency on multiple GPUs and nodes
3. Ease to use
4. Flexibility with respect to hardware

Performance

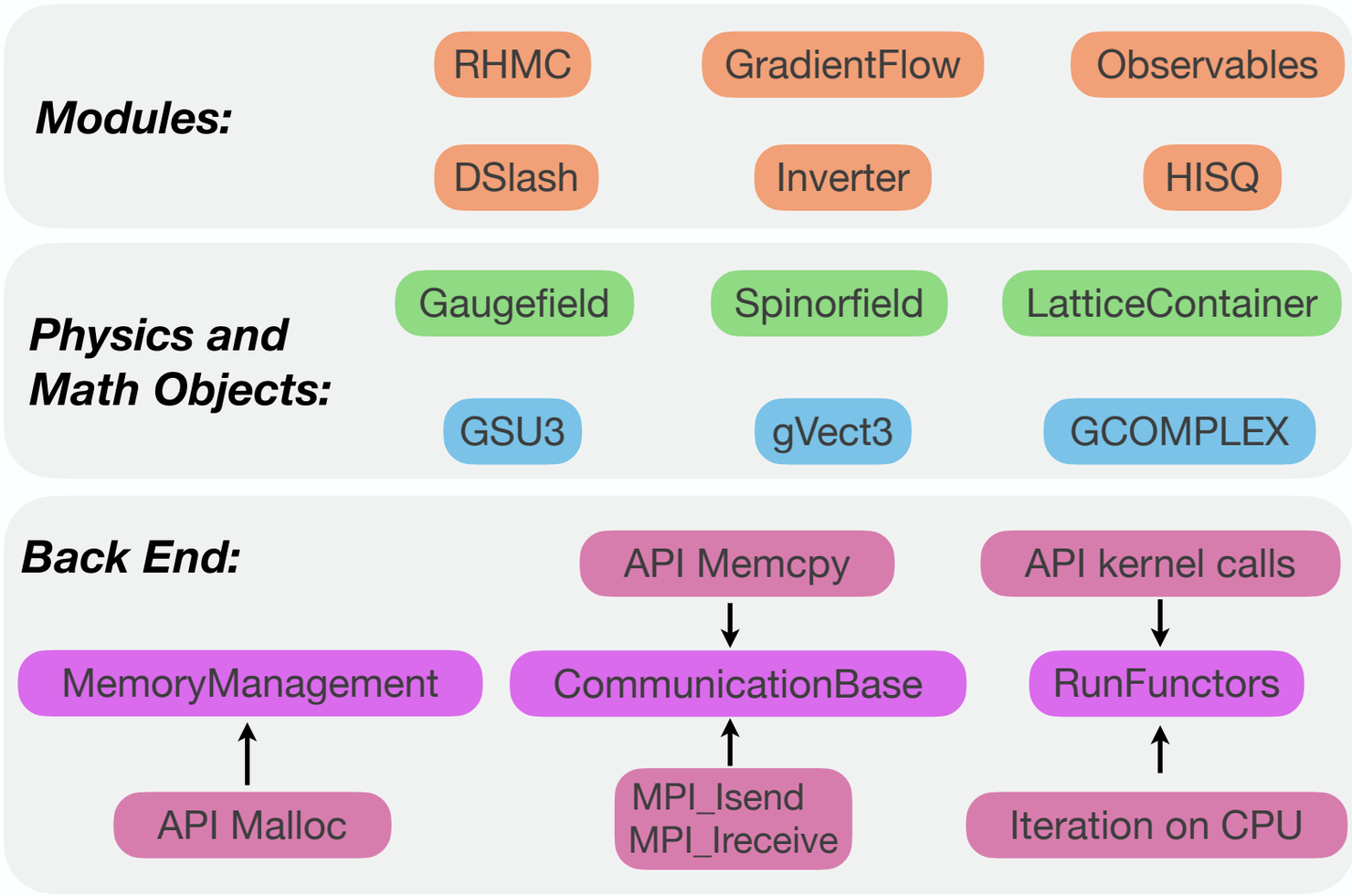
- All calculations are done on the GPU, no transfers via PCIe except I/O to file
- Memory access is optimal (structure of array)
- SU(3) and U(3) matrices are reconstructed from 12, respective 14 floats
- Memory management class





Multi-GPU and multi-note

- Need to communicate halos via explicit memcpy calls
- Use Direct P2P communication between GPUs on the same note, use CUDA aware MPI for communication between notes (Automatic detection)
- Communication is parallel to calculation on the bulk



Ease to use

* Modular code

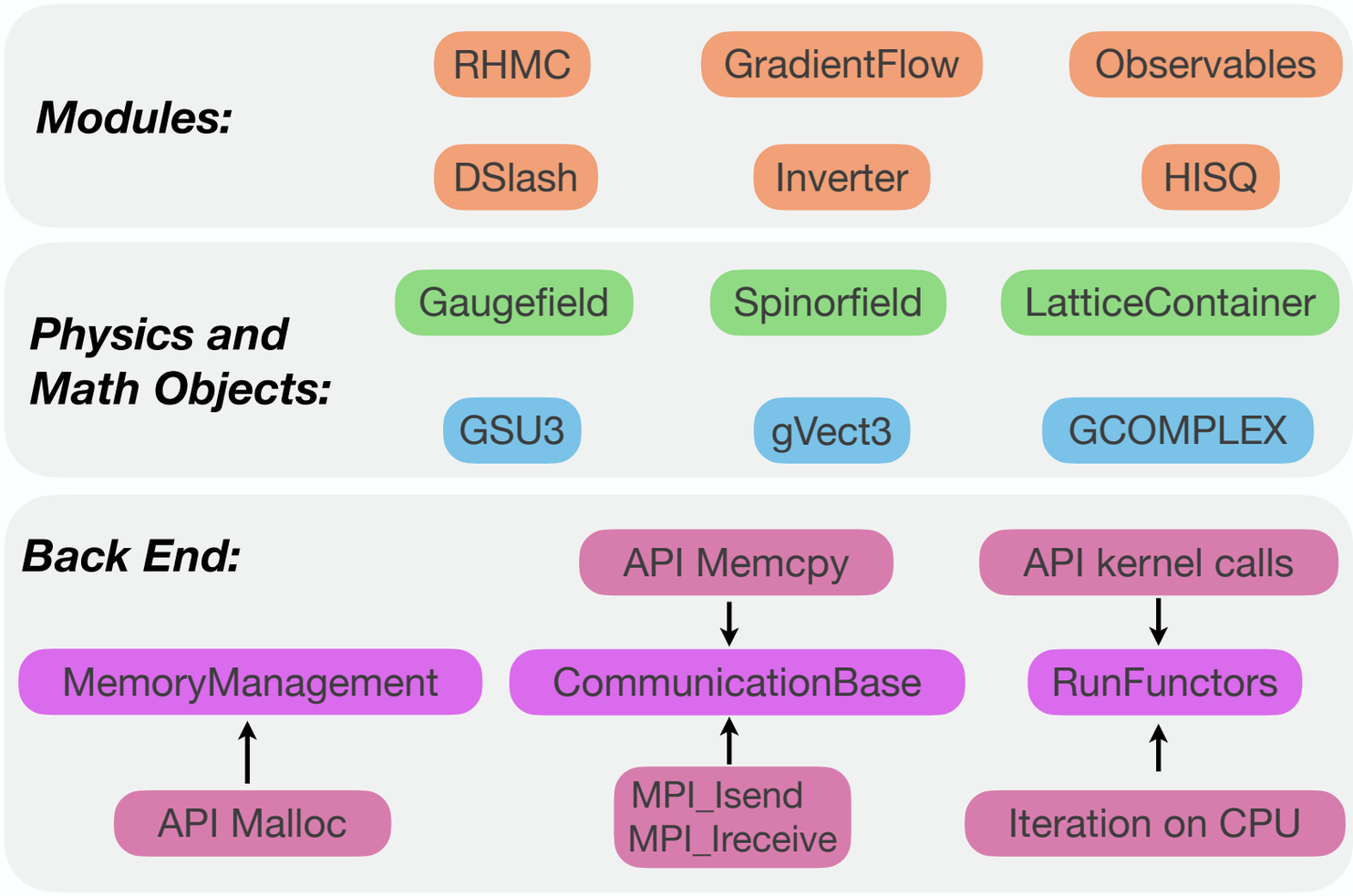
```
template<class floatT, bool onDevice, size_t HaloDepth, CompressionType comp>
struct plaquetteKernel
{
  gaugeAccessor<floatT, comp> gAcc;

  plaquetteKernel(Gaugefield<floatT, onDevice, HaloDepth, comp> &gauge) : gAcc(gauge.getAccessor()) {}

  __device__ __host__ floatT operator()(gSite site){

    floatT result = 0;
    for (int nu = 1; nu < 4; nu++) {
      for (int mu = 0; mu < nu; mu++) {
        GSU3<floatT> tmp = gAcc.template getLinkPath<All, HaloDepth>(site, nu, mu, Back(nu));
        result += tr_d(gAcc.template getLinkPath<All, HaloDepth>(site, Back(mu)), tmp);
      }
    }
    return result;
  }
};
```

Ease to use

- Modular code
- Define arbitrary operators to iterate over arbitrary sets of sites
- Abstract away much of the hardware specific API

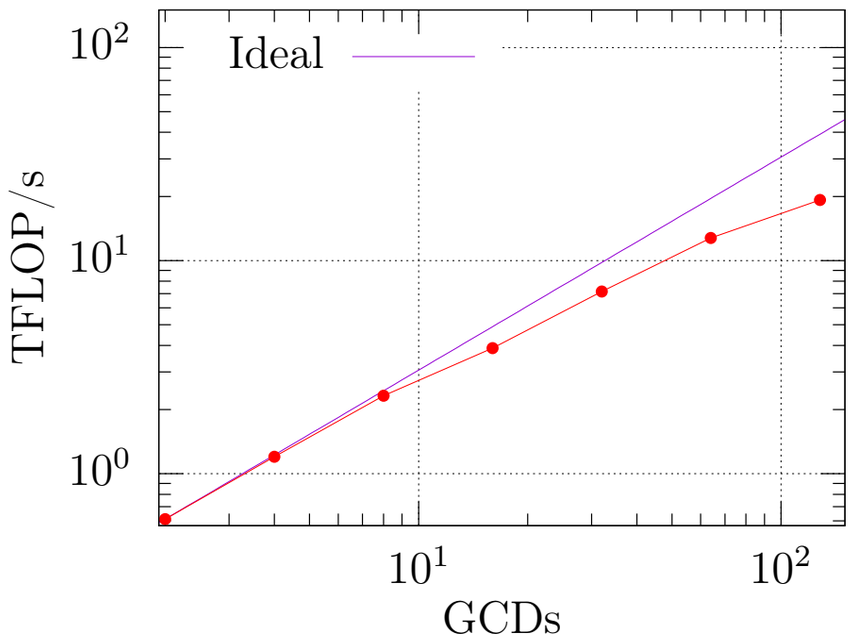
Flexibility with respect to hardware

- Hardware specific APIs collected in backend modules
- Can be easily exchanged
- Currently supported CUDA and HIP

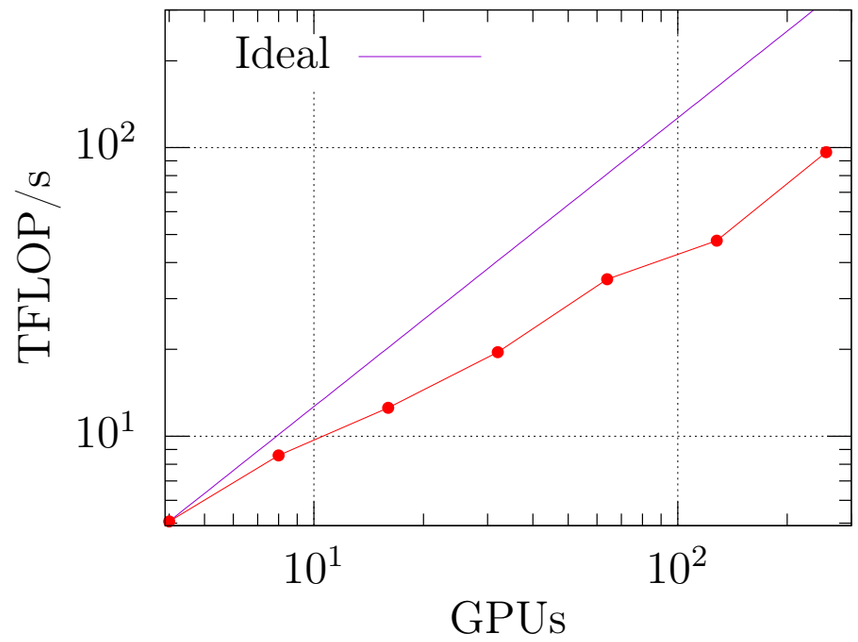


Benchmarks of HISQ Dslash

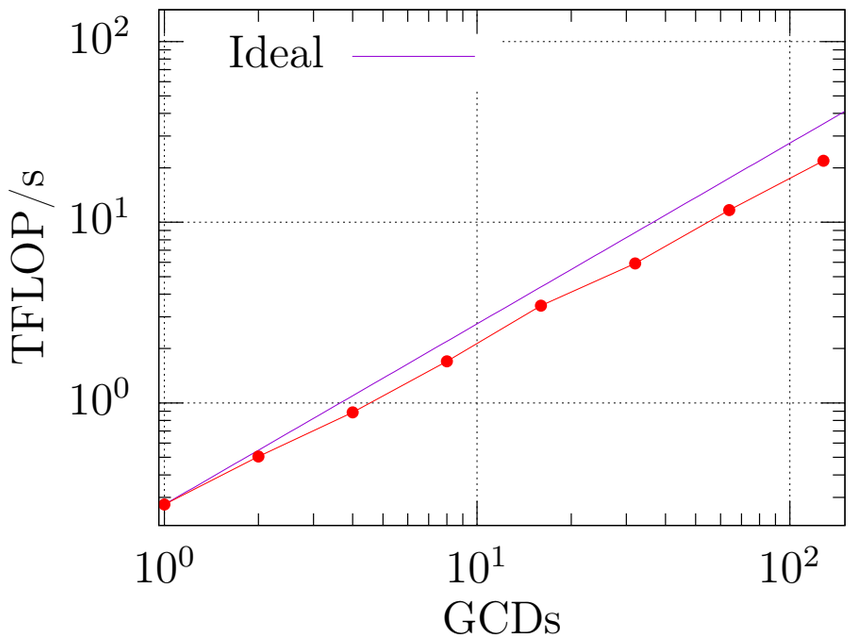
LUMI

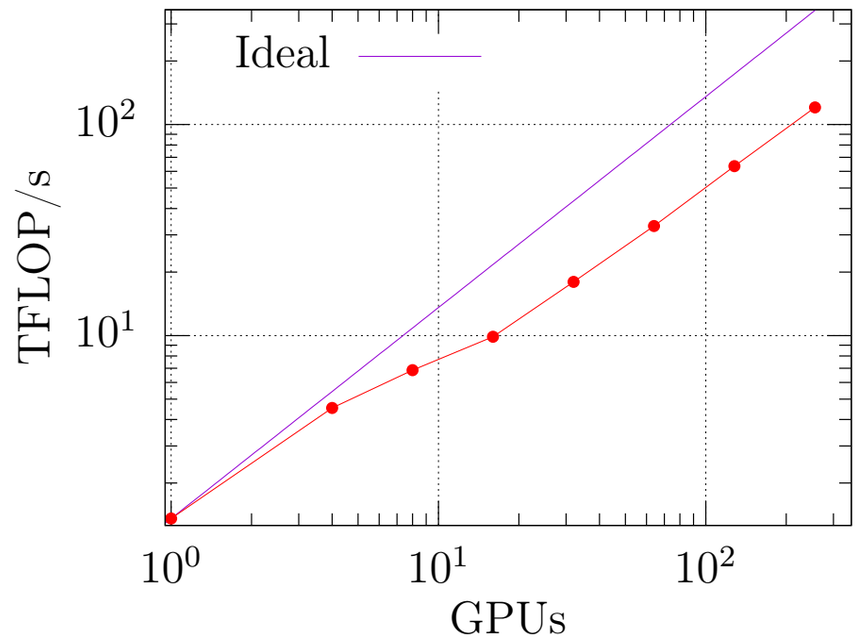


Perlmutter



Strong scaling 96^4





Weak scaling, local lattice 32^4

Outlook

- New actions: Wilson, …
- New Backends: SYCL, oneAPI
- More ILDG tools

Outlook

- New actions: Wilson, …
- New Backends: SYCL, oneAPI
- More ILDG tools

Your are welcome to participate!

Than you for your attention!