SIMULATeQCD - A simple multi-GPU lattice code for QCD calculations

Dennis Bollweg[1], David Clarke[2], Lukas Mazur[3], Olaf Kaczmarek[4]

[1]Brookhaven National Lab,
[2]University of Utah, [3]Paderborn Center for Parallel Computing,
[4]Bielefeld University

Lattice 2023, 07/31/2023

Outline

SIMULATeQCD

Brookhaven National Laboratory

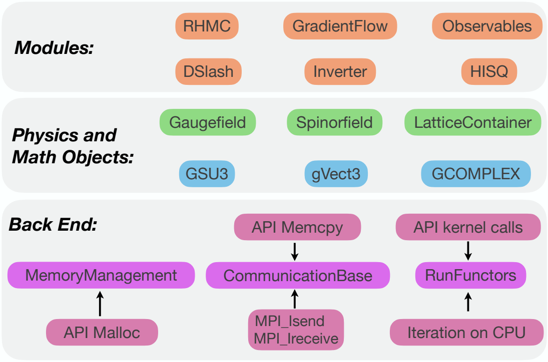
- Open source library for lattice QCD calculations on GPUs
  - Gauge field generation: HISQ RHMC, heat bath & over-relaxation for quenched.
  - Measurements: thermo observables, topology, correlators, etc.
- Written in C++ with CUDA and HIP backends for GPU acceleration.
- Multi-GPU support, D2D communication with GPU-aware MPI or P2P (IPC).
- Available on GitHub: `https://github.com/LatticeQCD/SIMULATeQCD`
- Technical paper: `https://arxiv.org/abs/2306.01098`

Code design

We have worked to develop code that:

- works efficiently on multiple GPUs and nodes;
- is flexible to changing architecture and hardware;
- is easy to use for lattice practitioners with intermediate C++ knowledge.

- Follow OOP paradigm.
- Separate low-level GPU code from high-level "physics" code.



Code design - Compute

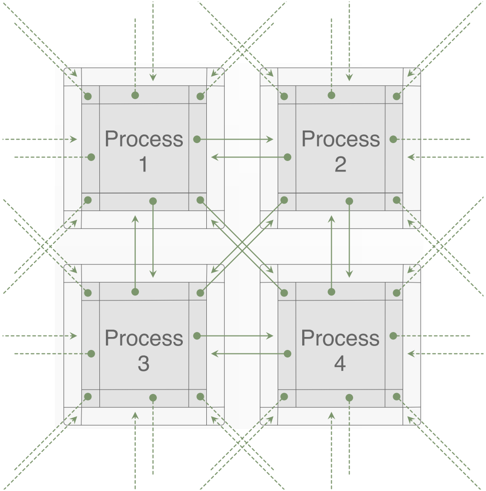
- HIP/CUDA kernel launches hidden away in `RunFunctors` class.
- Define kernels via functor that takes `gSite` as argument.
- Launch via physics objects `.iterateOver()` methods.
- Alternatively: compose via expression template engine, kernel launch triggered at assignment operator.

```
template<class floatT, /*...*/>
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<...>(site, nu, mu, Back(nu));
        result += tr_d(gAcc.template getLinkPath<...>(site, Back(mu)), tmp);
      }
    }
    return result;
  }
};
```

Code design - Communication

Stencil & Halo approach:

- Global lattice split onto GPUs.
- Local lattices extended by halos containing neighboring data.
- Update halos before/after stencil computations from physics objects `.updateAll()` methods.
- Halo data injection/extraction kernels overlap with memcpys.
- `.updateAll()` can be restricted to planes, corners, etc. by user.



Code Performance

$\sim$60% of (HISQ) RHMC run time is spent performing matrix inversions (CG) dominated by $\not{D}\psi_x$ computation.

$$\not{D}\psi_x = \sum_{\mu=0}^{4}\left[\left(V_{x,\mu}\chi_{x+\hat{\mu}} - V^{\dagger}_{x-\hat{\mu},\mu}\chi_{x-\hat{\mu}}\right) + \left(W_{x,\mu}\chi_{x+3\hat{\mu}} - W^{\dagger}_{x-3\hat{\mu},\mu}\chi_{x-3\hat{\mu}}\right)\right]$$

$$V_{x,\mu}: \quad 3\times 3 \text{ complex matrix}, \quad W_{x,\mu}: \quad U(3) \text{ matrix}$$

- 1146 FLOP/site, 1560 byte/site $\rightarrow$ FLOP/byte $\sim$ 0.73. $\not{D}\psi_x$ computation is bandwidth bound!
- Arithmetic intensity can be increased by applying the gauge field to multiple right-hand sides (rhs) at once.

- $\not{D}$-kernel achieves up to 1.36TB/s memory throughput on single A100 and 1.3 TFLOP/s compute.
- Multi-RHS allows to reach 11 TFLOP/s on 4xA100 node.
- Good weak and strong scaling on Perlmutter with Slingshot 11 network.

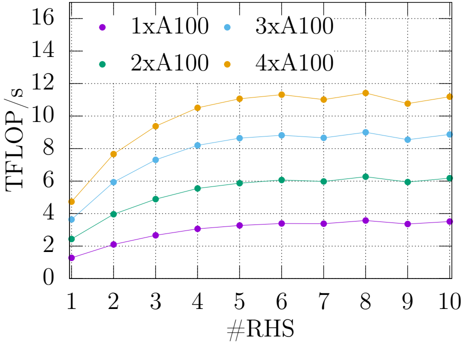
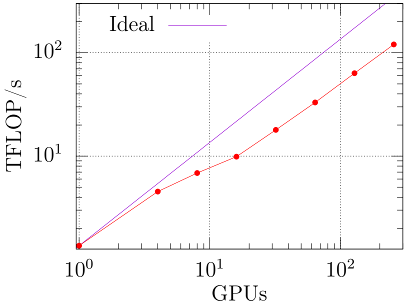


Figure: Multi-RHS $\not{D}$ Benchmark on 4xA100 node.

Code Performance - CUDA

- $\not{D}$-kernel achieves up to 1.36TB/s memory throughput on single A100 and 1.3 TFLOP/s compute.
- Multi-RHS allows to reach 11 TFLOP/s on 4xA100 node.
- Good weak and strong scaling on Perlmutter with Slingshot 11 network.



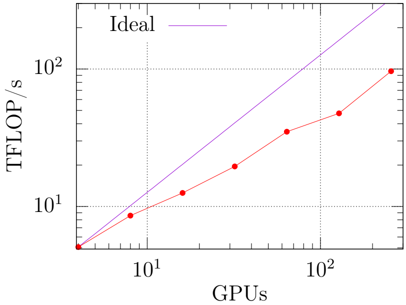


Figure: Left: Weak scaling with 32^4 local lattice. Right: Strong scaling with 96^4 global lattice on Perlmutter.

Code Performance - HIP

- $\not{D}$-kernel on single GCD of MI250x achieves about 900 GFLOP/s, 950GB/s mem throughput.
- Very good weak and strong scaling on Frontier & Lumi-G systems.
- Multi-RHS $\not{D}$ on MI250x does not yield perf. increase yet.

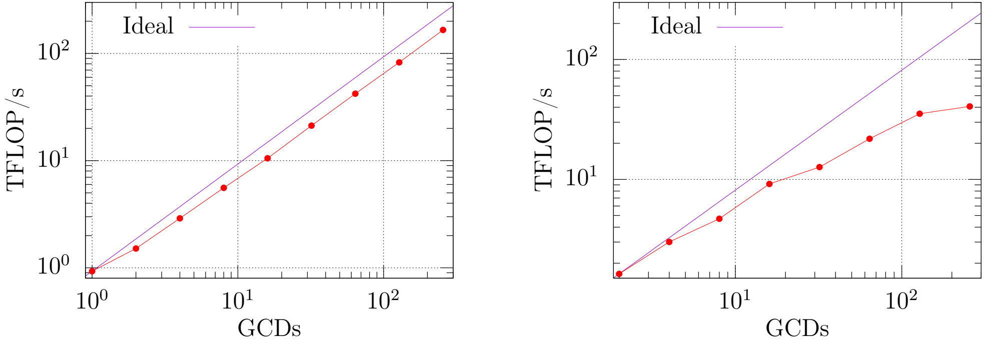


Figure: Top: Weak scaling with 32^4 local lattice. Bottom: Strong scaling with 96^4 global lattice on Frontier.

Summary and Outlook

- Version 1.0 release of SIMULATeQCD, an open source lattice QCD library targeting GPUs.
- Optimal single GPU performance for HISQ RHMC on Nvidia systems, good scaling to multiple nodes.
- Good single GCD performance on new AMD MI250x machines and great scaling on Frontier.

Work in progress:

- Optimization effort for MI-250x with EuroHPC, AMD & CRAY started recently.
- First steps towards OneAPI backend under way.