Software and algorithms for HISQ and Domain Wall Fermions

Dennis Bollweg
Brookhaven National Laboratory

SciDAC-5 kickoff meeting, 12/2/2022, JLab

Outline

Brookhaven National Laboratory

SciDAC-5 BNL science cases

Brookhaven National Laboratory

- Microscopy of the QGP with heavy quarks
  - Heavy quarks are created in very early stages of heavy ion collisions and are subject to the QGP during its entire evolution.
  - Non-perturbative determination of heavy quark potential, heavy quark diffusion coefficent needed for dynamical models of e.g. quarkonia.
  - Need large, fine lattices with high statistics.
  - Goal: Ready SIMULATeQCD HISQ code for exascale generation of supercomputers.
- Hadron Structure
  - Full 3D structure of the proton is encapsulated in generalized parton distributions (GPDs) and transverse momentum dependent distributions (TMDs).
  - GPDs and TMDs are defined on the light-front and therefore not directly accessible in lattice QCD calculations.
  - Recent development of LaMET circumvent this by relating them to quasi-GPD/TMDs that are defined with space-like separations via factorization.
  - Goals: Develop hadron structure measurement code and DDHMC for Domain Wall Fermions.

SIMULATeQCD

Brookhaven National Laboratory

- GPU Code for lattice QCD (quenched, staggered or HISQ gauge field generation, gradient flow, etc ...).
- Started in late 2017/2018 by Lukas Mazur as a thesis project, now central part of HotQCDs software stack.
- Written in C++ with CUDA & HIP back-ends for GPU acceleration.
- Multi-GPU support, D2D communication via CUDA P2P or GPU-aware MPI.
- Used in large-scale computing projects on Top500 systems including Perlmutter (NERSC), Summit (OLCF), Marconi100 (CINECA), JUWELS (JSC), Piz Daint (CSCS).
- Available on GitHub: `https://github.com/LatticeQCD/SIMULATeQCD`

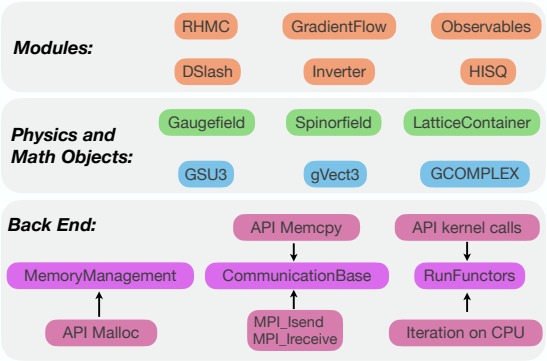
Code design

Brookhaven National Laboratory

We have worked to develop code that is:

- high-performant;
- 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.
- Expression templates and overloading to express calculations intuitively without sacrificing performance.
- Custom kernels via function objects.

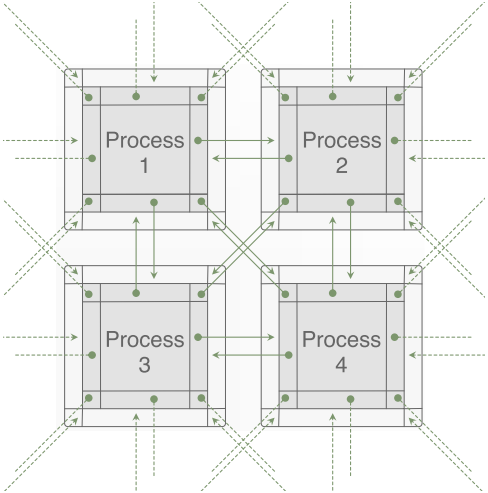


Code design

Brookhaven National Laboratory

Distribution onto multiple GPUs:

- Split lattice onto GPUs (filled squares).
- Extend local lattices by halos (empty squares).
- Communicate halos for stencil computations.
- Overlap communication & computation using async. memcpy.



HIP support

Brookhaven National Laboratory

Approach for incorporating HIP:

- Wrapped CUDA API functions with macros in central wrapper header.
- Used hipify tool to translate CUDA code to HIP.
- Due to code design, only few lines in back-end code are changed.
- Preprocessor macros to switch between HIP/CUDA code in few remaining places (kernel launches, using external libraries).

```
template<...>
void RunFunctors<onDevice, Accessor>::iterateFunctor(...)
{
    //...

    if (onDevice) {
#ifdef USE_CUDA
        performFunctor<<<gridDim,blockDim,0,stream>>>(getAccessor(), op, calcReadInd,
                                                      calcWriteInd, elems_x);
#elif defined USE_HIP
        hipLaunchKernelGGL(performFunctor, dim3(gridDim), dim3(blockDim), 0, stream,
                           getAccessor(), op, calcReadInd, calcWriteInd, elems_x);
#endif
    } else {
        /// ...CPU implementation
    }
}
```

Code Performance

Brookhaven National Laboratory

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

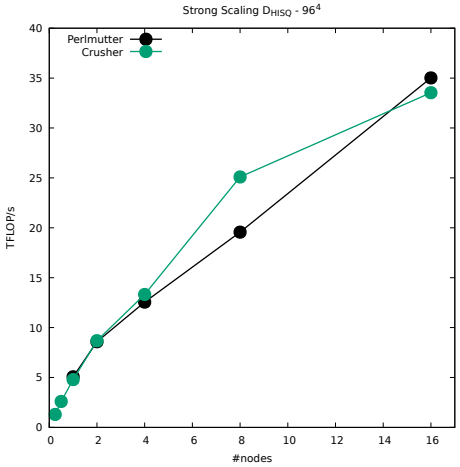
$$\not{D}\psi_x = \sum_{\mu=0}^{4} \left[\left(V_{x,\mu}\chi_{x+\hat{\mu}} - V^{\dagger}_{x-\hat{\mu},\mu}\chi_{x-\hat{\mu}} \right) + \left(W_{x,\mu}\chi_{x+3\hat{\mu}} - W^{\dagger}_{x-3\hat{\mu},\mu}\chi_{x-3\hat{\mu}} \right) \right]$$

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

- 1146 FLOP/site, 1560 byte/site $\rightarrow$ FLOP/byte $\sim$ 0.73. $\not{D}\psi_x$ computation is bandwidth bound!
- 16-point stencil with long legs: high inter-node bandwidth requirement compared to more "local" actions in order to achieve scaling.

Code Performance

- Matrix inversions (CG) dominate computational cost.
- Kernels are bandwidth bound: use link-compression when possible.
- $\not{D}$ achieves up to 1.36TB/s memory throughput on A100 (1.6TB/s bandwidth).
- Great on-node and good multi-node scaling on NVIDIA systems.
- First benchmarks on MI250x systems (Crusher, Lumi-G) show potential for optimizations.
- Next goal: Proper profiling & tune-up of HISQ RHMC on MI250x.



Hadron structure measurements using GPT

Brookhaven National Laboratory

- Goal: Development of measurement code for calculating qPDF, qTMD of hadrons on DWF gauge field configurations.
- Leverage existing expertise with DWF at BNL:
  - GRID[1] C++ library for lattice QCD calculations: highly optimized DWF routines, support for all relevant architectures: vectorized CPU code, GPU support with CUDA, HIP and SYCL back-ends.
  - GPT[2] (Grid Python Toolkit): offers simple python interface to most of GRIDs functionalities, modular solvers than can be combined easily, allows for rapid prototyping without sacrificing performance.

[1] `https://github.com/paboyle/Grid`
[2] `https://github.com/lehner/gpt`

Hadron structure measurements using GPT

Brookhaven National Laboratory

```
light_innerL_inverter = g.algorithms.inverter.preconditioned(
          g.qcd.fermion.preconditioner.eo1_ne(parity=g.odd),
          g.algorithms.inverter.sequence(
              g.algorithms.inverter.coarse_deflate(
                  eig[1],
                  eig[0],
                  eig[2],
                  block=400,
                  fine_block=4,
                  linear_combination_block=32,
              ),
              g.algorithms.inverter.split(
                  g.algorithms.inverter.cg({"eps": 1e-8, "maxiter": 200}),
                  mpi_split=g.default.get_ivec("--mpi_split", None, 4),
              ),
          ),
      )

light_exact_inverter = g.algorithms.inverter.defect_correcting(
    g.algorithms.inverter.mixed_precision(light_innerL_inverter, g.single, g.double),
          eps=1e-8,
          maxiter=12,
      )
```

Figure: Example of modular solver setup: eo-preconditioned, deflated, mixed-precision CG

Hadron structure measurements using GPT

Brookhaven National Laboratory

Current, early stage status:

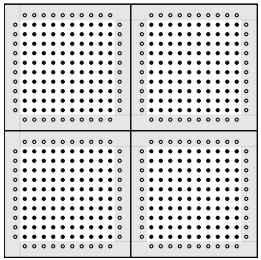
- Pion qPDF and distribution amplitude implemented.
- First production size tests of pion TMD wavefunction just started.
- Proton qPDF implemented.
- Proton qTMD work in progress.

DDHMC for DWF

Brookhaven National Laboratory

- Network performance cannot keep up with increase in on-node computing power in upcoming machines.
- Domain-decomposed HMC with single domain per node.

$$D = \begin{pmatrix} D_\Omega & D_\partial \\ D_{\bar\partial} & D_{\bar\Omega} \end{pmatrix}, \quad \det D = \det D_\Omega \det D_{\bar\Omega} \det \left\{ 1 - D_\Omega^{-1} D_\partial D_{\bar\Omega}^{-1} D_{\bar\partial} \right\}, \tag{1}$$



Interior, on-node: Ω, Exterior (gray): $\bar\Omega$

- Keep links in $\bar\Omega$ frozen during evolution, random translation between trajectories.
- Can implement boundary term as ratio and put on coarse timestep

$$\left\{ 1 - D_\Omega^{-1} D_\partial D_{\bar\Omega}^{-1} D_{\bar\partial} \right\} = \det D / (\det D_\Omega \det D_{\bar\Omega}). \tag{2}$$

Under active development, methods not set in stone!

Summary

Brookhaven National Laboratory

- SIMULATeQCD HISQ code shows good performance on A100 based systems (Perlmutter, Polaris, Booster,...).
- Completed HIP port and ran first large multi-node runs on MI250x (Crusher, Lumi-G), needs performance tune-up: We would appreciate discussions with SciDAC-5 members that have experience with rocprof and other AMD profiling tools!
- Started code development on hadron structure measurements using GPT and making quick progress towards first physics results.
- GPT offers many opportunities for trying new methods, solver combinations, algorithms, ... and we are excited to explore them.
- DDHMC and related techniques look very promising for scaling DWF gauge generation on upcoming systems like Aurora.