Porting the openQCD-FASTSUM code to GPUs using CUDA

Efficient simulations on GPU hardware,
24-27/10/2022, ETH Zürich

Felix Ziegler
in collaboration with B. Gursoy, B. Jäger, S. Ryan, J. Skullerud
27/10/2022

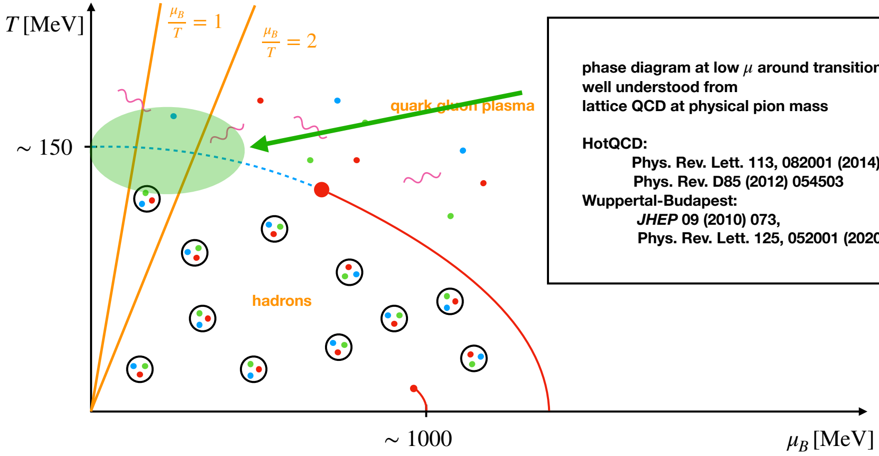
The University of Edinburgh

Overview

1. Motivation for porting openQCD to GPUs
2. openQCD software framework
3. State of the port
4. Conclusion and plans

Motivation for porting openQCD to GPUs

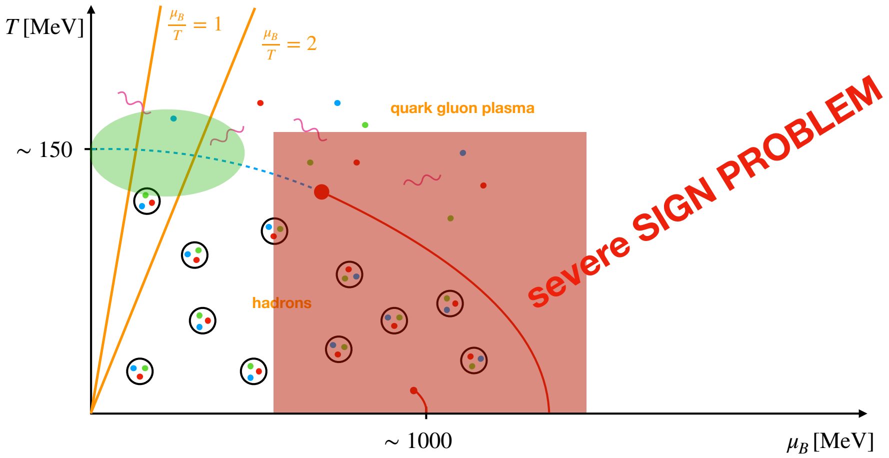
Status of the QCD phase diagram from lattice QCD



phase diagram at low μ around transition line
well understood from
lattice QCD at physical pion mass

HotQCD:
Phys. Rev. Lett. 113, 082001 (2014),
Phys. Rev. D85 (2012) 054503
Wuppertal-Budapest:
JHEP 09 (2010) 073,
Phys. Rev. Lett. 125, 052001 (2020)

Status of the QCD phase diagram from lattice QCD



Physics motivation

- **HIC Experiments:** Uncovering QCD phase diagram is top priority at LHC, RHIC, FAIR
- **Theory:** Make predictions from first principle $\rightarrow$ lattice QCD

- **Sign problem:** Euclidean action complex due to $\mu_B > 0$.
- **Computational challenge:** despite there exist algorithms to circumvent the sign problem, the Dirac matrix is severely ill-conditioned for moderate to large chemical potentials.

$$D_{xy}[U;T,\mu] = (4+m)\delta_{xy} - \frac{1}{2}\sum_{\nu=0}^{3}\left[\Gamma_\nu\, e^{\mu\delta_{\nu,0}}U_{x,\nu}\delta_{x+\hat{\nu},y} + \Gamma_{-\nu}\, e^{-\mu\delta_{\nu,0}}U^{-1}_{x-\hat{\nu},\nu}\delta_{x-\hat{\nu},y}\right] .$$

Note: $\mu = \mu_B/3$.

Fermion matrix at finite chemical potential

- There exists variety of algorithms to circumvent the sign problem.
- The **complex Langevin** (CL) method enables access to low temperatures.
- **Price to pay:** we must work in $SL(3,\mathbb{C})$ where lattice gauge fields are no longer unitary.
- reduced γ_5-symmetry in the Dirac operator

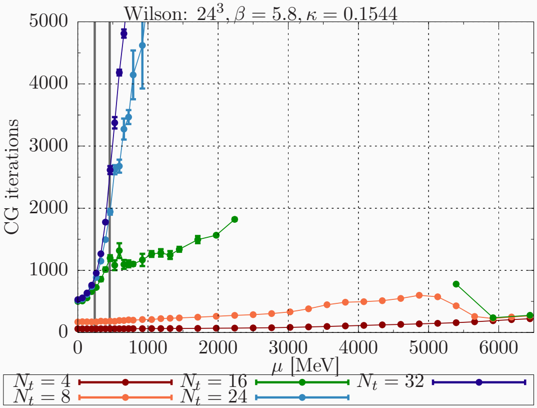
For $U \in SU(3)$

$$D[U;T,\mu]^\dagger = \gamma_5 D(-\mu^*)\gamma_5$$

$$\det D[U;T,-\mu^*] = \left(\det D[U;T,\mu]\right)^* \ ,$$

and slightly more involved for $U \in SL(3,\mathbb{C})$ where $U^\dagger \neq U^{-1}$.

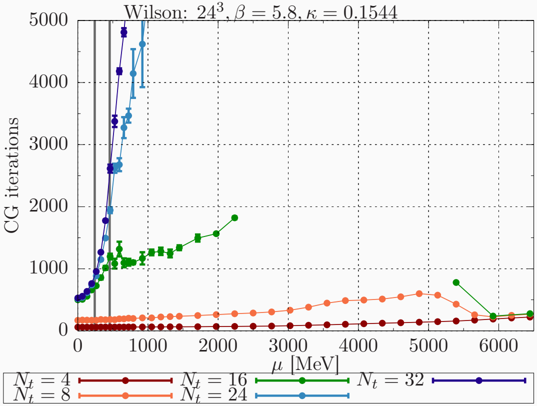
Performance of the eo-prec. mixed precision CG algorithm



Computed on a single node on Hawk (Stuttgart) of CPUs using our CL modified version of openQCD-1.6.

For the physics results, see arXiv:2203.13144 [hep-lat].

Performance of the eo-prec. mixed precision CG algorithm



Lowering temperature T and increasing μ leads to **prohibitively ill-conditioned Dirac matrix!**

color coding from red to blue: hot to cold

Our plan

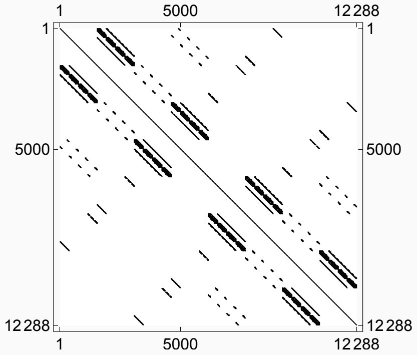
- Speed CG up on GPU accelerators $\Rightarrow$ port openQCD to GPUs with CUDA
- Our aim for physics results: 10 % error
- Gauge field generation time consuming part, analysis can be done on the fly (density, chiral condensate, Polyakov loop)
- **Inversion of the Dirac matrix is by far the most time consuming part (ca. 95 % of the time).**
- lattice sizes of up to 64×32^3
- target resource: 1 GPU or 1 node of 2-4 GPUs (e.g. nVidia A100)
- deflated solver not applicable when $\mu > 0$ $\Rightarrow$ focus on CG (for now)

openQCD software framework

openQCD

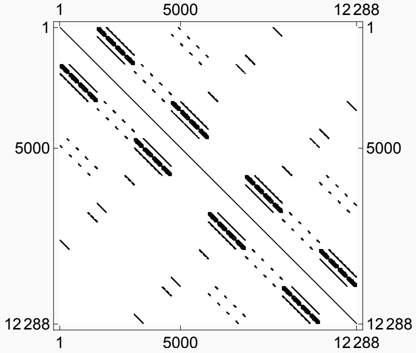
- MPI + openMP parallelized lattice QCD code written in C by M. Lüscher et. al.
- highly optimized for CPU architecture (e.g. AVX2 vector instructions etc.)
- performent solvers for Wilson fermions, in particular the deflated solver DFL+SAP+GCR
- Personal opinion: comparably well-documented
- specific to $N_c = 3$ and 4 space-time dimensions

- local gauge field is stored at the odd points and data layout is in even-odd ordering
- for more info see `https://luscher.web.cern.ch/openQCD/`



openQCD-FASTSUM

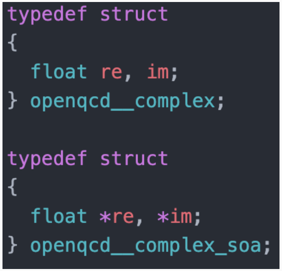
- extension of openQCD-1.6 including smearing and anisotropc actions, developed by J. Glesaaen and B. Jäger
- allows for openQCD to be compiled into a library
- lattice sizes and MPI layout can be specified at run time
- production code used by the FASTSUM collaboration, see e.g. Phys.Rev.D 105 (2022) 3, 034504
- On Gitlab: `https://fastsum.gitlab.io`



State of the port

Data layout and lattice geometry

- move from original openQCD arrays of structures (AoS) to structures of arrays (SoA) to allow for better memory access patterns and coalesced reads / writes.
- Format conversion done on the device.
- at the moment: use `iup` and `idn` look-up tables that are generated by the original openQCD code's `geometry.c` module. $\Rightarrow$ bottleneck as cache blocking is highly optimized for CPU but not for the GPU



```
typedef struct
{
  float re, im;
} openqcd__complex;

typedef struct
{
  float *re, *im;
} openqcd__complex_soa;
```

Status of the port

- Memory transfer by hand (cudaMalloc, cudaMemcpy)
- mixed-precision CG solver
  - Dirac operator in single and double precision
  - inner products and reductions
- single GPU done

Performance of the single precision Dirac operator

- lattice size: 32×24^3
- **CPU**: 90 GFLOPs (one node of 16 Intel Xeon E5-2680 using AVX2)
- **GPU**: 500 GFLOPs (single nVidia Quadro RTX 4000, 8 GB)

At the moment we achieve ca. 10% of the peak performance for the single precision Dirac operator on the GPU.

A ToDo list from the profiler

- performance is bound by memory transfer, threads run out of work because they wait for data to be loaded from memory
- increase arithmetic intensity (for example by inverting the gauge links on the fly instead of loading them from memory)
- make use of shared memory, data locality / reuse, tiling(?)
- optimize data layout replacing the original openQCD look-up tables for better coalesced reads and writes
- refactor stencil for the Dirac operator, see talks by K. Clark and P. Boyle

Conclusion and plans

Conclusion and perspectives

- eo-prec. mixed-precision CG + multi-shift CG implemented on GPU
- In HMC or CL based runs the computation of the fermionic force is outsourced to the accelerator.
- gitlab repo to be updated soon
- ToDo: performance optimization
- future steps: SAP and deflated solver

Thank you very much and if you like to collaborate please feel free to contact me.