ETH zürich

D-PHYS
HPCP

Accelerating the HMC in openQxD using QUDA

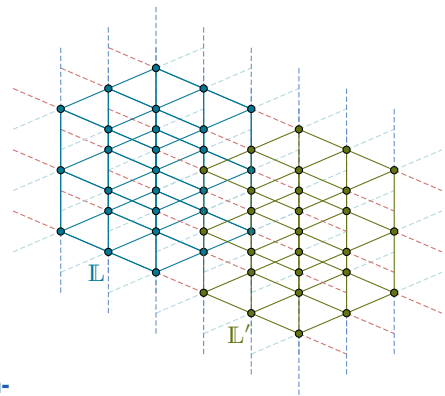
J. T. Kuhlmann
for the RC* collaboration

LATTICE conference 2026, Maryland

Motivation

Towards the physical point

- Within RC*:
  [L. Holan, Mon 14:20; E. Bäske, Mon 15:00; K. Kersic, Tue 16:10; P. Tavella, Tue 16:30; K. Phan, Wed 10:20]
  - towards 4-flavour improved Wilson fermions at physical point ($N_\mathrm{f} = 1+2+1, 1+1+1+1$)
  - implement $\mathrm{QCD} + \mathrm{QED}_C$ via orbifold construction
  - isospin breaking effects
- tune towards physical point
- fine, large volume lattices
- HMC is serial, no trivial parallelization
- datacentre shift towards GPU



$\Rightarrow$ computationally expensive ensembles with many configurations are needed!

$\Rightarrow$ shift towards GPU computations

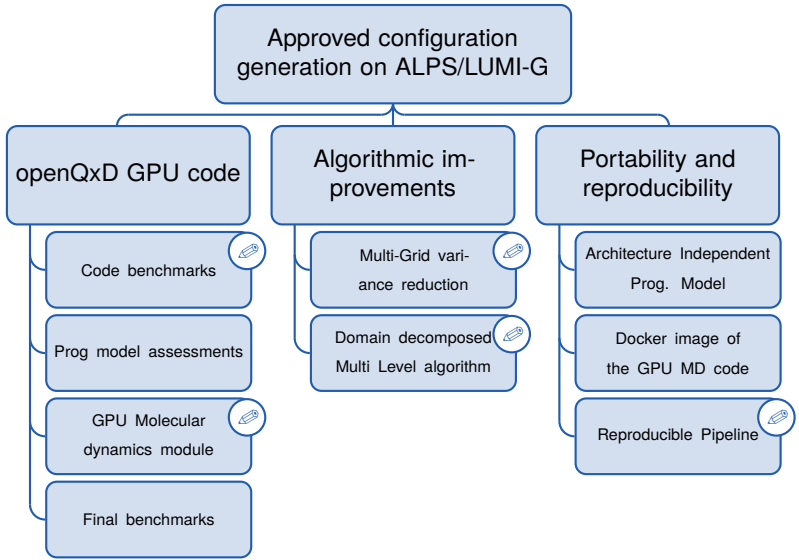
The story so far

Towards the physical point

C* periodic ensembles [https://rcstar.gitlab.io/ensembles/]

Ensemble	T	L_1	L_2	L_3	gauge group	β	α
A360a50b324	64	32	32	64	SU(3)$\times$ U(1)	3.24	0.05
A380a07b324	64	32	32	64	SU(3)$\times$ U(1)	3.24	0.007299
A400a00b324	64	32	32	64	SU(3)	3.24	—
A500a50b324	64	32	32	64	SU(3)$\times$ U(1)	3.24	0.05
B400a00b324	80	48	48	96	SU(3)	3.24	—
C380a50b324	96	48	48	96	SU(3)$\times$ U(1)	3.24	0.05
D300a00b324	128	64	64	128	SU(3)	3.24	—
D400a00b324	128	64	64	128	SU(3)	3.24	—
D300a00b343	128	64	64	128	SU(3)	3.43	—

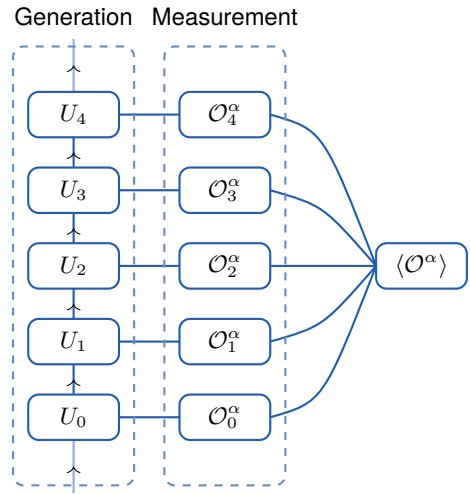
Activities


Approved configuration generation on ALPS/LUMI-G
openQxD GPU code
Code benchmarks
Prog model assessments
GPU Molecular dynamics module
Final benchmarks
Algorithmic improvements
Multi-Grid variance reduction
Domain decomposed Multi Level algorithm
Portability and reproducibility
Architecture Independent Prog. Model
Docker image of the GPU MD code
Reproducible Pipeline

OpenQxD GPU offloading

Interfacing QUDA

- Based on OpenQCD 1.6
  [https://luscher.web.cern.ch/luscher/openQCD]
- RC* implemented C* boundary conditions
  [1908.11673]
  - Latest release open source
    [https://gitlab.com/rcstar/openQxD/-/tags/1.1.0]
- Merged interface into QUDA
  [https://github.com/lattice/quda/pull/1414]
  - Easy integration into measurement code
  - High absolute performance gain in measurement
- **Current state:** measurement with QUDA
  - Started to offload D_{W} and D_{W}^{-1} in measurement
- **Next step: generation with GPU enabled HMC**



Using QUDA in the HMC

Challenges:

- Gauge fields change in every step of the MD integration
  - Setup of Multi Grid preconditioning
    - currently with each transfer of the gauge field
  - CPU/GPU communication + reordering with many MD steps

In this presentation:

- First timings for **partly offloaded** force and action for the HMC

Towards interfacing the HMC

1. Preparations: ✓
   - Benchmark current implementation
   - Understanding Multi-Grid refresh(s)
2. Minimally adapted version ($N_f = 2$, $\mu = 0$)
   - use QUDA solver ✓
   - test correctness ✓
   - benchmark ✓
3. Enable further functionality
   - $\mu \neq 0$
   - RHMC ($N_f = 3$)
   - chronological solver
4. Offload further kernels

Helpful resources:

- PhD thesis Roman Gruber [DOI:10.3929/ethz-c-000783366]
- openQCD/openQxD documentation
- QUDA implementation
- ETMc implementation

A small look into the code

What does the implementation look like?

action1

$$\mathcal{S}_{\mathrm{pf}} = (\phi, (D^\dagger D)^{-1}\phi)$$

$\phi \leftarrow \gamma_5\phi$	`mulg5`
$\psi \leftarrow D^{-1}\phi$	`solver`
$\phi \leftarrow \gamma_5\phi$	`mulg5`
$\mathcal{S} \leftarrow \lvert\psi\rvert^2$	`sum`

force1

$\phi \leftarrow \gamma_5\phi$	`mulg5`
$\psi \leftarrow D^{-1}\phi$	`solver`
$\phi \leftarrow \gamma_5\phi$	`mulg5`
$\psi \leftarrow \gamma_5\psi$	`mulg5`
$\chi \leftarrow D^{-1}\psi$	`solver`
$\psi \leftarrow D\chi$	`dirac`
$\psi \leftarrow \gamma_5\psi$	`mulg5`
$X_{\mu\nu} \leftarrow f(\psi, \chi, x)$	`add_sw`
$\pi \leftarrow \pi + X_{\mu\nu}$	
$X_\mu \leftarrow g(\psi, \chi, x)$	`add_hop`
$\pi \leftarrow \pi + X_\mu$	

Test setup

CLS D5d

$N_\mathrm{f} = 2$
periodic
$48 \times 24 \times 24 \times 24$
$\beta = 5.3$
$\kappa_\mathrm{u} = \kappa_\mathrm{d} = 0.136250$

Machines

Alps.Eiger

2 nodes
2x AMD Epyc 7742
96 (of 2x64) cores in use
256 GB DDR4
Solver: DFL SAP GCR

Alps.Daint

2 nodes
4x nVidia Grace
64 (of 288) cores in use
4x nVidia Hopper GPU
Solver: QUDA MG

Tests

Force/Action

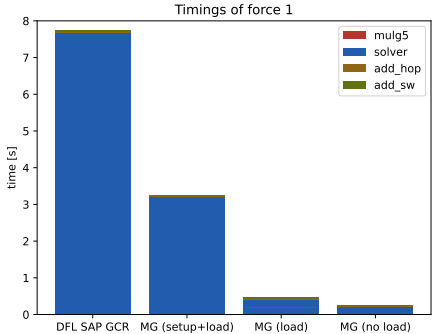
1x `force1` call on config
4x `action1` calls on min. SU(3) rotated configs

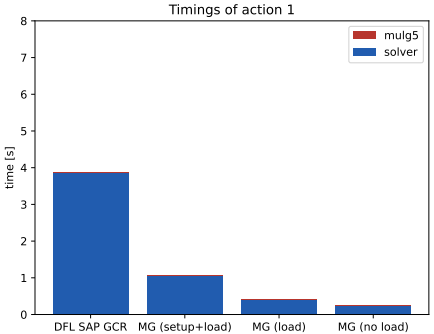
HMC trajectory

Time 1 trajectory, breakdown calculation of force and action

First results

Current state of the interface

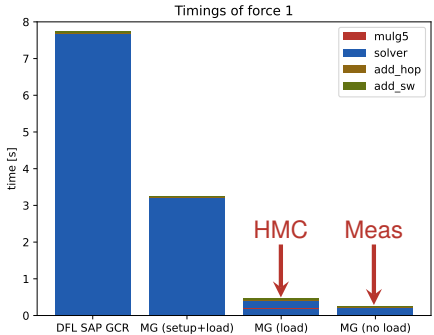


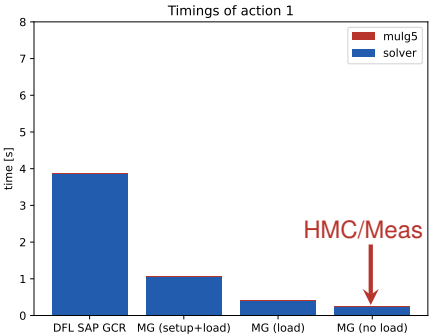


DFL SAP GCR: (CPU | 2x 96 cores)
QUDA Multi Grid: (GPU | 2x 64 cores | 8x GPU)

First results

Current state of the interface



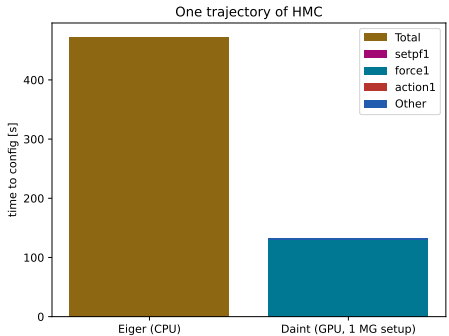


DFL SAP GCR: (CPU | 2x96 cores)
QUDA Multi Grid: (GPU | 2x 64 cores | 8x GPU)

First results
HMC trajectory

- Simple setup
- 2 level integrator
  - **Level 0:** Leap frog, 1 step, gauge force
  - **Level 1:** OMF4, 48 steps, fermion force
- solver tolerance 10^{-10}
- setup the MG solver only at trajectory start
- $48 \times 5 + 1 = 241$ force calls

>3.3x speed-up!



DFL SAP GCR: (CPU | 2x96 cores)
QUDA Multi Grid: (GPU | 2x 64 cores | 8x GPU)

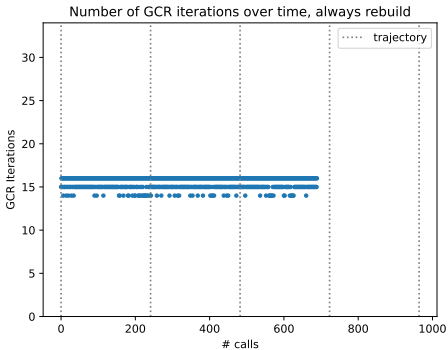
Conclusion & Outlook

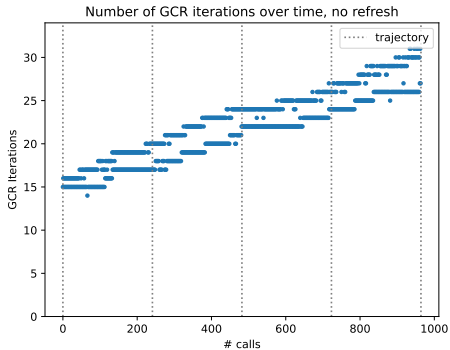
- Offloaded solver in measurement programs
- Started offloading intensive parts of the HMC
  - `force1`/`action1` as a first study
- Gained deeper understanding of implementation
  - Faster solver setup

- Refresh logic
  - Better understanding of the solver setup in the HMC context
  - Extend/reconfigure interface
- Re-enable OpenQCD solver features
  - RHMC ($N_f = 3$)
  - $\mu \neq 0$
  - chronological solver
- Offload more kernels (e.g. OpenMP)

Questions or Suggestions?

Backup


Number of GCR iterations over time, always rebuild
trajectory
GCR Iterations
calls
0
5
10
15
20
25
30
0
200
400
600
800
1000


Number of GCR iterations over time, no refresh
trajectory
GCR Iterations
calls
0
5
10
15
20
25
30
0
200
400
600
800
1000