Towards Lattice QCD+QED Simulations on GPUs

R. Gruber, A. Kozhevnikov, M. K. Marinkovic, T. Schulthess, R. Solca

RC* collaboration: Anian Altherr, Lucius Bushnaq, Isabel Campos-Plasencia, Marco Catillo, Alessandro Cotellucci, Madeleine Dale, Roman Gruber, Patrick Fritzsch, Javad Komijani, Jens Luecke, Marina Marinkovic, Sofie Martins, Agostino Patella, Joao Pinto Barros, Nazario Tantalo and Paola Tavella

July 4, 2023

Overview

QCD on the lattice

Yang-Mills Lagrangian (4D, with P and T invariance):

$$\mathcal{L} = -\frac{1}{4} F^a_{\mu\nu} F^{\mu\nu,a} + \bar{\psi}\left(i\not{D} - m\right)\psi$$

$$\begin{aligned}
\not{D} &= \gamma^\mu D_\mu \\
F^a_{\mu\nu} &= \partial_\mu A^a_\nu(x) - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu(x) A^c_\nu(x) \\
D_\mu &= \partial_\mu + igA^a_\mu(x) T^a,
\end{aligned}$$

with action $\mathcal{S} = \int d^4x \mathcal{L}$. In path integral formalism, we have the Minkowski expectation value

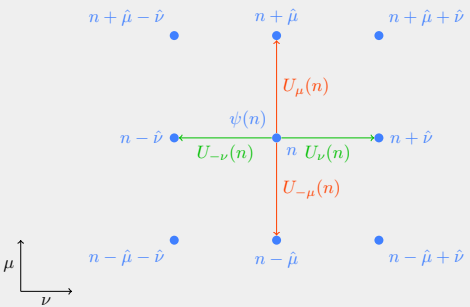
$$\begin{aligned}
\langle \mathcal{O}[\psi, \bar{\psi}, A] \rangle &= \lim_{t \to \infty(1-i\epsilon)} \frac{1}{Z} \int \mathcal{D}\psi \mathcal{D}\bar{\psi} \mathcal{D}A \mathcal{O}[\psi, \bar{\psi}, A] e^{iS[\psi, \bar{\psi}, A]} \\
Z &= \int \mathcal{D}\psi \mathcal{D}\bar{\psi} \mathcal{D}A e^{iS[\psi, \bar{\psi}, A]}.
\end{aligned}$$

4D lattice

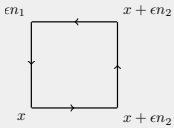
$$\Lambda = \{n = (n_0, n_1, n_2, n_3) \mid n_i \in \{0, 1, \ldots, L_i - 1\}, i \in \{0, 1, 2, 3\}\}.$$

$$A^b_\mu(x) = A^b_\mu(an) \longrightarrow aA^b_\mu(n),$$
$$\psi(x) = \psi(an) \longrightarrow a^{3/2}\psi(n)$$
$$U(x, x + \epsilon\hat{\mu}) = U(an, an + a\hat{\mu}) \longrightarrow U_\mu(n).$$

- lattice extents $L_i \in \mathbb{N}$
- lattice constant a
- compensator field (link variable) $U(x, y)$ ($\frac{\partial U}{\partial x^\mu} = igA^a_\mu(x)T^a$)



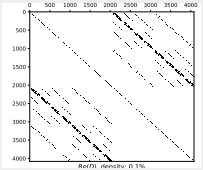
The full discretized Yang-Mills action is



$$\mathcal{S} = \mathcal{S}_G + \mathcal{S}_F,$$

$$\mathcal{S}_G = \frac{1}{g^2} \sum_{n \in \Lambda} \sum_{\mu,\nu} Re\ tr \left[\delta_{\mu\nu} \cdot id - \hat{U}_{\mu\nu}(n) \right],$$

$$\mathcal{S}_F = a^4 \sum_{n \in \Lambda} \bar{\psi}(n) \underbrace{\left[\sum_{\mu} \frac{\gamma^\mu}{2} (D_{+\mu} + D_{-\mu}) + m \right]}_{D} \psi(n)$$



- $D_{\pm\mu}$ are forward- and backward covariant derivatives
- D is the Dirac operator
- Large sparse matrix, dim $d = 12V$, (eg. 64^4 lattice: $d \approx 10^8$, naively 500 petabytes)

Rotating to the Minkowski action $e^{i\mathcal{S}} \xrightarrow{\text{WR}} e^{-\mathcal{S}_E}$ we obtain the Euclidean expectation value

$$\langle \mathcal{O}[\psi, \bar{\psi}, A] \rangle = \lim_{\tau \to \infty} \frac{1}{Z} \int \mathcal{D}\psi \mathcal{D}\bar{\psi} \mathcal{D}A \mathcal{O}[\psi, \bar{\psi}, A] e^{-S[\psi, \bar{\psi}, A]}$$
$$Z = \int \mathcal{D}A \det(D) e^{-S_G[A]}.$$

Interpret the exponential as probability density, $P(U) = Z^{-1} e^{-S}$,

$$\langle \mathcal{O}[U] \rangle = \lim_{\tau \to \infty} \frac{1}{Z} \int \mathcal{D}U e^{-S[U]} \mathcal{O}[U] \longrightarrow \sum_U P(U) \mathcal{O}[U],$$
$$\langle \mathcal{O}[U] \rangle \approx \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}[U_i].$$

Lattice QFT on the CPU

- large sparse linear operator (complex dimension $O(10^8)$)
- Markov chain alters using Hybrid Monte Carlo [1]
- operator needs to be inverted repeatedly (solve $D\psi = \eta$)
- most used compute intensive kernel: **sparse matrix-vector multiplication** (SpMV), $D\psi$
- sparse linear algebra is **memory bound** (speed limited by memory bandwidth, not compute power)
- using supercomputers ($O(100)$ nodes with $O(100)$ cores) $\rightarrow$ high parallelizability
- large problem is decomposed into many smaller problems

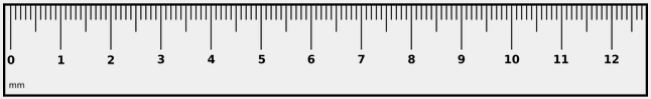
- based on openQCD 1.6 written by Martin Lüscher and Stefan Schäfer [2]
- implementing dynamical QCD+QED simulations
- addition of C* boundary conditions
- available via https://gitlab.com/rcstar/openQxD
- MPI 1.2, ISO C90
- currently no threading, GPU support

Data types

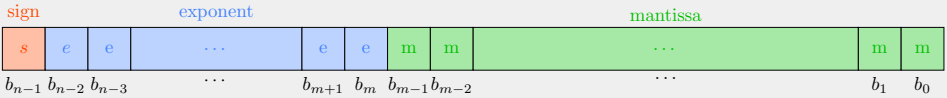
Floating point formats

Floating-point formats				
name	s	e	m	machine ϵ (2^{-m})
binary64 [3]	1	11	52	2.2×10^{-16}
binary32	1	8	23	1.2×10^{-7}
binary16	1	5	10	9.8×10^{-4}
bfloat16 [4]	1	8	7	7.8×10^{-3}
tensorfloat32[5]	1	8	10	9.8×10^{-4}
posit32 [6]	1	es=2		7.5×10^{-9}
posit16	1	es=1		2.4×10^{-4}
posit8	1	es=0		3.1×10^{-2}

Table: Commonly used floating-point formats, where s is the number of sign bits, e the number of exponent bits and m the number of mantissa bits.



Floats [3]



- $f = (-1)^s \cdot M \cdot 2^E$
- two representations of 0
- $-\infty$ and $+\infty$
- lots of NaNs
- may over- or underflow
- complicated comparison operations
- existence of subnormals
- many cases $\Longrightarrow$ large on-chip FPU $\Longrightarrow$ small operations per watt count

Examples:

`0 0...0 0...0` $= +0$

`1 0...0 0...0` $= -0$

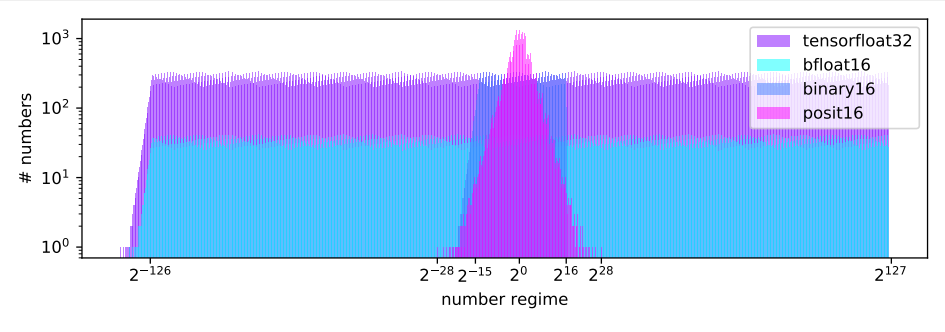
`0 1...1 0...0` $= +\infty$

`1 1...1 0...0` $= -\infty$

Posits [6]

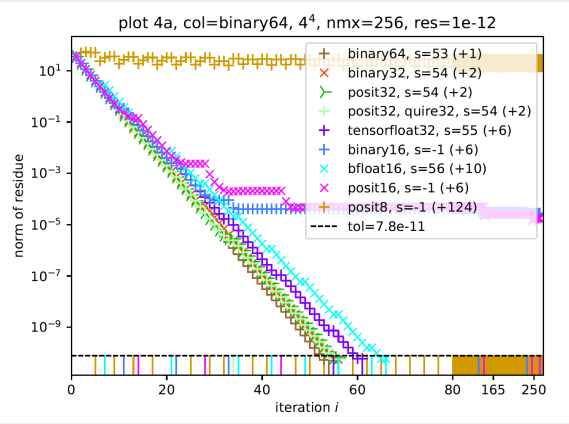
- $p = (-1)^s \cdot (2^{2^{es}})^k \cdot M \cdot 2^E$
- no NaNs, but calculation is interrupted
- only one representation of 0
- every number has a reciprocal
- reciprocal of 0 is $\pm\infty$
- may not over- or underflow
- simple comparison (bits identical $\equiv$ numbers equal)
- Posit processing unit takes less circuitry than an FPU $\implies$ higher operations per watt count

Floats vs. Posits



Conjugate gradient

Conjugate Gradient - Convergence analysis



Conclusions:

- binary16 or bfloat16 are sufficient
- use binary64 in reduction variables (norms)
  $\left\|\vec{x}\right\| = \sqrt{\sum_i x_i^2}$
- implement a general mixed precision solver

- solver-kernel consists of norms, scalar products, applications of D, axpys
  $\Longrightarrow$ memory bound operations
- Residue: $\rho_i = \eta - D\psi_i$
- binary16: 5 exponent, 10 mantissa bits
- bfloat16: 8 exponent, 7 mantissa bits

SAP-GCR ALGORITHM

- Decompose the lattice into many blocks
- Only nearest-neighbour interaction $\Longrightarrow$ blocks of same color are independent
  - solve gray blocks
  - update boundaries
  - solve white blocks
  - update boundaries

  $\Longrightarrow$ one Schwarz-cycle (alternate between black and white blocks)
- Preconditioning phase:
  - n_{cy} Schwarz-cycles
  - n_{mr} MR-steps on each blocked problem

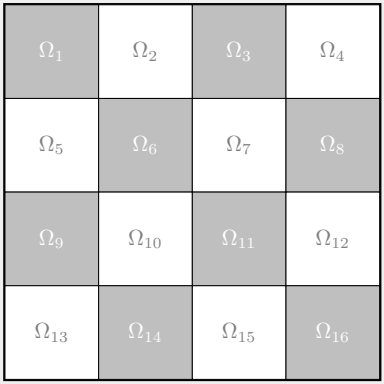


Figure: A two dimensional example of a decomposition of a lattice $\Omega = \bigcup_{i=1}^{n} \Omega_i$ into $n = 16$ domains Ω_i [7].

Measurements

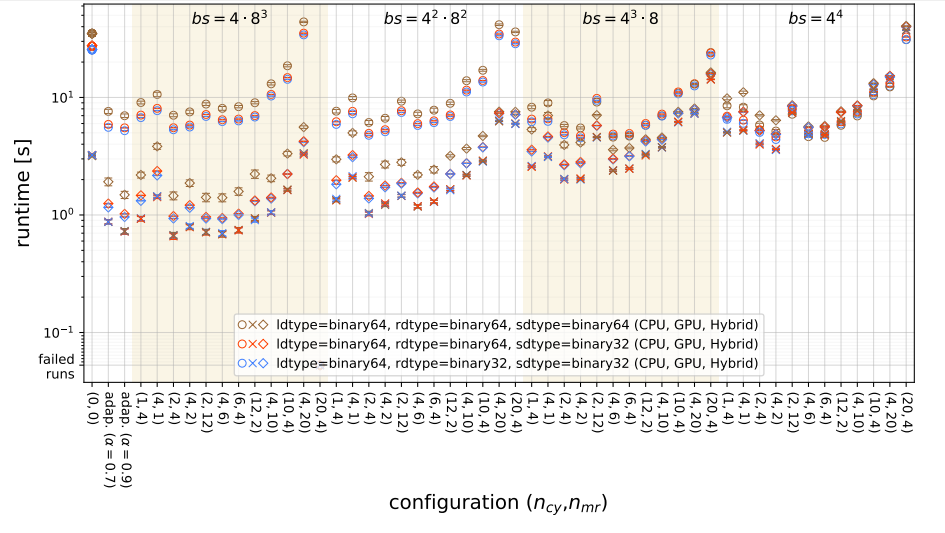


Figure: Time measurements for the SAP_GCR kernel on different matrices and configurations. The measurements were conducted on an AMD EPYC 7742 CPU @ 2.25GHz with 512 GB memory and an NVIDIA A100 (via SXM4) GPU with 40 GB memory.

SAP+GCR - Conclusions

- Heavy and non-intuitive run-time dependence on input parameters (n_{cy}, n_{mr})
- Existence of exceptional configurations with extremely long run-times, non-convergence
- Adaptive variant (tries to find the optimal config every GCR-iteration anew, avoiding exceptional configs, suitable long running simulations, where D vastly changes its condition)
  - upper bound: $n_{cy} = 20$ and $n_{mr} = 20$
  - lower bound: $n_{cy} = 1$ and $n_{mr} = 4$
  - after every Schwarz cycle, exit if residual satisfies

$$\|\rho_i\| \geq \|\rho_{i-1}\|$$

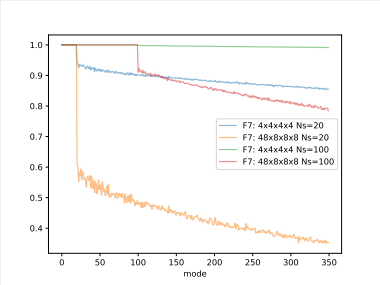
  - after every MR-step, exit if **blocked** residual satisfies

$$\|\rho_i\| \geq \alpha\|\rho_{i-1}\| \text{ where } \alpha \in \{0.7, 0.9\}$$

SAP-GCR solver based on inexact deflation

DFL SAP GCR

- Dirac operator has a property called local coherence [8]
  $\rightarrow$ mode modes look "the same" locally
  $\rightarrow$ few low modes are enough to build large deflation subspaces by block projection
- Second stage of preconditioning
- D restricted to the deflation subspace: little Dirac operator A
- In every outer GCR step we solve the little equation



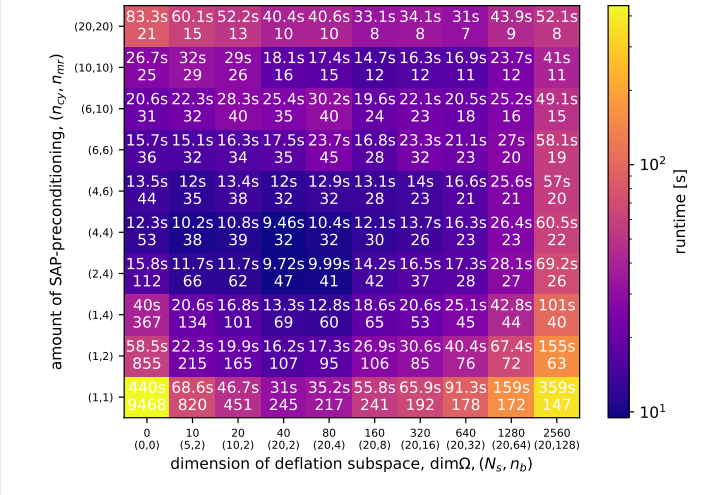


Figure: Time measurements for the DFL_SAP_GCR kernel on different matrices and configurations. The measurements were conducted on an Intel(R) 6130 @ 2.10GHz with 1.5 TB memory and an NVIDIA V100 (via PCIe) GPU with 16 GB memory.

Summary and conclusions

Summary and conclusions

- Convergence analysis of CG using different data types
  - Mixed precision solver desirable, specially employing 16-bit data types (i.e. bfloat16 on NVIDIA A100)
  - Mild benefits from posits as opposed to floats
  - These insights apply to the other solvers as well
- SAP-GCR solver
  - Sensitive to choice of input parameters
  - Adaptive variant might be beneficial
- Deflated SAP-GCR solver
  - Non-trivial interplay between deflation and SAP

Outlook and current state

Outlook and current state

Motivated by the above analysis

- Successfully coupled openQxD to QUDA [9]
  - Gauge field transfer
  - Multi-GPU support
- Still missing steps to tackle:
  - Fermion field transfer
  - C* boundaries
  - QCD+QED

Thank you for listening!

We acknowledge access to Piz Daint at the Swiss National Supercomputing Centre, Switzerland under the ETHZ's share with the project IDs s299 and c21.

References I

[1] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo", Physics Letters B **195**, 216–222 (1987).

[2] *openQCD*, `https://luscher.web.cern.ch/luscher/openQCD/`, Accessed: 2023-06-23.

[3] Institute of Electrical and Electronics Engineers. Computer Society. Standards Committee and Stevenson, David, *IEEE standard for binary floating-point arithmetic*, (IEEE, 1985).

[4] S. Wang and P. Kanwar, "Bfloat16: the secret to high performance on cloud TPUs", Google Cloud Blog (2019).

[5] R. Krashinsky, O. Giroux, S. Jones, N. Stam, and S. Ramaswamy, "NVIDIA ampere architecture in-depth", NVIDIA blog: https://devblogs. nvidia. com/nvidia-ampere-architecture-in-depth (2020).

[6] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: posit arithmetic", Supercomputing Frontiers and Innovations **4**, 71–86 (2017),

[7] M. Lüscher, "Lattice QCD and the schwarz alternating procedure", Journal of High Energy Physics **2003**, 052–052 (2003),

References II

[8] M. Lüscher, "Local coherence and deflation of the low quark modes in lattice QCD", Journal of High Energy Physics **2007**, 081–081 (2007),

[9] M. A. Clark, R. Babich, K. Barros, R. C. Brower, and C. Rebbi, "Solving lattice qcd systems of equations using mixed precision solvers on gpus", Computer Physics Communications **181**, 1517–1528 (2010).