Performance optimizations for porting the openQ*D package to GPUs

Roman Gruber
Supervisor: Prof. Thomas C. Schulthess (CSCS, ETH)
Co-Supervisors: Prof. Marina Krstić Marinković (ETH, RC*), Dr. Raffaele Solcà (CSCS), Dr. Anton Kozhevnikov (CSCS)

July 30, 2021

RC* collaboration: Lucius Bushnaq, Isabel Campos-Plasencia, Marco Catillo, Alessandro Cotellucci, Madeleine Dale, Patrick Fritzsch, Jens Luecke, Marina Marinkovic, Agostino Patella and Nazario Tantalo

Table of contents

Introduction/Motivation

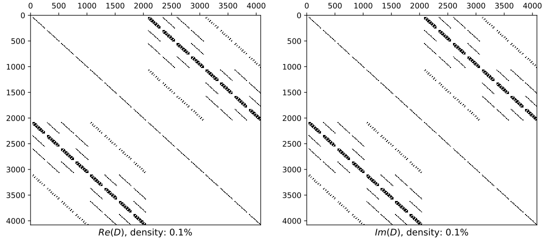
Other talks involving OpenQ*D [Campos et al. arXiv:1908.11673]:

- Jens Luecke: An update on $QCD + QED$ simulations with $C*$ boundary conditions (July 29)
- Madeleine Dale: Baryon masses from full $QCD + QED_C$ simulations (July 29)
- Lucius Bushnaq: Implementing noise reduction techniques into the OpenQxD package (July 30)

Introduction/Motivation

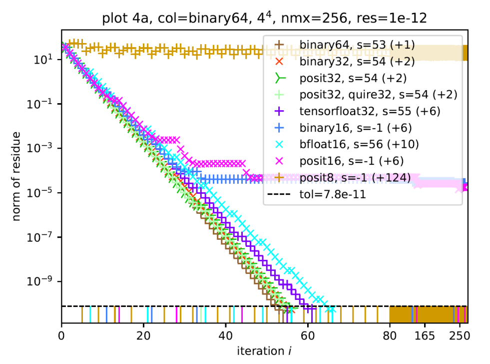
$$D\psi = \eta \tag{1}$$

- source spinor $\eta \in \mathbb{C}^n$, given
- Dirac operator D, sparse, $n \times n$-matrix, complex, given
- Wilson Dirac Operator with a clover term
- Lattice QCD: $n = 4 \cdot 3 \cdot V$, with $V = L_0 L_1 L_2 L_3$
  Example: 96×64^3-lattice $n = 3 \cdot 10^8$



Conjugate Gradient with different real number formats

Conjugate Gradient - Convergence analysis



Conclusions:

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

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

SAP-preconditioned GCR algorithm

SAP-preconditioned GCR algorithm (SAP+GCR)

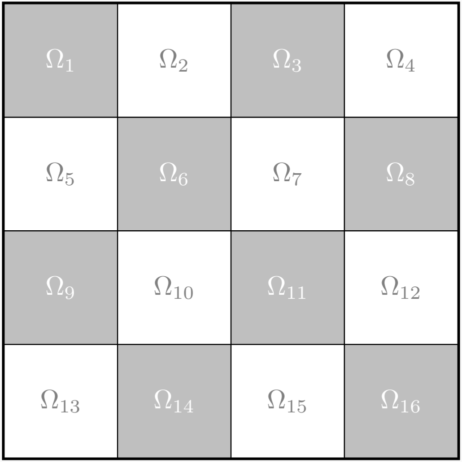


Figure 1: A $d = 2$ dimensional example of a decomposition of a lattice Ω into domains named Ω_i. [Lüscher arXiv:hep-lat/0304007]

SAP-preconditioned GCR algorithm (SAP+GCR)

- Only nearest-neighbour interaction $\implies$ blocks of same color are independent
  - solve gray blocks
  - update boundaries
  - solve white blocks
  - update boundaries

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

SAP-preconditioned GCR algorithm (SAP+GCR)

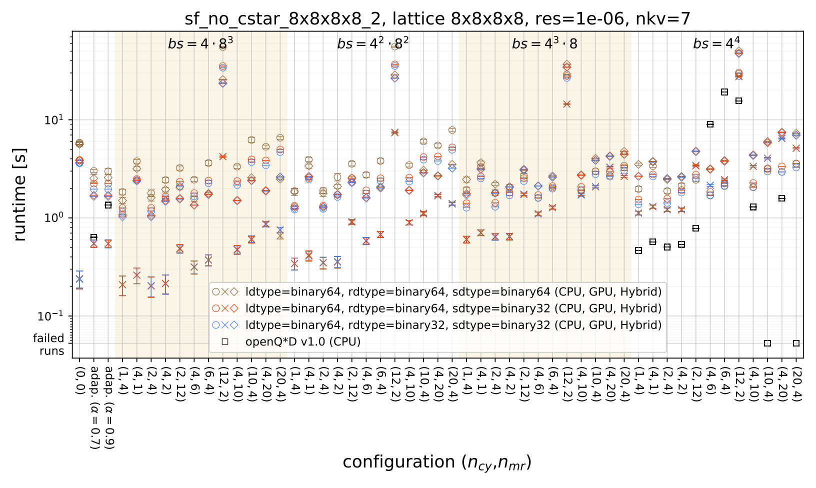


Figure 2: Machine: Intel(R) 6130 @ 2.10GHz with 1.5 TB memory, an NVIDIA V100 (via PCIe) GPU with 16 GB memory.

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-preconditioned GCR algorithm (SAP+GCR)

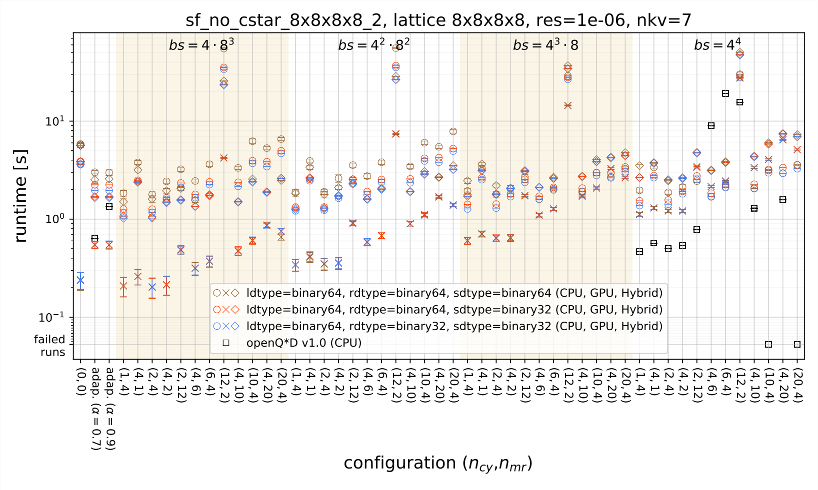


Figure 3: Machine: Intel(R) 6130 @ 2.10GHz with 1.5 TB memory, an NVIDIA V100 (via PCIe) GPU with 16 GB memory.

Summary

Summary

- Dirac-operator in reduced precision, mixed precision solvers, expected speedup $\leq 2x$
- GPU gives significant performance increase, 2x - 10x
- Hybrid solution gives significant performance increase, up to 8x
- Adaptive SAP+GCR, more versatile, suitable long running simulations (no fixed choice of n_{cy}, n_{mr})
- Future: heterogeneous computing, speedup unknown
- Thanks 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.

Backup slides

Floating-point format limits

Floating-point format limits				
name	f_{max}	f_{min}	f_{smin}	sign.
binary64	1.8×10^{308}	2.2×10^{-308}	4.9×10^{-324}	≤ 15.9
binary32	3.4×10^{38}	1.2×10^{-38}	1.4×10^{-45}	≤ 7.2
binary16	6.6×10^{4}	6.1×10^{-5}	6.0×10^{-8}	≤ 3.3
bfloat16	3.4×10^{38}	1.2×10^{-38}	9.2×10^{-41}	≤ 2.4
tensorfloat32	3.4×10^{38}	1.2×10^{-38}	1.1×10^{-41}	≤ 7.2
binary24	1.8×10^{19}	2.2×10^{-19}	3.3×10^{-24}	≤ 5.1
binary128	1.2×10^{4932}	3.4×10^{-4932}	6.5×10^{-4966}	≤ 34
binary256	$1.6 \times 10^{78,913}$	$1 \times 10^{-78,912}$	$1 \times 10^{-78,983}$	≤ 71.3

Table 1: Summary of highest representable numbers, minimal subnormal and non-subnormal representable numbers above 0 in any IEEE 754 floating-point format together with their approximated precision in decimal.

SAP+GCR - ill-conditioned system

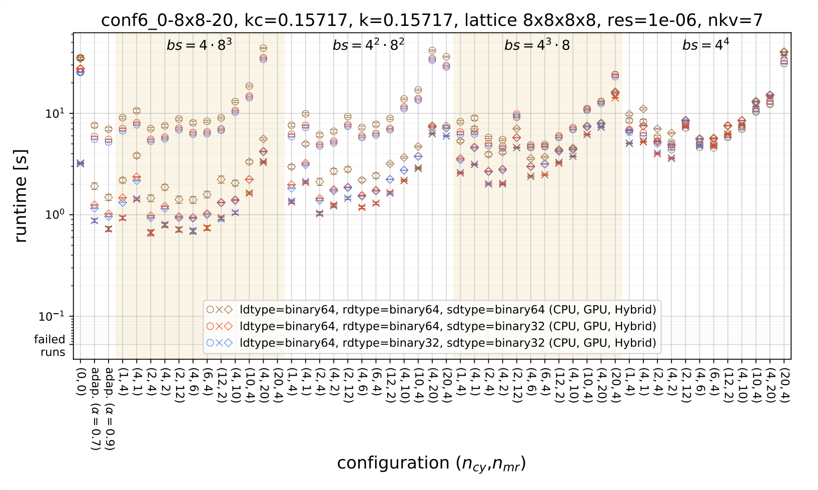


Figure 4: Time measurements for the SAP_GCR kernel on different configurations. Node: AMD EPYC 7742 CPU @ 2.25GHz, 512 GB memory, NVIDIA A100 (via SXM4) GPU with 40 GB memory.

Floating point formats

Floating-point formats				
name	s	e	m	comment
binary64	1	11	52	double precision, IEEE 754
binary32	1	8	23	single precision, IEEE 754
binary16	1	5	10	half precision, IEEE 754
bfloat16	1	8	7	Googles Brain Float
tensorfloat32	1	8	10	NVIDIAs TensorFloat-32
binary24	1	7	16	AMDs fp24
binary128	1	15	112	IEEE 754
binary256	1	19	236	IEEE 754

Table 2: 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.

Dirac-matrix

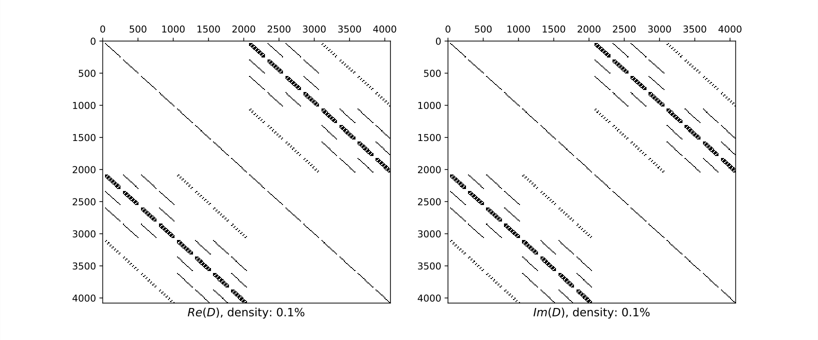


Figure 5: An example plot of a Dirac-matrix of an 8^4-lattice with SF-boundary conditions. Every pixel consists of 192×192 real numbers. If the average over that numbers is non-zero the pixel is drawn black, else the pixel is drawn white. The density gives the overall percentage of non-zero values.

Scheme of Dirac-matrix

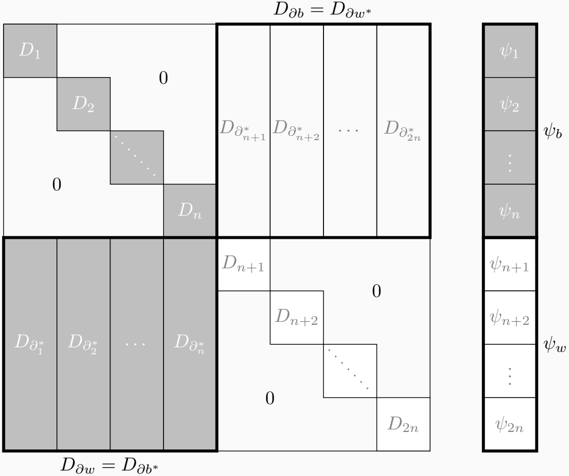


Figure 6: Schematic of the Dirac-operator in terms of a large sparse matrix.

Arithmetic intensity

The **arithmetic intensity** I is the ratio of the **work** W and the **memory traffic** T appearing in a considered piece of code,

$$I = \frac{W}{T}.$$

The work W needs to be given as number of floating-point operations and the memory traffic T in terms of stored and loaded bytes. The unit of arithmetic intensity I is then floating-point operations per byte and depends on the data type of the involved quantities.