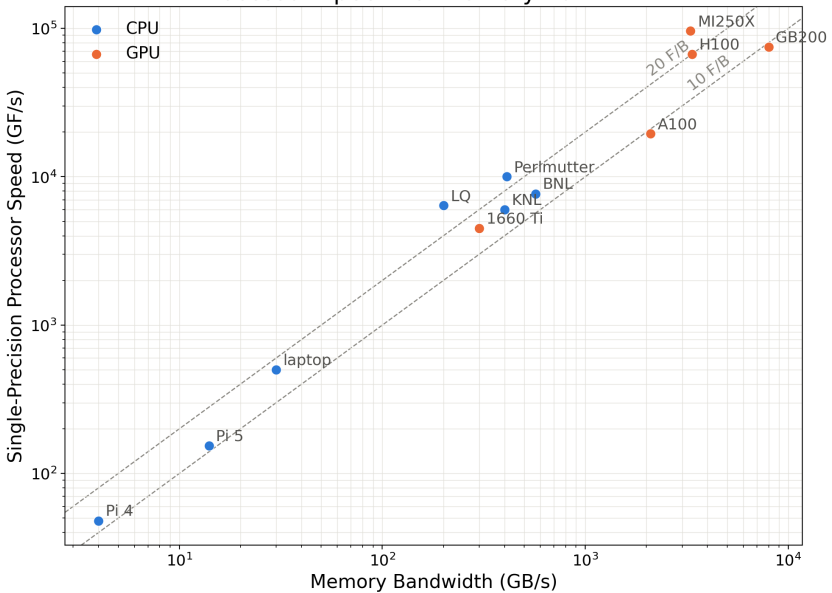
Toward Compute-Bound Lattice-QCD Software

Anthony V. Grebe

July 27, 2026


Processor Speed vs. Memory Bandwidth
CPU
GPU
MI250X
H100
GB200
20 F/B
10 F/B
A100
Perlmutter
BNL
LQ
KNL
1660 Ti
laptop
Pi 5
Pi 4
Single-Precision Processor Speed (GF/s)
Memory Bandwidth (GB/s)
10^5
10^4
10^3
10^2
10^1
10^2
10^3
10^4

The Memory Roofline

- Saturating processor means performing 10–20 (single-precision) flops per byte of memory
- Arithmetic intensity (AI) of algorithm determines maximum flop-to-byte ratio possible
  - Vector addition: 1 F / 12 B = 0.08 F/B
  - Complex dot product: 8 F / 16 B = 0.5 F/B
  - Complex matrix multiply: $(8N^3)$ F / $(24N^2)$ B = $N/3$ F/B ($\to \infty$ as $N \to \infty$)

Memory Roofline of Lattice QCD

- Historically computed under three main assumptions:
  1. Most of compute spent in propagator solves
  2. Most solve time spent applying fine-grid $\not{D}$
  3. Different right-hand sides solved separately
- Conclusion: AI $\sim$ 1–2, optimal performance is $\sim$10% of processor roofline

Computing $\not{D}$

- Flops/site: 1920 (CPC 181, 1517 (0911.3191))
  - Wilson term: (complex 3×3 matvec + γ projection) $\times$ 2 spins $\times$ 8 neighbors = 1368 F
  - Clover term: two 6×6 complex matvecs = 552 F
- Bytes/site: 768–1824 (depending on cache reuse)
  - Clover term: 288 B
  - Gauge links: 288–576 B
  - Fermions: 96 B (out), 96–864 B (in)
- Total AI: 1–2.5

Computing $\not{D}$ (N RHS)

- Flops/site: 1920**N**
  - Wilson term: (complex 3×3 matvec + γ projection) $\times$ 2 spins $\times$ 8 neighbors = 1368**N** F
  - Clover term: two 6×6 complex matvecs = 552**N** F
- Bytes/site: $(192\mathbf{N} + 576)$–$(960\mathbf{N} + 864)$ (depending on cache reuse)
  - Clover term: 288 B
  - Gauge links: 288–576 B
  - Fermions: 96**N** B (out), (96–864)**N** B (in)
- Total AI: 1–2.5

Cache Reuse

- Computing $\not{D}$ requires 9 fermion sites (x + all neighbors)
- Each fermion site feeds 9 $\not{D}$
- Perfect reuse: read fermion once
- Cache size $\ll$ lattice size $\rightarrow$ perfect reuse impossible
- Try to maximize reuse ($\sim$ 2–3 full reads instead of 9)
- Requires blocking (or space-filling curve) to maximize reuse

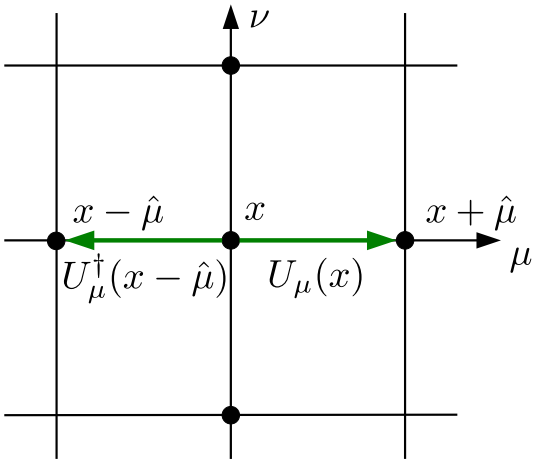


Figure credit: LNCS 10962 (1811.00893)

Multigrid

- Multigrid cycle:
  - Restrictor R to coarse grid
    - Built from M near-null vectors
  - Coarse solve of $D_C = RDP$
  - Prolongator $P = R^\dagger$
  - Fine-grid smoother
- Wrapped in outer solver (GCR, FGMRES)
- R, P are $2M \times N$ per site, D_C is $2M \times 2M$

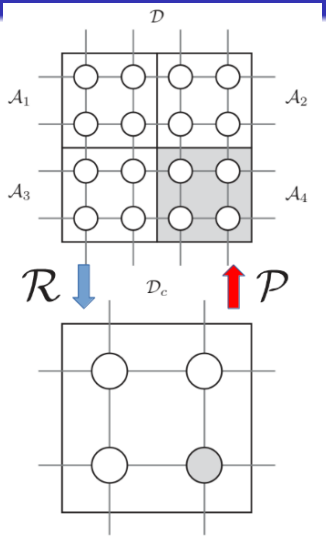


Figure credit: adapted from PRD 94, 114509 (1610.02370)

Multigrid

- Multigrid cycle:
  - Restrictor R to coarse grid
    - Built from M near-null vectors
  - Coarse solve of $D_C = RDP$
  - Prolongator $P = R^\dagger$
  - Fine-grid smoother
- Wrapped in outer solver (GCR, FGMRES)
- R, P are $2M \times N$ per site, D_C is $2M \times 2M$
- Recursive: coarse solve preconditioned by inner-layer MG

Image credit: Wikimedia Commons

Multigrid

- Coarse operator: $2M \times 2M$ matrix per site
- Coarse fermion: $2M \times N$ per site
- Arithmetic intensity: $32M^2N/32M(M+N) = MN/(M+N)$
  - Cache reuse + hermiticity can reduce further
- For $M = 32$,

N	1	16	32	∞
AI	0.97	10.7	16	32

- MG benefits from multi-RHS even more than fine $\not{D}$ (K. Clark, Lattice 2024)

Domain Decomposition

- MG recursion designed to remove low (IR) modes
- Post-smoothing removes residual high (UV) modes
- Simplest idea: repeated $\not{D}$ applications
- Domain decomposition: focus on UV modes contained to MG outer block

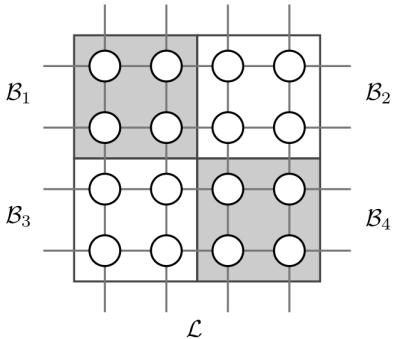


Figure credit: M. Rottmann, PhD thesis

Schwarz Alternating Procedure (SAP)

- (Approximately) solve within small block
- Boundary contributions ignored during this solve
- After intra-block solve, update boundaries
- Proceed to next block, iterate
- Red-black checkerboarding: parallelize over half the blocks on lattice

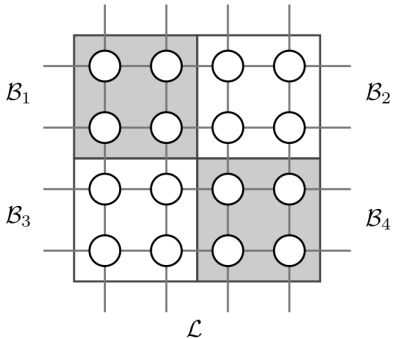


Figure credit: M. Rottmann, PhD thesis

Domain Decomposition

- Originally intended as algorithmic improvement (fewer flops)
- Major benefit: per-block solve is **cache-local**
  - Typically done iteratively
  - All iterations after first cost zero additional memory reads
  - AI increases by factor of inner iter count, independent of N

Domain Decomposition

- Originally intended as algorithmic improvement (fewer flops)
- Major benefit: per-block solve is **cache-local**
  - Typically done iteratively
  - All iterations after first cost zero additional memory reads
  - AI increases by factor of inner iter count, independent of N
- Can perform at each level of MG recursion
  - Intermediate blocks typically 2^4
  - **Precompute dense matrix inverse** $\rightarrow$ no iteration
  - Application is **dense matmul**: AI $\approx 2N \sim 30$

DDαAMG

- Written by Matthias Rottmann to accompany PhD thesis
- Most development $\geq$ 10 years ago
  - Single-RHS only
  - Targets pre-AVX CPUs (SSE only)
- Primary goal was algorithmic (reducing flops) rather than maximizing flops/sec
- Several ports over the years to modern machines

SISC 36, 4, A1581 (1303.1377); 1307.6101; PhD thesis (Rottmann, Wuppertal, 2016)

qc Library

- Development started in 2021 for HOPE Collaboration computations on KNLs
- Many design decisions tried and rejected over years
- Near-full rewrite beginning last year with goal of porting DDαAMG
  - Also significant inspiration from QUDA (CPC 181, 1517 (0911.3191))
- Multi-RHS designed in from the start
- Development accelerated in past couple months due to LLMs

qc Library

- Mixed precision (double outer, single inner)
- Vectorize over RHS
- Tiled traversal of lattice during Dslash (improve cache reuse)
- Prefetch next direction (hide latency)
- ASM (not intrinsics) for inner kernels (much easier with LLMs)
  - Inspired by Grid (hep-lat/1512.03487)
- Emphasis on reducing memory overhead (larger local volumes)
- `#define` parameters (L_x, M, N) at compile time
  - Compiler optimizations (e.g. index computation)
  - Recompilation cheap ($\sim$ 2 min)

Performance

- Tested on AMD EPYC cluster at BNL
  - Co-designed by Peter Boyle
- $64^3 \times 128$ near-physical configuration with 16 RHS
- 5-level MG solve
  - 4^4 outer block
  - Three 2^4 inner blocks
  - $2^3 \times 4$ exact bottom solve
- Achieves 3110 GF/s (>**40% of processor roofline**)

```
+-------------------------------------------------------------------+
| mixed-precision solve profiling (double FGMRES + single K-cycle)  |
+-------------------------------------------------------------------+
| FGMRES:  16 outer iters,   192 coarse bottom-solve iters          |
| precond K-cycle applies:   16   mg_cycle calls (all levels):  300 |
| fine A=D applies:   0+3   total solve wall time:   111.3234 s     |
+-------------------------------------------------------------------+
| category                        time(s)   %total    GFlop/s       |
+-------------------------------------------------------------------+
| fine Dslash restart r (double)    4.3597     3.9%     709.31      |
| outer FGMRES (Arnoldi/GS)         9.8742     8.9%     341.88      |
| prolongation / restriction        7.9223     7.1%    4970.30      |
| fine-grid SAP (single)           62.9845    56.6%    2735.95      |
| coarse-grid Dslash (single)       3.2771     2.9%    5850.59      |
| coarsest Dslash (Dc_lvl)          0.8608     0.8%    4894.88      |
| coarse-grid SAP (single)         18.1994    16.3%    5571.53      |
| coarse bottom dense apply         0.0965     0.1%    1068.07      |
| K-cycle accel (FGCR GS)           1.1727     1.1%     242.11      |
| coarse e/o Schur setup            0.2875     0.3%     100.84      |
| coarse transfers (d>=1)           1.2809     1.2%    2233.10      |
| V-cycle vector ops (single)       0.1464     0.1%      66.56      |
| other (malloc/fork/skew)          0.8614     0.8%         --      |
+-------------------------------------------------------------------+
| total                           111.3234   100.0%                  |
+-------------------------------------------------------------------+
| instrumented flops:         3.462e+14  (all flop-bearing rows)    |
|   flop / lattice site:      1.032e+07                             |
|   aggregate flop rate:        3110.20 GFlop/s                     |
+-------------------------------------------------------------------+
```

Coarse Grid Performance

- Coarse grid on BNL runs at ~ 5.5 TF/s ($> 70\%$ of roofline)
- Achieves ~ 10 F/B (STREAM bandwidth: ~ 570 GB/s)
- Speed gated by processor flop rate
  - Requires ASM kernels to saturate processor
  - Dense matrix inverse runs at ~ 20 F/B on memory-constrained systems
- Inner iteration of fine SAP (after initial load) also runs at $\sim 70\%$ of roofline

Ongoing Work

- GPU port is work in progress
  - Currently achieves $\sim$ 10–11 TF/s in double-half precision on single A100
  - Exploring double-quarter using INT8 tensor cores
  - $\sim$ 80% intra-node scaling efficiency, $\sim$ 70% 2-node efficiency on LQ2
- HMC with multi-RHS using Clark-Kennedy many-pseudofermions idea (PRL 98, 051601 (hep-lat/0608015)) (with Jo Moscoso)
- Wavefunction smearing

Nuclear Contractions

- Contractions in multi-nucleon systems are $O(1)$ fraction of cost (comparable to inversions)
- Typically not memory bound
- Many algorithmic improvements here:
  - Maximize reuse of common elements
    - Nucleon blocks
    - Color-contracted diquarks
  - FFT for higher-momentum shells
  - Restructure into matmul whenever possible
  - Hard-coded ASM (generated by LLMs)

Summary

- Modern developments in LQCD make traditional memory rooflines less binding
- Lattice-wide fine $\not{D}$ no longer bulk of compute time
- LLMs have greatly accelerated code development, prototyping
- 50% efficiency or higher may be achievable
  - More science per node-hr, \$, tCO_2
- Many thanks to Peter Boyle, Kate Clark, Will Detmold, Bálint Joó, Gurtej Kanwar, Issaku Kanamori, Jo Moscoso, Andrew Pochinsky and many others for useful conversations and USQCD (LQ, BNL, Jlab), MIT, Academia Sinica, and DiCOS for compute time