New Computational Trends in Lattice Gauge Theory

Peter Boyle (BNL)
pboyle@bnl.gov

- Lattice Gauge theory
- Electronic and hardware trends: what can we say about the future?
- Algorithmic response and trends
- Software response and trends

arXiv:2204.00039 "Lattice QCD and the Computational Frontier"
arXiv:2202.05838 "Applications of Machine Learning to Lattice Quantum Field Theory"
arXiv:1904.09725 "Status and Future Perspectives for Lattice Gauge Theory Calculations to the Exascale and Beyond"

Lattice Gauge Theory

- Numerical Euclidean path integral in discretised 'femto-box' large $t \leftrightarrow$ hadronic ground state physics
- Wilson: preserves SU(N) gauge symmetry with $U_\mu(x) = e^{iagA_\mu(x)}$ connecting lattice sites
- Partial derivatives replaced by finite difference

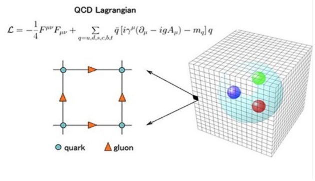
$$(\partial_\mu - igA_\mu(x))\psi(x) \to \frac{U_\mu(x)\psi(x+a\hat{\mu}) - U^\dagger_\mu(x-a)\psi(x-a\hat{\mu})}{2a}.$$

- Importance sample partition function: 10^{10} degrees of freedom.
  Auxiliary momentum (π) and pseudofermion (ϕ) integrals.
  Fermion determinant included stochastically via gaussian integral

$$Z = \int d\pi \int d\phi \int dU \quad e^{-\pi^2/2} e^{-S_G[U]} e^{-\phi^*(D^\dagger D)^{-1}\phi}.$$

- Compute correlation functions

$$\langle \mathcal{O} \rangle = \frac{1}{Z}\int_U e^{-S[U]}\mathcal{O}(U)dU \to \frac{1}{N}\sum_i \mathcal{O}(U_i)$$


QCD Lagrangian
$\mathcal{L} = -\frac{1}{4}F^{\mu\nu}F_{\mu\nu} + \sum_{q=u,d,s,c,b,t} \bar{q}\,[i\gamma^\mu(\partial_\mu - igA_\mu) - m_q]\,q$
quark
gluon

2205.15373, 2204.00039, 2203.15810, 2203.10998
Scientific need for at least 10x current exascale computers in Lattice QCD
$128^3 \times 512 \times 16$ lattices for direct B simulation

- g-2; hadronic vacuum polarisation and light by light
- weak matrix elements and flavor physics
- hadronic form factors, structure, PDFs
- CP violation in B and Kaon sector
- rare decay amplitudes

Algorithms

Phase-1: serially dependent gauge sampling

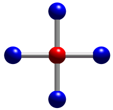
- Metropolis-Hastings algorithms: Markov Chain Monte Carlo sampling arbitrary probability distributions
- Other key developers:
  M. Rosenbluth, A. Rosenbluth, E. Teller and A. Teller.
- Hybrid Monte Carlo (HMC)

Phase-2: trivially parallelisable correlation function measurement

- Dirac solvers:
  - Krylov space methods ↔ polynomial approximation

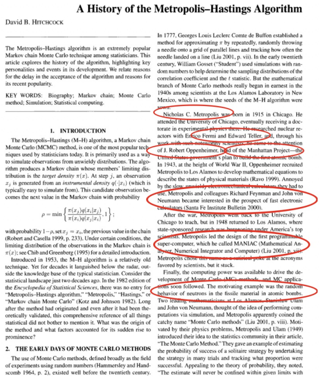
    $$\mathscr{P}(\nabla^2) = c_n p^{2n} \sim \frac{1}{p^2}$$

    In a background gauge field eigenmodes are not plane waves: approximate Green's function as polynomial of discrete Dirac operator.
  - newer covariant multigrid methods



Stencil for a nearest neighbour finite difference in 2D

Arianna Wright Rosenbluth was an American physicist who contributed to the development of the Metropolis–Hastings algorithm. She wrote the first full implementation of the Markov chain Monte Carlo method.



A History of the Metropolis–Hastings Algorithm

David B. Hitchcock

The Metropolis–Hastings algorithm is an extremely popular Markov chain Monte Carlo technique among statisticians. This article explores the history of the algorithm, highlighting key personalities and events in its development. We relate reasons for the delay in the acceptance of the algorithm and reasons for its recent popularity.

KEY WORDS: Biography; Markov chain; Monte Carlo method; Simulation; Statistical computing.

1. INTRODUCTION

The Metropolis–Hastings (M–H) algorithm, a Markov chain Monte Carlo (MCMC) method, is one of the most popular techniques used by statisticians today. It is primarily used as a way to simulate observations from unwieldy distributions. The algorithm produces a Markov chain whose members' limiting distribution is the *target density* $\pi(x)$. At step j, an observation x_j is generated from an *instrumental density* $q(\cdot|x_i)$ (which is typically easy to simulate from). This candidate observation becomes the next value in the Markov chain with probability

$$\rho = \min\left\{\frac{\pi(x_j)q(x_i|x_j)}{\pi(x_i)q(x_j|x_i)}, 1\right\};$$

with probability $1-\rho$, set $x_j = x_i$, the previous value in the chain (Robert and Casella 1999, p. 233). Under certain conditions, the limiting distribution of the observations in the Markov chain is $\pi(x)$; see Chib and Greenberg (1995) for a detailed introduction.

Introduced in 1953, the M–H algorithm is a relatively old technique. Yet for decades it languished below the radar, outside the knowledge base of the typical statistician. Consider the statistical landscape just two decades ago. In the 1982 edition of the *Encyclopedia of Statistical Sciences*, there was no entry for "Metropolis–Hastings algorithm," "Metropolis," "Hastings," or "Markov chain Monte Carlo" (Kotz and Johnson 1982). Long after the method had originated and even after it had been theoretically validated, this comprehensive reference of all things statistical did not bother to mention it. What was the origin of the method and what factors accounted for its sudden rise to prominence?

2. THE EARLY DAYS OF MONTE CARLO METHODS

The use of Monte Carlo methods, defined broadly as the field of experiments using random numbers (Hammersley and Handscomb 1964, p. 2), existed well before the twentieth century. In 1777, Georges Louis Leclerc Comte de Buffon established a method for approximating π by repeatedly, randomly throwing a needle onto a grid of parallel lines and tracking how often the needle landed on a line (Liu 2001, p. vii). In the early twentieth century, William Gosset ("Student") used simulations with random numbers to help determine the sampling distributions of the correlation coefficient and the t statistic. But the mathematical branch of Monte Carlo methods really began in earnest in the 1940s among scientists at the Los Alamos Laboratory in New Mexico, which is where the seeds of the M–H algorithm were sown.

Nicholas C. Metropolis was born in 1915 in Chicago. He attended the University of Chicago, eventually receiving a doctorate in experimental physics there. He researched nuclear reactors with Enrico Fermi and Edward Teller, and, through his work with such noteworthy scientists, he came to the attention of J. Robert Oppenheimer, head of the Manhattan Project—the United States government's plan to build the first atomic bomb. In 1943, at the height of World War II, Oppenheimer recruited Metropolis to Los Alamos to develop mathematical equations to describe the states of physical materials (Ravo 1999). Annoyed by the slow, unwieldy electromechanical calculators they had to use, Metropolis and colleagues Richard Feynman and John von Neumann became interested in the prospect of fast electronic calculators (Santa Fe Institute Bulletin 2000).

After the war, Metropolis went back to the University of Chicago to teach, but in 1948 returned to Los Alamos, where state-sponsored research was burgeoning under America's top scientists. Metropolis led the design of the first programmable super-computer, which he called MANIAC (Mathematical Analyzer, Numerical Integrator and Computer) (Liu 2001, p. vii). Metropolis chose this name as a satirical poke at the acronyms favored by scientists, but it stuck.

Finally, the computing power was available to drive the development of Monte Carlo (MC) methods, and MC applications soon followed. The motivating example was the random behavior of neutrons in the fissile material in atomic bombs. Two leading mathematicians at Los Alamos, Stanislaw Ulam and John von Neumann, thought of the idea of performing computations via simulation, and Metropolis apparently coined the catchy name "Monte Carlo methods" (Liu 2001, p. viii). Motivated by their physics problems, Metropolis and Ulam (1949) introduced their idea to the statistics community in their article, "The Monte Carlo Method." They gave an example of estimating the probability of success of a solitaire strategy by undertaking the strategy in many trials and tracking what proportion were successful. Appealing to the theory of probability, they noted, "The estimate will never be confined within given limits with

Historical DOE/Manhattan project connection

Electronic and hardware trends: what can we say now about the future?

Location	System	Interconnect (GB/s) per node (X+R)	Floating point performance (GF/s) per node	Memory Bandwidth (GB/s) per node	Year	System peak (PF/s)	FP / Interconnect	FP / Memory	Memory / Interconnect
LLNL	BlueGene/L	2.1	5.6	5.5	2004	0.58	2.7	1.0	2.6
ANL	BlueGene/P	5.1	13.6	13.6	2008	0.56	2.7	1.0	2.7
ANL	BlueGene/Q	40	205	42.6	2012	20	5.1	4.8	1.1
ORNL	Titan	9.6	1445	250	2012	27	150.5	5.8	26.0
NERSC	Edison	32	460	100	2013	2	14.4	4.6	3.1
NERSC	Cori/KNL	32	3050	450	2016	28	95.3	6.8	14.1
ORNL	Summit	50	42000	5400	2018	194	840.0	7.8	108.0
RIKEN	Fugaku	70	3072	1024	2021	488	43.9	3.0	14.6
NERSC	Perlmutter/GPU	200	38800	6220	2022	58	194.0	6.2	31.1
ORNL	Frontier	200	181200	12800	2022	>1630	906.0	14.2	64.0

- All DOE Exascale computing is GPU accelerated
- Huge gains in floating point
  not matched by gains in memory (14x) and interconnect (300x)
- Machines increasingly better suited for dense matrices and machine learning
- Lots of diversity and difficulty: ECP and SciDAC essential
  - Systems with AMD, Intel, Nvidia GPUs
  - Systems with CPU cores
  - HIP, SYCL, CUDA and conventional programming
  - Host memory, GPU memory
- Why so complicated??
- What can we actually say about short and mid-term future??

Argonne
Aurora
The Argonne Leadership Computing Facility's exascale system will be used to dramatically advance scientific discovery and innovation.

Forthcoming systems will increase floating point performance dramatically, but not interconnect.

- Lattice gauge theory algorithms for gauge field sampling *must* change to exploit.
- Lattice gauge theory correlation function calculations can run brilliantly

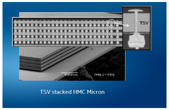
Why? You canny change the laws of physics

- Wire delay: time to charge a $L \times \pi r^2$ rod of metal depends on resistance & capacitance (RC time constant)†

$$RC \sim 2\rho\varepsilon \frac{L^2}{r^2} / \log(r_0/r) \sim \mathrm{const} \times \frac{L^2}{r^2},$$

$$\boxed{\text{Aspect ratio dictates wire delay - does not change when rod is shrunk}}$$

- Long thin wires are slow, short broad wires are fast
- 2.5D = chips edge to edge; 3D = vertical stacking : lots of short broad wires !
- Partitioned memory of modern HPC/GPU servers is dictated by electronics
  - On package memory, and advanced 3D packaging are here to stay.
  - Pools of fast small memory + pools of large slow memory.
    Data must be physically close to computational elements.
  - Future may bring additional layers (e.g. non-volatile)
    Soon: fast memory, slow host memory, SLOW host memory



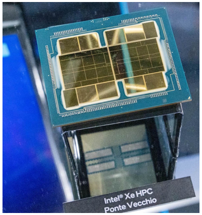
3D chipstacking enables many more wires and less energy per bit
Logical limit is silicon cubes even for computational logic to reduce wire delay
Cooling is a practical issue; TSV's conduct.

†: $\rho \equiv$ metal resistivity, $\varepsilon \equiv$ dielectric permittivity. $r_0 \ll L$ length scale small enough that potential well approximated by long rod approximation

Use transistors more effectively post-Moore?

Why GPU's?

- Simpler arithmetic units tailored to array processing increase floating point density
- Constraining: consecutive loop iterations should perform same operation on consecutive elements of data
- Less constrained than increasing SIMD vector lengths in CPUs
  - ORNL/Summit - Nvidia 6x V100 GPU + IBM Power CPU
  - NERSC/Perlmutter - Nvidia 4x A100 GPU + AMD x86 CPU
  - ORNL/Frontier - AMD 4x MI250-X GPU + AMD x86 CPU
  - ANL/Aurora (2023) - Intel 6x Ponte Vecchio GPU + Xeon
- As Moore's law approaches the atom barrier: better use of transistors could increase efficiency.
  - We have up to 100 billion of transistors in modern chips
  - 256MB of SRAM cache consumes 14Bn transistors but does no computation
- Near term: continued innovation
  - Nvidia Ampere → Hopper,
  - Intel Ponte Vecchio → Rialto Bridge (ISC 2022)
  - AMD → MI-300
- Nvidia, AMD, Intel competition is intense.
  Drives significant gains for Lattice gauge theory
- MPI messaging between nodes is a growing bottleneck that requires algorithm response



Intel Ponte Vecchio multi-die package

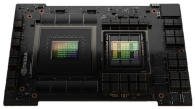


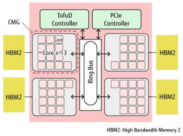
Figure 4. Grace Hopper Superchip

Nvidia Grace (ARM CPU) + H100 integrated device

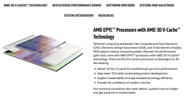
Memory innovation for CPUs

GPU's use aggressive memory systems. e.g. 2.5D HBM memory or GDDR

- Seeing memory system innovation spreading beyond GPUs
- New multi-die packaging options: provide many short wires (Fujitsu/ARM, AMD x86, Intel x86, Nvidia Grace)
- Multicore CPU's are acquiring novel memories - what is the right scheme?
- Integration of CPU and GPU
  - Nvidia Grace/Hopper
  - AMD APU's: LLNL/El Capitan AMD MI300 combines GPU and CPU
- Dataflow / reprogrammable hardware (FPGA/spatial acceleration) may further increase floating point density
- Machine learning specific acceleration



Fujitsu/ARM Fugaku A64FX



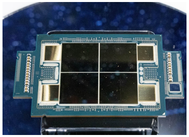
AMD EPYC™ Processors with AMD 3D V-Cache™ Technology

Technical computing workloads² like Computational Fluid Dynamics (CFD), Electronic Design Automation (EDA), and Finite Element Analysis (FEA) require intense computing power. Discover the performance gains that come with AMD EPYC™ processors with AMD 3D V-Cache™ technology. These are the first server processors to leverage true 3D die stacking.

- Deliver 3X the L3 cache for breakthrough per-core performance³
- Help lower TCO while accelerating product development
- Support sustainability through exceptional energy efficiency
- Provide the confidence of modern security

Run technical simulations like never before, quicken time-to-insight, and get products to market faster.

AMD vcache 3D chipstack



Intel Sapphire Rapids + HBMe (Vision 2022)

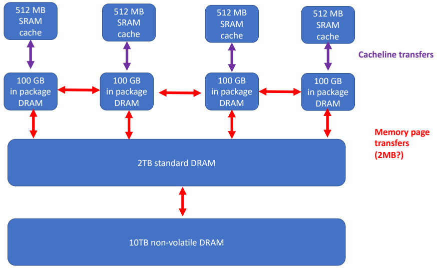
Programming models

Programming interfaces: syntactical details can be wrapped and hidden

- No fundamental reasons for “native” HIP / SYCL / CUDA 'mayhem'
- Expect standardisation of interfaces
    - OpenMP target for acceleration
    - Parallel STL /C++17

Memory models: fundamental semantic differences cannot be hidden

- HIP, SYCL and CUDA introduce virtual memory page based migration
- Even if physically partitioned, single virtual address space simplifies
- Performance penalty if pages are only 4KB ⇒ need bigger pages.
- DOE/ASCR is in a position to dictate more flexibility and good performance for virtual memory
- Expect will eventually include: non-volatile, dram, in-package and on-chip cache.



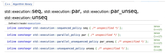
NVIDIA CUDA

SYCL

AMD
HIP Programming Guide v4.5

OpenMP

Standard C++


C++ / Algorithm library

std::execution::seq, std::execution::par, std::execution::par_unseq, std::execution::unseq

Defined in header <execution>

```
inline constexpr std::execution::sequenced_policy seq { /* unspecified */ };                          (since C++17)
inline constexpr std::execution::parallel_policy par { /* unspecified */ };                            (since C++17)
inline constexpr std::execution::parallel_unsequenced_policy par_unseq { /* unspecified */ };          (since C++17)
inline constexpr std::execution::unsequenced_policy unseq { /* unspecified */ };                       (since C++20)
```

Software impact

- Dealing with all this brings productivity challenges to scientists
- Field is heavily reliant on QUDA and Grid for performance on GPUs.
- Abstract the APIs in a thin layer, capture loop bodies in “device lambda functions” through macros
  - Prudent to expect to use “native” compiler tools for each system: e.g. HIP, SYCL, CUDA
  - ECP project effort to port QUDA to HIP and SYCL almost complete
  - Prudent to expect to add OpenMP and Parallel C++ in near future. These may simplify and replace.
- Explicit data motion vs. unified memory choice vs. software caching
- Continuous effort and support to target new programming models and optimise on emerging architectures is required.
- ECP & SciDAC are enormously important. Humanpower AND early access to prepare for forthcoming hardware.

Algorithmic impact

Finer simulations must cover a growing range of length scales. Current algorithms display criticla slowing down. USQCD is actively researching the following multiscale algorithm directions:

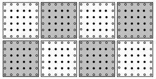
- Domain wall and staggered fermion multigrid
  - Multigrid has been transformational for the Wilson fermion discretisation
  - Accelerates observable calculations with minimal memory footprint
  - Similar acceleration is required for domain wall and staggered fermions
- Gauge invariant fourier acceleration to address critical slowing down as continuum limit is taken
  - Riemannian Manifold HMC
  - field transformation HMC
- Domain decomposition rational Hybrid Monte Carlo (next slide)

Why domain decomposition

https://arxiv.org/pdf/2203.17119.pdf

Divide space into red and black hypercubes: Dirac operator may be written as

$$D = \begin{pmatrix} D_\Omega & D_\partial \\ D_{\bar\partial} & D_{\bar\Omega} \end{pmatrix}.$$



This can be LDU factorised: the factorised determinant is:

$$\det D = \det D_\Omega \det D_{\bar\Omega} \det\left\{1 - D_\Omega^{-1} D_\partial D_{\bar\Omega}^{-1} D_{\bar\partial}\right\},$$

The first two factors are *local* and last is non-local.

- Updating these on different timesteps in HMC $\Rightarrow$ cost can be predominantly local.
- Avoids communication limits on future computers
- Size domains to computing nodes - tunes algorithm to islands of high performance created by multi-GPU computers.
- "fix-up" term can be bounded: if only evolve gauge links separated from boundary
- Generalised to odd flavours (2203.17119)

Summary

- Scientific need for at least 10x current exaflop computers in lattice QCD
- Modern electronics industry is raising bandwidths by strapping chips together intelligently
- Optimal solution is unclear: recommended for elegant solutions to data placement and motion
  - e.g. improved virtual memory paging as the best method for hierarchical memory
- Current proliferation of APIs is unsustainable and standards should take over
- Using internal abstraction layers helps cope, needs continued software development support
- Massive opportunities in computing for lattice gauge theory, but also signficant challenges
- New algorithms are required to handle multiscale physics
  AND to match physics locality to supercomputer locality
- Continuous support (ECP, SciDAC) is essential to realise the potential
- CompF2 made recommendations on training, career paths, software support and lab-university joint positions.