A random walk in lattice fields and extreme computing

Bálint Joó
Jefferson Lab, Newport News, VA, USA

Graduate Student Seminar
Wed, Dec 17, 2014
Why Lattice QCD?

- Asymptotic Freedom in QCD
  - Running Coupling
  - Small at high energy scales $\Rightarrow$ perturbation theory
  - Large at low energies $\Rightarrow$ need non-perturbative method

- Lattice QCD is a reformulation of QCD on a space-time lattice
  - First principles & model independent
  - "proper-theory" - renormalizable quantum field theory
  - Based on the Feynman Path Integral formalism
  - Analogous to e.g. condensed matter crystal system
  - Admits non-perturbative (Monte Carlo) calculations

From CERN Courier (cerncourier.com)
Lattice QCD

- Replace Continuum by a 4-D Lattice
- Rotate to Euclidean Space: \( \exp(-it) \rightarrow \exp(-t) \)
- Quark fields on Lattice Sites: \( \psi(x) \)
- Gauge Fields on ‘Links’
  - “Parallel Transporters” between sites: \( U_\mu(x) = e^{i\alpha g_0 \frac{\lambda}{2} A_\mu^a(x)} \)
  - SU(3) matrices (a.k.a link matrices): \( U_\mu^{\dagger}(x) = U_\mu^{-1}(x) \)
- Transformations:
  - Gauge transformation: \( G(x) \)
  - Gauge fields transform as: \( U_\mu(x) \rightarrow G^{-1}(x) \ U_\mu(x) \ G(x + \mu) \)
  - Fermions transform as: \( \psi(x) \rightarrow G^{-1}(x) \ \psi(x) \)
- Path Integrals -> High dimensional integrals

\[
\langle \mathcal{O} \rangle = \frac{1}{Z} \int D\bar{\psi} \ D\psi \ D\mathcal{A} \ D\bar{\psi} \ D\psi \ \mathcal{O} \ e^{-S(A,\bar{\psi},\psi)}
\]

\[
\langle \mathcal{O} \rangle = \frac{1}{Z} \int \prod_{\text{all links}} dU \prod_{\text{all sites}} d[\bar{\psi}, \psi] \ \mathcal{O} \ e^{-S(U,\bar{\psi},\psi)}
\]
Gauge Actions

• Basic Wilson Plaquette Action
  • has $O(a^2)$ discretization error
  • as $a \to 0$ we have:

$$S_g \to \int d^4x \frac{1}{8} F^a_{\mu\nu}(x) F^a_{\mu\nu}(x)$$

• Can further improve $S_g$: e.g. adding rectangle term
can cancel $O(a^2)$ term:

$$S_g = -\frac{\beta}{N_c} \sum_x \sum_{\mu<\nu} \text{Re} \, \text{Tr} \, P_{\mu\nu}(x)$$

\[
\begin{align*}
\text{Tr} \, P_{\mu\nu}(x) & \rightarrow \frac{5}{3} \text{Tr} \, P_{\mu\nu}(x) - \frac{1}{12} \left( \text{Tr} \, R_{\mu\nu}(x) + \text{Tr} \, R_{\nu\mu}(x) \right)
\end{align*}
\]
Fermions & Pseudo-Fermions

- Fermions are Grassmann Numbers
  - but can do the Gaussian Integral
- Gives a determinant weight to the partition function
  - but determinants are nasty to evaluate so
- Bosonize the determinant
  - write as an integral over ‘pseudo fermion’ (boson) fields
  - NB: Now fermion matrix appears as an inverse: \((M^\dagger M)^{-1}\)

\[
\mathcal{Z} = \int D\bar{\psi} \ D\psi \ e^{-\bar{\psi} M^\dagger(U) M(U) \psi - S_g(U)}
\]

\[
\mathcal{Z} = \int D\bar{\psi} \ D\psi \ e^{-\bar{\psi} M^\dagger(U) M(U) \psi - S_g(U)}
\]

\[
\mathcal{Z} = \int D\phi^\dagger \ D\phi \ e^{-\phi^\dagger (M^\dagger M)^{-1} \phi - S_g(U)}
\]
Fermions & Pseudo-Fermions

• Fermions are Grassmann Numbers
  - but can do the Gaussian Integral
  \[ Z = \int D\psi \bar{D}\psi \ D\bar{\psi} \ D\bar{\psi} \ e^{-\bar{\psi} M^\dagger(U)M(U)\psi - S_g(U)} \]

• Gives a determinant weight to the partition function
  - but determinants are nasty to evaluate so
• Bosonize the determinant
  - write as an integral over ‘pseudo fermion’ (boson) fields
  - NB: Now fermion matrix appears as an inverse: \( (M^\dagger M)^{-1} \)

NB: Fermions have other issues: so called “Fermion Doubling” leading to a diversity of lattice fermion formulations, but that is beyond the scope of this talk
Evaluating Path Integrals: Monte Carlo

- On a lattice we have $4\times$Volume links.
  - e.g. $32^3 \times 256$ Lattice: ~33.6M Links
- Carrying out a 4V dimensional integral directly is unfeasible
- Turn to Monte-Carlo methods (the random walk part)

\[
\langle O \rangle = \frac{1}{Z} \int \prod_{\text{all links}} dU \ O \ e^{-S(U)} \rightarrow \bar{O} = \frac{1}{Z} \sum_{\text{configuration}} O(U) \ P(U)
\]

- Recipe:
  - Generate Configurations: $U$
  - Evaluate the Observable on each configuration
  - Form the “ensemble average” - which is the approximation to the Path Integral
Evaluating Path Integrals: Monte Carlo

- On a lattice we have 4xVolume links.
  - e.g. $32^3 \times 256$ Lattice: ~33.6M Links

- Carrying out a 4V dimensional integral directly is unfeasible

- Turn to Monte-Carlo methods

  Recipe:
  - Generate Configurations: $U$
  - Evaluate the Observable on each configuration
  - Form the “ensemble average” - which is the approximation to the Path Integral

Problem: Since equilibrium probability is sharply peaked, random sampling can pick samples that are not very important and contribute little to the average.
Importance Sampling

- Pick Configuration ‘U’ with probability $P(U)$
- Ensemble average then becomes a ‘regular average’

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \prod_{\text{all links}} dU_i \ \mathcal{O} \ e^{-S(U)} \rightarrow \bar{\mathcal{O}} = \frac{1}{N} \sum_{N} \mathcal{O}(U) \quad \sigma(\bar{\mathcal{O}}) \propto \frac{1}{\sqrt{N}}$$

- E.g.: Metropolis Algorithm
  - Start from some initial configuration $U$
  - Pick trial config $U'$ from $U$ reversibly: ie $P_c(U \rightarrow U') = P_c(U' \rightarrow U)$
  - Accept with Metropolis probability
    $$P(U' \leftarrow U) = \min \left( 1, \frac{e^{-S(U')}}{e^{-S(U)}} \right)$$
  - If we reject, next config is $U$ again
Global Updating

- Metropolis Algorithm would proceed link by link
- For each link one would need to evaluate the quark part of the action

\[ S_f = \phi^\dagger (M^\dagger M)^{-1} \phi = \langle \phi | X \rangle \]

- where

\[ (M^\dagger M) \ X = \phi \]

- and again, M is the fermion matrix
- With 4V links this is prohibitive and so one needs a global update method
1. Refresh momenta from Gaussian Heatbath
   - generate \((U, p)\) from \((U, p_{\text{old}})\)
2. Compute \(H = H(U, p)\)
3. Perform Molecular Dynamics (MD) trajectory
   - generate \((U', p')\)
   - MD must be reversible and ‘area preserving’
4. Compute \(H' = H(U', p')\)
5. Accept with Metropolis probability
   \[ P = \min \left( 1, e^{-H(U', p') + H(U, p)} \right) \]
6. If rejected new state is \((U, p)\)

- \(O(10000)\) trajectories per ensemble
- 60-80% of work in Linear Solvers (Quark MD Forces)
Observables

• Lattice QCD Observables are “correlation functions”

• E.g. For mesons (quark-antiquark pairings):

\[
C(\vec{p}, t) = \sum e^{i\vec{p} \cdot \vec{x}} \ \text{Tr} \ \Gamma G^\dagger(\vec{x}, t; 0, 0) \ \Gamma G(\vec{x}, t; 0, 0)
\]

• G is the quark propagator defined as:

\[
G(x, y) = M_{x,y}^{-1} S(x)
\]

• M is the Fermion matrix

• Computing G involves solving a system of linear equations (Solvers)

Meson: e.g. the π meson (a.k.a pion)

Baryon: e.g. proton or neutron

Meson to 2 meson decay: e.g. ρ → 2π

2 meson in to 2 meson out: e.g. 2π → 2π
Observables

- E.g. For mesons (quark-antiquark pairings):
  \[
  C(\vec{p}, t) = \sum e^{i\vec{p} \cdot \vec{x}} \text{Tr} \Gamma G^\dagger(\vec{x}, t; 0, 0) \Gamma G(\vec{x}, t; 0, 0)
  \]

- Contains contributions from many states
  - that couple to the operators at source and sinks
- Typically the states have a form like:
  \[
  C(t) = \sum_i A_i e^{-E_i t} \quad t \to \infty \quad C(t) \approx A_0 e^{-E_0 t}
  \]
  - Excited states die out as t becomes large
  - Getting at excited states (for spectroscopy) is a formidable challenge.

Meson: e.g. the π meson (a.k.a pion)

\[
\text{Meson: } \pi \text{ meson (a.k.a pion)}
\]

\[
G \quad G^\dagger
\]

\[
C(t) = \sum_i A_i e^{-E_i t} \quad t \to \infty \quad C(t) \approx A_0 e^{-E_0 t}
\]

\[
-\log C(t)/C(t+1)
\]

\[
\text{Excited state contamination}
\]

\[
\text{Signal Region}
\]

\[
\text{Noise etc.}
\]

\[
\text{a mpi } = 0.0755 \pm 0.003 \text{ ChiSq/DOF}=1.75 \text{ Q } = 1.86e-1
\]
Connecting with Reality

- **Chiral extrapolation**: extrapolate to physical quark masses
  - usually simulate with unphysically heavy quarks
  - if quarks light enough: connect to chiral perturbation theory

- **Continuum Extrapolation**: remove the lattice to get a continuum result
  - compute at more than 1 lattice spacing and extrapolate (rigorous)
  - use improved actions and hope/show/convince self that artifacts are small

---

![Graph showing chiral extrapolation](image1.png)

**Chiral Extrapolation**

Allton. et. al.
Phys. Lett. B 628, p125, 2005

---

![Graph showing scaling of \( m_V \) with lattice spacing](image2.png)

**Scaling of \( m_V \) with Lattice Spacing**

for Wilson and Clover actions

Edwards, Joo, Lin,
Steps in an LQCD Calculation

- **Gauge Generation**
  - Generate Gauge Configurations (e.g. HMC Method)

- **Analysis**:
  - Generate the quark propagators: Solving *Lots and Lots* of Linear Equations
  - Contract propagators into Correlation Functions

- **Fitting**:
  - Extract Physics from Correlation Functions
The Cost of Gauge Generation

- Used to be dominated by cost of Gauge Generation

\[ C = 0.024 \left( \frac{L^3 T}{(6 fm)^4} \right)^{5/4} \left( \frac{135 MeV}{m_\pi} \right)^2 \left( \frac{0.1 fm}{a} \right)^6 \left( \frac{\text{#Traj}}{10^{14}} \right) \text{ PFLOPS years} \]

- Growth with V is mild: \( \sim V^{5/4} \) (or even \( \sim V^{9/8} \) if using 4th-order integrators)
- But seems to go as \( \sim 1/m_\pi^2 \) or \( \sim 1/a^6 \)
- Note the units: PFLOPS years = years of running at \( \text{O}(10^{15}) \) FLOPS => HPC Needed
- Analysis techniques have improved and become more expensive
  - nowadays **Gauge Generation is no longer the dominant part** of the calculation
Cost of the Analysis

- Compute Energy Spectrum of 2-meson system
- Quark propagation from $t$ to same $t$ (blue) is the dominant cost.
- For every one of 220 field configurations:
  - 256 values of $t$
  - $\times$ 386 sources
  - $\times$ 4 values of spin
  - $\times$ 2 (light and strange quarks)
  - = **790,528 individual solves per configuration**
- Single precision is good enough
- Just 1 diagram… Complex calculations have 10000’s
  - **contractions get very expensive also**, (Work to reduce this: Edwards, Chen, …)
High Performance Computing

- High Performance Computing in our case is Parallel Computing
- These days, all computing is parallel computing (even your smartphone)

In Core:
(SIMD) Vector Parallelism and Instruction Level Parallelism (ILP)
Thread Level Parallelism (TLP)

Multiple Cores
- 1000s of low power cores
- grouped as SMX’s
- 192 ‘cores’ per SMX
- Vector, ILP, TLP

GPUs:
- 1000s of low power cores
- grouped as SMX’s
- 192 ‘cores’ per SMX
- Vector, ILP, TLP

Many Cores:
- 60+ cores
- Long Vectors
- ILP, Vector and TLP

In Core:
6 Core Intel Xeon
from computing.llnl.gov

Multiple Cores

GPUs:
NVIDIA GK110 Chip Die
from www.geeks3d.com

Many Cores:
Xeon Phi Knight’s Corner
from www.storagereview.com
High Performance Computing

- More levels of parallelism: node/blade -> System

Cray XK Blade (4 Nodes)

4 GPUs
+ 4x16 core CPUs

JLab 9G Cluster Node, about 2 years ago

OLCF Titan: 18688 nodes, 4672 Blades, World #2 on Top 500, Nov. 2014 - 27 PFlops

JLab 9G Cluster: 48-49 Nodes
Parallelism In LQCD

- Two basic patterns in LQCD Computations
  - do the same thing at every lattice site
    - either independently or
    - depending on nearby sites
  - perform some kind of global reduction (sum, inner product)

\[
P_{\mu\nu}(x) = U_\mu(x) \, U_\nu(x + \mu)U_{\mu}^\dagger(x + \nu)U_\nu^\dagger(x)
\]

- This is a classic data parallel pattern.
Parallelism In LQCD

- Two basic patterns in LQCD Computations
  - do the same thing at every lattice site
    - either independently or
    - depending on nearby sites
  - perform some kind of global reduction (sum, inner product)

This is a classic data parallel pattern.

The trick is:
How to map the parallelism of LQCD to the parallelism of the hardware???
Coding for HPC

• Exploiting all the levels of parallelism is challenging. How can we be productive on these machines?

• Write as little parallel code as possible… (Huh??)
  - Use Optimized Libraries
    • General: ATLAS, MKL, CUBLAS etc
    • Domain Specific: QUDA, QPhiX — highly Optimized
    • These have often taken care of machine specific aspects
    • But combinations of them may not play nice together :(  
  - Use ‘Frameworks’ (if available):
    • These often hide parallel aspects from you
    • Free you to think of the scientific problem
    • examples: QDP++ & QDP-JIT, QDP/C
  - Use existing applications with new configurations/inputs (e.g. Chroma)
If you really want to write parallel code

- You will need some programming model
  - CUDA, OpenACC, OpenCL, maybe OpenMP-4.0 for GPUs
  - OpenMP, Threaded Building Blocks, pthreads for CPUs
  - MPI for inter-node message passing

- Stick with Standards
  - This will help portability later

- Isolate low level code into libraries

- Think Parallel
  - there are ‘parallel patterns’: data-parallel, pipeline, etc
  - there are ‘parallel algorithms’: e.g. scans, etc.
  - think of bottlenecks that can inhibit parallelism

```c
#pragma omp parallel for simd
for(i=0; i < N; i++) {
    a[i] = b[i]*c[i]+d[i]
}
```

Image from EPCC: www.epcc.ed.ac.uk

Plenty of parallelism, but still bottlenecked
Low Level Optimization v.s. Algorithms

- Low level optimization is a bit of a black art
- Requires intimate knowledge of
  - what you are trying to optimize
  - the architecture you are optimizing for
    - vectors, caches, memory characteristics, etc.
- Typically decreases all other software quality metrics
- Only worth it if the cost of development amortized
  - e.g. if 1 core hour ~ 10c, 50M core hours ~ $5M,
  - 2x speedup => $2.5M = many times my salary
- Algorithmic improvement
  - reduces computer cycles needed (rather than just gets them faster)
  - can be done at a higher, more productive level
  - if the improvement is successful, one can come back to optimize it
threshold. In each case we present the influence of partial-waves, constant matrix, while the region around the MeV, 19 levels to constrain the 1 amplitudes – open circles on axis show points determined directly without parameterization of the vector amplitude from three volumes: 16 extraction – larger solid points show in degrees. Lower panel: inelasticity. Points in center show the energy levels on three volumes used to constrain the amplitudes; in particular note that the low-energy behavior, which is real and continuous across the threshold and a further 24 levels contribute). Similarly in Fig. 2(c), the energy region of the rapid rise of the phase-shift. This region is is negligible in this case.

Architecture and Performance

- Baseline Single Precision Wilson Dslash performance in Chroma on Sandy Bridge: \(~35.8\) GF
- Optimization for GPU and Xeon Phi had significant benefit: 8x-9x speedup
- Optimizations from Xeon Phi fed back to regular Xeon: >3.4x speedup over 35.8 GF
- Improved performance “forward scaling”
  - Improves on new generation GPU and Xeon
Architecture and Algorithms

- Older Algorithm on GPU: BiCGStab
  - Scaling bottlenecked by inter GPU communications
- Lead to development of new algorithm
  - GCR Solver + Domain Decomposed preconditioner
  - Reduced communication needs
  - Preconditioner can benefit from faster running in reduced precision
- Much improved scaling over BiCGStab
Algorithms and Performance

- Highly optimized GPU version of BiCGStab gave ~8x speed up over regular CPU
- New Multi-Grid algorithm on CPU competitive with GPU for light quarks.
- Need to re-optimize the new algorithm to get back (some/all) of the performance advantage offered by GPU
The future…

• Near Future (ongoing):
  - Current calculations on our clusters, and at a variety of LCFs (OLCF, ALCF, NCSA etc)
  - Preparing for the next generation of machines
    • Cori System at NERSC: Intel Xeon Phi, Knight’s Landing
    • Summit System at OLCF: Next gen Volta GPU + IBM Power CPU

• Exascale is coming
  - Expected in the Generation after Summit, at around 2020-2022.
Summary

• We discussed the basics of Lattice Calculations
• We discussed how Lattice calcs tie in with High Performance Computing
• I briefly mentioned my thoughts about programming for these systems.
• We looked at a couple of cases, of how physics, algorithms, architecture and performance all influence each other
• Looked into the future…

• Questions?