Thematic CERN School of Computing

STUDENTS LIGHTNING TALK

Checkerboard Metropolis Algorithm for Large Size Lattice Simulations in GPU

Aravind T S

Tata Institute of Fundamental Research, Mumbai, India

Outline

“ The Problem “

The Problem

Markov Chain Monte-Carlo

Checkerboard Metropolis Algorithm

Implementation

Results

“ The Problem “

- The Lagrangian of the φ^4 theory is described as // A Scalar field theory

$$\mathcal{L} = -\frac{1}{2}(\partial_\mu \phi)^2 - \frac{1}{2}m^2\phi^2 - \frac{1}{4!}\lambda\phi^4$$

- Dynamics / Physics of the system is given by the integral over the path defined from a point in phase space q’ to q”

$$\langle q', t_2 \mid q'', t_1 \rangle = \int [dq] \exp\{i\mathcal{S}\}$$

$$\mathcal{S} = \int \mathcal{L} d^4 q$$

- The integral can be calculated by discitizing the path and doing Monte-Carlo integration of the discretized ***lattice***

The Problem

- φ^4 → is a function of (t , x, y, z) ⇒ The integral is from q' → (t',x',y',z') to q'' → (t'',x'',y'',z'')

$$S_E = i \sum_n \left(\sum_{\hat{\mu}} \frac{1}{2} \phi(n) \left(2\phi(n) - \phi(n-\hat{\mu}) - \phi(n+\hat{\mu})\right) + \frac{m^2}{2}\phi(n)^2 + \frac{\lambda}{4!}\phi(n)^4 \right)$$

- After discretization we get a 4 dimensional lattice of scalar φ

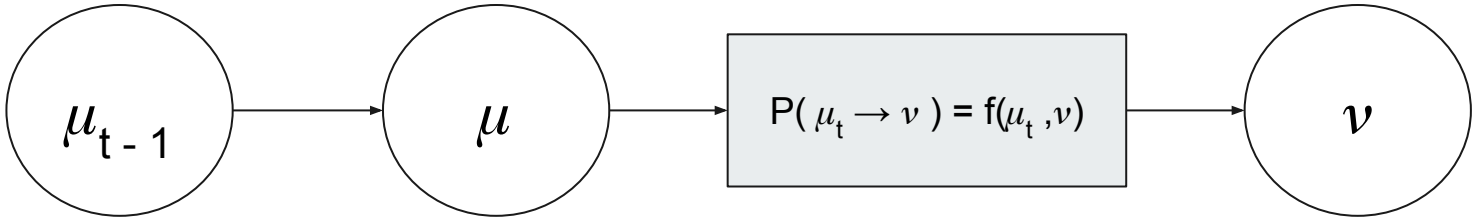
$$\langle q', t_2 \mid q'', t_1 \rangle = \int [dq] \exp\{i\mathcal{S}\}$$

  - Each lattice point with the value of φ at (t , x , y , z)
  - **Lagrangian is Local**
  - Observables has to be calculated from the lattice
    - weighted according to the *negative exponential of action*
    - Equivalent to doing a ***importance sampling*** of the lattice configurations according to same weights

- Perform Monte-Carlo integration on the lattice configurations

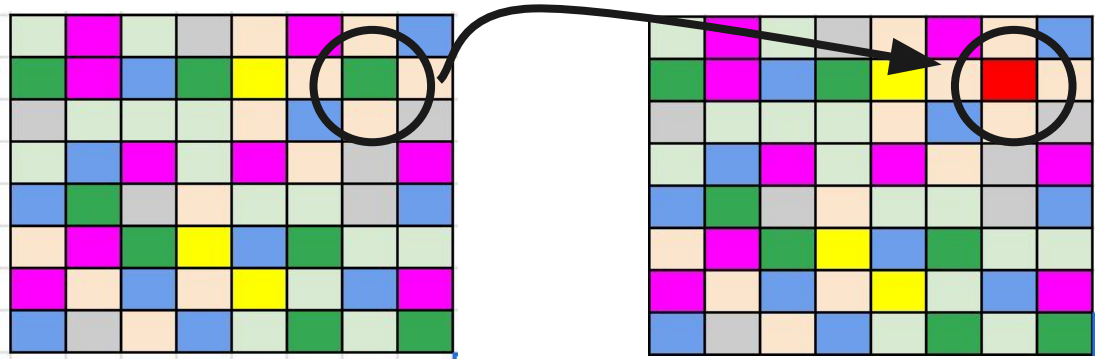
Markov-Chain Monte Carlo

- Markov Chain
  - A sequence of states $\{\mu_t\}$ with $P(\mu_t \rightarrow \mu_{t+1})$ is independent of the $\{\mu_{t-1}\}$
  - An Ergodic markov chain is proven to have a unique stationary state.
  - Further imposition of time reversal symmetry precisely gives us the transition probabilities in this case



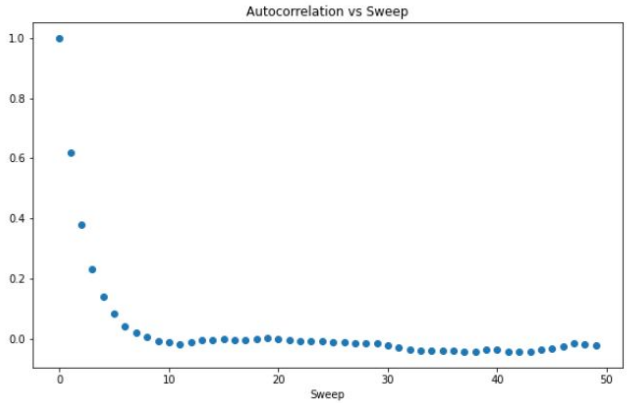
Markov-Chain Monte Carlo

- Given a lattice configuration (μ_t) we generate a new configuration (ν) as a perturbation to the current configuration
  - Perturb each site on the lattice at a time : O(L^4) scaling
    - New configuration is accepted with a probability $min(1, e^{-\beta\delta E})$
    - 1 Sweep = L^4 site probes
  - O(L^4) scaling : Double the lattice Dimension ⇒ Order of magnitude increase in time



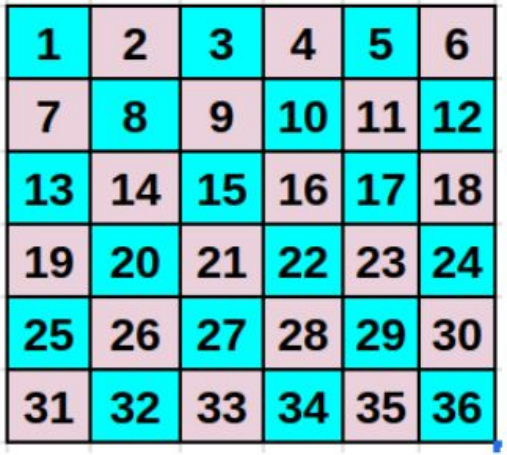
Markov-Chain Monte Carlo

- Sequence of the lattice configurations generated have auto-correlation
  - Quantified by the autocorrelation time of the generated observables
- For integrations we require independent sampling of the lattice phase-space
  - → Large number of Sweeps needs to be realized



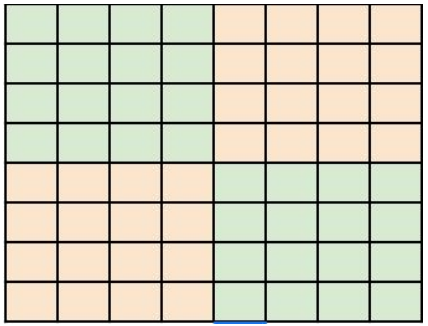
Checkerboard Metropolis Algorithm

- Given a lattice configuration (μ_t) we generate a new configuration (ν) as a perturbation to the current configuration
- A better parallelizable logic for this update ?
  - Divide the lattice in to odd-even meshes
  - Update each mesh parallely
    - Suitable for a GPU implementation
      - Locality of Lagrangian
  - Ergodic ! → guaranteed correctness



Implementation

- Targeted NVIDIA GPUs with CUDA Cores
- For lattices fitting in the GPU Memory
    - Fixed memory for lattice/observable in the GPU
- Mapping of the Lattice sites to DRAM such that the neighbours are close to each other in memory physically
    - A coarse grain implementation // Could be improved further
- Use of ***curand*** for large throughput random number generation
- Minimal data and control transfers between GPU and CPU



Results and Summary

- The scalability of the implementation to large lattices was better

Lattice Length	Lattice Size	Time for 10^3 NCLC	Throughput (Hz)
4	256	4501.61	222.14
8	4096	12263.4	81.54
10	10000	16183.9	61.79
14	38416	18830.7	53.10
16	65536	20074.4	49.81

- Definite improvements possible
  - memory access patterns ??
  - Hope to learn thighs properly here

- **Supposed to be** a part of the ML pipeline for a reinforcement learning project. [1 , 2]

Thank you !

More detailed report

gitHub repo with the implementation