PIMC Introductory Meeting

Brendan Philbin, Chris Herdman

22 June 2020

Not expected to work full time for the entire 2 months

Similar to the stipend that a summer research assistant would receive

View it as a 6-8 week full-time commitment

Formally not a job, not logging hours, etc.

Things needed

* Monthly meetings
  + Give overview of project w/ slides at first meeting (July)
  + Provide updates at subsequent meetings
  + Give a brief conclusion presentation at the last meeting (August or September)

Project

* Will be talking with Adrian more on Thursday
  + Adrian is the Vermont leader for NE Cyberteam
* Formally, Chris is the project owner and Adrian will be my mentor

Project Abstract

“Monte Carlo simulations often can take advantage of HPC hardware by using embarrassingly parallel workloads. However, these algorithms must first equilibrate from their initial state before useful statistical samples can be generated. In simulations of certain physical systems (especially those with "glassy" dynamics) this equilibration processes is one of the most computationally demanding parts of the calculation, yet cannot easily benefit from parallelization without communication between tasks. This project will involve implementing the "population annealing" method to speed up the equilibration process of a Monte Carlo code. The implementation of population annealing will use distributed computing techniques, in particular using MPI, to allow the equilibration process to take advantage of a large number of CPU cores in an HPC environment.

Monte Carlo

* Begin with a new configuration, tweak the configuration, update the system
* Every time you change the state you end up with a new, related state
  + “Markov chain”
* Once simulation is equilibrated, the parallelization is pretty straightforward
  + The equilibration itself is a fairly intensive aspect
* When simulation starts, the system needs to be put into the state you actually want to sample
  + Ex. Simulation begins at a high energy and needs to be equilibrated to a lower energy, which is what we’re looking for
* **Research: Monte Carlo calculations, Population annealing, Markov chain**
  + [**MIT Monte Carlo Simulation Lecture**](https://www.youtube.com/watch?v=OgO1gpXSUzU)
* Can write algorithm where we don’t equilibrate each seed and instead use HPC to equilibrate with multiple cores
  + Dividing equilibration up among multiple cores
  + Don’t need to throw away equilibration time over and over
* One possibility is to equilibrate one calculation and then spawn X equilibrations from that calculation
  + Want to parallelize the equilibration itself
  + This is the general idea of the project

Population Annealing

* Start calculations independently and then check in periodically and shuffling the configurations accordingly
  + Ex. Check in every couple of steps, take the lowest energy configuration, respawn all configurations from this point going forward. Rinse and repeat. (Simplified)
* Equilibration processes swap between configurations
* This is generic to any sort of Monte Carlo
* **Goal: implement this in the PIMC code**
  + Might make sense to implement this in a simpler code base first
  + Chris has other research projects that could also benefit from this
  + Get familiar with the process itself, test in on a cluster, and then apply it to the PIMC
* [Ising model](https://en.wikipedia.org/wiki/Ising_model)

Things to look into

* Monte Carlo simulations
* Ising model
  + Metropolis Algorithm section, specifically
* Markov chains
* Population annealing

To do

* Read up on Monte Carlo simulations, population annealing, Markov Chains, Path Integral Quantum Monte Carlo
* Implement basic Monte Carlo to become familiar with the algorithm
* Try to get simple population annealing going from the above basic Monte Carlo Ising model
  + Make your own or use the [Simplest Monte Carlo Ising Model We Found](http://physics.usc.edu/~shaas/516/isingprograms/)
    - Also see last pages of “Ising Model” PDF
  + Look for simpler ones to work on implementing population annealing
  + Looking for case where every spin interacts with every other spin
    - Most codes usually depend on each of four neighbors
    - “All to all case”
    - Simple to do with matrix multiplication (Use Boost library to gain familiarity?)
* is a vector of all of the spins (+1 or -1)
* T is the transpose of
* J is a matrix of the bond types between all of the particles
  + J has 0s along the main diagonal
  + For the simple code, pick the J(ij) values to be randomly +1 or -1 (except for the main diagonal)
    - For simplest, have main diagonal be 0 and all other values to be +1
* Repeat the Monte Carlo loop a certain number of times (algorithmic parameter)

What we want to look at

* Run for a certain number of updates
* Check energy at each update
* Look at the trend of energy as a function of the number of updates to see the equilibration process