Parallel Implementation of Gibbs Ensemble Monte Carlo Simulation

AM5080

Veerendra Harshal Budhi

Indian Institute of Technology Madras

15 05 2021

Abstract

The Gibbs Ensemble Method in Molecular Dynamics belongs to a class of Monte Carlo Methods which rely on Random Walks which are inherently sequential in nature and don't allow for a straight forward divide and conquer parallelisation. Hence, a robust class of parallel programs not restricted by simultaneity is essential.The programs used in this implementation are designed to be generic and applicable outside of the scope of the Simulation.

Theory

Lennard-Jones Potential

The simulation runs on the Lennard Jones Potential Model which makes it easy to find the energy of a given configuration and hence ultimately lead to the probabilities.The complexity of calculating the energy of a given configuration with N particles is $O(N^2)$.

Importance Sampling

The process of accepting new configurations is handled by the Metropolis Hastings Algorithm which imparts probabilities so that the final collection of accepted configurations are in accord to a desired probability distribution. This makes it easy to calculate ensemble thermodynamic averages making what would have been a weighted average, a simple average.

Theory

Gibbs Ensemble

Useful to simulate vapour-liquid co-existence. Does so by simulating liquid and gas in separate boxes. Keeps T, Total Volume, Total Number of Particles constant. Allows particle displacement, volume exchange, particle exchange as trial moves. This helps to maintain the same T (Thermal Equilibrium), P (Mechanical Equilibrium) and μ (Chemical Equilibrium) across the two boxes. [4]

Algorithm

The Gibbs ensemble method was introduced by Panagiotopoulos and Panagiotopoulos et al. for the direct simulation of gas-liquid and liquid-liquid equilibria. It combines canonical (NVT), isobaric-isothermal (NPT), and grand canonical (μVT) Monte Carlo techniques in a single simulation using two boxes at given initial densities at the desired temperature. [3]
When equilibrium is attained the gas is sampled in one of the boxes, the liquid in the other, in such a way that the chemical potentials and pressures in the two boxes become equal according to the conditions for phase coexistence.

Pseudo Code - GEMC

```
#Program Begins
input data.in                          #Import Input Variables.
coord = Initialise_Lattice()           #Current coordinates
for step in range(1,Nsteps):
    mode = Choose_Trial_Move()         #Choose Trial Move
    if mode = 'Particle Displacement':
        coord = Displace(coord)
    elif mode = 'Volume Exchange':
        coord = Volume(coord)
    elif mode = 'Particle Exchange':
        coord = Swap(coord)
    data_summary.append(data_interpret(coord))
endfor
postprocess()                          #Perform Calculations
```

Pseudo Code - Trial Move

```
def Trial_Move(coord):
    c_o = coord
    c_n = trial_gen(coord)  #Propose a Trial Configuration
    acc_prob=calc_p(c_o,c_n)#Calculate P(Acceptance)
    if rand(0,1) < acc_prob:
        coord = c_n         #Accept
    return coord
```

Time Calculation

Probabilities and sampling frequency for the trial moves and writing functions $= \vec{\eta}$. $< T_i >= a_i N^2 + b_i N + c_i$.These coefficients are stored row-wise in a matrix A. Also, $\vec{N}_2 = [N^2, N, 1]^T$

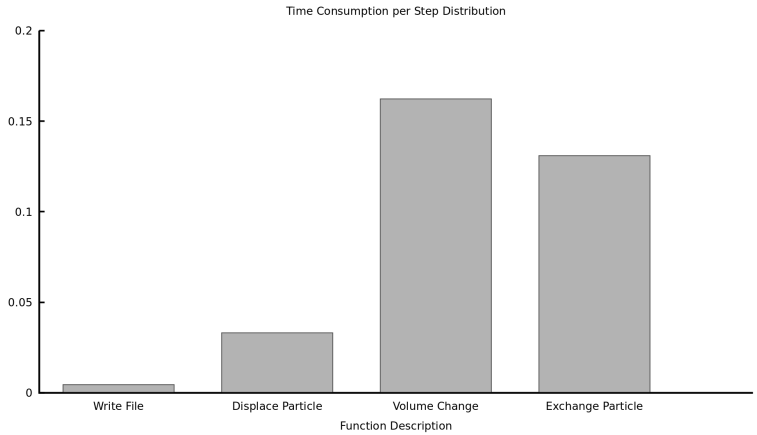
$< T_{total} >= S \times < T > +d = S\vec{\eta}^T < \vec{T} > +d =$

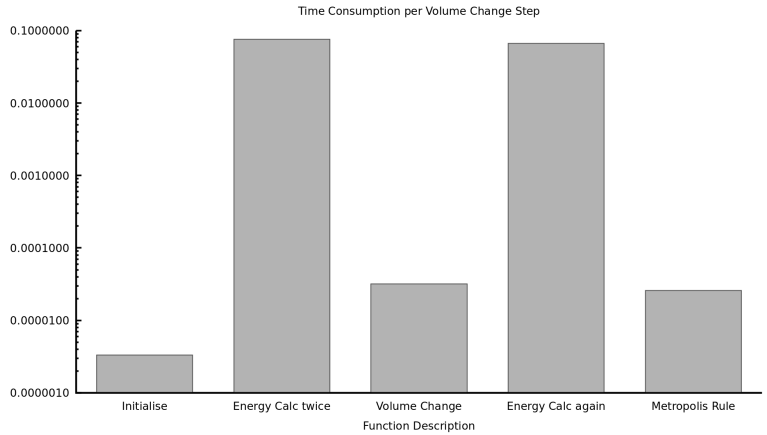
$S \times \vec{\eta}^T A \vec{N}_2 + d = S \times (aN^2 + bN + c) + d$

Ignoring non-leading terms in N,

$T_{total} = O(S, N^2)$

Time Consumption Breakdown





- Trial Moves are more time consuming than Sampling routines.
- Most time consuming function call calculates the energy of the box using the Lennard-Jones Potential Model.
- The algorithm is of $O(N^2)$.

Problem Statement

Difficulties

- As Monte Carlo Random Walk is sequential and highly stochastic, it is unwise to sample together simultaneously run results.
- Function Calls when made more than once, might need a correspondingly equivalent number of support routines in the other processes.
- Within a step, code irrelevant to the next step is not a priority.Storing it to save time wastes memory.

Solutions

- A Master and Slave approach can be useful to problems that are sequential. Main program can keep running in the master node and slave processes pitch in at time-intensive function calls.
- The slave processes must be able to support multiple function calls.
- Less priority work can be handled down to slave processes.

Parallel Functions

Parallel Function Receiver

When the Master Node is running a program, it might come across a bottleneck function that takes significant time to run. The idea is to keep the Slave Nodes awaiting instructions on the function call. The Parallel processing is limited only the duration of the function call after which the slave nodes get inactive awaiting further instructions

Background Worker

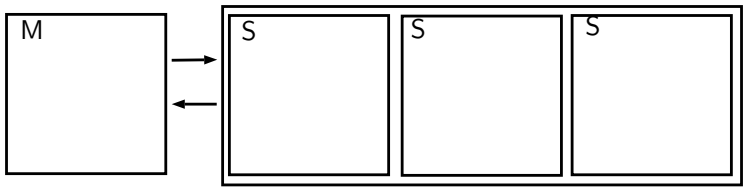
The master node runs the program and leaves in the workspace data that need further work. If the result of the work is not particularly pertinent to that of the master node, then the work can be delegated to the Slave Nodes by the master who moves on. The result has to be stored according to some protocol without bothering the master node.

Self-Motivated Worker

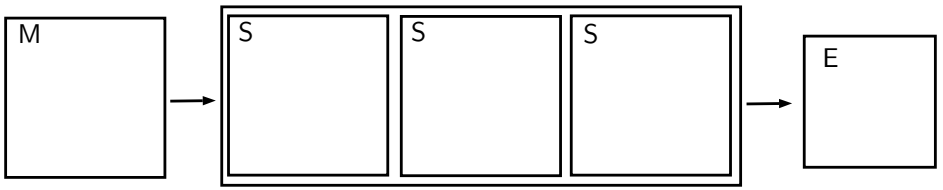
A Dedicated set of processes can be left dedicated to accomplish minor activities within the major program. Printing Output to file, Animation, plotting etc can slow down the progress of the master node and hence is best left to dedicated workers who can further parallelize amongst themselves.

Parallel Functions

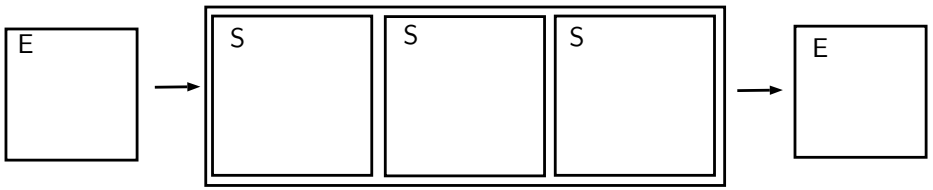
The data flow is signified here with arrows.
M = Master Process S = Slave Process E = External Input



(a) Function Receiver Mode



(b) Background Worker



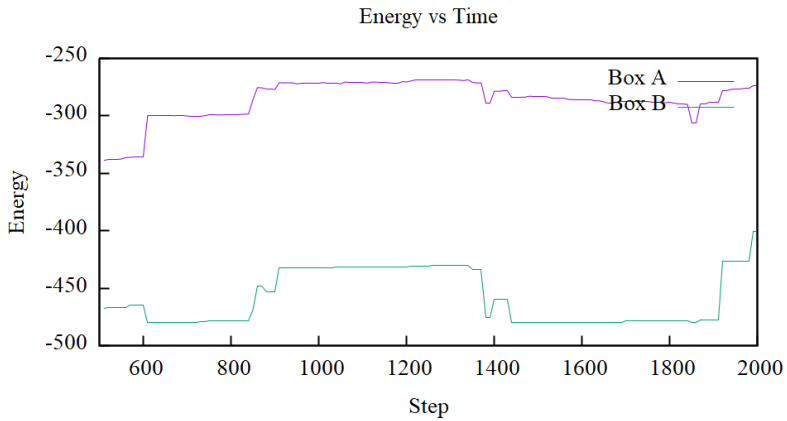
(c) Self Motivated Worker

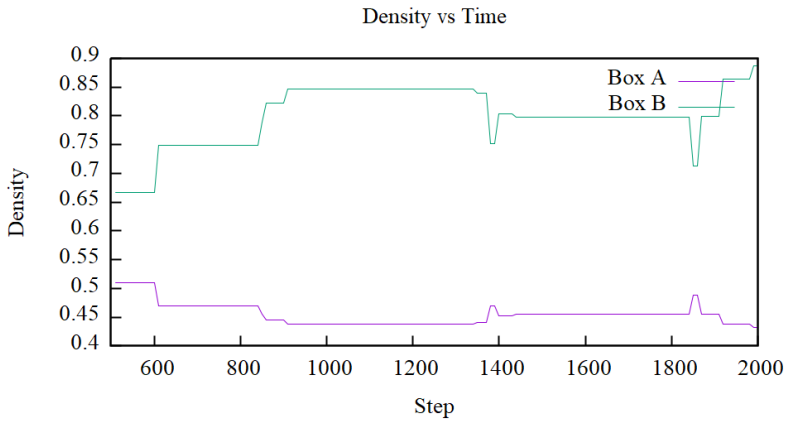
Simulation Results

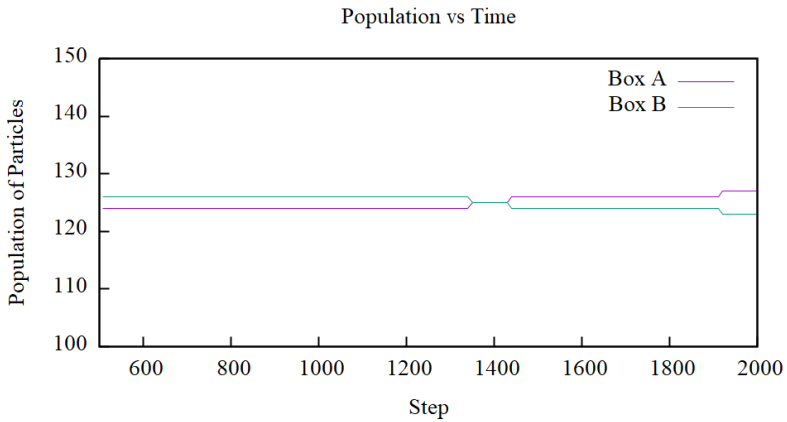
Simulation run using 4 processors.[1, 2]

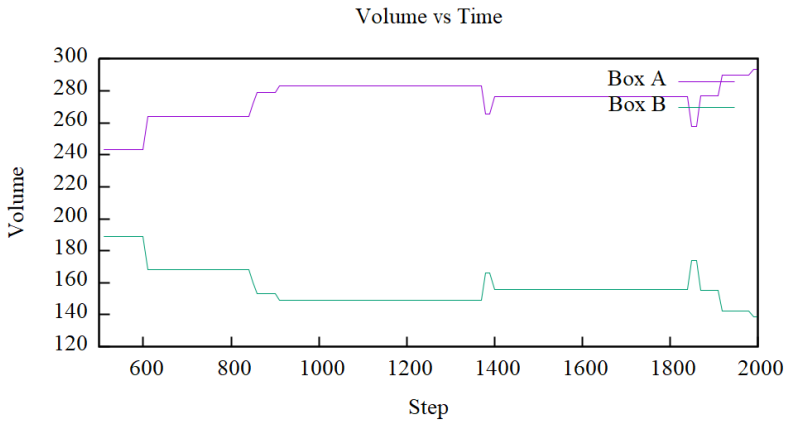
```
1  Gibbs Ensemble Monte Carlo Simulation of
2  the Lennard - Jones Fluid
3  Temperature of the System = 0.8
4  Volume of the Simulation = 432
5  Number of Particles = 250
6  Net Density = 0.5787037037037037
7  Step Number: 2000
8  Density 1 = 0.45517336468375685
9  Density 2 = 0.7997939736392827
10 Standard Deviation 1= 0.01921138833633007
11 Standard Deviation 2= 0.05203479644687997
12 Time 14m2.203s
```

Simulation Results





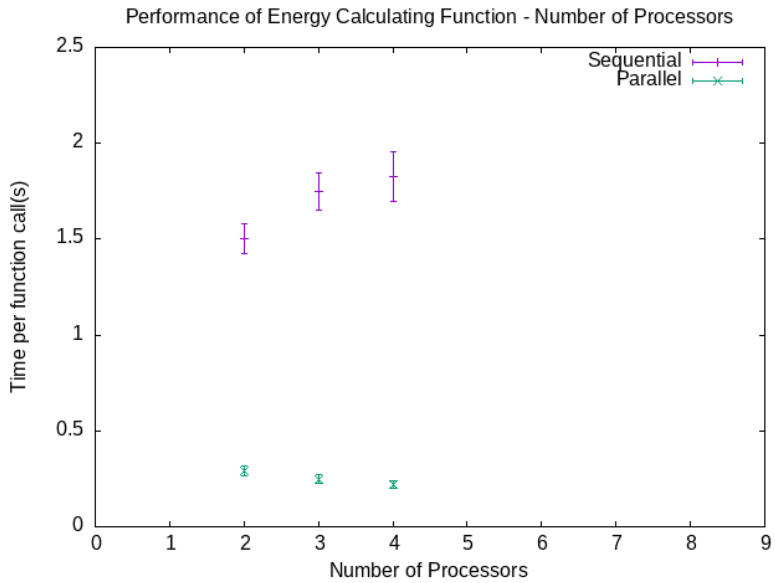




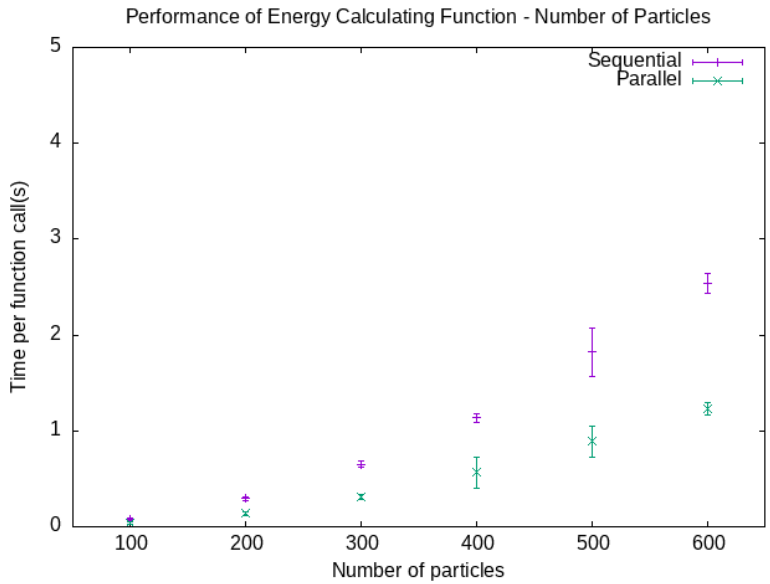
Simulation Results

	$\bar{\rho}$	σ_{rho}	$\lvert\mathbf{E}\rvert$	σ_E
Liquid	0.805598	0.030788	469.061	21.28593
Gas	0.453827	0.009155	285.323	5.875716

Results pertinent to stabilized regions.



200 Particles



4 Processors

References

[1] L. Dalcin.
Tutorial mpi for python 3.0.3 documentation, 2021.

[2] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo.
Parallel distributed computing using python.
Advances in Water Resources, 34(9):1124–1139, 2011.

[3] D. Frenkel and B. Smit.
Understanding molecular simulation.
Academic Press, 2002.

[4] T. J. H. Vlugt, J. P. J. M. v. d. Eerden, M. Dijkstra, B. Smit, and D. Frenkel.
Introduction to molecular simulation and statistical thermodynamics.
2008.

Thanks!