Project Report - III on

Energy Calculation for Molecular Dynamics Simulation using OpenMP

For

CSE4001 - Parallel and Distributed Computing

Slot : C2

Satvarsh Gondala **18BCE2098**

Tharun Tej **18BCE0105**

Faculty :

Mr. Manoov R.
SCOPE

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Date of Submission : October 30th 2020

Project Roles:

Satvarsh Gondala	Coding Implementation, Design, Working, Project Idea, Report Preparation, Presentation, Literature Review 1,3,4 Papers
Tharun Tej	Project Idea, Report Formatting, Preparing Testing Table, Literature Review 2nd paper

Abstract:

MD(Molecular Dynamics) simulation is a progressed numerical method that shows the use of a suitable algorithm to solve the classical equations of motion for atoms relating with a known interatomic potential. These techniques are useful for studying surface phenomena, as they give a qualitative explanation of surface structure. This is not a topic that is often recognized as important and also the one that is highly benefited by the inclusion of Parallel Processing. There is a significant difference even by applying surface level openMP programs.

Important Terms: Molecular Dynamics, Parallel Processing, openMP

Introduction:

Molecular Dynamics has been used for many decades now to understand the temperature and pressure dependencies of dynamical concepts in liquids, solids, and liquid-solid interfaces. MD simulation techniques are also Ill matched for understanding surface phenomena, as they give a qualitative understanding of surface structure and dynamics. This Project uses MD methods to better comprehend surface disorder and pre melting.

Generally, it examines the temperature dependence of structure and vibration dynamics at surfaces of face centered cubic (FCC) metals-mostly Ag, Cu, and Ni. It also makes exchange with results from other theoretical and experimental approaches. While the importance in this Project is on metal surfaces, the MD technique has been

applied to a wide variety of surfaces with those of semiconductors, insulators, alloys, glasses, and simple or binary liquids. A full review of the pros and cons of the technique as verified to these very exciting systems is ahead the range of this Project.

HoIver, it is important to point out that the success of the classical MD simulation depends on the accuracy with which the forces acting on the ion cores can be determined. On semiconductor and insulator surfaces, the success of molecular dynamics simulations has made them more appropriate for such designs rather than standard MD simulations.

One of the important factors as to why I think this project is a perfect match for PDC is the requirement of scalability. For big projects, these techniques could be executed with millions of atoms and molecules in mind. Parallel Processing helps overcome this obstacle whilst improving the core technique.

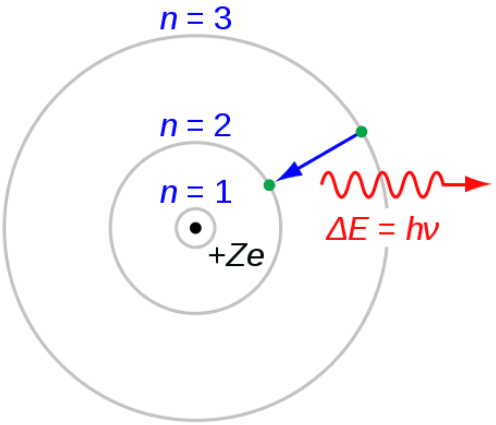
Advantages of Molecular Dynamics: (In terms of Project Impact)

1) Atoms are never still

Meaning that atoms are always in constant motion, they don't have fixed energies and always keep on interchanging. Different orbits require different energy levels. With this I can derive one important property:

- Probability of detecting specific arrangement of atoms is function of potential energy

Important Term: Bohr Atomic Model

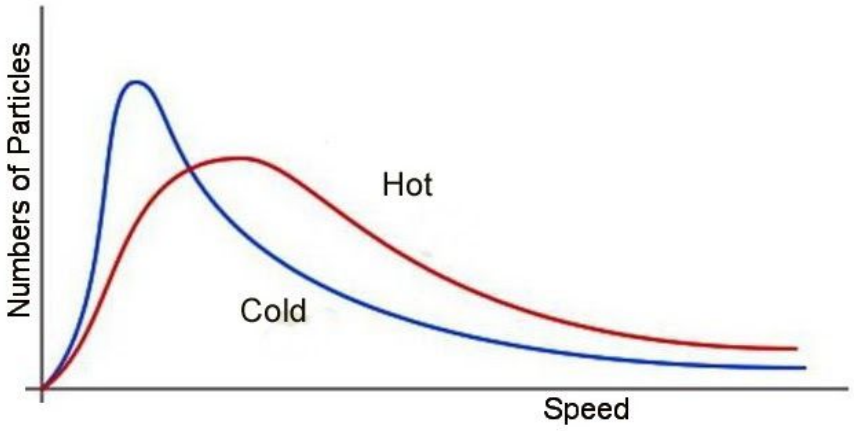


2) **Conservation of Energy:**

Total Energy is given as the sum of potential and kinetic energy. It is generally conserved

- Speed of atom is inversely proportional to potential energy
- In our simulations I maintain a common temperature to prevent any change of energy

Important Term: Boltzmann Distribution



Limitations of Molecular Dynamics: (In terms of Project Impact)

1) Scalability of Time:

Due to the nature of the atoms, the scale of time is extremely small, often in microseconds, milliseconds or nanoseconds. I can just assume one set of time period and use parallel computing to get the results for multiple units of time.

All the above pros and cons have an impact in terms of the actual code and it's working. Each point is based on a principle. Our goal is to be as accurate to the theory/principles as I are to the practical code

Literature Review:

Back when I Ire looking for various ideas, I found this website:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4655909/

Our goal back then as suggested by you was to focus on a real time application that uses parallel and distributed computing in an effective way. Upon reading the article I found that PDC is very useful in creating efficient simulations of molecular dynamics. I then Int to IEEE and found the following research papers, that helped us shape our model

[1] Elucidating membrane protein function through long-timescale molecular dynamics simulation

How this helped?

This was the first paper that brought the notion of a time scale, especially for a larger time scale. One of the successful models that included scalability. If you look at our Limitation, I mentioned time scale as a problem as the time is generally in terms of microseconds, nano seconds. This model features time-scale in minutes and helped us understand how models can change according to the scaling of time.

What are the shortcomings?

Everything else except for the timescale focuses on Elucidating membrane, a topic that is too advanced for us and is not something that I am attempting to simulate and account for in our MD simulation.

Link: https://ieeexplore.ieee.org/document/5335057

[2] Adaptive Resolution Molecular Dynamics Simulation with Constant in H

How this helped?

This paper focuses on the concept that temperature can change the properties of atoms drastically from speed to physical changes. This gives an in depth notion for Boltzman Constant. This introduced the idea of thermostat - meaning a constant fixed temperature. This also provides various solutions to account for changing temperatures, but that topic is too advanced to pull off.

What are the shortcomings?

The main focus of this paper is the adaptive nature of its model. Meaning that 1240 out of 1270 simulations used changing temperatures as a base, making this model highly different in the way base calculations occur.

Link: https://ieeexplore.ieee.org/document/8776732

[3] Fabrication Sub-10nm Metallic Gratings with Carbon Nanotube – A Study by Molecular Dynamics Simulation Method

How this helped?

Atoms are mostly confined to a 3D space, meaning you have to account for three dimensions being x,y and z. This model helps us realize and bring our model to a 2D space by forcing all the reactions to occur in a single nanotube. This also helped us realise most relative formulas that helped in energy calculation.

What are the shortcomings?

This paper is mostly for fabrication and not energy calculations, making most of the formulas irrelevant to our model.

Link: https://ieeexplore.ieee.org/document/6017499

[4] Parallelization of Molecular Dynamics code

How this helped?

This helped us the most by giving the idea for the core loop on how to efficiently execute MD simulation via parallel simulation. This paper includes three versions of code accounting for various libraries of PDC software used.

What are the shortcomings?

The main focus of this is CUDA and MPI which are NOT what I are focusing on. Also there is no proper explanation for calculating energies in this model making most of the goals in this code irrelevant.

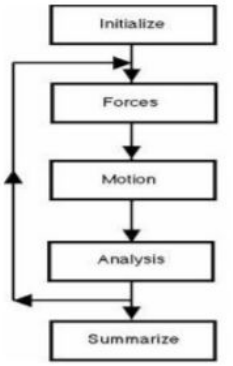
Link: https://ieeexplore.ieee.org/document/6621390

Proposed Methodology:

I will write a code which will expect the kinetic and potential energy of an atom at all time steps which will help us in foreseeing the motions (velocity, position acceleration etc) of the atoms. The sum of kinetic energy and potential energy must be constant and equal to total energy. Our code will also print the error in total energy which will tell us about what percentage of error I can get in our calculations.

Input	Processing	Output
Spatial Dimension	At every step, our simulation should report the potential and kinetic energies	Relative Error in Total energy.
No. of particles in Simulation	The sum of these energies should be a constant.	Relative Error in Total energy.
No. of Time Steps	As an accuracy check, our simulation should also print the relative error in the total energy.	Time difference between Serial and parallel Execution.
Size of Each Time step	I should parallelize the code and associate the time taken in both serial and parallel execution	Time difference between Serial and parallel Execution.

My main draw is still to reduce the amount of time taken to compute the potential and kinetic energies of atoms. This can be achieved via Parallel Processing



Our application code will have the following functions:

COMPUTE - computes the forces and energies of atoms.
Initialize - INITIALIZE initializes the positions, velocities, and accelerations.
Timestamp - TIMESTAMP prints the current YMDHMS date as a timestamp.
Update - UPDATE updates positions, velocities and accelerations.
Displacement - DISPLACEMENT computes the displacement between two particles.
CPU Time - CPU_TIME reports the elapsed CPU time.

Compute and update functions are the most time overriding functions as they are only about computation and solving mathematical equations so I am going to parallelize both of them. Some parallel functions which I am about to use in our code are pragma omp parallel, pragma omp reduction, pragma omp for, pragma omp shared, pragma omp private.

Output: (for energy calculation)(elaborated version)

```
D:\C\newasd\bin\Debug\newasd.exe
30 October 2020 09:33:36 PM

OPENMP Version 1.7

 Energy Simulation for MD

 NP, the number of particles in the simulation is 1000
 STEP_NUM, the number of time steps, is 400
 DT, the size of each time step, is 0.000100
 Please refer to the review -II report for full list of parameters used

 Number of processors available = 12
 Number of threads = 12

 Initializing positions, velocities, and accelerations.

 Computing initial forces and energies.

 At each step, we report the potential and kinetic energies.
 The sum of these energies should be a constant.
 As an accuracy check, we also print the relative error
 in the total energy.
```

In the next picture I wanted to highlight the difference between OpenMP version with parallel computing and the other one without parallel computing, the difference is big in terms of the execution time, while the result is the same.

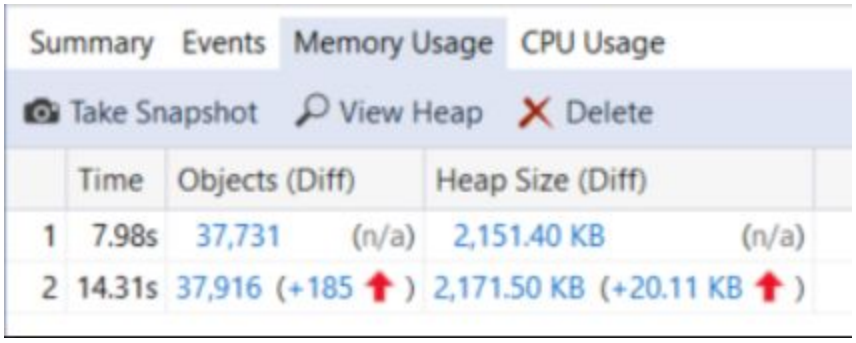
```
Step Potential Kinetic (P+K-E0)/E0
Energy P Energy K Relative Energy Error

       0  498129.505358       0.000000  0.000000e+000
      40  498129.453544       0.051822  1.737701e-011
      80  498129.289900       0.215466  1.609210e-011
     120  498129.014287       0.491077  1.251969e-011
     160  498128.626592       0.878768  5.488677e-012
     200  498128.126652       1.378702 -6.162215e-012
     240  498127.514261       1.991085 -2.360981e-011
     280  498126.789162       2.716171 -4.803571e-011
     320  498125.951055       3.554263 -8.064176e-011
     360  498124.999589       4.505708 -1.225549e-010
     400  498123.934369       5.570902 -1.749562e-010

Elapsed time for main computation:
21.401700 seconds.
end of execution.
```

without parrelization for 12 processes 21sec

```
Step Potential Kinetic (P+K-E0)/E0
Energy P Energy K Relative Energy Error

       0  498129.505358       0.000000  0.000000e+000
      40  498129.453544       0.051822  1.737701e-011
      80  498129.289900       0.215466  1.609222e-011
     120  498129.014287       0.491077  1.251981e-011
     160  498128.626592       0.878768  5.488677e-012
     200  498128.126652       1.378702 -6.162448e-012
     240  498127.514261       1.991085 -2.360981e-011
     280  498126.789162       2.716171 -4.803583e-011
     320  498125.951055       3.554263 -8.064199e-011
     360  498124.999589       4.505708 -1.225550e-010
     400  498123.934369       5.570902 -1.749561e-010

Elapsed time for main computation:
8.044000 seconds.
end of execution.

30 October 2020 09:33:44 PM
```

with parallelization 8 sec.

Performance Profiler test for memory usage: using memory usage function from visual studio code



You can notice the difference between the normal and parallelized process when computing for 8 processes

Here you can see the added support of manual input, currently only given for number of processes:

- **Parralization (4.7 sec)**

```
Enter number of processes: 6

0  498129.505358       0.000000  0.000000e+000
40  498129.453544       0.051822  1.737701e-011
80  498129.289900       0.215466  1.609210e-011
120  498129.014287       0.491077  1.251969e-011
160  498128.626592       0.878768  5.488677e-012
200  498128.126652       1.378702 -6.162215e-012

Elapsed time for main computation:
4.700630 seconds.
end of execution

Process returned 0 (0x0)   execution time : 5.768 s
Press any key to continue.
```

- **With parallelization and Boltzman constant (New function) - 6sec**

```
Enter number of processes: 6

boltz: 0.6
0  498129.505358       0.000000  0.000000e+000
boltz: 0.3
40  498129.453544       0.051822  1.737701e-011
boltz: 0.7
80  498129.289900       0.215466  1.609210e-011
boltz: 0.9
120  498129.014287       0.491077  1.251969e-011
boltz: 0.4
160  498128.626592       0.878768  5.488677e-012
boltz: 0.3
200  498128.126652       1.378702 -6.162215e-012

Elapsed time for main computation:
6.480090 seconds.
end of execution

Process returned 0 (0x0)   execution time : 7.338 s
Press any key to continue.
```

- **Without parallelization - 13 sec**

```
Enter number of processes: 6

0  498129.505358       0.000000  0.000000e+000
40  498129.453544       0.051822  1.737701e-011
80  498129.289900       0.215466  1.609210e-011
120  498129.014287       0.491077  1.251969e-011
160  498128.626592       0.878768  5.488677e-012
200  498128.126652       1.378702 -6.162215e-012

Elapsed time for main computation:
13.053000 seconds.
end of execution

Process returned 0 (0x0)   execution time : 14.856 s
Press any key to continue.
```

Results and Conclusion:

The thing I wanted to focus on this project was mostly nailing down the importance of parallelization, while staying true to the base principles in theoretical part

In this project I present a new approach to accelerate molecular dynamics simulations with inexpensive product graphics hardware. To derive an effective mapping onto this type of computer architecture, I have used the new Compute Unified Device Architecture programming interface to implement a new parallel algorithm.

My experimental results show that the parallel algorithm based method allows speedups of up to fifteen seconds compared to the corresponding sequential implementation. This speed up can also be enhanced when I increase the number of atoms and the number of time steps.

The energy or force calculation is the most time using part of almost all Molecular dynamics Simulation. If I take a model system with pairwise additive interaction (as done in many molecular simulations), I have considered the contribution to the forces on the particle I by all its neighbors. If I do not shorten the interaction, this implies that, for a system of N particles, I must evaluate N(N-1)/2 pair interaction. Even if I shorten the potential, I still would have to compute all N(N-1)/2 pair distances to define which pairs can interact. This implies that, if I use no tricks, the time needed for assessment of the energy scales as N2 . There exist efficient techniques for speeding up the assessment of both short-term and long term contact in such a way that the computing time scales as N3/2, rather than N2 . The technique which I have used is a combination of Verlet and cell lists.

The first question that rises is when to use which method. This depends mostly on the details of the system. In any event, I always start with an arrangement as simple as possible, hence no trick at all. Although the algorithm scales as N2 , it is straightforward to implement and thus the probability of programming errors is relatively small. In addition I should consider how often the program will be used. The use of verlet list become beneficial if the number of particles in the list is significantly less than the total number of particles in three dimension this means **$nv = (4/3)*\pi r3\ \rho << N$**

Hence with this I am able to not only provide an application that's unique but also something that is deeply beinfited by use of parallel processing with the average difference in a result for (~12) processors being 7 seconds

Test cases as mentioned in the demo video:

Number of processes	Walltime (main computation) Parallel	Parallel and boltzmann constant	Not parallized/ without parallel processing
5	3.4 sec	5.1 sec	8.6 sec
	2.9 sec	3.8 sec	10 sec
	3.9 sec	4.5 sec	9.2 sec
	3.5 sec	4.2 sec	8.4 sec
	3.4 sec	5.9 sec	9.6 sec
	3.1 sec	4.7 sec	10.9 sec
10	7.5 sec	8.0 sec	13.9 sec
	7.9 sec	9.2 sec	14.6 sec
	8.2 sec	8.9 sec	14.9 sec
	7.1 sec	9.3 sec	14.7 sec
	6.6 sec	9.3 sec	15.6 sec
	8.1 sec	10.3 sec	13.0 sec
15	13.8 sec	16.0 sec	20.2 sec
	15.1 sec	16.4 sec	21.9 sec
	13.0 sec	15.7 sec	24.2 sec
	14.1 sec	16.8 sec	21.7 sec
	13.6 sec	14.9 sec	21.5 sec
	13.8 sec	16.1 sec	22.1 sec

Demo Video Link:

Review-2: Theoretical part + old code demo (link below consists of 2 parts/videos)
https://drive.google.com/drive/folders/1so3oKLgWKlID_Kc1gN-VqTUenhiHLtJV?usp=sharing
Review-3: New cold comparison, new functions,inputs and novelty justification
https://drive.google.com/file/d/1Bre_OJjsM8UGIwPjzQ-IXcaHldiOT9FF/view?usp=sharing
(Sorry for some pauses)

References:

[1] Elucidating membrane protein function through long-timescale molecular dynamics simulation
Link: https://ieeexplore.ieee.org/document/5335057

[2] Adaptive Resolution Molecular Dynamics Simulation with Constant in H
Link: https://ieeexplore.ieee.org/document/8776732

[3] Fabrication Sub-10nm Metallic Gratings with Carbon Nanotube – A Study by Molecular Dynamics Simulation Method
Link: https://ieeexplore.ieee.org/document/6017499

[4] Parallelization of Molecular Dynamics code
Link: https://ieeexplore.ieee.org/document/6621390

[5] Structural and Dynamical Properties of Nanographene Molecular Wires: a Molecular Dynamics Study
Link: https://ieeexplore.ieee.org/document/7388737

Molecular Dynamics Simulation
Link: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4655909/

Drive Link to all Papers referred:
https://drive.google.com/drive/folders/11VKG9z2KqIE9cJjC6PptyX26csiaNTQ1?usp=sharing