Work Assignment 1 - Optimization Techniques Applied To a Single Threaded Molecular Dynamics Simulation Code

Hugo Marques
Universidade do Minho
Braga, Portugal
pg47848@alunos.uminho.pt

Nuno Mata
Universidade do Minho
Braga, Portugal
pg44420@alunos.uminho.pt

***Abstract*—This paper investigates the application of diverse optimization techniques to a single-threaded Molecular Dynamics Simulation program. The computational demands of such simulations are often substantial, necessitating the need for improved algorithms and software implementations in order to manage hardware resources more efficiently. As such, the focus of this paper is to study the impact surrounding the integration of optimization techniques that result in improvements in computational efficiency and execution speed. The paper aims to highlight the critical role of the implementation of such optimization techniques in accelerating the performance of computationally demanding programs and to better leverage the capabilities of modern processor and memory architectures. The experimental results demonstrate significant performance gains enabling the Molecular Dynamics Simulation program to achieve much lower execution times while maintaining computational efficiency. These findings further emphasize the crucial role of tailored optimization strategies in accelerating the execution of computationally demanding programs, thereby facilitating their applicability in a wider range of computational resources.**

***Index Terms*—optimization, single-thread, computational efficiency, parallelism, memory hierarchy, data structures, vectorisation, loop unrolling**

I. Introduction

Molecular dynamics simulations programs are computational techniques used to study the behavior of atoms and molecules over time. These simulations have been conducted on computers rather than in real life due to several practical and technical reasons, such as: **Scale and Complexity**, since systems studied in molecular dynamics simulations often involve a large number of particles, and creating physical setups to mimic such complex systems is practically unfeasible. Computers also allow for **Precise and Controlled** manipulation of various parameters, including temperature, pressure, and external forces which enable researchers to explore a wide range of conditions. The **Cost and Time** of real-life experiments can be extremely expensive and time-consuming, especially when dealing with intricate molecular interactions. Alternatively, molecular dynamics simulations provide a cost-effective and time-efficient solution. Lastly, another big factor is **Observability** since computer simulations offer a unique advantage by providing insights into the atomic and molecular processes that are challenging to observe directly in real experiments.

In conclusion, the simulations aim to provide insights into the dynamics and properties of materials at the atomic and molecular levels, including their structural changes, thermodynamic properties, and behavior under different conditions. By simulating the trajectories of particles under the influence of inter-atomic or inter-molecular forces, it's possible to better understand concepts such as phase transitions, diffusion, chemical reactions, and complex molecular interactions.

II. Explaining The Molecular Dynamics Simulation Algorithm

The program provided with this assignment aims to simulate a molecular dynamics system for a noble gas (Helium, Neon, Argon, Krypton, or Xenon). It initializes a set of particles with specified positions and velocities, and then updates their positions and velocities over a series of time using the Velocity Verlet algorithm (the velocity Verlet algorithm provides both the atomic positions and velocities at the same instant of time). The program then calculates various thermodynamic properties referent to the simulation environment, such as temperature, pressure, kinetic energy, potential energy, mean squared velocity, gas constant, and compressibility (the measure of how much a given volume of matter decreases when placed under pressure). The program uses the Lennard-Jones potential (which is a simplified model that describes the essential features of interactions between simple atoms and molecules) to compute inter-molecular forces and accelerations of the particles. After executing the program, it generates two output files, one with all the calculated values for the physical properties surrounding the environment in the executed simulation and another with just the averages of all the previous values. One more thing to mention is that the simulation initially takes 3 user inputs, the noble gas to use in the simulation, the initial temperature in Kelvin and lastly the number density in $moles/m^3$. Finally, throughout the simulation, in the CLI (command line interface), there gets printed some information: Prompts asking for the initial parameters if they weren't provided via text file, the overall

simulation progress percentage towards the completion, and the results stored in the averages files mentioned previously.

III. Optimization techniques

In this section of the paper, the optimization techniques that were implemented in the optimized version of the code are demonstrated. In the optimized version, several optimization techniques were employed to improve the overall efficiency and execution time, such as:

- **Mathematical optimization** - The expression `L = cbrt(Vol)` was added using the `cbrt` (cube root) function from the `<cmath>` library, which is more efficient than calculating the cube root manually.
- **Calculation Optimization** - In the optimized version, calculations such as " double quot 2= quot * quot " and "$f = 24 \times \left(\frac{2}{rSqd^8} - \frac{1}{rSqd^4} \right)$" have been introduced in order to reduce the number of repetitive multiplications.
- **Changes to the Loop Structure** - Some loops have been modified/restructured in order to try to improve the efficiency of the loops, to take advantage of data locality or to reduce the total number of iterations.
- **Memory Hierarchy** - No specific techniques (like blocking) are used to exploit cache memory. The arrays are accessed in a fairly straightforward manner. However, some local variables are used in loops, which can be beneficial for cache usage.
- **Data Structures Organization** - The original code already has two-dimensional arrays that are used to store information about the particles, such as position (r), velocity (v) and acceleration (a). This is an efficient organization, as it allows contiguous access in memory, which is beneficial for cache performance.
- **Efficient Arithmetic Operations** - In the `Potential` function, instead of using `pow` for computing power, as in the the original version, multiplication is used, which is computationally cheaper.
- **Use of Efficient Algorithms** - The Gaussian distribution number generator is used for initializing velocities.
- **Compiler Flags** - After experimenting with many compilation flags we achieved what performed best for our case which improved the performance of compiled code.

We ended up implementing some optimization techniques, but there is room for further improvement, especially in terms of exploiting modern hardware features such as vectorisation, parallelization and the implementation of parallelization libraries.

IV. Tests and Results

This section presents a study to the improvements achieved in the developed MD simulations optimised code. We will compare the tests results between the base code provided with the assignment *vs* our optimized version. As for the tests performed the main focus is going to be the analysis of performance statistics since we mainly followed the Lab Practical Sessions learnings and applied these techniques in order to study the performance of each version. These testing techniques are based on execution flags that enable the output of statistics when the program is executed. Lastly we will be able to understand the improvements achieved after implementing the techniques mentioned in the previous section.

Tables bellow introduce the results of many tests performed in the cluster and the values written are the average values across all the documented tests.

All the testing was done in the university's cluster, which has got a Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz CPU Model.

TABLE I
Solution Tests: Base *vs.* Optimised

Simulation Version	**Cluster Testing Results**		
	#CC	***#I***	***Inst per Cycle***
Base	335 877 million	623 371 million	1.86
Optimized	19 923 million	28 236 million	0.96

TABLE II
Solution Tests: Base *vs.* Optimised

Simulation Version	**Cluster Testing Results**		
	CPI	***L1_DMiss***	***Execution Time***
Base	0.8	2 579 million	133.1s
Optimized	1.2	349 million	6.01s

Looking at the tables, we can observe several significant improvements, including:

- 95% faster execution time;
- 86% lower number of cache misses meaning the optimised version is utilizing the cache more effectively;
- 95% lower number of instructions which can be an indicating factor of a reduction of redundant or unnecessary calculations that could have been minimized or eliminated.

V. Conclusion

In this study, we tackled the optimization of a molecular dynamics simulation implementation. Through the initial code analysis, we pinpointed several potential areas for enhancement and introduced specific optimizations mentioned in section II.

While the original version of the code was functional and provided accurate results, the optimizations introduced brought about some significant improvements in computational efficiency, as verified in the results presented previously. However, it was observed that additional optimization opportunities remain. Techniques such as the implementation of neighbor lists, enhanced vectorization and the use of dedicated parallel libraries such as openmp and cuda, could be further explored in future works.

In conclusion, with the comparison of the tests between the base version and the optimized version, we can observe significant improvements not only in terms of performance but also in the efficiency of the use of computational resources, and therefore, we are pleased with the result of our work.

REFERENCES

[1] Abraham MJ, Murtola T, Schulz R, Páll S, Smith JC, Hess B, and Lindahl E (2015). GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX 1, 19–25. [Google Scholar]

[2] K. Chen and K. Zeng, "Performance Optimization Model of Molecular Dynamics Simulation Based on Machine Learning and Data Mining Algorithm,". Department of Basic Science, Jiaozuo University, Jiaozuo 454000, Henan, China.