Assignment 2
Particles Movement Simulator using MPI

CS3210 – 2019/20 Semester 1

October 25, 2019

Learning Outcomes

Assignment 2 is designed to enhance your understanding of parallel programming with distributed memory (MPI). Apply a different parallelization model to implement the particle movement simulation.

Problem Scenario

In Assignment 1, you have implemented and parallelize a particle movement simulator using shared memory (OpenMP) and GPU programming (CUDA). In assignment 2, you need to implement in parallel the same particle movement simulator using MPI, and compare with your shared memory programs in terms of parallelization strategy and performance.
The requirements for the particle movement simulator are the same with those described in Assignment 1, sections on Problem Scenario, Input and Output, and Physics engine. Instead of implementing the particle movement simulator with OpenMP and CUDA, you are now tasked to model the particle movement simulator as a distributed memory system and implement it with MPI.

Reflection on Assignment 1 Implementation

Together with this assignment, we are releasing a few simple test cases and a particle visualizer that can be used to check the correctness of your implementations. Observe the correctness of your implementations for OpenMP and CUDA before starting your MPI implementation. For comparison purposes, you might want to fix/adjust your OpenMP and CUDA implementations if needed. Up to 3 bonus marks will be awarded if you are successful in fixing your bugs, depending on the extent of the fixes.
For comparison purposes, you are allowed to change your implementation for OpenMP and CUDA for a better comparison of results of the same strategy across various implementations. Up to 3 bonus marks will be awarded if you decide to change your Assignment 1 implementation, depending on the extent of the changes. (Note: at most 3 bonus marks in total can be accumulated for changes made to your Assignment 1).
Requirements for Assignment 1 mention that the collisions can happen anytime during the step. But some parallelization approaches discretized the simulation by assuming that collisions happen only in the beginning of a step, followed by movement the particles. Note that you are not expected to completely change the assumptions you have made in assignment 1 as long as your simulation looks visually accurate (no deduction will be applied if you keep the same assumptions for your MPI implementation). However, you should try to minimally fix your implementation if your implementation is visually incorrect.

Parallelization Approach for Assignment 2

The first approach that comes to mind might be to just port your solution from OpenMP to MPI (or CUDA to MPI). However, you must be aware of the different programming model, and try to make your implementation more efficient.

Many students have chosen to parallelize the detection of collision (i.e one thread per particle) for their Assignment 1. While this can be directly translated into MPI (i.e, one process per particle for collision detection), your implementation will not be efficient due to communication overhead. Hence, you should try to find a better suited parallelization approach for MPI.
The challenging part is to come up with a parallel implementation that scales when increasing input size, number of processes and hardware capabilities. As such, you might need to try several approaches to parallelize the algorithm and several parallel implementations. You are advised to retain your alternative implementations and explain the incremental improvements you have done. Clearly state in your README and report which is the approach you used for measurements (and you want to be graded on).
Additional Note: For your midterm, you have seen a method to increase the efficiency of finding nearby bodies in the N-body simulator by using a quadtree. Note that this is only valid because of spatial locality – in that case, we assume that there are non-negligible gravitational force between bodies when they are within a certain radius of each other. This is not true for the particle movement simulator – it is possible for a particle to collide with another particle far away from itself, provided that the velocity is high enough. Do not assume that spatial locality holds when you are attempting your parallelization of the particle movement simulation.

Running your Simulation

For MPI, run your experiments varying the input size (number of particles, surface, etc), the number of processes used, and the number of machines. For performance measurements, run each simulation at least 3 times and take the shortest execution time. You can use any number of machines from the lab for your measurements:

- Dell Optiplex 7050 (Intel Core i7-7700)
- Dell Precision 7820 (intel Xeon Silver 4114)

FAQ

Frequently asked questions (FAQ) received from students for this assignment will be answered in this file. The most recent questions will be added at the beginning of the file, preceded by the date label. Check this file before asking your questions.

If there are any questions regarding the assignment, please post on the LumiNUS forum or email Rong Hua (ronghualui@u.nus.edu).

Assignment Submission instructions

Similar to Assignment 1, you are allowed to work in groups of maximum two students for this assignment. You can discuss the assignment with others as necessary but in the case of plagiarism both parties will be severely penalized. The assignment has to include the code for OpenMP, CUDA, MPI, a readme file explaining how to compile and run each program, and a report.
Assignment 2 is due on **Mon, 18 Nov 2019, 11am**. You may get a maximum of 10 marks for completing this assignment, and an additional 3 bonus marks for changes made to your Assignment 1 code. The bonus marks count towards your assignment marks (and cannot be used to increase marks for your other grade components).
Your report should include:

- Explanation of your parallelization approach for MPI (how it works) and implementation assumptions.
- Explanation of your parallelization approach for OpenMP and CUDA. Point out any changes made from your Assignment 1 submission.
- Comparison of your parallelization approaches for MPI, OpenMP and CUDA.
- Performance measurements for your MPI implementation using the machines in the lab. Use an increasing number of machines (and core/processes) to run your implementation and observe the point when the performance improvement is slowed down by the communication overhead.
- Performance comparison among the implementations in MPI, OpenMP, and CUDA. Compare the best performance obtained (for the same input) for each implementation, and explain your observations.

There is no minimum or maximum page length for the report. Be **comprehensive**, yet **concise**.

Submit your assignment before **Mon, 18 Nov 2019, 11am** under LumiNUS Files. Each student must submit one zip archive named with your student number(s) (A0123456Z.zip - if you worked by yourself, or A0123456Z_A0173456T.zip - if you worked with another student) containing:

1. Your C/C++ code using MPI along with three sample input files, and the output produced by running your code on these files.
2. Your implementations for OpenMP and CUDA.
3. `README` file, minimally with instructions on how to **compile** and **execute** all your code submissions. Mention the machines (computers) you used to run your code.
4. Report in PDF format (a2_report.pdf)

Note that for submissions made as a group, only one most recent submission (from any of the students) will be graded, and both students receive that grade.
Penalty of 10% per day for late submissions will be applied.