Assignment 5: N-Body

Due: Fri., 2025/11/28 23:59

Problem Description

In this assignment, you will touch on the simulation of a dynamical system of particles under the influence of gravity. The dataset is an imitation of our Milky Way galaxy. In this homework, you are asked to use ROCMs with 2 GPUs to perform the simulation. For a more detailed description of the problem, please refer to the assignment slide.

You can implement the parallel code with a brute force algorithm. No need to implement Barnes-Hut Algorithm.

Formulas & Parameters

Gravity Simulation:

1. Calculating the force vector on the body i caused by the body j:

$$f_{ij} = \frac{Gm_i m_j}{\|q_j - q_i\|^2} \cdot \frac{q_j - q_i}{\|q_j - q_i\|} = \frac{Gm_i m_j (q_j - q_i)}{\|q_j - q_i\|^3}$$

2. Calculating the total force vector on the body i caused by other $N - 1$ bodies:

$$F_i = \sum_{j \neq i} f_{ij} = \sum_{j \neq i} \frac{Gm_i m_j (q_j - q_i)}{\|q_j - q_i\|^3}$$

3. To avoid the gravitational force growing indefinitely in situations when two bodies come too close, we add a softening factor ϵ:

$$F_i \approx \sum_{j \neq i} f_{ij} = \sum_{j \neq i} \frac{Gm_i m_j (q_j - q_i)}{(\|q_j - q_i\|^2 + \epsilon^2)^{\frac{3}{2}}}$$

4. Update velocities by:

$$v_i(t) = v_i(t - \Delta t) + a_i(t) \cdot \Delta t$$

5. Update position by:

$$q_i(t) = q_i(t - \Delta t) + v_i(t) \cdot \Delta t$$

Parameters:

- Simulation steps: $200,000$
- Time difference between each time step (sec): $\Delta t = 60$
- Softening factor (m): $\epsilon = 10^{-3}$
- Gravity constant ($m^3 kg^{-1} s^{-2}$): $G = 6.674 \times 10^{-11}$

- Gravity devices' mass (kg): $m(t) = m(0) + 0.5 \cdot m(0) \cdot |sin \frac{t}{6000}|$
- The asteroid hits the planet if their distance at a time step is below (m): 10^7
- Missile traveling speed (m/s): 10^6
- Missile cost: $10^5 + (step + 1) \times \Delta t \times 10^3$

All the units given above are SI units, so you do not need to do the conversion. (If you find otherwise, it is a bug, please let us know.)

Input Format

The input is a text file looking like this:

```
N planet-id asteroid-id
qx0 qy0 qz0 vx0 vy0 vz0 m0 type0
qx1 qy1 qz1 vx1 vy1 vz1 m1 type1
qx2 qy2 qz2 vx2 vy2 vz2 m2 type2
…
```

The first line of the input file contains three integers, which are:

1. N: the number of celestial bodies
2. ID_{planet}: the (0-indexed) ID of the planet
3. $ID_{asteroid}$: the ID of the asteroid

Then there are N lines following, each containing 8 values:

1. $q_{xi}(0)$, $q_{yi}(0)$, $q_{zi}(0)$: the initial position of the body i (`double` format)
2. $v_{xi}(0)$, $v_{yi}(0)$, $v_{zi}(0)$: the initial velocity of the body i (`double` format)
3. $m_i(0)$: the initial mass of the body (`double` format)
4. $type_i$: the type of the body (`string` format)

Output Format

The output is a text file looking like this:

```
min-dist
hit-time-step
gravity-device-id missile-cost
```

- The first line is the minimal distance between the asteroid and the planet if there were no gravity devices. This value is considered correct if the relative error is within 10^{-8}.
- The second line is the first time step that the asteroid hit the planet. If the asteroid did not hit the planet, output -2. This value is considered correct if the absolute error is within $\pm$ 1.
- The third line contains two numbers: `gravity-device-id` is the ID of the gravity device that we want to destroy, `missile-cost` is the cost of the missile. If there is no need to destroy the gravity devices, output -1 0. If destroying one gravity device could not prevent the asteroid from hitting the planet, output -1 0 as well. missile-cost is considered correct if the absolute error is within 6×10^4.

Compilation

In this homework, we use GPUs that are provided by AMD.

The `Makefile` is used to build your code. The default `Makefile` for this homework is located at `/home/pp25/share/hw5/sample/Makefile`.

If you wish to change the compilation flags, please include `Makefile` in your submission.

To use `Makefile` to build your code, make sure `hw5.cpp` is in the working directory, then run `make` on the command line and it will build `hw5` for you. To remove the built files, run `make clean`. The `make` command is used to compile your code.

Sequential Code:
The provided directory for this assignment is located at `/home/pp25/share/hw5/`. You can first copy the folder to your root directory with the following command:

```
cp -r /home/pp25/share/hw5 $HOME/hw5
```

The example sequential code can be found in the samples folder in the hw5 directory.

Please note that in contrast to the previous assignments, this provided sequential code is incomplete, and can not be executed directly.

Execution

Your code will be executed with a command as follows:

```
srun -t 00:10:00 --gres=gpu:2 ./hw5 input_file output_file
```

Validation

To ensure that your implementation is correct, we provide a validation file `validate.py` under the hw5 directory. The following gives an example usage of the validation file. Suppose you have finished the implementation and compiled your code to be a binary file hw5 and your directory is structured as:

```
hw5
|-- samples
|      |-- Makefile
|      |-- hw5.cc
|      |-- hw5
|
|-- testcases
|      |-- b20.in
|      |-- b20.out
|
|-- validate.py
|-- parse.py (used by validate.py)
```

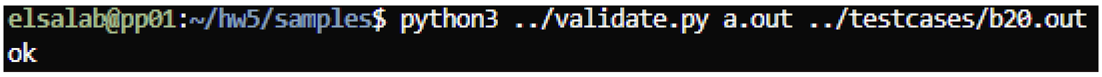
You can execute the program with the following command if you are under the samples folder:

```
srun -t 00:10:00 --gres=gpu:2 ./hw5 ../testcases/b20.in my_output.out
```

The answers to the required problems will be saved into the output file *my_output.out*.
To ensure that your answer is correct, you can check it with:

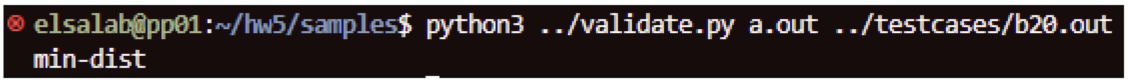
```
python3 ../validate.py my_output.out ../testcases/b20.out
```

If the answers are correct, a single line 'ok' will be outputted. Otherwise, the incorrect field will be shown in the terminal. The examples are:
Correct:



```
elsalab@pp01:~/hw5/samples$ python3 ../validate.py a.out ../testcases/b20.out
ok
```

Incorrect:



```
elsalab@pp01:~/hw5/samples$ python3 ../validate.py a.out ../testcases/b20.out
min-dist
```

Output Verification

The `hw5-judge` command can be used to automatically judge your code against all sample test cases, it also submits your execution time to the scoreboard so you can compare your performance with others. Scoreboard link: link

After four homework, we believe that you are familiar with it d(`･∀･)b. To use it, run `hw5-judge` in the directory that contains your code `hw5.cpp`. It will automatically search for `Makefile` and use it to compile your code, or fallback to the provided `Makefile` otherwise. If code compilation is successful, it will run all of the sample test cases, show you the results as well as update the scoreboard.

Please note once again that `hw5-judge` and the scoreboard have nothing to do with grading. Only the code submitted to NTU Cool is considered for grading purposes.

To check the command `hw5-judge`, please type `hw5-judge --help` to see supported options.

Report

Your report must answer the following questions in either English or Traditional Chinese, please include your name and Student ID.

1. Implementation:
   - What is your parallelism strategy?
   - Optimization techniques?
   - How do you manage the 2-GPU resources?
2. If there are 4 GPUs instead of 2, what would you do to maximize the performance?
3. If there are 5 gravity devices, is it necessary to simulate 5 n-body simulations independently for this problem?
4. (Optional) Any suggestions or feedback for the homework is welcome

Grading

1. (50%) Correctness. Proportional to the number of test cases solved.
2. (20%) Performance. Based on the total time you solve for all of the test cases.
3. (30%) Report.

Submission

Please zip the following files to a single archived file named **<StudentID>.tar**:

- `hw5.cpp` – the source code of your implementation.
- `report.pdf` – your report.
- `Makefile` – optional. Submit this file if you want to change the build command.

Please note that the first character of your student ID in the file name **MUST** be lowercase, and you should run the archiving command on the directory (also in lowercase), not the files themselves. **Any submission violation or compilation error would lead to a point deduction.**

For instance, the correct way to archive your files is as follows:

```
mkdir b10000000
cp <your files> b10000000/
tar cvf b10000000.tar b10000000
```

Submit the file `b10000000.tar`

Appendix

Sample Test Cases

The sample test cases are located at `/home/pp25/share/hw5/testcases`.