CS3210: Parallel Computing – AY 2015/2016 Semester I

Assignment 2: N-body Simulation on Distributed-memory Systems
Individual Submission (20 marks)

Deadline: 15 Nov 2012

The objective of this assignment is to reinforce the learning of developing a parallel solution for a given problem using the message-passing model on distributed-memory systems. In physics and astronomy, an ***N*-body simulation** [1] is a simulation of a dynamical system of particles, usually under the influence of physical forces. In this assignment, you will develop and optimize a parallel N-body simulator for a Universe of planets and stars. In simulating the interactions of particles (bodies), you should apply the steps in Foster's model, namely, partitioning, communication, agglomeration and mapping.

To start, we have included a sequential N-body simulator program (nbody-seq.c), a *makefile* for the sequential program, input (*universe.in*) and output (*universe.out*) files for the Solar system, a bash script (*plot_universe.sh*) to visualize the planetary positions for the Solar system example, and a program (*nbody-gen.c*) to generate an input file with a larger number of objects.

Background of N-body Simulation of the Universe

For large systems consisting of many objects (bodies), it is impractical or impossible to write a closed-form equation to express the interactions between them. For example, astronomers are interested in the movement (velocity, position) of astronomical bodies, such as stars, planets, meteorites. But since it's tedious to write a closed-form equation for this, simulations are used where the velocities and positions are computed in discrete time steps. In this assignment, you will help astronomers to write such a simulator.

Gravity is the main force in universe and dictates complex interactions between objects. For simplicity, this assignment considers a 2D (rather than 3D) space and Newton's law of gravity [2] (rather than the theory of relativity). This law states that two bodies *i* and *j* with masses m_i and m_j and a distance of $r_{i,j}$ between them, attract each other with a force:

$$F = G\frac{m_i m_j}{r_{i,j}^2}$$

where G is the gravitational constant ($G = 6.673 \cdot 10^{-11}$ N · $(m/kg)^2$).

Given *N* bodies with masses m_1, m_2, …, m_N, there are *N(N-1)* gravitational forces in the system. The *N-1* forces acting on a body *i* are added to get the final force F_i and the acceleration of body is computed as:

$$a_i = \frac{F_i}{m_i}$$

Considering *T* time steps for the simulation, the velocity at time t_k (*k=1..T*) is computed as:

$$v_{i,t_k} = v_{i,t_{k-1}} + a_i dt$$

where $dt = t_k - t_{k-1} = \cdots = t_1 - t_0$ is the time step.

The forces, accelerations and velocities are vectors in a 2D space, thus, you need to decompose them into their *x* and *y* components. For example, given two bodies *i* and *j* placed at (x_i, y_i) and (x_j, y_j) with the Euclidian distance $r_{i,j}$, the force exercised by body *i* on body *j* on *x* axis, $Fx_{i,j}$ is:

$$Fx_{i,j} = F_{i,j} \frac{(x_j - x_i)}{r_{i,j}}$$

Finally, the position of body *i* is updated:

$$(x_i, y_i)_{t_k} = (x_i, y_i)_{t_{k-1}} + \overrightarrow{v_{i,t_k}} dt$$

N-body Sequential code

The sequential N-body simulator consists of three loops: (i) a main loop on the time steps, (ii) a loop for each of the N objects to get its next state and (iii) a loop on the other N-1 objects to get the forces they exercise on the object in second loop. You are provided with the C program (*nbody-seq.c*) implementing the sequential N-body simulator for the Universe. You may use this program as a starting point for the parallel program and to verify the results.

Assignment

Write an MPI program that simulates the interaction between N bodies. **The program must run on the two-node cluster systems in our lab, and used all eight cores in first node and all 4 cores in second node.** To test the correctness of your program, you will start by simulating the Solar system, for which you are given an example of input file and the expected output. However, in addition to ***correctness*** your final solution will be tested on a much larger N-body system. Hence, you have to design and implement your application such that it is ***scalable***. For this, you can use *Foster's model* for designing parallel programs [3]. This model consists of four steps:

1. *Partitioning* – how to split the dataset among parallel tasks. [**Hint**: You can have one object per task or multiple objects per task.]

2. *Communication* – how to send data among parallel tasks. [**Hint**: For the N-body simulation, each object needs to know the state of the other N-1 objects.]

3. *Agglomeration* – how to increase the granularity of tasks in order to improve the efficiency. [**Hint**: You can have one object per task or multiple objects per task.]

4. *Mapping* – how to assign the tasks to the processors. [**Hint**: In the case of an MPI program, there are two layers of abstraction: (i) MPI logical process and (ii) physical CPUs. Usually, for an N-body system and a computing platform with P processors, there are much more objects than processors ($N >> P$), thus, one can assign one or multiple tasks to a logical process, and one or multiple logical process to a CPU core.]

There are many decisions to be made during the design phase. You are required to state and explain how your decisions affect the performance of the parallel program. From each of the steps above, when possible quantify the performance improvements either in terms of the number of communication steps reduced, execution time, etc.

Input, Output and Details
Your program has to consider the following fixed input and output formats. The input file (*universe.in*) has on the first line two integer numbers: N – the number of bodies in the system to be simulated, and T – the number of time steps for the simulation. On each of the next N lines, there are five floating point numbers representing position in 2D, initial velocity in 2D and the mass of each body (object).

Input file *(universe.in):*
N T
x_0 y_0 v_{x0} v_{y0} m_0
…
x_{N-1} y_{N-1} v_{xN-1} v_{yN-1} m_{N-1}

The output file (*universe.out*) consists of N lines containing the final positions (i.e. after T time steps) of the bodies. All output numbers are floating points that should be printed using the scientific notation (in *printf*, use *%lE*). (***otherwise, 1 mark will be deducted***).

Output file *(universe.out):*
x_0 y_0
…
x_{N-1} y_{N-1}

Other details
The first body (index *0*) is considered fixed in the origin of coordinates systems (i.e. *(0,0)*) and its mass is much larger compared to the others. Hence, you can assume that its position remains unchanged.

The duration of one time step is set to be 60 seconds.

You are given the sequential N-body simulator (*nbody-seq.c*), a *makefile* for the sequential program, input and output files (*universe.in/universe.out*) for the Solar system, a bash script (*plot_universe.sh*) to visualize the planetary positions for the Solar system example, and a program to generate random N-body systems (*nbody-gen.c*). To make use of these files:

```
$ make
$ ./nbody-seq
$ ./plot_universe.sh
```

Open *universe.in.pdf* and *universe.out.pdf* to see planetary positions.

To generate a new input file (*universe.in*) with N objects and T time steps:

```
$ make gen
$ ./gen <N> <T>
```

Implementation and Deliverables
Your program must run on a cluster formed by the two *x86* systems of each group in the lab. You need to upload to IVLE Folder Assignment 2, under Assignments directory a **zip archive** containing the **source code** and **a report in PDF format**. The report must contain:

1. A walkthrough of your design, highlighting the important design decisions and the explanation for your decisions.

2. A *makefile*, or instructions on how to build and run your program, as well as the configuration (*machinefile* or *rankfile*) needed to run the program (*failure to provide these comes with a 4 marks penalty*).

The source code must be properly commented (*2 marks penalty*) and indented (*2 marks penalty*).

Deadline is 15 November 2015, 23:59. Penalty for missing the deadline is 2 marks per day.

Resources

[1] https://en.wikipedia.org/wiki/N-body_simulation

[2] https://en.wikipedia.org/wiki/Newton%27s_law_of_universal_gravitation

[3] I. Foster, *Designing and Building Parallel Programs*, Addison Wesley, 1995, http://www.mcs.anl.gov/~itf/dbpp/text/book.html

[4] MPI Tutorial, https://computing.llnl.gov/tutorials/mpi/