CS542200 Parallel Programming
Homework 2: Single Roller Coaster Car Problem And N-Body Problem

Due: November 15, 2015

1 GOAL

This assignment helps you to get familiar with Pthread and OpenMP by implementing 2 problems - Single Roller Coaster Car Problem and N-body Problem.

Pthread and OpenMP are shared memory programming tools. You will practice how to parallelize these two questions using Pthread and OpenMP, and prevent synchronization problem. The working items are as below:

1. Simulate Single Roller Coaster Car Problem and prevent synchronization by using conditional variable or mutex lock.
2. Parallelize N-body's sequential code by using Pthread and OpenMP.
3. Implement Barnes-Hut Algorithm by Pthread.
   - Parallel building tree phase
   - Parallel simulation phase
4. Compare the performance of those N-body versions and sequential code.

2 SINGLE ROLLER COASTER CAR PROBLEM

2.1 PROBLEM DESCRIPTION

Suppose there are *n* passengers and one roller coaster car. The passengers repeatedly wait to take rides in the car, which holds *C* passengers. However, the car can go around the track only when it is full. The car takes the same number of milliseconds (*T*) to go around the track each time it fills up. After getting a ride, each passenger wanders around the amusement park for a random amount of time before returning to the roller coaster for another ride. Write a program using Pthread to simulate this problem. The program should ask for *n*, *C*, *T* and *N*, and generate *n* passenger threads and one roller coaster car thread. The program should exit after roller coaster car going around *N* times.

- In the passenger threads, you need to simulate as described below.
  1. Passenger wanders around the amusement park for a random amount of time. (use usleep() function to wait for a random time).

2. Then, the passenger will return for another ride.
3. Repeat step 1 and 2 until program exit.

- In the roller coaster car thread, you need to simulate as described below.
  1. Repeatedly checking if there are *C* passengers in the queue.
  2. When there are *C* passengers in the queue, the car will take *T* milliseconds to go around.(use usleep() function too, but wait for *T* milliseconds)
  3. Then release passengers.
  4. Repeat step 1, 2 and 3 *N* times and exit the program.

2.2 INPUT

./hw2_SRCC *n C T N*

- ./hw2_SRCC: your execution file
- *n*: number of passengers ($2 \leq n \leq 10$)
- *C*: capacity of car
- *T*: time for car going around the track, representing *T* milliseconds, is integer
- *N*: number of simulation steps

2.3 OUTPUT

- In the passenger threads,
  (a) When a passenger is going to wander around the amusement park, you should print something like "*3rd passenger wanders around the park*".
  (b) When a passenger returns for another ride, you should print "*3rd passenger returns for another ride*".
- In the roller coaster car thread,
  (a) When car is going to departure, you should print "*car departures at 15 millisec. 3rd, 5th and 6th passengers are in the car*".
  (b) When car arrives, you should print "*car arrives at 45 millisec. 3rd, 5th and 6th passengers get off*".

The sentences can be different, but the **passenger's ID number**, **car's departure and arrival time** and **who is on the car** should be printed.

3 N-BODY PROBLEM

3.1 PROBLEM DESCRIPTION

Giving *N* celestial bodies with the same mass *m*, each body has its initial position *(x, y)* and velocity *(vx, vy)*. Simulate their movement *T* times by using *t* seconds between each step.

3.2 INPUT

./hw2_NB_{version} *#threads m T t FILE* θ *enable/disable* x_{min} y_{min} *length Length*

- ./hw2_NB_{version}: your execution file
- *#threads*: number of threads
- *m*: mass, is double precision floating point number
- *T*: number of steps
- *t*: time between each step, is double precision floating point number
- *FILE*: input file name
- θ*:* use in Barnes-Hut algorithm
- *enable/disable*: enable/disable Xwindow
- x_{min} , y_{min}: the upper left coordinate of Xwindow
- *length*: the length of coordinate axis
- *Length*: the length of Xwindow's side (*Length* will be 10^n times of *length*)

You don't need to read the last 4 argument values if Xwindow is disabled.

Please use double precision numbers inside your simulation.

In the input file, the first line will contain a number $N(1 \leq N \leq 10^6)$, which means the number of bodies. For the following *N* lines, each line contains 4 floating point numbers $x_i,\ y_i,\ v_{xi},\ v_{yi}$ representing the i_{th} body's initial position and velocity.

3.3 OUTPUT

If configuration of Xwindow is "enable", show celestial bodies on Xwindow each step.

4 GRADING

Your grade will be judged by correctness, report and demonstration as described in below:

4.1 CORRECTNESS (50%)

During the demo time, TA will test your program with some test cases to check whether your output is correct.

- Single Roller Coaster Car Problem (10%)
- N-Body Problem – OpenMP version (10%)
- N-Body Problem – Pthread version (10%)
- N-Body Problem – Barnes-Hut Algorithm version (20%)

4.2 REPORT (30%)

1. Design

   Explain your implementation of Barnes-Hut algorithm.

   (a) How do you parallelize each phase of Barnes-Hut algorithm?

   (b) How do you partition the task?

   (c) How do you prevent from synchronization problem?

   (d) What technique do you use to reduce execution time and increase scalability?

   (e) Other efforts you've made in your program

2. Performance analysis of Single Roller Coaster Car Problem

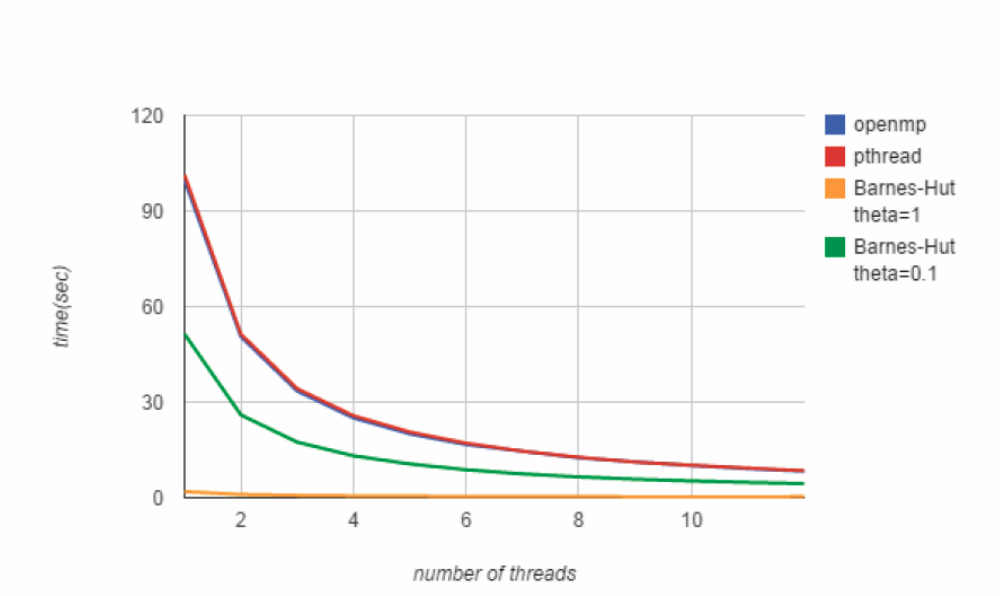
   (a) Plot the average waiting time vs the input parameter *C* or *T*

3. Performance analysis of N-Body Problem

   Compare three parallel versions (OpenMP, Pthread, Barnes-Hut Algorithm) and one sequential version. Plot some charts described below to show the performance of each version, and explain why causes the performance of these three versions different.

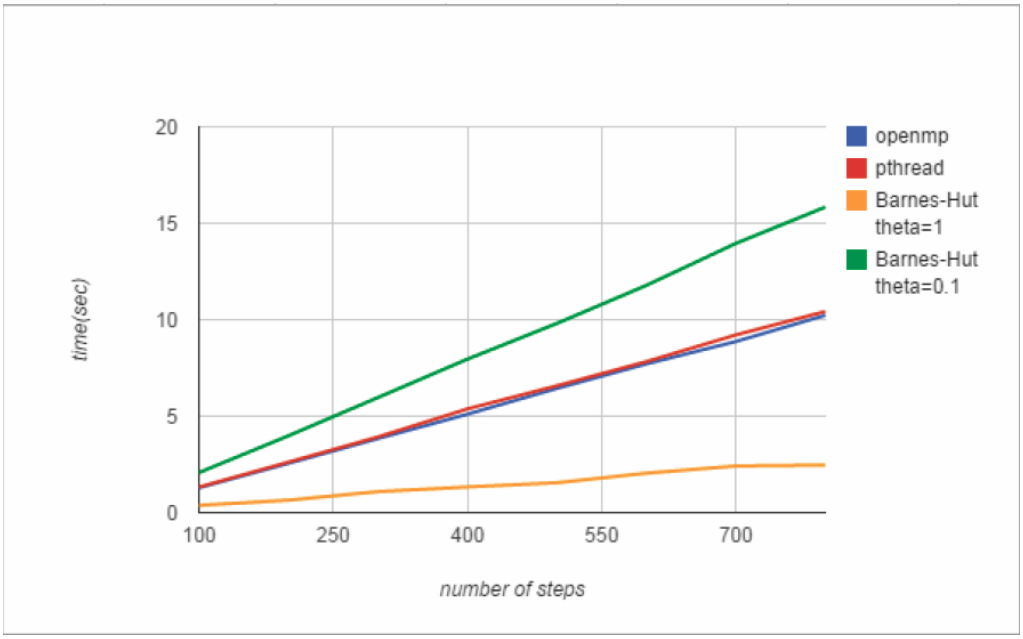
   (a) Strong scalability

      i. Fix *N*, *T*, and manipulates *#threads*. Compare execution time of each version.

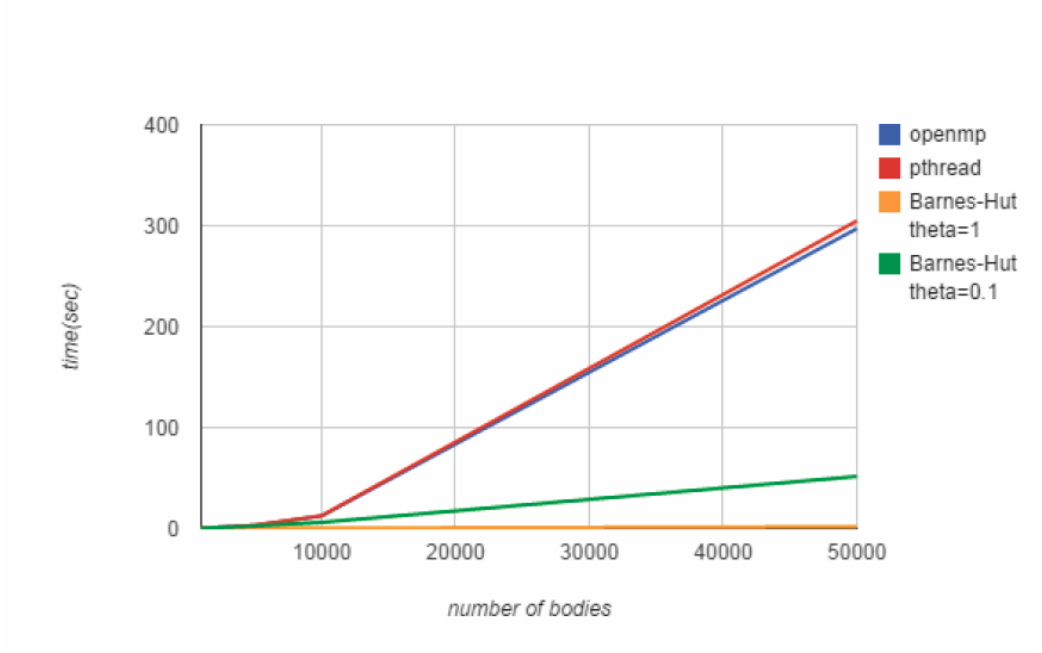


(b) Weak scalability

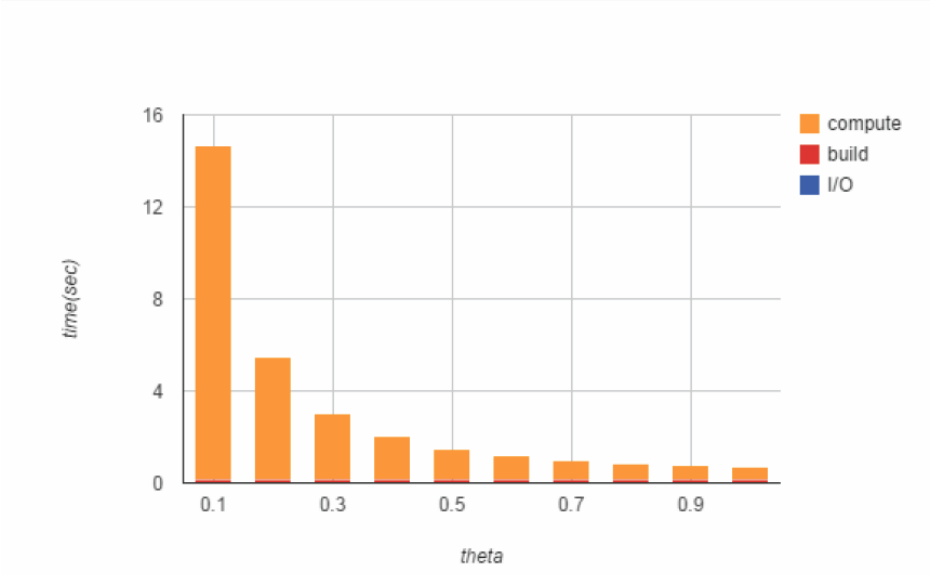
i. Fix *N*, *#threads*, and manipulates *T*. Compare execution time of each version.



ii. Fix *T*, *#threads*, and manipulates *N*. Compare execution time of each version.



(c) For Barnes-Hut algorithm, Show the time of three phases (I/O, building tree, and computing) based on different θ



(d) Other experiments you have done to compare the performance.

ps1. Please show your input settings below each experiment.

ps2. Please show the comparison between each version, plot them on the same chart, not just the scalability of individual programs.

4. Experience

(a) What have you learned from this assignment?

(b) What difficulty did you encounter when implementing this assignment?

(c) If you have any feedback, please write it here.

4.3 DEMO (20%)

- Test the correctness of your codes.
- Go through your codes. Make sure your implementation follows the requirements.
- We will ask some questions about your codes and report.

5 REMINDER

1. Here are some examples about compiling your code with OpenMP(-fopenmp), Pthread(-pthread), and Xwindow(-lX11).

   i、 Compiling your code with Pthread and Xwindow

   ```
   g++ -pthread -lX11 sample.cpp (gcc for XXX.c)
   ```

   ii、 Compiling your code with OpenMP and Xwindow

   ```
   g++ -fopenmp -lX11 sample.cpp
   ```

   iii、 After compiling, execute your program as ordinary c/c++ program.

   ```
   ./a.out arg1 arg2 ….
   ```

2. Please package your codes and report in a file named **HW2_{student-ID}.zip** and name your codes/report as follows:

   i、 Single Roller Coaster Car Problem: **hw2_SRCC.c (or .cpp)**

   ii、 N-Body Problem OpenMP version: **hw2_NB_openmp.c (or .cpp)**

   iii、 N-Body Problem Pthread version: **hw2_NB_pthread.c (or .cpp)**

   iv、 N-Body Problem Barnes-Hut Algorithm: **hw2_NB_BHalgo.c (or .cpp)**

   v、 Report: **hw2_report.pdf**

   Please write a makefile and package it with your codes or explain how to compile your codes in your report.

   And upload to iLMS before **11/15(Sun) 23:59**

3. Since we have limited resources for you guys to use, please start your work ASAP. Do not leave it until the last day!

4. Late submission penalty policy please refer to the course syllabus.
5. **Do NOT try to abuse the computing nodes by ssh to them directly**. If we ever find you doing that, you will get 0 point for the homework!
6. Asking questions through iLMS or email are welcomed!