Parallel Programming HW2
Single Roller Coaster Car Problem
and N-body Problem

TA: Tony Wei

Due on 2014/11/16

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. So 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. Parallel 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 seconds (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, then 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 sleep() 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, you need to simulate as described below.
  1. Repeat to check if there are C passengers in the queue.
  2. When there are C passengers in the queue, the car will take T seconds to go around.(use sleep() function too, but wait for T seconds)
  3. Then release passengers.
  4. Repeat step 1, 2 and 3 N times and exit the program.

2.2 Input

./a.out n C T N

- a.out: your execution file
- n: number of passengers ($2 \leq n \leq 10$)
- C: capacity of car
- T: time for car going around the track, represent T millisecond, is a integer.
- N: number of simulation step

2.3 Output

- In the passenger threads,
  1. When passenger is going to wander around the amusement park, you should print something like "3rd passenger wander around the par".
  2. When passenger returns for another ride, you should print "3rd passenger return for a ride".
- In the roller coaster car,
  1. When car is going to departure, you should print "car departure at 15 sec. 3rd, 5th and 6th passengers are in the car".
  2. When car arrives, you should print "car arrives at 45 sec. 3rd, 5th and 6th passengers get off".

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

3 N-body Problem

3.1 Problem Description

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

3.2 Input

./a.out *#threads* m T t $FILE$ θ *enable/disable* x_{min} y_{min} *length Length*

- a.out: your execution file
- *#threads*: number of threads
- m: mass, is a float number
- T: number of steps
- t: time between each step, is a float number
- $FILE$: input file name
- θ: use in Barnes-Hut Algorithm
- *enable/disable*: enable or disable Xwindow
- x_{min}, y_{min}: the lower left coordinate of Xwindow
- *length*: the length of the coordinate axis
- *Length*: the Length of Xwindow's side
  Length will be 10^n times of *length*.

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

In the input file, the first line will contains a number N ($1 \leq N \leq 10^6$), which means the number of bodies. Following N lines, each line will contains 4 float numbers x_i, y_i, v_{xi}, v_{yi}, ($1 \leq x_i, y_i, v_{xi}, v_{yi} \leq 100$), representing i_{th} body's initial position and velocity.

3.3 Output

If configuration of Xwindow is "enable", show celestial bodies on Xwindow at 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, TAs will test your program with some testcases 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 about Barnes-Hut Algorithm.
   (a) How do you parallel each phase on 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 do in your program.
2. Performance Analysis on Single Roller Coaster Car Problem
   (a) plot the average waiting time v.s. the input parameters C or T.
3. Performance Analysis on 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 about each version, and explain why cause these versions' performance different.
   (a) Strong Scalability
      i. Fix N, T and manipulate $\#threads$. Compare execution time with each version.
   (b) Weak Scalability
      i. Fix N, $\#threads$ and manipulate T size. Compare execution time with each version.
      ii. Fix T, $\#threads$ and manipulate N size. Compare execution time with each version.
   (c) Barnes-Hut Algorithm: Show the time of 3 phases (I/O, building tree and computing) based on different θ.
   (d) Other experiments you do to compare the performance.

ps1. Show your input setting below each experiment.
ps2. Show the comparison between each version, 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, you can also write it here.

4.3 Demo (20%)

- Test your program's correctness.
- Go through you codes. Make sure that you codes are followed the requirement.
- Some questions about your programs and report.

5 Reminder

1. Late submission penalty policy please refer to syllabus.
2. Please compress all of your codes and report in the file named **${ID}_${name}_HW2.zip** and upload to iLMS before **11/16(Sun.) 23:59 pm**.
3. Name your codes clearly. For example:
   - Single Roller Coaster Car Problem: hw2-SRCC.cpp
   - N-body Problem OpenMP version: hw2-NB-openmp.cpp
   - N-body Problem Pthread version: hw2-NB-pthread.cpp
   - N-body Problem Barnes-Hut Algorithm version: hw2-NB-BHalgo.cpp
4. Because we have limited machines for you guys for tuning. Please start your work ASAP and do not leave it until the last day!