Low-level Parallel Programming (course 1DL550)
Uppsala University – Spring 2015
Report for Lab 3 by Team 14

Fredrik Larsson Jimmy Holm Per Bergqwist

10th March 2015

1 System specification

The CPU of the system used for gathering the data presented was an Intel i7 2600k running at a frequency of 4GHz. The system was able to use four cores with Hyper-threading enabled, meaning a possibility of using eight logical cores simultaneously with the drawback that the performance gain from using more than four cores varies depending on the tasks.

2 Plot

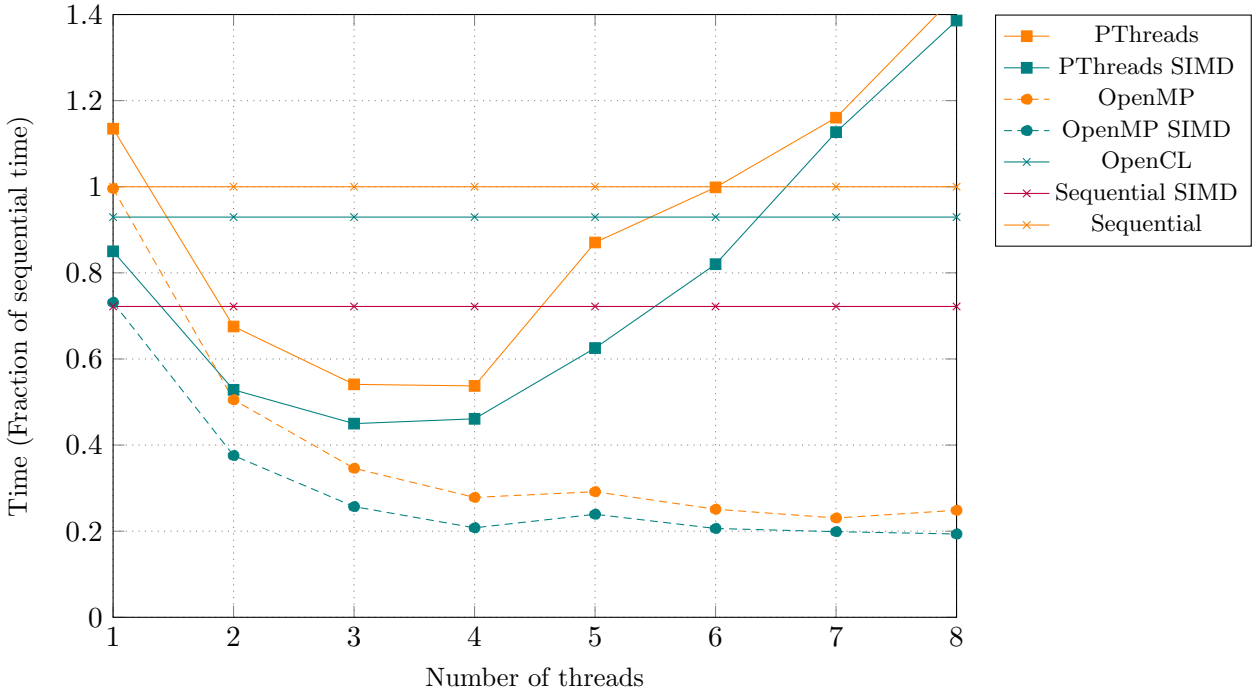

Figure 1: Normal scenario

3 Questions

A. **Is it possible to parallelize the given code with a simple omp parallel for (compare lab 1)? Explain**
No, that is not possible because the agents position changes during the loop, leading to race-condition.

B. **How would the global lock solution (sec 2.1) perform with increasing number of threads? Would the second simple solution perform better in this regard? Can the scenario affect the relative performance of the second simple solution?**
The global lock solution would stall every other thread during a change in agent position making it unacceptably slow. The second simple solution require no locking and movment inside a region produce no race-condition. However care must be taken when moving between regions. If all agents is within a single section it will perfom as sequential but with overhead.

C. **Is the workload distributed evenly across the threads?**
Yes, because of OpenMP thread-pool implementation the workload is distributed as evenly as possible. The code make use of OpenMP sections to create tasks of the regions which are handed over to OpenMp to be executed.

D. **Would your solution scale on a 100+ core machine?**
Practically speaking it is hard to tell how well it would scale. The bottle-neck would be agents moving between tasked regions and the number of tasks available. Moving between regions is a sequential operation and would not benefit from additional cores. Similarly if it is not enough tasks available the number of cores would be underutilized.

4 Bonus point part

5 How to run

The demo runs with a GUI and using the sequential implementation if no flag has overwritten the settings. The following flags can be used to alter the demo and change the implementation for the tick function.

--timing-mode - Without gui.

--omp - Executes the OpenMP implementation.

--threads *number* - Tells the demo to use *number* threads when executing.

--silent - Does not print to standard output.

--plot - Stores the timing data to the file testdata.txt

--rebuild - Rebuilds the quadtree each tick.

Work effort

Report was made by Jimmy and Per.