CS 6868: Programming Assignment 3 Report

March 20, 2013

Mahesh Gupta
CS12M018

The summary of execution of programing exercise 3 is as follows.

1. To compare the runtime of both programs:

   The initial version of the code was taking around 360 seconds when summed all 4 stages. This version of the code is taking around 60 seconds for entire execution. While merging the code, certain optimization has been done in game of life code, where instead of creating too many threads; each thread is assigned certain substantial work to perform. Hence the speedup of about 4 has been obtained.

2. To compare the peak memory usage :

   Memory wise no better improvement has been done. Peak memory usage is around 8 Mb for matrix of size 1000*1000 and for 1000 * 10000 size matrix it is around 82 Mb of memory. Arrays are stored in 2D matrix form and each entry is integer, later when threshold step reaches, Boolean matrix is used then and on.

3. Parallelization Strategy:

   The four steps of the program can be executed one after other only, except for the threshold step for which calculating statistics step can be done while generating the matrix only, it does not give huge improvement but saves effort later.

   For shuffling step, row shuffle and column shuffles can be performed independently and row shuffle can be done for multiple entries provided they are not conflicting with previous shuffle.

   For game of life, as 2D matrix is used, each entry is checked. Instead of creating large no of threads, each for one cell, we divide the matrix into various parts and each thread works with one part. This helps by removing overhead of thread creation and removal where each thread is doing very little task.

4. Execution Results :

Input Specification	Execution time	Memory usage	Application Parallelism
Size: 1000 X 1000 shuffles: 1000 p : 1000 Generation: 1000	59.92 seconds	~8 Mb	82.87
Size: 1000 X 10000 shuffles: 1000 p : 1000 Generation: 1000	794.35 seconds	~82 Mb	140.52
Size: 10000 X 10000 Shuffles: 1000 P: 1000 Generation: 10*	61.58 seconds	~600 Mb	180.35

- Due to the fact that execution was taking too much only 10 generations have been generated to see the output