Programming Concurrent Systems: Assignment #1

Due on Monday, November 3, 2014

Alyssa - Ilias

November 3, 2014

Contents

Introduction

For this assignment we are asked to implement a simulation of heat dissipation on the surface of a cylinder. The implementation is sequential and will serve as a base to the following assignments where we will provide a parallelized solution.

The surface is represented as a N x M matrix where each point is the temprature value. There is another same sized matrix where each point is the respective conductivity, where values are in the interval [0,1]. Depending on the conductivity of each point and its 8 surrounding neighbours, a new temprature is recomputed in an iterative manner. Specifically, the following formula describes the computations used for the simulation.

$t' = c_t * t + (1 - c_t) * \frac{\sqrt{2}}{\sqrt{2}+1} * \frac{1}{4} * \sum_{i=1}^{4} a_i + (1 - c_t) * \frac{1}{\sqrt{2}+1} * \frac{1}{4} * \sum_{j=1}^{4} b_j$, where

t' is the new temprature value
t is the point's temprature value
c_t is the point's respective conductivity
a_i are the direct neighbour temprature values
b_j are the diagonal neighbour temprature values

The M columns of the matrix represent the cyclic dimension of the cylinder so that tha first and last column are neighbours whereas the other dimension has fixed boundary on the first and last row. The repsective neighbours of the boundary have a fixed temprature of the initial temprature value.

Solution Description

In our solution we use two matrices. **t_prev** and **t_next**. The matrix **t_prev** first contains the initial temprature values. Then in an iterative process we compute for every point the new value as depicted by Listing 1 and store in **t_next**. When all points have been recomputed, we perform a swap on the two matrices. The iterative process ends when the maximum temprature difference is less than a user provided threshold, or the number of iterations have reached a user provided number.

Listing 1: Snippet of the main calculation

```
double result = row[x] * ourcnd;
double directsum = rowup[x] + rowdown[x] + row[prev] + row[next];
result += directcnd * directsum/4.0;
double diagsum = rowup[prev] + rowup[next] + rowdown[prev] + rowdown[next];
result += diagcnd * diagsum/4.0;
```

During the iterations we gather information on the heat dissipation process. This information is computed either every iteration or periodically depending on user input. We call this a reduction step. In the reduction step between other information we calculate the maximum temprature difference. It is important that we do a reduction step in order to effectivly stop iterating in the case the difference is less than the threshold, but the reduction step also causes unnecessary overhead.

To deal with the boundary issue we do an explicit check whether we are on the first or last row and fetch the initial values directly from the input.
Similarly for the wraparound of the cyclic dimension we do an explicit check whether we are on the first or last column and fetch the opposite's value.

Evaluation

For the evaluation of our code we run a set of experiments. Following are the outcome observations:

From the following graph we can see that if we do the reduction step on every iteration the impact is performance drawback. But we see that if we report periodically even often the overhead is little and the convergence is computed as well.