Technische Universität Berlin

School IV – Electrical Engineering and Computer Science
Chair Communication and Operating Systems (KBS)
http://www.kbs.tu-berlin.de

Integrated Course

Parallel Programming

in Summer Term 2013

J. Schönherr

Assignment 2

Issued: Wednesday, 24th April 2013

Due: Wednesday, 15th May 2013 (parts are due earlier, see below)

Information

Please upload your solution via the ISIS page of this course until 23:55 of the due date. Your upload should consist of a PDF document covering the theoretical parts and a zip/tar.gz archive with your source files. Please list the names and matriculation numbers of all group members inside your uploaded documents.

Exercise 1 – Sequential Overspecification

Consider the following sequential algorithm that calculates `max_sum`:

```
int a[N] = { -1, 5, -2, 0, 1, 4, 6, -4, 3, -6, ... };
int max_sum = 0;
int cur_sum = 0;

for (i = 0; i < N; i++) {
      cur_sum += a[i];
      if (cur_sum < 0) cur_sum = 0;
      if (cur_sum > max_sum) max_sum = cur_sum;
}
```

There is no straightforward way to parallelize the loop as each iteration depends on the previous iteration. Yet, the underlying problem can be parallelized.

(a) Analysis

Analyze the algorithm in order to extract the original problem. What is the purpose of the given algorithm?

(b) Parallel Algorithm

Devise a parallel algorithm based on your problem description!

Write down your algorithm in (commented/explained) pseudo code using collective operations when necessary!

This exercise will be discussed/solved cooperatively on our Friday slots!

Please prepare a) until 3.5., and b) until 10.5.

Exercise 2 – Conway's Game of Life

The goal of this exercise is an implementation of a distributed memory version of Conway's Game of Life. See for instance http://en.wikipedia.org/wiki/Conway%27s_game_of_life for an introduction.

Your implementation should not focus on absolute speed, but instead on correct interaction patterns and a bottleneck-free design. There are two variants presented here: A (simple) and B (advanced). (If you are interested in speed, have a look at *Hash Life*.)

Variant A: Regular Parallelism

A matrix consisting of 1s and 0s is used as a representation of the playing field.

Variant B: Irregular Parallelism

Storing a complete matrix is not necessary as (normally) most cells are dead and do not change. Therefore, it is possible to store only coordinates of living (or interesting) cells, as only living cells or dead cells with living neighbors may change. (Another side-effect is that one can get rid of borders as coordinates may have arbitrary values.)

(a) Problem decomposition

For *both* variants, consider different strategies to distribute work and data across multiple processes – basically the last two steps in Foster's design methodology.

Use the discussion forum in ISIS to present *one* way to decompose the problem (for A *or* B). Present a decomposition that was not yet given. In your suggestion, include the data and work distribution and the necessary data exchanges between nodes. ***Please do so until 3.5!***

For your submission, compare different suggestions (for A *and* for B) and highlight their advantages and disadvantages – also considering different target systems..

(b) Implementation

Select one variant, A *or* B, and implement a strategy with MPI, which is suitable for our cluster!

(It does not have to be the strategy, that you posted in the ISIS forum.)

Hint: For debugging (and testing) purposes, include the ability to output the playing field in a human readable manner after each iteration (e. g., lots of Xs separated by spaces). (The code required for this, should not be considered as integral part of the program, i. e., do not design your program around your output code.)