Uppsala University
Department of Information Technology
Scientific Computing

Programming of Parallel Computers, 10hp, 2010-06-01

Time: $8^{00} - 13^{00}$
Help: None
Note: There are two exams, one for the *old* course with 6hp and one for the *new* course with 10hp. This is the 10hp exam. If you are taking the 6hp course you need to ask for that exam.

Each of the six problems below can give up to five points. For maximum points, you must give detailed answers and motivate your assumptions.

1. In OpenMP we have three different directives for work sharing; *for, sections, task*. Explain how these directives work and point out what their differences are. Give also an example of a situation for each of the directives in which they are suitable.

2. In the Pthreads API we have something called *mutexes* and *condition variables*. Explain what these are, how they work and what they can be used for.

3. In MPI we have the concepts of *communicators* and *virtual topologies*. Explain what these are, how they work and what they can be used for.

4. Recently graphical processing units (GPU) have gained popularity to be used for general purpose computing.

   a) Give at least two important characteristics that an application must have to be really suitable for a GPU.

   b) Give at least two examples of problems (algorithms or applications) that are not ideally suited to GPU architecture, and explain why they are not ideal.

5. The Quick-sort algorithm can be described as:

```
function quicksort(array, left, right)
   if (right>left) then
      pivotIndex=left+(right-left)/2
      pivotNewIndex=partition(array, left, right, pivotIndex)
      quicksort(array, left, pivotNewIndex-1)
      quicksort(array, pivotNewIndex+1, right)
   end if
end function
```

Figur 1: Quick-sort algorithm

Where the function `partition` reorders the array into two parts, with elements smaller respectively larger than the pivot and returns the split point `pivotNewIndex`. Assume that you have the functions `quicksort` and `partition` available. Then, sketch a parallel implementation of the Quick-sort algorithm using Pthreads.

6. This task is to simulate Conway's Game of Life. The universe of the Game of Life is a two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbors, which are the cells that are directly horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

   - Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
   - Any live cell with more than three live neighbours dies, as if by overcrowding.
   - Any live cell with two or three live neighbours lives on to the next generation.
   - Any dead cell with exactly three live neighbours becomes a live cell.

   A new generation is created by applying the above rules simultaneously to every cell. Sketch a parallel implementation of the Game of Life using MPI.

Good Luck!