Uppsala University
Department of Information Technology
Scientific Computing

Programming of Parallel Computers, 10hp, 2011-03-18

Time: $8^{00} - 13^{00}$
Help: None

Each of the six problems below can give up to five points. For maximum points, you must give detailed answers and motivate your assumptions. Grade 3: 12p, Grade 4: 18p, Grade 5: 24p

1. When parallelizing an application we can use different techniques, *(a) Manager-Worker, (b) Peer,* and *(c) Pipeline*. Describe these techniques and give an example of an application or describe a situation where it is suitable to use the respective techniques.

2. Describe the key differences between MPI, Pthreads and OpenMP.

3. Describe the key differences between CUDA and OpenCL.

4. In the course we have been looking at the *Enumeration sort* algorithm as a case study and parallelized it with both OpenMP and Pthreads.

Enumeration sort

```
for j=1 to n
   rank(j)=0
   for i=1 to n
      if (indata(i)<indata(j)) rank(j)=rank(j)+1
   end for
end for
```

Figur 1: The Enumeration sort algorithm.

Sketch a distributed memory parallelization of the algorithm using MPI (the syntax does not have to be exact and the program not complete but the relevant code segments, details and arguments must be included). An additional requirement is that both the indata and the rank arrays are partitioned and distributed over the processors. Moreover, neither of the arrays should be gathered into any of the processors as a global array.

5. 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. The game stops when the number of live cells exceeds a given maximal number of live cells or when all cells are dead. Sketch a parallel implementation of the Game of Life using Pthreads.

6. 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 2: 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`.

   One idea to parallelize the Quick-sort algorithm is to let one processor do recursion in a few levels. When it hits the maximal number of recursion levels it creates two tasks where each task is to sort the remaining fraction of the list serially with Quick-sort. It puts the tasks to a work queue from where the other processors picks up work. Sketch a parallel implementation in OpenMP of Quick-sort using the idea above. Assume that you have the function `partition` available.

Good Luck!