CS2101 Due: Monday 3-rd of November 2014

Problem Set 2

CS2101 *Submission instructions on last page*

PROBBLEM 1. [20 points]

In this problem, we consider different ways of computing the dot product of two vectors.

Question 1. [5 points] Write first a Julia function `dotproduct_serial(U,V)` computing the dot product of the vectors of `U` and `V` in a serial fashion, that is, without using any parallel constructs.

Question 2. [5 points] Next, write a Julia function `dotproduct_parallel_reduction(U,V)` computing the dot product of the vectors of `U` and `V` using parallel reduction and thus Julia's construct `@parallel`.

Question 3. [5 points] Next, write a function `dotproduct_parallel_dnc(U,V)` computing the dot product of the vectors of `U` and `V` using a divide-and-conquer approach (with a base case to be optimized experimentally). For this function, we shall assume that `U` and `V` are shared arrays.

Question 4. [5 points] Run the following tests and record the timings

```
N = 100000
U = [rand() for i=1:N]
V = [rand() for i=1:N]
@time dotproduct_parallel_reduction(U,V)
@time dotproduct_serial(U,V)
Us = convert(SharedArray, U)
Vs = convert(SharedArray, V)
@time dotproduct_dnc(Us, Vs)

N = 1000000
U = [rand() for i=1:N]
V = [rand() for i=1:N]
@time dotproduct_parallel_reduction(U,V)
@time dotproduct_serial(U,V)
Us = convert(SharedArray, U)
Vs = convert(SharedArray, V)
@time dotproduct_dnc(Us, Vs)

N = 10000000
```

```
U = [rand() for i=1:N]
V = [rand() for i=1:N]
@time dotproduct_parallel_reduction(U,V)
@time dotproduct_serial(U,V)
Us = convert(SharedArray, U)
Vs = convert(SharedArray, V)
@time dotproduct_dnc(Us, Vs)
```

Explain these experimental results.

PROBBLEM 2. [40 points] The goal of this exercise is to obtain a parallel `Julia` implementation of the merge-sort algorithm. You can review this algorithm at `http://en.wikipedia.org/wiki/Merge_sort`

You will find there several ways of presenting this algorithm. The one of the *Top-down implementation* section is overwrites the input array. In other words, this sorting process can be seen as in place [1] To learn more about this idea of *working-in-place*, read the page `http://en.wikipedia.org/wiki/In-place_algorithm`

To learn more about performing merge-sort in-place, read the page `http://stackoverflow.com/questions/2571049/how-to-sort-in-place-using-the-merge-sort-algorithm` or the page `http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/mergeSort.htm`

Merge-sort can also be done *out-of-place*, that is, without modifying the input array and by returning a new and sorted array. To see an example, look at the first pseudo-code area at the top of the page `http://rosettacode.org/wiki/Sorting_algorithms/Merge_sort`

If the in-place solution seems more efficient (as it consumes less resources) the out-of-place is less tricky to implement. For that reason, we want to consider both in this exercise.

Note that, in class, we presented Julia code for both the in-place and out-of-place (serial) Merge-sort. We also show a parallel version of the out-of-place Merge-sort, with relatively poor performance.

Question 1. [20 points] Adapt the serial in-place Merge-sort seen in class such that it runs in a parallel fashion. Using a shared array would be necessary. Observe that using a threshold (as in Fibonacci) for switching between the serial and parallel modes would be needed

Question 2. [10 points] Collect experimental data (similarly to what we did with Fibonacci) in order to optimize the threshold for switching between the serial and parallel modes. Using plots (with the Winston package) is required.

[1] In fact, this is not completely true since an intermediate array, or *work array*, `B` is used.

Question 3. [10 points] Compare experimentally this optimized version of parallel in-place Merge-sort with the parallel out-of-place Merge-sort seen in class. Using plots (with the Winston package) is required.

PROBBLEM 3. [40 points]

In this problem, we will realize a parallel implementation of a simulation based on a 4-point stencil, called *Jacobi iteration*. Note that this stencil is 2-D (whereas the example seen in class is 1-D) but involves only two time steps at each iteration (whereas the example seen in class uses three time steps, namely past, present and future).

In the chapter `http://www.netlib.org/utk/papers/mpi-book/node44.html` the example called *Jacobi iteration - sequential code* explains how to obtain one time from the previous one.

Question 1. [10 points] Write first a Julia function `Jacobi_serial(A, N, T)` working on a 2-D grid of order `N` and performing `T` time steps of the Jacobi iteration in a serial fashion.

Question 2. [20 points] Write first a Julia function `Jacobi_parallel(A, N, T)` working on a 2-D grid of order `N` and performing `T` time steps of the Jacobi iteration in a parallel fashion using the 2-D (block,block) partition suggested in `http://www.netlib.org/utk/papers/mpi-book/node44.html`. Clearly will use an inner function working in a divide-and-conquer manner with a threshold and using a shared array for `A`.

Question 3. [10 points] Compare experimentally the functions `Jacobi_serial(A, N, T)` and `Jacobi_parallel(A, N, T)`.

Submission instructions.

Format: Problems 1, 2 and 3 involve programming with Julia: the corresponding programs must be submitted as three input **text** files to be called `Pb1.jl`, `Pb2.jl`, `Pb3.jl` respectively. Each of these three files must be a valid input file for Julia. In addition, each user defined function must be documented, using comments. Moreover, the answers to Questions 1 and 4 of Problem 1, and Question 4 of Problem 3 should be typed and submitted as a PDF file called `OtherAnswers.pdf`. No format other than PDF will be accepted. To summarize: each assignment submission consists of four files: `Pb1.jl`, `Pb2.jl`, `Pb3.jl` and `OtherAnswers.pdf`.

Submission: The assignment should be returned to the instructor **and** the TA by email.

Collaboration. You are expected to do this assignment *on your own* without assistance from anyone else in the class. However, you can use literature and if you do so, briefly list your references in the assignment. Be careful! You might find on the web solutions

to our problems that are not appropriate. So please, avoid those traps and work out the solutions by yourself. You should not hesitate to contact the instructor or the TA if you have any question regarding this assignment. We will be more than happy to help.

Marking. This assignment will be marked out of 100. A 10 % bonus will be given if your answers are clearly organized, precise and concise. Messy assignments (unclear statements, lack of correctness in the reasoning, many typographical or language mistakes) may give rise to a 10 % malus.