CS3101 Due: Wednesday 12-th of February 2014

Problem Set 1

CS3101 *Submission instructions on last page*

PROBBLEM 1. [30 points] The goal of this problem is to experiment, within `Julia`, with the Sieve of Eratosthenes for determining all prime numbers less than a given positive integer. At the Wikipedia page `en.wikipedia.org/wiki/Sieve_of_Eratosthenes` you will find a description of this algorithm.

Question 1. [10 points] Write a serial `Julia` function `Eratosthenes_serial` which takes a positive integer `n` as input and returns all prime numbers less than `n`.

Question 2. [20 points] Taking advantage of distributed arrays in `Julia`, write a function `Eratosthenes_distributed` which takes two positive integers `n` and `p` as input and returns all prime numbers less than `n` by using `p` processors.

PROBBLEM 2. [70 points] The goal of this problem is to experiment, within `Julia`, with a matrix multiplication scheme which was designed for distributed computing by Lynn Elliot Cannon, see the Wikipedia page `http://en.wikipedia.org/wiki/Cannon'salgorithm`.

Let us assume that we have p processors available and that p is a perfect square, say $p \in \{4, 9, 16, \ldots\}$. Our input matrices A and B are assumed to be square of order n. We denote by C their product.

We divide A, B, C into p square blocks of order $m := n/\sqrt{p}$, thus assuming that $\sqrt{p}$ divides n. We denote by $A_{i,j}$, or $B_{i,j}$, or $C_{i,j}$ the square block of A, or B, or C at the intersection of the i-row and the j-th column, for $0 \leq i < \sqrt{p}$, $0 \leq j < \sqrt{p}$.

We view our p processors as the points of two-dimensional square grid of order $\sqrt{p}$. Hence, we denote by $P_{i,j}$ the processor at the intersection of the i-row and the j-th column, for $0 \leq i < \sqrt{p}$, $0 \leq j < \sqrt{p}$.

The goal of Processor $P_{i,j}$ is to compute $C_{i,j}$. Recall that we have:

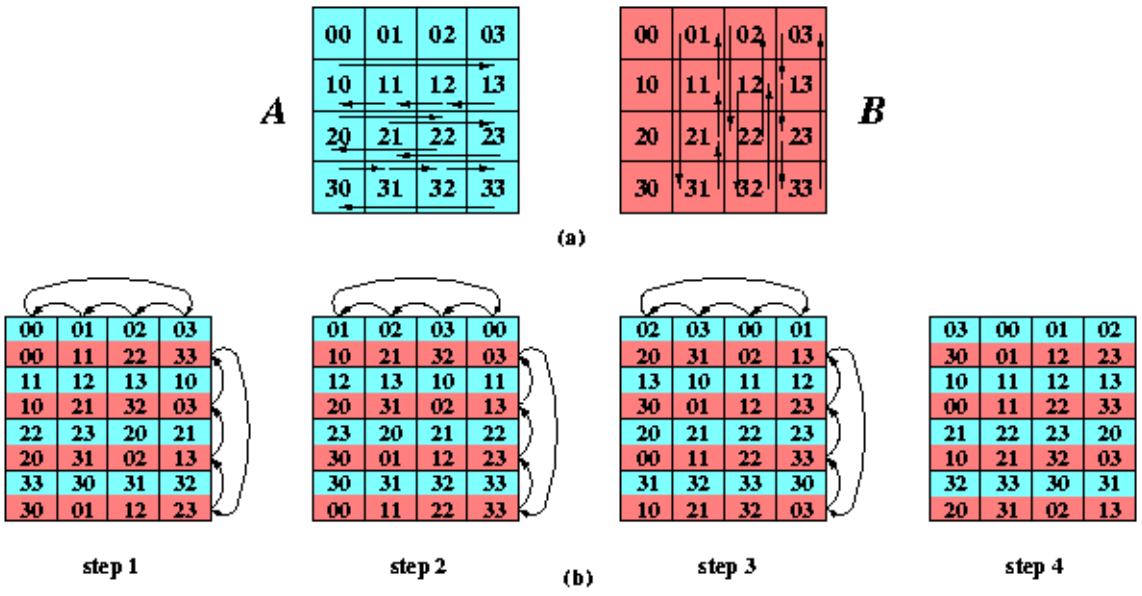
$$C_{i,j} \;=\; \sum_{k=0}^{\sqrt{p}} A_{i,k}\, B_{k,j}. \tag{1}$$

Initially, Processor $P_{i,j}$ holds $A_{i,j}, B_{i,j}, C_{i,j}$.

The key idea of the algorithm is to shift blocks of A horizontally and blocks of B vertically so that matching blocks $A_{i,k}$ and $B_{k,j}$ meet in $P_{i,j}$ at the same time, for all $0 \leq i < \sqrt{p}$, $0 \leq j < \sqrt{p}$.

Letting $r := \sqrt{p}$, the algorithm precisely writes as follows:

for $i = 0, \ldots, r-1$ **do_in_parallel**
    rotate blocks of A in row i by i positions leftward
for $j = 0, \ldots, r-1$ **do_in_parallel**
    rotate blocks of B in column j by j positions upward
repeat r times
  for each processor $P_{i,j}$ **do_in_parallel**
    multiply current blocks of A and B and add the result to $C_{i,j}$
  for $i = 0, \ldots, r-1$ **do_in_parallel**
    rotate blocks of A in row i by 1 position leftward;
  for $j = 0, \ldots, r-1$ **do_in_parallel**
    rotate blocks of B in column j by 1 position upward;



Question 1. [10 points] Write a serial `Julia` function `Cannon_serial_2` which

- takes as input two square matrices `A` and `B` (encoded as non-distributed arrays) of the same order `n` and
- returns their product `C` in two steps as follows

$$\begin{array}{lcllcl} C_{0,0} & := & A_{0,0}B_{0,0} & C_{0,1} & := & A_{0,1}B_{1,1} \\ C_{1,0} & := & A_{1,1}B_{1,0} & C_{1,1} & := & A_{1,0}B_{0,1} \end{array} \tag{2}$$

$$\begin{array}{lcllcl} C_{0,0} & := & C_{0,0} + A_{0,1}B_{1,0} & C_{0,1} & := & C_{0,1} + A_{0,0}B_{0,1} \\ C_{1,0} & := & C_{1,0} + A_{1,0}B_{0,0} & C_{1,1} & := & C_{1,1} + A_{1,1}B_{1,1} \end{array} \tag{3}$$

where the $A_{i,j}$, $B_{i,j}$ are square blocks.

Question 2. [20 points] Write a distributed `Julia` function `Cannon_distributed_2` which

- takes as input two square matrices `A` and `B` (encoded as distributed arrays) of the same order `n` and
- returns their product `C` using the scheme of Question 1.

This function will be using 4 processors (regarded as a 2×2-grid) with the notations and hypotheses described above for Cannon's algorithm.

Question 3. [30 points] Write a distributed `Julia` function `Cannon_distributed_p` which

- takes as input two square matrices `A` and `B` (encoded as distributed arrays) of the same order `n`, a positive integer `p`, and
- returns their product `C` using Cannon's algorithm on `p` processors.

Question 4. [10 points] Compare experimentally the running times of the above three implementations of matrix transposition. This implies to choose a series of matrix sizes and apply these three implementations to each selected size.

Submission instructions.

Format: Problems 1 and 2 involve programming with Julia: they must be submitted as two input files to be called `Pb1.jl` and `Pb2.jl`, respectively. Each of these two files must be a valid input file for Julia. In addition, each user defined function must be **documented.** Experimental data for Question 4 of Problem 2 can be appended as comments to the file `Pb2.jl`. To summarize, each assignment submission consists of tow files: `Pb1.jl` and `Pb2.jl`.

Submission: The assignment should be returned to the instructor 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. For instance, because the cache memory model is different. So please, avoid those traps and work out the solutions by yourself. You should not hesitate to contact the instructor if you have any question regarding this assignment. I 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.