CS3101 Due: Sunday 23-rd of March 2014

Problem Set 2

CS3101 *Submission instructions on last page*

Preliminaries. The prefix sum is a fundamental operation in computer science with many basic applications, see `http://en.wikipedia.org/wiki/Prefix_sum`. Given a vector $\vec{x} = (x_1, x_2, \ldots, x_n)$, the *prefix sum* of $\vec{x}$ is the vector $\vec{y} = (y_1, y_2, \ldots, y_n)$ such that $y_i = \sum_{i=1}^{j=i} x_j$ for $1 \leq j \leq n$. For instance, the prefix sum of $\vec{x} = (1, 2, 3, 4, 5, 6, 7, 8)$ is $\vec{y} = (1, 3, 6, 10, 15, 21, 28, 36)$. So a `Julia` implementation of the above specification would be:

```
function prefixSum(x)
   n = length(x)
   y = fill(x[1],n)
   for i=2:n
      y[i] = y[i-1] + x[i]
   end
   y
end

n = 10

x = [mod(rand(Int32),10) for i=1:n]

prefixSum(x)
```

The i-th iteration of the loop is not at all decoupled from the $(i-1)$-th iteration. Hence, it looks like it is impossible to parallelize, right?

In fact, there is a parallel version (even several ones) of the above algorithm. In this assignment, we propose to implement one based on a divide and conquer strategy. We shall do it in both `Julia` and `CilkPlus` in order to have a good understanding of what parallelizes well in each of these concurrency platforms.

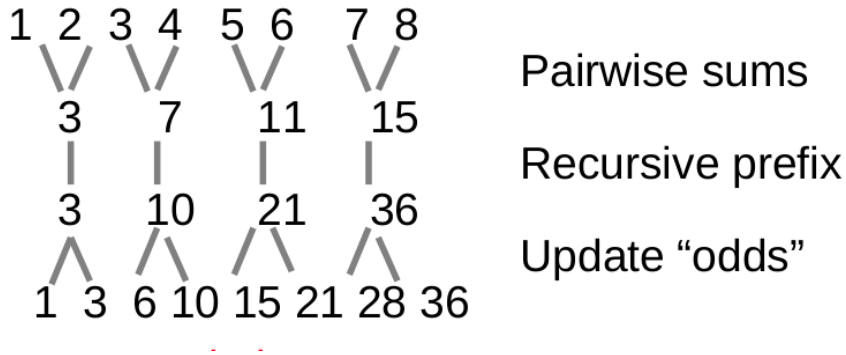
This divide and conquer strategy is a beautiful idea: it is a powerful recipe to turning a serial algorithm into a parallel one. Consider again the input vector $x[1], x[2], \ldots, x[n]$ where n is a power of 2.

(S1) For all even k's between 2 and n, replace $(x[k-1], x[k])$ with $(x[k], x[k] + x[k-1])$

(S2) If $n = 2$ return x. Otherwise, make a recursive call on $x[2], x[4], \ldots, x[n]$

(S3) For all even k's between 2 and n, compute $x[k-1] = x[k] - x[k-1]$.

The above picture illustrates this algorithm with $n = 8$.



PROBBLEM 1. [50 points]

Write a `Julia` function `DistributedPrefixSum` which takes as an input a distributed array `x` and the array `Ws` of workers IDs owning `x`. This function will may apply the above algorithm in one of the following ways

Method 1

1. Each worker performs Step (S1) in its local part
2. Unless $n = 2$ Step (S2) requires a recursive call. Note that:
   - this recursive function will need to pass the stride.
   - Also, for n small enough, it will be more efficient to use the serial prefix sum (listed at the beginning of this statement)
   - Note that, every worker except the leftmost one, will need a value form its left neighbor.
3. Each worker performs Step (S3) in its local part. Note that here again every worker except the leftmost one, will need a value form its left neighbor.

Method 2

(T1) Each worker performs the entire serial algorithm on its local part. So in the case of two workers and $\vec{x} = (1, 2, 3, 4, 5, 6, 7, 8)$ owning respectively the first and second halves of $\vec{x}$, this returns $\vec{x} = (1, 3, 6, 10, 11, 18, 26)$, whereas we expect $(1, 3, 6, 10, 21, 28, 36)$,

(T2) Each worker (except the leftmost one) passes to each of his neighbors on the right, its rightmost value; on our example, this means that the first worker passes 10 to the second worker.

(T3) Each worker adds the sum of the values it receives to each entry of its local part; on our example, this means that the second worker adds 10 to each entry of $(11, 18, 26)$, hence now we have $\vec{x} = (1, 3, 6, 10, 21, 28, 36)$, as desired.

You should measure running times of both the serial function `prefixSum` and the distributed function `DistributedPrefixSum` on arrays of sizes 10^8, 5.10^8, 10^9 with random numbers as entries.

PROBBLEM 2. [50 points]

Write a `CilkPlus` function `ParallelPrefixSum` which takes as an input an array `x` and applies one of the above divide and conquer strategy (or any other efficient algorithm) for computing the prefix-sum of `x`:

- if **Method 1** is used, a `cilk_for` loop will be used for Steps (S1) and (S3),
- if **Method 2** is used, a `cilk_for` loop will be used for Step (T3).

Of course, there is no notion of distributed arrays in `CilkPlus` and ordinary arrays should be used. As for workers, they are the cores that are used to run your `CilkPlus` program.

The chapter called *parallel scanning* of the CS2101 `http://www.csd.uwo.ca/~moreno/cs2101a_moreno/index.html` gives other possible algorithms for computing prefix-sum.

You should measure running times of both the serial function `prefixSum` and the parallel function `ParallelPrefixSum` on arrays of sizes 10^8, 5.10^8, 10^9 with random numbers as entries.

Submission instructions.

Format: Problems 1 and 2 involve programming with `Julia` and `CilkPlus`: they must be submitted as two input files to be called `Pb1.jl` and `Pb2.cpp`, respectively. Each of these two files must be a valid input file for `Julia` and `CilkPlus` respectively. In addition, each user defined function must be **documented.** Experimental data can be appended as comments in each file `Pb2.jl`.

To summarize, each assignment submission consists of tow files: `Pb1.jl` and `Pb2.cpp`.

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.