Distributed and Parallel Systems — Due on Friday, February 17, 2017

Assignment 1

CS4402B / CS9535B — *University of Western Ontario*

Submission instructions.

Format: The answers to the problem questions should be typed.

- If these are programs, input test files and a `Makefile` (for compiling and running) are required.
- If these are algorithms or complexity analyzes, LaTeX is highly recommended; in any case a PDF file should gather all these answers.

All the files should be archived using the UNIX command `tar`.

Submission: The assignment should submitted through the OWL website of the class.

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 which are not appropriate. For instance, because the parallelism model is different. So please, avoid those traps and work out the solutions by yourself. You should not hesitate to contact me if you have any questions 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 paper is clearly organized, the answers are precise and concise, the typography and the language are in good order. Messy assignments (unclear statements, lack of correctness in the reasoning, many typographical and language mistakes) may give rise to a 10 % malus.

PROBBLEM 1. [20 points] The objective of this problem is to prove that, with respect to the Theorem of Graham & Brent, a greedy scheduler achieves the stronger bound:

$$T_P \leq (T_1 - T_\infty)/p + T_\infty.$$

Let $G = (V, E)$ be the DAG representing the instruction stream for a multithreaded program in the fork-join parallelism model. The sets V and E denote the vertices and edges of G respectively. Let T_1 and T_∞ be the work and span of the corresponding multithreaded program. We assume that G is connected. We also assume that G admits a single source (vertex with no predecessors) denoted by s and a single target (vertex with no successors) denoted by t. Recall that T_1 is the total number of elements of V and T_∞ is the maximum number of nodes on a path from s to t (counting s and t).

Let $S_0 = \{s\}$. For $i \geq 0$, we denote by S_{i+1} the set of the vertices w satisfying the following two properties:

(i) all immediate predecessors of w belong to $S_i \cup S_{i-1} \cup \cdots \cup S_o$,

(ii) at least one immediate predecessor of w belongs to S_i.

Therefore, the set S_i represents all the units of work which can be done during the i−-th parallel step (and not before that point) on infinitely many processors.

Let $p > 1$ be an integer. For all $i \geq 0$, we denote by w_i the number of elements in S_i. Let ℓ be the largest integer i such that $w_i \neq 0$. Observe that $S_0, S_1, \ldots, S_\ell$ form a partition of V. Finally, we define the following sequence of integers:

$$c_i = \begin{cases} 0 & \text{if } w_i \leq p \\ \lceil w_i/p \rceil - 1 & \text{if } w_i > p \end{cases}$$

Question 1. For the computation of the 5-th Fibonacci number (as studied in class) what are $S_0, S_1, S_2, \ldots$?

Question 2. Show that $\ell + 1 = T_\infty$ and $w_0 + \cdots + w_\ell = T_1$ both hold.

Question 3. Show that we have:

$$c_0 + \cdots + c_\ell \leq (T_1 - T_\infty)/p.$$

Question 4. Prove the desired inequality:

$$T_P \leq (T_1 - T_\infty)/p + T_\infty.$$

Question 5. Application: Professor Brown takes some measurements of his (deterministic) multithreaded program, which is scheduled using a greedy scheduler, and finds that $T_8 = 80$ seconds and $T_{64} = 20$ seconds. Give lower bound and an upper bound for Professor Brown's computation running time on p processors, for $1 \leq p \leq 100$? Using a plot is recommended.

PROBBLEM 2. [30 points]

In the chapter *Analysis of Multithreaded Algorithms*, we studied the 2-way and 3-way construction of a tableau.

Question 1. Describe, in plain words, how to construct a tableau in a k-way fashion, for an arbitrary integer $k \geq 2$, using the same stencil (the one of the Pascal triangle construction) as in the lectures.

Question 2. Determine the work and the span for an input square array of order n.

Question 3. Realize a `Julia` or `CilkPlus` a multithreaded implementation of that algorithm. Collect running times (both serial and parallel) for increasing values of n (say consecutive powers of 2) and different values of k (at least 2 and 3).

PROBBLEM 3. [50 points] Let G be a directed graph with n vertices. For simplicity we identify the vertex set to the set of positive integers $\{1, 2, \ldots, n\}$. To each couple (i, j), with $1 \leq i, j \leq n$, we associate a weight $w_{i,j}$ such that:

(*i*) $w_{i,j}$ is a non-negative integer if and only if (i, j) is an arc in G,

(*ii*) $w_{i,j}$ is $+\infty$ if and only if (i, j) is not an arc in G.

We assume $w_{i,i} = 0$ for all $1 \leq i \leq n$. If $x_1, x_2, \ldots, x_m$ are $m \geq 2$ vertices of G such that (x_1, x_2), (x_2, x_2), $\ldots$, (x_{m-1}, x_m) are all arcs of G, we say that $p = (x_1, x_2, \ldots, x_m)$ is a path in G from x_1 to x_m; moreover the weight of p is denoted by $w(p)$ and defined by

$$w(p) = w_{x_1,x_2} + w_{x_2,x_3} + \cdots + w_{x_m,x_{m-1}}$$

For each couple (i, j) which is not an arc in G it is natural to ask whether

(1) there is a path in G from i to j, and

(2) if such path exists, then compute the minimal weight of such a path.

This question is often referred as ASAP for *All-Pair Shortest Paths*. The celebrated *Floyd–Warshall algorithm* solves ASAP by computing a matrix `path` as follows:

```
for k = 1 to n
   for i = 1 to n
      for j = 1 to n
         path[i][j] = min ( path[i][j], path[i][k]+path[k][j] );
```

after initializing `path[i][j]` to $w_{i,j}$. For more details, please refer to the Wikipedia page of the Floyd-Warshall algorithm.

Question 1. [5 points] Is it possible to turn the *FloydWarshall algorithm* into a parallel algorithm for the fork-join parallelism? If yes, analyze the work, the span and the parallelism of this algorithm.

Question 2. [5 points] Discuss the data locality of the above sequential FloydWarshall algorithm (not your parallel version of it). Doing a formal cache complexity analysis is not required.

One way to obtain a better algorithm for ASAP (in terms of parallelism and data locality) is to apply a divide and conquer approach. To this end we view $(w_{i,j})$ as an $n \times n$-matrix, denoted by W. We also view the targeted result, namely the values `(path[i][j])` as an $n \times n$-matrix, denoted by $\overline{W}$.

Before stating the divide and conquer formulation, we introduce a few notations. Let X, Y be square matrices (of the same order) whose entries are non-negative integers or $+\infty$. We denote by

- XY the *min-plus product* of X by Y (obtained from the usual matrix multiplication by replacing + (resp. $\times$) by min (resp. +))
- $X \vee Y = \min(X, Y)$ the element-wise minimum of the two matrices X and Y.

We are ready to state the divide and conquer formulation. If we decompose W into four $n/2 \times n/2$-blocks, namely

$$W = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

then we have

$$\overline{W} = \begin{pmatrix} E & EB\overline{D} \\ G & \overline{D} \vee GB\overline{D} \end{pmatrix}$$

where we have $E = \overline{A \vee B\overline{D}C}$ and $G = \overline{D}CE$. We shall admit that these formulas are correct (even though proving them is not that hard).

Question 3. [5 points] Propose an algorithm for computing $\overline{W}$ in the fork-join parallelism model.

Question 4. [5 points] Analyze the work, the span and the parallelism of your algorithm.

There exist alternative algorithms for the ASAP problem which rely on the min-plus multiplication. A simple one is based on the observation that $\overline{W} = W^n$ (and in fact W^{n-1}) where is the n-th power W is computed for min-plus multiplication using repeated squaring.

Question 5. [5 points] Propose such an algorithm. You are welcome to use the literature or simply to use the one suggested above.

Question 6. [5 points] Analyze the work, the span and the parallelism of this third algorithm.

Question 7. [20 points] Realize a `Julia` or `CilkPlus` a multithreaded implementation of that algorithm. Collect running times (both serial and parallel) for increasing values of n (say consecutive powers of 2) and different values of k (at least 2 and 3).