Assignment #1 – Functional Programming for Parallel Processing

Introduction

This assignment aims to consolidate your understanding of several topics taught in this course, such as applied functional programming and parallel /distributed processing. You are required to build and evaluate several parallel versions of a simple but fundamental algorithm, Longest Common Subsequence (LCS). This algorithm is very important, e.g. in bio-informatics, and efficient parallel versions are still sought and proposed. For our purpose, we only require the LCS length, not the longest sequences.

Overview

The assignment has two required parts and an optional part (for a possible bonus).

1. Required: three basic LCS versions, based on a space-aware dynamic programming (DP) model.

   1.1 Sequential implementation (SEQ), used as a baseline to evaluate the parallel speedup. Long input strings are expected, so you need to implement a diagonal wavefront, instead of the textbook array approach.

   1.2 Parallel implementations (PAR), following a coarse granularity model. The conceptual array is divided into rectangular sub-arrays, further allocated to actors (possibly on different machines).

   Each actor works sequentially, using exactly the sequential implementation SEQ. Thus, each actors evolves on its own internal diagonal wavefront. There is also one global diagonal wavefront which activates the actors. There are two versions of this global activation:

       1.2.1 Parallel synchronous (PAR-SYNC): a strict multi-phase algorithm, with a barrier which ensures that all actors on the same diagonal complete before the next set of actors starts.

       1.2.1 Parallel asynchronous (PAR-ASYNC): a relaxed algorithm, where any actor can start as soon as he receives the needed boundary data (no need for barriers or any other form of synchronisation).

The figures (1, 2, and 3) schematically illustrate these three cases. Green areas indicate active cells, currently on a diagonal wavefront. Yellow areas indicate already processed cells. White areas indicate unprocessed cells. Red lines indicate boundaries between actors.

2. Required: A report with literature evaluation and empirical evaluations of your results (more details below)

3. Optional (OPT): suggestions:

   3.1 A Multiple LCS (MLCS) version, for the LCS of 3 strings (or more); or

   3.2 An advanced LCS version, at your choice, with a faster parallel implementation; or

   3.3 A container-based implementation of your required parallel versions, where each actor is allocated its own container.
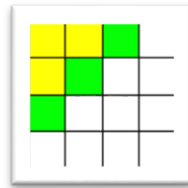
Figure 1: SEQ
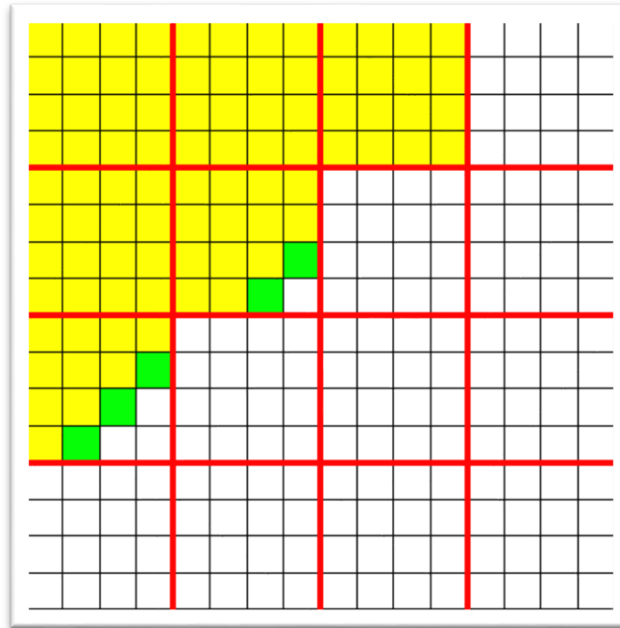
Figure 2: PAR-SYNC
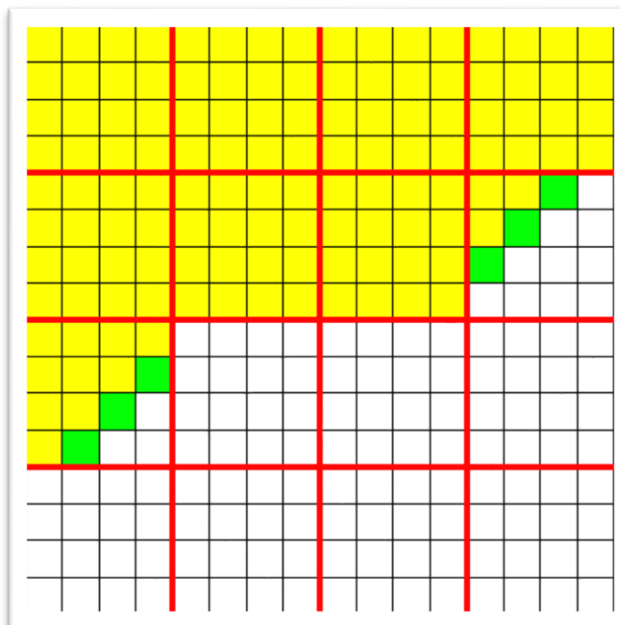
Figure 3: PAR-ASYNC

Development Tools

For all implementations, use F# and Akka.NET actors. You can develop your program with Visual Studio, if you wish so. However, your submission must only include F# .FS source files plus a batch file to compile your program, which should be called A1.EXE (for consistency with the marking tools).

For the optional containers part, use Docker containers and images based on F# and Akka.NET.

Running the Program

Please strictly follow these conventions (for consistency with the marking tools)!

1. Invoking the program – one of the following lines:

```
A1.EXE  /S1:path-to-file1  /S2:path-to-file2  /SEQ

A1.EXE  /S1:path-to-file1  /S2:path-to-file2  /PAR-SYNC

A1.EXE  /S1:path-to-file1  /S2:path-to-file2  /PAR-ASYNC
```

Where file1 and file2 are .TXT files using the UTF8 encoding and containing the two input strings.

2. Standard output: one single integer, the LCS length – only this number!

3. Error output: may contain any additional traces which you want. However, these may affect the runtime, so you should disable these for the final submission.

Report

Your report should be structured and written like a research article. It should contain: title, author, abstract, introduction, good and updated literature overview, brief descriptions of your algorithms, clear empirical evaluations of the runtimes of your implementations (with tables and plots), conclusions, and bibliography.

As indicated by the highlights, the focus should be on the literature evaluation and your own empirical evaluation. Note that you can start working on your report from day one: only the empirical evaluation and conclusions need to wait for your results.

Suggestion to use a good and "standard" format for scientific publications, such as the LNCS article style, either in LaTeX (preferable) or Word.

Deliverables and Submission

Submit electronically, to the new COMPSCI web dropbox, an upi.7z archive containing an upi folder with:

- o   your report as PDF;

- o   your F# source file(s);

- o   a _COMPILE.BAT batch file to compile your file(s) into an A1.EXE executable.

Please keep your electronic dropbox receipt and follow the instructions given in our Assignments web page (please read these carefully, including our policy on plagiarism):

http://www.cs.auckland.ac.nz/courses/compsci734s1c/assignments/

Deadline

Monday 25 April 2016, 18:00! Please do not leave it for the last minute. Remember that you can resubmit and, by default, we only consider your last submission.

After this deadline, submissions will be still accepted for 5 more days, with a gradually increasing penalty of 0.5% for each hour late. For example:

Monday 25 April 2016, 20:00:   -1%

Tuesday 26 April 2016, 18:00:   -12%

Wednesday 27 April 2016, 18:00:   -24%

Thursday 28 April 2016, 18:00:   -36%

Friday 29 April 2016, 18:00:   -48%

Saturday 30 April 2016, 18:00:   -60%

After this, no more submissions are accepted!

Appendices

A#1-LCS-Demo.xls: A simple intuitive Excel demo (recall that Excel is a functional language)

Others sections will follow, if more detailed clarifications are necessary…