CSC4005 High Performance Computing (2018/19)

HPC Programming Project

Contact: Dr Blesson Varghese
b.varghese@qub.ac.uk

Deadline:
For submitting programs - Thursday, November 29, 23:59
For submitting presentations - Friday, December 07, 23:59
For re-submissions - Friday, December 07, 23:59

Overview

The final programming project builds on your previous assignments. Given a collection of p patterns and t texts the aim is to detect and report the positions of certain patterns in certain texts. In some cases you will be required to detect whether the pattern occurs in the text, for others you will be required to detect every occurrence of the pattern in the text. There will be a separate control file to specify which patterns are to be sought in which texts.

You will design and implement two solutions, one using OpenMP (`project_OMP.c`) and one using MPI (`project_MPI.c`). The challenge is to design programs that will exploit the parallel programming and high-performance computing techniques you have studied to solve the problem as quickly as possible. You have complete flexibility in the strategy you adopt. For example, you may opt for an embarrassingly parallel solution or a more sophisticated fine-grain solution. The faster your solution *correctly* solves the problem the more marks you will accumulate. (Note: you are not required to count comparisons.) However, your program must be based on the straightforward pattern matching algorithm described in Assignment 1. The pattern must be searched from left to right and each character tested separately (in particular you must not use the `memcmp` function). You may assume that you have 4 cores to use for your OMP and MPI programs, but each core may execute more than one thread.

Inputs

You will be provided with an example folder called `small-inputs` in which you will find

- p patterns, `pattern0.txt`, `pattern1.txt`, ...
- t texts, `text0.txt`, `text1.txt`, ... and
- a control file, `control.txt`.

Each line in the control file will contain three numbers and will correspond to a single search.

- The first number will be 0, meaning find whether the pattern occurs, or 1, meaning find every occurrence.
- The second number will indicate which text to use.
- The third number will indicate which pattern to use.

You will also be provided with a file `small-inputs.sorted` which contains the expected output for this input.

You may assume that no text or pattern file will exceed 20MB. However, a pattern file may exceed the size of a text file. You may also assume that $1 \leq p \leq 20$ and $1 \leq t \leq 20$.

Outputs

Your program must output its results to a single file, `result_OMP.txt` or `result_MPI.txt`, where each line will contain three integers indicating, respectively, the text id, the pattern id and a result from the search. If searching for any occurrence the result should be -2 to indicate that the pattern was found and -1 to indicate that it was not found. If searching for every occurrence the pattern positions should be reported, with -1 reported if the pattern is not found. Text positions should be numbered from 0.

In the example below are 2 texts and 2 patterns. Here `pattern0.txt` is found at positions 445 and 666 in `text0.txt` and is not detected in `text1.txt`; `pattern1.txt` is found at position 3334 in `text0.txt` and at the start of `text1.txt`. Note that the lines in the output file may appear in any order and that the horizontal lines shown in the output here are for clarity only. Use a single space (no tabs) to separate the numbers on each line.

Input

0	0	0
1	0	0
0	1	0
1	1	0
0	0	1
1	0	1
0	1	1
1	1	1

Output

0	0	-2
0	0	455
0	0	666
1	0	-1
1	0	-1
0	1	-2
0	1	3334
1	1	-2
1	1	0

Test Data

Test data is available in the folder called `small_inputs` and the results file, `small_inputs.sorted` in `HPCProject.tar.gz`. These should be used to check the logic, not the performance, of your programs. It is strongly recommended that you devise your own test data to test the performance of your programs. There are also script files `execute_OMP` and `execute_MPI` which must be used to compile and run your programs.

These script files take a single parameter which is the name of a folder containing data. This will be linked to the name `inputs` for your program to read from. Note that any existing folder called `inputs` will be deleted so you should not store data in a file or folder called `inputs`. You may find the Unix commands `sort` and `diff` helpful when checking the output of your programs, for example your batch job might contain:

```
./execute_OMP small_inputs
sort -k 1,1n -k 2,2n -k 3,3n result_OMP.txt >sorted_OMP.txt
diff sorted_OMP.txt small_inputs.txt
```

Assessment

The assessment will be carried out in two parts:

i **Two programs - 20 marks each**: On Monday, December 03, 2018, the program submitted on QOL will be executed on the Kelvin cluster using both the supplied small input and a larger unseen collection of data. The same data sets will be used for assessing all student submissions. The programs will be run in batch, using the same `execute_OMP` and `execute_MPI` scripts that you have been given. Elapsed execution time will be returned by the `time` utility. You will be allocated marks depending on the correctness and performance of your programs.

Note: It is **mandatory** that you attend this session. Further details on the arrangements for the day will be available in due course.

ii **Presentation - 10 marks**: On Monday, December 10, 2018 you will need to individually give an oral presentation for **not more than 10 minutes** in which you describe, justify and defend your two HPC strategies. Further details will be available in due course.

Note: If I am unable to execute your program, then you will need to re-submit a working program. In the event of a resubmission that executes properly, the mark for your program will be capped at 8/20.

Submission

Your two programs must be submitted through Queen's Online as a zip file named ⟨student-number⟩.zip. A report is not required for the programming project.

Your presentation must be submitted in .ppt, .pptx or .pdf format.

All the best!