Assignment 2

Please read the document, ***General Requirements Programming Assignments***, posted on the course website and make sure you follow them when you do this assignment. This assignment has two parts. The first part is your first MPI program, a simple program designed as a warm-up exercise so that you overcome the technical hurdles of writing and running MPI programs and do not have to devise a complex algorithm. The second part is a performance analysis of the program.

1 Details

1. (80%) Design a parallel algorithm that, given two command line arguments, a string called the ***pattern***, and the name of a text file, searches for all occurrences of the pattern in the lines of the text file. For each line in which the pattern occurs, it prints that line on the standard output. If the pattern occurs more than once in a line, it just prints the line once. The order in which the lines are printed does not matter in this problem. For example, if the name of the program is find_matches, then this command

   ```
   find_matches  "parallel algorithm" lecture_notes
   ```

   should print every line in the file named `lecture_notes` that contains the string "parallel algorithm". Note that a string can contain blanks but it must be enclosed in quotes so that the shell treats it as one word. Write an MPI program to implement your algorithm.

   The program should exit if one or more arguments are missing, or if the input file cannot be opened for reading.

2. (20%) Do a performance analysis of your solution, i.e., the speedup of this program, like the ones we have done in Chapters 3 and 4 so far. Assume that the amount of time to check whether one string is contained in another string is a constant times the sum of the lengths of the two strings, i.e., it is $c \cdot (m+n)$, where m and n are the lengths of the two strings, and c is some arbitrary constant. Assume that the communication parameters are L and B, for the latency and bandwidth respectively. (This way you do not have to type Greek letters.)

   This part of the homework must comply with the ***General Requirements for Non-Programming Assignments***, i.e,. it must be typewritten and converted to a PDF. Create a title page with nothing but your name on it, and a second page with your analysis.

2 Program Grading Rubric

The program will be graded based on the following rubric out of 80 points:

- The program must compile and run on any `cslab` host. If it does not compile and link on any `cslab` host, it loses 60 points.
- **Correctness** (40 points)
  - The program should print exactly those lines that contain the pattern and no others.
  - The program should work correctly no matter how many processes it is given.
  - It should handle errors correctly.

- **Performance** (16 points)

  If this program is distributed among p processors, the speedup compared to a sequential implementation that checks the lines in sequence one by one, should be roughly proportional to p.

- **Design and clarity** (8 points)

  Is the program organized clearly? Are functions and variables designed in an appropriate way? If not, the program will lose points.

- **Compliance with the Programming Rules** (16 points)

  Are all of the rules stated in that document observed? Programs that violate them will lose points accordingly.

3 Submitting the Homework

This is a short assignment, due on a non-class meeting day. It is due by the end of the day (i.e. 11:59PM, EST) on October 14, 2019. Follow these instructions precisely to submit it.

Create a directory named named *`username_assignment2`*. where *username* is to be replaced by your login name on our network. Put the program source file and the performance analysis PDF into that directory. ***Do not place anything else into this directory*** - no object code, no data files, nothing else. You will lose 5% for each file that does not belong there, and you will lose 5% if you do not name the directory correctly[1].

Next, create a zip archive for this directory by running the zip command

```
zip -r username_hwk4.zip ./username_hwk4
```

This will compress all of your files into the file named `username_hwk4.zip`. ***Do not use the tar compress utility.***

Assuming this file is in your current working directory you submit by entering the command

```
submithwk_cs49365 -z  2  username_hwk4.zip
```

The `-z` tells the command it is a zip archive. The program will copy your zip file into the `hwk2` subdirectory

```
/data/biocs/b/student.accounts/cs493.65/hwks/hwk2/
```

and if it is successful, it will display the message, "`File ...  successfully submitted.`"

You will not be able to read this file, nor will anyone else except for me. But you can double-check that the command succeeded by typing the command

```
ls -l /data/biocs/b/student.accounts/cs493.65/hwks/hwk2/
```

and making sure you see a non-empty file there.

If you put a solution there and then decide to change it before the deadline, just replace it by the new version. Once the deadline has passed, you cannot do this. I will grade whatever version is there at the end of the day on the due date. You cannot resubmit the program after the due date.

[1] I have scripts that process your submissions automatically and misnamed files force me to manually override them.