Assignment 2

Parallel Searching using MPI

CSC4005 - High Performance Computing: Principles of Parallel Programming
Semester 1 (2020/21)

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

Submission Deadline: Thursday, 19 November 2020, 18:00

Contents

1 Assignment

This assignment builds on the previous assignment, please re-read carefully the specification given in Assignment 1. The primary aim is to design, implement and experiment with MPI programs to perform the same search process. You will be provided with a single text file, `text.txt`, and eight different pattern files, `pattern1.txt`, `pattern2.txt`, ... , `pattern8.txt`. These can be found in the folder `inputs`.

1.1 Test0

1. Modify the program `searching_sequential.c` so that it searches for the 8 patterns, sequentially, in a single batch run. You should use, `$time ./searching_sequential` to determine the total elapsed time which will be used as a reference point for comparison with your MPI implementations.

1.2 Test1

1. Re-engineer `searching_sequential.c` into a coarse-grain embarrassingly parallel MPI program, `searching_MPI_0.c`. Here each MPI process will perform a sequential search for a single pattern. For example, with 2 MPI processes P_0 will search for patterns 1, 3, 5 and 7 and P_1 will search for patterns 2, 4, 6 and 8. With 4 MPI processes P_0 will search for patterns 1 and 5, P_1 will search for patterns 2 and 6, P_2 will search for patterns 3 and 7 and P_3 will search for patterns 4 and 8. With 8 MPI processes P_i will search for pattern $i + 1$.

2. Using 8 cores in a batch job on the Kelvin cluster, run `searching_MPI_0.c` with 2, 4, and 8 processes. Use time to determine the elapsed time.

    a. Draw a graph of the parallel speedup (PS) vs the number of processes. PS is the elapsed time taken by the sequential program divided by the elapsed time taken using P processes.

    b. Draw a graph of the parallel efficiency (PE) vs the number of processes. PE is simply PS/P.

1.3 Test2

Consider the results of Test0 and Test1.

1. How can you improve the PS of `searching_MPI_0.c` **without** modifying the program. Note that this improvement will be specific to this input data.

2. Verify your hypothesis by running `searching_MPI_0.c` with 2, 4 and 8 processes.

    a. Draw a graph of the parallel speedup (PS) vs the number of processes.

    b. Draw a graph of the parallel efficiency (PE) vs the number of processes.

1.4 Test3

1. Design a fine-grain MPI program `searching_MPI_1.c`. This program should search for the 8 patterns in sequence but each search should be performed using n MPI processes, where n =2, 4 or 8. You must adopt a master-slave model. First, P_0, the master process, should read the text file, `text.txt`, and send the appropriate part to the slave processes. For each of the 8 patterns in sequence the master:

    a. should send the pattern to each slave;

    b. all processes, including the master, should search their allocated portion of the text;

    c. the master should collect the search results from the slave processes using `MPI_Reduce` and report the overall search result.

2. Using 8 cores in a batch job on the Kelvin Cluster, run `searching_MPI_1.c` with 2, 4, and 8 processes. Use time to determine the elapsed time.

a. Draw a graph of the parallel speedup (PS) vs the number of processes.

b. Draw a graph of the parallel efficiency (PE) vs the number of processes.

3. Comment on the performance of `searching_MPI_1.c`. Discuss how the performance might be improved.

2 Submission Instructions

You should submit your assignment to Canvas as a zip file named ⟨`student-number`⟩`.zip`. This must contain:

- A `source` folder, which contains the source code of the sequential program and two MPI programs along with corresponding job scripts. The source code submitted must be readable and written to high standards.
- An `output` folder, which contains the output of executing your three programs and/or screenshots of the output.
- A `report` folder in .PDF format, which contains a high quality report addressing all points raised above. The content of the report must be organised in relevant sections.

Note: The program names should be as specified. There will be penalties for not adhering to the constraints provided in the brief. Do not include the input folder containing the test cases with your submission.

3 Overview of Mark Distribution

This assignment carries 20% of the module mark.

The following is an overview of the mark distribution.

Source Code/Input/Output: 12/20 marks

- Test 0 - 2 marks
- Test 1 - 4 marks
- Test 2 - 1.5 marks
- Test 3 - 4.5 marks

Report: 8/20 marks

- Organisation of report and elegance of writing – 2 marks
- Results (graphs, tables) and discussion of algorithms as required each test – 4 marks
- Discussion on results obtained in each test – 2 marks

Note: Please adhere to hard constraints provided in the brief to avoid any penalties.

All the best!