**Project report for 5011CEM Big Data Programming module**

**Student ID: 10135033**

**Introduction**

During through whole semester, we have been provided with various MATLAB tasks (to learn about how to work with big data files) which overall combined would be the project we have to present. This code which will be presented here represents sequential processing, parallel processing. Also has an ability to test for not a number errors, text errors which accounts to automated testing, more information below. Main goal here is to compare sequential processing and parallel processing and provide a verdict.

**Code description**

The code itself is built that main file has ability to load data and do function calls. The main file also tests for text errors, if there are any, the code immediately stops from running and it is reported to the log file with the processing hour where the text error is. I’ve put other code into the functions (refer to list below, this is a brief description of functions)

* Sequential Processing – sequentially processes the data, records processing times, only called with a Sequential Processing Loop.
* Sequential Processing Loop – runs Sequential processing with specified data options also does a function call for NaN testing, runs until the end of the data options. Function is called in the main file.
* Parallel Processing – processes the data in parallel, records processing times. Function is called in the Parallel Processing Loop.
* Parallel Processing Loop – runs Parallel Processing with specified data options, number of processors, saves results for each processing times with different number of processors, also does NaN testing before beginning parallel processing data.
* Text Error Testing – tests for text errors, if they are present show in code line, does not report it to log file with processing hour where the error is.
* NaN Testing – tests for Not a Number errors if they are present, report it to log file with processing hour where the error is.
* Graph Plotting – the results that are from sequential and parallel processing are then and there is a graph made to compare results.

Keep in mind, whenever there is a NaN error, that processing hour will be just skipped and error logged in the log file, code keeps running with the next processing hour. There will be code explained in the other sections.

(CONTINUE)

**Comparisons of parallel and sequential timing**

During the development of the project, I have done basic flowcharts for the sequential processing and the parallel processing.

A picture containing diagram

Description automatically generated

Above there is a flowchart for the sequential processing. How it works, after running the code it must load the data, the number of processors will be only one as sequential processing will always use 1 processor. Then it processes sequentially the data (one by one), the code would record the processing time after processing each bit of data and then before the code finishes running it would plot the graph. The graph plot is required to see the results and compare them if required, that will be presented after.

(CONTINUE)

Also, I have done a flowchart for the parallel processing which looks like this:

A picture containing diagram

Description automatically generated

How parallel processing works is after running the code it loads the data, set the number of processors to 2 (mistake in the flowchart, should start from 2 processors instead of 0), then parallel process the data (how it works is allocating a processor to specific part of data, having multiple processors it will process the data much faster), records the processing time for each bit of data, try to increase a number of processors by 1 if it does not exceed max number of processors, parallel processing will be with 3 processors now and so on, if it will eventually exceed the max number of processors it will plot the results in graph.

(CONTINUE)

I have done a flowchart that mostly relates to the project code:

Diagram

Description automatically generated

Project code works in the way this flowchart is presented. To explain it as soon as main code is run (main code contains mostly function calls and data load, text error check too). The data is loaded, then there is a text error check, if there are text errors present, the code stops from running and logs in which processing hour there is an error, if not, the code goes into the not a number testing stage, if there are any non a number errors the processing hours with the errors will be logged and skipped. If there are no not a number errors the code will start sequentially processing the data with the given data options(specified in code), it will record the processing time for each bit of data, then as we approach parallel data processing, also processing times will be recorded and then increase number of processors by 1 if we don’t reach max number of processors, code keeps processing the data in parallel until max number of processors is reached, then plot the graph for sequential and parallel results to compare.

Processing 5000 data, 5 hours, maximum 8 processors.

Chart, line chart

Description automatically generated

This graph is produced by MATLAB. It represents processing time vs number of processors. The red star on the graph is the result of the sequential processing, it only uses 1 processor. Moving on to the blue line which has a downtrend meaning decrease of the time required to process the data. Best result in this case with parallel processing is achieved with 6 processors. The yellow line represents a prediction, as it produces negative values (those must be ignored), overall yellow line shows the decrease of time to process using more processors.

In comparison, sequential processing produces a slower result rather than parallel processing, you would ask why. Sequential processing only uses one processor, whilst parallel processing does use more processors, and that number of processors do work simultaneously, that makes a processing time reduction and increase in efficiency. That is a huge pro to parallel processing. Regarding of sequential processing, best result could be achieved with small data files, as parallel processing takes time to load up, also x amounts of processors with small data files would work slow in parallel.

(CONTINUE)

Regarding testing (For tests, I did 250 data, 5 hours, MAX number of processors – 3)

Lets test for the text errors with the TestFileText.nc

Graphical user interface, text, application, email

Description automatically generated

As soon as there is a text error present, the code stops from running.

Lets test for the NaN errors with the TestFileNaN.nc

Text

Description automatically generated

As soon there is a NaN error present, it is eported to the log file with the processing hour and that hour is skipped, the processing continues.

**SMART Targets**

* **Present code**
* **Compare sequential and parallel processing**
* **Show that the tests in the code are working**
* **Give recommendations**
* **Provide with the project summary**

**Conclusions and recommendations**

As solution is presented, we can conclude that parallel processing is faster than sequential processing, and itself it said the solution is for big data files. The code is universal and can be adapted to any data file, could process different amounts of data. Test text feature lack’s ability to report to log file. The recommended number of processors in this case is 6. 8 is too much. Also processing time can be affected with different data files, systems, it will never stay the same. Best choice is parallel processing.

**Comments to the report**

Github repository link: https://github.coventry.ac.uk/5011CEM-2122JanMay/5011CEM2122\_zimuse

CODE FOR THIS ASSESSMENT IS IN THE GITHUB REPOSITORY /assessment FOLDER

Gantt Chart is also in the github repository named “chart-10135033.gan”

Log book is also in the github repository named “logbook-10135033.docx”

**References**

ECMWF. (n.d.). *European Centre for Medium-Range Weather Forecasts*. Retrieved 17 April 2022, from https://www.ecmwf.int/

*Graph and Network Algorithms - MATLAB & Simulink - MathWorks United Kingdom*. (n.d.). MathWorks. Retrieved 17 April 2022, from https://uk.mathworks.com/help/matlab/graph-and-network-algorithms.html

Hyde, R. (n.d.). *5011CEM - Big Data Programming Project - 2122JANMAY*. Aula. Retrieved 17 April 2022, from https://coventry.aula.education/?#/dashboard/6a53a27d-9fda-42d6-913b-d3fb3fecb4bd/journey/materials/58d84445-ee0e-40fe-9b10-597b811b4589

*MathWorks - Academia*. (n.d.). MATLAB & Simulink. Retrieved 17 April 2022, from https://uk.mathworks.com/academia.html