Big Data Programming

Introduction:

Data processing is a crucial component of modern businesses and organizations, and the ability to process large volumes of data quickly and efficiently is essential for success. two different methods of processing data are Sequential and parallel processing, with each approach having its own advantages and disadvantages. Sequential processing refers to the processing of data one task at a time, while parallel processing involves the processing of multiple tasks simultaneously. This report will compare parallel and sequential data processing, looking at their respective advantages and disadvantages.

The report will also provide an estimation of the number of processors required to process 25 hours' worth of data in under two hours, along with testing the code to evaluate its handling of text and NaN values and suggestions to prevent code crashes.

The report will summarize the evidence generated during the project to arrive at conclusions, and recommendations on how to make informed decisions about how to process data quickly and efficiently.

Sequential Processing:

Sequential processing is a method of processing data where each task is executed in a specific order. The computer system performs one task at a time. Sequential processing is simple and easy to understand, but it can be time-consuming when dealing with a large amount of data. In sequential processing, the computer system must complete one task before starting the next, which can lead to the processor receiving more requests than it can process if the tasks are complex and take a long time to complete.

Parallel Processing:

Parallel processing is a method of processing data where multiple tasks are executed simultaneously. In parallel processing, the computer system divides the data into smaller chunks and processes each chunk at the same time. This results in faster processing times, as the computer system can process multiple tasks simultaneously. Parallel processing is particularly useful when dealing with large amounts of data, as it can significantly reduce the processing time.

Differences between Sequential and Parallel Processing:

Speed: Parallel processing is much faster than sequential processing, as it can process multiple tasks simultaneously, while sequential processing can only process one task at a time.

Complexity: Parallel processing is more complex than sequential processing, as it requires specialized hardware and software to divide the data into smaller chunks and process them simultaneously.

Resource Utilization: Parallel processing requires more resources, such as CPUs and memory, than sequential processing, as it processes multiple tasks simultaneously.

Coordination: Parallel processing requires coordination between the different tasks, as they are processed simultaneously. Sequential processing, on the other hand, does not require coordination between tasks, as they are executed one at a time.

Error Handling: In parallel processing, errors can occur due to the complexity of the system, as well as the interaction between different tasks. In sequential processing, errors are easier to handle, as each task is executed one at a time.

Smart Targets:

1.Load and process 1 hour of climate model output data at a time using parallel processing and ensure that the code can handle and report any errors within 4 weeks.

2.Develop and implement automated tests to ensure that the code can run without user intervention and provide accurate results for both normal and error cases within 4 weeks.

3.Increase the processing speed of analysing 1-hour climate model output data from 2.5 hours to under 2 hours using parallel processing within 7 weeks.

4.Test and compare the processing speed of sequential and parallel processing using a sample data set of 250 and 5000 data values output data and provide a report with detailed findings and recommendations within 10 weeks.

5.Reduce the time required to analyse 25 hours of climate model output data from over 60 hours to under 2 hours using parallel processing within 10 weeks.

6.Extrapolate the findings of the processing speed comparison to estimate the number of processors required to achieve the target processing time for the entire data set of over 277,000 data values within 12 weeks.

The data file is firstly tested to check for any values that may produce errors when processing

TestText.m is used to examine a file and check for non-numeric values (chars only).

The code retrieves the contents of the file and stores it in a variable "Contents". It then opens the file with a read-only permission.

The script loops through each variable in the file and reads the data type of each variable. It stores the data type of each variable in an array "datatype".

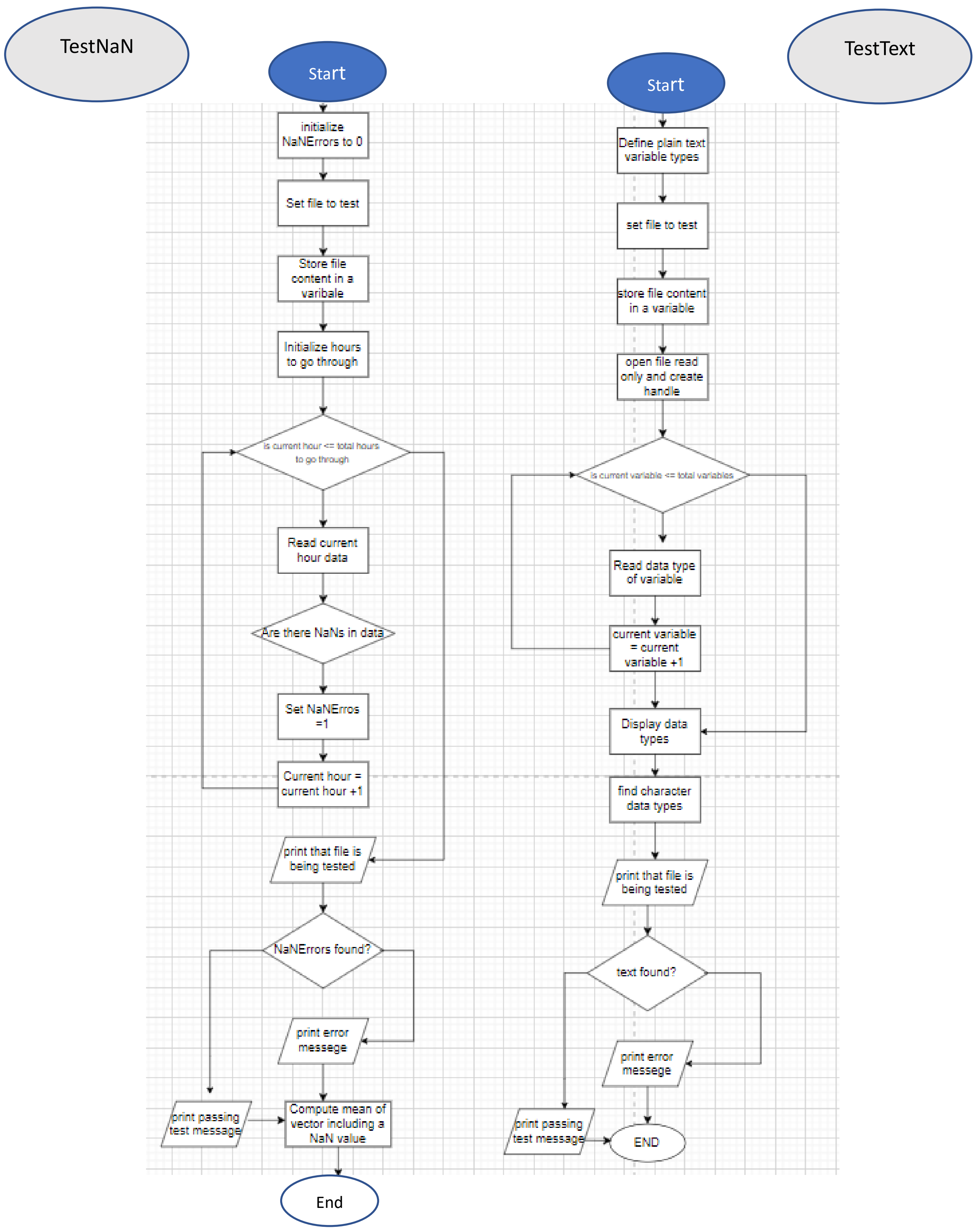
It then displays the data types in the file and checks if there are any character data types by comparing the data in the file to NaN values. If there are any character data types, the script prints an error message. If there are no character data types, the script prints a message stating that all data is numeric and continues with the analysis.

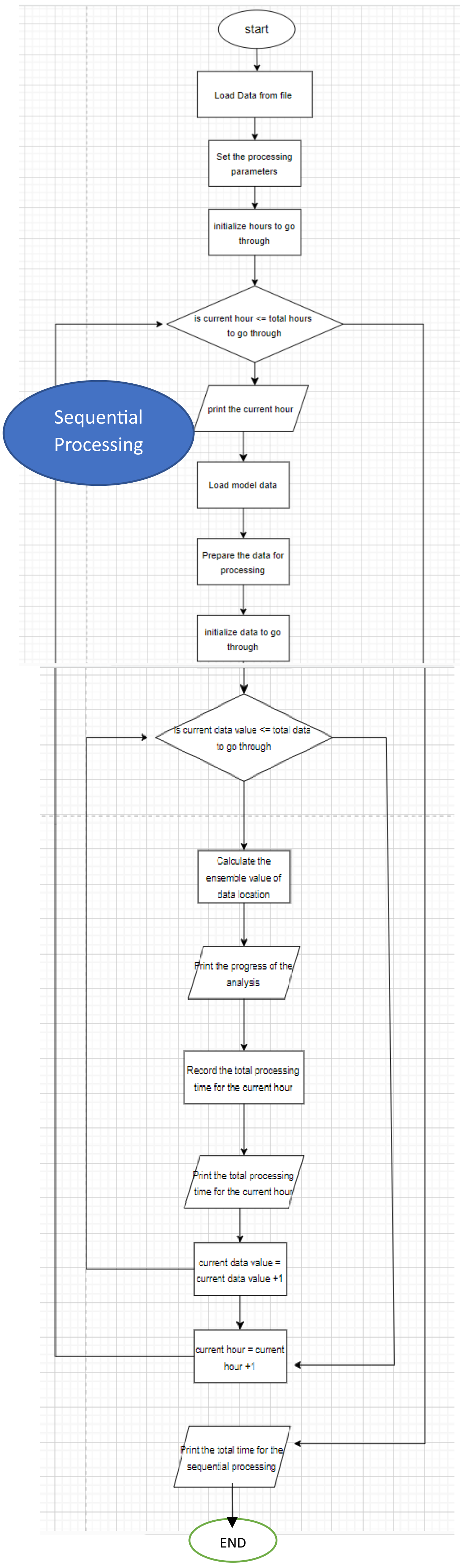
TestNan.m is used to examine a file and checks for NaN (not a number)

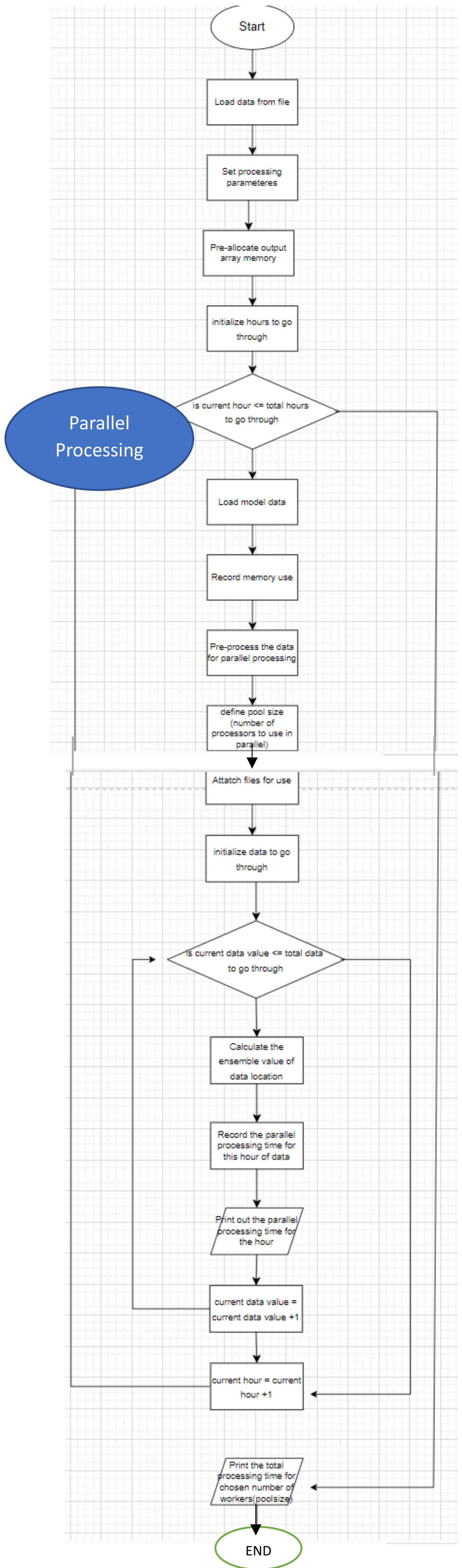
This code also retrieves the contents of the file and stores it in a variable "Contents".

the code checks for NaNs If any NaNs are present in the data code prints a message indicating whether NaN errors are present or not.

The data passed both tests and can now be processed.


TestNaN
Start
initialize NaNErrors to 0
Set file to test
Store file content in a varibale
Initialize hours to go through
is current hour <= total hours to go through
Read current hour data
Are there NaNs in data
Set NaNErros =1
Current hour = current hour +1
print that file is being tested
NaNErrors found?
print error messege
print passing test message
Compute mean of vector including a NaN value
End
TestText
Start
Define plain text variable types
set file to test
store file content in a variable
open file read only and create handle
is current variable <= total variables
Read data type of variable
current variable = current variable +1
Display data types
find character data types
print that file is being tested
text found?
print error messege
print passing test message
END


start
Load Data from file
Set the processing parameters
initialize hours to go through
is current hour <= total hours to go through
Sequential Processing
print the current hour
Load model data
Prepare the data for processing
initialize data to go through
is current data value <= total data to go through
Calculate the ensemble value of data location
Print the progress of the analysis
Record the total processing time for the current hour
Print the total processing time for the current hour
current data value = current data value +1
current hour = current hour +1
Print the total time for the sequential processing
END


Start
Load data from file
Set processing parameteres
Pre-allocate output array memory
initialize hours to go through
is current hour <= total hours to go through
Parallel Processing
Load model data
Record memory use
Pre-process the data for parallel processing
define pool size (number of processors to use in parallel)
Attatch files for use
initialize data to go through
is current data value <= total data to go through
Calculate the ensemble value of data location
Record the parallel processing time for this hour of data
Print out the parallel processing time for the hour
current data value = current data value +1
current hour = current hour +1
Print the total processing time for chosen number of workers(poolsize)
END

Coded files have been created to process the data accordingly. The first file to be explained is the SequentialProcessing file.

SequentialProcessing is split into parts. The first part defines the file name and path of the NetCDF file containing the data to process specifically atmospheric ozone data. It also loads the relevant information needed from that file.

The second part sets the processing parameters set by the customer. These parameters define how the data will be processed.

The main part of the code goes through each hour of data in the NetCDF file. The hours can be changed to run from any certain hour to another. It then loads the required data and analyses it loading and printing the analysis to the console every 50 iterations.

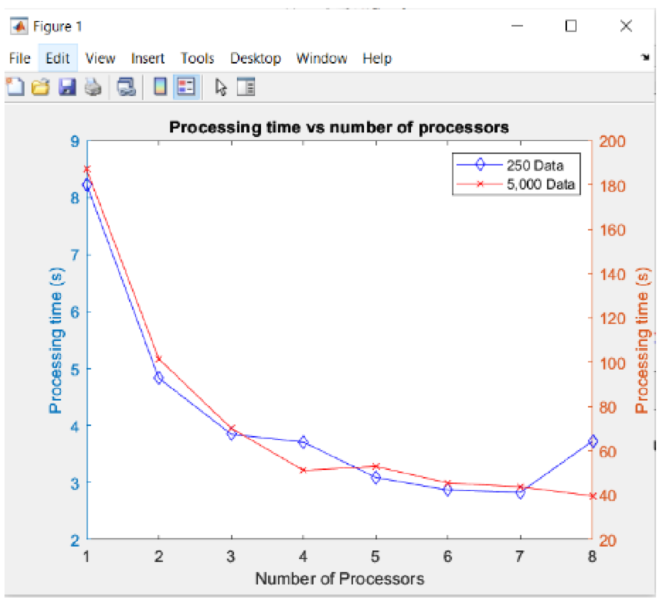
Finally the code records, stores and prints the total time it took to process and analyse the total data which is also set and can be changed at any time.

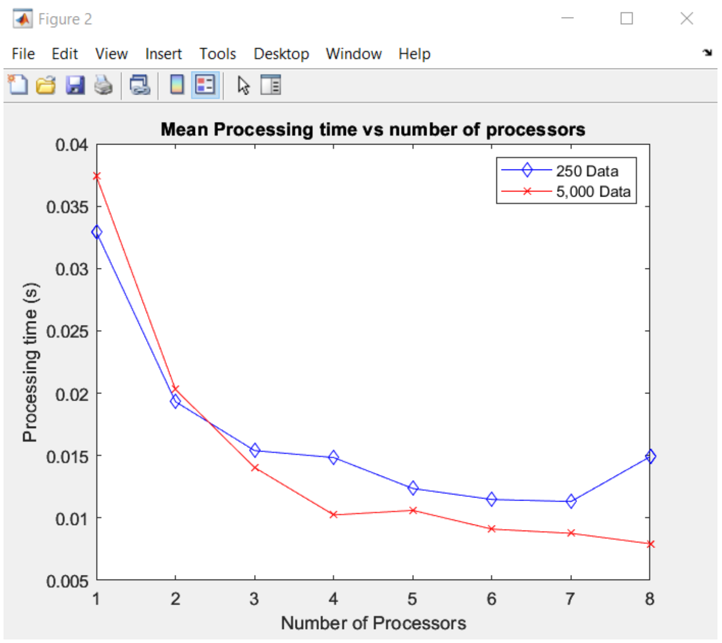
ParallelProcessing is similar to SequentialProcessing however now the total process time can be decreased. In this file the number of processors used can be changed to execute multiple tasks simultaneously. The machine used to process the data has a total of 8 processors so this code file was used to compare the total process time using different amounts of processors.

The results from SequentialProcessing and ParallelProcessing were used to create graphs using the Graphs.m file. This file plots two graphs in MATLAB that show how the number of processors affects the processing time of the data. The graph uses two different values for the amount of data to be processed for this.

The first graph shows the two sets of data on different y-axes with the actual data values. The second graph shows the two sets of data on the same y-axis, representing the mean processing time for each set of data.

The data values and y and x values are manually inputted in the file.





Matlab R2023a, Release 2023

The graphs show how the processing time of data with values 250 and 5,000 changes with different number of processors. From figure 1 it is highlighted that as the number of processors increase the processing time decreases. There is a significant change in the processing time from 1 processor (sequential processing) to 7/8 processors.

With 250 data values the processing time decreased by more than half. From 8.2 seconds to 3 seconds with 7 processors.

The processing time for 5,000 data values decreased significantly more. With 1 processor the processing time was 190 seconds however with 8 processors the processing time was 40 seconds.

From 1 processor to 2 there is a big change in the processing time. As it shows in figure 1 with 250 data values it changes from 8.2 seconds to 4.9 and with 5,000 data values it changes from 190 seconds to 100 seconds. After the second processer it gradually decreases. However as it can be seen sometimes from one processor to the next the time may not change much and instead of decreasing it increases slightly for example from processor 4 to 5 with 5,000 data values and from processor 7 to 8 with 250 data values. With the larger dataset it is generally decreasing with the highest number of processors achieving the lowest processing time. However, with smaller datasets since in the first place it doesn't take long to processes it sequentially using 8 processors as can be seen may instead hinder the performance. This is because it is using more processors than needed. There is also no need to use multiple processors for small data sets since it would cost more and consume more energy.

However, the data to be processed is very large so in this case using more processors is better. The processors represent the number of workers. To process the required 25 hours' worth of data in under 2 hours the graph in figure 1 is used to extrapolate the values needed to calculate how to achieve this processing goal.

The time is multiplied by the total number of data divided by the number of data actually analysed. This will give the total time for one hour. This value is then multiplied by 25 to get the time for all the hours.

Using the values from 5,000 data values with 8 processors the following calculation is carried:

40 x (277 804 / 5000) = 2222.4 (1dp)

= 0.62 hours (2dp) = 1 hour worth of data

25 hours = 0.62 x 25 = 15.5 hours

This is how long it takes 8 workers to process the 25 hours' worth of data

To get this value to less than 2 hours the data needs to be processed at least 8 times faster. To estimate how many workers would be needed for this the number of current workers is multiplied by how many times faster we need the data to be.

The following calculation is carried:

8 x 8 = 64

To

25 hours = 0.62 x 25 = 15.5 hours

8x8 = 64

In order to achieve the processing goal; 64 workers are needed.

It is important to consider the limitations of the analysis and how different systems may perform differently. This can be influenced by a variety of factors, such as the hardware configuration, software stack, and workload characteristics for example some systems may have less processors than the one used.

Another limitation to consider is that as the number of processors increases, the amount of communication required to coordinate and synchronize the workload also increases. This may become a bottleneck (flow of data is slowed down).

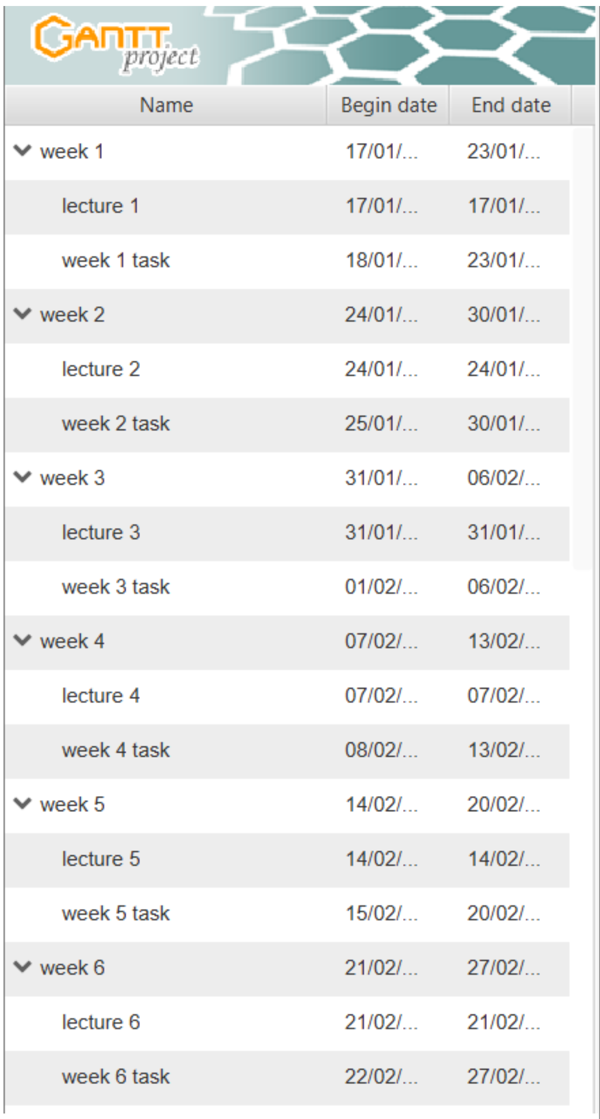
The results from the analysis may also be possibly improved as the use of HPC (High-Performance Computing) may significantly improve the results. HPC systems typically consist of clusters of servers, each equipped with multiple processors, and large amounts of memory and storage. Making them ideal for tasks that are required to be processed in as little time as possible (Wikipedia contributors, 2023, March 30).

The use of cloud computing would also be beneficial especially for a large scale of data values. Cloud computing platforms allow users to access large-scale computing resources on-demand without the need for large upfront investments in hardware for storing the data (Chai, W., & Bigelow, S. J, 2022, November 10).

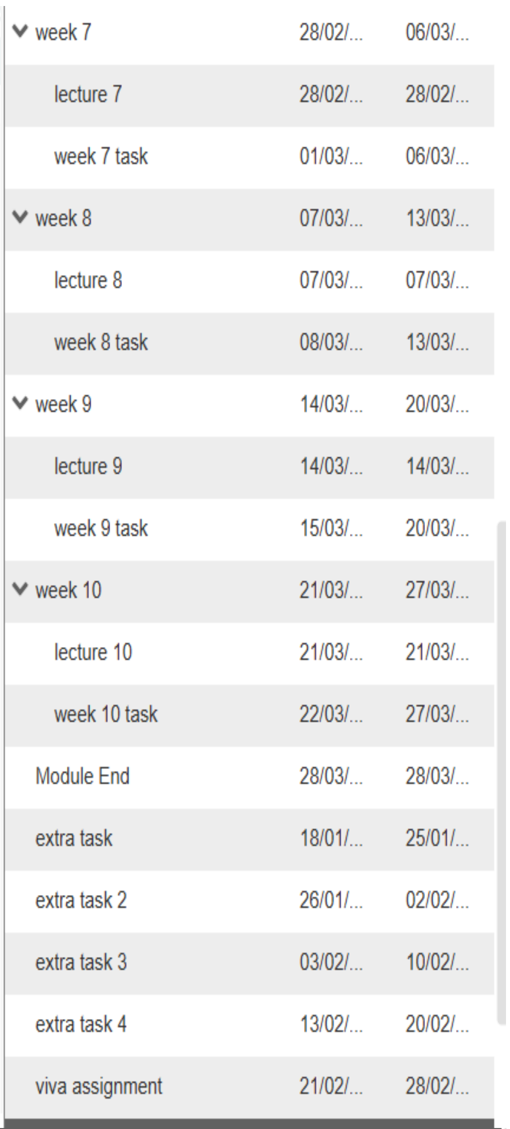
Conclusion:

In conclusion, parallel processing is faster and more efficient than sequential processing, but it requires more resources and is more complex. Sequential processing is simpler and easier to understand, but it can be time-consuming when dealing with a large amount of data. The choice between sequential and parallel processing depends on the task and the resources available. For tasks that require a small amount of data to be processed, sequential processing would be suitable, while for tasks that require a large amount of data to be processed, parallel processing may be necessary.

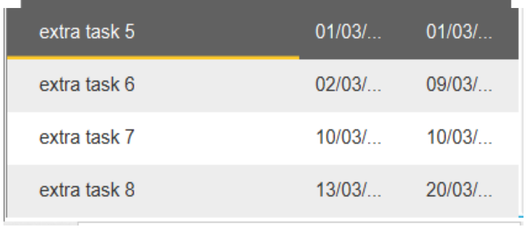
Gannt Chart



Name	Begin date	End date
week 1	17/01/...	23/01/...
lecture 1	17/01/...	17/01/...
week 1 task	18/01/...	23/01/...
week 2	24/01/...	30/01/...
lecture 2	24/01/...	24/01/...
week 2 task	25/01/...	30/01/...
week 3	31/01/...	06/02/...
lecture 3	31/01/...	31/01/...
week 3 task	01/02/...	06/02/...
week 4	07/02/...	13/02/...
lecture 4	07/02/...	07/02/...
week 4 task	08/02/...	13/02/...
week 5	14/02/...	20/02/...
lecture 5	14/02/...	14/02/...
week 5 task	15/02/...	20/02/...
week 6	21/02/...	27/02/...
lecture 6	21/02/...	21/02/...
week 6 task	22/02/...	27/02/...



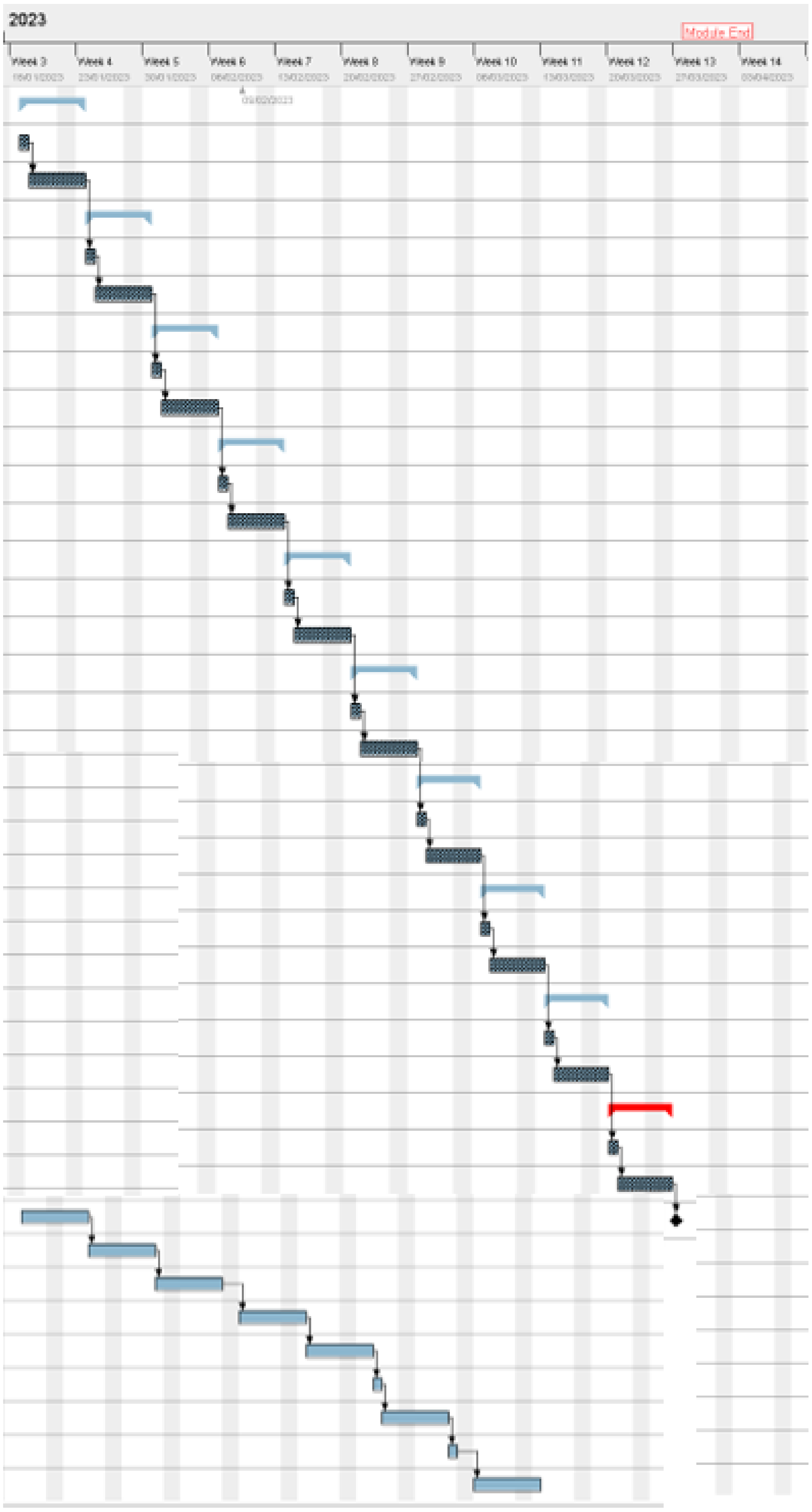
week 7	28/02/...	06/03/...
lecture 7	28/02/...	28/02/...
week 7 task	01/03/...	06/03/...
week 8	07/03/...	13/03/...
lecture 8	07/03/...	07/03/...
week 8 task	08/03/...	13/03/...
week 9	14/03/...	20/03/...
lecture 9	14/03/...	14/03/...
week 9 task	15/03/...	20/03/...
week 10	21/03/...	27/03/...
lecture 10	21/03/...	21/03/...
week 10 task	22/03/...	27/03/...
Module End	28/03/...	28/03/...
extra task	18/01/...	25/01/...
extra task 2	26/01/...	02/02/...
extra task 3	03/02/...	10/02/...
extra task 4	13/02/...	20/02/...
viva assignment	21/02/...	28/02/...



extra task 5	01/03/...	01/03/...
extra task 6	02/03/...	09/03/...
extra task 7	10/03/...	10/03/...
extra task 8	13/03/...	20/03/...

GanttProject, Release 2023

j


2023
Module End
Week 3
16/01/2023
Week 4
23/01/2023
Week 5
30/01/2023
Week 6
06/02/2023
Week 7
13/02/2023
Week 8
20/02/2023
Week 9
27/02/2023
Week 10
06/03/2023
Week 11
13/03/2023
Week 12
20/03/2023
Week 13
27/03/2023
Week 14
03/04/2023
09/02/2023

Log book

5011 logbook

18 jan - started planned project on Gannt chart. struggling to create a subtask end date for week 1 changes to the end date of sub task added.

20 jan - asked for help with gannt chart subtask. added 2 subtasks and made sure that the end date of the last sub task was the same as the end date of the week. now it doesnt change.

18 feb - added 03_surface file to 2a and 2b folder.

commit all changes to github (changed file names)

ran MemorySaving.m added breakpoints at lines 23, 46, 72.

output for code ran up to line 22 :

```
Loading chimere_ozone

Loading emep_ozone

Loading ensemble_ozone

Loading eurad_ozone

Loading lotoseuros_ozone

Loading match_ozone

Loading mocage_ozone

Loading silam_ozone

Memory used for all data: 224.000 MB
```

output for code ran from line 23 - 45

```
Loading chimere_ozone

Loading emep_ozone

Loading eurad_ozone

Loading lotoseuros_ozone

Loading match_ozone
Loading mocage_ozone
Loading silam_ozone
Memory used for 1 hour of data: 7.840 MB
```

output for code ran from line 46 to 71

```
Loading chimere_ozone
Loading emep_ozone
Loading eurad_ozone
Loading lotoseuros_ozone
Loading match_ozone
Loading mocage_ozone
Loading silam_ozone
Loaded Hour 25, memory used: 7.840 MB
```

code ran from 72

```
Results:
Memory used for all data: 224.00 MB
Memory used for hourly data: 7.84 MB
Maximum memory used hourly = 7.84 MB
Hourly memory as fraction of all data = 0.03
```

divided functions in memorysaving file into seprate scripts within the same folder

added all

copied

19/02/23

followed example to turn LoadAllData into a function

had an error needed to add end to the end of the function

commented out original code from LoadAllData and added new code.

tested out the function in the main script and recieved the same output as before so it works fine.

commited all changes and pushed to git.

commented out the original code in loadHours.

turned it into a function. had a syntax error forgot the capital n in FileName.

tested the function in the meain script MemorySaving.m = code works.

commited changes and pushed onto github

coppied the code again at the bottom and commented out original code (keep for refrence incase ant mistake is made) from LoadAllHours and added function [HourMem] = LoadAllHours(FileName)

Contents = ncinfo(FileName) turned into a function

added function to main script checked it works = it did

commit changes and push to git

1. LoadAllData

2. LoadHours

3. LoadAllHours

4. Reportresults

sripts to git

commit changes and push

ran SequentialProcessing.m got a message teling me whether to add file to path or change it. i chose to add it. it then ran.

predicted time for all the data to run is around 2 hrs chnaged line 27 in sequentialProcessing from 1:25 to 1:1

and started running it

Processing hour 1 - 2090.33 s

Total time for sequential processing = 2091.71 s

changed line 27 to process hour 12

Processing hour 12 - 2610.31 s

Total time for sequential processing = 2611.52 s

changed the for loop in sequential processing line 45 to only go thru 1 data:

from 1: to 1:1

ran the code and got the following code:

```
>> SequentialProcessing
Processing hour 12
Creating segments....Segments created

ans =

     277804

Processing hour 12 - 0.10 s

Total time for sequential processing = 1.27 s
```

now will change line 27 to run all 25 hours and see if they all work 1:25

output:

```
>> SequentialProcessing
Processing hour 1
Creating segments....Segments created

ans =

     277804

Processing hour 1 - 0.08 s

Processing hour 2
Creating segments....Segments created

ans =

     277804

Processing hour 2 - 0.11 s

Processing hour 3
Creating segments....Segments created

ans =

     277804

Processing hour 3 - 0.12 s

Processing hour 4
```

```
Creating segments....Segments created

ans =

      277804

Processing hour 4 - 0.14 s

Processing hour 5
Creating segments....Segments created

ans =

      277804

Processing hour 5 - 0.16 s

Processing hour 6
Creating segments....Segments created

ans =

      277804

Processing hour 6 - 0.18 s

Processing hour 7
Creating segments....Segments created

ans =
    277804

Processing hour 7 - 0.19 s

Processing hour 8
Creating segments....Segments created

ans =

    277804

Processing hour 8 - 0.20 s

Processing hour 9
Creating segments....Segments created

ans =

    277804

Processing hour 9 - 0.21 s

Processing hour 10
Creating segments....Segments created

ans =

    277804

Processing hour 10 - 0.23 s
Processing hour 11
Creating segments....Segments created

ans =

      277804

Processing hour 11 - 0.24 s

Processing hour 12
Creating segments....Segments created

ans =

      277804

Processing hour 12 - 0.26 s

Processing hour 13
Creating segments....Segments created

ans =

      277804

Processing hour 13 - 0.27 s

Processing hour 14
Creating segments....Segments created

ans =
    277804

Processing hour 14 - 0.29 s

Processing hour 15
Creating segments....Segments created

ans =

    277804

Processing hour 15 - 0.31 s

Processing hour 16
Creating segments....Segments created

ans =

    277804

Processing hour 16 - 0.32 s

Processing hour 17
Creating segments....Segments created

ans =

    277804

Processing hour 17 - 0.34 s
Processing hour 18
Creating segments....Segments created

ans =

     277804

Processing hour 18 - 0.35 s

Processing hour 19
Creating segments....Segments created

ans =

     277804

Processing hour 19 - 0.38 s

Processing hour 20
Creating segments....Segments created

ans =

     277804

Processing hour 20 - 0.40 s

Processing hour 21
Creating segments....Segments created
```

```
ans =

     277804

Processing hour 21 - 0.42 s

Processing hour 22
Creating segments....Segments created

ans =

     277804

Processing hour 22 - 0.44 s

Processing hour 23
Creating segments....Segments created

ans =

     277804

Processing hour 23 - 0.46 s

Processing hour 24
Creating segments....Segments created

ans =

     277804
```

```
Processing hour 24 - 0.47 s


Processing hour 25
Creating segments....Segments created

ans =

    277804


Processing hour 25 - 0.50 s


Total time for sequential processing = 31.02 s
```

they all ran successfully

in order to do task 3 i changed line 27 in sequential processing to NumHour = [1,12,24] to run these 3 seprate hours.

also changed line 45 to idx = [1,12,24]

ran the code

output:

```
>> SequentialProcessing
Processing hour 1
Creating segments....Segments created

ans =

    277804
```

```
ans =

     277804


ans =

     277804

Processing hour 1 - 0.15 s

Processing hour 12
Creating segments....Segments created

ans =

     277804


ans =

     277804


ans =

     277804

Processing hour 12 - 0.20 s
Processing hour 24
Creating segments....Segments created

ans =

      277804



ans =

      277804



ans =

      277804

Processing hour 24 - 0.24 s

Total time for sequential processing = 3.73 s
```

20/02/23

commented out waitbar in parallel processing line 109.

ran sequentialprocessing with for idx = 1:250

output:

```
......
Total 250 of 277804, last 50 in 0.32 s  predicted time for all data 43860.3 s
```

```
Processing hour 24 - 5.12 s

Total time for sequential processing = 9.58 s
```

changed pool size in parallel processing to 1. output:....

```
Parallel processing time for hour 5 : 1.5 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 1 workers = 8.23 s
```

pool size = 2

output:...

```
Parallel processing time for hour 5 : 0.9 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 2 workers = 4.83 s
```

pool size = 3

output:...

```
Parallel processing time for hour 5 : 0.7 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 3 workers = 3.85 s
```

pool size = 4

output:...

```
Parallel processing time for hour 5 : 0.6 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 4 workers = 3.71 s
```

pool size = 5

output:...

```
Parallel processing time for hour 5 : 0.4 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 5 workers = 3.09 s
```

pool size = 6

output:...

```
Parallel processing time for hour 5 : 0.4 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 6 workers = 2.87 s
```

pool size = 7

output:Parallel processing time for hour 5 : 0.4 s

```
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 7 workers = 2.83 s
```

(the 1st time i ran with pool size = 7 the total processing time was around 3.5 seconds

pool size = 8

output:...

```
Parallel processing time for hour 5 : 0.5 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 8 workers = 3.73 s
```

i will run 1 more time to see if the time goes down...

output:

```
Parallel processing time for hour 5 : 0.4 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 8 workers = 4.00 s
```

time increased. maybe 8 processors is too much changed for idx to idx = 1:2500

output:....

```
Total 2500 of 277804, last 50 in 0.60 s  predicted time for all data 83271.9 s
Processing hour 24 - 73.12 s

Total time for sequential processing = 76.52 s
```

changed for idx to idx = 1:5000 in sequential processing

output:

```
Total 5000 of 277804, last 50 in 0.46 s  predicted time for all data 64034.1 s
Processing hour 24 - 129.52 s

Total time for sequential processing = 132.89 s.
```

chnaged parfor idx to parfor idx = 1:5000

poolsize = 1

output:....

```
Parallel processing time for hour 5 : 37.1 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 1 workers = 186.97 s
```

poolsize = 2

output:...

```
Parallel processing time for hour 5 : 20.0 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 2 workers = 101.51 s
```

```
poolsize = 3

output:...

Parallel processing time for hour 5 : 12.4 s

Parallel pool using the 'Processes' profile is shutting down.

Total processing time for 3 workers = 70.23 s


poolsize = 4

output:...


Parallel processing time for hour 5 : 9.6 s

Parallel pool using the 'Processes' profile is shutting down.

Total processing time for 4 workers = 51.22 s


poolsize = 5

output:...

Parallel processing time for hour 5 : 10.9 s

Parallel pool using the 'Processes' profile is shutting down.

Total processing time for 5 workers = 53.03 s


poolsize = 6

output:...

Parallel processing time for hour 5 : 8.6 s

Parallel pool using the 'Processes' profile is shutting down.

Total processing time for 6 workers = 45.53 s


poolsize = 7

output:...

Parallel processing time for hour 5 : 7.3 s

Parallel pool using the 'Processes' profile is shutting down.

Total processing time for 7 workers = 43.83 s
```

poolsize = 8

output:...

```
Parallel processing time for hour 5 : 7.4 s
Parallel pool using the 'Processes' profile is shutting down.
Total processing time for 8 workers = 39.66 s
```

from the graphs we can generally say that as the number of processors increase the processing time (s) decrease. however with smaller number of data such as 500 it doesnt make that much difference and in some cases the processing time increases with more processors.

however with larger number of data e.g 5000 the processing time continues to decrease as the number of processors increase.

however as it reaches 6 processors it doesnt change much after that. is it worth using more processors and therefore storage?

commited changes and pushed onto git

task 4 do it laterrrrr

created flowchart for sequential processing

complete later task 3 and 4.

SMART Targets task 1

Outline the main steps and describe each.

1. Boil the kettle
2. Put the tea bag in the mug
3. Pour in the water
4. Add sugar

5. Add milk

1. get kettle, open tap, fill kettle enough for your cup of tea, close the tap, put kettle in its place, make sure kettle is plugged in, turn the kettle on, wait until the kettle has boiled and come to a stop.

2. get a clean mug, grab the tea bag of your choice, put the teabag in the mug.

3. get the kettle, fill the mug up, let the tea bag soak then take it out.

4. get the sugar, get a tea spoon, put as many tea spoons of sugar as you'd like, mix the tea using the teaspoon.

5.get the milk, open it, pour in the desired amount, mix with the teas poon, tea is ready!

21/02/23

changed filename on TestText.m line 39 to 'TestyTest.nc'

output after running TestText.m :..........

```
Testing file: TestyTest.nc
All data is numeric, continue analysis.
```

changed line 11 in TestText.m to filename = 'TestyTest.nc'

changed file name in line 39 to filename = 'C:/Users/Fatima Al-Babily/Downloads/code5011/Model1/TestFileText.nc'

the 1st file should pass the test and the second test shouldnt pass the test.

output:....

```
Testing file: TestyTest.nc
All data is numeric, continue analysis......
Testing file: C:/Users/Fatima Al-Babily/Downloads/code5011/Model1/TestFileText.nc
Error, text variables present:
```

copied testfilenan.nc to 6b folder

inserted 'TestFileNaN.nc into file name in folder TestNan.m line 12 and 46

added mean([2, NaN, 4]) to line 40 and 75 in file TestNan.m

```
output =

>> TestNan

NaNs present

Testing files: TestFileNaN.nc

NaN errors present!


ans =


   NaN


Testing files: TestFileNaN.nc

NaNs present during hour 12

NaN errors present!


ans =


   NaN
```

commented out mean([2, NaN, 4]) and under it put nanmean([2, NaN, 4]).

```
output:>> TestNan

NaNs present

Testing files: TestFileNaN.nc

NaN errors present!


ans =
    3
```

```
Testing files: TestFileNaN.nc
NaNs present during hour 12
NaN errors present!

ans =

    3
```

The results are what was expected

mean([2, NaN, 4]) returned NaN because of the presence of NaN (not a number) value in the input, whereas nanmean([2, NaN, 4]) returned 3, which is the mean of [2, 4] after excluding the NaN value.

as mean function returns the arithmetic mean of the given values, whereas nanmean function returns the mean of the given values after excluding any NaN (Not-a-Number) values.

22/02/23

task 1 automated Testing...

commented out'clear all' line 2 in sequentialProcessing and parallelprocessing

a. Change the loop size to a variable in the processing code.

b. Define the loop variable options in your test code, i.e. the number of data to

process.

to do this i changed line 26 to LoopSize = [1,12,24];

then i also wrote under it for NumHour = LoopSizes

changing code again...

line 26 sequentialProcessing =

LoopSize = 3; % define the number of hours to loop through

LoopOptions = [1, 12, 24]; % define the hour options for the loop

tic

for LoopIndex = 1:LoopSize %loop through each hour

moved the tic function after defining the variables above the loop#

changed line 49...

dataLength = size(Data2Process, 1);

for idx = 1:dataLength

deining a variable loop allows for more flexibility in changing the number of data locations to process

researched how to store the result of sequentailprocessing into an array.

found out i can either use a cell array or char array. char array would limit me to only text. cell array allows me to store any data tyoe including other arrays.

used the function sprintf to put together a string and other data type

changed line 74 to..

SeqResult = sprintf('Total time for sequential processing = %.2f s\n\n', tSeq);

% store the SeqResult into a call array

result{1} = SeqResult;

disp(result{1})

realised i cant change the size of the data processed now.

so changing codee...

line 50 dataLength = 100;

for idx = 1:dataLength

i used a small data length to check that it works...

output:

```
>> SequentialProcessing

Processing hour 1

Creating segments....Segments created

Total 50 of 277804, last 50 in 0.29 s  predicted time for all data 40014.0 s

Total 100 of 277804, last 50 in 0.26 s  predicted time for all data 35844.5 s

Processing hour 1 - 0.79 s


Processing hour 12

Creating segments....Segments created

Total 50 of 277804, last 50 in 0.28 s  predicted time for all data 39565.7 s

Total 100 of 277804, last 50 in 0.28 s  predicted time for all data 39455.2 s

Processing hour 12 - 1.22 s


Processing hour 24

Creating segments....Segments created

Total 50 of 277804, last 50 in 0.26 s  predicted time for all data 36590.4 s

Total 100 of 277804, last 50 in 0.26 s  predicted time for all data 36575.5 s

Processing hour 24 - 1.64 s


Total time for sequential processing = 4.60 s
```

it works!

part 3 in task1 automated testing says to change the PoolSize to a variable but its already a variable.

24/02/23

part 3b change parfor loop size into a variable...

line 102..

comment out original code

dataLength = 100;

parfor idx = 1:dataLength Num2Process;

check code still works.. it works

matlab states that Num2Process on line 104 produces code that may be unused i will remove it and see if code still works

when removed the total processing was reduced from 4.24 s to 3.76 s

committed and pushed changes to git

06/03/23

change loop size sequential processing line 27 from = 3 to = 25

change loopoptions line 28 from [1, 12, 24]; to 1;

these changes were made in order to complete the projcet scenario:

3. Each day, they need to analyse the new data set for that day, so they wish to complete the analysis of the data for a 24-hour period (25 data sub-sets) in under 2 hours.

4. It is not possible to hold all this in memory at one time, so the new process should load only 1 hour of data for processing at a time. If parallel processing is to occur, then 1 hour of data per worker can be loaded as needed.

ran code got following error:

```
Index exceeds the number of array elements. Index must not exceed 1.

Error in SequentialProcessing (line 32)
    NumHour = LoopOptions(LoopIndex);
```

changed line 27 loopsize from 25 to 1:25

changed for loopIndex line 31 from = 1:loopszie to =loopsize changed line 28 loopoptions to = loopsize

chanhed loopsize = 1:25

loopoptions = 1:25

changed loopsize bk to 25

changed LoopIndex bk to 1:LoopSize

change datalength variable line 50 from 100; to Data2Process recieved error message that it needs to be a scalar value

now changed datalength to = 277804 (total amount of data)

07/03/23

made a new script named seqnew.m ub 3a sequential folder

put in the original code of sequential

in seqnew changed line 27 from 1:25 to 1:1

change line 45 to go thru only one hour from 1: to 1:1 commented out rest of code on that line

added the new script to git.

changed line 27 in seqnew to 1:25 to run through 25 hrs

and line 45 to run through 1:500 to run thru only 500 bits of data out of the 277804

ran the code

output:

Total time for sequential processing = 158.91 s now i need to calculate how long it would have taken to process all the data

calculation:

277804/500 = 555.608

555.608 x 158.91s = 88,292 seconds = around 24.5 hrs to process all the data

this how long it would take to process all the data using sequential processing.

changing one of the graphs in graphs.m to represent sequential processing.

i will change the 250 graph.

changing 250 to 500.

what i should do...

have 1 line represent sequential and 1 line represent parallel.

x axis = data size

y axis = execution time (seconds)

changing the size of data in seqnew line 45 to 100

total time for sequential processing of 100 bits of data = 63.51s

changing the size of data to 750 line 45

total time for sequential processing of 750 bits of data = 205.84s

changing the size of the data to 1000

total time for sequential processing of 1000 bits of data = 226.78s changing the size of data line 45 to 1500

total time for sequential processing of 1500 bits of data = 325.59

changing size of data line 45 to 3000

total time for sequential processing of 3000 bits of data =
789.11s

27/03/23
changed line 39 in TestText.m to TestyTest.nc and commit changes

fixed path in file CreateTestData_Text.m line 5
changed the model2change line 19 to a number >7 changed to 8

recieved output:

```
Testing file: TestyTest.nc
All data is numeric, continue analysis.
```

29/03/23
finalised flowcharts

30/03/23
created automated test for sequential processing
creating automated test for parrallel

31/03/23
added additional comments.

References

Chai, W., & Bigelow, S. J. (2022a, November 10). cloud computing. Cloud Computing. https://www.techtarget.com/searchcloudcomputing/definition/cloud-computing

Wikipedia contributors. (2023b, March 30). High-performance computing. Wikipedia. https://en.wikipedia.org/wiki/High-performance_computing

Matlab R2023a, Release 2023. Computer Software.

GanntProject, Release 2023. Computer Software.

Link to GitHub repository : https://github.coventry.ac.uk/5011CEM-2223JANMAY/5011CEM2223_babilyf.git