Nikita Gaurihar

NUID - 002980962

ASSIGNMENT 03

CSYE7105

High Performance Parallel Machine Learning and AI

**Part 1**

1. **Use Pool.apply() to get the row wise common items in list\_a and list\_b; and print the result. (5 pts)**

**list\_a = [[1, 2, 3], [5, 6, 7, 8], [10, 11, 12], [20, 21]]  
  
list\_b = [[2, 3, 4, 5], [6, 9, 10], [11, 12, 13, 14], [21, 24, 25]]**

**Graphical user interface, text, application, email

Description automatically generated**

**Submission:**

**Python Notebook Submitted -Question01\_Part01.ipynb**

**Text

Description automatically generated with medium confidence**

1. **Use Pool.map() to run the following python scripts in parallel; and print the result. (5 pts)**

**Script names: ‘script1.py’, ‘script2.py’, ‘script3.py’**

**Hint: you can put any content in the three scripts.**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Submission:**

**Following files were submitted as a part of this submission -**

* Three Scripts – Script1.py, Script2.py, Script3.py
* Python File - Question02\_Part01.py

**Output of Script 1 –**

**Text

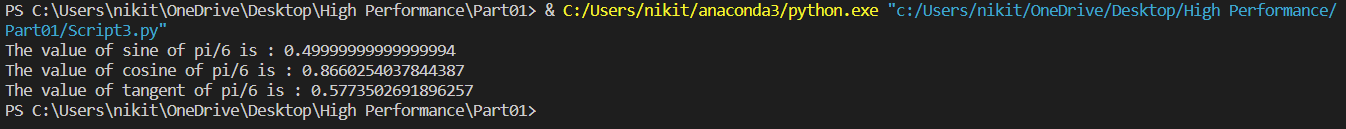
Description automatically generated**

**Output of Script 2-**

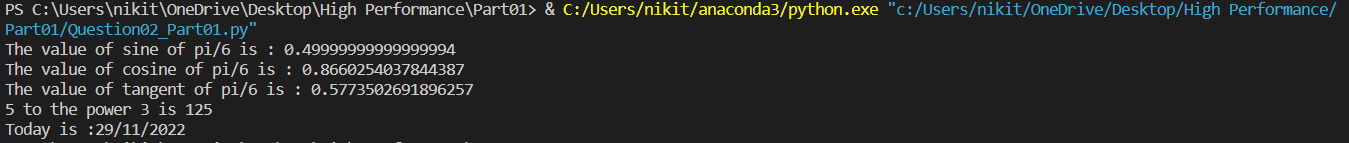
**Text

Description automatically generated**

**Output of Script 3-**

****

**Part 1 – Question 2: Output after Multiprocessing –**

****

1. **Normalize each row of 2d array (list) list\_c to vary between 0 and 1. Parallelize the function with any subfunction of Pool; and print the result. (5 pts)**

**list\_c = [[2, 3, 4, 5], [6, 9, 10, 12], [11, 12, 13, 14], [21, 24, 25, 26]]**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Submission:**

**Following file is submitted as a part of this submission -**

**Python File - Question03\_Part01.py**

**Part 2**

**Parallelize a panda DataFrame. Please complete the following tasks:**

* **Create a DataFrame. The required values are arbitrary numerical numbers, the shape of the values is 20000 x 100. (5 pts)**
* **Define a function to find the maximum and minimum values of each column, sum the squares of these two numbers, and then find the square root. (5 pts)**
* **Parallelize the function with any method of multiprocessing.Pool. (5 pts)**
* **Set a timer to calculate the elapsed time for the parallelized code when CPU=1, 2, 4, and 8.   (4 pts)**
* **Using matplotlib (or other tool) to plot the trend curve of speedup as the number of CPU (1, 2, 4, 8) and save the figure as an image file. (4 pts)**
* **Create a Word document. In this Word document, you should do:**
  + **Create a table and fill in the elapsed time (seconds or minutes) you obtained when running the parallelized program using different CPU numbers. (2 pts)**
  + **Insert the plot image in this Word document and analyze your results: the speedup, the overhead and optimal results, etc.                (5 pts)**

**Note: finish part 2 in one python file. Submit this program file and the Word file.**

**Submission: Following file is submitted as a part of this submission -**

**Python File - Question01\_Part02.py**

**Text

Description automatically generated**

|  |  |
| --- | --- |
| **No of CPUs used** | **Time Taken** |
| 1 | 1.216592788 |
| 2 | 1.218005657 |
| 4 | 1.322038888 |
| 8 | 1.741500139 |

**Chart, line chart

Description automatically generated**

**Analysis**:

* The above figure plots the trend curve of speedup as the number of CPUs = 1,2,4 and 8
* Clock speed is one of the significant methods of measuring the processor performance.
* As we can see from the graph, there is a dip in the processing time from CPU = 1 to CPU = 2, and the processing time increased as we increased the no of CPUs used. Here, for eg.: CPU = 4 and 8.
* From this, we can infer that as the no. of CPUs are increased, the Speedup time will increase to perform the task parallelly.
* With the help of Amdahl’s Law, through the medium of scientific computation, the theoretical speedup in a particular Task’s execution can be calculated, where S-overall is the max of how many times that parallel computing could be faster than single-thread performance

A picture containing diagram

Description automatically generated

* When increasing the number of processors to Spart, for parallel processing, the maximum speedup can be calculated to single processing.

**Part 3**

**Please complete the following tasks in JupyterLab or Jupyter Notebook:**

* **Program to use xgboost classifier to train a model with a given dataset “train.csv”**[**train.csv**](https://northeastern.instructure.com/courses/123482/files/18381466?wrap=1)[**Download train.csv**](https://northeastern.instructure.com/courses/123482/files/18381466/download?download_frd=1)**. (10 points)**
* **Set a timer and build a for-loop to calculate the elapsed time for the parallelized code when CPU=1, 2, 4, 8. (5 pts)**
* **Using matplotlib (or other tool) to plot the trend curve of speedup as the number of CPU increases (1, 2, 4, 8) in JupyterLab. (5 pts)**

**Chart

Description automatically generated with medium confidence**

**Following file is submitted as a part of this submission -**

**Python File - Question01\_Part03.ipynb**

**Part 4**

**Performance comparison between numpy array and dask array: set a timer to calculate the time of the following operations:**

* **In numpy array, draw random samples from a normal (Gaussian) distribution, the mean of the distribution is 10, the standard deviation of the distribution is 0.1, and the output shape is 20000x20000. Then take the "mean" of x along axis=0 with a step of 100.                                              (10 pts)**
* **In dask array, do the same thing as above. (Tips: you can set the size of “chunks” to 1000x1000) (10 pts) Note: finish part 4 in one Jupyter file.**

**Submission:**

**Following file is submitted as a part of this submission -**

**Python File - CSYE7105\_Assignment03\_Part04.ipynb**

**Output of Dask Dashboard -**

**Graphical user interface, application, Excel, bar chart

Description automatically generated**

**Part 5**

**Please install nycflight dataframe package in your Anaconda environment (you can do it on “local” or “discovery”):**

**pip install nycflights13  
  
from nycflights13 import flights**

* **Remove the samples with “NaN” in the feature “dep\_delay” in this dataframe. (5 pts)**
* **Start a Dask Client and set “n\_workers=4” for this client.                (5 pts)**
* **Using dask dataframe to compute the mean and standard deviation for departure delay “dep\_delay” of all flights.     (5 pts)**
* **You are required to take a screenshot of Dask Dashboard for running the step 3 with the 3 observed processes “Dask Process”, “Dask Graph”, “Dask Task Stream” in a same JupyterLab window, or use static Dask performance report. (Hint: since the dataset is small, the execution would be very fast. You may need to take a very quick screen capture.)     (10 pts)**

**Note: finish the part 5 in a Jupyter file. Submit this Jupyter file and the screenshot image in the step 4.**

**Submission:**

**Following file is submitted as a part of this submission -**

**CSYE7105\_Assignment03\_Question01\_Part05.ipynbGraphical user interface, text, application, chat or text message

Description automatically generated**

**While computing mean –**

1. **Dask Processing:**

Graphical user interface, application

Description automatically generated

1. **Task Streaming:**

**Graphical user interface, text, application, email

Description automatically generated**

**While computing Standard Deviation-**

1. **Dask Processing:**

Graphical user interface, application

Description automatically generated

1. **Task Streaming:**

**Graphical user interface, text, application, email

Description automatically generated**