**Assignment – 2**

Submission format: compact all files into one file and name it with your name for the convenience of TA

Submission through Canvas

Deadline is by the end of March 15th. (Consider changing to daylight saving time on March 14th, changing the deadline to March 15th)

**Part 1: 15 points**

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]]

**Ans: Code Attached. (Part1/question1.py)**

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.

**Ans: Code Attached. (Part1/question2.py)**

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]]

**Ans: Code Attached. (Part1/question3.py)**

**Part 2: 25 points**

A given panda dataframe here:

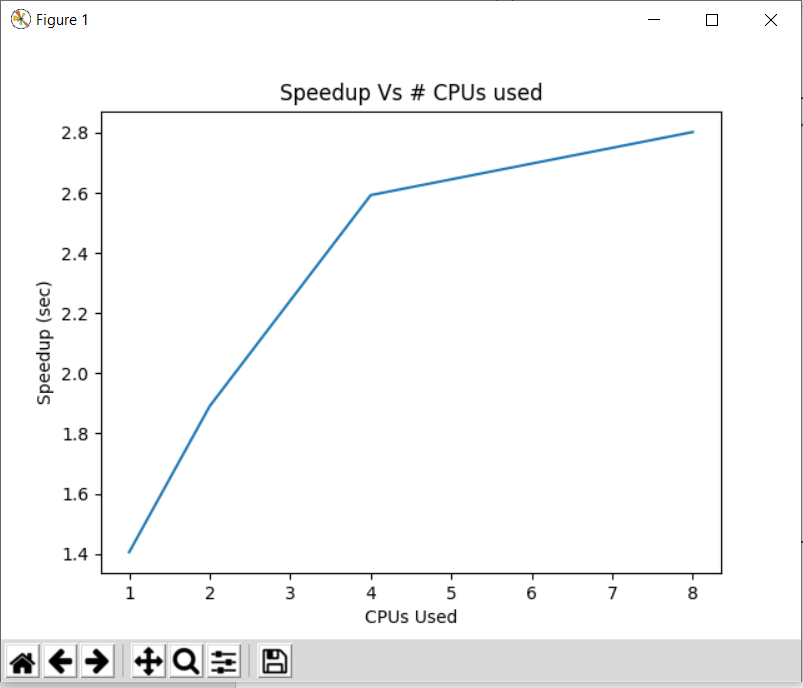
df = pd.DataFrame(np.random.randint(3, 10, size=[20000, 100]))

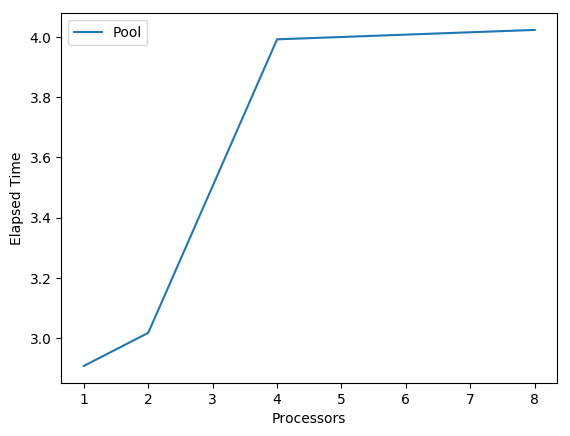
Please complete the following tasks:

* Define a function to sum the square of each element along a column.     (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.                                                        (5 pts)
* Plot the trend curve of speedup as the number of CPU (1, 2, 4, 8). (5 pts)
* Copy (or save) this figure from your code. Paste it in a Word document and analyze your results briefly in this Word file.         (5 pts)

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

**Ans: Code Attached. (Part2/question1.py and Part2/question2.py)**



****

**Analysis:**

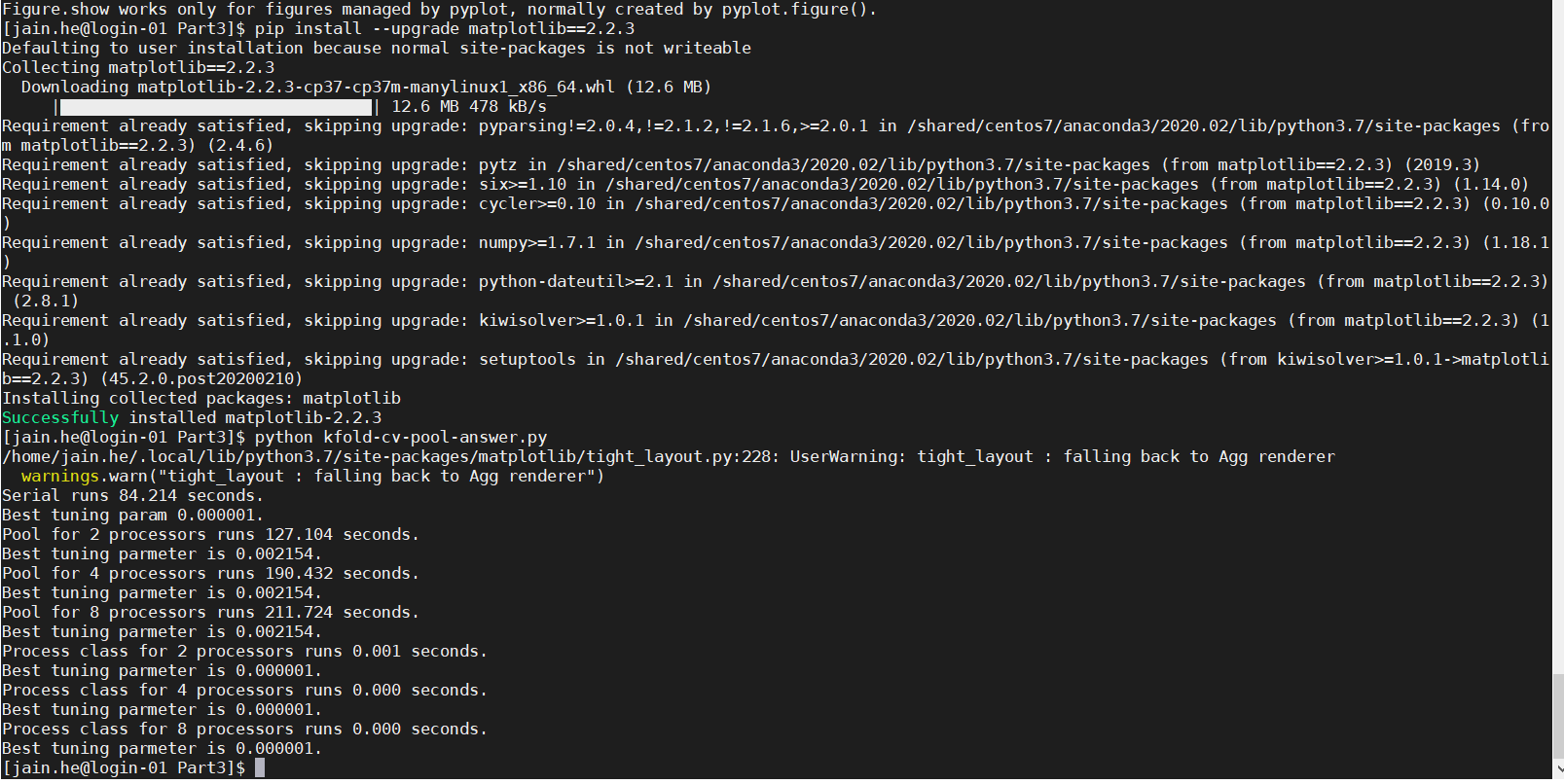
More cores will lead to high speed up of highly parallel applications, a powerful core will favor highly sequential applications. A system will adapt to the actual level of parallelism though, if feasible, changing the core configuration will incur some overhead and will challenge application developers. Even considering the re-configuration overhead, the speedup of automatic core configuration will be superior to either symmetric or asymmetric core design.

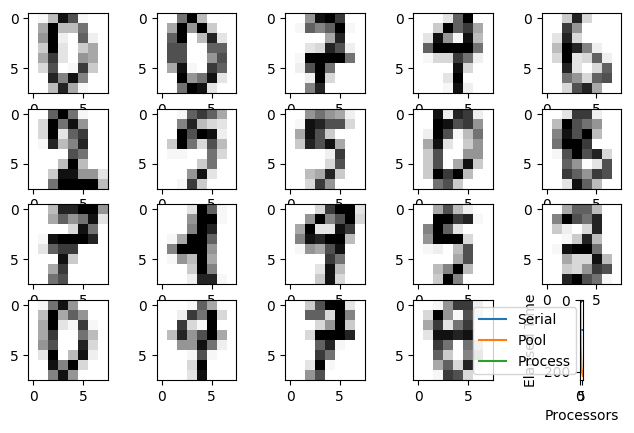
**Part 3: 15 points**

* Based on the parallelized code file “kfold-cv-pool.py”, set a timer and build a for-loop to calculate the elapsed time for the parallelized code when CPU=1, 2, 4, and 8.                          (5 pts)
* Plot the trend curve of speedup as the number of CPU (1, 2, 4, 8).        (5 pts)
* Copy (or save) this figure from your code. Paste it in a Word document and analyze your results briefly in this Word file.       (5 pts)

Note: finish the part 3 in one python file. Submit this python file and the Word file.

**Ans: Code Attached. (Part3/** **kfold-cv-pool-answer.py)**





**Trend curve of speedup as the number of CPU (1, 2, 4, 8)**.A picture containing diagram

Description automatically generated

**Analysis:**

More cores will lead to high speed up of highly parallel applications, a powerful core will favor highly sequential applications. A system will adapt to the actual level of parallelism though, if feasible, changing the core configuration will incur some overhead and will challenge application developers. Even considering the re-configuration overhead, the speedup of automatic core configuration will be superior to either symmetric or asymmetric core design.

**Part 4: 20 points**

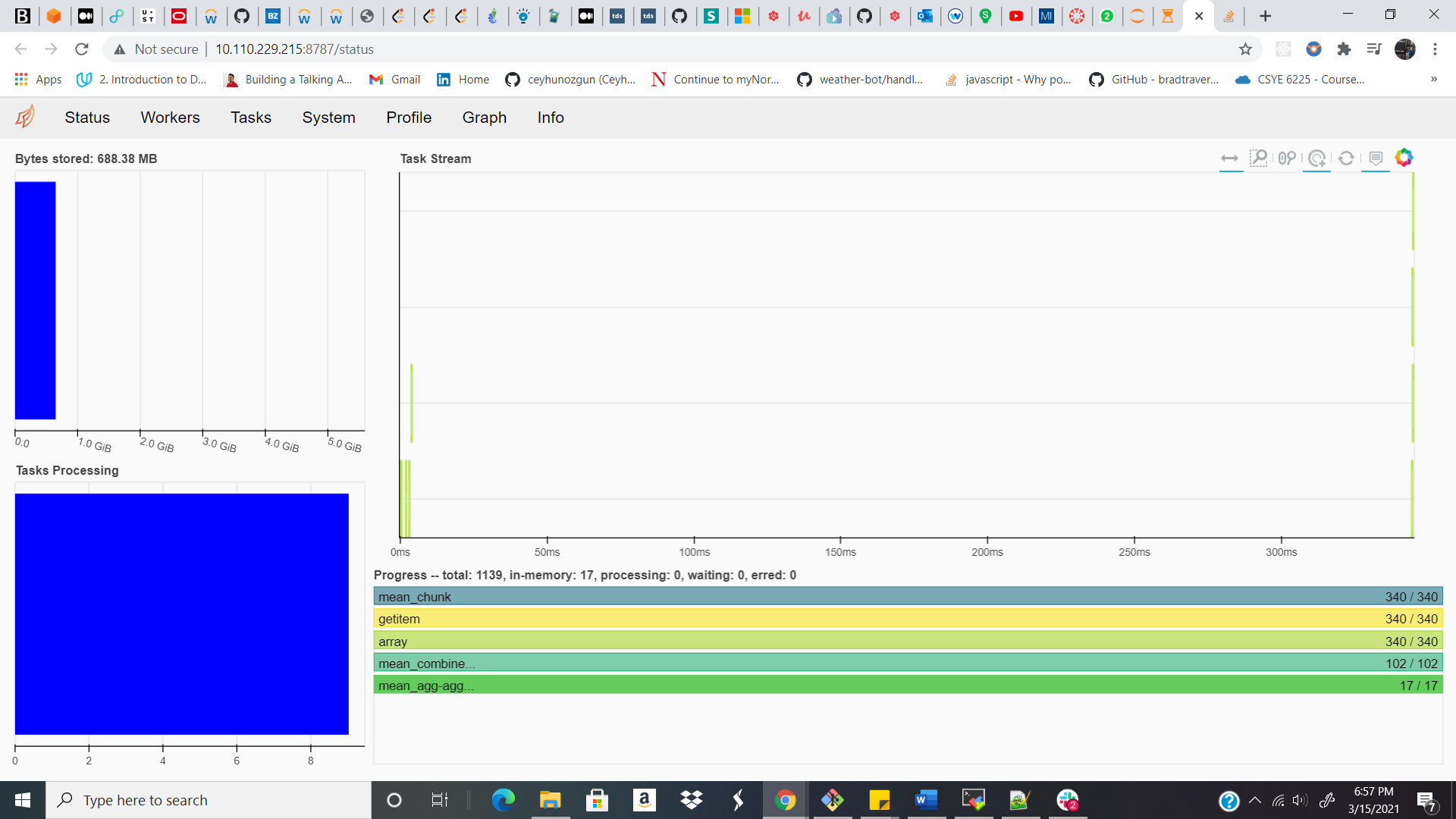
In numpy array, calculate the time of the following operation.

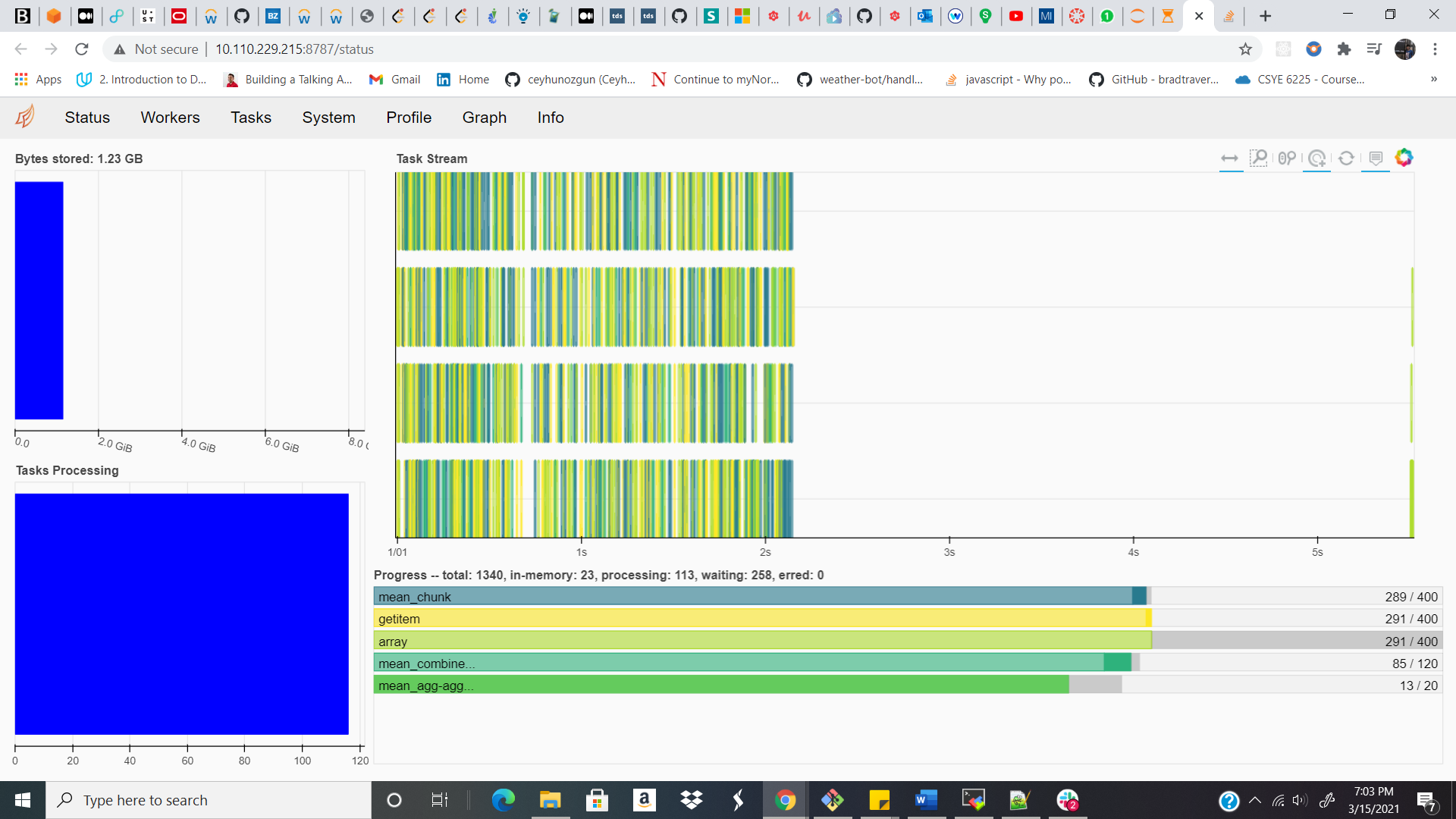
* Draw random samples from a normal (Gaussian) distribution, the mean of the distribution is 10, the standard deviation of the distribution is 0.1, and the output shape is 20000x20000. Then take the "mean" of x along axis=0 with a step of 100.                 (10 pts)
* In dask array, do the same thing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

Note: finish the part 4 in one Jupyter file.

**Ans: Code Attached. (Part4/** **part4.ipynb)**







**Part 5: 25 points**

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 within JupterLab extension 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 for running the step 3 with the 3 observed processes “Dask Process”, “Dask Graph”, “Dask Task Stream” in a same JupyterLab window, which I showed in the class. (Tip: since the dataset is small, the execution would be very fast. You may need to take a very fast 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.

**Ans: Code Attached. (Part5/** **part5.ipynb)**

