Assignment #2: HPC with Python

Total score: 100

This assignment can be completed **individually** or by a **team** of a maximum of three members.

Due date: See the course page

You are strongly encouraged to utilize a Large Language Model (LLM) (e.g., ChatGPT, GitHub CoPilot) to assist with this exercise. However, treat the LLM as a learning tool or assistant to help you learn the relevant subjects and techniques, so that you can tackle more complex problems in the future.

Setting up the application development environment

Install the Python interpreter and an IDE tool (such as VS code or JupyterLab), and add any necessary extensions of your choice to the IDE. Create a virtual environment "venv". Optionally, configure additional tools such as version control and source code management for team collaboration. Review the Python tutorial if needed and familiarize yourself with the "**pip**" command for installing the required Python packages. Ensure that all packages are installed within the virtual environment.

Goals of this assignment

The main goal of this assignment is to learn how to develop proficiency in creating Python programs for AI applications that utilize HPC resources.

1. Basic data processing [40]

(a) Complete the functions defined in "**data_process1.py**" for:

- Loading a dataset
  Declare a Python array named "GasProperties" and load the dataset "GasProperties.csv" (posted on the course page) into "GasProperties". The dataset consists of five columns: *temperature (T)*, *pressure (P)*, *thermal conductivity (TC)*, *sound velocity (SV), and quality index (idx)*.

- Data normalization

  Write a function named "normalize_array" that takes the data array (e.g., "GasProperties"), calculate Mean (μ), Max (x_{max}), Min (x_{min}), and standard deviation σ for each column except "idx" column, normalize each value (except for "idx"), using $x' = \frac{x-\mu}{x_{max}-x_{min}}$, where x' is the normalized value. Treat any value greater than 2σ from the mean as an outlier and exclude it from the normalized dataset. Save the normalized dataset into a file "GasProperties_norm.csv". The function returns the number of rows in the normalized dataset.
- Performance comparison

  Repeat the above steps using NumPy arrays instead of Python built-in array (lists). Compare and report the computation time for both to see if using NumPy is faster than Python array for this task.

(b) Complete the functions defined in "**data_process2.py**" that:

- Load normalized data

  Load "GasProperties_norm.csv" into a Panda DataFrame.
- Split the data

  Write a function "split_xy(df)" that splits the DataFrame into "X_data" (NumPy array containing columns T, P, TC, and SV) and "Y_data" (NumPy array containing only the "idx" column. Write another function "split_training_test(X_data,Y_data)" that splits both X_data and Y_data into training sets, "X_training" and "Y_training" (80% of data), testing sets, "X_testing" and "Y_testing" (20% of data).

(c) Re-organize the code from "data_process1.py" and "data_process2.py" into the following files:

- "data_loader.py" contains all function definitions from the original programs and written so that it can be imported as a module in other Python programs that need these functions.
- "data_tester.py" contains code to test the functions defined in "data_loader.py".

2. Basic vector and matrix operations [10]

Complete the functions defined in "**vector_product.py**" that:

(a) Dot product using a Python loop:

- In the GasProperties dataset, each column except for “idx” column is considered x column and “idx” column is considered y vector. Load one x column vector from X_data and one y column vector from Y_data.
- Perform element-wise multiplication between the x and y values using a Python “for” loop and sum all the products and return the final scalar result.
- Repeat the same task for each column in X_data with y and identify the column that returns the largest dot product value. Measure the total computation time for this loop-based approach.

Note: when the arrays are vectors, this element-wise multiplication and summation is called the “dot product”, producing a scalar value that can indicate similarity between the two vectors.

(b) Dot product using NumPy

- Repeat the above steps without any loops, using NumPy dot product, np.dot(a,b) where a and b are NumPy arrays (treated as vectors).
- Compare the performance between “Loop-based dot product” and “NumPy dot product”. If there is any difference in computation times, briefly explain the reason.

(c) Matrix multiplication

- Create a matrix from X_data containing all features (x) columns. Perform matrix multiplication (X_data x X_data) twice: once using 32-bit precision and once using 64-bit precision. Compare the computation times for both. If there is any difference in computation times, briefly explain the reason.

(d) GPU acceleration

- Use **at least one GPU** to perform the matrix multiplication (X_data x X_data) in PyTorch using 64-bit precision from the previous step, and compare the performance with and without GPU acceleration.
- List the key hardware specification that influence computation performance such as CPU clock speed, # of CPU cores, system RAM size, GPU model and clock speed, # of cores in GPU, GPU memory size and bandwidth.
- Using the above specifications, estimate the theoretical computing performance of the machine used in FLOPS (floating-point operations per second).
- Compare theoretical and actual computation times for above dot product and matrix multiplication tasks and explain possible reasons for any differences.

What to submit

- A report file in Word or PDF format that includes the name or names of the team members, and the % contribution of each member. If there is no agreement on individual contributions, briefly describe the tasks assigned and completed by each member. Different scores may be awarded based on individual contributions. If all members contributed equally, simply state "equal contribution." The **report** should also **include** descriptions of the answers (if questions are asked) and a portion of the screenshot demonstrating the functionality of each program.
- **Python program files. Each program file individually**. DO NOT submit any zip file as **Canvas cannot open them**.
- Only one report file and the program file(s) for the entire team.

Grading criteria

- Does each program successfully perform the required tasks and generate the correct results?
- Does the report demonstrate that the team has a clear understanding of the relevant concepts, programming techniques, functional requirements? Does it show how the team applied this knowledge to successfully complete the tasks?
- Is the effort put into the project clearly reflected in both the report and the program files?