\*\*Project Objective: \*\* The objective of this project is to compare the performance and demonstrate the computational differences between CPU and GPU. It aims to determine whether parallel operations are better performed on the CPU with respect to its number of cores or on the GPU.

To achieve this objective, two computational models were defined for CPU and GPU calculations:

- GPUMatrixOperations.py: For matrix computations on the GPU.

- CPUMatrixOperations.py: For matrix computations on the CPU.

Subsequently, separate modules were implemented in cpu\_operations.py and gpu\_operations.py, following the algorithm described below:

1. Calculating the average execution time of a matrix with a consistent size, for example, 10 times.

- Function: calculate\_average\_execution\_time.

2. Providing a list of different matrix sizes to the calculate\_and\_save\_averaged\_results function, which utilizes the calculate\_average\_execution\_time function to calculate the average and save the results in a JSON file. The output of these functions is stored in JSON format for the purpose of generating comparative reports.

By executing the separate modules, cpu\_operations.py and gpu\_operations.py, in a CPU and GPU environment (e.g., Google Colab), the results are obtained, line charts are plotted, and another function, plot\_average\_from\_json.py, is written to calculate the average of two JSON files, averaged\_execution\_gpu\_times.json and averaged\_execution\_cpu\_times.json. The line charts provide a more accurate comparison.

By comparing these results, it is observed that matrix computations are faster on the GPU. This is due to the larger number of threads available for parallel computations on the GPU. Since matrix computations require no specific processing and involve a series of small calculations, they can be broken down and performed on different threads, with the final result obtained by combining these thread calculations.

Hence, it is concluded that matrix computations utilizing thread parallelization on the available CPU or GPU cores are faster. In the example mentioned above, since the GPU has a larger number of threads at the user level, the computation is performed faster on the GPU.

Important Note: The CPUMatrixOperations.py module, used for task creation in the mentioned modules, utilizes the numpy module, which can create a matrix with random values and perform matrix multiplication using the dot function in a parallel computing manner.

The cpu\_operations.py module utilizes various functions to perform matrix computations on the CPU and save the results. These functions work as follows:

1. `save\_to\_json`: This function is used to save data in a JSON file. It saves the input data (data) in JSON format in a file with the specified filename.

2. `calculate\_average\_execution\_time`: This function calculates the average execution time of a matrix operation on the CPU for a specific matrix size and a specified number of executions. For each execution, it calls the `matrix\_multiplication` function from the CPUMatrixOperations module to measure the execution time. It then calculates the average execution time and returns it.

3. `calculate\_and\_save\_averaged\_results`: This function is used to calculate the average execution time for different matrix sizes and save the results in a JSON file. It first calculates the average execution time for each matrix size using the `calculate\_average\_execution\_time` function. Then, it saves the results in a dictionary in JSON format and stores it in the file "averaged\_execution\_cpu\_times. json".

Overall, the cpu\_operations.py module is used to calculate the average execution time of matrix operations on the CPU for different matrix sizes and save the results in a JSON file.

For further investigation, a second scenario for testing image processing on the CPU and GPU is also considered. The goal of this test is to determine whether heavy computational tasks such as image processing perform better on the GPU or CPU. The results of this analysis are stored in a JSON file, img-proc-cpu.json, and a line chart, img-proc-cpu.png, is plotted.

To further explore the impact of parallelization on improving computation performance, another module named matrix\_multiplier.py is defined in the performance\_report directory. This module takes the matrix size as input and implements two functions: one in a parallel manner and the other in a serial manner. They calculate a matrix with randomly generated numbers based on the given size. The output of the calculation, which is a single number, is returned. Then, in the main project directory, a module named performance\_report\_run.py is defined to measure the execution time of these two functions, save it in a JSON file, and display the difference using a bar chart. It clearly demonstrates the significant effect of parallelization on the speed of operations.

Based on these findings, it is recommended to use parallelization for computations or tasks that involve numerous small calculations, such as matrix computations. It is evident that GPUs provide a larger number of threads for parallel operations, resulting in improved performance.