GIK INSTITUTE
OF ENGINEERING SCIENCES AND TECHNOLOGY

Student Name: Abdullah Ejaz Janjua

Reg: 2023038

Assignment No: 01

Instructor: Dr. Taj Khan

GitHub Repo: Link to Repository

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Ghulam Ishaq Institue of Engineering Sciences and Technology, Topi, Swabi

Task 01: Environment Setup

Relevant file: task1.md

I have set up a GPU coding environment using Google Colab with NVIDIA CUDA.

Environment Details

- **GPU Instance:** Tesla T4 (via Google Colab free tier)
- **CUDA Version:** 12.4 (driver), 12.5 (toolkit)
- **Compiler:** `nvcc` (NVIDIA CUDA Compiler)
- **Debugger:** `cuda-gdb` (Available in runtime)

Workflow

My development pipeline involves writing code locally, syncing to Google Drive via `rclone`, and compiling on Colab using:

```
!nvcc -arch=sm_75 <gpu-kernel>.cu <main>.cpp -o <exec>
```

The `-arch=sm_75` flag ensures compatibility with the T4 GPU.

Task 02 & Task 03: Matrix Addition

Relevant files: Task 2 Code, Task 3 Code

File Structure Explanation

The input file structure is defined as follows:

- **Header:** Two integers defining `rows` and `cols`.
- **Matrix A:** Sequence of $rows \times cols$ elements (row-major).
- **Matrix B:** Sequence of $rows \times cols$ elements (row-major).

Example Input Format:

```
2          # Number of Rows
3          # Number of Columns

1 2 3      # Matrix A (Row 1)
4 5 6      # Matrix A (Row 2)

1 2 3      # Matrix B (Row 1)
4 5 6      # Matrix B (Row 2)
```

Task 04: Performance Analysis

Relevant file: task4

Methodology

I measured execution time for matrix sizes ranging from 100×100 to 2000×2000.

- **CPU Time:** Measured using `std::chrono`.
- **GPU Time:** Measured using `cudaEvent_t`. The duration includes memory allocation, Host-to-Device transfer, Kernel computation, and Device-to-Host transfer.

Results

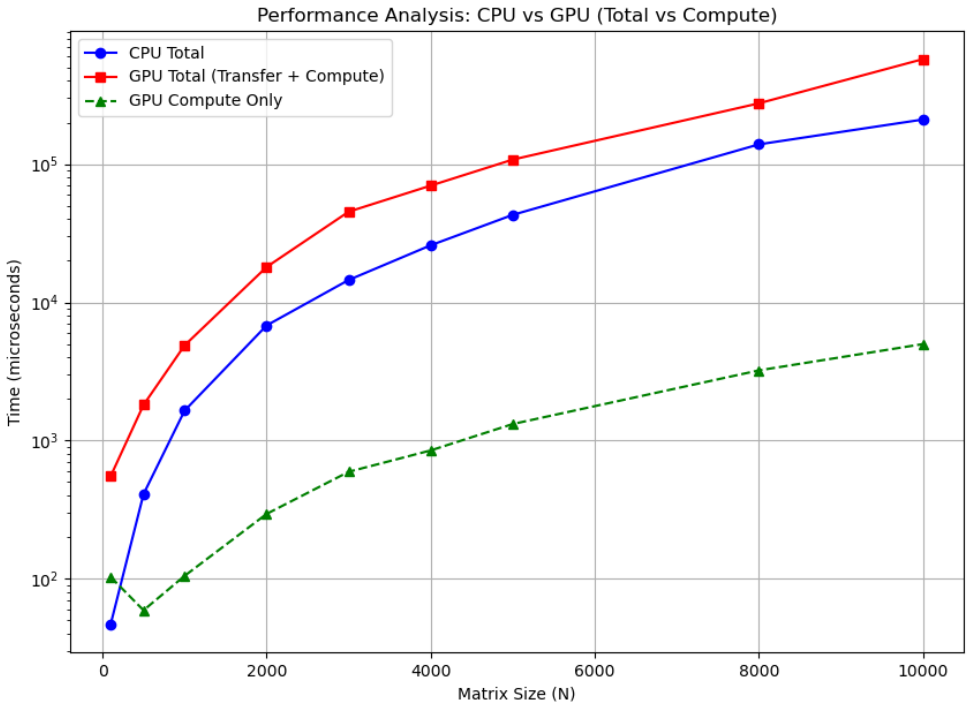

Figure 1: Execution Time vs. Matrix Size

Commentary

The results show that the GPU implementation is slower than the CPU for all tested matrix sizes (up to N=10,000). This is expected for element-wise matrix addition because the operation has low arithmetic intensity. The execution time is dominated by memory transfer latency rather than computation. Because the entire input matrices must be transferred to the device memory before the kernel can launch, the GPU's compute resources remain idle for the majority of the total duration.

The graph (log scale) reveals that the GPU Kernel (Green line) is orders of magnitude faster than the CPU (Blue line). However, the GPU Total Time (Red line) is slower than the CPU for this range. This proves that the data transfer is the bottleneck. The GPU computes the $10,000 \times 10,000$ addition in just 5ms, but it takes $\sim$570ms to move the data to and from the device.