Name-Ketki Khati

NU ID-002436886

Assignment 2: Naive CUDA GEMM using Modal

Objective

The goal of this assignment is to implement and evaluate a **naive matrix multiplication (GEMM)** CUDA kernel and execute it on a GPU environment using **Modal**, since a local NVIDIA GPU was not available. The implementation computes:

$$C = A \times B$$

where:

- $A \in \mathbb{R}^{M \times K}$
- $B \in \mathbb{R}^{K \times N}$
- $C \in \mathbb{R}^{M \times N}$

System & Environment

Local Machine

- OS: Windows
- GPU: Intel® Arc™ 130V (non-CUDA, not supported by NVIDIA CUDA)
- CUDA execution locally was **not possible**

Remote GPU via Modal

- Platform: **Modal**
- CUDA Version: **12.2**
- GPUs used:
  - NVIDIA **A10**
  - NVIDIA **L4**
- Compiler: nvcc
- Execution: Remote GPU workers provisioned automatically by Modal

---

Implementation Details

CUDA Kernel

A naive GEMM CUDA kernel was implemented where:

- Each thread computes one element of matrix C
- No shared memory optimizations were used
- Kernel configuration:
    - 2D grid
    - 2D thread blocks

Correctness Verification

- GPU result is compared with a CPU reference implementation
- Maximum absolute error is reported

Execution via Modal

Because no CUDA-capable NVIDIA GPU was available locally, the code was executed remotely using **Modal**.

Workflow

1. CUDA source file gemm_naive.cu was mounted to the Modal container
2. CUDA container image (nvidia/cuda:12.2.0-devel-ubuntu22.04) was used
3. Compilation performed inside Modal using nvcc
4. Execution performed on Modal GPU workers
5. Performance and correctness metrics recorded

Command used:

```
python -m modal run run_gemm_modal.py
```

Experimental Results

Matrix Size

- $M = 512$
- $N = 512$
- $K = 512$

Run 1

- **GPU:** NVIDIA A10
- **Kernel Time:** 0.157696 ms
- **Throughput:** 1702.23 GFLOP/s
- **Max Absolute Error:** 9.53674×10^{-6}

Run 2

- **GPU:** NVIDIA L4
- **Kernel Time:** 0.164864 ms
- **Throughput:** 1628.22 GFLOP/s
- **Max Absolute Error:** 9.53674×10^{-6}

Discussion

- Both GPU executions produce **numerically correct results**, with very small floating-point error consistent with FP32 arithmetic.
- Performance differs slightly due to:
  - Different GPU architectures (A10 vs L4)
  - Different clock speeds and memory characteristics
- Despite being a **naive kernel**, the achieved throughput exceeds **1.6 TFLOP/s**, demonstrating the effectiveness of GPU parallelism.
- No shared memory or tiling optimizations were used; therefore, significant performance improvements are possible with optimized kernels.

Challenges Faced

- CUDA could not be executed locally due to absence of an NVIDIA GPU
- Required setting up Modal authentication and remote execution
- File mounting and container compilation issues were resolved during development

---

Conclusion

This assignment successfully demonstrates:

- Implementation of a naive CUDA GEMM kernel
- Remote GPU execution using Modal
- Performance benchmarking on modern NVIDIA GPUs
- Validation of correctness against CPU results

The use of Modal enabled seamless access to CUDA-capable GPUs and allowed successful completion of the assignment without local NVIDIA hardware.