Week 3 Assignment

GPU Memory Hierarchy & Performance Optimization

GPU Programming using CUDA and Triton (WiDS'25)

Abhineet Agarwal
Dept. of Electrical Engineering, IIT Bombay

Overview

This assignment focuses on the most important aspect of GPU programming: **memory behavior**. We will experiment with memory access patterns, use shared memory to optimize performance, and begin profiling GPU kernels.

Submission Instructions

Submit a single PDF summarizing your results and analysis, along with source files. Place your submission in the following folder:

```
week3/
  assignment.pdf
  coalesced.cu
  non_coalesced.cu
  shared_memory.cu
  cpu_baseline.py
```

1 Task 1: Global Memory Access Patterns

1.1 Coalesced Access Kernel

Implement a CUDA kernel where consecutive threads access consecutive memory locations. For example, a vector operation where thread i accesses element i.

Requirements:

- Use a 1D grid and block layout
- Implement bounds checking
- Measure execution time using CUDA events

1.2 Non-Coalesced Access Kernel

Implement a logically equivalent kernel where memory access is *strided*. For example, thread i accesses element $i \times k$, where $k > 1$.

Requirements:

- Keep computation identical to the coalesced version
- Only change memory access patterns
- Measure execution time using the same methodology

1.3 Analysis

In your PDF, answer the following:

- Compare execution times of the two kernels
- Explain the difference using warp-level memory access behavior
- Relate your results to the concept of memory coalescing

2 Task 2: Shared Memory Optimization

In this task, we will reduce global memory traffic using shared memory.

2.1 Baseline Kernel (Global Memory)

Implement a kernel that performs a simple computation (e.g., stencil update, sliding window sum, or block-wise reduction) using only global memory.

2.2 Optimized Kernel (Shared Memory)

Reimplement the same computation using shared memory.

Requirements:

- Load data into shared memory
- Use `__syncthreads()` correctly
- Avoid out-of-bounds access

2.3 Performance Comparison

Measure and compare:

- Kernel execution time
- Speedup factor

Explain:

- Why shared memory improves performance
- Whether bank conflicts may still exist

3 Task 3: CPU Baseline Submission (Deadline)

This is the **final deadline** to submit your CPU baseline for the workload identified in Week 1.

Provide:

- `cpu_baseline.py`
- Input size(s) used
- Measured runtime(s)

This baseline will be used to evaluate GPU acceleration in Weeks 4–6.

Analysis Guidelines

Your PDF should include:

- Clear plots or tables comparing runtimes
- Short but precise explanations
- Correct technical terminology (warp, coalescing, latency, bandwidth)

Notes

- You may use Google Colab or a local GPU
- You may reference NVIDIA documentation and samples
- All submitted code must be your own

End of Week 3 Assignment