Week 1 Assignment

GPU Intuition & Compute Foundations

GPU Programming using CUDA and Triton (WiDS'25)

Abhineet Agarwal
Dept. of Electrical Engineering, IIT Bombay

Overview

This assignment is designed to help you understand how GPU architecture differs from CPU architecture, and to analyze a computation-heavy workload from a technical perspective. You will not write CUDA kernels this week – instead, you will build the conceptual and analytical foundation required for effective GPU programming.

Submission Instructions

Submit a single PDF containing your answers, and include any diagrams as embedded images or separate attachments. Place the completed assignment in the repository folder:

```
week1/
  assignment1-solution-<your-name>.pdf
  cpu_baseline-<name>-<task>.py
```

1 Task 1: Identify and Analyze a GPU-Accelerable Workload

Choose a computation-heavy workload from your own research, engineering project, or area of interest. Example categories include:

- Simulation loops (finite difference methods, Monte Carlo, N-body, particle updates)
- Numerical kernels (matrix multiplication, reductions, convolutions)
- Machine learning operations (custom layers, loss functions, attention mechanisms)
- Data processing pipelines (pairwise distances, filtering, transforms)

Write a detailed technical analysis (1–2 pages) addressing the following:

1.1 Operation Breakdown

- What does the computation do? Describe it precisely.
- Provide pseudocode or a mathematical formulation.
- Specify input sizes, tensor dimensions, or loop bounds.
- Identify independent work units (opportunities for parallelism).

1.2 Compute vs Memory Analysis

Using concepts from Week 1 materials:

- Is the operation compute-bound or memory-bound?
- Estimate its approximate arithmetic intensity (qualitative is acceptable).
- Are there reuse opportunities suitable for shared memory?
- Identify dependencies or sequential steps that may limit parallelism.

1.3 Expected Behavior on a GPU

- How would CUDA threads map onto the iteration space?
- Would the workload scale well to thousands of threads?
- What challenges might arise (warp divergence, irregular access, small batch sizes)?

1.4 CPU Baseline (Can be submitted in Week 1/2/3)

Provide the runtime of a CPU implementation using Python's `time.perf_counter()` or similar timing tools. This baseline will be used later when implementing the GPU version.

2 Task 2: CUDA Execution Model Mapping Diagram

Create a clear diagram illustrating the CUDA execution hierarchy:

`Grid` → `Blocks` → `Warps` → `Threads`

Annotate your diagram with:

- How your chosen workload's iteration space maps to grid, blocks, and threads
- Where synchronization might be required
- Potential memory bottlenecks
- Where shared memory might be inserted for optimization

You may draw this diagram by hand (scan/photo) or create it digitally.