Derivation of CUDA Kernel Execution-Time Prediction

This document explains the full reasoning process used to derive the CUDA kernel performance prediction template. The model follows the principles of GPU performance engineering, including FLOP analysis, memory bandwidth limits, and kernel launch latency behavior observed on NVIDIA GPUs.

1. Overview of Performance Model
Every CUDA kernel runtime can be approximated using three components:
• Work performed by threads (iterations × FLOPs)
• Memory traffic (bytes loaded/stored)
• GPU hardware limits (peak FLOPs, memory bandwidth, launch latency)

This mirrors the NVIDIA Roofline Model and CUDA optimization best practices.

2. Workload Derivation from the Kernel
Given a loop of the form:
for (offset = threadIdx.x; offset < K; offset += blockDim.x) { float t = a - b; temp += t * t; } We derive:
• iters_per_thread = ceil(K / blockDim.x)
• Each iteration performs ~3 FLOPs (subtract, multiply, add)
• Each iteration loads 8 bytes (two float loads)

Threads are organized as:
threads_per_block = Bx × By × Bz
active_blocks = Gx × Gy × Gz (adjusted by boundary checks)

Total iterations:
N_iters_total = iters_per_thread × threads_per_block × active_blocks

3. FLOP Analysis
Total FLOPs = N_iters_total × FLOPs_per_iteration
This follows GPU throughput-based modeling. FLOP counts determine the compute-bound time when compared to GPU peak performance.

Compute-bound time:
t_compute = FLOPs_total / Peak_FP32

4. Memory-Traffic Analysis
Bytes per iteration = (#float_loads × 4) + (#float_stores × 4)
Total memory bytes = N_iters_total × bytes_per_iter

Memory-bound time:
t_mem = bytes_total / memory_bandwidth

Whichever is larger (t_compute or t_mem) determines the kernel body time:
t_body = max(t_compute, t_mem)

5. Kernel Launch Overhead
NVIDIA GPUs exhibit a typical kernel launch latency of ~4–6 microseconds (Turing, Ampere, Ada; verified in NVIDIA Developer Forums).

This overhead dominates for small kernels.

Final predicted runtime:

$t_{total} \approx t_{body} + t_{launch}$

6. Why This Model Is Correct
• Matches NVIDIA Roofline methodology
• Matches Nsight Compute performance formulas
• Matches CUDA programming and memory-throughput theory
• Matches empirical kernel launch latency observations

7. Template Summary
1. Compute iterations
2. Compute FLOPs
3. Compute bytes
4. Compute compute-bound and memory-bound times
5. Add launch overhead

This derivation forms the basis of the prediction template and the GPU-specific runtime predictions generated earlier.