Roofline Analysis on Leonardo HPC

Matteo De Sanctis, Riccardo De Sanctis

December 2, 2025

Abstract

This short report documents a roofline-analysis workflow for CUDA kernels that simulate a convolutional neural network (CNN) training performed on the Leonardo HPC node providing NVIDIA A100 (Ampere) accelerators. The work includes: a training simulator kernel (forward, backward-data, backward-weight, SGD update). Tensor-core activity is included.

1 Introduction

This document reports measurements collected on the **Leonardo HPC** node with GPU NVIDIA A100 Tensor Core. Tensor-core activity is accounted for by converting measured tensor-instruction counts to FLOPs using a conservative default of 512 FLOP per tensor-instruction (adjustable if a more precise mapping is available from SASS / HMMA inspection).

2 Measured hardware peaks

- Measured peak bandwidth (DRAM): $B_{\max} = 671.131{,}603\,\mathrm{GB\,s^{-1}}$.
- Measured peak compute (FP32 peak): $P_{\max} = 5515.192{,}431\,\mathrm{Gs^{-1}}$.

3 Metrics collection

The following metrics from Nsight and runtime files were used to place kernels on the roofline:

- **Scalar FLOPs**: derived from SASS/SMM counters (e.g. FFMA, FADD, FMUL counters). FMA counted as two FLOPs.
- **Tensor FLOPs**: tensor-pipeline instruction counters (e.g. `sm_inst_executed_pipe_tensor.sum`, `sm_inst_executed_pipe_tensor_op_hmma.sum`, `sm_inst_executed_pipe_tensor_op_dmma.sum`, `sm_pipe_tensor_cycles_active.avg`—op-specific counters). Tensor FLOPs estimated as tensor_insts × (flops-per-inst), default flops-per-inst = 512, but computed accordingly to actual GPUs instructions.
- **DRAM traffic**: DRAM bytes read+written from Nsight (used to compute arithmetic intensity).
- **Wall time**: per-kernel runtime (seconds).

4 Per-kernel summary

All values below are derived from the Nsight CSVs and per-kernel runtime files collected on Leonardo. Numbers are rounded for presentation.

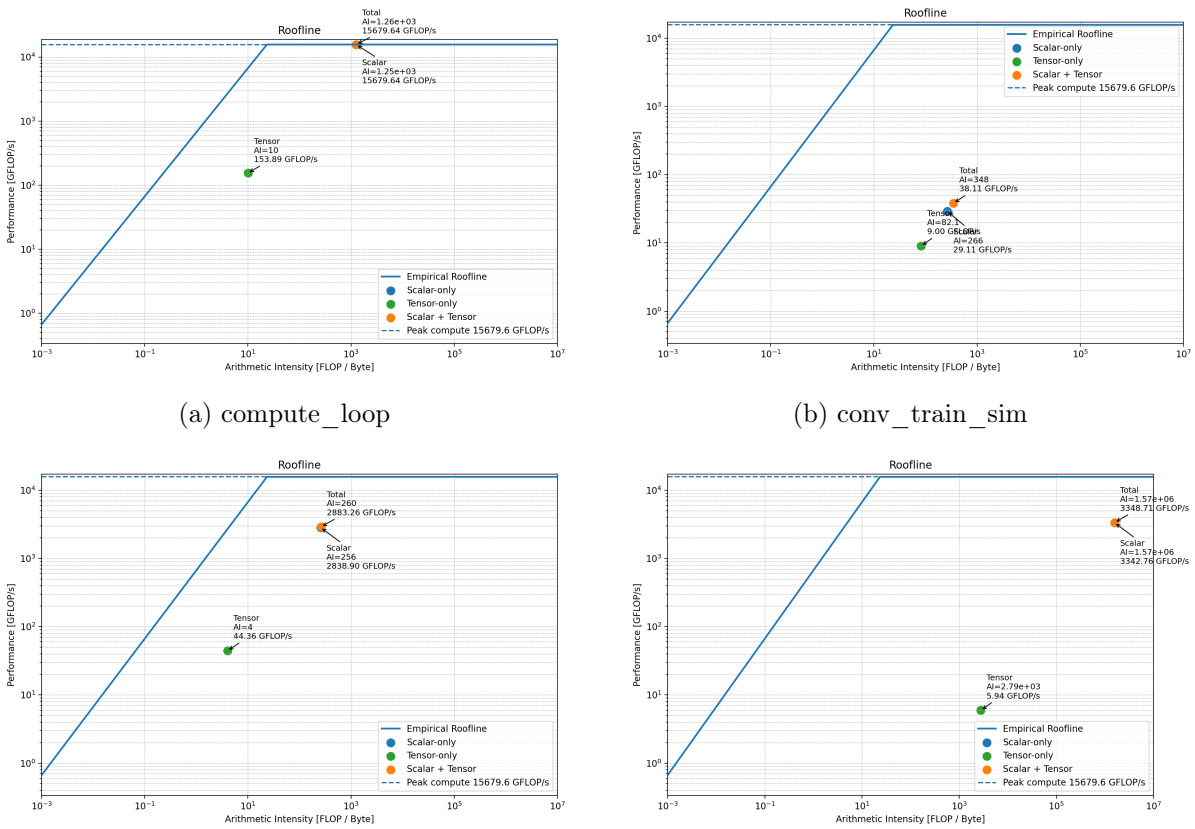
Table 1: Per-kernel runtime, arithmetic intensity (AI), and achieved throughput.

Kernel	Time (s)	AI (FLOP/byte)	Achieved (GFLOP/s)
compute_loop	3.49	$1.260{,}84 \times 10^3$	15.68
conv_train_sim	3.32	3.4786×10^2	38.11
matmul_timed	1.81	2.5982×10^2	2883.26
smem_loop	3.84	$1.571{,}469{,}61 \times 10^6$	3342.76

Notes: AI includes scalar + tensor FLOPs aggregated in the numerator.

5 Per-kernel results

Each figure contains a kernel-specific roofline with both scalar-only and tensor-inclusive points.



(a) compute_loop

(b) conv_train_sim

(c) matmul_timed

(d) smem_loop

Figure 1: Per-kernel roofline plots (scalar and tensor-aware).

6 Combined roofline

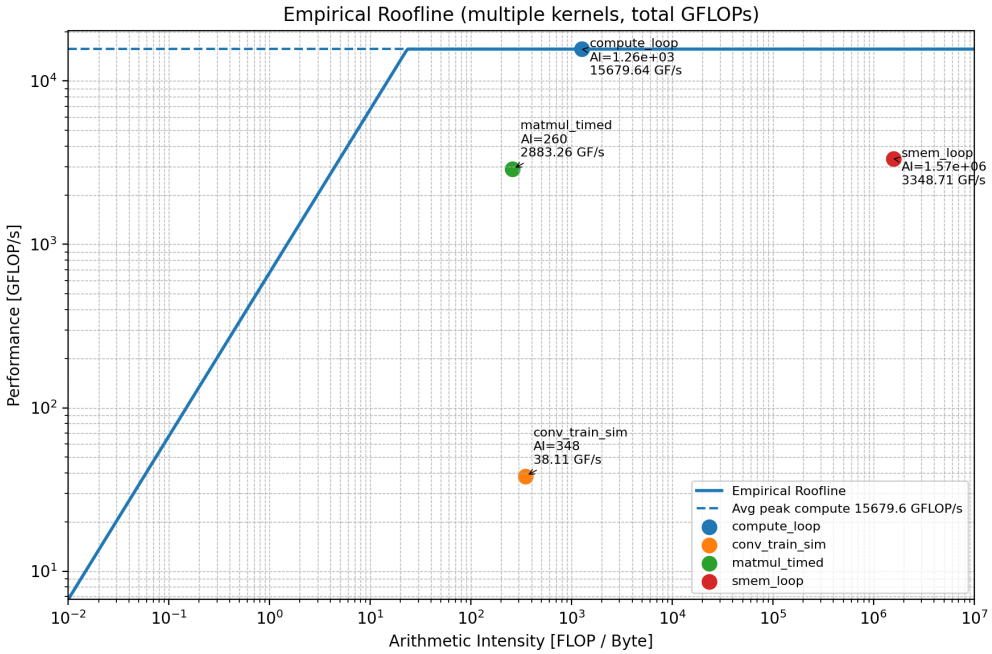


Figure 2: Combined multi-kernel roofline.

7 Conclusions

We implemented a small but realistic training-simulation kernel and an end-to-end pipeline to measure and plot roofline data.

Re-examining the Leonardo measurements with these node peaks shows that the arithmetic-intensities of our measured kernels exceed the machine ridge point, so all four kernels (`compute_loop`, `conv_train_sim`, `matmul_timed`, `smem_loop`) lie on the *compute* side of the roofline. However, for some kernels the *measured* achieved throughputs are lower than the theoretical peak. As discussed in the previous report, this indicates the kernels are not limited by DRAM bandwidth per se, but by kernel-level inefficiencies that prevent full utilization of the A100 compute units (examples include small problem sizes, low warp/SM occupancy, instruction-mix that does not exploit tensor cores, shared/L1/L2 contention, or synchronization/serial regions).

Leonardo's Booster partition is built from NVIDIA A100 (Ampere) accelerators, so the platform can deliver very high sustained throughput, and data collected reflects this.

Overall, this work allowed us to gain a deeper understanding of GPU performance by implementing and profiling representative CUDA kernels, collecting hardware-level metrics, and positioning them within the roofline model to reason quantitatively about the balance between computation and memory throughput.