Profiling CUDA Benchmarks for Performance Analysis on Modern GPUs

Matyas Krizek, Alondra Lona, and Rohit Basu
Dr. Shubbhi Taneja

KORET FOUNDATION

SONOMA STATE UNIVERSITY

Objectives

- To investigate the efficiency of two GPUs utilizing benchmarks to stress the GPUs and collect metrics to give us insight into the activities of the hardware.
- To compare the performance of the two systems when running the same parameters and how to improve the overall performance.

Motivations

- Increase in demands for accelerator-based systems like Graphics Processing Units (GPU) for computations by scientific community
- GPUs being power-hungry devices and the need to optimize code executed on them

Goals

- To develop an understanding of effective use of hardware by collecting the profiling data available through six performance counters, laying the framework for future analysis under power and energy constraints.

Research Question/Hypotheses

- What are the performance bottlenecks of a Volta VS. Turing GPU when increasing the thread and block size on an application?
- Does the increase in available resources always create a speedup on a GPU?

Methodology

- Collected six performance metrics available on two modern GPUs to analyze the profiling data
- Utilized four benchmarks **(1)** Jacobi, **(2)** Gaussian ELimination, **(3)** LU Decomposition & **(4)** Streamcluster -- these four are representatives of popular application domains such as data mining, steaming applications, and dense linear algebra.

Hardware / Chosen Applications

- Two GPU architectures used:: Volta & Turing
- Data collection tool: NVIDIA Nsight Compute

Criteria	Lassen	Monolith
GPU Architecture	Volta	Turing
Nvidia GPU Model	Tesla V100 SXM2 16 GB	RTX 2080 Ti
Compute Capability	7.0	7.5
Number of SMs	84	68
Memory/SM	96 KB	64 KB
Memory Bandwidth	900 GB/sec	616 GB/sec
L2 Cache Size	6144 KB	5632 KB
Cuda Version	10.1.243	10.0.130
Nsight Version	2020.1.0	2020.1.2

- Chose six metrics from Nsight that tell us about important characteristics of the GPU during runtime.

Dram_bytes.sum.per_second

L1tex__t_bytes.sum.per_second

Lts__t_bytes.sum.per_second

Sm_inst_executed.sum.per_second

Sm_warps_active.avg.pct_of_peak_sustained_active

Smsp_cycles_active.avg.pct_of_peak_sustained_active

- These metrics measure the amount of Global memory access per second, the amount of L1 texture cache access per second, the number of L2 cache access/sec, # instructions executed per second, % of the maximum active warps, and % of active cycles where those warps have work to do, respectively.

Results & Observations

Figure 1. **Volta**

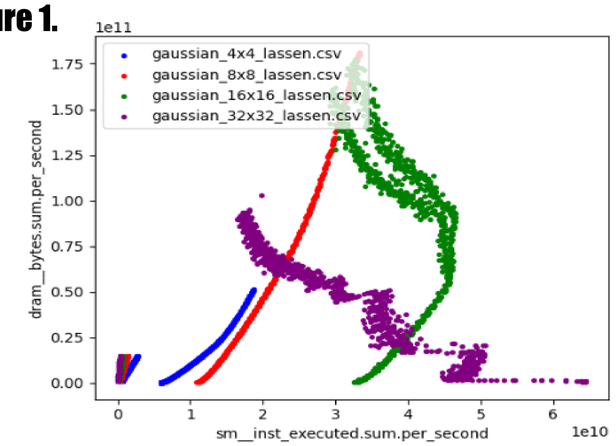
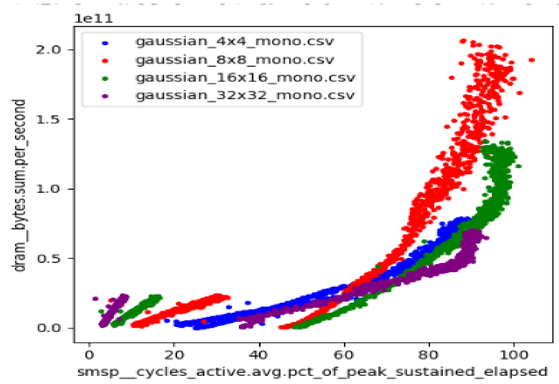


Figure 2. **Turing**



The figures shows that the relationship between performance and adding more hardware resources is not always linear.

- Using the Gaussian application on **Volta (Figure 1)** and **Turing (Figure 2)** architectures we are able to derive that exponential growth in terms of DRAM bytes executed per second tapers off when we reach 16x16 and 32x32 block dimensions.
- This is attributed to the amount of memory requests that comes as a result of the larger block size. This fills the memory bus which creates a delay in the process.