

UNIVERSITY OF ILLINOIS AT URBANA CHAMPAIGN

APPLIED PARALLEL PROGRAMMING

Team: wandering-gpu

---

## Final Project

---

*Author*

Alvin Sun

Yuqi Xue

Yan Miao

*Net ID*

yixiaos3

yuqixue2

yanmiao2

April 7, 2019

# Milestone 1

## 1 Kernel Statistics

Time(%)	Time	Calls	Avg	Min	Max	Name
40.10%	16.788ms	20	839.42us	1.1200us	16.155ms	[CUDA memcpy HtoD]
20.18%	8.4497ms	1	8.4497ms	8.4497ms	8.4497ms	void cudnn::detail::implicit_convolve_sgemm
11.81%	4.9434ms	1	4.9434ms	4.9434ms	4.9434ms	volta_cgemm_64x32_tn
7.05%	2.9497ms	2	1.4748ms	25.568us	2.9241ms	void op_generic_tensor_kernel
5.69%	2.3830ms	1	2.3830ms	2.3830ms	2.3830ms	void fft2d_c2r_32x32
5.59%	2.3404ms	1	2.3404ms	2.3404ms	2.3404ms	volta_sgemm_128x128_tn
4.55%	1.9059ms	1	1.9059ms	1.9059ms	1.9059ms	void cudnn::detail::pooling_fw_4d_kernel
4.18%	1.7480ms	1	1.7480ms	1.7480ms	1.7480ms	void fft2d_r2c_32x32

## 2 CUDA API Statistics

Time(%)	Time	Calls	Avg	Min	Max	Name
41.94%	2.94373s	22	133.81ms	13.721us	1.52482s	cudaStreamCreateWithFlags
34.43%	2.41664s	24	100.69ms	97.853us	2.41057s	cudaMemGetInfo
20.93%	1.46898s	19	77.315ms	817ns	393.98ms	cudaFree

## 3 Differences Between Kernels & API Calls

A CUDA kernel is an extended C function that, when called, are executed multiple times in parallel by different CUDA threads on the GPU. The CUDA APIs are programming interfaces that allow the programmer to use the CUDA device, i.e. the GPU. Kernel functions are written by the programmer and are meant to execute specific (mostly computation-intensive) tasks on the GPU, while API calls are provided by the CUDA library and are to manage the CUDA runtime environment and mostly prepare for the execution of kernels. While kernels are always executed by CUDA cores, CUDA APIs do not necessarily involve the execution of CUDA cores.

## 4 MXNet CPU Execution

```

0 Loading fashion-mnist data... done
1 Loading model... done
2 New Inference
3 EvalMetric: {'accuracy': 0.8236}

```

Run Time. 5.06s

## 5 MXNet GPU Execution

```

0 Loading fashion-mnist data... done
1 Loading model... done
2 New Inference
3 EvalMetric: {'accuracy': 0.8236}

```

**Run Time:** 4.40s

## Milestone 2

Full CPU Time	11.32s	Full CPU Time	0.250576s	Full CPU Time	1.08s
First Layer Time	2.405296s	First Layer Time	0.756567s	First Layer Time	0.035050s
Second Layer Time	7.342860	Second Layer Time	2.03s	Second Layer Time	0.075312s
10000 images		1000 images		100 images	

Table 1: CPU Run Time Statistics

## Milestone 3

### 1 Execution Summary

Accuracy	0.8397	Accuracy	0.852	Accuracy	0.84
First Layer Time	9.252ms	First Layer Time	0.677ms	First Layer Time	0.121ms
Second Layer Time	19.331ms	Second Layer Time	1.968ms	Second Layer Time	0.248ms
10000 images		1000 images		100 images	

Table 2: GPU Run Time Statistics

4616.75 ms 4617 ms 4617.25 ms 4617.75 ms 4618 ms 4618.25 ms 4618.5 ms 4618.75 ms

- Thread 59889408
  - Runtime API
  - Driver API
- Thread 2530758016
  - Runtime API
  - Driver API
- Thread 4176164608
  - Runtime API
- Thread 4167771904
  - Runtime API
- Thread 4159379200
  - Runtime API
- Profiling Overhead
- [0] TITAN V
  - Context 1 (CUDA)
    - MemCpy (HtoD)
    - MemCpy (DtoH)
    - Compute
      - 62.2% msnnet-op...

cudaMemGetInfo cudaMalloc cudaMemGetInfo cuda... cudaMemGetInfo cudaMalloc cudaDeviceSynch... cudaMemGetInfo cudaMemGetInfo

msnnet:op:for... msnnet:... msnnet:op:for... msnnet:...

volta...

Layer 1 Kernel Execution

Layer 2 Kernel Execution

The screenshot displays the NVIDIA Nsight Systems interface, showing a detailed timeline of GPU activity. The top timeline bar indicates time in seconds, ranging from 4.745 s to 4.752 s. The left sidebar lists the hierarchy of events, including threads (Thread 934430464, Thread 926037760, Thread 884090624, Thread 1113122560), context switching (Context 1 (CUDA)), and various CUDA operations (MemCpy (HtoD), MemCpy (DtoH), Compute). The main timeline area shows a sequence of operations, with two specific events circled in red and labeled with red arrows and text: 'Layer 1 Kernel Execution' and 'Layer 2 Kernel Execution'. The timeline also shows a significant gap between these two events, indicating a period of inactivity or a long-running operation. The bottom of the timeline shows a 'volta\_sgemm\_32x1...' operation, which is likely a matrix multiplication kernel.

The screenshot displays the NVIDIA Nsight System Profiler interface. The top timeline shows system events with timestamps from 4.84 s to 4.885 s. The left sidebar lists the execution context: Thread 3100022528, Thread 3019896576, Thread 2810177280, Thread 2768230144, [0] TITAN V, Context 1 (CUDA), MemCpy (HtoD), MemCpy (DtoH), and Compute. The main visualization area shows a detailed timeline of GPU activity. Two specific periods of activity are highlighted with red hand-drawn circles. The first circle, labeled 'Layer 1 kernel Execution', encompasses a series of blue bars representing kernel execution. The second circle, labeled 'Layer 2 kernel Execution', encompasses a longer series of blue bars, also representing kernel execution. The timeline is marked with vertical lines and labels such as 'cudaStreamCreate', 'cudaDeviceSynchronize', and 'cudaStr...'. The bottom status bar indicates a utilization of 81.4% for the mnet:op:forward\_kernel...

Figure 1: Kernel Execution Timelines generated by NVVP.