Project Milestone 1-4
Report

Caiting Wu (cwu72)
Hua Chen (huachen4)
Zekun Hu (zekunhu2)

Milestone 1

1. A List of all Kernels that Collectively Consume More than 90% of the Program Time:

Time(%)	Time(ms)	Name
34.00	**118.46**	void fermiPlusCgemmLDS128_batched<...>
26.94	**93.879**	void cudnn::detail::implicit_convolve_sgemm<...>
12.65	**44.060**	void fft2d_c2r_32x32<...>
8.19	**28.540**	sgemm_sm35_ldg_tn_128x8x256x16x32
6.65	**23.153**	[CUDA memcpy HtoD]
4.06	**14.157**	void cudnn::detail::activation_fw_4d_kernel<...>
3.81	**13.289**	void cudnn::detail::pooling_fw_4d_kernel<...>
1.71	**5.9454**	void fft2d_r2c_32x32<...>
1.16	**4.0542**	sgemm_sm35_ldg_tn_64x16x128x8x32

2. A list of all CUDA API calls that collectively consume more than 90% of the program time:

Time(%)	Time(ms)	Name
43.48	**1.92**	cudaStreamCreateWithFlags
27.11	**1.20**	cudaFree
20.71	**916.54**	cudaMemGetInfo
7.31	**323.45**	cudaStreamSynchronize
1.08	**47.75**	cudaMemcpy2DAsync
0.16	**7.15**	cudaMalloc
0.03	**1.37**	cuDeviceTotalMem

3. Explain the difference between kernels and API calls

Kernels are function launched to be executed on the device while APIs are used to communicate between different software components.

4. Show Output of rai Running MXNet on the CPU:

```
Successfully installed mxnet
*  Running /usr/bin/time python m1.1.py
Loading fashion-mnist data...
done
Loading model...
done
New Inference
EvalMetric: {'accuracy': 0.8444}
13.16user 12.04system 0:11.76elapsed 214%CPU (0avgtext+0avgdata 2830016maxresident)k
0inputs+2624outputs (0major+36851minor)pagefaults 0swaps
```

Figure: the snapshot of the result we have by running MXNet on the CPU

5. Program Run-Time (CPU):

From the result shown in the PowerShell, we can tell the run-time on the CPU is 11.76 seconds.

6. Show Output of rai Running MXNet on the GPU:

```
Successfully installed mxnet
*  Running /usr/bin/time python m1.2.py
Loading fashion-mnist data...
done
Loading model...
[04:54:48] src/operator/./../cudnn_algoreg-inl.h:112: Running performance tests to find
o disable)
done
New Inference
EvalMetric: {'accuracy': 0.8444}
2.22user 1.09system 0:02.81elapsed 117%CPU (0avgtext+0avgdata 1139152maxresident)k
0inputs+3136outputs (0major+159479minor)pagefa
ults 0swaps
```

Figure: the snapshot of the result we have by running MXNet on the GPU

7. Program Run-Time (GPU):

From the result shown in the PowerShell, we can tell the run-time on the GPU is 2.81 seconds.

Milestone 2

1. List whole program execution time:

a). 10000 images (default): 30.10 *s*

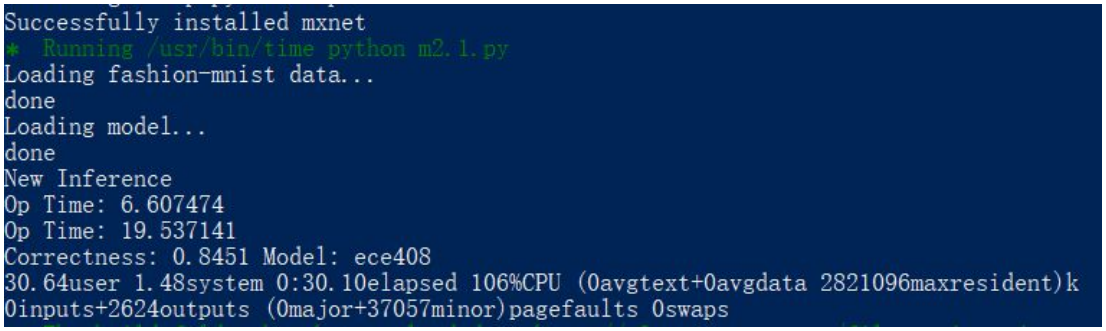

Figure: the snapshot of the result we have by running ConvNet on the CPU with 10000 images

b). 100 images: 1.15 *s*

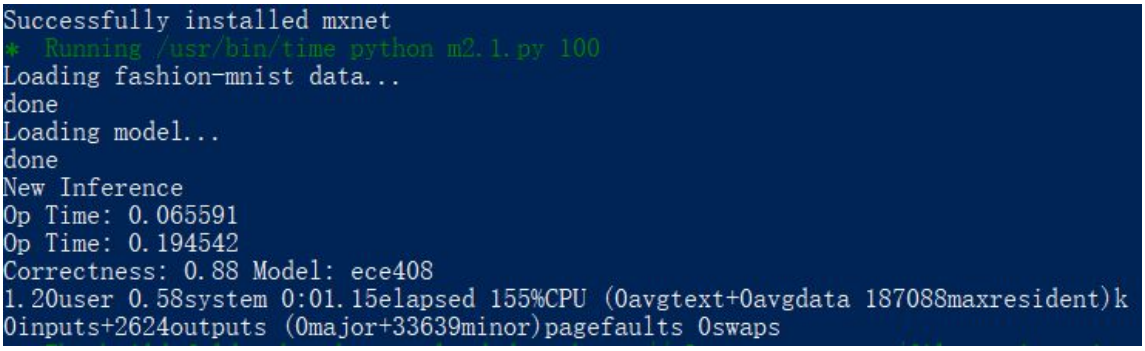

Figure: the snapshot of the result we have by running ConvNet on the CPU with 100 images

c). 10 images (default): 0.87 *s*

```
Successfully installed mxnet
*  Running /usr/bin/time python m2.1.py 10
Loading fashion-mnist data...
done
Loading model...
done
New Inference
Op Time: 0.006572
Op Time: 0.019520
Correctness: 1.0 Model: ece408
0.86user 0.51system 0:00.87elapsed 156%CPU (0avgtext+0avgdata 170392maxresident)k
0inputs+2624outputs (0major+31144minor)pag
efaults 0swaps
```

Figure: the snapshot of the result we have by running ConvNet on the CPU with 10 images

2. List Op Times:

a). 10000 images (default):

First Layer Op Time:	6.60747 *s*
Second Layer Op Time:	19.537141 *s*

b). 100 images:

First Layer Op Time:	0.065591 *s*
Second Layer Op Time:	0.194542 *s*

c). 10 images:

First Layer Op Time:	0.006572 *s*
Second Layer Op Time:	0.019520 *s*

Milestone 3

1. : nvprof profiling the execution (10000 default data size):

```
* Running nvprof python m3.1.py
Loading fashion-mnist data...
done
Loading model...
==314== NVPROF is profiling process 314, command: python m3.1.py
done

New Inference
Op Time: 0.355486
Op Time: 0.652564

Correctness: 0.8451 Model: ece408
==314== Profiling application: python m3.1.py

==314== Profiling result:
```

Time (%)	Time	Calls	Name
89.76	**1.00787 s**	2	mxnet::op::forward_kernel
2.61	**29.329 ms**	1	sgemm_sm35_ldg_tn_128x8x256x16x32
2.60	**29.224 ms**	14	[CUDA memcpy HtoD]
1.96	**22.008 ms**	2	mshadow::cuda::MapPlanLargeKernel<...>
1.26	**14.158 ms**	2	cudnn::detail::activation_fw_4d_kernel<...>
1.19	**13.310 ms**	1	cudnn::detail::pooling_fw_4d_kernel<...>
0.36	**4.0730 ms**	1	sgemm_sm35_ldg_tn_64x16x128x8x32
0.11	**1.2805 ms**	1	mshadow::cuda::MapPlanLargeKernel<...>
0.10	**1.1112 ms**	1	mshadow::cuda::SoftmaxKernel<...>
0.02	**177.44 us**	13	mshadow::cuda::MapPlanKernel<...>
0.01	**146.82 us**	2	mshadow::cuda::MapPlanKernel<...>
0.01	**129.60 us**	1	sgemm_sm35_ldg_tn_32x16x64x8x16
<0.01	**23.264 us**	1	mshadow::cuda::MapPlanKernel<...>
<0.01	**9.6320 us**	1	[CUDA memcpy DtoH]

```
==314== API calls:
Time(%)    Time    Call    Avg       Min       Max       Name
 37.07%  2.01992s   18  112.22ms 18.700us 1.00956s    cudaStreamCreateWithFlags
 23.45%  1.27787s   10  127.79ms  1.0610us  359.44ms  cudaFree
 18.90%  1.02999s    6  171.66ms  16.423us  652.50ms  cudaDeviceSynchronize
 18.12%  987.04ms   27  36.557ms  249.68us  978.97ms  cudaMemGetInfo
  1.15%  62.495ms   29  2.1550ms  5.9750us  32.128ms  cudaStreamSynchronize
  1.09%  59.196ms    9  6.5774ms  10.270us  28.498ms  cudaMemcpy2DAsync
  0.12%  6.6691ms   45  148.20us  9.9270us  950.38us  cudaMalloc
  0.03%  1.4217ms    4  355.43us  336.63us  403.41us  cuDeviceTotalMem
  0.02%  1.0917ms 352  3.1010us     512ns  88.433us  cuDeviceGetAttribute
  0.02%  827.59us 114  7.2590us     897ns  305.72us  cudaEventCreateWithFlags
  0.01%  681.04us   28  24.322us  11.212us  67.046us  cudaLaunch
  0.01%  407.12us    6  67.853us  28.668us  137.98us  cudaMemcpy
  0.01%  287.64us    4  71.909us  36.702us  160.69us  cudaStreamCreate
  0.00%  134.58us 168     801ns     522ns  13.878us  cudaSetupArgument
  0.00%  120.73us    4  30.182us  18.224us  36.266us  cuDeviceGetName
  0.00%  96.465us 104     927ns     627ns  2.3410us  cudaDeviceGetAttribute
  0.00%  89.771us   34  2.6400us     925ns  7.5090us  cudaSetDevice
  0.00%  45.847us 2  22.923us  21.098u   24.749 uscudaStreamCreateWithPriority
  0.00%  45.809us   28  1.6360us     779ns  4.3910us  cudaConfigureCall
  0.00%  30.516us   10  3.0510us  1.6350us  7.0540us  cudaGetDevice
  0.00%  17.136us   20     856ns     583ns  1.2970us  cudaPeekAtLastError
  0.00%  7.0090us    6  1.1680us     550ns  2.5230us  cuDeviceGetCount
  0.00%  6.0670us    2  3.0330us  2.5290us  3.5380us  cudaStreamWaitEvent
  0.00%  5.6780us    6     946ns     604ns  1.2170us  cuDeviceGet
  0.00%  5.3790us    2  2.6890us  1.8120us  3.5670us  cudaEventRecord
  0.00%  5.2660us    1  5.2660us  5.2660us  5.2660us  cudaStreamGetPriority
  0.00%  4.7120us    5     942ns     726ns  1.2410us  cudaGetLastError
  0.00%  4.1180us    3  1.3720us  1.2970us  1.4390us  cuInit
  0.00%  3.8380us    2  1.9190us  1.5640us  2.2740us
cudaDeviceGetStreamPriorityRange
  0.00%  3.5850us    3  1.1950us     907ns  1.3690us  cuDriverGetVersion
  0.00%  1.7840us    1  1.7840us  1.7840us  1.7840us  cudaGetDeviceCount
```

2. nvprof profiling the execution (100 data size):

```
* Running nvprof python m3.1.py 100
Loading fashion-mnist data...
done
Loading model...
==311== NVPROF is profiling process 311, command: python m3.1.py 100
done

New Inference
Op Time: 0.003272
Op Time: 0.006654

Correctness: 0.88 Model: ece408
==311== Profiling application: python m3.1.py 100
```

==311== **Profiling result:**

Time (%)	Time	Calls	Name
78.67	**9.8344 ms**	2	mxnet::op::forward_kernel
9.15	**1.1435 ms**	14	[CUDA memcpy HtoD]
5.45	**680.92 us**	1	sgemm_largek_lds64
2.02	**252.67 us**	2	void mshadow::cuda::MapPlanKernel<...>
1.58	**197.18 us**	14	void mshadow::cuda::MapPlanKernel<...>
1.30	**162.62 us**	2	void cudnn::detail::activation_fw_4d_kernel<...>
1.12	**140.51 us**	1	void cudnn::detail::pooling_fw_4d_kernel<...>
0.33	**41.248 us**	1	sgemm_sm35_ldg_tn_32x16x64x8x16
0.15	**18.239 us**	1	void mshadow::cuda::SoftmaxKernel<...>
0.09	**11.808 us**	1	void mshadow::cuda::MapPlanKernel<...>
0.06	**7.0720 us**	2	void mshadow::cuda::MapPlanKernel<...>
0.04	**4.5760 us**	1	[CUDA memcpy DtoH]
0.03	**3.9680 us**	1	void scal_kernel<...>
0.01	**1.4080 us**	1	[CUDA memset]

==311== **API calls:**

Time(%)	Time	Call	Avg	Min	Max	Name
47.00%	1.99860s	16	124.91ms	18.774us	999.00ms	cudaStreamCreateWithFlags
29.76%	1.26567s	10	126.57ms	1.3710us	357.94ms	cudaFree
22.72%	966.04ms	27	35.779ms	337.00us	957.04ms	cudaMemGetInfo
0.24%	10.142ms	6	1.6903ms	4.9720us	6.6253ms	cudaDeviceSynchronize
0.07%	2.8118ms	9	312.43us	10.485us	783.21us	cudaMemcpy2DAsync
0.07%	2.7961ms	45	62.134us	9.4710us	187.97us	cudaMalloc
0.04%	1.6533ms	29	57.010us	5.1150us	650.77us	cudaStreamSynchronize
0.03%	1.3740ms	4	343.51us	341.96us	344.80us	cuDeviceTotalMem
0.02%	1.0064ms	352	2.8590us	502ns	77.755us	cuDeviceGetAttribute
0.02%	759.06us	112	6.7770us	866ns	296.13us	cudaEventCreateWithFlags
0.01%	530.04us	28	18.930us	9.7590us	55.998us	cudaLaunch
0.01%	463.17us	6	77.195us	26.242us	129.89us	cudaMemcpy
0.01%	330.98us	4	82.743us	21.216us	228.00us	cudaStreamCreate
0.00%	125.71us	4	31.426us	22.651us	35.699us	cuDeviceGetName
0.00%	111.02us	158	702ns	520ns	1.6090us	cudaSetupArgument
0.00%	99.803us	104	959ns	684ns	2.5010us	cudaDeviceGetAttribute
0.00%	80.671us	34	2.3720us	801ns	6.8400us	cudaSetDevice

```
  0.00%  45.615us     2  22.807us  20.577us  25.038us
cudaStreamCreateWithPriority
  0.00%  36.588us    28  1.3060us     655ns  3.9660us  cudaConfigureCall
  0.00%  34.730us     1  34.730us  34.730us  34.730us  cudaMemsetAsync
  0.00%  30.635us    10  3.0630us  1.4260us  6.9790us  cudaGetDevice
  0.00%  15.498us    20     774ns     575ns  1.1170us  cudaPeekAtLastError
  0.00%  8.5720us     8  1.0710us     530ns  2.6530us  cudaGetLastError
  0.00%  7.9230us     1  7.9230us  7.9230us  7.9230us  cudaEventQuery
  0.00%  6.8350us     6  1.1390us     545ns  2.7790us  cuDeviceGetCount
  0.00%  5.0810us     6     846ns     687ns  1.1550us  cuDeviceGet
  0.00%  4.1240us     3  1.3740us  1.3230us  1.4450us  cuInit
  0.00%  3.9170us     2  1.9580us  1.6040us  2.3130us
cudaDeviceGetStreamPriorityRange
  0.00%  2.8320us     1  2.8320us  2.8320us  2.8320us  cudaEventRecord
  0.00%  2.7000us     3     900ns     896ns     907ns  cuDriverGetVersion
  0.00%  1.8210us     1  1.8210us  1.8210us  1.8210us  cudaGetDeviceCount
```

3. nvprof profiling the execution (10 data size):

```
* Running nvprof python m3.1.py 10
Loading fashion-mnist data...
done
Loading model...
==314== NVPROF is profiling process 314, command: python m3.1.py 10
done

New Inference
Op Time: 0.000260
Op Time: 0.000733

Correctness: 1.0 Model: ece408

==314== Profiling application: python m3.1.py 10

==314== Profiling result:
```

Time (%)	Time	Calls	Name
38.50	**897.27 us**	2	mxnet::op::forward_kernel
36.27	**845.40 us**	14	[CUDA memcpy HtoD]
10.71	**249.60 us**	1	void sgemm_largek_lds64
7.67	**178.85 us**	14	void mshadow::cuda::MapPlanKernel<...>
1.74	**40.448 us**	1	sgemm_sm35_ldg_tn_32x16x64x8x16
1.37	**31.840 us**	2	void mshadow::cuda::MapPlanKernel

1.11	**25.823 us**	2	void cudnn::detail::activation_fw_4d_kernel<...>
0.81	**18.976 us**	1	void cudnn::detail::pooling_fw_4d_kernel<...>
0.63	**14.720 us**	1	void mshadow::cuda::SoftmaxKernel<...>
0.48	**11.168 us**	1	void mshadow::cuda::MapPlanKernel<...>
0.27	**6.2720 us**	2	void mshadow::cuda::MapPlanKernel<...>
0.20	**4.6400 us**	1	[CUDA memcpy DtoH]
0.18	**4.1280 us**	1	void scal_kernel<...>
0.06	**1.4400 us**	1	[CUDA memset]

```
==314== API calls:
Time(%)    Time  Call   Avg       Min       Max     Name
 47.16%  2.03746s 16  127.34ms  19.098us  1.01841s  cudaStreamCreateWithFlags
 29.59%  1.27843s 10  127.84ms  1.2330us  363.76ms  cudaFree
 22.87%  988.11ms 27  36.597ms  337.32us  979.12ms  cudaMemGetInfo
  0.09%  3.9655ms  4  991.36us  23.032us  3.8549ms  cudaStreamCreate
  0.06%  2.5601ms 45  56.891us  7.8580us  208.19us  cudaMalloc
  0.06%  2.4786ms  9  275.40us  12.252us  758.74us  cudaMemcpy2DAsync
  0.03%  1.3842ms  4  346.06us  340.94us  360.30us  cuDeviceTotalMem
  0.02%  1.0547ms 352  2.9960us    516ns  117.54us  cuDeviceGetAttribute
  0.02%  988.21us  6  164.70us  5.2810us  702.96us  cudaDeviceSynchronize
  0.02%  935.43us 29  32.256us  5.7630us  218.72us  cudaStreamSynchronize
  0.02%  859.79us 112 7.6760us     930ns  294.14us  cudaEventCreateWithFlags
  0.01%  538.01us 28  19.214us  9.2970us  58.700us  cudaLaunch
  0.01%  477.63us  6  79.605us  42.779us  130.92us  cudaMemcpy
  0.00%  120.71us  4  30.176us  18.506us  35.825us  cuDeviceGetName
  0.00%  109.93us 158     695ns     414ns  1.7770us  cudaSetupArgument
  0.00%  100.97us 104     970ns     571ns  2.1010us  cudaDeviceGetAttribute
  0.00%  83.217us 34  2.4470us     941ns  7.5720us  cudaSetDevice
  0.00%  42.570us  2  21.285us  20.207us  22.363us
cudaStreamCreateWithPriority
  0.00%  37.688us 28  1.3460us     628ns  4.0480us  cudaConfigureCall
  0.00%  35.023us  1  35.023us  35.023us  35.023us  cudaMemsetAsync
  0.00%  31.994us 10  3.1990us  1.8510us  7.2940us  cudaGetDevice
  0.00%  16.583us 20     829ns     596ns  1.1390us  cudaPeekAtLastError
  0.00%  9.4350us  8  1.1790us     555ns  3.8670us  cudaGetLastError
  0.00%  6.7170us  6  1.1190us     525ns  2.4450us  cuDeviceGetCount
  0.00%  6.5240us  1  6.5240us  6.5240us  6.5240us  cudaEventQuery
  0.00%  5.8530us  6     975ns     673ns  1.9630us  cuDeviceGet
  0.00%  4.5010us  3  1.5000us  1.3600us  1.6470us  cuInit
  0.00%  3.8660us  2  1.9330us  1.5120us  2.3540us
cudaDeviceGetStreamPriorityRange
  0.00%  2.8840us  3     961ns     956ns     968ns  cuDriverGetVersion
  0.00%  2.5270us  1  2.5270us  2.5270us  2.5270us  cudaEventRecord
  0.00%  1.6070us  1  1.6070us  1.6070us  1.6070us  cudaGetDeviceCount
```

4. Op Times Comparison (CPU vs GPU):

a). 10000 images (default):

	CPU Time	**GPU Time**
First Layer Op Time:	6.60747 *s*	0.355486 *s*
Second Layer Op Time:	19.537141 *s*	0.652564 *s*

b). 100 images:

	CPU Time	**GPU Time**
First Layer Op Time:	0.065591 *s*	0.003272 *s*
Second Layer Op Time:	0.194542 *s*	0.006654 *s*

c). 10 images:

	CPU Time	**GPU Time**
First Layer Op Time:	0.006572 *s*	0.000260 *s*
Second Layer Op Time:	0.019520 *s*	0.000733 *s*

4. Discussion:

In this milestone we have completed a GPU implementation of the forward convolution. Though there is further optimization added into the new code, we could still observe the significant speed-boost brought by CPU computation. For the default data size, the time spent has been shrunk from 6.6 s to within half a second. This much performance enhancement is exactly why GPU computing is so powerful in many occasions.

Milestone 4

1. **Describe the Optimization:**

   - **Shared Memory:**

   In this optimization, we store input X for all threads in one block in shared memory.

   - **Constant Memory:**

   In this optimization, we store all weight matrix in constant memory to reduce iterations of global memory reads therefore significantly reduce runtime.

   - **Shared Memory Unrolled Multiplication:**

   First we unroll the convolution matrix and use share memory to store the matrix.

   - **Four Streams:**

   in the shared memory unrolled multiplication, we also used 4 different streams to improve execution efficiency.

2. **Demonstrate nvprof profiling the execution:**

   Shared Memory Convolution with Constant Memory Weight:

```
Loading model...
==311== NVPROF is profiling process 311, command: python m3.1.py
done
New Inference
The Value of N = 10000
The Value of M = 6
The Value of C = 1
The Value of H = 64
The Value of W = 64
The Value of K = 5
Op Time: 2.120822
The Value of N = 10000
The Value of M = 16
```

```
The Value of C = 6
The Value of H = 30
The Value of W = 30
The Value of K = 5
Op Time: 0.328743
Correctness: 0.8451 Model: ece408
==311== Profiling application: python m3.1.py
==314== Profiling result:
```

Time (%)	Time	Calls	Name
73.19	**107.96 ms**	1	mxnet::op::forward_kernel
15.89	**23.439 ms**	14	[CUDA memcpy HtoD]
9.93	**14.646 ms**	1	mshadow::cuda::MapPlanLargeKernel
0.87	**1.285 ms**	1	mshadow::cuda::MapPlanLargeKernel
0.12	**178.11 us**	13	mshadow::cuda::MapPlanKernel
<0.01	**4.128 us**	1	[CUDA memcpy DtoD]

```
==311== API calls:
```

Time(%)	Time	Calls	Avg	Min	Max	Name
44.32%	1.84788s	16	115.49ms	19.322us	923.48ms	cudaStreamCreateWithFlags
29.50%	1.23001s	10	123.00ms	1.0680us	360.37ms	cudaFree
21.77%	907.83ms	26	34.916ms	136.70us	903.51ms	cudaMemGetInfo
2.94%	122.63ms	2	61.313ms	14.654ms	107.97ms	cudaDeviceSynchronize
1.14%	47.495ms	8	5.9369ms	8.8220us	22.926ms	cudaMemcpy2DAsync
0.19%	7.8190ms	44	177.70us	9.7640us	1.1571ms	cudaMalloc
0.05%	1.9510ms	22	88.680us	10.595us	1.2945ms	cudaStreamSynchronize
0.02%	1.0186ms	352	2.8930us	518ns	70.621us	cuDeviceGetAttribute
0.02%	843.50us	112	7.5310us	902ns	202.79us	cudaEventCreateWithFlags
0.02%	714.20us	4	178.55us	177.08us	180.80us	cuDeviceTotalMem
0.01%	444.89us	4	111.22us	26.802us	309.52us	cudaStreamCreate
0.01%	378.59us	6	63.098us	25.125us	90.336us	cudaMemcpy
0.01%	347.93us	16	21.745us	11.244us	50.589us	cudaLaunch
0.00%	115.98us	4	28.993us	25.418us	33.711us	cuDeviceGetName
0.00%	101.30us	104	974ns	692ns	1.9910us	cudaDeviceGetAttribute
0.00%	72.260us	30	2.4080us	996ns	6.5100us	cudaSetDevice
0.00%	52.148us	71	734ns	531ns	2.2610us	cudaSetupArgument
0.00%	41.668us	2	20.834us	19.904us	21.764us	cudaStreamCreateWithPriority
0.00%	39.679us	1	39.679us	39.679us	39.679us	cudaMemcpyToSymbol
0.00%	28.373us	10	2.8370us	1.5860us	7.0320us	cudaGetDevice
0.00%	22.827us	16	1.4260us	828ns	3.3400us	cudaConfigureCall
0.00%	12.146us	15	809ns	640ns	1.1540us	cudaPeekAtLastError
0.00%	6.0130us	6	1.0020us	551ns	2.0380us	cuDeviceGetCount
0.00%	4.7240us	6	787ns	576ns	1.0530us	cuDeviceGet
0.00%	4.0290us	1	4.0290us	4.0290us	4.0290us	cudaProfilerStart
0.00%	3.5200us	2	1.7600us	1.5790us	1.9410us	cudaDeviceGetStreamPriorityRange

```
 0.00%  3.4920us     3  1.1640us  1.0900us  1.3060us  cuInit
 0.00%  3.0650us     3  1.0210us     925ns  1.0930us  cuDriverGetVersion
 0.00%  1.4400us     1  1.4400us  1.4400us  1.4400us  cudaGetDeviceCount
```

Shared / Unrolled Multiplication w/ Multiple Streams(10000 Size):

```
Loading model...
==314== NVPROF is profiling process 314, command: python m3.1.py
done
New Inference
The Value of N = 10000
The Value of M = 6
The Value of C = 1
The Value of H = 64
The Value of W = 64
The Value of K = 5
Op Time: 3.247097
The Value of N = 10000
The Value of M = 16
The Value of C = 6
The Value of H = 30
The Value of W = 30
The Value of K = 5
Op Time: 0.357719
Correctness: 0.8451 Model: ece408
==314== Profiling application: python m3.1.py


==314== Profiling result:
```

Time (%)	Time	Calls	Name
46.20	**209.83 ms**	10000	mxnet::op::unroll_Kernel
45.33	**205.89 ms**	10000	mxnet::op::matrixMultiply
4.92	**22.367 ms**	14	[CUDA memcpy HtoD]
3.22	**14.602 ms**	1	mshadow::cuda::MapPlanLargeKernel
0.28	**1.2839 ms**	1	mshadow::cuda::MapPlanLargeKernel
0.04	**178.65 us**	13	mshadow::cuda::MapPlanKernel
<0.01	**4.0320 us**	1	[CUDA memcpy DtoD]

```
==314== API calls:
Time(%)    Time    Call    Avg       Min       Max     Name
 42.69%  1.79415s  16  112.13ms  26.762us  896.99ms  cudaStreamCreateWithFlags
 28.78%  1.20969s  10  120.97ms  1.3990us  352.32ms  cudaFree
```

```
 21.33%  896.38ms  26  34.476ms  136.98us  892.18ms  cudaMemGetInfo
 3.17%  133.12ms 20015 6.6500us 5.9230us  62.417us cudaLaunch
 1.98%  83.127ms  150062  553ns     523ns  247.64us  cudaSetupArgument
  1.10%  46.044ms   8  5.7555ms  12.935us  22.203ms  cudaMemcpy2DAsync
  0.35%  14.611ms   1  14.611ms  14.611ms  14.611ms  cudaDeviceSynchronize
  0.27%  11.539ms 20015   576ns     542ns  10.440us  cudaConfigureCall
  0.19%  8.0929ms  45  179.84us  10.430us  1.1575ms  cudaMalloc
  0.05%  1.9462ms  22  88.464us  11.102us  1.2944ms  cudaStreamSynchronize
  0.03%  1.2066ms 112  10.772us  1.3490us  362.71us  cudaEventCreateWithFlags
  0.02%  1.0157ms 352  2.8850us     513ns  86.580us  cuDeviceGetAttribute
  0.02%  775.30us   8  96.912us  22.294us  448.95us  cudaStreamCreate
  0.02%  739.33us   4  184.83us  178.05us  203.58us  cuDeviceTotalMem
  0.01%  263.67us   6  43.945us  26.839us  78.282us  cudaMemcpy
  0.00%  115.67us 104  1.1120us     919ns  2.3130us  cudaDeviceGetAttribute
  0.00%  104.02us   4  26.004us  20.197us  30.667us  cuDeviceGetName
  0.00%  72.011us  30  2.4000us     989ns  8.9160us  cudaSetDevice
  0.00%  64.586us   2  32.293us  26.879us  37.707us  cudaStreamCreateWithPriority
  0.00%  33.753us   1  33.753us  33.753us  33.753us  cudaMemcpyToSymbol
  0.00%  20.783us  10  2.0780us  1.3100us  2.9880us  cudaGetDevice
  0.00%  10.683us  15     712ns     624ns     985ns  cudaPeekAtLastError
  0.00%  5.7560us   6     959ns     558ns  1.8060us  cuDeviceGetCount
  0.00%  5.4450us   6     907ns     651ns  1.6720us  cuDeviceGet
  0.00%  4.4200us   2  2.2100us  1.9340us  2.4860us  cudaDeviceGetStreamPriorityRange
  0.00%  3.4730us   1  3.4730us  3.4730us  3.4730us  cudaProfilerStart
  0.00%  3.2450us   3  1.0810us  1.0000us  1.2350us  cuInit
  0.00%  2.9010us   3     967ns     869ns  1.0260us  cuDriverGetVersion
  0.00%  1.3920us   1  1.3920us  1.3920us  1.3920us  cudaGetDeviceCount
```

3. Use NVVP to analyze your optimization:

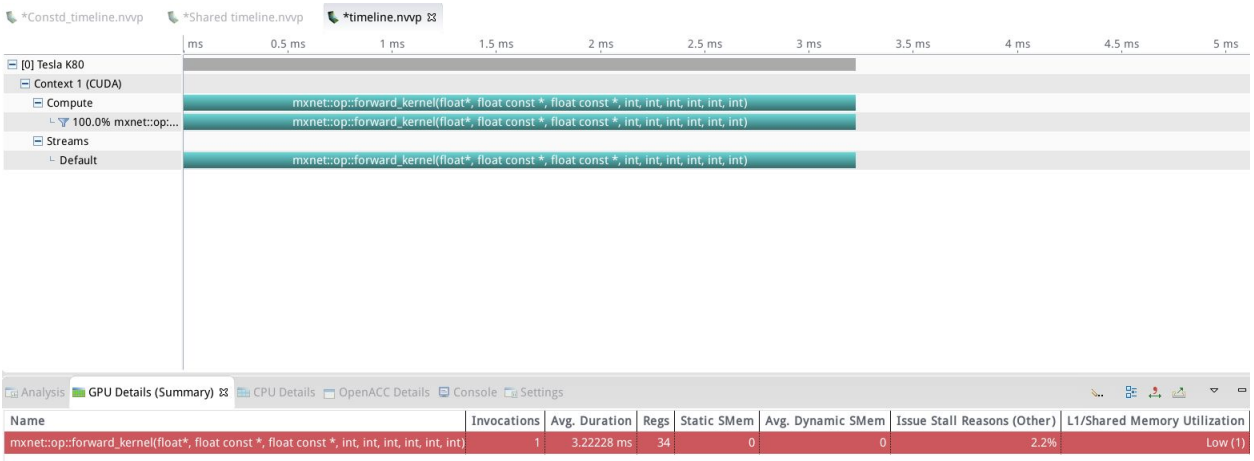

Figure: The original running time on first layer with 100 input size

Based on one of the student suggested on Piazza @310, we decided to focus on the first layer of 100 input size to compare across all of our implementations, since generating the timeline report could take a while and the rai is not quite stable throughout the time.

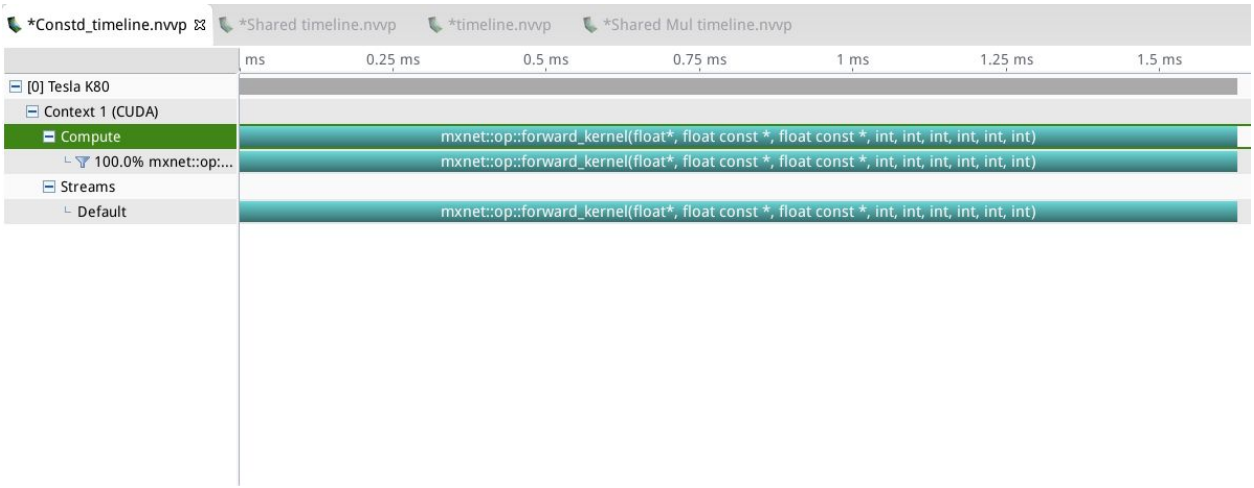

Figure: The running time on first layer with 100 input size with Constant Memory

The running time for the first layer with 100 input size on the original forward kernel is 3.22228 *ms*. After we put the weight matrix into the constant memory, we get a 1.62703 *ms*. We can see that using constant memory alone already provides a huge optimization.

Figure: The running time on first layer with 100 input size with Constant Memory & Shared Memory Convolution

Combining constant memory with shared memory convolution and constant memory with shared memory unrolled multiplication, we see that the former has a better result. The Constant Memory and Shared Memory Convolution gives us a 1.02945 *ms*.

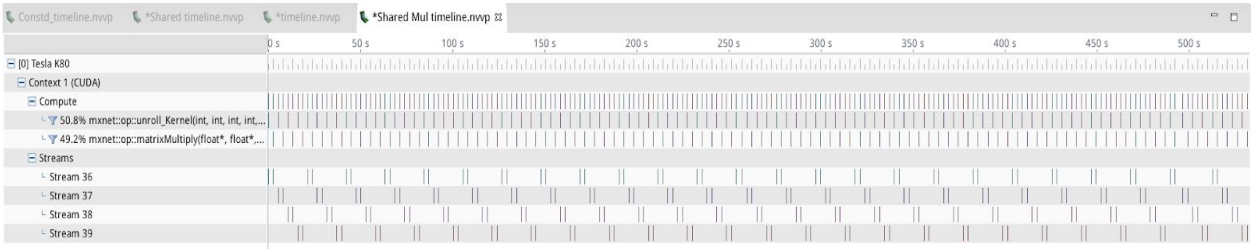

Figure: The running time on first layer with 100 input size with Constant Memory & Shared Memory Matrix Multiplication with 4 streams

The later methods has a bit longer running time: 1.15476 *ms*. The session time in total is 531.2457 *s*, which we think is resulted from the profiler needs to keep track of each kernel call, and there are 100 iteration in the for-loop based on our design. This is mostly likely due to a suboptimal multiplication methods we implemented - tiling multiplication. Besides, the way that we have implemented using different streams could be immature and has a lot of improvement potential. In our future works, we wish to further optimize our multiplication kernels, for example GEMM. Also, we suspect that parameter tuning would also be instrumental in this project, and we wish to further analyze it in the final submission.