CS483/ECE408 Final Project

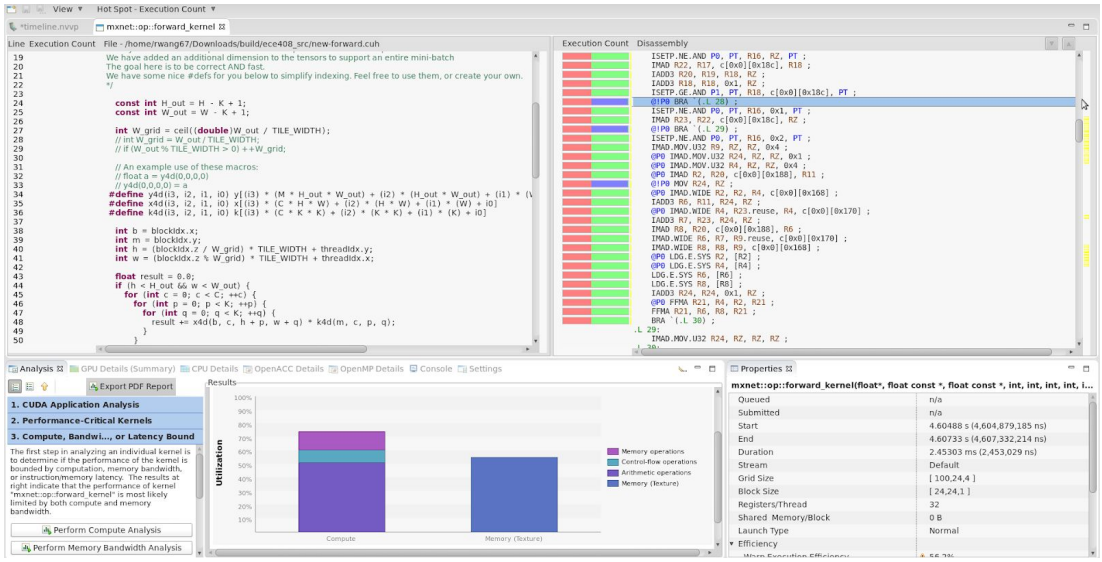
Team: kriskr
Team member:

Ruoxi Yang	ryang28
Rongzi Wang	rwang67
Shiqi Sun	shiqis2

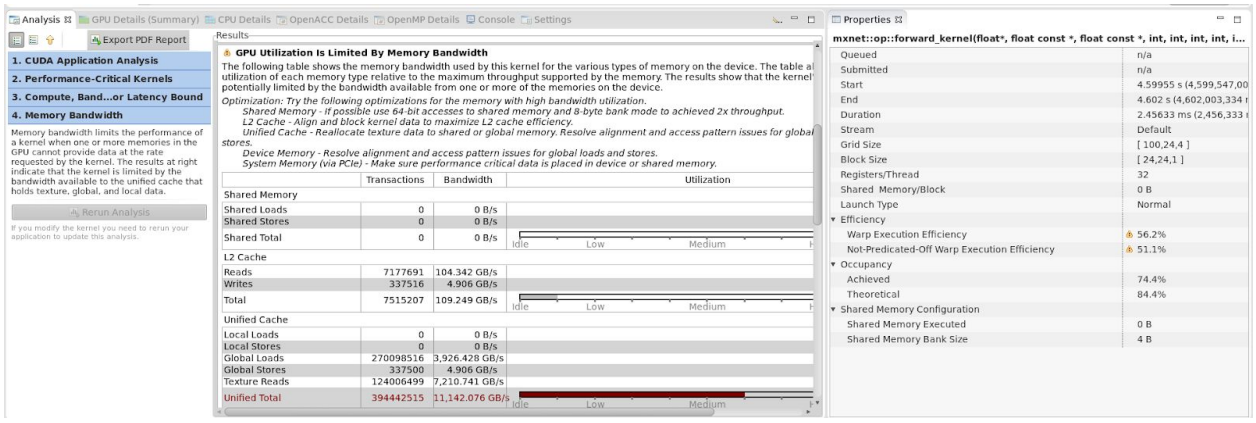
Milestone 3

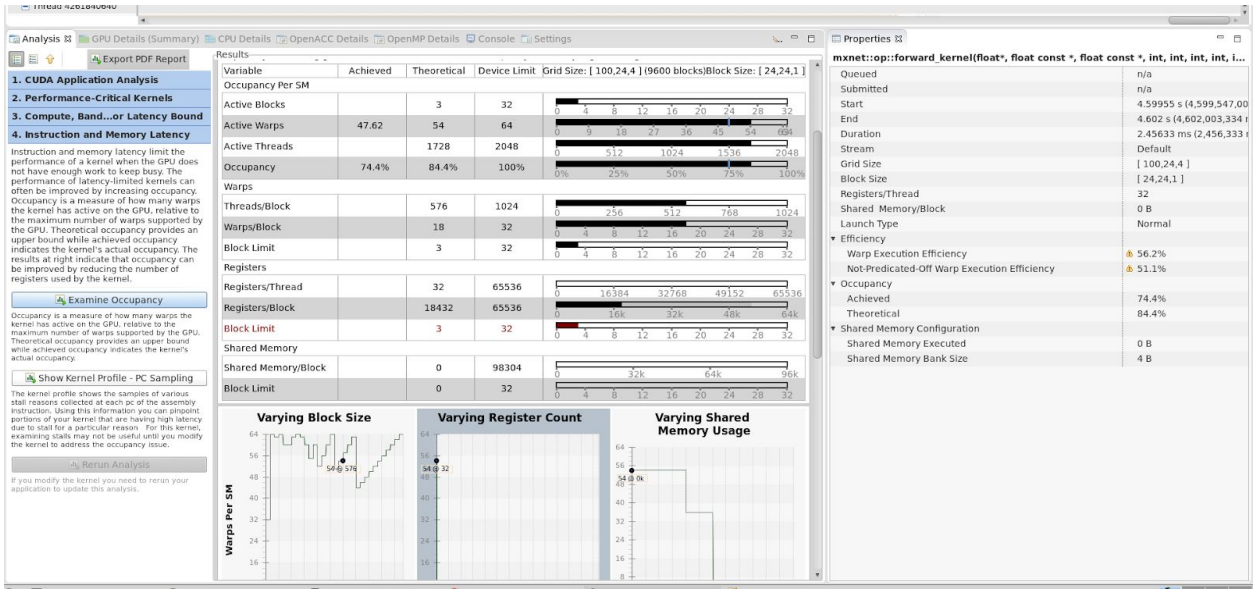
Dataset size	Correctness	Timing
100	0.85	Op Time: 0.000417 Op Time: 0.002454
1000	0.827	Op Time: 0.004056 Op Time: 0.024617
10000	0.8171	Op TIme:0.040040 Op Time:0.287162

Yes, By analyzing the timeline.nvvp file, since we are not using share memory , we are having coalescing issues which leads to bad bandwidth utilization. We have a low global memory load efficiency and low shared memory efficiency.
For our kernel performance, most of the time is spend in arithmetic operations and the second largest utilization is memory operations.

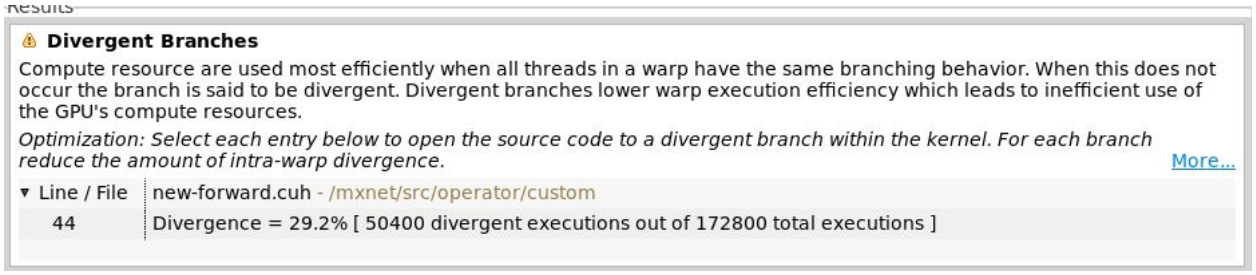


The GPU utilization is limited by memory bandwidth.





Our code also has a bad divergence (29.2%), and divergent branches is a big limiting factors in the performance:



We are also not using the share memory and the usage is zero:

Memory Bandwidth And Utilization

The following table shows the memory bandwidth used by this kernel for the various types of memory on the device.
The table also shows the utilization of each memory type relative to the maximum throughput supported by the memory. More...

	Transactions	Bandwidth	Utilization
Shared Memory			
Shared Loads	0	0 B/s	
Shared Stores	0	0 B/s	
Shared Total	0	0 B/s	Idle Low Medium High Max
L2 Cache			
Reads	7177691	104.342 GB/s	
Writes	337516	4.906 GB/s	
Total	7515207	109.249 GB/s	Idle Low Medium High Max
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	270098516	3,926.428 GB/s	
Global Stores	337500	4.906 GB/s	
Texture Reads	124006499	7,210.741 GB/s	
Unified Total	394442515	11,142.076 GB/s	Idle Low Medium High Max
Device Memory			
Reads	1727652	25.115 GB/s	
Writes	328108	4.77 GB/s	
Total	2055760	29.885 GB/s	Idle Low Medium High Max
System Memory [PCIe configuration: Gen3 x16, 8 Gbit/s]			
Reads	0	0 B/s	Idle Low Medium High Max
Writes	5	72.685 kB/s	Idle Low Medium High Max

Milestone 2

Name	Layer 1	Layer 2
Op Time	22.041992s	105.005034s

Whole program execution time: 127.09s

Milestone 1

A list of Kernels that collectively consume more than 90% of the program time:

Name	Time	Percentage
CUDA memcpy HtoD	37.345ms	38.63%
volta_scudnn_128x32_relu_interior_nn_v1	20.831ms	21.55%
implicit_convolve_sgemm	19.117ms	19.78%
activation_fw_4d_kernel	7.4430ms	7.70%

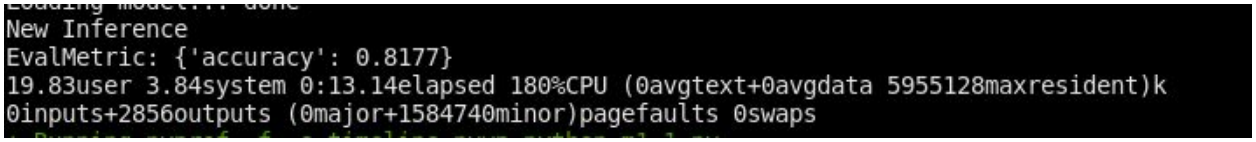
volta_sgemm_128x128_tn	6.7882ms	7.02%
pooling_fw_4d_kernel	4.4341ms	4.59%

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

Name	Time	Percentage
cudaStreamCreateWithFlags	2.80865s	39.54%
cudaMemGetInfo	2.47336s	34.82%
cudaFree	1.55123s	21.84%

Explanation of the difference between kernels and API calls:
Kernel is a C function that programer define and expect the cuda threads to execute. The runtime API eases device code management by providing implicit initialization, context management, and module management, making it easier to code. During the runtime, all the kernels are automatically loaded during initialization and stay loaded for as long as the program runs.

Output of rai running MXNet on the CPU: The accuracy is 0.8177.
Program run time: The elapsed time is 0:13:14



Output of rai running MXNet on the GPU: accuracy is 0.8177
Program run time: The elapsed time is 0:05.80

```
New Inference
EvalMetric: {'accuracy': 0.8177}
4.31user 2.85system 0:05.80elapsed 123%CPU (0avgtext+0avgdata 2836900maxresident)k
0inputs+4568outputs (0major+703473minor)pagefaults 0swaps
```