ECE 408 Final Report - Fall 2019

Team: sig_sev
Arkin Shah (arkinas2)
Kevin Lee (kevinl8)
******Disclaimer***:*** *The week of 11/11/19, Siddarth (sa10) had to drop the course as required for a medical leave due to a serious spine injury.*

Contributions:
MS1 & MS2 & MS3: All three group members worked equally on all parts of the project.
MS4 & final: Both two group members worked together equally on all parts of the project.

Optimization 4: Optimizing tiling width
Filename: new-forward4.cuh
Op time:

Dataset size 10000

```
New Inference
Op Time: 0.096068
Op Time: 0.258385
Correctness: 0.7653 Model: ece408
5.30user 3.58system 0:05.19elapsed 171%CPU (0avgtext+0avgdata 2973960maxresident)k
0inputs+4568outputs (0major+736050minor)pagefaults 0swaps
```

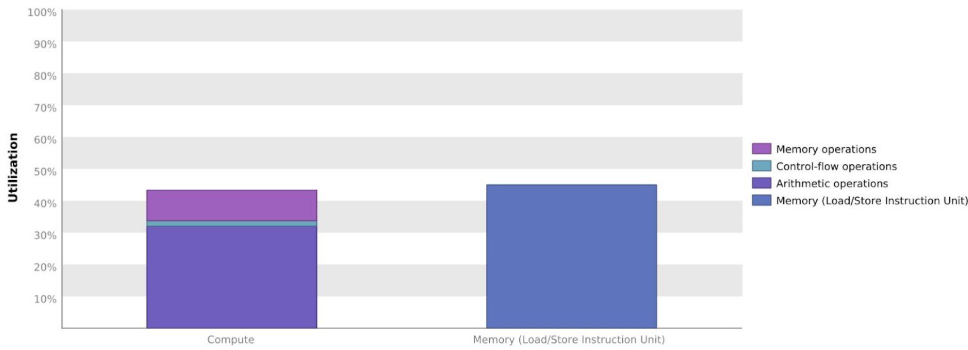
Dataset size 1000

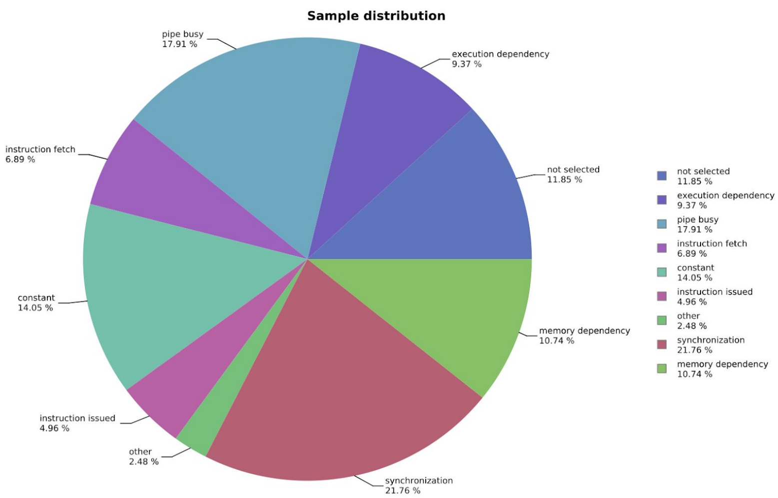
```
New Inference
Op Time: 0.009737
Op Time: 0.027539
Correctness: 0.767 Model: ece408
4.99user 3.11system 0:04.56elapsed 177%CPU (0avgtext+0avgdata 2785172maxresident)k
0inputs+4568outputs (0major+639388minor)pagefaults 0swaps
```

Dataset size 100

```
New Inference
Op Time: 0.000994
Op Time: 0.002767
Correctness: 0.76 Model: ece408
4.87user 3.23system 0:04.48elapsed 180%CPU (0avgtext+0avgdata 2771596maxresident
0inputs+3128outputs (0major+634263minor)pagefaults 0swaps
```

Nvprof:





The purpose of this small optimization was to optimize the tile width in the shared matrix multiplication kernel. The tile width was changed from 32 to 24. While this may seem like a small change, there are some interesting effects on performance.

This optimization attempted to resolve the memory utilization bottleneck from previous optimizations. Based on the nvprof, there was an increase in the memory load/store instruction usage and a decrease in memory arithmetic operations. However, as can be seen in the pie chart figure, performance hiderenace from memory dependence decreased significantly. Issues with synchronization increased thus serving as a more significant bottleneck to performance than before. This is likely due to the smaller tile size causing issues with warps and potentially increased divergence.

Further, the occupancy also decreased to 61.5% and active warps 39.3.

Line 49	Divergence = 2.8% [205 divergent executions out of 7380 total executions]

As predicted, divergence occurs affecting warp behavior and efficiency from the decreased tile size.

Optimization 5: Implementing Constant Memory Matrix multiplication
Filename: new-forward5.cuh

Op time:
Dataset size 10000

```
New Inference
Op Time: 0.118075
Op Time: 0.468743
Correctness: 0.7653 Model: ece408
5.53user 3.54system 0:05.40elapsed 168%CPU (0avgtext+0avgdata 2971212maxresident)k
0inputs+4656outputs (0major+734236minor)pagefaults 0swaps
```

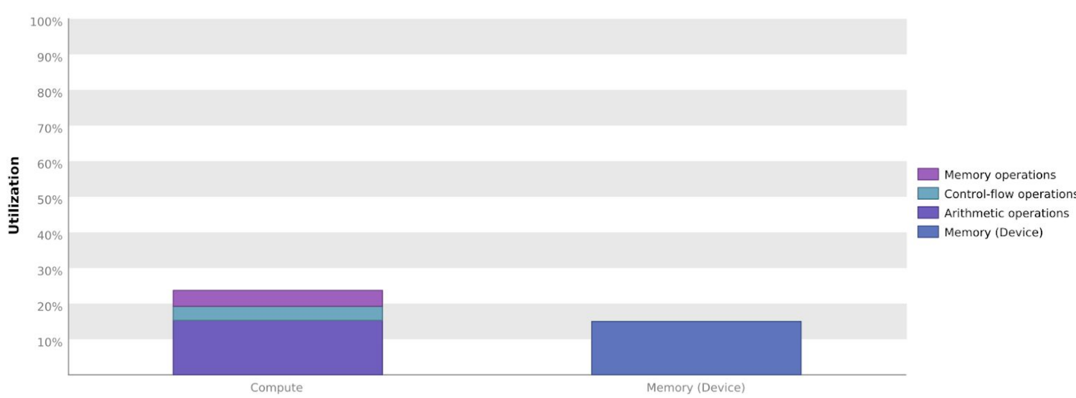
Dataset size 1000

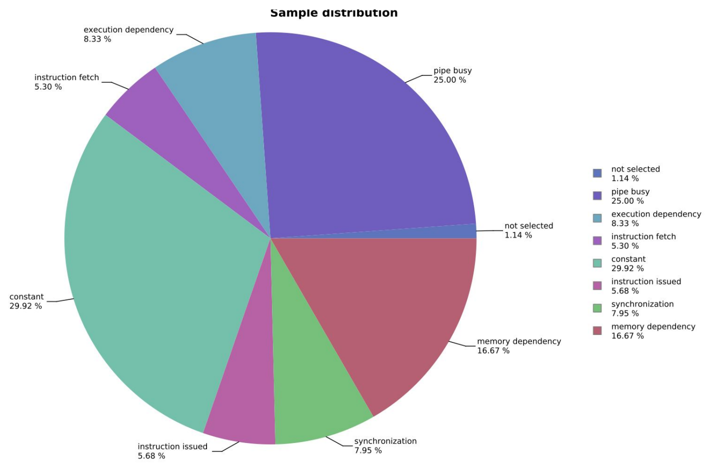
```
New Inference
Op Time: 0.015938
Op Time: 0.052050
Correctness: 0.767 Model: ece408
4.91user 3.81system 0:05.18elapsed 168%CPU (0avgtext+0avgdata 2812800maxreside
0inputs+4656outputs (0major+641920minor)pagefaults 0swaps
```

Dataset size 100

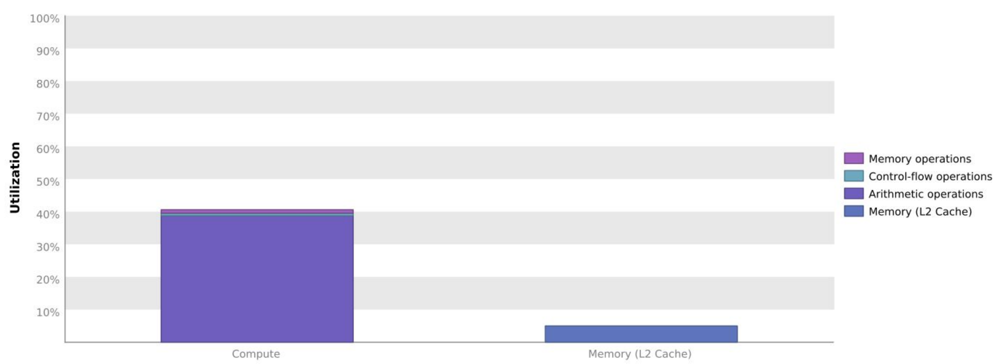
```
New Inference
Op Time: 0.002665
Op Time: 0.005774
Correctness: 0.76 Model: ece408
4.93user 3.23system 0:04.48elapsed 182%CPU (0avgtext+0avgdata
0inputs+4656outputs (0major+640768minor)pagefaults 0swaps
```

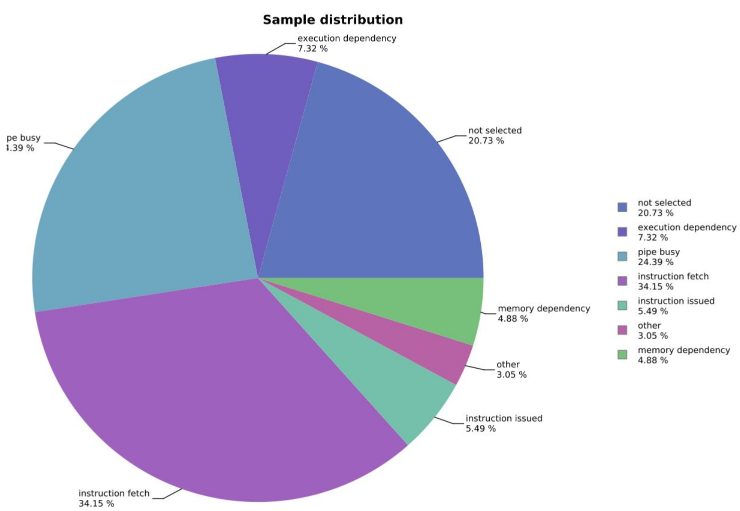
Nvprof:
Matrix Mult kernel





Unroll kernel





This optimization attempts to integrate constant memory into the kernel in an attempt to alleviate memory utilization issues/bottlenecks present in previous optimizations. As seen from optimization one, memory (device) decreased where as control-flow operations increased. According to the nvprof pie chart performance distribution, memory dependence did significantly decrease from optimization 1 despite a somewhat increase in op time.

However, bottlenecks from instruction fetch and lack of available compute resources may point to potential issues with synchronization and/or memory latency. Active warps and occupancy remained relatively the same. Our next optimization will attempt to mitigate these performance issues through integration of parallelism.

Optimization 6: 3d Parallelism

Filename: new-forward6.cuh

Op time:

Dataset 10000

```
New Inference
Op Time: 0.040691
Op Time: 0.054257
Correctness: 0.7653 Model: ece408
5.15user 3.43system 0:04.87elapsed 176%CPU (0avgtext+0avgdat
0inputs+4568outputs (0major+730153minor)pagefaults 0swaps
```

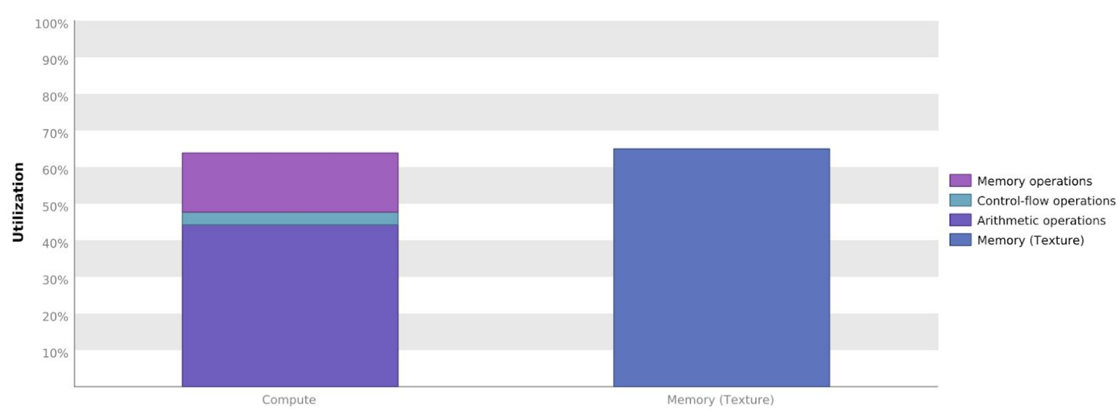
Dataset 1000

```
New Inference
Op Time: 0.004099
Op Time: 0.005466
Correctness: 0.767 Model: ece408
4.80user 3.14system 0:04.60elapsed 172%CPU (0avgtext+0avgdata
512inputs+4568outputs (2major+643931minor)pagefaults 0swaps
```

Dataset 100

```
New Inference
Op Time: 0.000442
Op Time: 0.000588
Correctness: 0.76 Model: ece408
5.06user 3.82system 0:05.14elapsed 172%CPU (0avgtext+0avgdat
0inputs+4568outputs (0major+634436minor)pagefaults 0swaps
```

Nvprof:



Shared Memory

Shared Loads	2384965567	8,303.112 GB/s	
Shared Stores	131606325	458.179 GB/s	
Shared Total	2516571892	8,761.292 GB/s	Idle Low Medium High Max

Unified Cache

Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	248803870	216.549 GB/s	
Global Stores	81720000	71.126 GB/s	
Texture Reads	2498290868	8,697.647 GB/s	
Unified Total	2828814738	8,985.322 GB/s	Idle Low Medium High Max

This optimization takes advantage of 3d parallelism. In previous optimizations, the primary aspect preventing optimal performance was memory utilization. Here, we can see a significant increase in memory utilization with the only significant limiting factor being the texture memory. This bottleneck alleviation is evident in the decreased op times.

Integrating parallelism also addresses bottlenecks from execution dependency by allowing inputs to be parallelly available instead of sequentially thereby slowing down performance. Occupancy increased to 80.7% and active warps up to 51.62. This is especially high considering the max theoretical active warps is 54 and occupancy 84.4%. This demonstrates that while optimizing by increasing occupancy through number of warps may help reach full theoretical levels, the benefits would be diminishing.

Line 51	Divergence = 3.1% [2000000 divergent executions out of 65520000 total executions]
Line 76	Divergence = 0.3% [90000 divergent executions out of 32760000 total executions]

Divergence occurs in two areas. As stated previously, changing the tile width might have to do with this divergence, but in this case the parallelism addressed issues with synchronization previously seen. As a result, performance increased significantly.

ECE 408 Milestone 4 Report

Team: sig_sev
Arkin Shah (arkinas2)
Kevin Lee (kevinl8)
Disclaimer: The week of 11/11/19, Siddarth (sa10) had to drop the course as required for a medical leave due to a serious spine injury.

Optimization 1: unroll + shared matrix multiplication

Filename: new-forward1.cuh
Op time:

Dataset Size 100

```
New Inference
Op Time: 0.001784
Op Time: 0.003025
Correctness: 0.76 Model: ece408
4.86user 3.01system 0:04.50elapsed 174%CPU (0avgtext+0avgdata 2806744maxresident)k
0inputs+4568outputs (0major+640141minor)pagefaults 0swaps
```

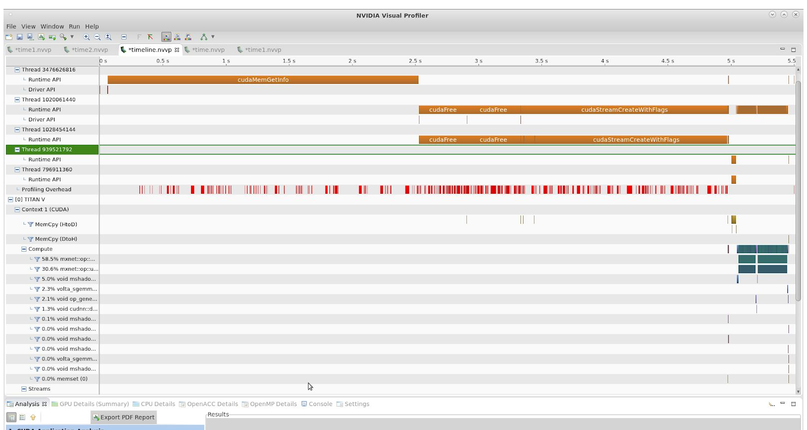
Dataset Size 1000

```
New Inference
Op Time: 0.016760
Op Time: 0.026176
Correctness: 0.767 Model: ece408
4.77user 3.00system 0:04.60elapsed 169%CPU (0avgtext+0avgdata 2802004maxresident)k
0inputs+4568outputs (0major+640426minor)pagefaults 0swaps
```

Dataset Size 10000

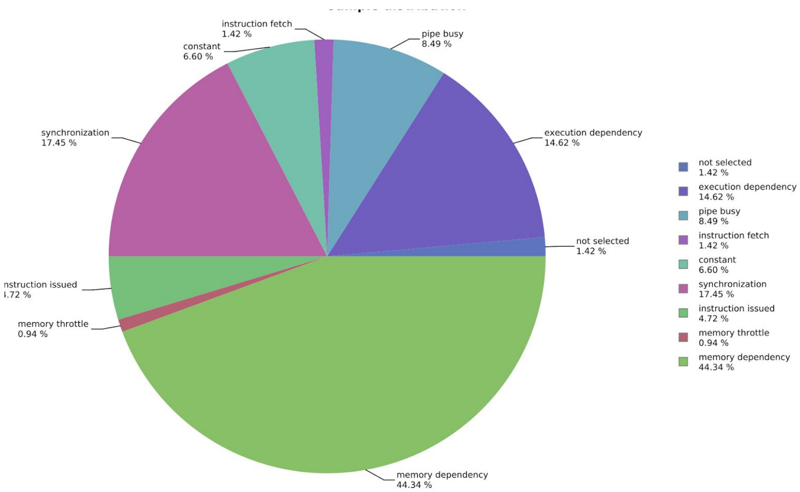
```
New Inference
Op Time: 0.106364
Op Time: 0.233283
Correctness: 0.7653 Model: ece408
5.41user 3.40system 0:05.15elapsed 171%CPU (0avgtext+0avgdata 2989996maxresident)k
0inputs+4568outputs (0major+733822minor)pagefaults 0swaps
```

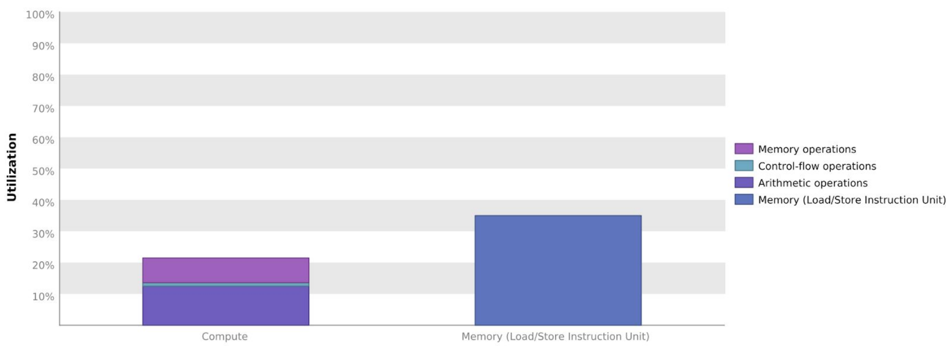
Nvprof



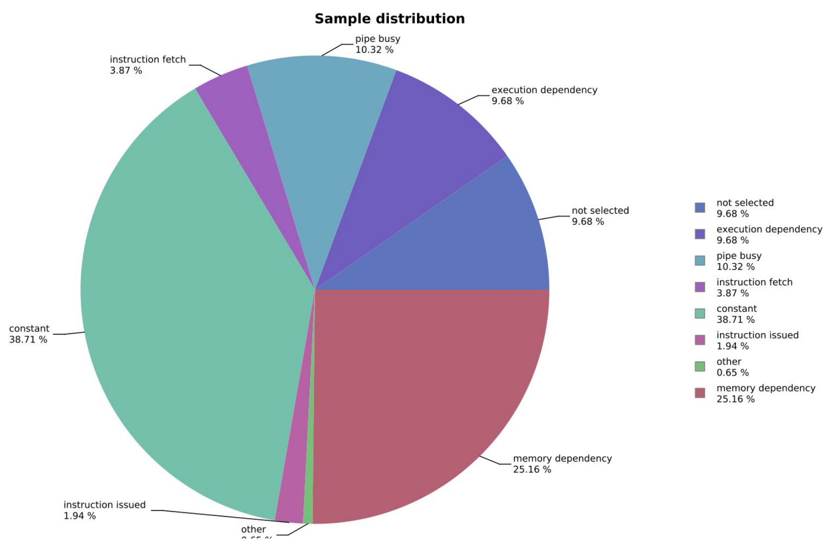
This optimization attempts to take advantage of shared memory which was not utilized at all during the MS3 original gpu implementation. Despite this, the OP times from the base implementation increased indicating other bottlenecks. This points to possibilities that either one or both kernels is limiting performance time.

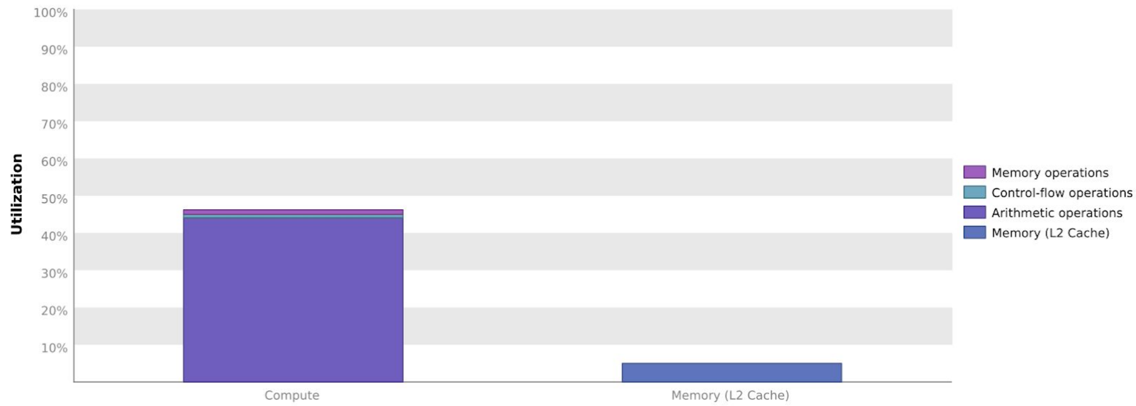
Matrix multiply kernel performance distribution (1st pass):





Unroll kernel performance distribution (1st pass):





This performance distribution for both kernels illustrates that performance is being limited by memory and constant usage. As such, both kernel's and thus performance are bounded by lack of memory utilization, both shared memory and L2 cache.

The nvprof indicates that performance is not being limited by occupancy. However, taking a closer look, the Matrix Multiply Kernel only achieves 37.9% accuracy and the unroll only 61%. This points to that while occupancy is a potential area of improvement, it is not the primary bottleneck.

Optimization 2: eliminating unrolling kernel in matrix multiplication
Filename: new-forward2.cuh
Optime:

Dataset size 10000

```
New Inference
Op Time: 0.082023
Op Time: 0.320906
Correctness: 0.7653 Model: ece408
5.29user 3.39system 0:05.17elapsed 168%CPU (0avgtext+0avgdata 2971556maxresident)k
0inputs+4568outputs (0major+729446minor)pagefaults 0swaps
```

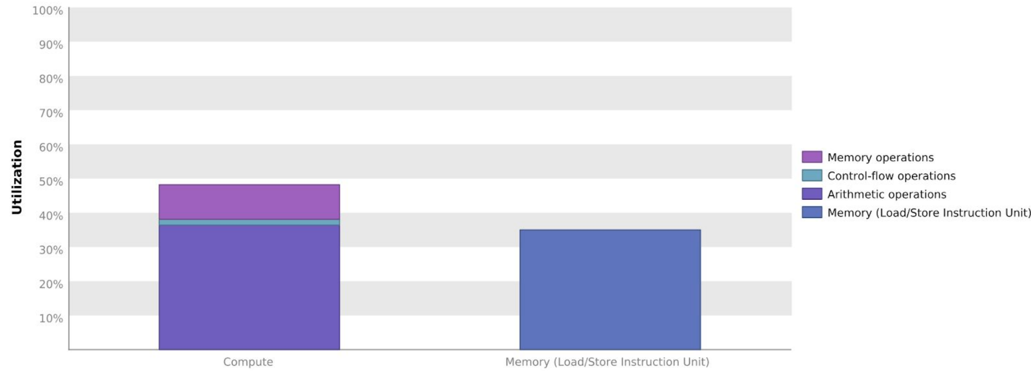
Dataset size 1000

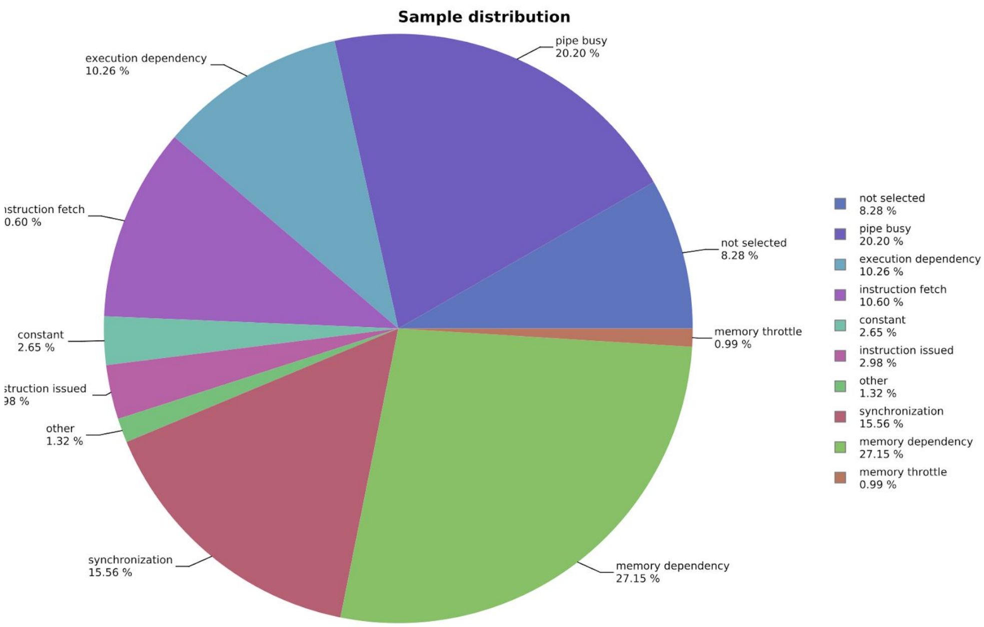
```
New Inference
Op Time: 0.008570
Op Time: 0.034922
Correctness: 0.767 Model: ece408
4.70user 3.33system 0:04.52elapsed 177%CPU (0avgtext+0avgdata 2817124maxresident)k
0inputs+4568outputs (0major+643296minor)pagefaults 0swaps
```

Dataset size 100

```
New Inference
Op Time: 0.000854
Op Time: 0.003538
Correctness: 0.76 Model: ece408
5.17user 3.09system 0:04.62elapsed 178%CPU (0avgtext+0avgdata 2807364maxresident)k
0inputs+3128outputs (0major+639821minor)pagefaults 0swaps
```

Nvprof:





This optimization eliminates the unrolling kernel thereby using only one kernel.. From the new kernel, we can see that performance, while still being limited by memory utilization, is much more dependent on other areas such as computing resource traffic/backup, blocked warps due to synchronization calls, and execution dependency.

Interestingly the occupancy from optimization 1 increased significantly to 83.3% and active warps increased to 53.32%. This suggests that fusing the kernels together may have caused some warp issues in which the increase in occupancy was not able to compensate for.

However, as shown the performance op times, this optimization did not significantly improve performance. This demonstrates that the unrolling kernel is not solely responsible as a bottleneck to performance

ECE 408 Milestone 3 Report

Team: sig_sev
Arkin Shah (arkinas2)
Kevin Lee (kevinl8)
Siddharth Agarwal (sa10)

Op Time results:

Dataset Size 100

```
New Inference
Op Time: 0.000271
Op Time: 0.000919
Correctness: 0.76 Model: ece408
4.79user 3.04system 0:04.45elapsed 176%CPU (0avgtext+0avgdata 2800684maxresident)k
```

Dataset Size 1000

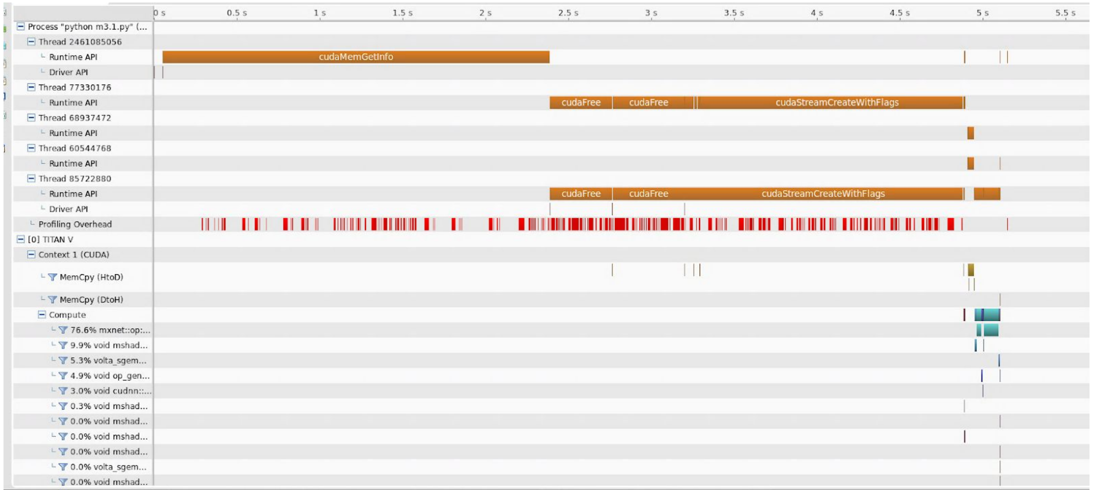
```
Loading model... done
New Inference
Op Time: 0.002951
Op Time: 0.009900
Correctness: 0.767 Model: ece408
5.15user 2.49system 0:04.64elapsed 164%CPU (0avgtext+0avgdata 2817964maxresident)k
```

Dataset Size 10000

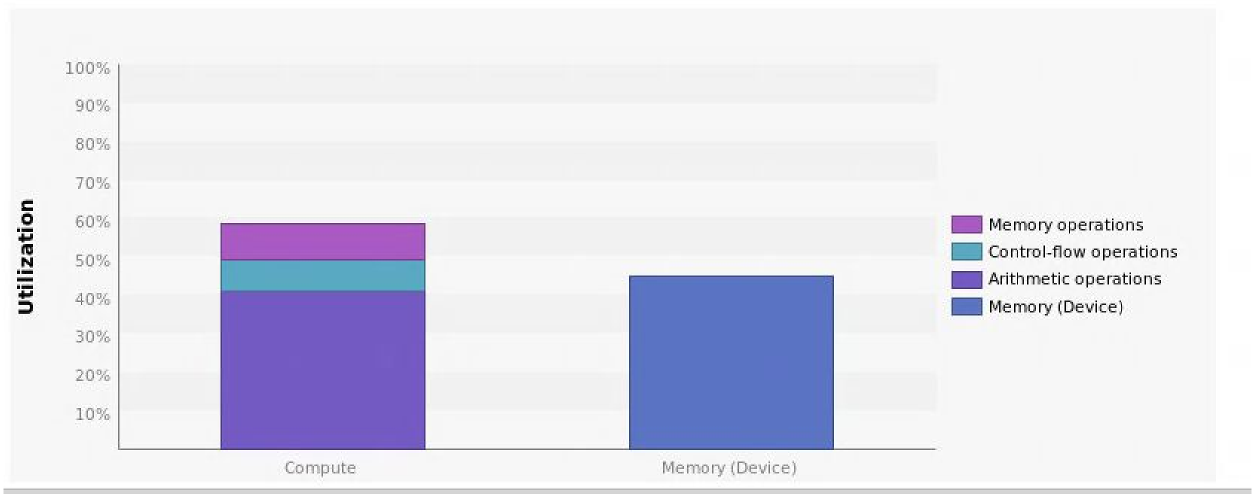
```
New Inference
Op Time: 0.028237
Op Time: 0.093702
Correctness: 0.7653 Model: ece408
4.93user 2.87system 0:04.91elapsed 159%CPU (0avgtext+0avgdata 2988660maxresident)k
```

Demonstrate nvprof profiling the execution:

Screenshot of timeline from nvprof:



Forward kernel performance:



The kernel exhibits low compute throughput and memory bandwidth utilization of under 60%. The performance is likely limited by and due to the latency of arithmetic or memory operations.

Computing resources:

Variable	Achieved	Theoretical	Device Limit	Grid Size: [10000,12,25] (3000000 blocks) Block Size: [16,16,1]
Occupancy Per SM				
Active Blocks		8	32	
Active Warps	51.95	64	64	
Active Threads		2048	2048	
Occupancy	81.2%	100%	100%	
Warps				
Threads/Block		256	1024	
Warps/Block		8	32	
Block Limit		8	32	
Registers				
Registers/Thread		32	65536	
Registers/Block		8192	65536	
Block Limit		8	32	
Shared Memory				
Shared Memory/Block		0	98304	
Block Limit		0	32	

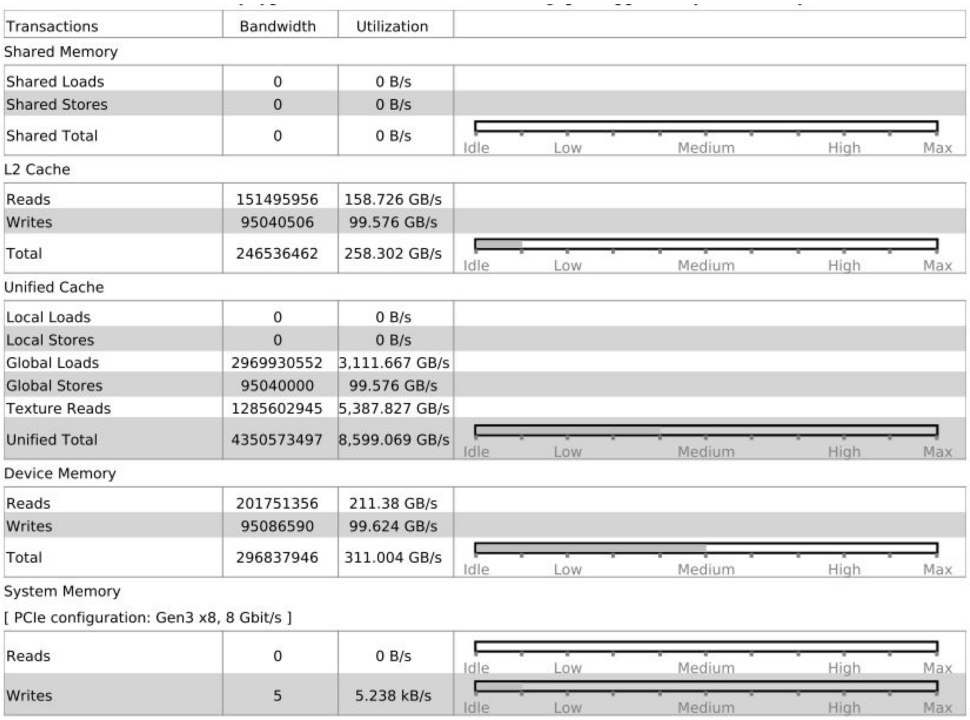
The block size, register usage, and shared memory usage in the kernel demonstrate how all warps on the GPU can be fully utilized. Hence, occupancy is not limiting the kernel's performance. Nvprof also illustrates no one function unit (e.g. load/store, texture, floating-point op., etc.) as "high utilization" meaning the kernel's performance is not limited by overuse of any function unit.

Line 43	Divergence = 16.5% [3960000 divergent executions out of 24000000 total executions]

As shown above in the nvprof, control divergence occurs at line 43. The divergence rate being 16.5%.

As a result, the warp execution efficiency of the kernel is 84.8% without predicated instructions. Taking predicated instructions into account, the efficiency decreases to 76.6% likely due to divergent branches and predicated instruction.

Memory bandwidth:



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	151495956	158.726 GB/s	
Writes	95040506	99.576 GB/s	
Total	246536462	258.302 GB/s	Idle Low Medium High Max
Unified Cache			
Local Loads	0	0 B/s	
Local Stores	0	0 B/s	
Global Loads	2969930552	3,111.667 GB/s	
Global Stores	95040000	99.576 GB/s	
Texture Reads	1285602945	5,387.827 GB/s	
Unified Total	4350573497	8,599.069 GB/s	Idle Low Medium High Max
Device Memory			
Reads	201751356	211.38 GB/s	
Writes	95086590	99.624 GB/s	
Total	296837946	311.004 GB/s	Idle Low Medium High Max
System Memory			
[PCIe configuration: Gen3 x8, 8 Gbit/s]			
Reads	0	0 B/s	Idle Low Medium High Max
Writes	5	5.238 kB/s	Idle Low Medium High Max

As can be seen from the graph, there is no utilization of shared memory. The lack of shared memory usage is an area for optimization to allow more throughput. The L2 Cache usage is also relatively low, as is the unified cache overall. Thus, memory bandwidth utilization (or lack there of) serves as a potential bottleneck.

ECE 408 Milestone 2 Report

Team : sig_sev
Arkin Shah arkinas2
Kevin Lee kevinl8
Siddharth Agarwal sa10

Include a list of all kernels that collectively consume more than 90% of the program time.

Time(%)	Time	Calls	Avg	Min	Max	Name
8.62%	9.7602ms	4	2.4401ms	2.0451ms	3.1598ms	voidfft2d_c2r_32x32<float,bool=0,bool=0,unsignedint=0,bool=0,bool=0>(float*,float2c

						onst*,int,int,int,int,int,int,int,int,int,float,float,cudnn::reduced_divisor,bool,float*,float*,int2,int,int)
6.48%	7.3306ms	2	3.6653ms	25.119us	7.3055ms	voidop_generic_tensor_kernel<int=2,float,float,float,int=256,cudnnGenericOp_t=7,cudnnNanPropagation_t=0,cudnnDimOrder_t=0,int=1>(cudnnTensorStruct,float*,cudnnTensorStruct,floatconst*,cudnnTensorStruct,floatconst*,float,float,float,float,dimArray,reducedDivisorArray)
6.42%	7.2610ms	4	1.8153ms	1.4450ms	2.2772ms	voidfft2d_r2c_32x32<float,bool=0,unsignedint=0,bool=0>(float2*,floatconst*,int,int,int,int,int,int,int,int,int,cudnn::reduced_divisor,bool,int2,int,int)
3.89%	4.4070ms	1	4.4070ms	4.4070ms	4.4070ms	voidcudnn::detail::pooling_fw_4d_kernel<float,float,cudnn::detail::maxpooling_func<float,cudnnNanPropagation_t=0>,int=0,bool=0>(cudnnTensorStruct,floatconst*,cudnn::detail::pooling_fw_4d_kernel<float,float,cudnn::detail::maxpooling_func<float,cudnnNanPropagation_t=0>,int=0,bool=0>,cudnnTensorStruct*,cudnnPoolingStruct,float,cudnnPoolingStruct,int,cudnn::reduced_divisor,float)
0.39%	440.64us	1	440.64us	440.64us	440.64us	voidmshadow::cuda::MapPlanLargeKernel<mshadow::sv::saveto,int=8,int=1024,mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,int=2,float>,float>,mshadow::expr::Plan<mshadow::expr::ScalarExp<float>,float>>(mshadow::gpu,unsignedint,mshadow::Shape<int=2>,int=2,int)
0.07%	75.135us	1	75.135us	75.135us	75.135us	voidmshadow::cuda::SoftmaxKernel<int=8,float,mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,int=2,float>,float>,mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,int=2,float>,float>>(mshadow::gpu,int=2,unsignedint)
0.06%	63.520us	13	4.8860us	1.1520us	24.384us	voidmshadow::cuda::MapPlanKernel<mshadow::sv::saveto,int=8,mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,int=2,float>,float>,mshadow::expr::Plan<mshadow::expr::ScalarExp<float>,float>>(mshadow::gpu,unsignedint,mshadow::Shape<int=2>,int=2)
0.02%	24.480us	2	12.240us	2.5280us	21.952us	voidmshadow::cuda::MapPlanKernel<mshadow::sv::plusto,int=8,mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,int=2,float>,float>,mshadow::expr::Plan<mshadow::expr::Broadcast1DExp<mshadow::Tensor<mshadow::gpu,int=1,float>,float,int=2,int=1>,float>>(mshadow::gpu,unsignedint,mshadow::Shape<int=2>,int=2)

0.01%	13.023us	1	13.023us	13.023us	13.023us	voidfft2d_r2c_32x32<float,bool=0,unsignedint=5,bool=1>(float2*,floatconst*,int,int,int,int,int,int,int,int,int,cudnn::reduced_divisor,bool,int2,int,int)
0.00%	4.6720us	1	4.6720us	4.6720us	4.6720us	voidmshadow::cuda::MapPlanKernel<mshadow::sv::saveto,int=8,mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,int=2,float>,float>,mshadow::expr::Plan<mshadow::expr::ReduceWithAxisExp<mshadow::red::maximum,mshadow::Tensor<mshadow::gpu,int=3,float>,float,int=3,bool=1,int=2>,float>>(mshadow::gpu,unsignedint,mshadow::Shape<int=2>,int=2)
0.00%	2.4000us	1	2.4000us	2.4000us	2.4000us	cudnn::gemm::computeOffsetsKernel(cudnn::gemm::ComputeOffsetsParams)

Include a list of all CUDA API calls that collectively consume more than 90% of the program time.

Time(%)	Time	Calls	Avg	Min	Max	Name
41.51%	3.10490s	22	141.13ms	14.330us	1.63066s	cudaStreamCreateWithFlags
33.02%	2.47034s	24	102.93ms	54.839us	2.46552s	cudaMemGetInfo
21.00%	1.57094s	19	82.681ms	1.2090us	423.36ms	cudaFree
1.66%	124.50ms	68	1.8309ms	6.1190us	105.27ms	cudaMalloc

Include an explanation of the difference between kernels and API calls

Kernels are user defined functions that are executed by the device (GPU). When kernel's are called, they are executed N times in parallel if there are N threads.

Cuda API calls are extensions to the C language and library functions provided by Nvidia to interact with the device. These are called in the host code and execute once when called.

Show output of rai running MXNet on the CPU

✱ Running /usr/bin/time python m1.1.py
Loading fashion-mnist data... done
Loading model... done
New Inference
EvalMetric: {'accuracy': 0.8154}

List program run time

17.02 user

4.45 system

0:08.95 elapsed

239% CPU

(0avgtext+0avgdata 6045208maxresident) k 0inputs+2824outputs

(0major+1596529minor) pagefaults

0 swaps

Show output of rai running MXNet on the GPU

✱ Running /usr/bin/time python m1.2.py

Loading fashion-mnist data... done

Loading model... done

New Inference

EvalMetric: {'accuracy': 0.8154}

List program run time

4.76 user

3.30 system

0:04.70 elapsed

171%CPU

(0avgtext+0avgdata 2958724 maxresident)k

0 inputs+4536 outputs

(0major+731874minor) pagefaults

0 swaps

List whole program execution time

88.30 user

9.93 system

1:18.37 elapsed

125 % CPU

(0avgtext+0avgdata 6043412maxresident)k

0 inputs + 2824 outputs

(0 major + 2310624 minor) pagefaults

0 swaps

List Op Times

For 10000 Input Images:

Op Time: 10.998686
Op Time: 60.145689