Milestone 1

Register your team in the google sheet.
Report: Include a list of all kernels that collectively consume more than 90% of the program time.
Report: Include a list of all CUDA API calls that collectively consume more than 90% of the program time.
Report: Include an explanation of the difference between kernels and API calls
Report: Show output of rai running MXNet on the CPU
Report: List program run time
Report: Show output of rai running MXNet on the GPU
Report: List program run time

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

1. **void fermiPlusCgemmLDS128_batched<bool=0, bool=1, bool=0, bool=0, int=4, int=4, int=4, int=3, int=3, bool=1, bool=1>(float2**, float2**, float2**, float2*, float2 const *, float2 const *, int, int, int, int, int, int, __int64, __int64, __int64, float2 const *, float2 const *, float2, float2, int)**
2. **void cudnn::detail::implicit_convolve_sgemm<float, int=1024, int=5, int=5, int=3, int=3, int=3, int=1, bool=1, bool=0, bool=1>(int, int, int, float const *, int, cudnn::detail::implicit_convolve_sgemm<float, int=1024, int=5, int=5, int=3, int=3, int=3, int=1, bool=1, bool=0, bool=1>*, float const *, kernel_conv_params, int, float, float, int, float const *, float const *, int, int)**
3. **void fft2d_c2r_32x32<float, bool=0, unsigned int=0, bool=0, bool=0>(float*, float2 const *, int, int, int, int, int, int, int, int, int, float, float, cudnn::reduced_divisor, bool, float*, float*)**
4. **Sgemm_sm35_ldg_tn_128x8x256x16x32**

5. **[CUDA memcpy HtoD]**
6. **void cudnn::detail::activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::tanh_func<float>>(cudnnTensorStruct, float const *, cudnn::detail::activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::tanh_func<float>>, cudnnTensorStruct*, float, cudnnTensorStruct*, int, cudnnTensorStruct*)**
7. **void cudnn::detail::pooling_fw_4d_kernel<float, float, cudnn::detail::maxpooling_func<float, cudnnNanPropagation_t=0>, int=0>(cudnnTensorStruct, float const *, cudnn::detail::pooling_fw_4d_kernel<float, float, cudnn::detail::maxpooling_func<float, cudnnNanPropagation_t=0>, int=0>, cudnnTensorStruct*, cudnnPoolingStruct, float, cudnnPoolingStruct, int, cudnn::reduced_divisor, float)**

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

1. **cudaStreamCreateWithFlags**
2. **cudaFree**
3. **cudaMemGetInfo**
4. **cudaMemcpy2DAsync**

Difference between kernels and API calls:

- A **kernel** is a low level program interfacing with the hardware on top of which applications are running. It is the lowest level program running on computers although with virtualization you can have multiple kernels running on top of virtual machines which themselves run on top of another operating system.
- An **API** is a generic term defining the interface developers have to use when writing code using libraries and a programming language. **Kernels have no APIs** as they are not libraries.

Show output of rai running MXNet on the CPU

Loading fashion-mnist data...

done

```
Loading model...

done

New Inference

EvalMetric: {'accuracy': 0.8444}
```

List program run time on CPU

```
 13.12user 8.31system 0:10.59elapsed 202%CPU
```

Show output of rai running MXNet on the GPU

```
   Loading fashion-mnist data...

done

Loading model...

done

New Inference

EvalMetric: {'accuracy': 0.8444}

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

Loading fashion-mnist data...

done

Loading model...

[04:33:40] src/operator/././cudnn_algoreg-inl.h:112: Running performance tests to find the
best convolution algorithm, this can take a while... (setting env variable
MXNET_CUDNN_AUTOTUNE_DEFAULT to 0 to disable)

done

New Inference
```

EvalMetric: {'accuracy': 0.8444}

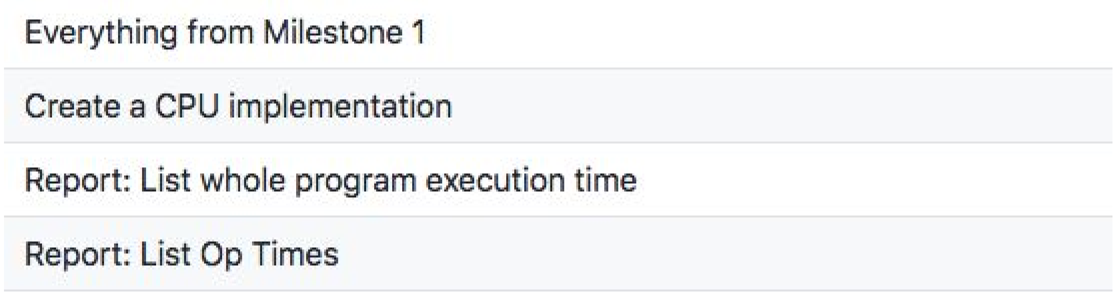
(0avgtext+0avgdata 1136388maxresident)k

0inputs+3136outputs (0major+158216minor)pagefaults 0swaps

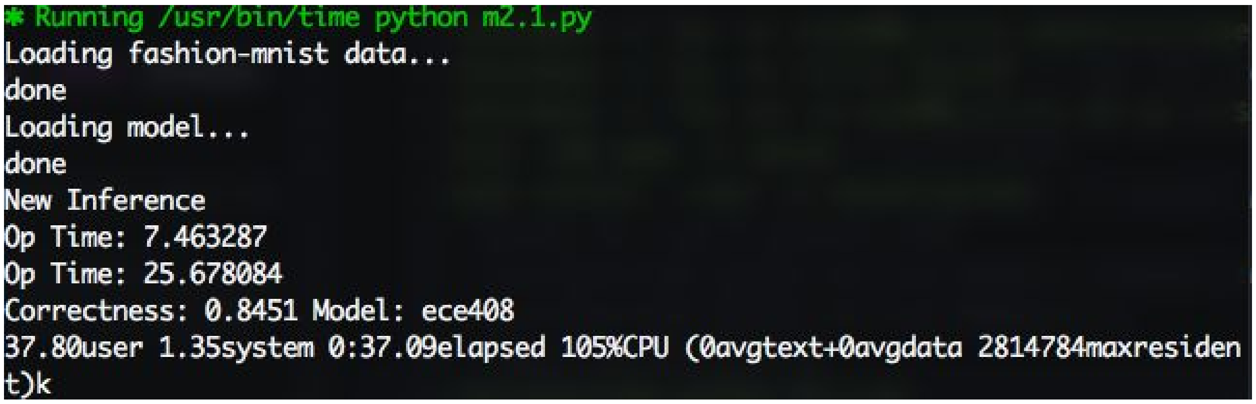
List program run time on GPU

2.27user 1.11system 0:02.84elapsed 119%CPU

Milestone 2



Everything from Milestone 1
Create a CPU implementation
Report: List whole program execution time
Report: List Op Times



```
* Running /usr/bin/time python m2.1.py
Loading fashion-mnist data...
done
Loading model...
done
New Inference
Op Time: 7.463287
Op Time: 25.678084
Correctness: 0.8451 Model: ece408
37.80user 1.35system 0:37.09elapsed 105%CPU (0avgtext+0avgdata 2814784maxresiden
t)k
```

The whole program execution time is 37.09 seconds

The first layer’s op time is 7.463287 seconds

The second layer’s op time is 25.678084 seconds