REPORT ECE408 Milestone1

Teamname: spicy-chicken
Members: Xiaocong Yu(xy21), Dawei Wang(dwang56) , Kexuan Zou(kzou3)

List of kernel calls that collectively consume more than 90% of the program time

```
        Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   40.07%  16.772ms        20  838.61us  1.1200us  16.152ms  [CUDA memcpy HtoD]
                   20.15%  8.4355ms         1  8.4355ms  8.4355ms  8.4355ms  void
cudnn::detail::implicit_convolve_sgemm<float, 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, float*, cudnn::detail::implicit_convolve_sgemm<float,
float, int=1024, int=5, int=5, int=3, int=3, int=3, int=1, bool=1, bool=0, bool=1>*, kernel_conv_params, int,
float, float, int, float, float, int, int)
                   11.82%  4.9474ms         1  4.9474ms  4.9474ms  4.9474ms  volta_cgemm_64x32_tn
                    7.04%  2.9486ms         2  1.4743ms  25.087us  2.9235ms  void
op_generic_tensor_kernel<int=2, float, float, float, int=256, cudnnGenericOp_t=7,
cudnnNanPropagation_t=0, cudnnDimOrder_t=0, int=1>(cudnnTensorStruct, float*, cudnnTensorStruct,
float const *, cudnnTensorStruct, float const *, float, float, float, float, dimArray, reducedDivisorArray)
                    5.71%  2.3909ms         1  2.3909ms  2.3909ms  2.3909ms  void fft2d_c2r_32x32<float,
bool=0, bool=0, unsigned int=1, bool=0, bool=0>(float*, float2 const *, int, int, int, int, int, int, int, int, int,
float, float, cudnn::reduced_divisor, bool, float*, float*, int2, int, int)
                    5.59%  2.3403ms         1  2.3403ms  2.3403ms  2.3403ms  volta_sgemm_128x128_tn
                    4.56%  1.9076ms         1  1.9076ms  1.9076ms  1.9076ms  void
cudnn::detail::pooling_fw_4d_kernel<float, float, cudnn::detail::maxpooling_func<float,
cudnnNanPropagation_t=0>, int=0, bool=0>(cudnnTensorStruct, float const *,
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)
                    4.21%  1.7638ms         1  1.7638ms  1.7638ms  1.7638ms  void fft2d_r2c_32x32<float,
bool=0, unsigned int=0, bool=0>(float2*, float const *, int, int, int, int, int, int, int, int, int,
cudnn::reduced_divisor, bool, int2, int, int)
```

List of API calls that collectively consume more than 90% of the program time

```
              Type  Time(%)      Time     Calls       Avg       Min       Max  Name
API calls:   41.37%  3.00690s        22  136.68ms  13.759us  1.58874s  cudaStreamCreateWithFlags
             33.85%  2.46087s        24  102.54ms  87.306us  2.45588s  cudaMemGetInfo
             21.29%  1.54779s        19  81.463ms     948ns  413.98ms  cudaFree
```

Explanation of the difference between kernels and API calls

“Summary mode is the default operating mode for nvprof. In this mode, nvprof outputs a single result line for each kernel function and each type of CUDA memory copy/set performed by the application. For each kernel, nvprof outputs the total time of all instances of the kernel or type of memory copy as well as the average, minimum, and maximum time. The time for a kernel is the kernel execution time on the device. By default, nvprof also prints a summary of all the CUDA runtime/driver API calls. Output of nvprof (except for tables) are prefixed with ==<pid>==, <pid> being the process ID of the application being profiled.” -- CUDA Toolkit Reference

From the official reference for nvprof tool, we know that the list for kernels and API are different since the API calls are mostly executed at CPU side (host code that may or may not invoke GPU), whereas the kernel calls are executed at GPU side (device code). There list some driver API calls including cudaGetDevice, cuDeviceGetName and Runtime API calls including cudaFunctionSetAttr, cudaMemsetAsync, etc.

API calls deals with collecting information for NVCC during compile time to help generate architecture and computing-ability specific executable code for CPU and GPU, and invoking kernel function on device so it will be executed much more times than kernel calls. Whereas Kernel time usage information above is collected for single kernel, so you can see the calculation functions are only called once. Time mostly comprised of kernel calculation function and cudaMemcpy from the GPU shared memory.

Output of rai running MXNet on CPU & GPU

Running /usr/bin/time python m1.1.py

```
Loading fashion-mnist data... done
Loading model... done
New Inference
EvalMetric: {'accuracy': 0.8236}
9.18user 3.70system
Time used 0:05.38 elapsed 239%CPU
(0avgtext+0avgdata2470492maxresident)k0inputs+2824outputs (0major+669491minor)pagefaults
0swaps
```

Time used for m1.1py

```
 0:05.38 elapsed 239%CPU
```

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

```
Loading fashion-mnist data... done
Loading model... done
New Inference
EvalMetric: {'accuracy': 0.8236}
4.31user 3.25system
0:04.23 elapsed
178%CPU
 (0avgtext+0avgdata2858044maxresident)k8inputs+1728outputs (0major+663219minor)pagefaults
0swaps
```

Time used for m1.2py

```
0:04.23 elapsed 178%CPU
```