ECE 408 Milestone 2 Report

Baoming Wang(baoming2), **Jun Luo**(junluo2), **Peiyuan Yang**(py6)

10/10/2019

Team Name: hmachine

On-campus

Final Submission: Due October 12, 2019

Introduction:
This report is to introduce the work for ECE 408 milestone 2. In this milestone, we installed RAI system and tried different code on RAI system. In m1.1 and m1.2, we compared and contrasted performances for the same code in GPU and CPU. In m2.1, we made a convolution code and tested its performance for different layer numbers (100, 1000, 10000).

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

	Time(%)	**Name**
1	**31.72%**	**[CUDA memcpy HtoD]**
2	**17.56%**	**volta_scudnn_128x64_relu_interior_nn_v1**
3	**16.98%**	**volta_gcgemm_64x32_nt**
4	**8.63%**	**void fft2d_c2r_32x32**
5	**7.71%**	**volta_sgemm_128x128_tn**
6	**6.49%**	**void op_generic_tensor_kernel**
7	**6.40%**	**void fft2d_r2c_32x32**
Total	**95.49%**	

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

	Time(%)	Name
1	41.83%	cudaStreamCreateWithFlags
2	33.40%	cudaMemGetInfo
3	21.02%	cudaFree
Total	96.25%	

3. **Include an explanation of the difference between kernels and API calls.**
   The API calls are all about memory set and control for the kernel variables. The kernels are the parallel portions of an application are executed on the device. The API calls only happen once, but the kernel calls are made by different threads in parallel.

   For the two different sections generated by *nvprof*:

   "GPU activities" is the timing information that represents the execution time on the **GPU.**

   "API calls" section is the timing that represents the execution time on the **host**.

4. **Show output of rai running MXNet on the CPU**
   Terminal output:

```
Loading fashion-mnist data... done
Loading model... done
New Inference
EvalMetric: {'accuracy': 0.8154}
17.02user 4.48system 0:08.85elapsed 242%CPU (0avgtext+0avgdata
6046980maxresident)k
0inputs+2824outputs (0major+
1601620minor)pagefaults 0swaps
```

5. **List program run time(CPU Version)**
   17.02user 4.48system 0:08.85elapsed

6. **Show output of rai running MXNet on the GPU**
   Terminal Output:

```
Loading fashion-mnist data... done
Loading model... done
New Inference
EvalMetric: {'accuracy': 0.8154}
5.14user 3.22system 0:04.72elapsed 177%CPU (0avgtext+0av
gdata 2981512maxresident)k
0inputs+1712outputs (0major+733933minor)pagefaults 0swaps
```

7. **List program run time(GPU Version)**
   5.14user 3.22system 0:04.72elapsed

8. **Create a CPU implementation**
   CUDA Code:

```
template <typename cpu, typename DType>
void forward(mshadow::Tensor<cpu, 4, DType> &y, const
mshadow::Tensor<cpu, 4, DType> &x, const mshadow::Tensor<cpu, 4,
DType> &k)
{
    /*
    Modify this function to implement the forward pass described in
Chapter 16.
    The code in 16 is for a single image.
    We have added an additional dimension to the tensors to support
an entire mini-batch
```

```
    The goal here is to be correct, not fast (this is the CPU
implementation.)
    */

    const int B = x.shape_[0];
    const int M = y.shape_[1];
    const int C = x.shape_[1];
    const int H = x.shape_[2];
    const int W = x.shape_[3];
    const int K = k.shape_[3];
    const int H_out = H-K+1;
    const int W_out = W-K+1;

    for (int b = 0; b < B; ++b) {
      for (int m=0; m < M; ++m) {
        for (int h=0; h < H_out; ++h) {
          for (int w=0; w < W_out; ++w) {
            y[b][m][h][w] = 0;
            for (int c=0; c < C; ++c) {
              for (int p=0; p < K; ++p) {
                for (int q=0; q < K; ++q) {
                  y[b][m][h][w] += x[b][c][h + p][w + q] *
k[m][c][p][q];
                }
              }
            }
          }
        }
      }
    }
}
```

9. **List whole program execution time**

   10000: 82.49user 8.20system 1:13.41elapsed
   1000: 12.15user 3.02system 0:08.42elapsed
   100: 4.64user 2.34system 0:01.96elapsed

10. List Op Times

10000: Op Time: 10.848974 Op Time: 59.013833
1000: Op Time: 1.081818 Op Time: 5.914613
100: Op Time: 0.117280 Op Time: 0.621784

Conclusion:
From this project milestone, we learned the differences between CPU and GPU while running the CUDA code. Furthermore, we had a deep understanding of the profit of GPU in CV calculation and the function of CUDA for convolution layers.