CUDA Programming Assignment 4 Bonus

Sree Charan Gundabolu, email:sgundabo@purdue.edu, github:181charan

May 14, 2021

Ambitious

I started this off as a really ambitious project! I wanted to extract the actual weights and biases from AlexNet and get it working. I spent a lot of time on figuring out how to extract the weights and biases from a pretrained model. I did, figure it out eventually but considering that we needed to show performance across batches I decided to drop this. Though, I hope to get back to this someday. Might come in handy. The following code snippet shows how I could get the weights for cov3 layer, this can be used to extract weights and biases from other layers too!

```
import numpy as np
import tensorflow as tf
import pandas as pd

!wget http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/bvlc_alexnet.npy
net_data = np.load(open("bvlc_alexnet.npy", "rb"), allow_pickle=True ,encoding="latin1").item()

keys = net_data.keys();
for k in keys:
  weights = (net_data[k][0])
  bias = (net_data[k][1])
  weights_f = (net_data[k][0]).flatten()
  bias_f = (net_data[k][1]).flatten()
  failTest = 0
  print(str(weights.shape)+" "+str(weights_f.shape))
  if weights.ndim == 4:
    print(str(weights.shape[0])+" "+str(weights.shape[1])+" "+str(weights.shape[2])+" "+str(weights.shape[3]))
    if k == 'conv3':
      weights_rearranged = np.empty((384,256,3,3))

    for x in range(weights.shape[0]):
      for y in range(weights.shape[1]):
        for d in range(weights.shape[2]):
          for c in range(weights.shape[3]):
            idx = x*(weights.shape[1]*weights.shape[2]*weights.shape[3]) + y*(weights.shape[2]*weights.shape[3]) +
                  d*weights.shape[3]+c
            temp = weights_f[idx]
            temp1 = weights[x][y][d][c]
            if k == 'conv3':
              weights_rearranged[c][d][x][y] = temp1
            if (temp != temp1):
              failTest = 1
    if(failTest == 1):
      print("Could not translate!")
  elif weights.ndim == 2:
    test = 2
  else:
    print("Dim error!")

weights = (net_data['conv3'][0])
new_flatten = weights_rearranged.flatten()
```

AlexNet

I must admit this was a very lazy implementation, there is a lot of other things I could have done to optimize. Especially I could have done without a lot of copying between device and host! Could have copied the weights and filter values to the texture/constant cache because they are read-only. I was so preoccupied with the simulation assignments that I genuinely could only get to doing this. However, I have tested my implementation for various batch sizes. There is almost a linear increase in execution time with increase in batch size. This can be attributed to the fact that the kernels have to compute more, but the copies between the host and the device are larger!

I finally took the advice on Piazza and got rid of those cold starts showing up as a large portion of the execution time!

It is interesting that freeing up host memory seems to be taking up a significant chunk of the total execution time irrespective of the batch size.

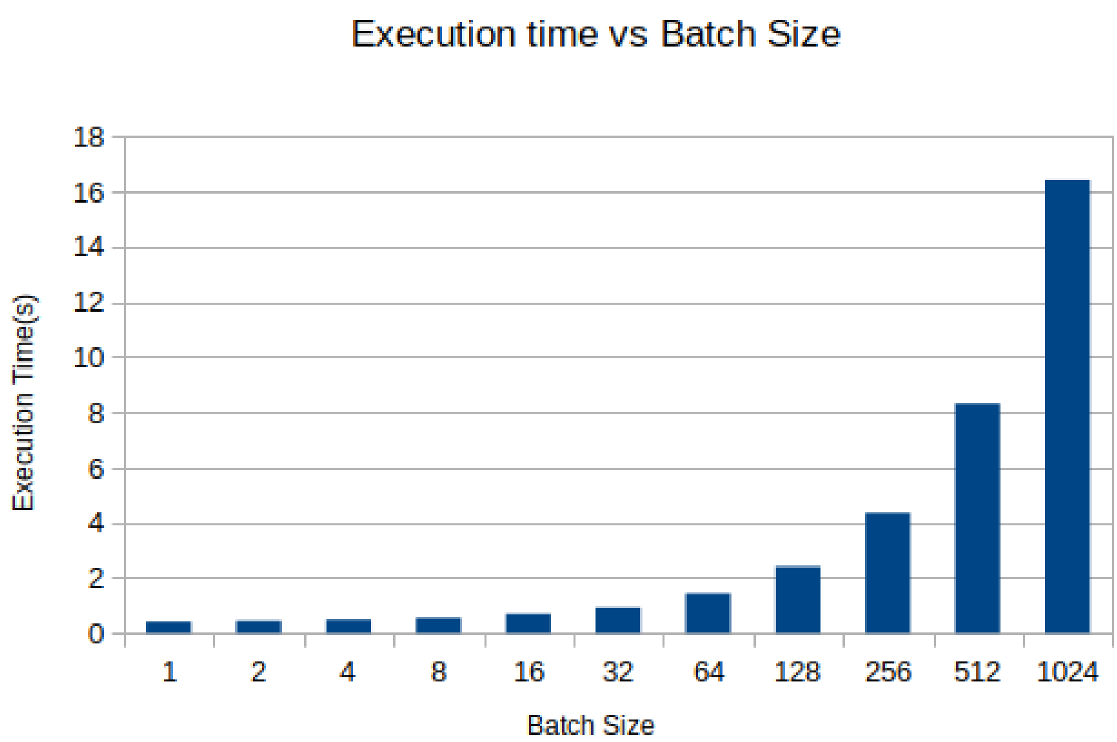

Figure 1: Total Execution time(s)

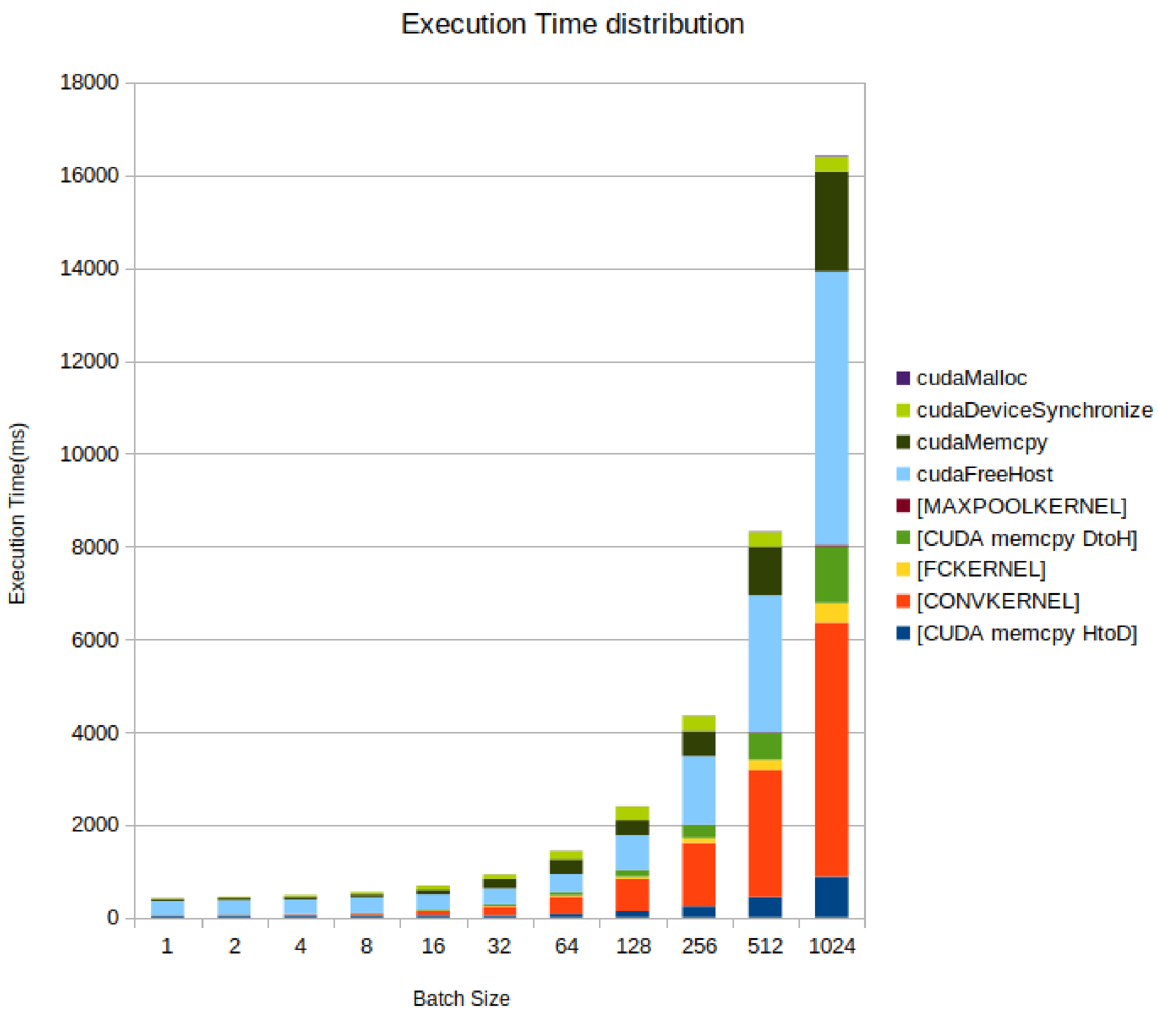

Figure 2: Execution Time Distribution

BatchSize	[CUDA memcpy HtoD]	[CONVKERNEL]	[FCKERNEL]	[CUDA memcpy DtoH]	[MAXPOOLKERNEL]	cudaFreeHost	cudaMemcpy	cudaDeviceSynchronize	cudaMalloc
1	40.16	7.6397	2.7477	0.24595	0.048095	307.19	44.235	10.408	2.314
2	38.947	13.349	2.7436	0.74591	0.074752	321.65	45.747	16.109	2.5634
4	40.703	25.019	4.7973	3.2918	0.13715	322.03	52.089	29.834	2.8689
8	43.317	48.832	5.3163	6.2785	0.27382	330.2	60.328	54.168	3.2172
16	49.861	95.92	9.2722	17.823	0.52975	322.12	105.2	79.179	3.9898
32	42.397	189.94	14.908	36.528	1.045	344.41	204.85	90.749	4.5022
64	88.002	361.16	26.611	72.042	2.105	387.78	314.26	172.66	4.7443
128	136.95	694.48	51.468	140.65	4.1318	745.99	327.09	293.67	5.2168
256	236.26	1372.56	102.95	283.28	8.3304	1475.53	539.2	325.95	6.4154
512	438.3	2741.51	218.74	573.3	17.305	2960.26	1040.36	315.72	9.1404
1024	883.58	5469.98	425.86	1215.64	35.604	5896.08	2148.71	322.61	18.422

Figure 3: Data