COMP-5011-FDE Machine Learning & Neural Network

Assignment 4

Name: Khushal Paresh Thaker

Student ID: 1106937

Part 1: Cuda Code (Improved performance by using *multi streams* when compared to previous code)

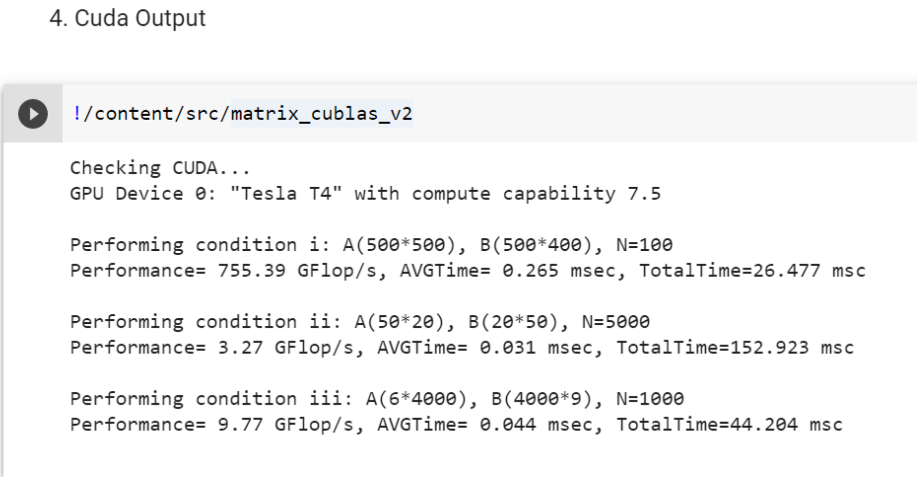
Output:

Performing condition i: A(500*500), B(500*400), N=100
Performance= 755.39 GFlop/s, AVGTime= 0.265 msec, TotalTime=26.477 msc

Performing condition ii: A(50*20), B(20*50), N=5000
Performance= 3.27 GFlop/s, AVGTime= 0.031 msec, TotalTime=152.923 msc

Performing condition iii: A(6*4000), B(4000*9), N=1000
Performance= 9.77 GFlop/s, AVGTime= 0.044 msec, TotalTime=44.204 msc

Screenshot:



4. Cuda Output

```
!/content/src/matrix_cublas_v2

Checking CUDA...
GPU Device 0: "Tesla T4" with compute capability 7.5

Performing condition i: A(500*500), B(500*400), N=100
Performance= 755.39 GFlop/s, AVGTime= 0.265 msec, TotalTime=26.477 msc

Performing condition ii: A(50*20), B(20*50), N=5000
Performance= 3.27 GFlop/s, AVGTime= 0.031 msec, TotalTime=152.923 msc

Performing condition iii: A(6*4000), B(4000*9), N=1000
Performance= 9.77 GFlop/s, AVGTime= 0.044 msec, TotalTime=44.204 msc
```

Analysing the code and comparing with Assignment 3 part 2 code,

We see significant improvement in all three conditions.

Runtime for previous code (Assignment 3 Part 2)

Cuda Runtime for 1st condition: 852.18 msc
Cuda Runtime for 2nd condition: 276.462 msc
Cuda Runtime for 3rd condition: 518.184 msc

This improvement in performance is due to usage of **multi streams**.

Part 2 Python Code for Assignment 3 Part 2

Output:

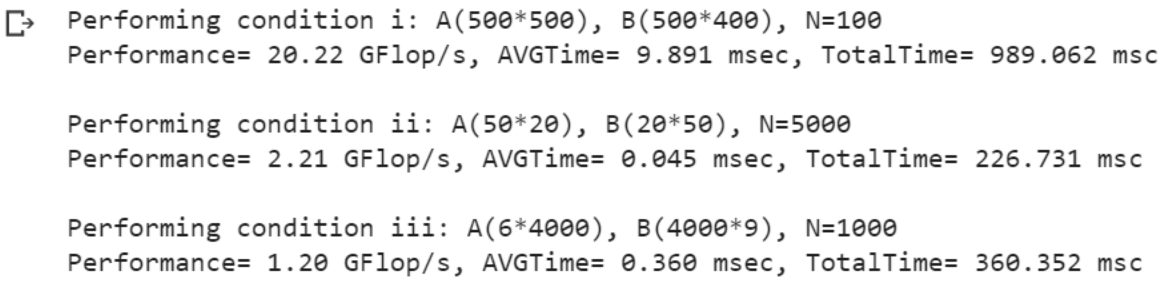
Output with threads,

Performing condition i: A(500*500), B(500*400), N=100
Performance= 20.22 GFlop/s, AVGTime= 9.891 msec, TotalTime= 989.062 msc

Performing condition ii: A(50*20), B(20*50), N=5000
Performance= 2.21 GFlop/s, AVGTime= 0.045 msec, TotalTime= 226.731 msc

Performing condition iii: A(6*4000), B(4000*9), N=1000
Performance= 1.20 GFlop/s, AVGTime= 0.360 msec, TotalTime= 360.352 msc

Screenshot:



```
Performing condition i: A(500*500), B(500*400), N=100
Performance= 20.22 GFlop/s, AVGTime= 9.891 msec, TotalTime= 989.062 msc

Performing condition ii: A(50*20), B(20*50), N=5000
Performance= 2.21 GFlop/s, AVGTime= 0.045 msec, TotalTime= 226.731 msc

Performing condition iii: A(6*4000), B(4000*9), N=1000
Performance= 1.20 GFlop/s, AVGTime= 0.360 msec, TotalTime= 360.352 msc
```

Comparing the performance of Cuda and python Code:

Cuda Runtime for 1st condition was 852.18 msc while with the python code was 989.062 msc. Average run time for each matrix is also higher for the python code at 9.891 msc where as it was 8.522 msc with the Cuda code. Similarly, the 2nd and 3rd condition for Cuda is better than python version.

Part 3 - Python code for Assignment 4,

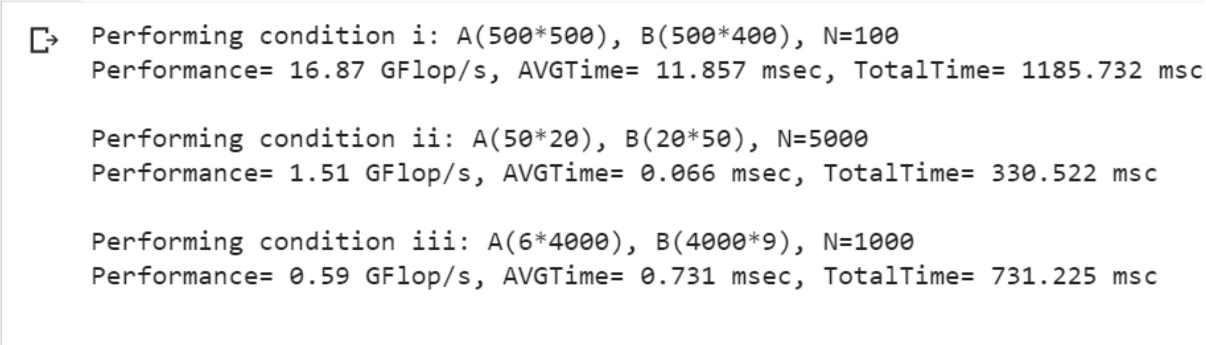
Output:

Performing condition i: A(500*500), B(500*400), N=100
Performance= 16.87 GFlop/s, AVGTime= 11.857 msec, TotalTime= 1185.732 msc

Performing condition ii: A(50*20), B(20*50), N=5000
Performance= 1.51 GFlop/s, AVGTime= 0.066 msec, TotalTime= 330.522 msc

Performing condition iii: A(6*4000), B(4000*9), N=1000
Performance= 0.59 GFlop/s, AVGTime= 0.731 msec, TotalTime= 731.225 msc

Screenshot:



```
Performing condition i: A(500*500), B(500*400), N=100
Performance= 16.87 GFlop/s, AVGTime= 11.857 msec, TotalTime= 1185.732 msc

Performing condition ii: A(50*20), B(20*50), N=5000
Performance= 1.51 GFlop/s, AVGTime= 0.066 msec, TotalTime= 330.522 msc

Performing condition iii: A(6*4000), B(4000*9), N=1000
Performance= 0.59 GFlop/s, AVGTime= 0.731 msec, TotalTime= 731.225 msc
```

Comparing it with the Cuda output,

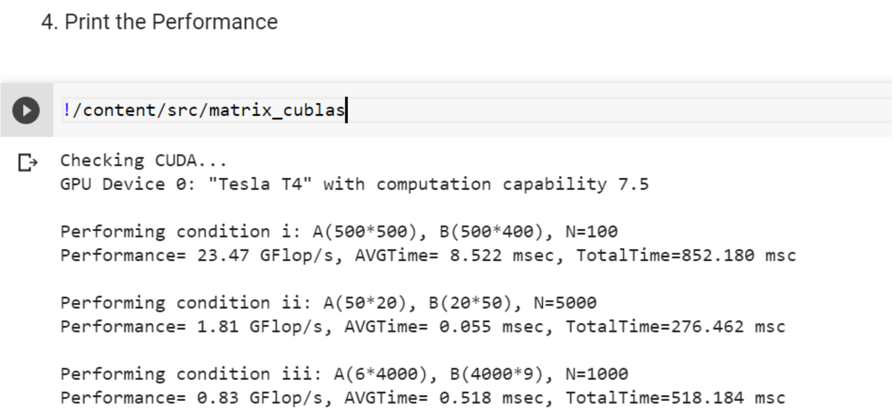
For the first condition, runtime with cuda code was 26.77 msc whereas with the python version is 1185.732 msc. Also, the average runtime for each matrix multiplying is higher for python code, at 11.857 msc for the first condition, whereas for the cuda code it is merely 0.26 msc. This is due to the enhanced performance boosted by using multi streams. Similarly, the 2nd and 3rd condition for Cuda is better than python version.

Cuda performance comparison

Cuda code for Assignment 4 has improved performance because of multi streams showcased by reduced runtime for each condition.

Screenshot for validation,

Assignment 3 Part 2,



4. Print the Performance

```
!/content/src/matrix_cublas
```

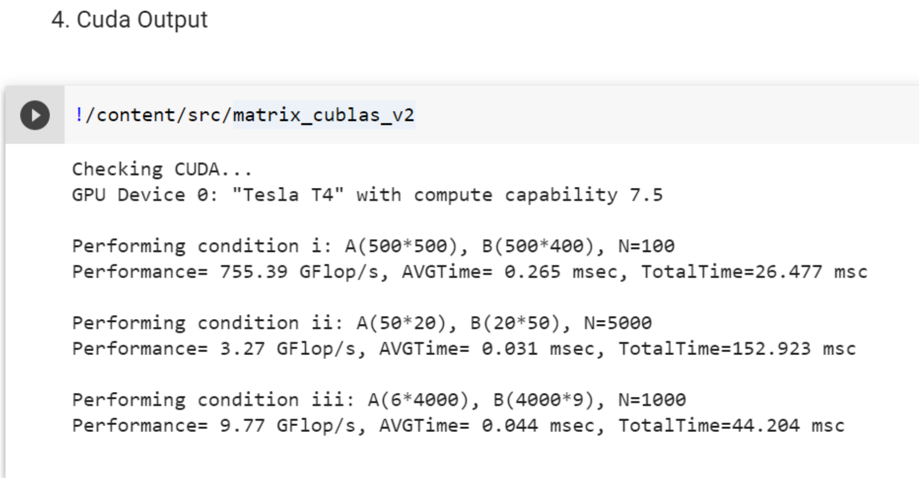
```
Checking CUDA...
GPU Device 0: "Tesla T4" with computation capability 7.5

Performing condition i: A(500*500), B(500*400), N=100
Performance= 23.47 GFlop/s, AVGTime= 8.522 msec, TotalTime=852.180 msc

Performing condition ii: A(50*20), B(20*50), N=5000
Performance= 1.81 GFlop/s, AVGTime= 0.055 msec, TotalTime=276.462 msc

Performing condition iii: A(6*4000), B(4000*9), N=1000
Performance= 0.83 GFlop/s, AVGTime= 0.518 msec, TotalTime=518.184 msc
```

Assignment 4,



4. Cuda Output

```
!/content/src/matrix_cublas_v2
```

```
Checking CUDA...
GPU Device 0: "Tesla T4" with compute capability 7.5

Performing condition i: A(500*500), B(500*400), N=100
Performance= 755.39 GFlop/s, AVGTime= 0.265 msec, TotalTime=26.477 msc

Performing condition ii: A(50*20), B(20*50), N=5000
Performance= 3.27 GFlop/s, AVGTime= 0.031 msec, TotalTime=152.923 msc

Performing condition iii: A(6*4000), B(4000*9), N=1000
Performance= 9.77 GFlop/s, AVGTime= 0.044 msec, TotalTime=44.204 msc
```

Tabulated Comparison

	A(500*500), B(500*400), N = 100	A(50*20), B(20*50), N = 5000	A(6*4000), B(4000*9), N = 1000
CUDA C (Assignment 3 Part 2)	852.180 msc	276.462 msc	518.184 msc
CUDA C (Assignment 4)	26.477 msc	152.923 msc	44.204 msc
Python (Assignment 3 Part 2)	989.062 msc	226.731 msc	360.352 msc
Python (Assignment 4)	1185.732 msc	330.522 msc	731.225 msc

Cuda C code for Assignment 4 outperforms all the other codes because of the multi stream used to code it.

I also developed the python code for the same without using the concept of threading, just a basic code, and that produced results like

		A(500*500), B(500*400), N = 100	A(50*20), B(20*50), N = 5000	A(6*4000), B(4000*9), N = 1000
With Threading	Python (Assignment 3 Part 2)	989.062 msc	226.731 msc	360.352 msc
	Python (Assignment 4)	1185.732 msc	330.522 msc	731.225 msc
Without Threading	Python (Assignment 3 Part 2)	897.718 msc	135.049 msc	261.683 msc
	Python (Assignment 4)	1135.036 msc	182. 493 msc	555.731 msc

We can notice that the code without threading performs faster which could be because of the matrix size.