

LP1 Assignment HPC 2

Date: 2nd September, 2020

Title: Parallel Computing using CUDA

Problem Definition:

Vector and Matrix Operations

Design parallel algorithm to

1. Add two large vectors
2. Multiply Vector and Matrix
3. Multiply two $N \times N$ arrays using n^2 processors

Learning Objectives:

- Learn Parallel Decomposition of Problem
- Learn Parallel Computing using CUDA

Learning Outcomes:

- I will be able to perform parallel computing using CUDA libraries
- I will be able to decompose a problem into sub problems, learn how to use GPUs, and solve problems using threads on GPU

S/W & H/W Packages:

1. Operating System : 64-bit Open source Linux or its derivative
2. Programming Language: C/C++
3. NVidia CUDA-enabled GPU
4. CUDA API
5. Google Colaboratory(uses Tesla K80 GPU)

Related Mathematics:

Let S be the system set:

$$S = \{s; e; X; Y; F_{me}; DD; NDD; F_c; S_c\}$$

s=start state

e=end state

X=set of inputs

$$X = \{X_1\}$$

where

X_1 = Elements of Vector

Y= Output Set (Sum or Product of elements of Vector/Matrix)

F_{me} is the set of main functions

$$F_{me} = \{f_1, f_2, f_3\}$$

where

f₁ = decomposition function

f₂ = function to find Sum / Product

f₃ = function to merge results

DD= Deterministic Data Vector / Matrix of Elements.

NDD=Non-deterministic data No non deterministic data

F_c =failure case: No failure case identified for this application

Concepts Related to Theory:

Dividing a computation into smaller computations and assigning them to different processors for parallel execution are the two key steps in the design of parallel algorithms. The process of dividing a computation into smaller parts, some or all of which may potentially be executed in parallel, is called decomposition.

Tasks are programmer-defined units of computation into which the main computation is subdivided by means of decomposition. Simultaneous execution of multiple tasks is the key to reducing the time required to solve the entire problem.

Tasks can be of arbitrary size, but once defined, they are regarded as indivisible units of computation. The tasks into which a problem is decomposed may not all be of the same size. In addition of two vectors, we have to add i th element from first array with i th element of second array to get i th element of resultant array.

We can allocate this each addition to distinct thread. Same thing can be done for the product of two vectors. There can be three cases for addition of two vectors using CUDA.

1. n blocks and one thread per block.
2. 1 block and n threads in that block.
3. m blocks and n threads per block.

In addition of two matrices, we have to add (i,j) th element from first matrix with (i,j) th element of second matrix to get (i,j) th element of resultant matrix.. We can allocate this each addition to distinct thread. There can be two cases for addition of two matrices using CUDA.

1. Two dimensional blocks and one thread per block.
2. One block and two dimensional threads in that block. Same cases can be considered for multiplication of two matrices.

References:

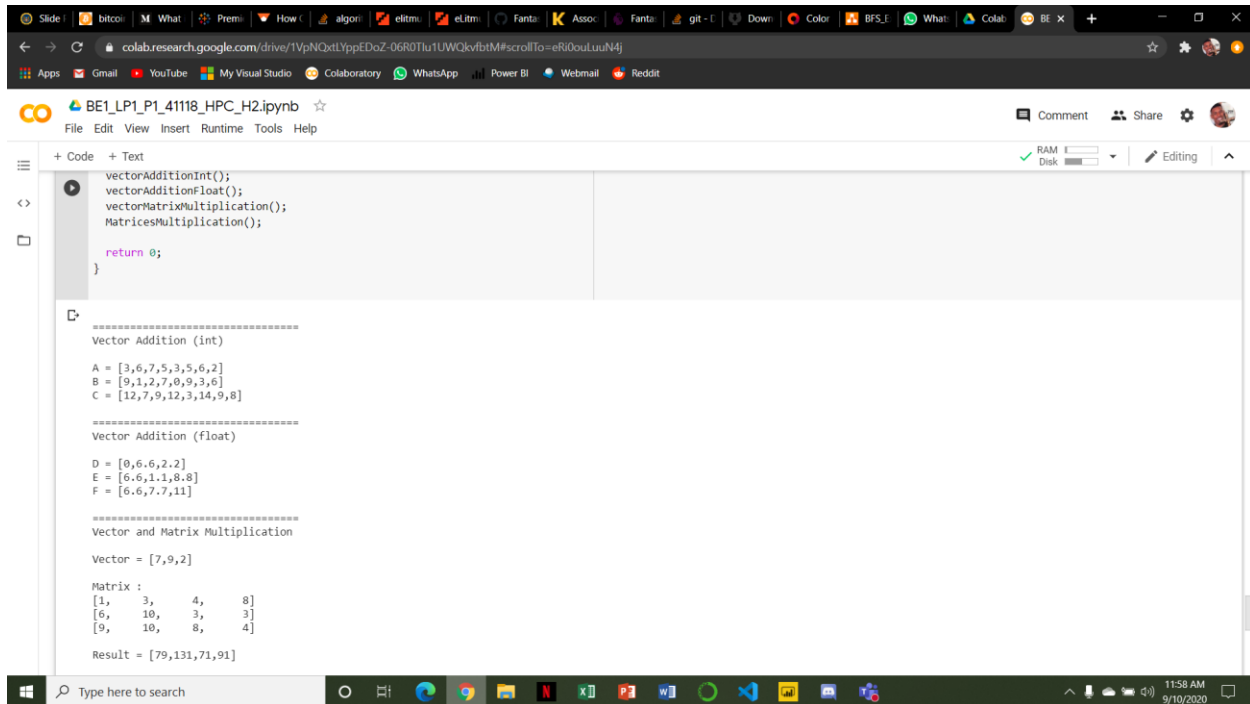
1. CUDA Documentation: <https://docs.nvidia.com/cuda/>

Steps for Execution:

1. Open Google Colaboratory
2. Change Runtime type from CPU to GPU
3. Get nvcc plugin for Jupyter Notebook
4. Start codes with %%cu

5. Compute sum, maximum, minimum, mean and standard deviation of a vector

Output:



The screenshot shows a Google Colab notebook titled "BE1_LP1_P1_41118_HPC_H2.ipynb". The code cell contains the following Python code:

```
vectorAdditionInt();
vectorAdditionFloat();
vectorMatrixMultiplication();
MatricesMultiplication();

return 0;
}
```

The output of the code is displayed below the code cell:

```
=====
Vector Addition (int)

A = [3,6,7,5,3,5,6,2]
B = [9,1,2,7,0,9,3,6]
C = [12,7,9,12,3,14,9,8]

=====
Vector Addition (float)

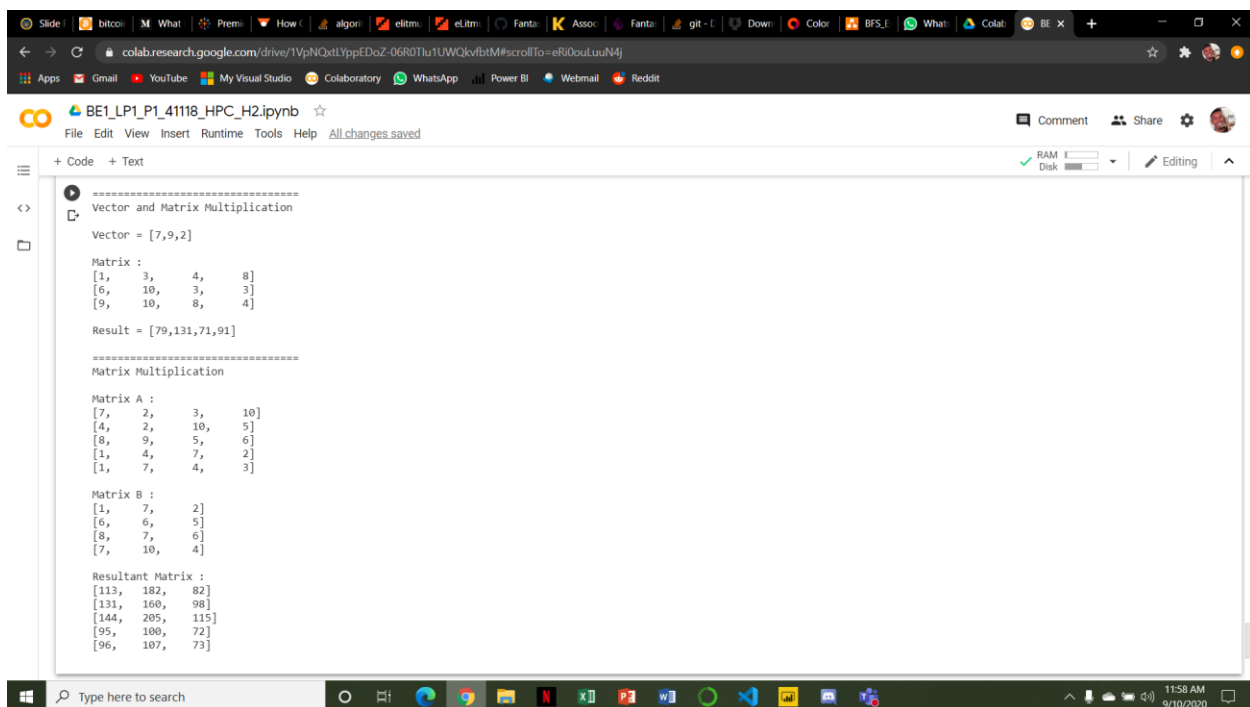
D = [0,6,6,2,2]
E = [6,6,1,1,8,8]
F = [6,6,7,7,11]

=====
Vector and Matrix Multiplication

Vector = [7,9,2]

Matrix :
[1, 3, 4, 8]
[6, 10, 3, 3]
[9, 10, 8, 4]

Result = [79,131,71,91]
```



The screenshot shows the same Google Colab notebook, but with the second code cell selected. The code cell contains the following Python code:

```
=====
Vector and Matrix Multiplication

Vector = [7,9,2]

Matrix :
[1, 3, 4, 8]
[6, 10, 3, 3]
[9, 10, 8, 4]

Result = [79,131,71,91]

=====
Matrix Multiplication

Matrix A :
[7, 2, 3, 10]
[4, 2, 10, 5]
[8, 9, 5, 6]
[1, 4, 7, 2]
[1, 7, 4, 3]

Matrix B :
[1, 7, 2]
[6, 6, 5]
[8, 7, 6]
[7, 10, 4]

Resultant Matrix :
[113, 182, 82]
[131, 160, 98]
[144, 205, 115]
[95, 100, 72]
[96, 107, 73]
```

Notebook Link:

<https://colab.research.google.com/drive/1VpNQxtLYppEDoZ-06R0Tlu1UWQkvfbtM?usp=sharing>

Conclusion:

I have successfully performed parallel computing using CUDA libraries and computed the addition of two vectors, multiplication of a vector with a matrix and matrix multiplication.