Assignment 2

Nachiket Erlekar,41434

August 21, 2020

1 Title

Parallel Computing Using CUDA

2 Objective

- Learn parallel decomposition of problem.
- Learn parallel computing using CUDA.

3 Problem Statement

Vector and Matrix Operations-
Design parallel algorithm to

- Add two large vectors
- Multiply Vector and Matrix
- Multiply two N N arrays using n2 processors

4 Outcome

We will be able to decompose problem into sub problems, to learn how to use GPUs, to learn to solve sub problem using threads on GPU cores.

5 Software and Hardware requirements

1. Operating System : 64-bit Linux or its derivative
2. Programming Language: C/C++
3. NVidia GPU
4. CUDA API

6 Date of Completion

August 21, 2020

7 Assessment grade/marks and assessor's sign

8 Theory- Concept in brief

The CUDA platform is designed to work with programming languages such as C, C++, and Fortran. This accessibility makes it easier for specialists in parallel programming to use GPU resources, in contrast to prior APIs like Direct3D and OpenGL, which required advanced skills in graphics programming.[3] CUDA-powered GPUs also support programming frameworks such as OpenACC and OpenCL;[4][2] and HIP by compiling such code to CUDA. When CUDA was first introduced by Nvidia, the name was an acronym for Compute Unified Device Architecture,[5] but Nvidia subsequently dropped the common use of the acronym

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 compu- tation 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 compu- tation. 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 ith element from first array with ith element of second array to get ith element of resultant array. We can allocate this each addition to distinct thread. Same thing can be done for the product of two vecors.

There can be three cases for addition of two vectors using CUDA.

n blocks and one thread per block.

1 block and n threads in that block.

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 ad- dition to distinct thread.

There can be two cases for addition of two matrices using CUDA.

Two dimensional blocks and one thread per block.

One block and two dimensional threads in that block.

8.1 How to run CUDA Program on Remote Machine

Open Terminal

Get log in to remote system which has GPU and CUDA installed in it. e.g. ssh student@10.10.15.21

Once you get logged in to system, create a cude file with extension .cu and write code in it.

cat ¿¿ sample.cu Write code here Press Ctrl+D to come outside the cat command.

Compile CUDA program using nvcc command. e.g. nvcc sample.cu

It will create executable file a.out. Run it. e.g. ./a.out

9 Test cases

Vector Add:
4 7 8 6 4 6 7 3
10 2 3 8 1 10 4 7
Results :
14 9 11 14 5 16 11 10

Vector Matrix Multiplication:
Initial array :
2 2 1
Initial matrix :
2 3 2 2
1 1 2 3
2 3 2 3
Initial resultant array :
0 0 0 0

Results :
8 11 10 13

Matrix matrix multiplication:
Initial matrix :
10 9 2 8
2 2 2 7
10 1 3 5
3 2 9 7
6 3 8 10

10 9 2
8 2 2
2 7 10

```
1 3 5

Result :
184 146 98
47 57 63
119 128 77
71 115 135
110 146 148
```

10 Conclusion

We have successfully conducted vector addition, vector and matrix multiplication and Matrix and matrix multiplication.