Assignment 2

- Title : Vector and Matrix Operations using CUDA.

- Problem Statement :
  Design a parallel algo to
  1. Add two large vectors
  2. Multiply vector & matrix
  3. Multiply 2 N×N arrays ≤ N^2 processors

- Objectives :
  - To learn abt CUDA architecture
  - To learn CUDA programming concepts

- Outcomes :
  We will be able to learn about CUDA architecture & programming

- Requirements :
  - OS : Fedora 20 / Ubuntu (64-bit)
  - CUDA API
  - nvcc compiler
  - Nvidia GPU (Geforce 920M)
  - Nsight Eclipse IDE

  - RAM : 4GB
  - HDD : 500GB

- Theory:

→ CUDA Architecture:

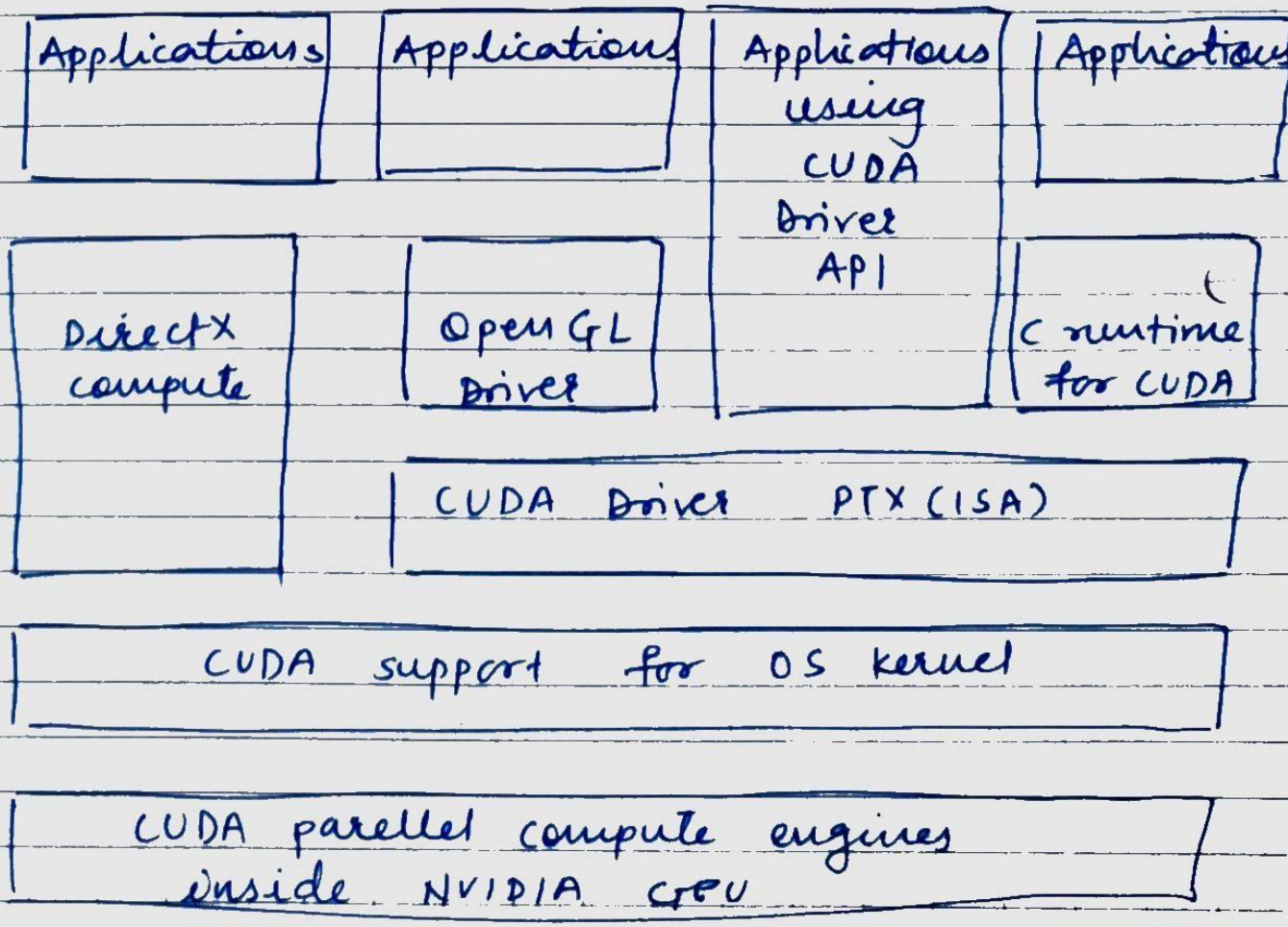
The Architecture consists of several components as

(1.) Parallel compute engines in NVIDIA GPU

(2.) OS-kernel level support

(3.) user-mode driver providing device-level API

(4.) PTX instruction set architecture for parallel computing kernel & functions.



→ Advantages of CUDA:

1. Unified Memory
2. scattered reads - code can be read from arbitrary addresses in memory
3. faster computational to & from GPU
4. full support for integer & bitwise operations
5. Shared memory between threads.

→ Limitations of CUDA:

1. Newer versions of CUDA are now processed acc. to C++ syntax rules instead C
2. Copying betⁿ host & device memory may incur a performance hit due to system bandwidth & latency.
3. CUDA enabled GPUs are available from NVIDIA only.
4. Threads should be running in groups of atleast 32 for best performance.

→ CUDA programming:

Let M & N be the input vectors (or matrices) & P be the result obtained such that

P = M+N (vector addition)

P = M*N (vector-matrix multiplication)

P = M*N (matrix multiplication)

- With CUDA programming, each element in P can be obtained from one thread
- As a result, the problem is decomposed into n (or nxn) threads for parallel reduction.

- Thread ID's are used to differentiate the data with execution & storing the result.

- Test cases and Analysis:-

Operation	I/p size	Sequential time	Parallel time	Efficiency
Vector Addition	n = 256	0.01	0.02	0.5
	n = 1024	0.01	0.02	0.5
	n = 2048	0.02	0.01	2.0
Vector Matrix Multip-lication	n = 256	0.009	0.082	0.037
	n = 1024	0.83	0.135	0.689
	n = 2048	0.367	0.133	2.759
Matrix Multipl-ication	n = 256	0.480	0.02	24.0
	n = 1024	0.620	0.133	4.66
	n = 2048	0.754	0.136	5.544

Camlin Page

$$\text{Efficiency} = \frac{\text{HCSA}}{\text{WCPA}}$$

Here, we observe that parelled algorithm is a major improvement for matrix-matrix multiplication while it is sufficiently significant for increasingly input size for vector-matrix multiplications. No significant improvement gained for vector addition of similar sizes but better for large values.

Input:

Vector 1 → 5, 7, 9, 11, 4, 7, 6, 2, 0, 1

Vector 2 → 6, 2, 7, 1, 0, 4, 13, 17, 12, 5

Output:

Sum of vectors → 11, 9, 16, 12, 4, 11, 19, 19, 12, 6

- Conclusion:

Thus, we successfully implemented various vector & matrix operations parellely using CUDA