Assignment 16

CUDA programming

1. **What is CUDA ?**

A. CUDA = Compute Unified Device Architecture
CUDA is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs).

It allows to access the raw computing power of CUDA GPUs to process data faster than with traditional CPUs. CUDA can achieve higher parallelism and efficiency than general-purpose CPU code using parallel processes. It enables parallel processing by breaking down a task into thousands of smaller "threads" executed independently.

2. **What is the prerequisite for learning CUDA?**

- Understanding parallel computing concepts such as **threads**, **blocks**, **grids**, and **synchronization** is crucial for CUDA programming.
- Many applications of CUDA involve numerical computations, such as **matrix operations** and **linear algebra**.
- CUDA enable or **capable GPU** devices, also have to installed **CUDA toolkit** and nvidia developer driver.

3. **Which are the languages that support CUDA?**

A. C++, C, C#, Fortran, Python, Java

4. **What do you mean by a CUDA ready architecture?**

A. CUDA Ready Architecture refers to a hardware architecture designed by NVIDIA to support CUDA and their parallel computing platform and programming model. In short the GPU architecture is designed and optimized to support CUDA

5. **How CUDA works?**

A. When a processor is given a task, it will pass the instructions for that task to the GPU. The GPU will then do its work, following the instructions from the CPU. GPUs run one kernel (a group of tasks) at a time. Each kernel consists of blocks, which are independent groups of ALUs. Each block contains threads, which are levels of computation. The threads in each block typically work together to calculate a value. Once the job is completed, the results from the GPU are given back to the CPU.

6. What are the benefits and limitations of CUDA programming?

A. Benefits :

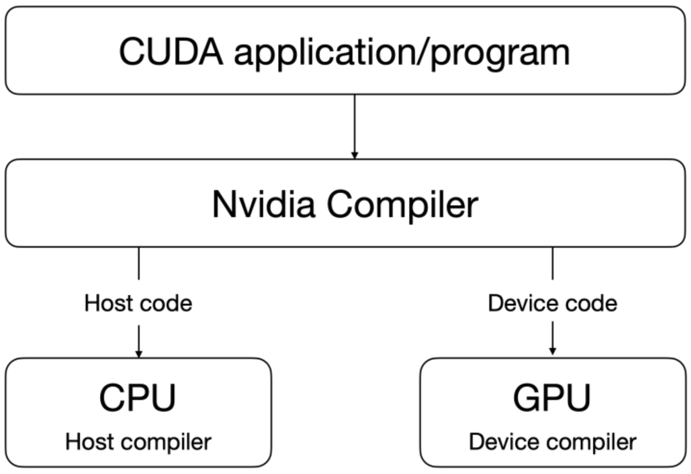
- Massive parallelism
- Performance boost
- Integrated memory and shared memory

Limitations :

- Works only on nvidia GPUs
- Highly Parallelizable Problems Only(Not all problems can be effectively parallelized for GPUs)
- Limited Portability (Porting to other GPU architectures requires significant effort.)

7. Understand and explain the CUDA program structure with an example.

A. CUDA programming includes code for both the GPU and CPU. By default, a typical C programme is a CUDA programme that simply contains the host code. The CPU is referred to as the host, while the GPU is referred to as the device. While the host code can be compiled using a regular C compiler such as GCC, the device code requires a specific compiler to comprehend the API functions that are used. For Nvidia GPUs, the compiler is known as the NVCC (Nvidia C Compiler).



The GPU runs the device code, whereas the CPU runs the host code. The NVCC runs a CUDA programme and isolates the host code from the device code. To do this, specific CUDA keywords are searched for. The code that is intended to run on the GPU (device code) is identified by special CUDA keywords known as 'Kernels' that label data-parallel functions. The NVCC further compiles the device code, which is then run on the GPU.

8. Explain CUDA thread organization.

A.

9. Install and try CUDA sample program and explain the same. (installation steps).

A.