Introduction to CUDA Programming

Last Updated : 16 Oct, 2021

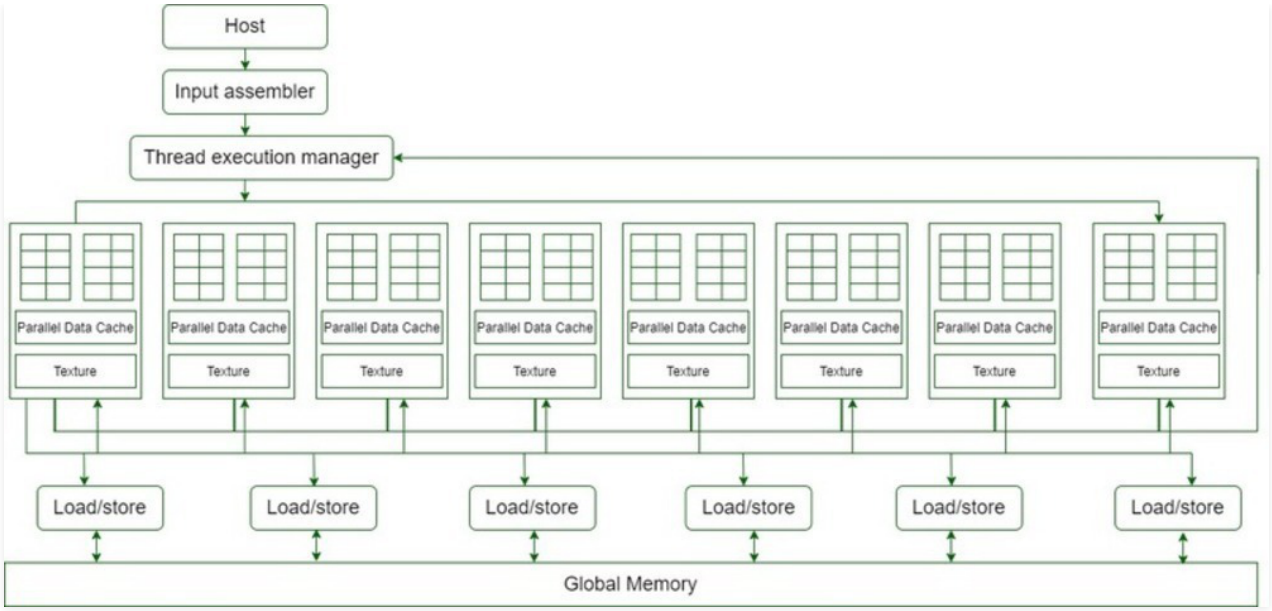
In this article, we will cover the overview of CUDA programming and mainly focus on the concept of CUDA requirement and we will also discuss the execution model of CUDA. Finally, we will see the application. Let us discuss it one by one.

CUDA stands for Compute Unified Device Architecture. It is an extension of C/C++ programming. CUDA is a programming language that uses the Graphical Processing Unit (GPU). It is a parallel computing platform and an API (Application Programming Interface) model, Compute Unified Device Architecture was developed by Nvidia. This allows computations to be performed in parallel while providing well-formed speed. Using CUDA, one can harness the power of the Nvidia GPU to perform common computing tasks, such as processing matrices and other linear algebra operations, rather than simply performing graphical calculations.

Why do we need CUDA?

- GPUs are designed to perform high-speed parallel computations to display graphics such as games.
- Use available CUDA resources. More than 100 million GPUs are already deployed.
- It provides 30-100x speed-up over other microprocessors for some applications.
- GPUs have very small Arithmetic Logic Units (ALUs) compared to the somewhat larger CPUs. This allows for many parallel calculations, such as calculating the color for each pixel on the screen, etc.

Architecture of CUDA



- 16 Streaming Multiprocessor (SM) diagrams are shown in the above diagram.
- Each Streaming Multiprocessor has 8 Streaming Processors (SP) ie, we get a total of 128 Streaming Processors (SPs).
- Now, each Streaming processor has a MAD unit (Multiplication and Addition Unit) and an additional MU (multiplication unit).
- The GT200 has 240 Streaming Processors (SPs), and more than 1 TFLOP processing power.
- Each Streaming Processor is gracefully threaded and can run thousands of threads per application.
- The G80 card supports 768 threads per Streaming Multiprocessor (note: not per SP).
- Eventually, after each Streaming Multiprocessor has 8 SPs, each SP supports a maximal of 96 threads. Total threads that can run – **128 * 96 = 12,228 times.**
- Therefore these processors are called **massively parallel.**
- The G80 chips have a memory bandwidth of 86.4GB/s.

- It also has an 8GB/s communication channel with the CPU (4GB/s for uploading to the CPU RAM, and 4GB/s for downloading from the CPU RAM).

How CUDA works?

- 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.
- Threads in the same block can share memory.
- In CUDA, sending information from the CPU to the GPU is often the most typical part of the computation.
- For each thread, local memory is the fastest, followed by shared memory, global, static, and texture memory the slowest.

How work is distributed?

- Each thread "knows" the x and y coordinates of the block it is in, and the coordinates where it is in the block.
- These positions can be used to calculate a unique thread ID for each thread.
- The computational work done will depend on the value of the thread ID.

For example, the thread ID corresponds to a group of matrix elements.

CUDA Applications

CUDA applications must run parallel operations on a lot of data, and be processing-intensive.

1. Computational finance
2. Climate, weather, and ocean modeling
3. Data science and analytics
4. Deep learning and machine learning
5. Defense and intelligence
6. Manufacturing/AEC
7. Media and entertainment
8. Medical imaging
9. Oil and gas
10. Research
11. Safety and security
12. Tools and management

Benefits of CUDA

There are several advantages that give CUDA an edge over traditional general-purpose graphics processor (GPU) computers with graphics APIs:

- Integrated memory (CUDA 6.0 or later) and Integrated virtual memory (CUDA 4.0 or later).
- Shared memory provides a fast area of shared memory for CUDA threads. It can be used as a caching mechanism and provides more bandwidth than texture lookup.
- Scattered read codes can be read from any address in memory.
- Improved performance on downloads and reads, which works well from the GPU and to the GPU.
- CUDA has full support for bitwise and integer operations.

Limitations of CUDA

- CUDA source code is given on the host machine or GPU, as defined by the C++ syntax rules. Longstanding versions of CUDA use C syntax rules, which means that up-to-date CUDA source code may or may not work as required.
- CUDA has unilateral interoperability(the ability of computer systems or software to exchange and make use of information) with transferor languages like OpenGL. OpenGL can access CUDA registered memory, but CUDA cannot access OpenGL memory.
- Afterward versions of CUDA do not provide emulators or fallback support for older versions.
- CUDA supports only NVIDIA hardware.