# Experiment No. – 01

**Aim:** Study of CUDA and CUDA in Data Structure Algorithm.

**Objective:** In this Case Study , 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.

**Theory:**

* **What is CUDA?**

CUDA is a software development platform used to accelerate parallel computing. It is a specialized programming language for writing programs that run on the GPU CUDA, and it works with most operating systems. CUDA technology enables parallel processing by breaking down a task into thousands of smaller "threads" executed independently. Nvidia CUDA is a technology that has been around since the mid-2000s when it first emerged as a way to boost the performance of NVIDIA GPUs. It's still used today by a wide range of industries and sectors, including but not limited to computer graphics, computational finance, data mining, machine learning, and scientific computing. CUDA is a software platform that enables accelerated computing. It is a specialized programming language that runs on the GPU CUDA, and it works with most operating systems.

* **How to use CUDA?**

CUDA was designed with ease of use in mind. NVIDIA CUDA provides a simple C/C++ based interface. The CUDA compiler leverages parallelism built into the CUDA programming model as it compiles your program into code.

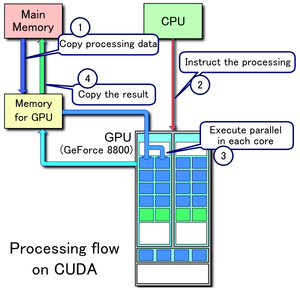
* **Is CUDA faster than CPU?**

CUDA is a parallel computing platform and programming interface model created by Nvidia for the development of software which is used by parallel processors. It serves as an alternative to running simulations on traditional CPUs. Cuda provides faster processing by utilizing the threads that run simultaneously thus boosting the processing power.

* **How Does Nvidia's CUDA Work?**

To understand how Nvidia CUDA works, consider the way a CPU works with a GPU: When a processor is given a task, it will pass the instructions for that task to the GPU. The CUDA GPU will then do its work, following the instructions from the CPU. Once the job is completed, the results from the GPU are given back to the CPU to be used by the software application as needed. This is just one way that CUDA software works.

In fact, with GPU parallel computing, the process is a lot more nuanced. Rather than having the CPU send instructions to the GPU and then wait for the results, the CPU will pass the data for the task to the GPU CUDA. The GPU will then process that data parallel with other GPU devices, using the CUDA programming language. Once the job is completed, the data will be sent back to the CPU so that the software application can use the results. Parallel processing is key to understanding how CUDA software works.



Example of CUDA processing flow

1. Copy data from main memory to GPU memory
2. CPU initiates the GPU compute kernel
3. GPU's CUDA cores execute the kernel in parallel
4. Copy the resulting data from GPU memory to main memory

* **What are Graphics Processing Units (GPU)?**

Graphics Processing Units (GPUs) make your computer’s graphics software much faster. Let's say you are playing a graphic-intensive game that has thousands of polygons on the screen at any one time. These polygons typically colour their animation, colour patterns, and unique light sources, and interacting with every polygon can be incredibly tiring for a CPU. This is where GPUs become so helpful. The GPUs take thousands of different polygonal movements and put them in batches. It has its own set of cores that parallel process all the separate graphic information for each polygon and then present this to your computer monitor or television screen as fast as possible.

This is what brings life to your video game just in time for you to grab that next high score.

* **How to test CUDA code without GPU?**

The new dev kit lets you get your app up and running effortlessly. As part of its key features, the IntelliSense code parser that produces the XML template for CUDA is easy to use without installing or downloading anything.

|  |  |
| --- | --- |
| **CPU** | **GPU** |
| Central Processing Unit | Graphics Processing Unit |
| Low Latency | High Latency |
| Several Cores | Many Cores |
| Good for serial processing | Good for parallel processing |
| Can do a few operations at once | Can do many operations at once |

* **CUDA GPU Vs CPU**

CPUs are known to be powerful. Still, they can only handle a few applications at a time because they are not built with parallel processing in mind. In contrast, while CUDA GPUs were initially thought of as specialized devices that could only work with graphical tasks, this same design has opened the door for parallel processing technology to play a significant role among millions of gamers.

The main difference between a CPU and a CUDA GPU is that the CPU is designed to handle a single task simultaneously. In contrast, a CUDA GPU is designed to handle numerous tasks simultaneously. CUDA GPUs use a parallel computing model, meaning many calculations co-occur instead of executing in sequence.

* **8 Common Uses of GPU Parallel Computing:**

1. Computer graphics Startups and significant-tech companies use GPUs to create photo-realistic 3D graphics for games, movies, etc. GPUs allow for more detailed and realistic graphics than CPUs are capable of.
2. High-performance computing With GPU compute, you can process large amounts of data in parallel, perfect for weather forecasting, financial predictions, and other high-performance computing tasks.
3. Deep learning has been used in various areas, including speech recognition, image recognition, and natural language processing.
4. Computer modelling GPU can be used to make highly accurate models, such as climate change models and models of the human brain.
5. Data mining It is a way of discovering useful information from large amounts of data. GPUs make data mining faster and more efficient.
6. Scientific computing is used in fields like chemistry, biology, and physics. With GPU parallel computing, you can speed up simulations and modelings.
7. Machine learning ML is used in a variety of fields. It's used in image recognition algorithms, text analysis, and predictive modeling.
8. \*\*Web services \*\* With GPU compute, you can speed up your services on the web, such as data analytics, weather forecasting, and stock price predictions.

* **3 Best Practices for Developing with GPU CUDA**

As you can see, you would want to implement CUDA Programming in your organization. But before you dive into the process, make sure you're following the best practices for developing with GPU CUDA. Here are some tips to help you get started.

1. Plan Ahead - Before you start thinking about CUDA GPU, you need to plan. Start by finding out how many GPUs you have in your organization. Then you can see how many CUDA-capable GPUs your hardware has. You'll also look at your software and see if it is CUDA compatible. This will help you figure out how many computers you need to purchase.
2. Get the Right Hardware - Not all CUDA GPUs are built the same. There are many different types of GPUs, each of which gives off an extra level of performance. You willneed high-end hardware to get the most out of your organization's CUDA deployment.
3. Invest in Training - One of the best ways to get the most out of your CUDA deployment is by investing in training. Make sure your team members know how to use the hardware and software properly and understand how to maximize their GPUs' performance.

**Conclusion:**

In conclusion, CUDA C++ provides a powerful programming model for GPU parallelism. By extending the C++ language with GPU-specific features and syntax enhancements, CUDA C++ enables the creation of highly parallel and efficient code for a wide range of applications.