Chapter 7
GPU Computation and Platforms

K. Bhargavi
Siddaganga Institute of Technology, India

Sathish Babu B.
Siddaganga Institute of Technology, India

ABSTRACT

The GPUs (Graphics Processing Unit) were mainly used to speed up computation intensive high performance computing applications. There are several tools and technologies available to perform general purpose computationally intensive application. This chapter primarily discusses about GPU parallelism, applications, probable challenges and also highlights some of the GPU computing platforms, which includes CUDA, OpenCL (Open Computing Language), OpenMPC (Open MP extended for CUDA), MPI (Message Passing Interface), OpenACC (Open Accelerator), DirectCompute, and C++ AMP (C++ Accelerated Massive Parallelism). Each of these platforms is discussed briefly along with their advantages and disadvantages.

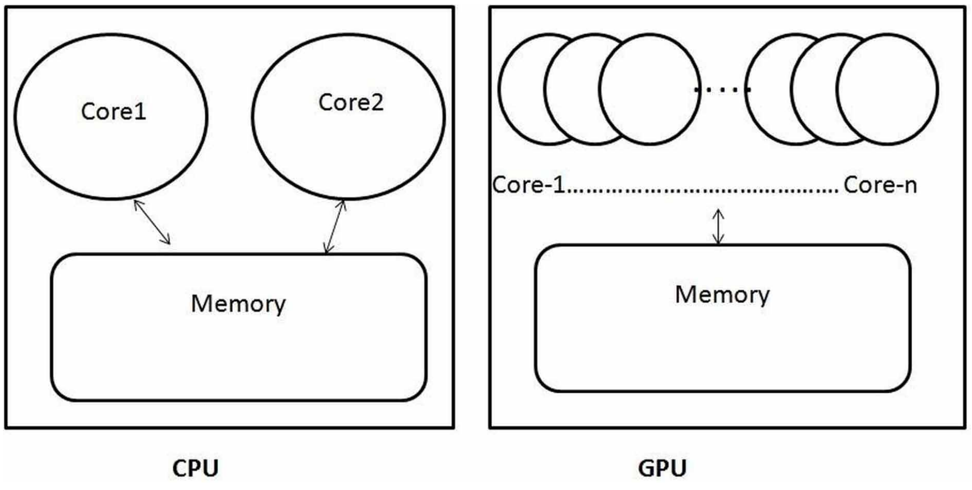
1. INTRODUCTION

CPUs are generally optimized to do basic arithmetic, logical, and control operations. CPU is made up of few cores and they are good at sequential processing of integer operations. GPU are optimized to perform trigonometric operations and graphic processing but the GPU performs computation at a faster rate than CPU. GPU usually consists of many cores and they are good at parallel processing of instructions with real data type. The system, which is the consolidation of both CPU and GPU, will perform better with respect to cost and efficiency. An abstract representation of CPU and GPU is given in Figure 1 (Varhol, 2010).

The history of GPU begins with the introduction of Atari 8-bit computer text chip during the year 1970 to 1980 then IBM PC professional graphics controller card was introduced amid of 1980 to 1990. During the span 1990 to 2000, a faster growth has happened in the production of wide varieties of GPU which includes S3 graphics cards, Hardware-accelerated 3 dimensional graphics, OpenGL graphics API, DirectX graphics Application Programming Interface (API) Nvidia GeForce and GPUs with programmable shading. From the year 2000, general computation job using GPUs are being used extensively (Chris, 2010).

DOI: 10.4018/978-1-4666-8853-7.ch007

Figure 1. Abstract representation of CPU and GPU



GPU relies mainly on parallel computation and offers several benefits over conventional CPU computation like high band-width availability, reduced power consumption, increased computing and offers several benefits over conventional CPU computation like high bandwidth availability, reduced power consumption, increased computing capacity, efficient bandwidth utilization, speedy execution of instructions, methodical resource allocation, and so on. Hence GPU computing are widely used in variety of applications like 3D gaming, video editing, bio-molecular simulations, quantum chemistry, numerical analytics, weather forecasting, fluid dynamics, animation, image processing, medical imaging, analyzing air traffic flow, visualizing molecules, etc.

1.1. CPU Parallel Computing vs. GPU Parallel Computing

One of the main aims of GPU is to achieve parallelism but the parallelism can also be achieved using many cores CPU. The parallelism achieved by CPU differs from the parallelism achieved by GPU. CPU parallelism can be achieved with multi cores. Here almost all transistors are devoted to per-form memory allocation and management activities. As the transistors spend more time in resource management, the computation speed will decrease and its performance with respect to bandwidth and throughput will be reduced. If the application that we are developing is small and has only a little number of tasks to be carried out in parallel then CPU is best option. Because, if we spool those tasks on GPU which has many cores, it is not possible to efficiently utilize all cores and this may in turn lead to slower rate of computation. CPUs are good at achieving parallelism at instruction level whereas GPUs are good at achieving parallelism at thread level. In GPU parallelization, while dividing a large program into smaller modules, care should be taken such that all blocks are of same size. Accessing of global memory has very high latency in GPU and branch diversions can also cause more bottleneck situations in GPU. A basic representation of CPU and GPU parallelism is given in Figure 2 (David & Greg, 2007) and (Mayank et al., 2011).