ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 6, Issue 2, February 2017

GPU Computing

Chandana G P
Oracle India Pvt ltd.

Abstract

The GPUs (Graphics Processing Unit) are 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 paper primarily discusses about GPU parallelism, applications, probable challenges , evolution of GPU architecture, advantages and objectives.

Keywords: GPU, Computing Application areas, Evolution, Architecture,G80, G200, Fermi, Kepler-GK110, Advantages, Objectives

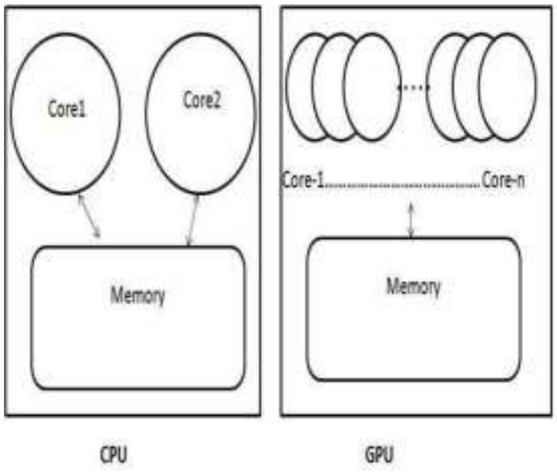

Figure 1: Abstract representation of CPU and GPU

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.

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 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 dimensional graphics, OpenGL graphics API, DirectX graphics Application Programming Interface Nvidia GeForce and GPUs with programmable shading. From the year 2000, general computation job using GPUs are being used extensively 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 versus GPU parallel computing

One of the main aim 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; Mayank, Ashwin, and Wu-chun, 2011).

2 GPU computing Application Areas

GPU computing is mainly used to enhance the execution speed of several computation intensive applications in the areas of medicine, physics, engineering, mechanics, geology, astronomy, and so on. Some of the important applications have been discussed below (GPU accelerated computing, 2014; GPU-ACCELERATED APPLICATIONS, 2014; Cristobal, Nancy, and Luis, 2014; GPU APPLICATIONS, 2014; Harris, 2004; In-Gu, and JungHyun 2006; Jorg, 2014; Moulik, and Boonn, 2011; Che, Li, Sheaffer, Skadron, and Lach, 2008).

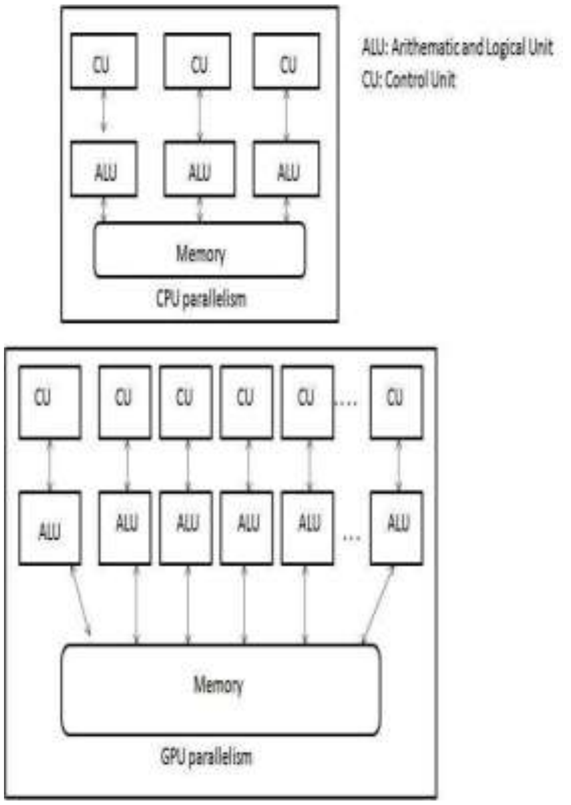

Figure 2: CPU Parallelism versus GPU parallelism

2.1 Weather and climate modeling

Climate is nothing but average weather and changes in the climate will have impact on weather cases. Climate usually changes according to their demographic region and it is one of the areas of research, which in turn needs the development of appropriate weather and climate models. The models will simulate various aspects of weather like atmosphere, oceans, lake, land, ice burgs, rainfall, storm, floods, etc. The simulation of above aspects of weather requires computation of complicated governing equations and it needs to be done within a shorter period of time. Such massive simulation requires hundreds of CPUs and performing operations using them is tedious. Therefore GPU parallel programming environment is used to model weather and climate changes.

2.2 Quantum Chemistry

Quantum Chemistry applies quantum mechanics to solve the problems related to Chemistry like behavior of atoms, movement of molecules, electrons collision, nuclear chain reaction, photons motion, visualization of molecular orbits, computation of electronic structures, and so on. The cost and time required for simulating data intensive and complex calculation involved jobs which requires massive number of CPU cycles and such limitation are threat for the wide spread use of Quantum Chemistry. The usage of conventional systems with CPUs for simulation of molecules and photons are very much time consuming. GPUs have many arithmetic units which work in parallel to fasten the calculation rate of such applications.

2.3 Medical imaging

Medical imaging is concerned with the interior view of the human body. Its main intention is to get the internal structure or pattern of human body components. It includes various techniques like X-ray, Magnetic Resonance Imaging (MRI), chemotherapy, Computed Tomography (CT) scan, medicine imaging, Electrocardiogram (ECG), etc. Medical imaging involves massive computation using 3D datasets; it basically relies on GPU computing for delivering accelerated medical

2.4 Dimensional computer graphics

Dimensional computer graphics includes movie, marketing, forecasting, gaming, and so on. It provides detailed description of both realistic and non-realistic images, audio, and video data. Graphics processing includes massive computation jobs like simulation of the appearance of real world objects, series of transformation of objects in 3 dimensional space, portraying the objects in 3 dimensional space, applying animation to the existing 2D objects, etc. Often video gaming applications are sensitive to the number of cores, on which they are operating on, these applications relies more on GPU enabled systems.

2.5 Environmental Science

Environmental science is a multi disciplinary area, which includes several domains like rainforest deforestation, agriculture, pollution control, natural resource management, global warming, soil contamination, etc. All these applications involve enormous amount of data and parallel computing is one effective way to simulate such heterogeneous computation intensive applications. GPU computing platforms like OpenCL and CUDA are extensively used for these purposes.

3 Evolution of GPU architecture

GPU architecture has evolved much after the year 2006 with respect to programmability, number of cores, pipelining units, computation intension, double precision operations, etc. In this section, we discuss the popular GPU architectures that came after 2006, which includes G80, GT200, Fermi, and Kepler (Henry, Misel, Maryam, and Andreas, 2010; NVIDIAs Next Generation CUDA Compute Architecture: Fermi, 2009; NVIDIAs Next Generation CUDA Compute Architecture: Kepler GK110, 2010; Nvidia G80 Architecture and CUDA Programming, 2007; Steve, 2007; Peter, 2009; Lars, and Stephen, 2012).

3.1 G80

NVIDIA G80 is based on Single Instruction Multiple Threads (SIMD) model and can spool around 768 threads and 768 blocks. It was officially released to market in the year November 2006 and was based on conventional C language. It consists of 8 TPCs (Texture/Processor Cluster), where multiple threads where spooled and all accessed the shared memory and synchronization was established among the threads. But one of the considerable limitations of G80 architecture was to efficiently utilize all available threads, which in turn requires efficient estimation of exact number of threads and blocks that has to be spooled and other factors like memory and data size within each thread have to be considered. Typically ten registers are used by each thread and however if we allocate more and more registers per thread there are chances of decrease in achievable concurrency level. Sample G80 architecture is shown in figure 3.

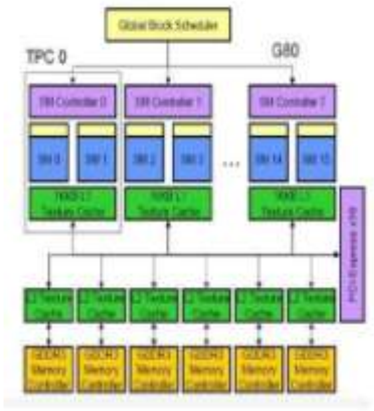

Figure 3: G80 architecture (Source: NVidias GeForce GTX 280)

3.2 GT200

GT 200 architecture is popularly referred as Tesla architecture and was released in the year June 2008. It is an extension to G80 architecture. It consists of 10 TPCs, which will increase the processing strength of GT200 by 87.5 percentage when compared to G80. It has been considered as one of the widest chip, because it is made up of around 1.4 million transistors. GT 200 has dual issue design and are capable to perform both addition and multiplication operation in one clock cycle. It can handle 32 pixels per clock cycle, which is greater when compared to GT80, because GT80 can handle only 12 pixels per clock cycle. It can perform around 933 Gigaflops of operations per unit time, which is almost double than that of the G80 series. GT 200 made a vigorous attempt to increase the floating power ratio from 14.1 to 19.4, and geometric shading capacity has also been enhanced. Sample GT200 architecture is shown in figure 4.

3.3 Fermi

Fermi architecture was released in the year 2010 and is also referred as GT300. It was one of the

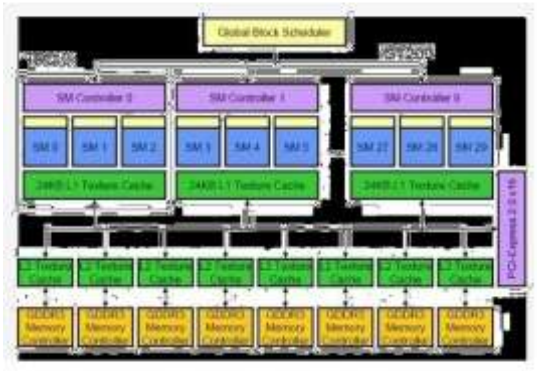

Figure 4: GT200 architecture (Source:NVIDIAs GeForce GTX 280)

best architecture released by NVIDIA, because after the release of G80, user feedback was taken all that all that was incorporated in Fermi architecture and feedback was taken mainly with respect to memory and performance rate. It has been honored as first complete GPU computing platform that aims to solve all scientific application problems. It consists of around 3.0 billion transistors and is capable of executing both floating point and integer operations per clock cycle. It supports parallel thread execution 2.0 instructions set. It was ten times faster in context switching and eight times faster in double precision floating point performance than execution 2.0 instructions set. It was ten times faster in context switching and eight times faster in double precision floating point performance than GT200. As a major enhancement all latches where made configurable, and there a support was extended for automatic memory

3.4 kepler-GK110

Kepler GK110 was released in the year 2010 is the fastest GPU available today and is made up around 7.1 billion transistors. Kepler GK110 is 60 percent more efficient than Fermi. Kepler is one of the most widely used GPU for High Performance Computing (HPC) applications. It efficiently manages the available grid locations and in turn supports dynamic parallelism. It has been built with Hyper-Q technologies, which helps in enhancing the performance by efficiently utilizing the GPU capacity and also increases the total number of connections that can be con-

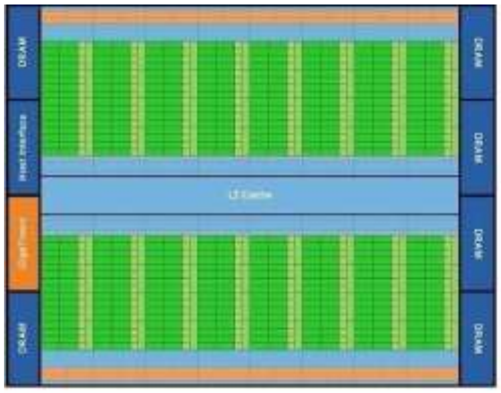

Figure 5: Fermi architecture (Source: NVIDIAs Next Generation CUDA Compute Architecture:Fermi)

nected simultaneously by providing a maximum connection limit to 32. It facilitates direct communication between GPUs without the intervention of the memory unit of GPU, thereby reducing the inter GPU communication cost, and also removes the latency bottleneck. A sample kepler GK110 architecture is shown in figure 6.

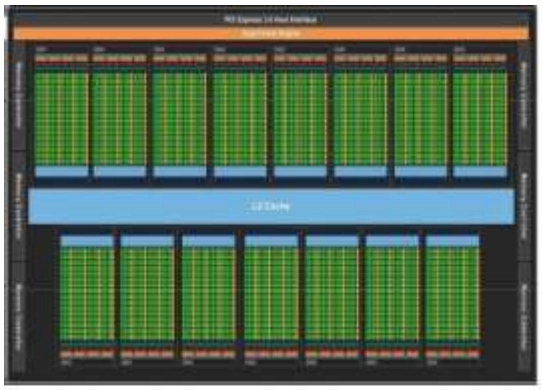

Figure 6: kepler-GK110 architecture (Source: NVIDIAs Next Generation CUDA Compute Architecture: Kepler GK110)

4 Challenges in GPU Computing

GPUs are extensively being used in variety of applications but GPU computation usually involves several challenges. Some of the challenges have been listed below (Nabeel, and Milind, 2013; GPU Optimization, 2011; Ethel, and Alastair, 2014; Byunghyun, Dana, Perhaad, and David, 2010; Hadi, Wilson, Mike, Joseph, and Tor, 2013; GPU Computing To Exascale and Beyond, 2010).

4.1 Manual Caching

While GPUs operation, user intervention is required during caching process, automatic caching of shared variables for certain parallel segments is not possible. It becomes a major constraint while performing computation intensive jobs.

4.2 Thread mapping

Usually many core systems consist of multiple cores, so efficient mapping of threads to appropriate core is very much important but there are several issues in it like selection of cores, choice of number of cores to be built inside a platform, and alertness of application variability.

4.3 Synchronization

In GPU computing more focus will be on performing the computation, but GPU computation usually involves several parallel threads that will be operating simultaneously. So, synchronization need to be achieved among the parallel threads and care should be taken that each thread gets right amount of data.

4.4 Memory access pattern

With the availability of massively parallel multithreaded GPU architecture, the pattern of memory access is very much important and there are chances that some of the execution units may remain idle or underutilized. So, care should be taken such that all execution units remains busy, which in turn improves the memory performance.

5 Advantages of GPU Computing

GPU computing offers several advantages over the traditional CPU computing. Some of the major advantages of GPU computing have been listed below.

5.1 Speed

GPUs are mainly used to speed up or accelerate the computation rate of the applications. Speed up rate basically depends on the kind of application and it varies from one GPU based platform to another. For example for scientific and engineering applications both OpenCL and CUDA perform at a very higher rate by attaining maximum available precision.

5.2 Efficiency

GPU executes more arithmetic operations per clock period time when compared to CPU and in turn reduces the execution time of parallel code. Typically GPU consumes 200pJ/instruction whereas CPU consumes 2nJ/instruction indicating its efficiency against CPU.

5.3 Throughput

CPU usually has less number of cores, which in turn limit its throughput rate where as the GPU enhances the throughput rate of the application by providing many parallel processing segments. Now GPUs are also available with highly parallel, multi core architecture, which delivers higher throughput for data intensive processing applications.

5.4 Bandwidth

Bandwidth is basically the amount information getting transferred. As GPU scale towards more and more applications they get limited by the bandwidth availability, therefore GPUZs are coupled with the Direct Memory Access (DMA) wraps, which in turn improves the bandwidth utilization rate by taking the advantage of parallelism at the memory and data level.

5.5 Scalability

GPU is referred as scalable parallel computing platform, which contains n number of threads that are operating in parallel and can extend up to hundreds or thousands of cores. CUDA is one of the parallel computing platforms, which allows both data and instructions to operate in parallel and thereby providing enhanced scalability.

5.6 Programmability

Earlier GPUs were programmable only using the assembly level languages, therefore allocating memory and usage of graphics primitives was extremely difficult. But with the advent of General Purpose (GP)-GPUs, programming became easier and it has been considered as one of the powerful tool that performs programming with very high precision and power.

5.7 Parallelism

GPU is built with parallel architecture and it has multiple cores, which are capable enough for handling several jobs simultaneously. This in turn increases the computation speed of the applications containing vast amount of data and optimizes the end result. Certain applications like simulation of ocean current, galaxy evolution, ray tracing, and so on are highly benefited from the parallel architecture. Objections to GPU computing

6 Ob jectives to GPU computing

GPU computing has become one of the buzz words nowadays, especially in the High Performance Computing (HPC) world. But we come across many objections to GPU computing and it has been said that GP-GPU computing is not the final solution to all HPC applications. Some of the frequent objections that we see around have been listed below.

6.1 Rewriting of code

If we are coding parallel implementation of CPU from a mostly serial code then rewriting of code is inevitable and for parallel recoding we can use MPI, OpenMP/POSIX threads, and CUDA or Streaming SIMD Extensions/ Advanced Vector Extensions (SSE/AVX). CUDA is an extension of C and it always remains as a programmers choice for parallel computation.

6.2 Performance requirement estimation

Parallel codes are either computation or memory restricted. Computation restricted applications prefers NVIDIA Fermi M2090 GPU cards and memory restricted applications prefers GPU with memory bandwidth 177 GB/Sec. When providing parallel solution to problems the programmer need to make choice between of computation or memory and that will be done based on the performance requirement.

6.3 PCIe bandwidth availability

GPU computation speed and bandwidth are limited by the PCIe (Peripheral Component Interconnect

express) bus bandwidth. If the PCIe v2.0 x16 bandwidth 6 GB/sec then it can fill only is 6 GB of data per second and as a result the GPU programmer must strive hard to keep more data on the GPU board as long as possible. If there are clusters of GPU then this limitation will in turn limit the rate of data transfer between the GPUs.

6.4 Amdahls law justification

According to Amdahls law, maximum expected improvement of the overall system can be found by improving only one part of the system. But main objection to Amdahls law during GPU computation is to determine which part of the system to be improved. Because it contains massive parallel codes running simultaneously on the board.

6.5 Main Memory

The M2090 and M2070 GPU boards have limited main memory capacity i.e., 6GB, which will become a problem when we try to simulate large amount of data. But we can also embed multiple cards into a node. For example Dell C410x board contain up to 16 NVIDIA boards that is on a total it offers 96 GB of memory. But when we have integrated boards decomposing a large problem to sub problems and assigning them to appropriate boards will become an issue.

6.6 Budget

To build an organization with more GPU enabled nodes, a decision need to be taken either to have all the nodes that are GPU enabled or preferably use some nodes that are GPU enabled. This decision need to be taken mainly on cost basis as deploying all GPU enabled nodes is costlier when compared to the CPU enabled nodes.

7 Conclusion

GPU computing is mainly used to accelerate computation intensive applications in the field of medicine, physics, engineering, mechanics, geology, astronomy, and so on. The acceleration is achieved via parallelism over multiple cores of GPUs. The usage of GPUs in computation intensive applications is found to be beneficial with respect to various parameters like power consumption, computing capacity, bandwidth utilization, speed of execution, resource allocation, and so on.

8 References

[1] GPU accelerated computing. (2014). Retrieved october 15, 2014, from http://www.nvidia.com/

[2] GPU-ACCELERATED APPLICATIONS.(2014)Retrieved october 22, 2014 from http://www.nvidia.com/content/PDF/gpu-acceleratedapplications.pdf

[3] GPU vs. CPU Computing. (2007). Retrieved october 23, 2014 from http://www.deskeng.com/de/gpu-vs-cpu-omputing/

[4] David, L., and Greg, H. (2007). How GPUs Work. IEEE Computer Society, 40(2), pp. 96-100.

[5] Chris, M. (2010). History and Evolution of GPU Architecture

[6] Jorg, K. (2014). Ab Initio Quantum Chemistry on Graphics Processing Units. http://www.prace-ri.eu/IMG/pdf/pracedays14_ kussmann.pdf

[7] Cristobal A. N., Nancy, H. K., and Luis, M. (2014). A Survey on Parallel Computing and its Applications in Data-Parallel Problems Using GPU Architectures. Communications in Computational Physics, 15(2). doi: 10.4208/cicp.110113.010813a

[8] Henry, W., Misel, M.P., Maryam S. A., and Andreas M. (2010) Demystifying GPU Microarchitecture through Microbenchmarking. IEEE International Symposium on Performance Analysis of Systems Software (ISPASS).

[9] Che, S., Li, J., Sheaffer, J. W., Skadron, K., and Lach, J. (2008). Accelerating compute-intensive applications with GPUs and FPGAs. In IEEE Application Specific Processors, pp. 101-107.

[10] Moulik, S., and Boonn, W. (2011). The role of GPU computing in medical image analysis and visualization. In SPIE Medical Imaging, pp. 79670L-79670L.

[11] In-Gu, k., and JungHyun, H. (2006). Real-Time animation of large crowds. Fast fluid dynamics simulation on the GPU. Retrieved october 22, 2014 from http://http.developer.nvidia.com/GPUGems/gpugems_ch38.html

[12] Expanding the boundaries of GPU computing.(2010). http://www.dell.com/downloads/global/power/ps3q10-20100491-gpu.pdf

9 About Author

Chandana G P is working as a Software developer in Oracle. Her area of interests are Big Data, Cloud Computing, Programming and High performance computing