Volume 6, Issue 3, March 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Thin Computing Using Graphics Processing Units

Chetan Ingle, Saurabh Gaikwad, Pratik Shenoy, Smita Bhoir
Department of Computer Engineering, Ramrao Adik Institute of Technology
Navi Mumbai, Maharashtra, India

Abstract*—*Graphics Processors were initially used mainly for gaming and entertainment purpose. But recent developments show that a Graphic Processing Unit can be used for performing many scientific and general purpose applications. Hence, the term General Purpose Graphic Processing Units (GPGPU) came into existence. There are many different architectures of GPU which can be used for General Purpose Computing. In this paper we make a comparative study of GPU and CPU properties and how GPU technology can enhance the functionality of CPU. At the end we propose the applications of GPGPU computing in Thin Computing and how GPGPU computing can increase the efficiency of Thin Computing Architectures.

Keywords*—*GPU; GPGPU; Thin Computing; NVIDIA KEPLER; Maxwell architecture

I. INTRODUCTION

A general purpose computing on graphical processing units is basically the use of a Graphical Processing Unit which handles operations for computer graphics and operations originally handled by the Central Processing Unit (CPU).

The Graphical Processing Units are high-performance processing units which are equipped with many-core processors to handle high computation. Though they were specially designed for computer graphics and were inconvenient to program, today's GPUs are general purpose parallel processors which handle programming interfaces and standard languages like C language. Speedups of order of magnitude vs. optimized CPU implementations are achieved by developers who port their applications to GPUs. The GPU computing model comprises of making use of the GPU and CPU together in a diverse co-processing computational model. The CPU runs the sequential part of the application and the GPU accelerates the computationally-insensitive part. Due to this the application runs faster as the high performance GPU boosts the CPUs performance.

As the Graphics Processing Unit features parallel processing, complex tasks are divided into smaller tasks upon which computation is done simultaneously. Due to this ability of a GPU, some of the most difficult computational problems can be addressed several orders of magnitude faster. Section II gives literature survey on thin computing and different GPU architectures, section III gives comparative analysis of different architectures. Section IV gives proposed work and finally section V gives conclusion and future scope of system.

II. LITERATURE SURVEY

A. *Thin Computing :*

A thin client [1] is a computer or a computer program that relies on some other computer (its server) to fulfill its computational roles. This is different from the conventional fat client, which is a computer designed to take on these roles by itself. The specific roles assumed by the server may vary, from providing data persistence (for example, for diskless nodes) to actual information processing on the client's behalf.

Today Computer Hardware is so advanced that general computing uses just 10% of its capability. Thin Computing is the technology to consume that unused capability. In this Technology a single CPU with good configuration or a Server is used to perform all operations. According to the configuration of the system we can increase the users. Each Thin Client has its interfacing units, both input as well as outputs.

Thin Clients are connected to the main server or CPU. In which a separate virtual machine is maintained and served for each of the thin client. Thin Clients send the server only mouse and keyboard events. The host traps all the events it is receiving from all clients and inserts them in respective virtual machines present inside the host. Virtual Machines work as same the real machine and provide the graphical output to the host. Host sends the output to respective thin client.

B. *Requirements of Thin Computing:*

Today's thin computing [2] has the following hardware and software requirements:

Hardware**:**

- Keyboard
- Pointing device (Mouse)
- Network connectivity
- Minimum 11.5 MB RAM and 4 MB ROM
- Approximately 1.2 GHz processor

Software**:**

- Minimum desktop resolution - 640 x 480
- Remote Desktop Protocol (RDP)

We find that the above Hardware and Software requirements can be easily handled by a GPGPU.
Now let us find whether a CPU can be replaced by a GPGPU.

C. CPU. v/s GPU.

The GPUs were initially designed so that large number of addition and multiplication operations which were performed in graphics rendering could be accelerated. Initially it was packaged as a video card which was attached to the PCI bus. Later this intensive computational unit was offloaded from the CPU. In the course of time as demands for high performance graphics increased, the GPU kept getting powerful and eventually it is now more powerful than the CPU.

The type of computation performed for graphics rendering is closely related to the simulation of scientific and engineering problems. Both of them carry out multiply and add operations heavily. However, the two important differences between them are as follows –

- In basic engineering and science 64-bits (8-bytes) of precision is required for the amount of information which is stored and processed with each data point. 32-bits (4-bytes) are normally required for graphics applications. Significant difference between data points is obtained from the result of solving differential equations. This is where the extra precision is required whereas only the value which is present at the data point is required for graphics applications.
- Quite often in graphics application the data is recomputed several times a second. Thus error in a data point is normally not noticed. However, in engineering and scientific applications, the current computation depends on the results of the previous computation. Due to this an error in one computation will cause the entire analysis to be useless.

Thus specifically for scientific and engineering workstations, the Nvidia Fermi and recent GPUs are designed. These GPUs have data storage, paths and arithmetic units with a precision of 64 bit. Also, they provide the reliability required for long simulations as they contain error correcting memory.

The following table compares the processing capability for current general purpose CPU processors with those found in GPUs.

Table I CPU v/s GPU [3]

Parameter	**CPU**	**GPU**
Number of cores	4	448
Flops per core	4	1
Clock speed(GHz)	2.5	1.15
Performance(Gflops)	40	515

The key to GPUs performance is the large number of processing cores it contains. Parallel computation on 32 data streams is performed by symmetric multiprocessor which is at the heart of the GPU. Depending on the architecture each GPU contains 14 to 16 multiprocessors for a total number of cores ranging from 448 to 512. Currently in a single system (node) 8 GPUs can be installed. The architecture of the world's fastest supercomputers consists of systems having thousands of nodes, with each node being installed with a GPU.

The benefits of the architecture of GPUs are as follows:

- Faster Processing - Over general purpose CPU, each GPU provides an order of magnitude or more in performance. Due to this it has ability to solve large problems and consuming less time.
- Lower capital cost - For the same cost GPUs provide an order of magnitude or greater in processing power.
- Reduced power consumption - The GPUs consume less power as they perform more floating point operations per watt.

D. Evoluion of GPU

In recent times GPUs [4] have evolved to contain large number of parallel threads and many cores. This is because real times graphic performance needed to render complex, high resolution 3D scenes still continue to be the driving force.

It is quite problematic to render high-definition graphic scenes with massive inherent parallelism. A single thread program that draws one pixel is written by a graphics programmer and then the GPU runs many instances of this thread in parallel-drawing multiple pixels in parallel. The Graphics programs scale transparently over a wide range of threads as they are written in shading languages such as Cg or High Level Shading Languages. Also GPU computing programs that are written in C++ or C with CUDA computing model or using API such as Direct Compute or OpenCL scale easily over a wide range of parallelism. Even due to the increasing transistor density, software scalability has made possible for the GPU's to rapidly increase parallelism.

Table II GPU technology development [5]

Date	Product	Transistors	CUDA cores	Technology
1997	RIVA 128	3 million	-	3d graphics accelerator
1999	GeForce 256	25 million	-	First GPU programmed with dx7 and open gl
2001	GeForce 3	60 million	-	First programmable shader GPU
2002	GeForce FX	125 million	-	32-bit floating pt. Programmable
2004	GeForce 6800	222 million	-	32-bit floating pt. Programmable scalable GPU
2006	GeForce 8800	681 million	128	First unified graphics and computing GPU programmed with c in CUDA
2007	Tesla T8, C870	681 million	128	First GPU programmed in CUDA
2008	GeForce GTX	1.4 billion	240	Unified graphics and computing
2008	Tesla T10, S1070	1.4 billion	240	GPU computing clusters, 64-bits
2009	Fermi	3 billion	512	GPU computing architecture, IEEE 754-2008 FP, ECC memory
2010	GeForce GTX 570	3 billion	480	Fermi architecture
2011	Radeon HD 7970	4.2 billion	-	GCN architecture
2012	Tesla K20	7.1 billion	-	Kepler-based architecture
2013	AMD R9 290 X	6.3 billion	-	GCN architecture
2014	GeForce GTX 980	5.2 billion	2048	Maxwell architecture
2015	GeForce GTX Titan X	8 billion	3072	Maxwell architecture

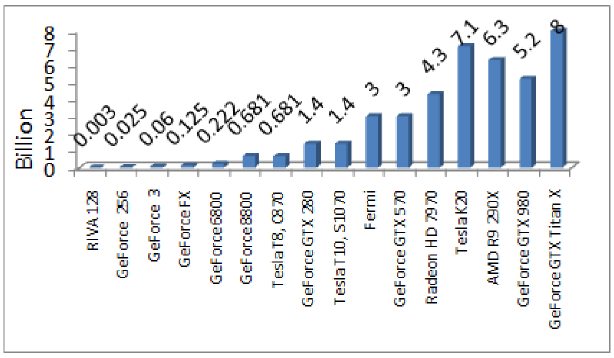


Fig 1 Evolution of GPUs

From the above figure we can see that the GPU transistor count is increasing exponentially. It is doubled almost every 18 months with increasing semiconductor density. The CUDA parallel computing cores per GPU also doubled roughly every 18 months since their introduction in 2006. In the early 1990s, as there were no GPUs, the Video Graphics Array (VGA) controllers were used to generate 2D graphics displays for PCs to accelerate graphical user interfaces. NVIDIA released the RIVA1283D in 1997, a single chip graphics accelerator for games and 3D visualization applications and was programmed with OpenGL and Microsoft Direct3D. Evolving to modern GPUs involved adding programmability incrementally—from fixed function pipelines to micro-coded processors, configurable processors, programmable processors and scalable parallel processors.

E. EarlyGPUs

The Nvidia GeForce 256 was the first GPU that was introduced in 1999 that had a single chip 3D real time graphics processor which included almost every feature of a high end 3D graphics work station of that generation. It had a 32-bit floating-point vertex transform which was configurable and had a lightening processor. It also contained a configurable integer pixel fragment pipeline, programmed using Microsoft DirectX7 and OpenGL.

The first GPUs calculated vertices and 3D geometry using floating point arithmetic and then applied it to colour values and pixel lightening to simplify programming and handle high dynamic range scenes. As they provided accurate floating point rounding, the GPUs were able to eliminate frame-varying artefacts on moving polygon edges that would sparkle at real-time frame rates.

Over a period of time as programmable shades emerged, the GPUs became easier to program and hence became more flexible. The Nvidia GeForce 3 launched in 2001 was the first GPU with a vertex processor that could be programmed and was able to execute vertex shader program. It consisted of 32-bit floating-point pixel-fragment pipeline which was configurable and was programmed with Microsoft DirectX8 and OpenGL. In 2002, AMD introduced ATI Radeon 9700 which consisted of 24-bit floating-point pixel-fragment programmable processor and was programmed using Microsoft DirectX9 and OpenGL. Nvidia's GeForce FX and GeForce 6800 launched in 2002 and 2004 respectively were programmed with DX9, Cg programs and OpenGL as they featured configurable 32-bit floating point pixel fragment and vertex processors. These processors were highly multithreaded and hence required creation of threads and execution of thread program for every vertex and every pixel fragment. The processor core architecture of GeForce 6800 supported multiple GPU implementations with processors having different number of cores.

Developing the Cg language for programming GPUs provided a scalable parallel programming model for the programmable floating-point vertex and pixel-fragment processors of GeForce FX, GeForce 6800, and subsequent GPUs. A Cg program resembles a C program for a single thread that draws a single vertex or single pixel. The multithreaded GPU created independent threads that executed a shader program to draw every vertex and pixel fragment. In addition to rendering real-time graphics, programmers also used Cg to compute physical simulations and other general purpose GPU (GPGPU) computations. Hence even though early GPGPU computing programs were difficult to write because developers had to express non-graphics computations with a graphics application program interface (API) such as OpenGL, they achieved high performance.

F. Unified computing and graphics GPUs

In 2006, Nvidia introduced GeForce 8800 which featured unified graphics and computing GPU for the first time which was programmable in C with the CUDA parallel computing model, in addition to OpenGL and DX10. Its unified streaming processor cores executed computing threads for CUDA C programs and also executed geometry, vertex, and pixel shader threads for DX10 graphics programs. Also the Hardware multithreading allowed the GeForce 8800 to execute maximum of 12,288 threads simultaneously in 128 processor cores.

For the first time, the GeForce 8800 GPU used scalar thread processor matching standard scalar languages like C unlike other GPUs which used vector processors. Thus the need to manage vector registers and program vector operations was eliminated. Instructions to support C and other languages such as IEEE 754 floating-point arithmetic, integer arithmetic, and load/store memory access instructions with byte addressing were added. It also provided hardware and instructions to support five parallel computation, communication, and synchronization—including shared memory, thread arrays, and fast barrier synchronization.

G. GPU computing systems

Previously supercomputers were built by connecting multiple GPUs to computers and cluster of GPU computing nodes were assembled. Therefore, responding to the demands for GPU computing systems, Nvidia in 2007 introduced the Tesla C870, D870, and S870 GPU card, desk side, and rack mount GPU computing systems containing one, two, and four T8 GPUs. The T8 GPU performed on parallel computing as it was based on GeForce 8800 GPU.

The Tesla C1060 and S1070 were the second generation GPU computing systems which were introduced in 2008 and made use of the T10 GPU which was based on GeForce GTX 280. It featured 1-teraflop-per second peak single-precision floating-point rate, IEEE 754-2008 double-precision 64-bit floating-point arithmetic, 240 processor cores and 4GB of DRAM memory. The Tesla family was basically designed for high performance computing systems in production and research.

The third generation of GPU architecture introduced by Nvidia in 2009 was the Fermi GPU computing architecture. It addressed several key areas in order to make GPU computing more applicable. Fermi consisted of increased double precision format as it implemented IEEE 754-2008. It also added 64-bit unified addressing, Error-Correcting Code (ECC) memory protection for GPU computing on a large-scale and instructions for Fortran, C, C++, Direct Compute and OpenCL.

H. Kepler GK110 Architecture[6]:

It consisted a total of 7.1 billion transistors, the Kepler GK110 was the fastest as well as one of the most complex GPU architecture. Many new innovative features were added to focus on computing performance. The GK110 was designed for the Tesla family as a parallel processing powerhouse. It provided a double precision throughput of 1TFlop with greater than 80% Double-Precision matrix-matrix multiplication (DGEMM) efficiency versus 60% on the Fermi architecture.

The new features included in Kepler GK110 which allowed increased utilization of GPU and simplified designing of parallel programming are:

- *Dynamic Parallelism* – It added the capability for the GPU to generate its own work for itself, control the scheduling of the work with the help of dedicated and accelerated hardware paths, and synchronize on the results, all without engaging the CPU which resulted in performing complex tasks and program implementation became easier.
- *Hyper-Q* – With the help of Hyper-Q multiple CPU cores could initiate work on a single GPU concurrently thus drastically increasing the utilization of the GPU and significantly reducing the CPU idle time.

- *Grid Management Unit* –A flexible, advanced grid management and dispatch control system was required for enabling powerful runtimes such as Dynamic Parallelism. Thus the GK110 comprised of a new Grid Management Unit which could prioritize and manage grids which are to be executed on the GPU. It ensured that the workloads generated by both CPU and GPU are managed and dispatched properly.
- *NVIDIA GPUDirect* – It enabled the direct exchange of data between GPUs on a single computer or GPUs located at different servers across a network, without the data actually passing through the CPU. It reduced the demand on system memory bandwidth and freed the GPUs Direct Memory Access (DMA) engines so that other CUDA tasks could use them.

I. The GCN architecture:

In the last two decades [7], the GPUs have evolved gradually and are on the verge of becoming an essential part of the computing environment. The initial GPU architecture designs featured special purpose hardware with some flexibility. The later architectures implemented less programming through shader programs, and finally became highly programmable computing devices. The Key industry standards such as Direct Compute, OpenCL and C++ have made GPUs accessible to programmers.

The GCN architecture was also optimised for power efficiency. The Heart of the GCN comprised of the new Compute Unit (CU) design that was used in the shader array. It is described in the Following sections:

- *Compute Unit (CU)* — These are the basic building blocks of the GCN architecture.
- *Scalar Execution and Control Flow* — For the efficiency of performance and power it was essential to introduce scalar pipelines in the GCN Compute Unit. It was mainly beneficial for general purpose computations which normally have more complicated control flow than graphics. Another scalar pipeline comprised of a full integer ALU that worked as an address generating unit (AGU) in order to read the scalar data cache. The actual ALU was 64 bit wide.
- *Vector Execution* —Majority of the GCN’s processing power came from the highly parallel SIMD’s (Single Instruction Multiple Data) which basically performed the calculations for general purpose computing and very fine graphics.
- *GCN Quad SIMD* —The SIMD’s in each CU have a combination of private and shared resources for efficiency. The instruction buffering, registers and vector ALU’s are private for each of the 4 SIMD’s to sustain high utilization and performance. The other resources such as branch unit, front-end and data cache are shared between the SIMD’s to achieve power and area efficiency.
- *Vector Registers* — Simple and high performance vector General Purpose Register (vGPR’s) file design was one of the biggest advantages of GCN’s architecture. They consisted of 64 lanes which are 32-bits wide. The adjacent vector registers could be combined for 64-bit or 128-bit data. Thus these vGPR’s eliminate port conflicts and thereby simplify compilers job.

J. The MaxwellArchitecture:

NVIDIA’s GM204 [8] is the first GPU to use the full realization of its 10th generation GPU architecture, Maxwell. The GM204 was designed to achieve two major goals:

- *Extraordinary Gaming Performance for the Latest Displays* **—** Screen resolutions are growing, with the first 4K displays for gamers hitting the market a year ago. In order to drive all these 4k pixels, GM204 GPUs were designed to deliver more performance at the highest resolutions, and also support new display tech for 4K like HDMI 2.0. In addition to the Dynamic Super Resolution, Maxwell introduced the ability to significantly enhance display quality on sub-4K displays by rendering at higher resolution and then applying an advanced resizing filter that leverages the high resolution and then applying an advanced resizing filter that leverages the high resolution information.
- *Incredible Energy Efficiency* — The Maxwell architecture was designed in such a way that it provided extra ordinary power efficiency and delivered unrivalled performance while simultaneously reducing power consumption from the previous generations of GPUs. The GPU’s based on Maxwell architecture were able to deliver 2x the performance per watt compared to Kepler based GPUs.

The Nvidia GeForce GTX 980 launched in 2014 featured four 64-bit memory controllers (256 bit total). Each memory controller was connected to 16 ROP (Render Output) units and consisted of 512 kb of L2 cache. The full chip comprised a total of 64 ROPs and 2048 KB of L2 cache where as GK104 (Kepler architecture) consisted of 32 ROPs and 512 KB of L2 cache. The GeForce GTX 980 was the first GPU with Maxwell architecture with 2048 CUDA Cores and 4GB of GDDR5 memory. Today GeForce Titan X which is also based on Maxwell architecture consists of 3072 CUDA Cores and 12 GB of GDDR5 memory. Thus the Maxwell based GPUs are not only the fastest GPUs in the world but also the most efficient of all.

III. COMPARATIVE ANALYSIS

Comparison between Maxwell architecture and its Preceders

Though the design of Kepler SMX (Next Generation Streaming Multiprocessor) architecture was exceptionally efficient for its generation, the GPU architects at Nvidia managed to design an even better architecture. This architecture was the Maxwell architecture and comprised of a completely new design for the Streaming Multiprocessor (SM) which improved the power efficiency drastically. The Maxwell SM far exceeded the Kepler SMX in all aspects as it consisted of improved control logic partitioning, clock-gating granularity, workload balancing, number of instruction issued per clock cycle, instruction scheduling and many other enhancements. In the new Maxwell SM architecture, the number of SMs increased to five whereas in the Kepler architecture it was only two.

Rapid Increase in the Transistor Count—

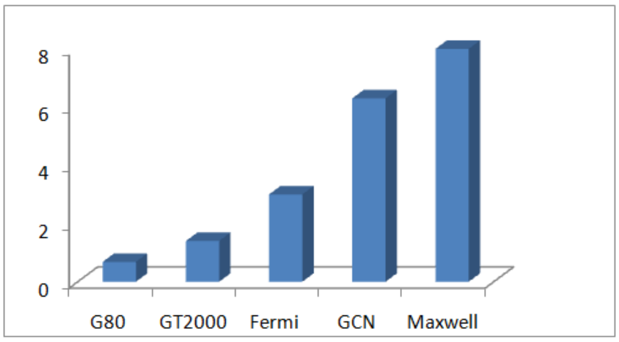


Fig 2 GPU transistors comparison

The above Figure 4 indicates that the number of transistors is increasing rapidly with time which results in the increased performance of the GPUs.

Increased CUDA Cores—

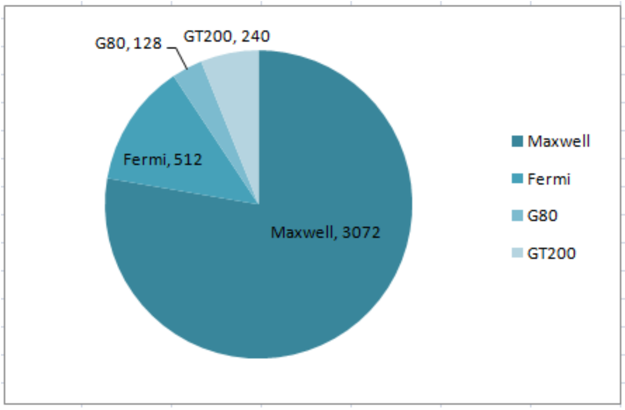


Fig 3 GPU CUDA Core count

Generally, a CPU is not re-programmable for certain functions. However, a GPU is basically a big processor which is programmable. A CUDA core is a core that can be programmed as per the needs of the programmer. Like a video game will use some of the cores for physics calculation, some for texture processing, etc. whereas video encoding software may use all the cores and optimize them for encoding video.

Eight Special Function Units—

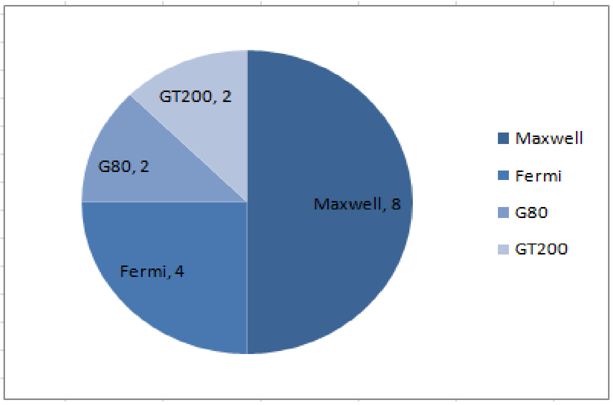


Fig 4 Special Function Units Comparison

The complex numerical instructions such as sine, cosine, reciprocal, square root are executed by Special Function Units (SFUs) in the GPU. The Figure 6 shows that GPUs based on Maxwell architecture consist of 8 such SFUs. Each Special Function Unit executes one instruction per thread per clock. As the SFU pipeline is separated from the dispatch unit, the dispatch unit is able to issue to other execution units even when the SFU is engaged.

The Maxwell Memory Subsystem—

In the GTX Titan X (Maxwell architecture), 16 ROP (Render Output) units are present in one ROP partition whereas in Kepler architecture 8 ROP units are present in each partition. Each ROP is capable of processing a single colour sample. With six such ROP partitions, a total of 96 ROP units are present in the Titan X, thrice the ROPs of its predecessor, thereby improving the ROP throughput dramatically.

The GTX Titan X (Maxwell) has a384-bit memory interface with 12GB GDDR5 memory, the fastest in the industry. It also features a unified 3072 KB L2 cache memory which is shared across the GPU and enhanced the memory compression implementation.

Quad Warp Scheduler—

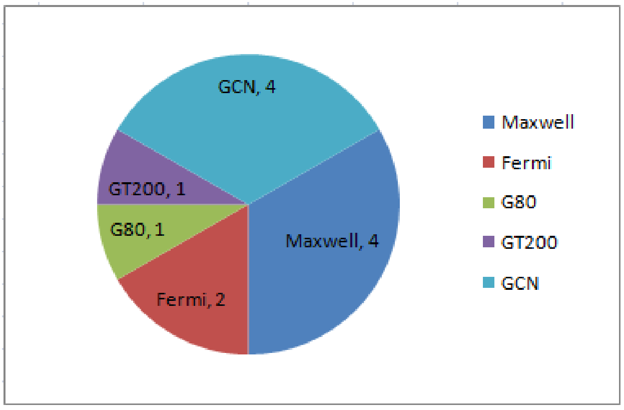
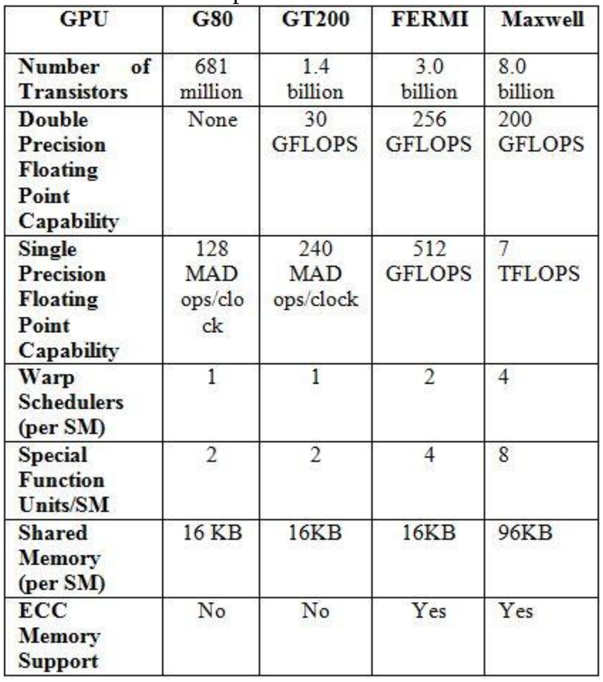


Fig 5 Warp Scheduler Comparison

Warps are groups of 32 parallel threads which are scheduled by the SM. In Maxwell as each SM consists of four warp schedulers, four warps can be issued and executed simultaneously. The Maxwell's Quad warp scheduler picks out two warps and assigns one instruction from each warp to a group of 8 Special Function Units (SFUs). Maxwell's schedulers do not check for dependencies from within the instruction stream since warps execute independently. As a result of this Maxwell achieves high hardware performance.

Overall Comparison of Various GPU's—

Table III Comparison GPU Architectures



GPU	G80	GT200	FERMI	Maxwell
Number of Transistors	681 million	1.4 billion	3.0 billion	8.0 billion
Double Precision Floating Point Capability	None	30 GFLOPS	256 GFLOPS	200 GFLOPS
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops/clock	512 GFLOPS	7 TFLOPS
Warp Schedulers (per SM)	1	1	2	4
Special Function Units/SM	2	2	4	8
Shared Memory (per SM)	16 KB	16KB	16KB	96KB
ECC Memory Support	No	No	Yes	Yes

Considering all the comparisons done and from the above table 3 we can say that the GPUs based on Maxwell architecture give the best performance (200 GFLOPS for Double Precision and 7 TFLOPS for single precision).

IV. PROPOSED WORK

Today computer architecture is dependent on CPUs. Thin Client System or Servers use very powerful CPUs and bigger RAM to provide enough space for each virtual machine and a bigger secondary storage for file storage of each system. Mostly Intel CPUs are used in such systems.

A general small organization system consists of a CPU with following configuration:
1) i7 processor quad core dual thread 2.2 GHz
2) 12 GB RAM
3) 5 TB Had Drive

With such configuration of CPU 15-20 thin clients can be connected according to requirement and performance of the system. This reduces hardware cost 10 times as the conventional fat client systems. Such systems are very useful in educational organization as well as higher educational institutes, where computers are very necessary.

As we have discussed before and from Table 1 we can say that GPUs are way better than CPUs in performance as well as accuracy and cost. Now let us discuss if we could implement GPUs in Thin Client Architecture. GPUs were made for geometric or graphical calculations but later it has been observed that GPUs can be used for the general purpose computing as well. So, the answer is yes, yes we can implement the GPU in thin client architecture. Now let's see how it can be achieved and advantages of GPU in thin client.

Performance of a computer is calculated in FLOPs, which determines the capability of the system to do work in particular time period. Same thing applies for the virtual machine. The performance of the Host is divided into its clients so as much faster the host will perform that much faster the client will perform in the ratio.

In general, the performance of a computer in FLOPS is given as:

$$Performance(FLOPS) = no.of\ cores * clock\,speed * \frac{FLOPs}{cycle}$$

Most processors today can do 4 FLOPs per clock cycle. Therefore, an average quad-core 2.5 GHz processor has a theoretical performance of 40 billion FLOPS = 40 GFLOPS.

These 40 GFLOPS are divided into each of the Thin Client connected to the Host. General purpose computing such as Internet Browsing, Video Streaming, Photo Editing, Document Generating etc. Require near about 1 GFLOP. That means 20 GFLOPS Host can serve 20 Thin Clients.

Hence we can generalize the result as:

$$N \propto P$$
$$\therefore N = k*P$$

Where k: Constant = 1
N: Number of thin clients
P: Performance (in FLOPS)

According to the table 1 and 2, Nvidia GeForce GTX Titan XGPU gives performance of 200 GFLOPS depending upon the operations performed. So as we need just 1 GFLOP for thin client approximately 200 clients can be connected to single host which is 5 times greater than CPU hosts.

Cost of the i7 processor and cost of the GeForce GTX Titan X is nearly equal. So using GPU in thin client system will not reduce the initial cost but will increase the number of thin clients, performance as well as accuracy.

V. CONCLUSION AND FUTURE WORK

A GPU has increased its demand in various scientific applications and high performance computing. A survey on various GPU architectures in terms of their evolution in past, number of cores, their computing capability, parallelism, multithreading etc. indicated that Maxwell architecture is highest performance architecture. Our proposed work indicates the application of General Purpose Graphics Processing Unit computing in Thin Computing. A Thin computing is very useful and cost efficient as well as energy efficient solution for organization and institutions, our analytical results proved that using the GPU architecture in thin computing allows more number of clients which result in decreased execution time and enhanced accuracy.

The proposed work can be implemented in an area where high performance with low cost is expected.

REFERENCES

[1] http://en.wikipedia.org/wiki/Thin_Client.php
[2] https://msdn.microsoft.com/en-us/library/ms927515.aspx
[3] www.fmslib.com/mkt/gpus.html
[4] J. Montrym and H. Moreton, ''The GeForce 6800,''IEEE Micro by IEEE Computer Society, vol. 25, no. 2, Page: 41
[5] cs.nyu.edu/courses/spring12/CSCI-GA.3033-012/the-gpu-computing-era.pdf
[6] http://www.nvidia.in/content/PDF/kepler/NVIDIA-Kepler-GK110-Architectutre-Whitepapaer.pdf
[7] https://www.amd.com/Documents/GCN_Architectutre_whitepaper.pdf
[8] https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_980_Whitepaper_FINAL.PDF