International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 06 | June - 2021 ISSN: 2582-3930

PERFORMANCE EVALUATION OF CPU-GPU WITH CUDA ARCHITECTURE

Aashish Patil[1], Yash Patel[2], Chetankumar Patil[3], Shruti Thorat[4] , Prof.Dr.Aarti A. Agarkar[5]

[1234] Department of Computer Engineering, Marathwada Mitra Mandal's College of Engineering, SPPU, Pune, India

---***---

Abstract - *Rendering is a process of generating image either as 2D or 3D models. For rendering, OpenGL APIs are used to maintain cross platform compatibility. A simple rendering of computer graphics would be using OpenGL with the help of textures and lighting. The parallel processing capability of Graphics Processing Unit (GPU) can be exploited by dividing computing tasks into 100s of smaller tasks that can be run concurrently. There are various ways to compare CPUs and GPUs. Traditionally they are compared on the basis of their computational power, Rendering is one of the way which can compare their capabilities. We can generate various graphical effects which requires high computation power and various CPUs, GPUs and CPU-GPU can be compared.*

Key Words:- **Rendering, Benchmarking, Central Processing Unit(CPU), Graphics Processing Unit(GPU), Compute Unified Device Architecture(CUDA), Parallel Programming.**

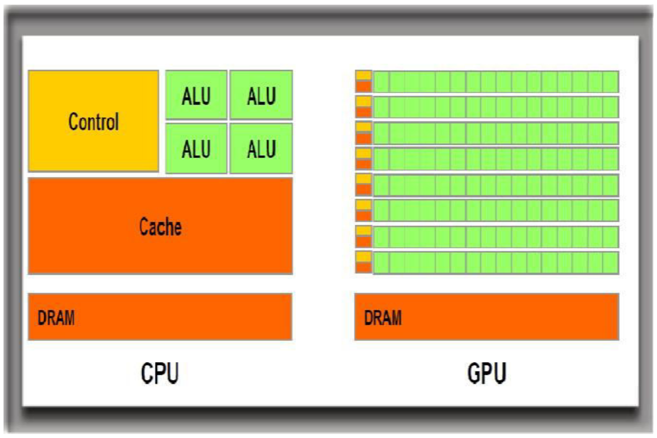
1. INTRODUCTION

In recent years, more and more multi-core/many-core processors are superseding sequential ones. Increasing parallelism, rather than increasing clock rate, has become the primary engine of processor performance growth, and this trend is likely to continue. Particularly, today's GPUs (Graphic Processing Units), greatly outperforming CPUs in arithmetic throughput and memory bandwidth, can use hundreds of parallel processor cores to execute tens of thousands of parallel threads. Researchers and developers are becoming increasingly interested in harnessing this power for general purpose computing, an effort known collectively as GPGPU (for "General-Purpose computing on the GPU"), to rapidly solve large problems with substantial inherent parallelism. Due to this large performance potential, GPU programming models have evolved from high-level shading languages such as Cg, HLSL, and GLSL to modern programming languages, alleviating programmers' burden and thus enabling GPUs to gain more popularity. Particularly, the release of CUDA (Compute Unified Device Architecture) by NVIDIA in 2006 has eliminated the need of using the graphics APIs for computing applications, pushing GPU computing to more extensive use. Likewise, APP (Advanced Parallel Processing) is a programming framework which enables ATI's GPUs, working together with the CPUs, to accelerate many applications beyond just graphics. All these programming frameworks allow programmers to develop a GPU computing application without mastering graphic terms, and enables them to build large applications easier.

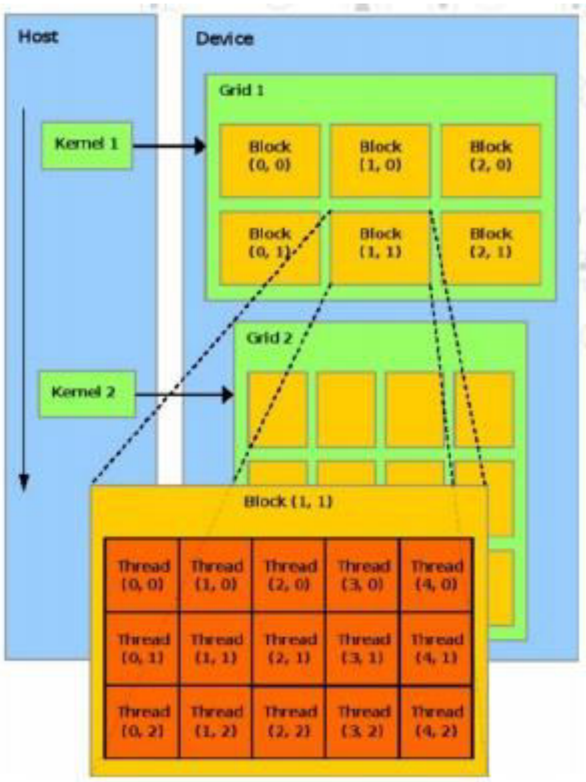
2. MOTIVATION

The world of high performance computing is a rapidly evolving field of study. Many options are open to business when designing a product. GPUs can provide astonishing performance using the hundreds of cores available. The field of high performance scientific computing lies at the crosswords of a number of disciplines and skill sets, Many IT industries have a HPP capable system yet they don't exploit the system completely which could give massive computational power.To harness the parallel computing power of GPUs, programmers can simply modify the performance critical portions of an application to take advantage of the hundreds of parallel cores in the GPU. The rest of the application remains the same, making the most efficient use of all cores in the system.

3. CUDA Architecture



CUDA is NVIDIA's parallel computing architecture which enables dramatic increases in computing performance by harnessing the power of the GPU (graphics processing unit).The programmer can choose to express the parallelism in high-level languages such as C, C++, FORTRAN or open standards as OpenACC Directives. The CUDA parallel computing platform is now widely deployed with 1000s of GPU-accelerated Applications shown below.

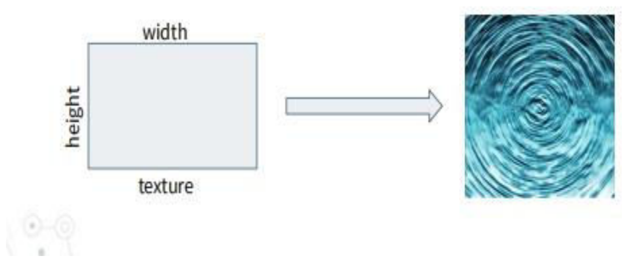


- CUDA architecture divides workload in thread, blocks and grids for parallel execution.
- CUDA supports Data Parallelism rather than Task Parallelism.
- System is referred as Host and GPU is referred as Device in CUDA architecture.
- Function executed by each thread in blocks and grids is known as kernels.
- A Separate barrier synchronization is provided by CUDA Architecture for advanced algorithms accessing same memory location throughout the kernel.
- We need to divide our task efficiently for parallel execution else simple mistakes

may cause data races which would result in termination.

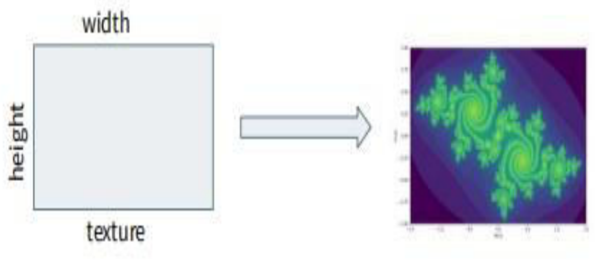
1. **Ripple Effect Simulation**:

A sine wave is a geometric waveform that oscillates (moves up,down or side-to-side) periodically, and is defined by the function y = sin x.



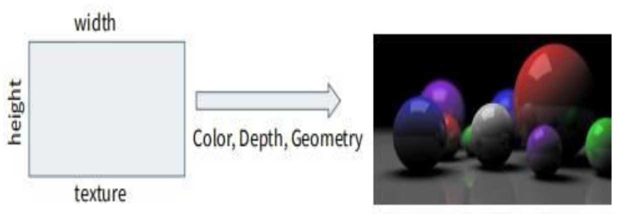
2. **Julia Set Animation:**

In the context of complex dynamics, the Julia set is defined from function $f_c(z) = z^2 + c$, where c is a complex parameter.Julia set fractals are normally generated by initializing a complex number z = x + yi where i2 = -1 and x and y are image pixel coordinates in the range of about -2 to 2. Then, z is repeatedly updated using: z = z2 + c where c is another complex number that gives a specific Julia set.



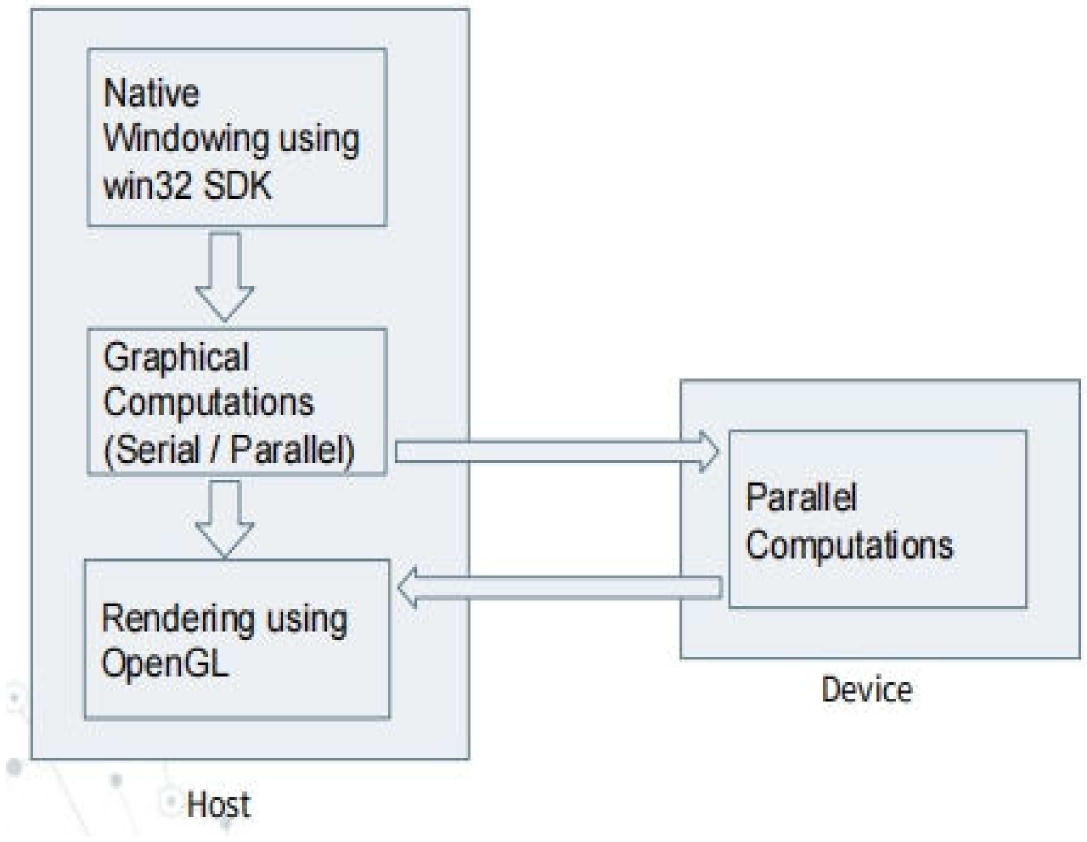
3. **Ray Tracing:**

In 3D computer graphics, ray tracing is a rendering technique for generating an image by tracing the path of light as pixels in an image plane and simulating the effects of its encounters with virtual objects.

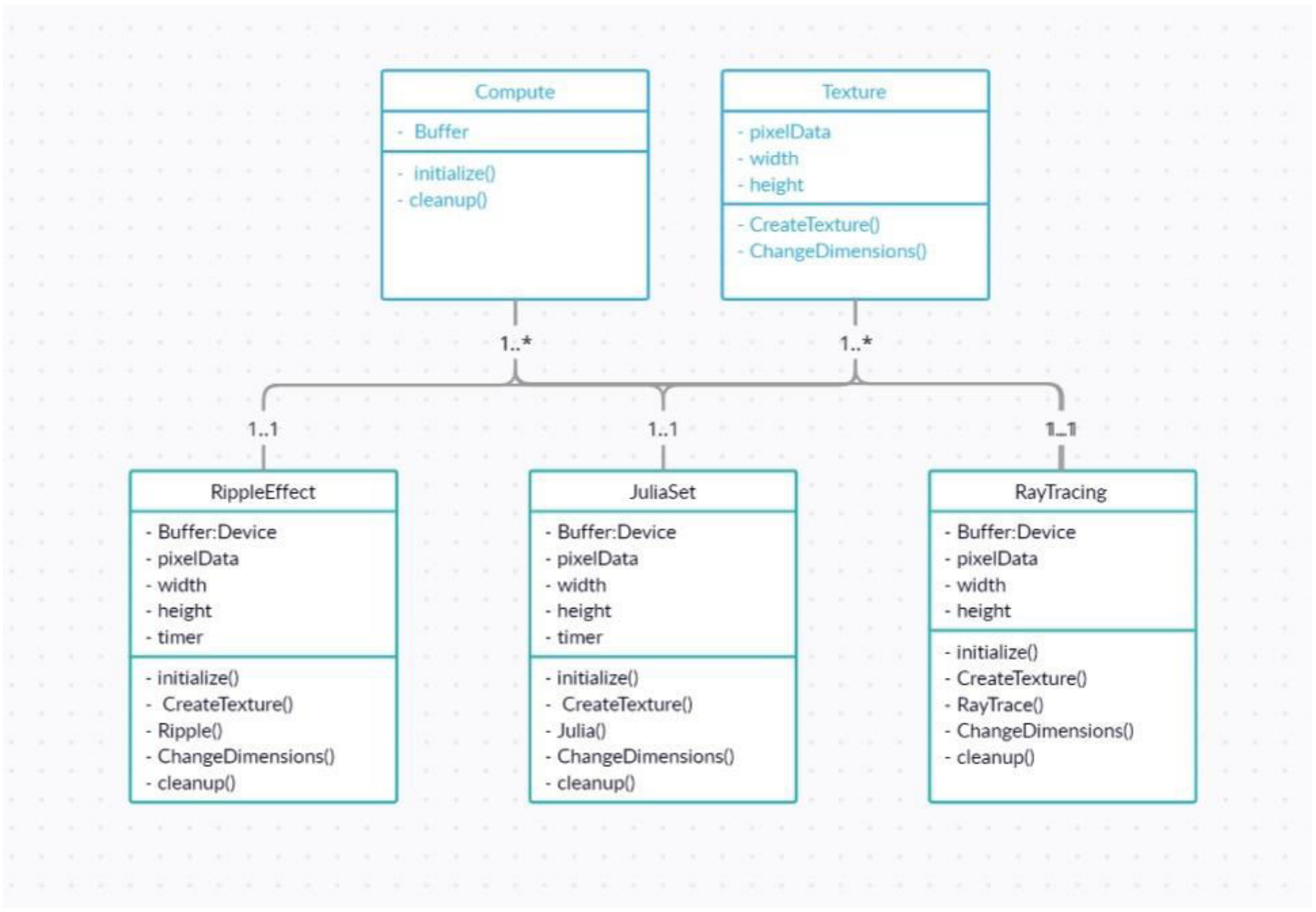


4. FIGURES

1. System Architecture:


Native Windowing using win32 SDK
Graphical Computations (Serial / Parallel)
Rendering using OpenGL
Host
Parallel Computations
Device

2. UML Class Diagram:



5. PROS AND CONS

Advantages

•Gain an independent perspective about how is your system's performance compared to others.

• Drill down into performance gaps to identify areas for improvement(Especially for manufacturers)in the next generation processors.

• Develop a standardized set of processes.

• Set Performance expectations.

• User does not need additional hardware or software

Limitations

• Currently executable on Windows platform only.

• Closed CUDA architecture, it belongs to NVIDIA.

• Will work specifically only on NVIDIA GPU's.

CONCLUSION:

The project has high feasibility due to technical resources are adequate and all the

requirements are met. As it has low user-end requirements, well documented tools and

technologies, the final product can be easily operated by some configuration. It also has a lot of future scope and extendability. In the current pandemic situation the world has inclined more towards online computation like in daily transactions,educational,various services. Parallel programming can play a big role in increasing compuation speed as well as the quality of virtual world.

6. FUTURE WORK:

• Trying Multi-GPU parallelisation.

• Implementing benchmarking on different platforms like linux, mac, android.

• Use of multiple streams on a single GPU so that the GPU can perform the kernel

computation and memory transfer in an asynchronous way in order to further hides the

latency.

• Integrating OpenCL benchmarking in same project.

REFERENCES:

[1] Performance Analysis of Ray Tracing Based Rendering Using OpenCL International

Conference on Innovations in Power and Advanced Computing Technologies *[i-PACT2017] Raju K., Dept. of Computer Science, NMAMIT, Nitte, India, rajuk@nitte.edu.in.*

[2] A Comprehensive Performance Comparison of CUDA and OpenCL 2011 International Conference on Parallel Processing *Jianbin Fang, Ana Lucia Varbanescu and Henk Sips Parallel*

and Distributed Systems Group Delft University of Technology Delft, the Netherlands Email: {j.fang, a.l.varbanescu, h.j.sips}@tudelft.nl.

[3] Performance Evaluation of Advanced Features in CUDA Unified Memory -2019 IEEE/ACM Workshop on Memory Centric High Performance Computing *Steven W. D. Chien, KTH Royal Institute of Technology Stockholm, Sweden Ivy B. Peng, Lawrence Livermore National Laboratory Livermore, USA Stefano Markidis, KTH Royal Institute of Technology Stockholm, Sweden*