Comparative Analysis for Time Complexity of Sieve Benchmark algorithm on CPU & GPUs

M. S. Joshi[1]*, S. L. Kasar[1] and A. S. Mundaware[2]

Department of Computer Science & Engineering

MGM's Jawaharlal Nehru Engineering College, Aurangabad

Mahatma Gandhi Mission (MGM), N-6, CIDCO, Aurangabad - 431003, Maharashtra, INDIA

[2]Afour Technologies Pune, Maharashtra, INDIA

***Corresponding Author:** E-mail: madhuris.joshi@gmail.com Tel.: +91 240 2482893 , Fax:+91 240 2482232

ABSTRACT

Problem solving is a major task carried out with the help of computers. Time required to solve a particular problem usually depends on the amount of data & the operations involved. The complex problems with huge data and/or large computation involved requires lot of time when executed on CPU. But if the same problem is accelerated by shifting complex & large computation to GPUs, the execution time is reduced. In this paper the kernel benchmark 'Sieve of Eratosthenes' algorithm is implemented using CUDA C language & executed on different GPUs (Quadro 6000,Tesla C2075, GeForce GTX 480) as well as on CPU (INTEL i7 4790K). The results are quite promising with GPU as compared to CPU. As the data grows time required to execute algorithm on CPU increases.

***Keywords**: CPU, GPU, CUDA, Quadro, Tesla, GeForce GTX, Fermi, Sieve*

1. Introduction

The necessity of using GPU is still questionable, because we can use multiple CPU motherboard and use two CPUs at a time. In this case one CPU will manage all the basic computer processes and other one will completely work on graphics rendering. This is partially true, since CPU can't manage high graphics rendering. The graphics rendering can be easily managed by GPU , but since GPU can't do the tasks like working on spreadsheets, listening, web browsing, multitasking like playing games while listening to music. Also the GPU is specialized in domains like Graphics rendering and larger calculation simultaneously. The graphics processing unit (GPU) has become an integral part of today's mainstream computing systems. The GPU's rapid increase in both programmability and capability has spawned a research community that has successfully mapped a broad range of computationally demanding, complex problems to the GPU[13]. GPU-accelerated computing is the use of a graphics processing unit (GPU) together with a CPU to accelerate scientific, analytics, engineering, consumer, and enterprise applications. Pioneered in 2007 by NVIDIA, GPU accelerators now power energy-efficient datacenters in government labs, universities, enterprises, and small-and-medium businesses around the world [14]. GPUs are accelerating applications in platforms ranging from cars, to mobile phones and tablets, to drones and robots. GPU-accelerated computing offers unprecedented application performance by offloading compute-intensive portions of the application to the GPU, while the remainder of the code still runs on the CPU. From a user's perspective, applications simply run significantly faster [1]. The working of GPU accelerated computing is shown in the Fig.1.

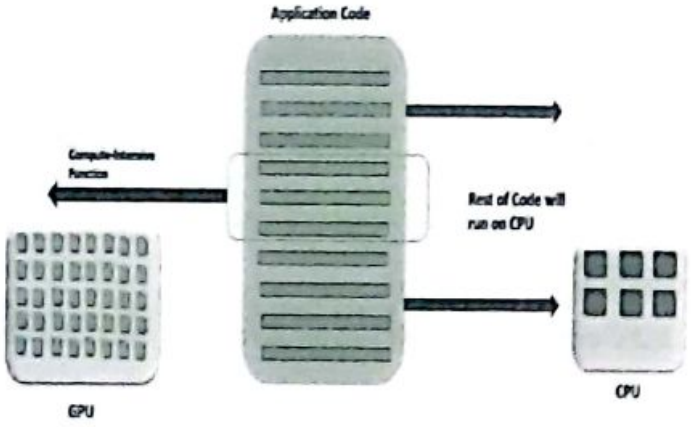

Fig. 1. Working of GPU Acceleration

1.1 Comparison of GPU & CPU

A simple way to understand the difference between a CPU and GPU is to compare how they process tasks.

A CPU consists of a few cores optimized for sequential serial processing while a GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously. GPUs have thousands of cores to process parallel workloads efficiently [13].

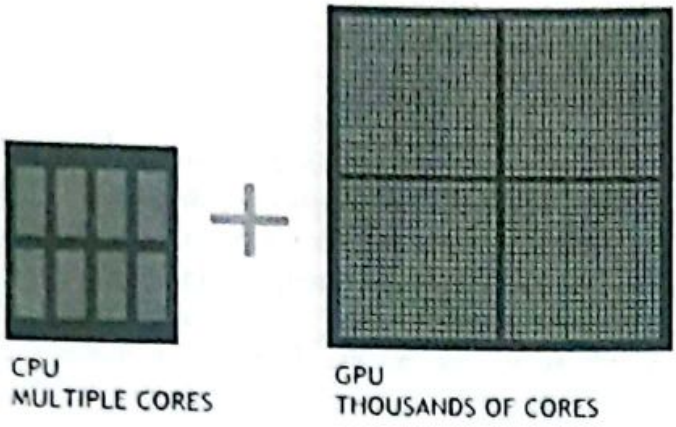

Fig. 2. Cores in CPU & GPU

A GPU is a multiprocessor, sometimes containing hundreds of processors. The intended purpose of a GPU is to perform graphics operations, which is what they do well, but they can be used for other computations as well. GPUs do not perform all of the operations that a CPU can perform; their instruction sets are narrowly focused on graphics acceleration. The programming interfaces to GPUs are high-level application programming interfaces such as OpenGL and DirectX, together with high-level graphics shading languages such as C for Graphics (Cg) and the High Level Shader Language (HLSL). These languages are supported by compilers that generate intermediate languages, which are optimized by the speci_c GPU driver software, which generates the speci_c machine instructions for the GPU. There is much more data parallelism in graphics applications than general purpose applications [7].

1.2 Compute Unified Device Architecture (CUDA)

CUDA is a parallel computing platform and programming model that makes using a GPU for general purpose computing simple and elegant. The developer still programs in the familiar C, C++, FORTRAN, or an ever expanding list of supported languages, and incorporates extensions of these languages in the form of a few basic keywords. These keywords let the developer express massive amounts of parallelism and direct the compiler to the portion of the application that maps to the GPU [10].

1.3 OpenACC:

OpenACC is a programming standard for parallel computing developed by Cray, CAPS, Nvidia and PGI. The standard is designed to simplify parallel programming of heterogeneous CPU/GPU systems. Like in OpenMP, the programmer can annotate C, C++ and FORTRAN source code to identify the areas that should be accelerated using compiler directives and additional functions. Like OpenMP 4.0 and newer, code can be started on both the CPU and GPU. OpenACC members have worked as members of the OpenMP standard group to merge into OpenMP specification to create a common specification which extends OpenMP to support accelerators in a future release of OpenMP [14].

2. Methodology

The proposed work is as shown the Fig. 3. The Sieve algorithm is used to analyze the computational efficiency and time complexity of CPU and GPU. The Sieve algorithm was executed on CPU as well as each of GPUs NVidia Tesla, Quadro and GTX.

2.1 Sieve of Eratosthenes

To test the performance of CPU and GPU there are many Benchmark algorithms. This paper focuses on Sieve of Eratosthenes Algorithm. In mathematics, the Sieve of Eratosthenes, one of a number of prime number sieves, is a simple, ancient algorithm for finding all prime numbers up to any given limit. It does so by iteratively marking as composite (i.e., not prime) the multiples of each prime, starting with the multiples of 2. The multiples of a given prime are generated as a sequence of numbers starting from that prime, with constant difference between them that is equal to that prime [3]. This is the sieve's key distinction from using trial division to sequentially test each candidate number for divisibility by each prime [4]. The sieve of Eratosthenes is one of the most efficient ways to find all of the smaller primes. It is named after Eratosthenes of Cyrene, a Greek mathematician.

The sieve kernel has been used to compare microprocessors, personal computers, and high-level

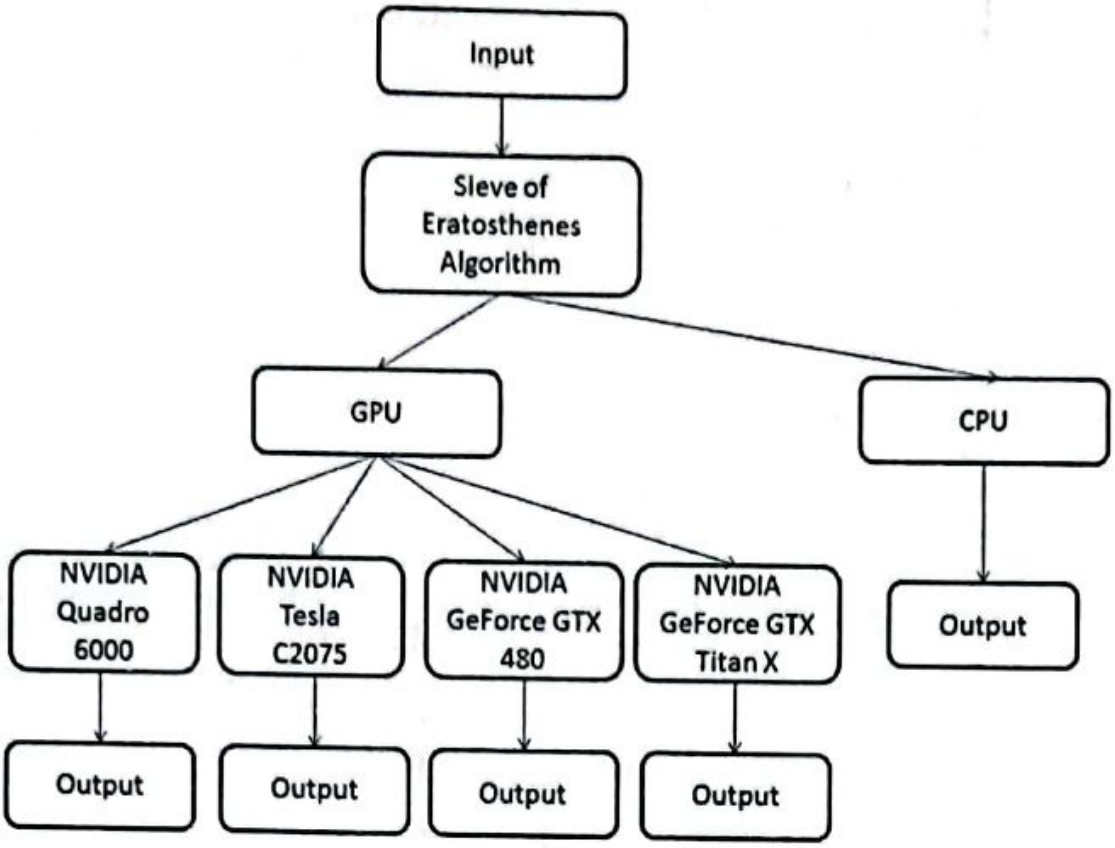

Fig 3. System workflow

languages. It is based on Eratosthenes' sieve algorithm and is used to find all prime numbers below a given number n.

The Sieve algorithm is as follows
Input: Take an integer value, $n > 1$
Let A be an array of Boolean values, initially all set to true.

for $i = 2, 3, 4, ...,$ not exceeding $\sqrt{n}$
if $A[i]$ **is true**:
for $j = i^2, i^2+i, i^2+2i, i^2+3i, ...,$ not exceeding n
$A[j]$:= **false**

Output: all i such that $A[i]$ **is true**.

2.2 GPUs

2.2.1 Tesla C2075

Tesla is a series of graphics card developed NVIDIA. By adding a Tesla processor, engineers, designers, and content creation professionals accelerate some of the most complex tools exponentially faster than by adding a second CPU. Tesla is used for analysis, simulation, and rendering tools within industry-leading applications and see results in as little as half the time. Tesla c2075 have 448 CUDA Cores. Total Dedicated Memory is 6GB (GDDR5).

2.2.2 Quadro 6000

Quadro is a series of graphics card developed NVIDIA. It is used for CAD CAM, video processing and computational fluid dynamics. Quadro 6000 have 448 CUDA Cores. Total Dedicated Memory is 6GB (GDDR5).

2.2.3 GeForce GTX 480

GeForce GTX is series of graphics card developed by NVIDIA. It is specially used for gaming purpose and has 480 CUDA cores. Standard Memory Configuration is 1536 MB and Memory Interface used is GDDR5.

2.2.4 GeForce GTX TITAN X

GeForce GTX is series of graphics card developed by NVIDIA. It is specially used for gaming purpose and has 3072 CUDA cores. Standard Memory Configuration is 12 GB and Memory Interface used is GDDR5.

All the GPUs used in this paper uses Fermi, Tesla & Maxwell Architecture.

2.3 Microarchitecture used in GPUs

2.3.1 Fermi Architecture

The Fermi architecture is the most significant leap forward in GPU architecture since the original G80. G80 was our initial vision of what a unified graphics and computing parallel processor should look like. GT200 extended the performance and functionality of G80. With Fermi, we have taken all we have learned from the two prior processors and all the applications that were written for them, and employed a completely new approach to design to create the world's first computational GPU [8]. Fermi is the codename for a GPU microarchitecture developed by NVidia as the successor to the Tesla microarchitecture. It was the primary microarchitecture used in the GeForce GeForce 400 series and GeForce 500 series. It was followed by Kepler, and used alongside Kepler in the GeForce 600 series, GeForce 700 series, and GeForce 800 series, in the latter two only in mobile GPUs. In the workstation market, Fermi found use in the Quadro x000 series, Quadro NVS models, as well as in Nvidia Tesla computing modules. All desktop Fermi GPUs were manufactured in 40 nm, mobile Fermi GPUs in 40 nm and 28 nm. Fermi is the oldest microarchitecture from NVIDIA that can support DirectX 12 [7].

2.3.2 Tesla Architecture

Tesla is the codename for a GPU micro architecture developed by Nvidia as the successor to their prior microarchitectures. Tesla is Nvidia's first microarchitecture to implement unified shaders. It was used with GeForce 8 Series, GeForce 9 Series, GeForce 100 Series, GeForce 200 Series, and GeForce 300 Series of GPUs manufactured in 90 nm, 80 nm, 65 nm, and 55 nm. It also found use in the GeForce 405, and in the workstation market in the Quadro FX, Quadro x000, Quadro NVS series, and Nvidia Tesla computing modules.

2.3.3 Maxwell Architecture

It is the GPU microarchitecture developed by Nvidia as the successor to the Kepler microarchitecture. The Maxwell architecture was introduced in later models of the GeForce 700 series and is also used in theGeForce 800M series, GeForce 900 series, and Quadro Mxxx series, all manufactured in 28 nm

3. Results

This paper demonstrates time complexity calculated for Sieve Benchmark algorithm when executed on GPU & CPU. The configuration used is Processor INTEL i7 4790k (3.6 GHz), RAM 8GB, GPUs Quadro 6000, Tesla c2075, GeForce GTX 480, GeForce GTX TITAN

Table 1. Time Complexity for Sieve algorithm on CPU & GPUs

Processing unit	Time Complexity (ms) n is range of calculating prime number					
	n=100	n=500	n=1000	n=10000	n=50000	n=100000
CPU INTEL i7 3.6GHZ	7	19	41	145	287	461
GPU NVIDIA Quadro 6000	0.312	0.346	0.402	1.743	7.715	15.173
GPU NVIDIA Tesla C2075	0.292	0.297	0.367	1.713	7.689	15.15
GPU NVIDIA GeForce GTX 480	0.282	0.340	0.278	1.380	6.543	12.436
GPU NVIDIA GeForce GTX TITAN X	0.323	0.368	0.356	0.732	2.31	4.246
Block utilized by GPUs to compute the results	10	22	31	100	223	316

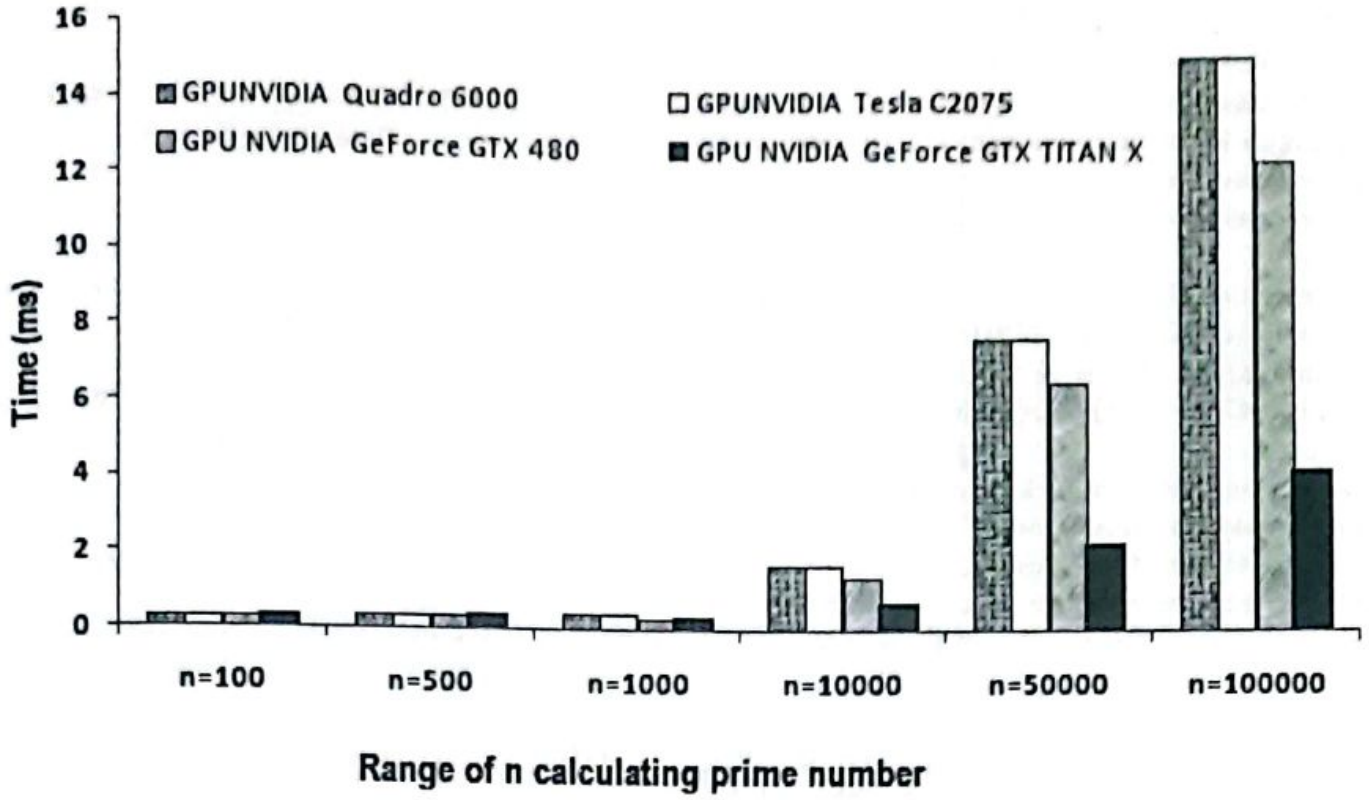

Fig. 4. Execution Time in ms of Sieve Algorithm for different values of n

X. Time complexity and number of blocks used are given in the table 1. There is significant difference between the time complexity of CPU and GPU. For n=100000 CPU requires 461 ms & Quadro 6000 requires 15.173 ms, Tesla c2075 requires 15.15 ms and GeForce GTX 480 requires 12.436 ms, GeForce GTX TITAN X requires the minimum time of 4.246 ms. Amongst all GPUs TITAN X requires least time as shown in fig. 4. In this figure the X axis maps the range n and Y axis denotes the time required in ms.

4. Discussion and Conclusion

Ever since the advent of Computer era Developer are trying to optimize relative performance of the CPUs , but to do this they need to know the extent of our CPU so we can fully utilize its peak performance.

4.1 Discussion

In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it. The term 'benchmark' is also mostly utilized for the purposes of elaborately designed benchmarking programs themselves.

Benchmarking is usually associated with assessing performance characteristics of computer hardware, for example, the floating point operation performance of a CPU, but there are circumstances when the technique is also applicable to software. Software benchmarks are, for example, run against compilers or database management systems.

Benchmarks provide a method of comparing the performance of various subsystems across different chip/system architectures.

Test suites are a type of system intended to assess the correctness of software.

Different types of benchmark includes real program, component benchmark / microbenchmark, Kernel, Synthetic Benchmark, I/O benchmarks, Database benchmarks: to measure the throughput and response times of database management systems (DBMS'), Parallel benchmarks: used on machines with multiple cores, processors or systems consisting of multiple machines.

4.2 Conclusion

The sieve of eratosthenes is a standard benchmark used to determine the relative speed of different computers or,

in this case, the efficiency of the code generated for the same computer by different compilers. The sieve algorithm was developed in ancient Greece and is one of a number of methods used to find prime numbers. The sieve works by a process of elimination using an array that starts with 2 and keeps all the numbers in position [9].

By analyzing the Time taken to execute Sieve of Eratosthenes Algorithm NVIDIA it is found that GeForce GTX TITAN X GPU was faster among INTEL i7 4790k (3.6 GHz), Quadro 6000, Tesla c2075.

The Kernel Benchmark algorithm Sieve of Eratosthenes was executed on CPU and different GPUs including Quadro 6000, Tesla c2075, GeForce GTX 480. The time required for execution with different value of n from 100 to 100000 is recorded for all executions. It is observed that the GPU gives better time complexity that is CPU requires more time as compared to GPUs. Also, in comparison of GPUs, TITAN X, Quadro 6000, Tesla C2075, GeForce GTX 480, GeForce GTX TITAN X performs better. For large value of n=100000 it takes minimum time of 4.246 ms.

Acknowledgments

We thank Nvidia Corporation for providing all the GPUs under GPU Education Center. This enabled us for the experimentation.

5. References

[1] www.nvidia.com accessed on Oct. 05, 2016

[2] Nicholas Wilt, The CUDA Handbook: A Comprehensive Guide to GPU Programming, 11-Jun-2013.

[3] Horsley, Rev. Samuel, F. R. S., The Sieve of Eratosthenes. Being an account of his method of finding all the Prime Numbers, *Philosophical Transactions* (1683–1775), Vol. 62. (1772), pp. 327–347.

[4] O'Neill, Melissa E., The Genuine Sieve of Eratosthenes, Journal of Functional Programming, Published online by Cambridge University Press 9 October 2008.

[5] Raj Jain, Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling, Wiley Computer Publishing, John Wiley & Sons, Inc. ISBN: 0471503363.

[6] Christopher Cullinan, Christopher Wyant, Timothy Frattesi Computing Performance Benchmarks among CPU, GPU and FPGA.

[7] NVIDIA Corporation, Whitepaper NVIDIA's Next Generation, CUDATM Compute Architecture: Fermi, 2009.

[9] www.keil.com accessed on Oct. 11, 2016

[10]David Luebke CUDA: Scalable parallel programming for high-performance scientific computing, 2008 5th IEEE International Symposium on Biomedical Imaging, Paris, 14-17 May 2008.

[11]Chao-Tung Yang, Hybrid CUDA, OpenMP, and MPI parallel programming on multicore GPU clusters, Computer Physics Communications, Volume 182, Issue 1, January 2011.

[12] Michael Garland, Scott Le Grand, John Nickolls, Joshua Anderson Jim Hardwick, Scott Morton, Everett Phillips, Yao Zhang, Vasily Volkov, Parallel Computing Experiences with CUDA, Issue 4 - July/ August (2008 vol.28).

[13] John D. Owens, GPU Computing, Proceedings of the IEEE (Volume: 96, Issue: 5), May 2008.

[14] Sandra Wienke, Paul Springer, Christain Terboven, Dieter and Mey, Open Acc - First Experiences with Real-World Applications, Euro-par 2012 Parallel Processing, Volume 7484 of the series Lecture Notes in Computer Science, 2012

[15] Henry Kasim, Verdi March, Rita Zhang, Simon See, Survey on Parallel programming Model, Network and Parallel Computing, Volume 5245 of the series Lecture Notes in Computer Science, 2008.