Comparative Study of Computationally Intensive Algorithms on CPU and GPU

Anala M.R  
Associate Professor, R V College of Engineering, Bangalore, India.

Kartik K.R  
Student, R V College of Engineering, Bangalore, India.

Madhusudhan Aithal M.  
Student, R V College of Engineering, Bangalore, India.

Jeevan D.C  
Student, R V College of Engineering, Bangalore, India.

Abstract  
This paper presents comparative study on one of the popular cryptographic algorithms AES algorithm, implemented using CUDA on GPU and on CPU. In present day scenario the AES algorithm suffers from very high CPU resource consumption, latency and low throughput. The AES cipher calculation is performed in blocks, with each block of size 128 bits. For a given plaintext, repetitions of transformation rounds are carried out in steps to get the cipher text. Unlike conventional method of CPU processing, the same is carried out in parallel by dividing the plain text into units of 16 bytes each followed by transformations pertaining to algorithm. The experiment is made on NVIDIA tesla K20 GPU accelerator and the same analysis is discussed in this paper.

Keywords: CUDA(Compute Unified Device Architecture); Cryptography; Parallel Programming; AES algorithm.

Introduction  
As the need to solve computationally intensive problems is on the rise, the time involved in solving these problems must reduce. This is accomplished with the upcoming technology of parallel programming. The mainstream parallel programming model such as Compute Unified Device Architecture (CUDA) [R5-1], is developed by NVIDIA and implemented only by the GPUs that they produce. Every processor in a multiprocessor of NVIDIA GPU has a set of 32bit local read/write registers. Shared memory is shared among all the processors and can be accessed in parallel [R16-2]. The comparison of performance in GPU and CPU with the help of an arithmetic computation is executed with different sizes of an array in [3]. Parallel programming is seen as cost effective method for easier and quicker results of computationally large and data-intensive problems. With the use of comparatively inexpensive parallel computers such as multiprocessors and cluster computing it has made parallel programming a must to the future of programming of very large data-set. Data-intensive applications such as encryption and decryption, data mining, transaction processing, information retrieval, and analysis provides an area of interest in the optimization using parallel platforms. Upcoming areas such as computational biology and nanotechnology have a wide requirement for algorithms and systems development, while changes in architectures, programming models and applications is going to have an impact on how parallel platforms are made available to users in the form of block services.

Cryptography:  
The process of protecting data from external sources by using encryption and decryption algorithms is known as cryptography. Cryptography in recent years has gained importance in commercial applications. The most important feature of the cryptographic process is the generation and storing of the encryption key. Implementation and analysis of RSA algorithm using GPU is discussed in [3]. Cryptography not only provides protection of data, but also provides data integrity, authentication. Cryptography enables only the intended person to decrypt the encrypted data thus ensuring data security. Certain challenges of various cryptographic algorithms include security, computational time and space complexity.

AES algorithm:  
Advanced Encryption Standard (AES) cryptographic algorithm is one of the most widely used cryptographic algorithms presently. Its original name is Rijndael algorithm which is developed by two cryptographers from Belgium, Joan Daemen and Vincent Rijmen. It is a symmetric key block cipher which means that the same key is used during the encryption and decryption of required data. It is generally used to secure Voice over Internet Protocol (VoIP) and has much more applications like in IPSec.

Current scenario of implementation of AES algorithm on CPUs:  
Security of data and information is an important and trending topic in the field of Computer Science nowadays. Currently data encryption using AES algorithm in many applications is done on the CPU where the encryption and decryption of data is done serially (serial execution of data).
Nowadays, the usage of GPU general purpose computation is steadily increasing. The usage of serial code for encryption and decryption has been found as inefficient and time consuming. Suppose we have a large volume of data to be encrypted and later decrypted, each time we can encrypt only 16 bytes of data using a key. This has to be repeated for remaining data to encrypt them as shown in Fig.1. If we perform the data encryption in this way, it is going to take large amount of time since we are executing serially. If the serial code is parallelized using CUDA to execute on GPU a noticeable reduction in the execution time of the encryption and decryption algorithm can be seen. This technique employed in GPUs is data level parallelism. In this paper we implement the AES algorithm for fast encryption of data thereby making the efficient use of GPU computing power.

**Computational power of GPUs:**

Graphic Processing Unit (GPU) is a specialized processor which has its own memory to perform required floating point operations for computer graphics. As we leap into talk on Computational capabilities of GPU, we hear Computer scientists today cite “Death of Moore’s law”. The main reason being constraints in speed, size, and energy dissipation. This difficulty is eyed through parallelism of the GPU as the only future for computing. A survey on the performance of different applications on GPU is discussed in [4]. A parallelized algorithm for RSA algorithm is designed using CUDA framework in[5] and the designed algorithm is realized for small prime numbers and large prime numbers. GPUs were originally designed by NVIDIA for graphic processing. Initially they were used for accelerating computer graphics and image processing, but currently they are being used for general purpose applications as GPGPUs (General Purpose GPUs) as well. They are generally used along with the CPU to increase the performance of a computer depending on the application.

CPUs are fast, a 3.2 GHz quad-core i7 with 24.6 GFLOPS are slow when compared to NVIDIA’s GeForceFX 7800 with 165 GFLOPs, 1066 MHz FSB Pentium Extreme Edition with 8.5 GB/s, ATI Radeon X850 XT Platinum Edition with 37.8 GB/s. GPUs are getting more advanced and faster with NVIDIA’s Kepler series touching 1.80 TFLOPS. With attribution of most efficient and fastest HPC ever built leads as the most architecturally complex microprocessor ever built. Adding towards its innovative features are dynamic parallelism, Hyper-Q, grid management unit, NVIDIA GPUDirect. With such computational capabilities and performances, GPUs have scaled the time from hours to minutes, minutes to seconds and seconds to nanoseconds. In response to this the applications vary in fields of Cryptography, Fluid dynamics, Bioinformatics and other fields where we can utilize extraordinary power of GPU.

**GPU programming using CUDA**

GPU is a throughput oriented processor where thousands of lightweight threads can be executed. This concept is different from the one which we see in CPU, which is a latency oriented device. The architecture used in GPU is Single Instruction Multiple Data (SIMD) where single instruction is executed on a number of streaming multiprocessors on different datasets. Generally a GPU consists of number of streaming multiprocessors, generally termed as SMs, which consists of number of blocks in which CUDA threads are executed. GPUs available today consist of hundreds of streaming processors. The AES algorithm is implemented in CUDA with Tesla K20 GPU card which consists of 2496 cores.

CUDA (Compute Unified Device Architecture) is a general purpose parallel computing platform and programming model created by NVIDIA. It enables the users to harness the computing performance of the Graphic Processing Units(GPUs) efficiently. CUDA has been successfully used as tool to accelerate the computations in mathematics, computational biology, cryptography and other applications. If there is an image to be processed or read, we have to process each and every pixel in the image for specified number of times per second. This has to be done simultaneously using multiple threads executing concurrently which is accomplished using GPUs. Some portion of the code which should be executed sequentially is executed on CPU since all codes cannot be parallelized and the code that is computationally intensive which can be parallelized is executed on GPU. In CUDA, the codes which are executed on GPU are written in the as functions known as kernels.
Initially we allocate the required memory for the data to be processed on the GPU from the CPU. While we call the kernel from the CPU we have to specify the block size, which is the number of threads that are allocated in a block of GPU in 3 or less dimensions for the execution of the kernel by the user and grid size, which is the number of blocks that are allocated in a grid of GPU by the user in 3 or less dimensions. When we call a kernel we launch a number of threads that are executing the same instruction (instructions present in kernel) simultaneously but on different datasets. As mentioned earlier, GPUs have large memory bandwidth due to which time taken for data transfer is less. The context switching in case of GPU takes less time. In CUDA, numerous threads launched will execute in multiple of warps. Warp is a group of 32 threads executing in parallel\[5\]. It is also possible to call kernel function from the GPU itself. With the addition of dynamic parallelism, it is possible to launch a number of concurrently executing threads from the GPU itself instead of launching those threads from the CPU.

**Implementation of AES algorithm on GPU using CUDA**

In AES encryption algorithm, data is encrypted in terms of blocks of 16 bytes at once but the key length varies. In 128 bit AES cryptographic algorithm, key length is 128 bits as shown in Fig 2. In 192 bit AES algorithm, length of the key is 192 bits and the key length is 256 bits in case of 256 bit AES cryptographic algorithm. We implemented the AES algorithm in CUDA such that the given text or data which has to be encrypted is divided into a number of groups of 16 bytes of data. The last group of data may contain less than 16 bytes of data. In the CUDA code for AES algorithm, we launch the required number of blocks of GPU threads where one thread in that block will encrypt 16 bytes of data. The data to be encrypted is stored in a 1D array. This array is copied to global memory of GPU where the thread in that block will encrypt the 16 bytes of data in that array based on their thread id. This array is copied from CPU memory to global memory of GPU using cudaMemcpy( ). The encrypted data is stored in this array by the threads in each block of GPU threads in, corresponding to the position in the array based on their thread id. This array containing output text or data which is encrypted is copied from GPU memory to CPU memory using cudaMemcpy( ). In this way, instead of encrypting the blocks of text serially one after the other, the blocks of text are encrypted simultaneously by a number of GPU threads using CUDA.

**Results**

Using CUDA as the encryption process is divided in blocks with each block containing numerous threads the computational time decreases drastically with increase in the size of the text file to be encrypted thus CUDA can be used to achieve excellent efficiency of problem solving with comparatively very low processing time.
As we can see initially as the size is low CPU has lower processing time and that of Tesla K20 card but as the size of text increases there is a drastic difference between the platforms.

Thus we can conclude that the processing time using CUDA is exponentially lesser than that of CPU code, i.e. Processing time in CPU(x) = (e^x)*Processing time in CUDA(x).

Acknowledgement
The success of this work would not have been possible without the Tesla K20 GPU card. The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K20 GPU used for this research.

References