

Parallelization of AES using MPI and CUDA

Submitted by

Akhil Vanukuri (160905314)

Varun Rajat Kumar (160905282)

Ratnakar Medepalli (160905314)

Semester VI, Section C

Department of Computer Science & Engineering



**MANIPAL
INSTITUTE OF TECHNOLOGY**

A Constituent Institute of Manipal University, Manipal

Abstract

Since the Data Encryption Standard (DES) fell out of favour among cryptographers due to its vulnerability to brute force attacks, the Advanced Encryption Standard (AES) has become the de facto standard for applications that require symmetric encryption. A major advantage of AES scheme is that it is parallelizable, and with the rapid advances being made in the development of GPUs, the performance of AES could benefit greatly from shifting the burden of heavy computations from the CPU to the GPU. Our goals for this project are to implement a parallelized version of the AES using Message Passing Interface (MPI) and CUDA as suggested in [1] and experimentally verify its performance. We may also be able to come up with a comparison between the different modes of operation as described in [2].

Introduction

The Advanced Encryption Standard (AES), is a specification of the Rijndael block cipher that was established as the standard for encryption by the National Institute of Standards and Technology (NIST) in 2001.

A block cipher is an encryption method that applies a deterministic algorithm along with a symmetric key to encrypt a block of text, rather than encrypting one bit at a time as in stream ciphers. For example, a common block cipher, AES, encrypts 128 bit blocks with a key of predetermined length: 128, 192, or 256 bits. Block ciphers are pseudorandom permutation (PRP) families that operate on the fixed size block of bits. PRPs are functions that cannot be differentiated from completely random permutations and thus, are considered reliable, until proven unreliable.

A block cipher by itself only handles the encryption or decryption of a group of bits called a block that have a fixed size. A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block.

Five modes of operation of the AES algorithm were standardized: ECB(Electronic Code Book), CBC(Cipher Block Chaining), CFB(Cipher FeedBack), OFB(Output FeedBack) and CTR(Counter). Each of the modes have their own unique set of characteristics that make them well suited to certain applications and ill suited to others.

Expected Results

We expect that our implementation of the AES in MPI and CUDA will perform better than a purely sequential implementation that runs only on the CPU and does not make use of the parallel nature of the algorithm.

Literature Survey

A lot of effort has been made into working towards the parallelization of AES. Le, Chang, Gou, Zhang and Lu (2010)[1] proposed a GPU based algorithm for fast encryption of the AES in CUDA. They achieved this through allocating one CUDA thread to handle the encryption of one block and hence, they initialize *numBlocks* CUDA threads in the beginning, where *numBlocks* is the length of the plaintext divided by number of characters processed per block.

Like all block ciphers, AES can be executed in any of the five modes that were standardized, namely ECB, CBC, CFB, OFB and CTR. Bielecki and Burak (2005)[2] showed that for all the modes apart from ECB and CTR, either encryption or decryption operations cannot be performed as subsequent blocks often require the use of the results of their preceding blocks. For the ECB mode and CTR mode however, both encryption and decryption can be parallelized as all blocks perform their operations independently. Hence, if there is any question of parallelizing AES, it must be operated under either of these two modes.

As reported by NIST [6], for a given key, the ECB mode encrypts any given plaintext block as the same ciphertext block. This property is not desirable as for longer messages, the cipher becomes susceptible to replay attacks. Hence, for our purposes we will only consider the CTR mode of operation.

Methodology

The algorithm divides the input into blocks of size of 128 bits. Each block undergoes 11 rounds of encryption to get the encrypted output.

In serial code, this is being done by using a for loop to iterate over different blocks, followed by the for loop to iterate over each round of encryption.

In the MPI Implementation of the serial code, a process is used to encrypt each block. Each process uses a for loop to iterate over each round of encryption.

In the CUDA Implementation of the serial code, we use a slightly different method to assign a input block of 128 bits to a thread. The number of threads is determined by dividing the number of blocks by the greatest factor of the number of block. Each thread is responsible for the encryption of one block.

Limitations and Possible Improvements

CUDA:

CUDA provides us a limitation in the number of blocks which can be processed. Each block undergoes 10 rounds during the encryption process and the number of blocks which can be encrypted is limited to $512 * 1024$.

MPI:

In MPI, each process is provided with one block to process. Each process does the encryption process on the block assigned. The limitation encountered here, is that, the number of blocks that can be processed is limited by the number which in turn, is limited by the number of processes which can be generated by the MPI runtime environment.

Possible Improvements:

The CUDA limitation requires us to use multidimensional CUDA grids and blocks, and if we want to achieve an increase in performance.

The MPI limitation can possibly be countered in the future if we allot ' n ' blocks to each process where $n = (\text{number of blocks})/(\text{number of processes})$.

We can also incorporate AES decryption in the future, as when using it in CTR mode it is also parallelizable.

Results

Due to limitations of our processors, we didn't manage to achieve as significant speedup in MPI as we did in CUDA. We did however find much scope of improvement, and believe that the project may scale better with more powerful computing devices. On CUDA, however we observed a very large increase in performance and even this can be

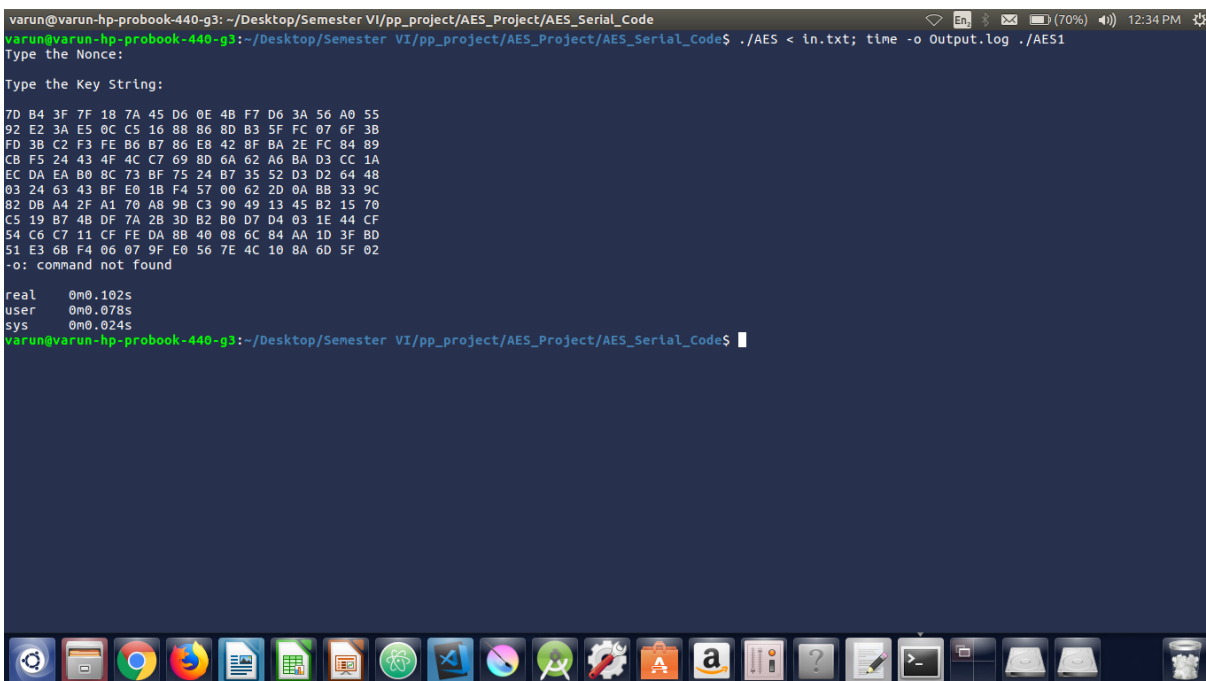
Taking three trials with the same input plaintext (of 10 blocks), nonce and key, we observed that for 10 blocks

CUDA on an average took 0.0088 seconds

MPI on an average took 0.0876 seconds

A purely serial implementation took 0.102 seconds

We expect to observe a significant increase in performance if we modify our MPI code to be more scalable. As it stands, the performance of our MPI implementation is not on par with CUDA.



```
varun@varun-hp-probook-440-g3: ~/Desktop/Semester VI/pp_project/AES_Project/AES_Serial_Code
varun@varun-hp-probook-440-g3:~/Desktop/Semester VI/pp_project/AES_Project/AES_Serial_Code$ ./AES < in.txt; time -o Output.log ./AES1
Type the Nonce:
Type the Key String:
7D B4 3F 7F 18 7A 45 D6 0E 4B F7 D6 3A 56 A0 55
92 E2 3A E5 0C C5 16 88 86 8D B3 5F FC 07 6F 3B
FD 3B C2 F3 FE B6 B7 86 E8 42 8F BA 2E FC 84 89
CB F5 24 43 4F 4C C7 69 8D 6A 02 A6 BA D3 CC 1A
EC DA EA B0 8C 73 BF 75 24 B7 35 52 D3 D2 64 48
03 24 63 43 BF E0 1B F4 57 00 62 2D 0A BB 33 9C
82 DB A4 2F A1 70 A8 9B C3 90 49 13 45 B2 15 70
C5 19 B7 4B DF 7A 2B 3D B2 B0 D7 D4 03 1E 44 CF
54 C6 C7 11 CF FE DA 8B 40 08 6C 84 AA 1D 3F BD
51 E3 6B F4 06 07 9F E0 56 7E 4C 10 8A 6D 5F 02
-o: command not found

real    0m0.102s
user    0m0.078s
sys     0m0.024s
varun@varun-hp-probook-440-g3:~/Desktop/Semester VI/pp_project/AES_Project/AES_Serial_Code$
```

Serial Implementation

Conclusion

As we observed there is a significant scope of improvement in performance of AES encryption when we implement parallelly. This improvement can similarly be extended to AES decryption.

References

- [1] D.Le, J.Chang, X.Gou, A.Zhang, and C.Lu, *Parallel AES algorithm for fast data encryption on GPU*, in *Computer Engineering and Technology (ICCET)*, 2010 2nd International Conference on, vol. 6, April 2010.
- [2] Bielecki, W., Burak, D., *Parallelization of Standard Modes of Operation for Symmetric Key Block Ciphers*, in *Image Analysis, Computer Graphics, Security Systems and Artificial Intelligence Applications Vol.I (ACS-CISIM 2005)*, Bialystok 2005.
- [3] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
- [4] <https://www.wolfssl.com/what-is-a-block-cipher/>
- [5] *Cryptography Engineering: Design Principles and Practical Applications*. Ferguson, N., Schneier, B. and Kohno, T. Indianapolis: Wiley Publishing, Inc. 2010. pp. 63, 64.
- [6] (NIST),Morris Dworkin. "*SP 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques*", csrc.nist.gov.