**Parallel-AES-Algorithm-using-CUDA**

Meril Koli

(811011005)

**Summary (Abstract)**

AES is rather sequential in nature due to the fact that each successive round depends on the output of the prior round. Being able to do this will afford us huge gains in efficiency and speedup. AES is a block cipher algorithm used to encrypt data using a 128-bit key. Data is divided up into 128-bit blocks and encrypted. Each block goes through 11 rounds of encryption, with 4 steps: SubBytes, ShiftRows, MixColumns, AddRoundKey. This project parallelized the process of AES encryption with CUDA. We can enhance the overall speedup to 3 times than the original serial version. AES program first encrypt the input file with the user-defined key file. Then, to check the correctness of our encryption, we decrypt the encrypted file using the same key file.

### Introduction

### Encryption is the process of converting original message into cipher information by using key. In this process the information must be in the form of hexadecimal or integer format only. Decryption is the process of converting cipher information to original information by using same key. In decryption the format of message must be in Hexadecimal or integer. In this paper the advanced encryption standard algorithm technique uses the symmetric key that means private key. The main advantage of symmetric key is provide security to the data and reduces the area also. This types of private keys mainly used in ATM machines and software’s as well.

### Steps for parallelization AES

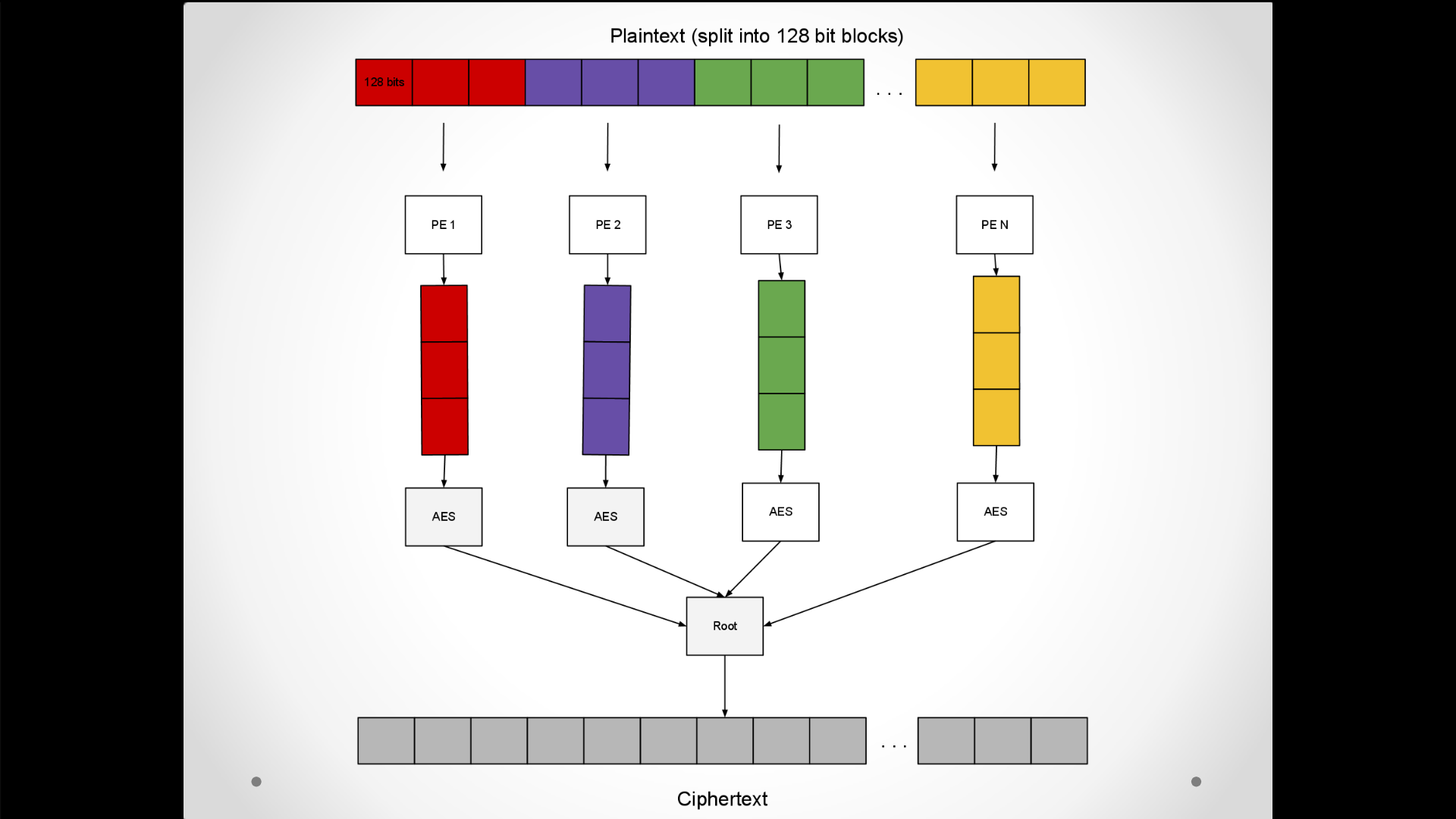
### Assigning each PE a copy of the entire data.

### Each PE is assigned a portion of the data, split into 128-bit blocks

### Each block is then encrypted by the PE’s to produce ciphertext blocks

### Each PE encrypts its blocks in parallel, but the blocks themselves are encrypted sequentially per PE.

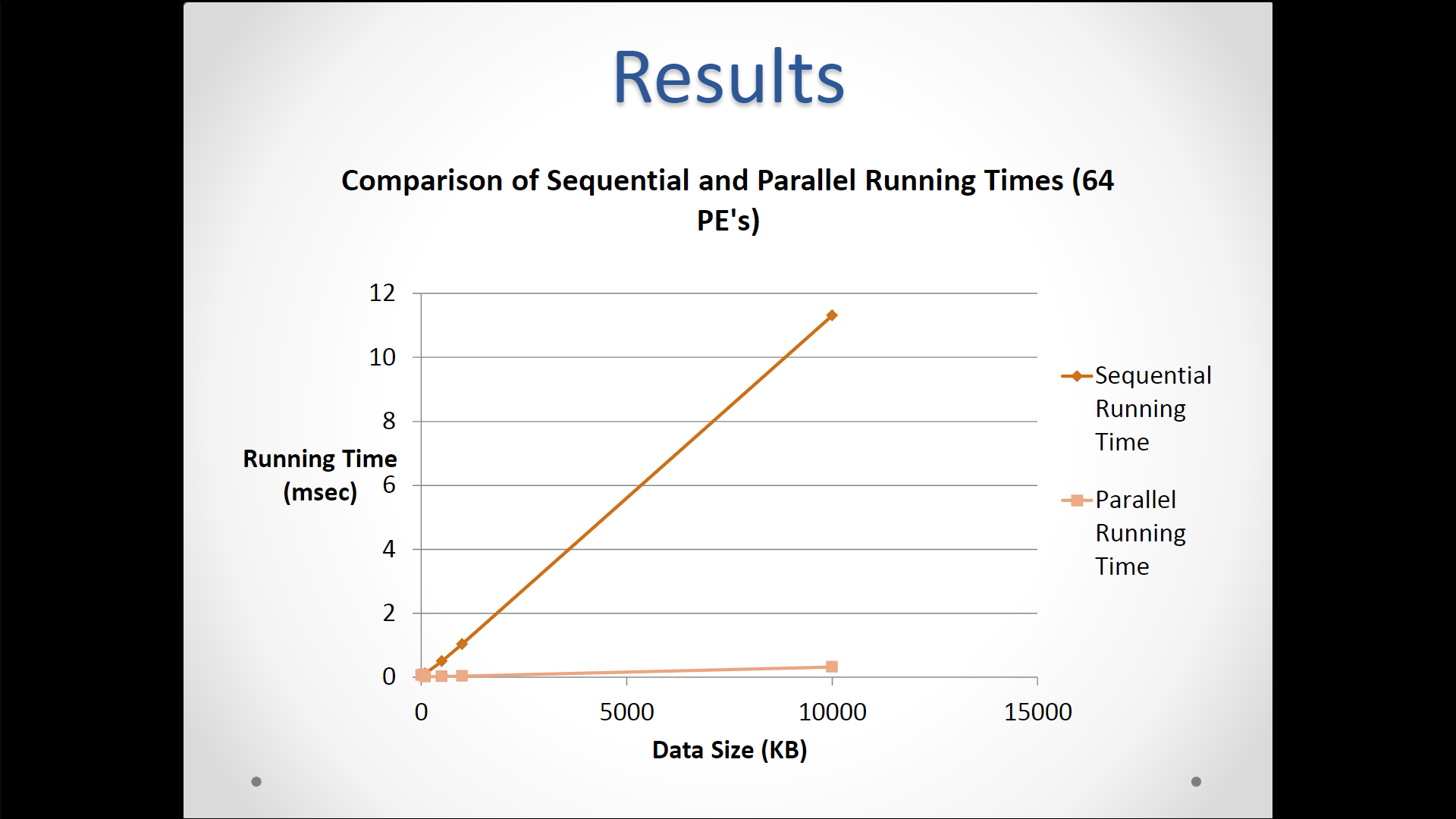
### Data is retrieved by root to ciphertext is written to output.



### Experimental Setup:

* Ran tests for file sizes of 2kb, 10kb, 50kb, 100kb, 500kb, 1MB, 10MB, 50MB, 100MB
* Utilized 2, 4, 8, 12, 16, 24, 36, 48, and 64 PE’s
* Used AES library to perform the encryption/decryption itself, and MPI for parallelization
* Each running time was the average of 3 runs.
* Times taken were from right before encryption (after data had been distributed) to right after root had gathered data.

### Results:



### Conclusion:

### Able to clearly see benefits by parallelization

### Extremely low running times for a high number of PE’s, but with added cost

### Encryption/decryption takes the same amount of time, as expected

### Considerable overhead for small files and high PE’s.

### Reference:

* <http://en.wikipedia.org/wiki/Advanced_Encryption_Standard>
* <http://ieeexplore.ieee.org.gate.lib.buffalo.edu/stamp/stamp.jsp?tp=&arnumber=5486259&isnumber=5485932>
* <https://nanopdf.com/download/project-report-parallel-aes-implementation-chris-norman_pdf>