Project Report: Parallel AES Implementation

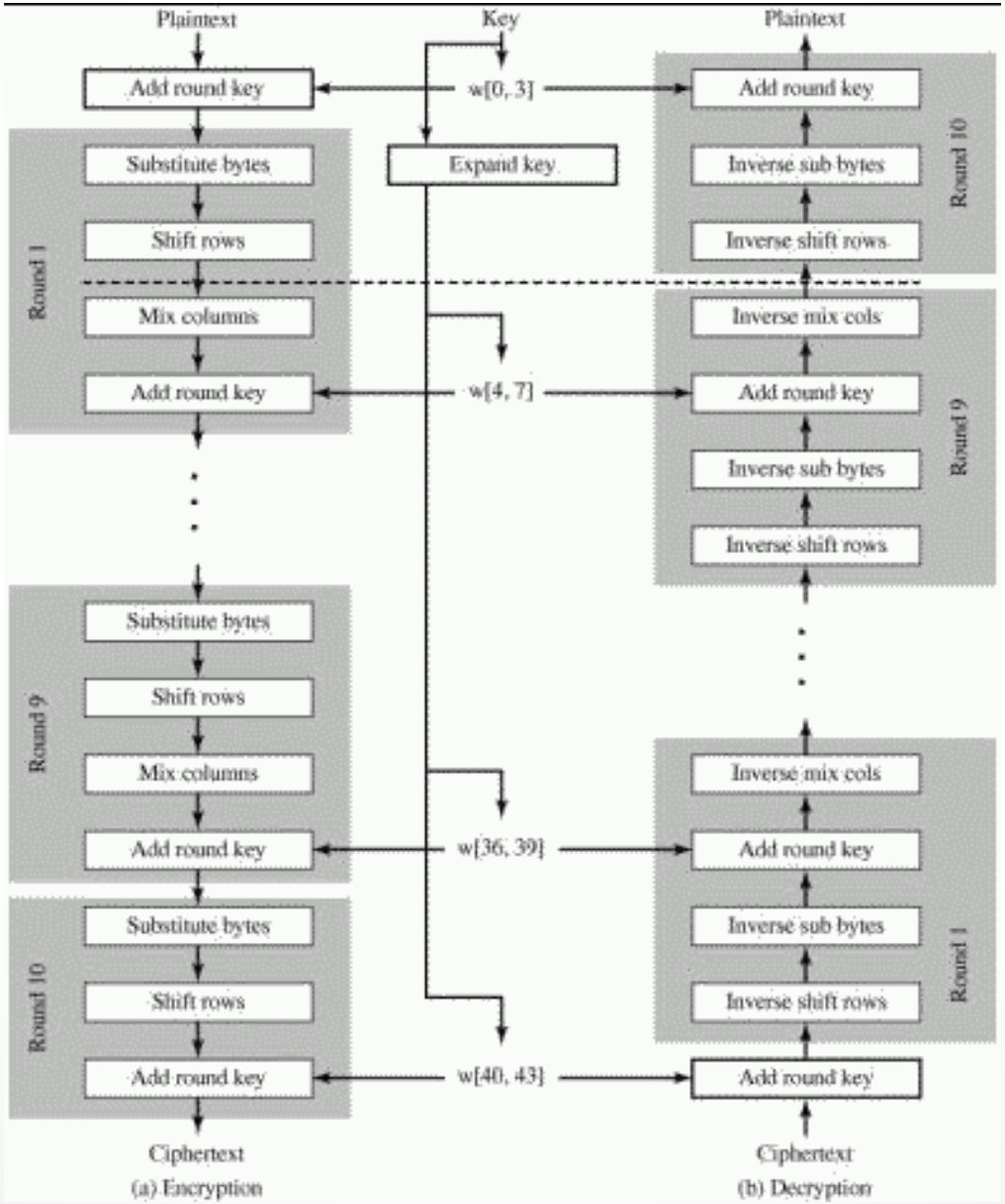
Chris Norman

CSE633 Fall 2011

Algorithm

- 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
- The ciphertext is produced and is recovered by performing decryption with the same 128-bit key
- In a sequential scheme, each block would be encrypted sequentially

Overview

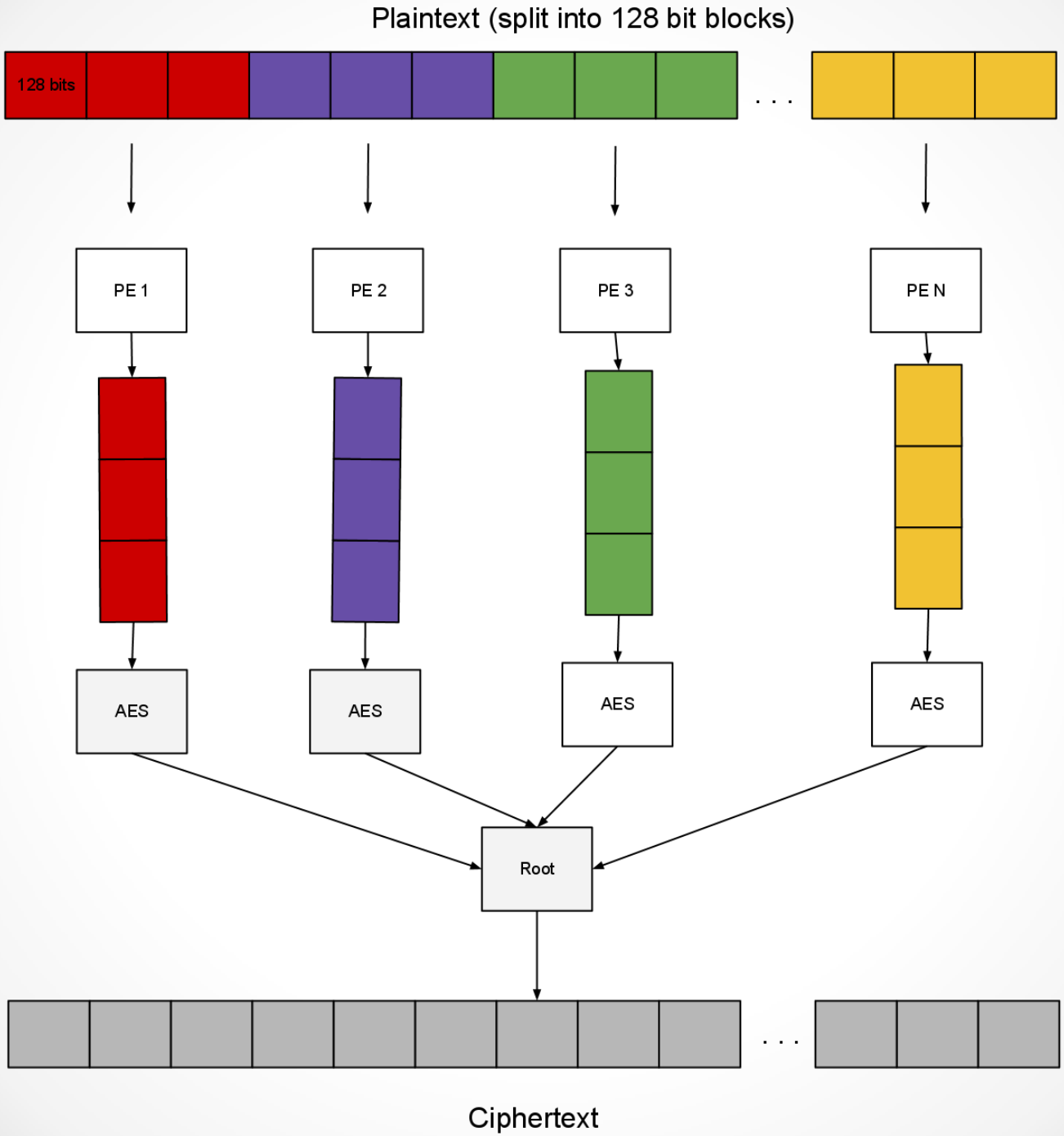

Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Round 10
Shift rows
Inverse shift rows
Round 1
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Round 9
Inverse shift rows
Substitute bytes
Shift rows
Mix columns
Inverse mix cols
Add round key
w[36, 39]
Add round key
Substitute bytes
Inverse sub bytes
Round 1
Round 10
Shift rows
Inverse shift rows
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption

Parallel Implementation

- As mentioned before, AES is rather sequential in nature due to the fact that each successive round depends on the output of the prior round
- So we're not interested so much in speeding up AES encryption itself, but rather encrypting the blocks in parallel
- Being able to do this will afford us huge gains in efficiency and speedup

Parallel Implementation

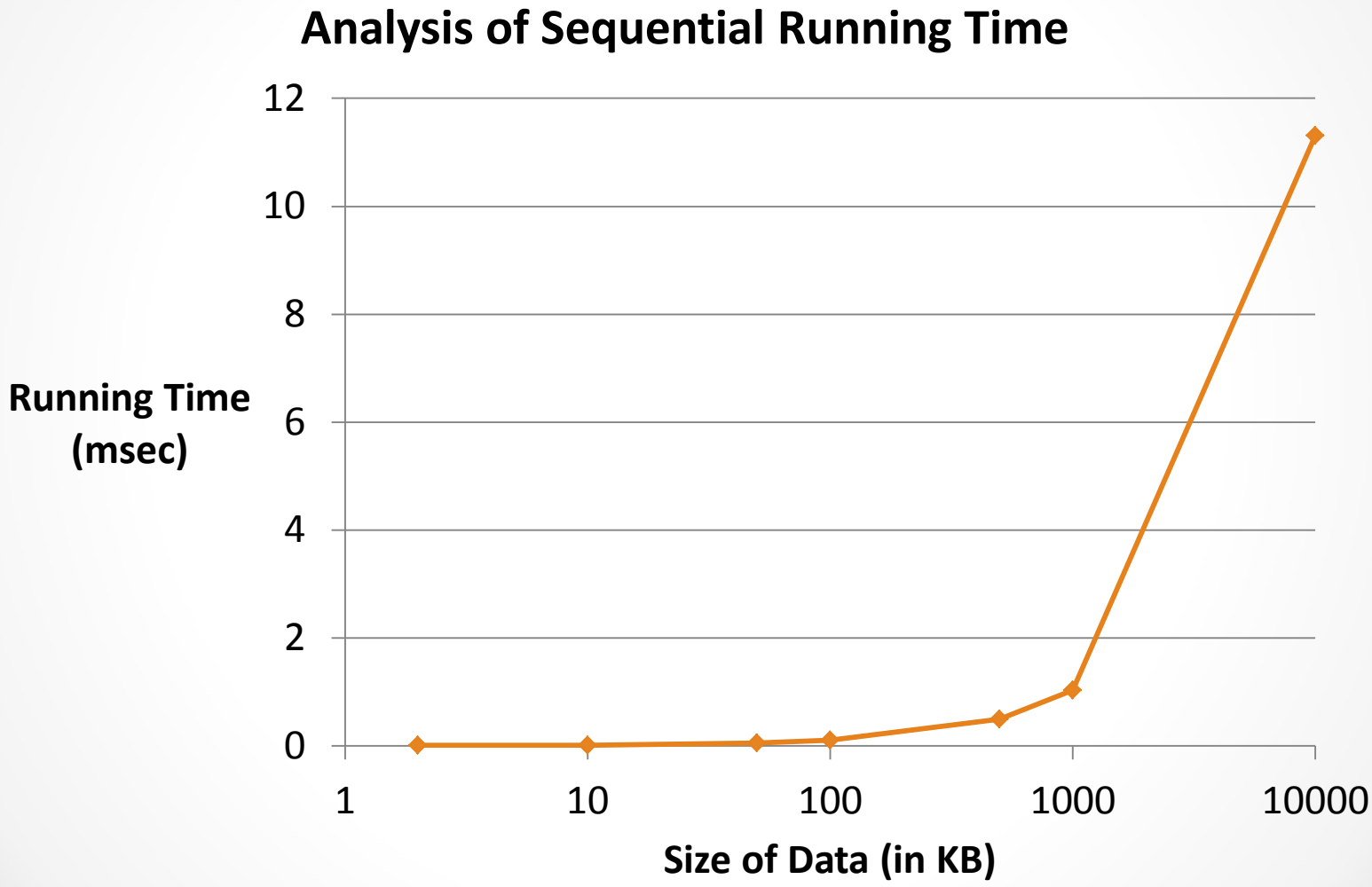
- Utilized PolarSSL's AES library to perform AES encryption
- Used MPI for parallelization
- Performed parallelization by:
  - Assigning each PE a copy of the entire data
  - Each PE is assigned a portion of the data, split into 128-bit blocks
  - Each block is then encrypted by the PE's to produce ciphertext blocks
    - Each PE encrypts its blocks in parallel, but the blocks themselves are encrypted sequentially per PE.
  - Data is retrieved by root by MPI_Gather and ciphertext is written to output


Plaintext (split into 128 bit blocks)
128 bits
PE 1
PE 2
PE 3
PE N
AES
AES
AES
AES
Root
Ciphertext

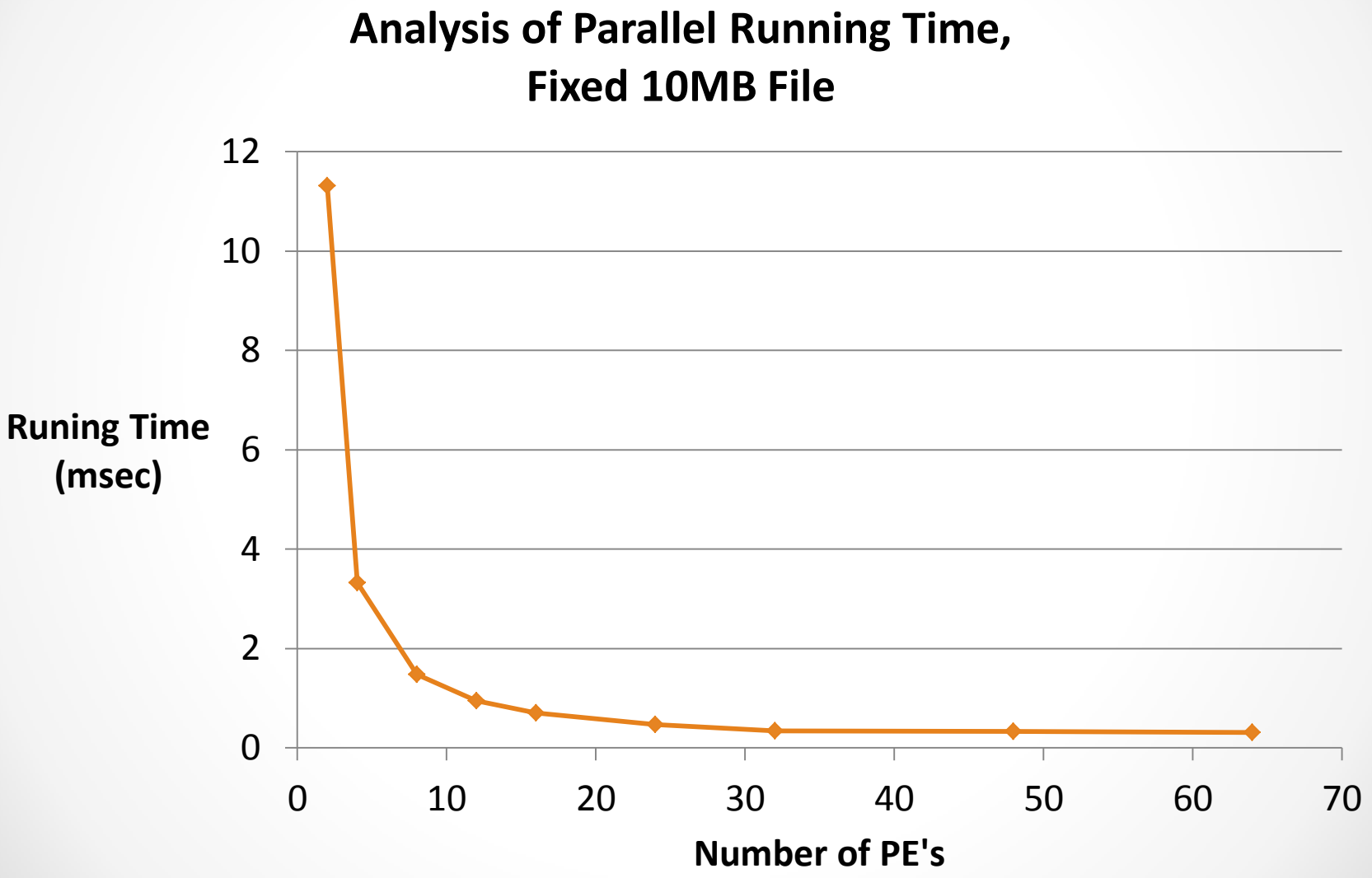
Experimental Setup

- Used the 8-core nodes with Infiniband for experimentation
- 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 PEs
- Used PolarSSL's 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

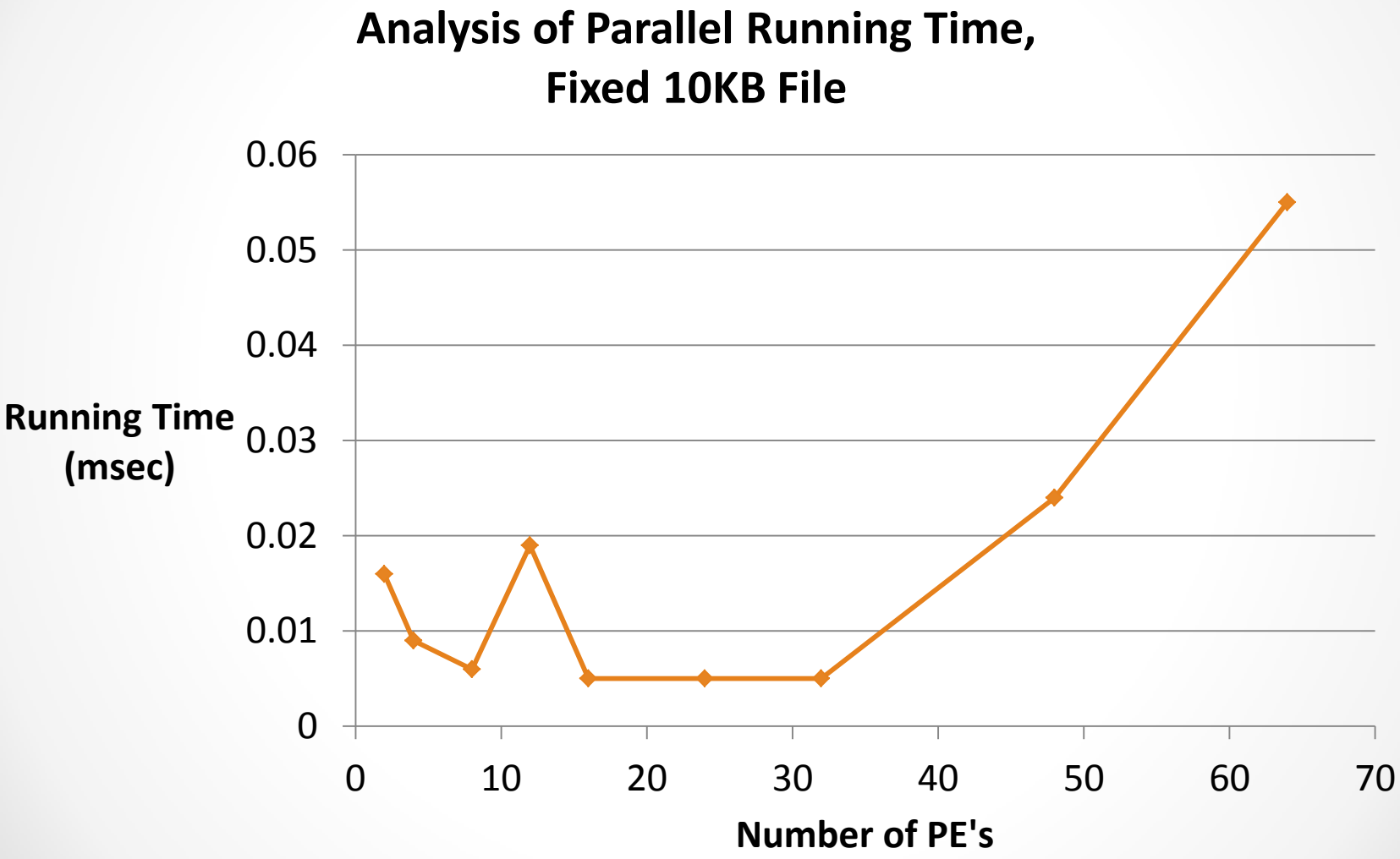
Results


Analysis of Sequential Running Time
Running Time (msec)
12
10
8
6
4
2
0
1
10
100
1000
10000
Size of Data (in KB)

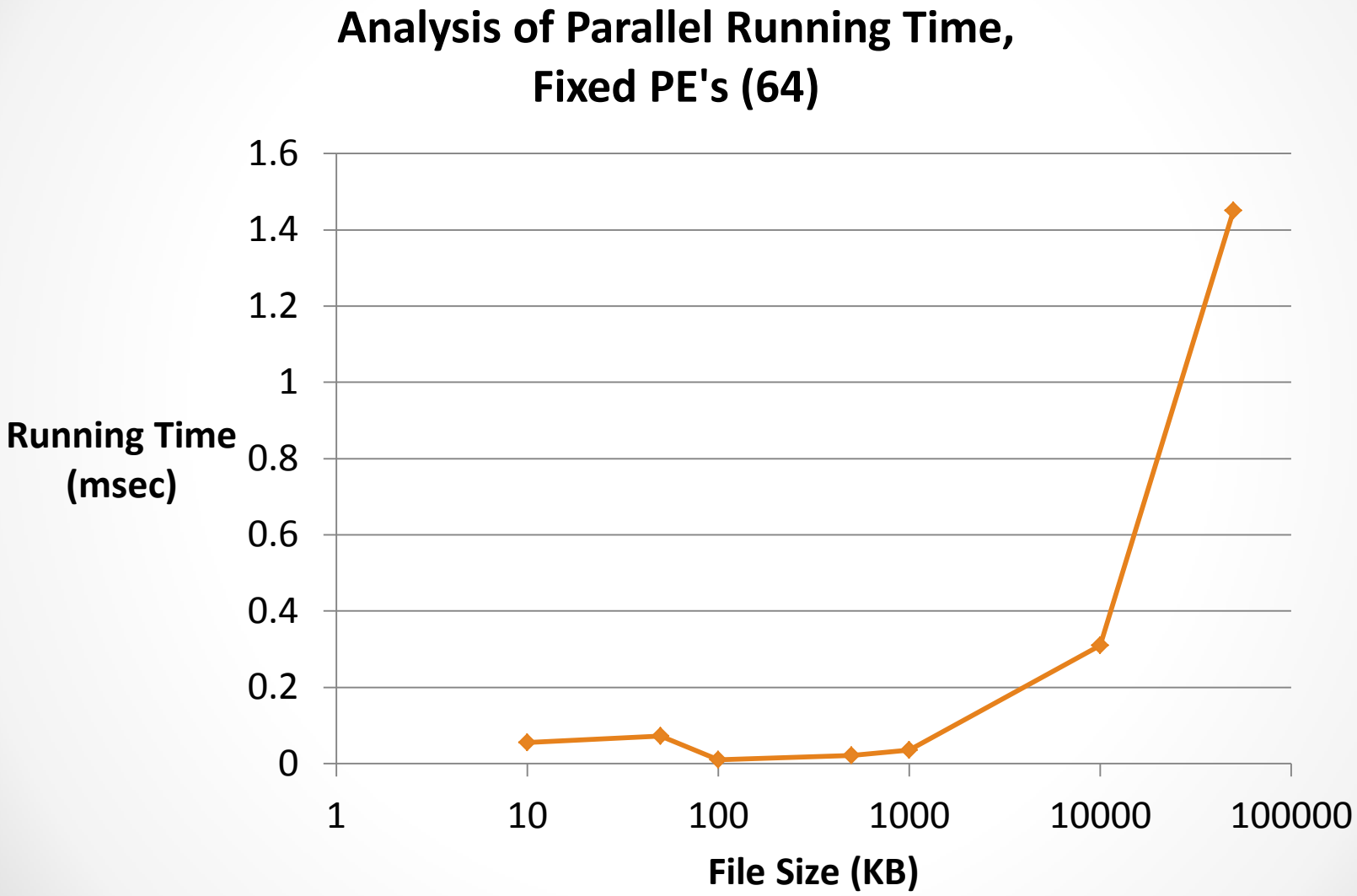
Results


Analysis of Parallel Running Time,
Fixed 10MB File
Runing Time
(msec)
12
10
8
6
4
2
0
0
10
20
30
40
50
60
70
Number of PE's

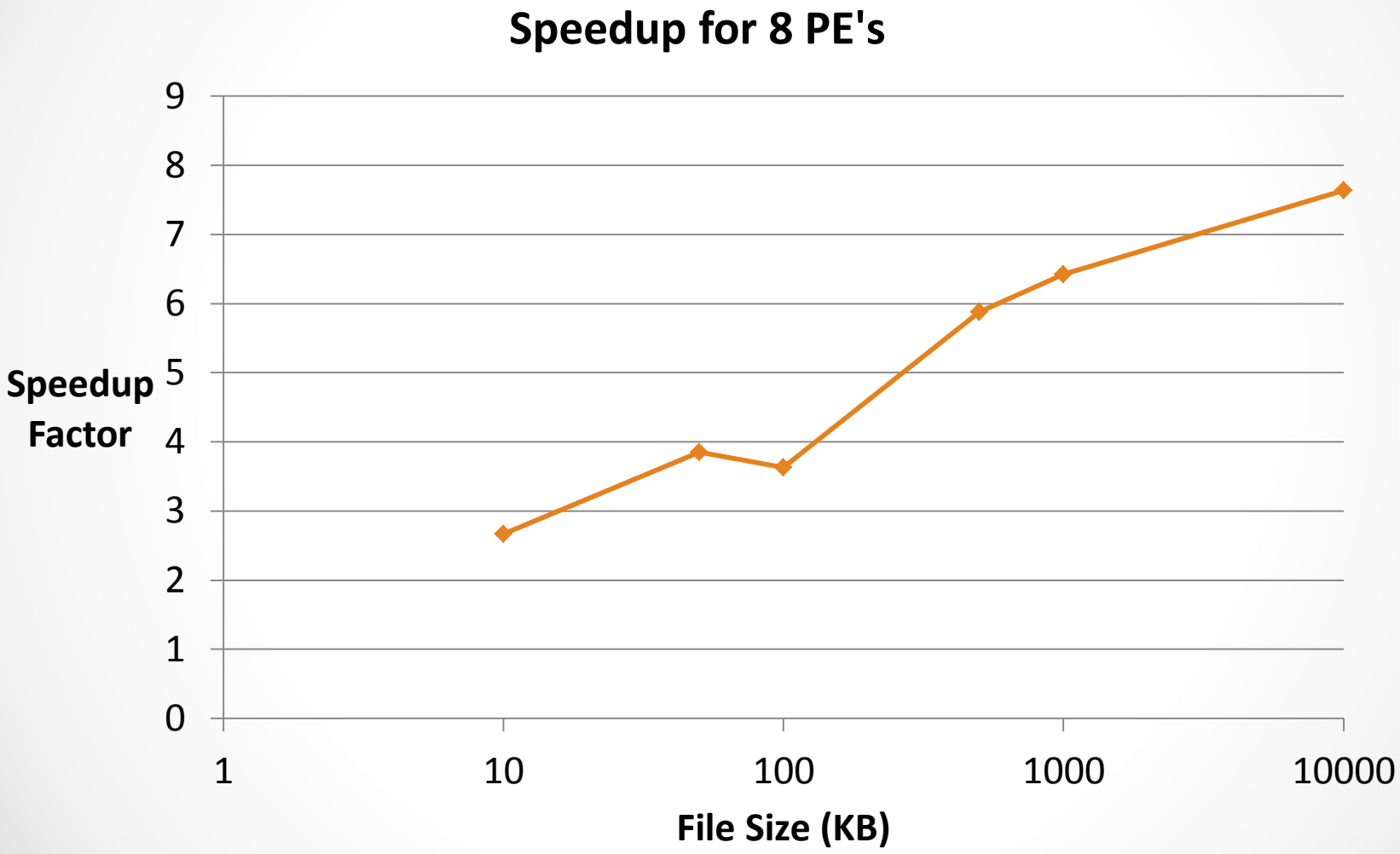
Results


Analysis of Parallel Running Time,
Fixed 10KB File
Running Time (msec)
0.06
0.05
0.04
0.03
0.02
0.01
0
0
10
20
30
40
50
60
70
Number of PE's

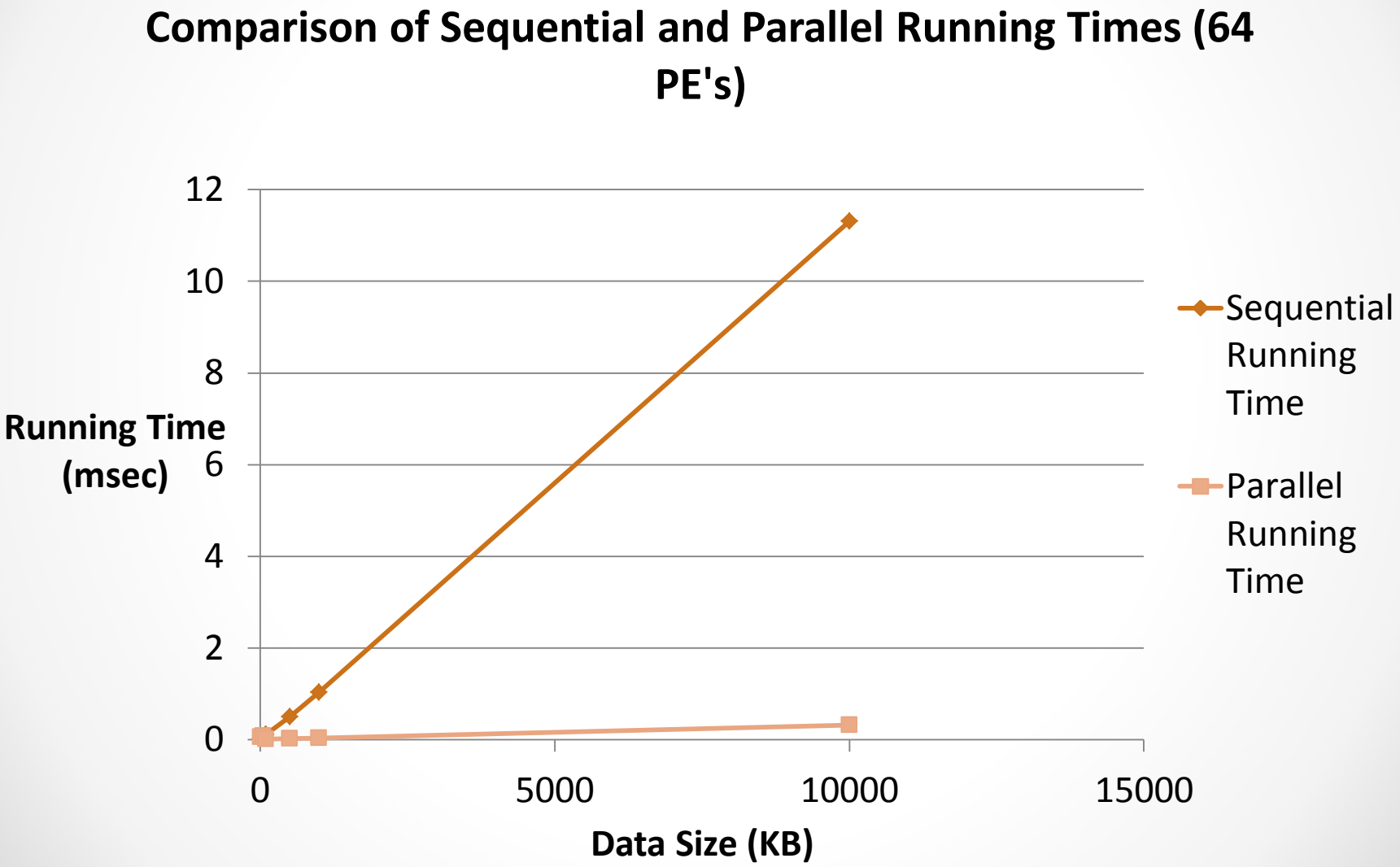
Results


Analysis of Parallel Running Time,
Fixed PE's (64)
Running Time (msec)
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
10
100
1000
10000
100000
File Size (KB)

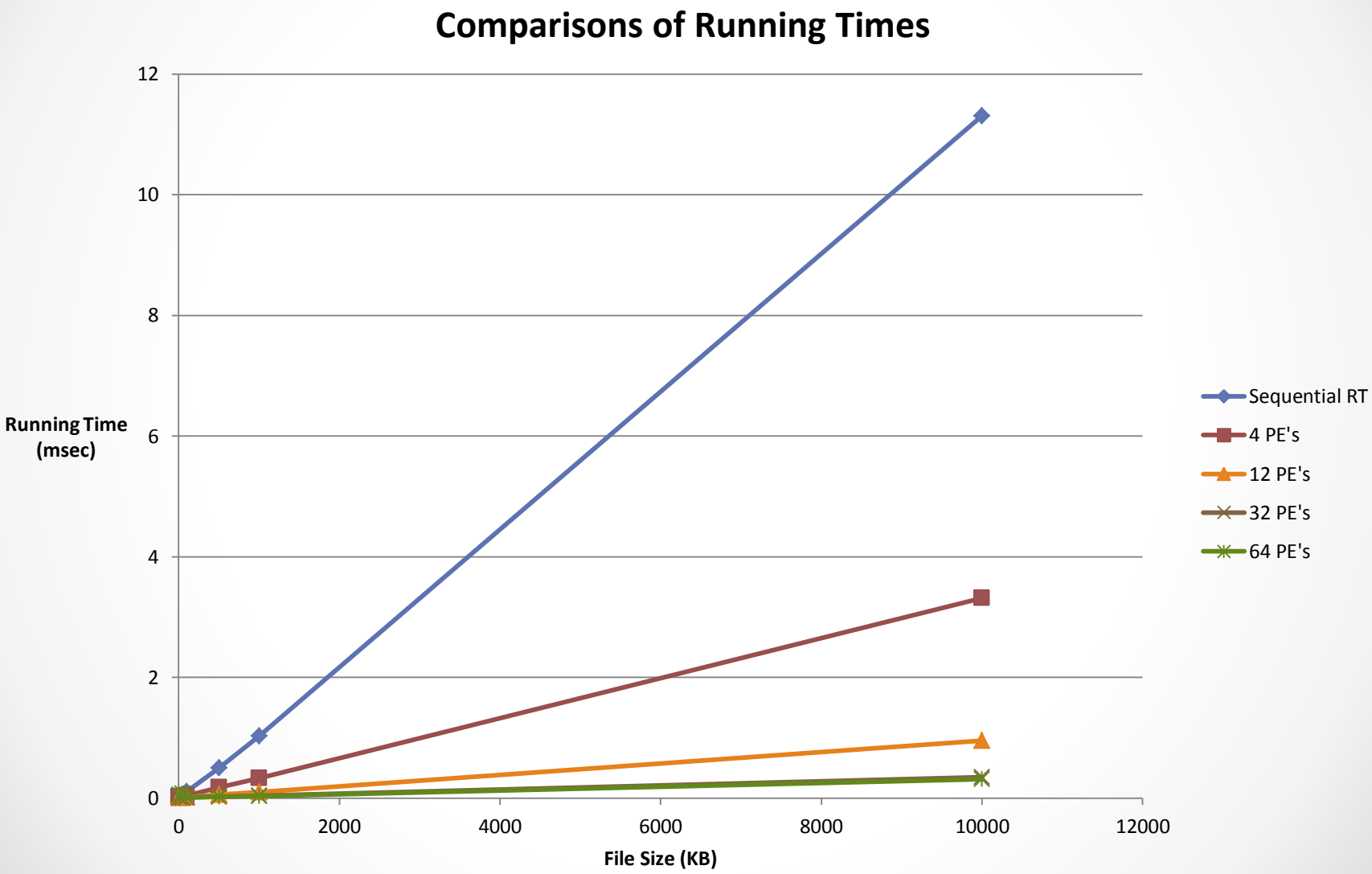
Results


Speedup for 8 PE's
Speedup Factor
9
8
7
6
5
4
3
2
1
0
1
10
100
1000
10000
File Size (KB)

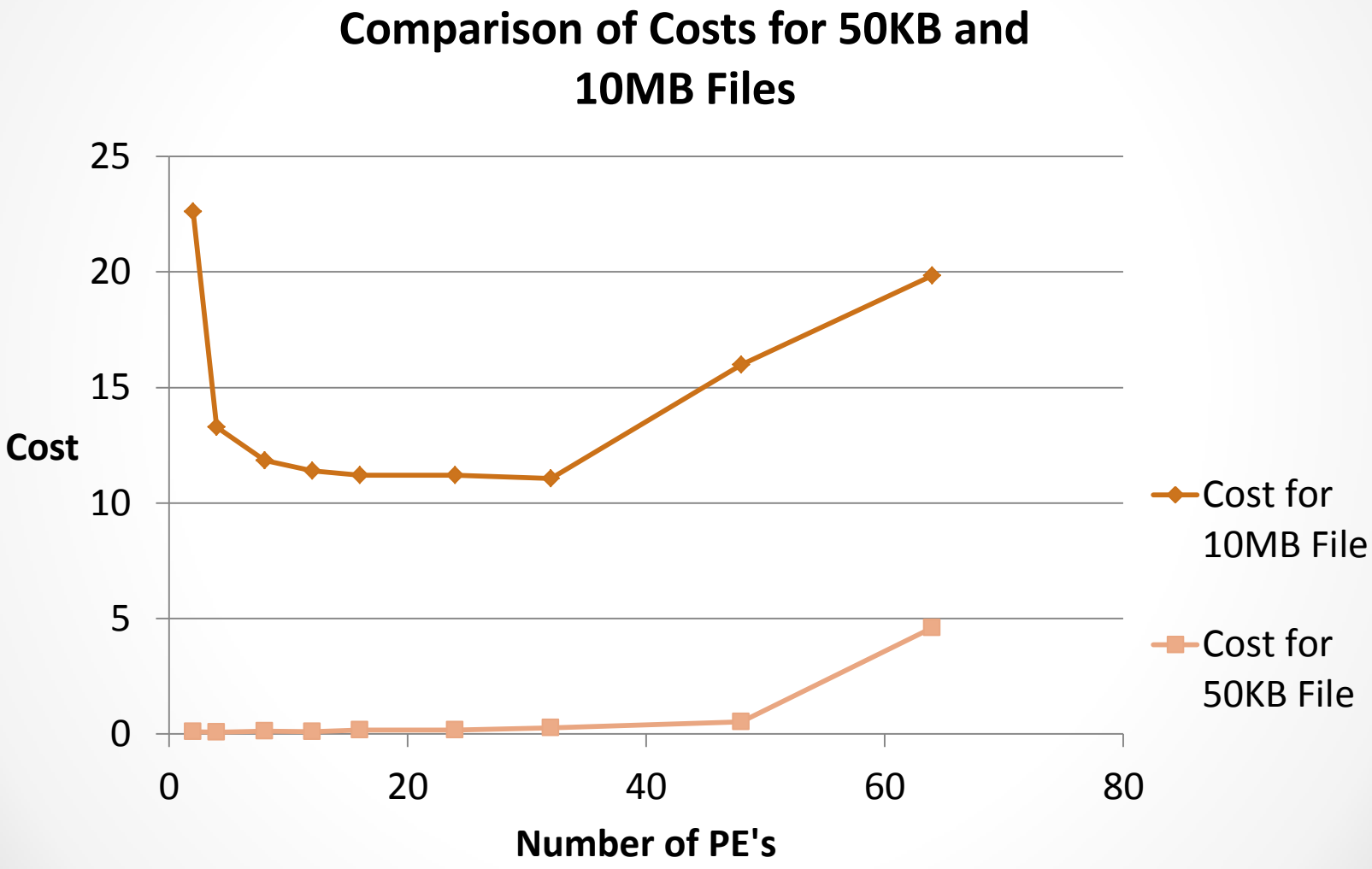
Results


Comparison of Sequential and Parallel Running Times (64 PE's)
Running Time (msec)
12
10
8
6
4
2
0
0
5000
10000
15000
Data Size (KB)
Sequential Running Time
Parallel Running Time

Results


Comparisons of Running Times
Running Time (msec)
12
10
8
6
4
2
0
0
2000
4000
6000
8000
10000
12000
File Size (KB)
Sequential RT
4 PE's
12 PE's
32 PE's
64 PE's

Results


Comparison of Costs for 50KB and 10MB Files
Cost
25
20
15
10
5
0
0
20
40
60
80
Number of PE's
Cost for 10MB File
Cost for 50KB File

Conclusions

- 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

Future Work

- Fix program so that the ciphertext written by the PE's is recoverable to plaintext
- Make program more space-efficient by not making n copies of the data for each PE to use
  - In addition, capture the 'true' running time of the algorithm by timing entire program

References

- [1] http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
- [2] Deguang Le; Jinyi Chang; Xingdou Gou; Ankang Zhang; Conglan Lu; , "Parallel AES algorithm for fast Data Encryption on GPU," *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on* , vol.6, no., pp.V6-1-V6-6, 16-18 April 2010
  doi: 10.1109/ICCET.2010.5486259
  URL: http://ieeexplore.ieee.org.gate.lib.buffalo.edu/stamp/stamp.jsp?tp=&arnumber=5486259&isnumber=5485932
- [3] http://www.codeproject.com/KB/security/SecuringData.aspx
- [4]http://www.polarssl.org/