International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

Improving the MapReduce Performance using Symmetric Key Algorithm

G. Siva Brindha[1], Dr. M. Gobi[2]

[1]Research Scholar, Department of Computer Science, Chikkanna Government Arts College, Tiruppur, Tamilnadu, India
e-mail: *gsbshivaa[at]gmail.com*

[2]Associate Professor, Department of Computer Science, Chikkanna Government Arts College, Tiruppur, Tamilnadu, India
e-mail: *mgobimail[at]yahoo.com*

Abstract: ***In modern networking age, offloading of data to the cloud makes marked significant reliance on compatible and easily available cloud infrastructure. The bias of safety regarding their personal data has also become unmindful that they completely merge with the cloud service providers. The basicity of data protection for reliable and secure cloud environments known as Encryption, that demands high costs in accordance to its data size is also at risks of facing some obstacles for the safety of big data security. Hence, this paper brings forth a structural framework to reduce the costs of Encryption using MapReduce that can work wonders in enhancement parallel processing and parameter tuning. In addition it enhances less time consumption of encryption that automatically results in minimal usage of system resources. The current research tries to meet the benefits appreciated through MapReduce-based parallel encryption computation.***

Keywords: Cryptography, Encryption Cloud infrastructure, MapReduce, Large data security

1. Introduction

In the fast moving technological era, people seek to have more convincing reasons to store their applications and data in both private and public clouds. For example the rapid elasticity, cost saving, quickness etc., require such cloud to preserve everything. In this stratum companies always cope up to the applications related to public cloud to establish their benefits. Due to its inevitable necessity and compatibility colossal data are created and preserved on the cloud. The originations of such information may vary ranging from mobile data to any kind of online transactions, Email, Social media sites etc., It is because of this compelling need of the hour cloud data centers and ingenious minds are working hard to provided effective and more undeniably reliable clouds storage to benefit wide range of business platforms to meet its expectations balancing the huge data available.

It is also a matter of fact that as a coin has two faces along with the benefits there are privacy and safety threat both in confidential personal and business data because of the security techniques that lack in its performance in the cloud. As a result the encryption technology also provides less application functionality in using data in the computational-resources. Adding to this storing data has become more pressuring one because of its velocity, volume and varieties of big data.

For scalable and efficient encryption schemes for big data on the cloud there has arisen an increased demand for unfailing security that seek to adopt and implement several encryption algorithms [5], [10]. The CPU suffers huge consumption of resources that eventually makes the encryption structures weigh larger and make them suffer. There is a solution called Parallel computation through which several computations can be brought up all together on multiple microprocessors. Moreover, there also lay this Multicore and multiprocessor computers designed to have multiple processing elements within a single machine that can be used for such parallel processing.

In this paper, using MapReduce we try to come up with a framework to serve as a programming model paralleling and distributing the encryption of large amounts of data. Subjecting this idea, the paper strive to improve the performance of encryption by using different conditions. In order to achieve good performance, we take up intensive steps to choose a set of configuration parameters to set up the framework. These parameters through which we work will gradually tend to affect the encryption performance in the MapReduce framework. So using AES encryption we usually carry forward the framework and compare the results to those results gained from standard sequential AES implementation without any sort of intervention of MapReduce. The results of the above mentioned problem achieve significant performance gains through MapReduce along with the selected configuration parameters.

2. MapReduce

MapReduce is the core component for data processing in Hadoop work as a sub-project of the Apache Hadoop project, which considers as a software framework that comfortable to deal with huge, long-running jobs that cannot be grip within the reach of a single request. It is useful for distributed processing with an efficient process and general data sets on the computing cluster [10]. it used to decrease the cost of security by automatically split the input data into a number of parts then run a program parallel on that data parts with handle most of the problems at once, such consistency and fault tolerance. It is a useful framework for big data to deal with high efficiency and guaranteed handling of a large cluster environment [11]. It has two separate tasks each of a distinct work where mapper doing as a converting task for a set of data from one form to

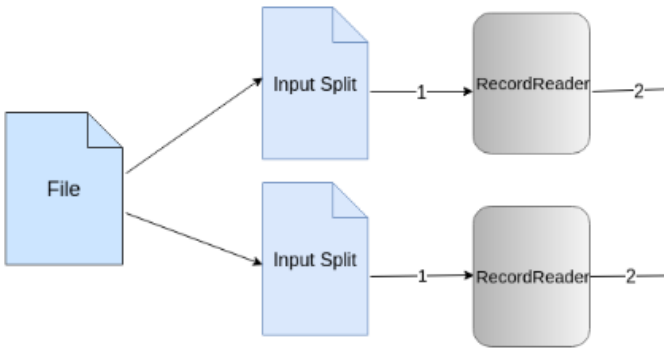
another by broken the individual elements down into tuples (key/value pairs). Where the reducer will do the opposite, which takes in the combines of those data tuples based on the key and accordingly modifies the value of the key. Fig.4 illustrates the general idea of MapReduce.

2.1 MapReduce Workflow

A MapReduce paradigm is given in Figure 4.1 MapReduce is designed to continue to work in the face of system failures. When a job is running, MapReduce monitors progress of each of the servers participating in the job. If one of them is slow in returning an answer or fails before completing its work, MapReduce automatically starts another instance of that task on another server that has a copy of the data. The complexity of the error handling mechanism is completely hidden from the programmer.

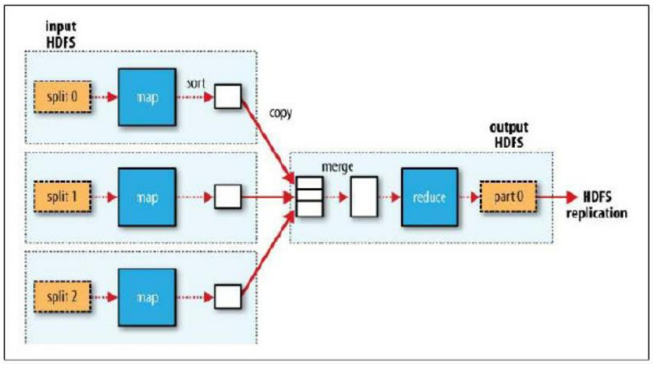
MapReduce is triggered by the map and reduce operations in functional languages,such as Lisp. This model abstracts computation problems through two functions: map and reduce. All problems formulated in this way can be parallelized automatically. Essentially, the MapReduce model allows users to write map/reduce components with functional-style code. These components are thencomposed as a dataflow graph to explicitly specify their parallelism. Finally, the MapReduce runtime system schedules these components to distributed resources for execution while handling many tough problems: parallelization, network communication, and fault tolerance.

A map function takes a key/value pair as input and produces a list of key/value pairs as output. The type of output key and value can be different from input:
map :: (key1; value1) □ list(key2; value2)… (1)
A reduce function takes a key and associated value list as input and generates a list of new values as output:
reduce :: (key2; list(value2)) -> list(value3)… (2)
A MapReduce application is executed in a parallel manner through two phases. In the first phase, all map operations can be executed independently from each other. In the second phase, each reduce operation may depend on the outputs generated by any number of map operations. All reduce operations can also be executed independently similar to map operations.



2.2 Uses of MapReduce

At Google:
– Index building for Google Search
– Article clustering for Google News
– Statistical machine translation
At Yahoo!:
– Index building for Yahoo! Search
– Spam detection for Yahoo! Mail
At Facebook:
– Ad optimization
– Spam detection



3. Advanced Encryption Standard

In 2001 NIST designed The Advanced Encryption Standard (AES) [3] and for a wide range of application it has been accepted as the approved standard. It has been said in AES that the same key is used for encryption and description of data in symmetric key algorithm. In addition to that the keys size in AES are given in 128, 192 or 256 bits. A block cipher that works on one block of data at a time is known as algorithm. There are various sizes of blocks that can be used.

AES is one of the most reliable and widely used in practice to safeguard the applications that store data in the cloud as shown in table 1. It is celebrated due to its effectiveness and proven security in its scheme. On the other hand when applied to large memory in cloud, the AES awaits memory requirements and time consumption that evidently result in few performance limitations. In addition to that it has also limited many application functionalities like the search function, logic operations, and mathematical calculation. Moreover, it has a biggest threat of providing a secure and scalable key management system of a symmetric AES encryption scheme in the cloud environment.

Table 1: Popular Cryptosystems Inindustry

Industry Cryptosystem	
Industry Product	**Encryption**
Cloudera, Navigate or Encrypt	AES 256
SafeNet, ProtectDB	AES, 3DES, DES, RSA, RC4, SHA-1, ACSHA-1
Thales, Hardware Security Modules (HSM)	AES(128, 192,256) 3DES, RSA ECC
Cloud Link RSA Data Protect Manager	AES – 256
HP, Altlla	AES

4. AES Encryption in MapReduce

A. AES in Paralle Computation

Advanced Encryption Standard (AES) cryptography algorithm in Parallel implementation has been taken forward

in varied researches to improve the performance levels of encryption. Parallel computation can be achieved through any single machines with multiple processing elements.

Over standard CPU's the Graphics Processing Units (GPUs) claim a large latent performance within stream processing applications. It is because of the distributed structural platform that resides in the form of large numbers of simple processing unit in GPU its performance also becomes highly commendable. The investigation carried out by Harrison and Waldron [11] shows that the GPU with large packet sizes gives out best results because of its adaptation to the applications that bulk data encryption/decryption. This paper also validates that the GPU can be used effectively when compared to other operating systems.

In an investigation carried out by Nagendra and Sekhar [6] the application of an AES cryptography algorithm on a dual core processor using OpenMP API to reduce execution time has been explored. It is rather different from the applications that we have seen so far that OpenMP (Open multiprocessing) works on multicore architecture to provide shared multithreaded memory parallelism. It is said that than in sequential implementation of other applications the parallel implementation of an AES block cipher using a dual- core (Intel Core 2 Duo) processor takes 40% ~ 45% less time to perform the encryption and decryption.

B. Parallelizing Encryption in MapReduce

Multicore and multiprocessor computers dwell in limitations in execution of big data in the cloud environment during Parallel computation. Segregation of equal work to process is always a pressurizing task in cloud environment. After that to associate the results from independent processes may further drag the actual processing. Finally, there may also arise a lack of processing capacity in a single machine. The coordination and reliability of the host sis called into question when we start using multiple machines.

MapReduce [4] is a programming model that groups the large data sets with a parallel algorithm that lay foundation for efficient processing and generating large data sets. Hadoop [12] is one of the most popular open source implementations of the MapReduce framework that is designed to write any application. It can process a huge amount of data available on cloud environment and store in the Hadoop distributed File System (HDFS). In a large cluster environment this framework spontaneously partitions input data and handles all problems related to consistency and fault tolerance. It has been used for tasks such as analyzing application logs, aggregating related data from external sources, transforming data from one format to another, and exporting data for external analysis.

Using MapReduce for parallelizing encryption: Multiple mappers can work at once on encrypted different blocks instead of encrypting those one by one, for which using MapReduce for parallelizing encryption processes can evidently improve performance [9]. Once it is done the reducer will cluster all blocks and store them in the HDFS. There may arise a number of variations in this process. For instance, a huge amount of key can possibly reduce the decryption process. This will make the mapper to produce the output in the form of blocks. Then, the reducer take those input in the form of key value pairs and store. The reducer will process the different keys use for encryption and with the same key it processes the blocks in HDFS in a coherent way. The total number of keys used in encryption depends on the size of data as well.

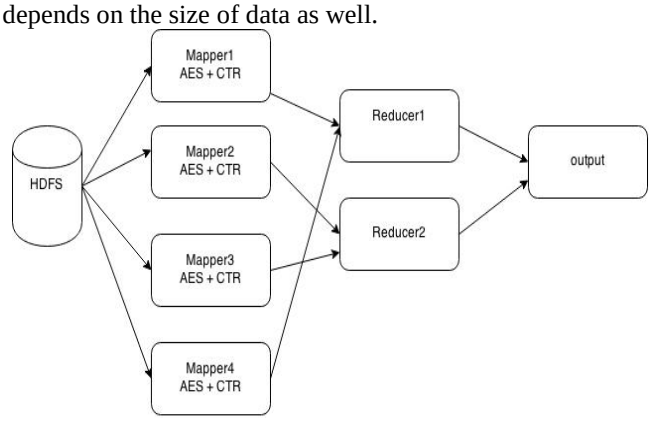


Figure 2: AES encryption using Hadoop MapReduce

Parameters in MapReduce

MapReduce is affected and its performance is delayed by the Tuning done by Hadoop configuration parameters under various conditions. Here, Table 1 lists down a few parameters of configuration parameters that we will investigate further. In this paper, based on the performance of AES in MapReduce we choose eight configuration parameters that will have its impact visibly.

Table 1: Configuration Parameters in MapReduce

Name	Parameter
Threshold	mapred.inmem.merge.threshold
Merge	io.sort.factor
Memory	mapred.job.shuffle.input.buffer.percent
	io.sort.mb
	mapred.job.shuffle.merge.percent
	mapred.job.reduce.input.buffer.percent
Reducer	mapred.reduce.tasks
Compression	mapred.compress.map.output

Here is a detailed description for the configuration parameters with respect to performance tuning:

a) Threshold: Once the reducer's task is over, the map task will be completed and the output will be copied to the tracker's memory buffer. When it reaches the threshold number of the buffer it is merged and written on the disk. The threshold can be certainly indicated by a mapred.inmem.merge.threshold parameter and the task of the parameter can save some time in merging that eventually decreases the total time consumption of the reducer.

b) b.Merge: The sorting order and the mapping output that is merged will be maintained, once the mapping process is completed. Based on the sorting factor (io.sort.factor) the filed will be selected and the default value 10 is also assigned for this parameter. In fact, for jobs this factor should be increased due to the large mapper output and huge writes on the disk.

c) Mapred.job.shuffle.input.buffer.percent: The output of the mapper is stored in memory buffer of reduce task – trackers, which is a parameter that controls the size of the memory buffer. For storing and sorting the mapper output the configuration parameter returns a percent

heap portion. In addition, the value of parameter depends on the size of the mapper output.The configuration setting of the parameters will decrease its number of disk pills in order to gain a larger mapper output.

d) d.io.sort.mb: This is a parameter that throws out the size of the buffer that is required for sorting and here the default size of the buffer is 100 MB. The size should be increased for encryption in larger clusters because the writes will be decreased in a large buffer size that will reflect its decrease in the overall time consumption.
e) Mapred.job.shuffle.merge.percent: In this parameter the threshold for the size of the task tracker's buffer is specified. It regulates on the basis of filled buffer that once it is full the buffer output will finally be written to disk.
f) Mapred.job.reduce.input.buffer.percent: It is a need that the map output is to be reserved during the reduce phase. The percentage of heap memory portion used in the process is noted by this parameter. When the percentage increases there will be a less merge that will reduce the result on the disk. Finally, during the reduce phase there will be a less reduction in the I/O time on the local disk.
g) Reducer: For a particular map- reduce job, this parameter (i.e. mapred.reduce.tasks) specifies the number of reducers to be run. It depends on both hardware configuration and the length of the resource to process the given clusters. A task-tracker, which is used to reduce tasks can improve its performance level when maximum number is set for large clusters.
h) Compression: Whether the mapper output is to be compressed or not is decided by this parameter called compression (i.e. mapred.compress.map.output). In this parameter the default value is false. While changing the parameter setting to true, we find the disk space can be saved and data transfer time can also be reduced.

5. Implementations and Evaluation

Based on the implementation on MapReduce the AES algorithm is examined and compared to the conventional AES implementation. The tests are made through Starfish [14] in Hadoop for comparing the performance of both the encryptions methods. Starfish is a reliable self-tuning system that uses a profiler and an optimizer. In this research we use Starfish profiler to gather statistical information about MapReduce programs. In addition, it is allowed to manually set the values of each configuration parameter for a MapReduce job. For each job the partial execution time and the overall time to encrypt a file is measured using this parameter.

The following are used in this process. Firstly we took up Hadoop 1.2.2 with Hadoop CryptoCodec Compressor 0.0.6. Intel 8 Cores was the testing machine used for the process with CPU E5606 2.13GHz and 12M RAM. For the experiment purpose we used a block size of 256 bits for AES and AES Counter Mode (CTR) was used additionally. CTR is used to create a pseudo random stream which is self-regulating of its plaintext. In order to eliminate the duplication, varied pseudo random streams are obtained and analyzed by counting up from the nonces of different kind otherwise the initialization vectors, using which the encryption is possible without per message randomness. It is possible that only the wrong bits are affected by the transmission errors in the process and Decryption and encryption are completely parallelizable without any hindrances.

In the MapReduce framework, the mapper first divides a file in chunks that is to be encrypted. Then, the reducer carries out the inputs with the chunked encrypted file and writes the output file to the HDFS safely through of the encryption as well as the decryption. After this process, to use the encrypt file a conventional AES algorithm is used. Crypto Codec is a commonly used compression algorithm provided by Hadoop. It is necessary that the methods of key managements and additional configuration to be provided to encryption algorithms, for which crypto codec framework is used [2].

Table 3: Used values for each configuration parameter

Parameter	Value
mapred.inmem.merge.threshold	84
io.sort.factor	49
mapred.job.shuffle.input.buffer.percent	0.734
io.sort.mb	573
mapred.job.shuffle.merge.percent	0.242
mapred.job.reduce.input.buffer.percent	0.3820
mapred.reduce.tasks	2
mapred.compress.map.output	False (for smaller file)

At first we tend to evaluate the performance of AES encryption with MapReduce or without MapReduce. Secondly, the encryption performance of the selected configuration and its impact on achieving the aim is analyzed. In Table 2 we find the derived specific values for each configuration parameter made by Starfish. Except for Fig. 3, where the total execution of time for AES encryption without or with MapReduce we have used 10MB file size for our test. The two major things are measured: Firstly, the average output written to disk by Reducer, which in turn decides the execution time for the Map and Reduce job. Secondly, using the selected configuration parameters the consumption of total time for execution.

5.1 Encryption Performance in Map reduce

Fig. 3 shows how in the MapReduce framework the encryption performance is carried out. It has achieved a far better improvement in encryption (i.e.,) between 20% ~ 30% when compared to the conventional encryption methods. Not only that when compared to the conventional AES the execution time was much greater for the AES algorithm to encrypt a 500 MB based on conventional AES. Hence, it is clear that the MapReduce allows encryption to be benefited out of parallel processing.

To innate performativity of MapReduce, parallelization serves as a core factor. Figure 3 picture that MapReduce markedly reduces the total encryption execution time. The splitting of files for encryption in accordance to the parallel processing reduces the time consumption for its possible execution. We find that in sit of using Crypto codec, the

noteworthy performance enhancement was due to the way mapper and reducer assign jobs. They serve both master and slave process in parallel to bring out the variance in encryption performance.

It is clear that compared to the map job, the performance of the reduce job may differ from configuration parameters. In Fig. 6 we find that the comparison in the overall time for the entire encryption job before and after tuning the configuration parameters. According to the eight figuring parameters we may conclude that we can obtain performance improvement by means of configuring the encryption job.

5.2 Execution Time at Reducer

As the first step, using Reducer the average output driven to disk is analyzed. Given in Fig. 4 before fine-tuning and after fine- tuning configuration parameters show the difference in the average outputs written to disk by reducer during execution time. It is proven through this that the average output written to disk by the reducer decreased by 40%, resulting in overall improvement of performance after the map-reduce job.

5.3 Total Execution Time at Mapreduce

Here the overall execution time for the map and reduce job in performing encryption in HDFS is compared. Fig. 5 brings forth the overall time execution for the map-reduce job before tuning and after tuning the configuration parameters.

5.4 Execution Time at Mapreduce

The total execution time for the map and reduce jobs are shown in the figure below. Fig. 7. It displays a comparison of the time of the map and reduce jobs to perform AES encryption in MapReduce. Through this we find that after tuning the configuration parameters, apparently the execution time of the mapper also increased, but with significant improvement in the execution time for the reducer job, the total time for the whole job was reduced.

The above stated experiments gives out two possible conclusions. The first one is that by tuning the configuration parameters there can be a marked improvement in the encryption performance. Secondly, through parallel processing using MapReduce encryption performance in terms of total execution time also improves significantly. In a conventional manner, the execution time of AES encryption for different sizes of files is examined using MapReduce with tuning positioning parameters. The results obtained using CTR AES-256 is compared to the execution time in a most predictable way and show that there is a 20% ~ 30% performance benefit with the use of the MapReduce framework. It also shows increasing performance enrichment as file size increases. Two reducers has been used for the test through which we find, the mapper needs to spend more time to segregating jobs for these multiple reducers. On the other hand, each of the reducers have only some part of the encrypted file and the time of the reducers can decrease due to the parameter settings.

6. Conclusion

The providence of any confidential data is considered as the core important thing to maintain in current scenario. In such case data leakage moving around in internet is highly indispensable. Many organizations in this domain are bound to process Big Data in a non-accessible way (i.e.,) a high quality of data services. It seems purely a difficult task of maintain such Big Data because they easily face weaknesses in designing scalable and efficient techniques. However, Data encryption helps us in finding a near solution to protect all types of data, it still lacks in scrutinizing different encryption techniques and also in investigation and evaluation of various encryption algorithms. This paper serves to be an eye-opener by providing varied literary reviews and market researches to the problem of engaging in intense scrutiny and analysis of available encryption algorithms.

Through MapReduce this paper has tested and envisioned encryption performance of the popular parallel processing platform. The configurations selected for the study has positively affected the job performances of Map- Reduce in varied conditions. These can be considered and implemented to reach maximum encryption performance.

References

[1] Craig Gentry, Shai Halevi, and Nigel P.Smart. *Homomorphic evaluation of the AES circuit*. In Reihaneh
[2] Issues.apache.org, '[HADOOP-9331] Hadoop crypto codec framework and crypto codec implementations – ASF JIRA', 2013. [Online].Available: https://issues.apache.org/jira/browse/HADOOP-9331. [Accessed: 15- Mar-2015].
[3] J. Daemen, V. Rijmen. Rijndael: *The Advanced Encryption Standard.* Dr. Dobb's Journal, March2001.
[4] Jeffrey Dean and Sanjay Ghemawat,*MapReduce: Simplied Data Processing on Large Clusters*, OSDI 2004
[5] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. *Can homomorphic encryption be practical?* In CCSW, pages 113– 124. ACM,2011.
[6] M. Nagendra and M. Chandra Sekhar, *Performance Improvement of Advanced Encryption Algorithm using Parallel Computation*. International Journal of Software Engineering and Technology, vol 8, issue 2, p287-296, 2014
[7] N. Coffey, 'Password-based encryption', Javamex.com, 2015. [Online]. Available: http://www.javamex.com/tutorials/cryptography/pbe_key _derivation.shtml. [Accessed: 15- Mar- 2015].
[8] P. Rogaway, *Evaluation of Some Block Cipher Modes of Operation*. Technical Report, Cryptography Research and Evaluation Committees (CRYPTREC),2009.
[9] Sujitha, G., Varadharajan, M., Raj Kumar, B., & Merey Shalinie, S. (2013). *Provisioning mapreduce for improving security of cloud data*. 220-228(Journal of Atificial Intelligence 6(3)))

[10] C. Gentry, *Fully homomorphic encryption using ideal lattices*, Symposium on the Theory of Computing (STOC), 2009, pp.169-178.
[11] Owen Harrison, John Waldron, *Practical Symmetric Key Cryptography on Modern Graphics Hardware*, Proc. of the 17th conference on Security symposium. San Jose, CA, 2008, pp.195-209.
[12] https://hadoop.apache.org/
[13] Impetus, *Hadoop Perfromance Tuning*, White Paper, Impetus Technologies Inc., Oct. 2009, Partners in Software R&D and Engineering. [Online]. Available: www.impetus.com.
[14] Herodotos Herodotou, Harold Lim, et. al., Starfish: A Self-tuning System for Big Data Analytics, in the Fifth Biennial Conference on Innovative Data Systems Research(CIDR), page 261-272,2011.