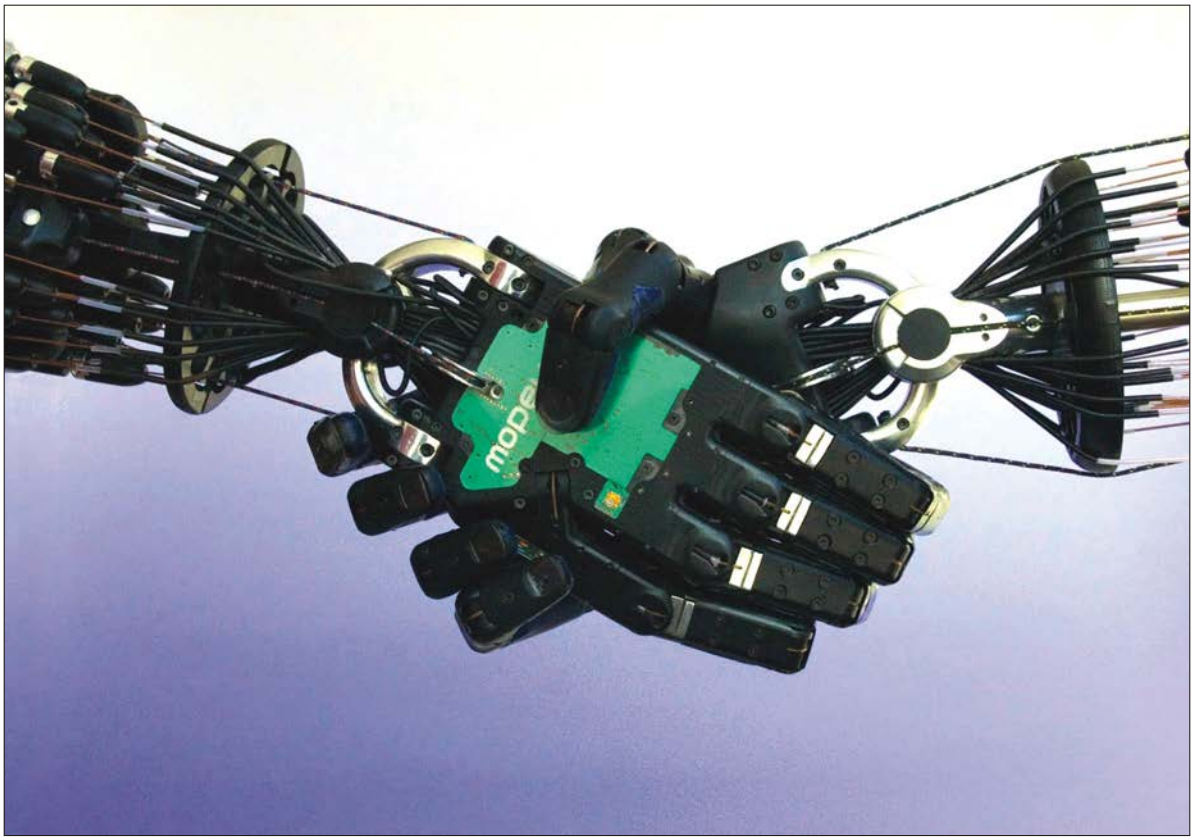

Journal of Artificial Intelligence

ISSN 1994-5450

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Artificial Intelligence 6 (3): 220-228, 2013
ISSN 1994-5450 / DOI: 10.3923/jai.2013.220.228

Provisioning Mapreduce for Improving Security of Cloud Data

[1]G. Sujitha, [2]M. Varadharajan, [1]B. Raj Kumar and [2]S. Mercy Shalinie
[1]Anna University, College of Engineering, Chennai, TamilNadu, India
[2]Anna University, Thiagarajar College of Engineering, Madurai, TamilNadu, India

Corresponding Author: G. Sujitha, Anna University, College of Engineering, Chennai, TamilNadu, India

ABSTRACT

The rising abuse of information stored on large data centres in cloud emphasizes the need to safeguard the data. Despite adopting strict authentication policies for cloud users, data while transferred over secure channel when reaches data centre, is vulnerable to numerous attacks. The most widely adaptable methodology of safeguarding cloud data is through encryption algorithms. Encrypting data at rest prevents unauthorized access of confidential information. Encryption of large data deployed in cloud is actually a time consuming process which needs to be controlled by an efficient application of the process in parallel mode. This study proposes a method to perform encryption in parallel, using standard XTS-AES (Xor based Tweaked Cipher Text Stealing-Advanced Encryption Standard) approach in MapReduce paradigm. The proposed methodology gives efficient results than using it in ECB (Electronic Code Book) mode. The time lapsed for performing the process is relatively less for user generated content. Parallel algorithm results show that AES encryption process on cloud data tends to be faster with mapper alone than running the encryption process under mapper and reducer. The results generated for encryption followed by gzip compression on different dataset like text, image audio and video proves that the proposed approach is well suited for protecting user generated data deployed in the cloud environment.

Key words: Cloud, data security, parallel algorithm, mapreduce, AES, compression

INTRODUCTION

The advent of Web 2.0 has made organizations move towards cloud with computing models using Information Technology (IT) as a Service. This emerging field in information technology focuses on applications running over the Internet such as SaaS (Software as a Service) or hardware systems available in datacenters. In both the cases management of data and services rendered by datacenters are not fully trustworthy. Enterprises moving towards emerging cloud or cloud computing model while expanding their infrastructure face risks due to security of their data (Chen *et al*., 2010). Major issues identified in cloud are data integrity, recovery, privacy and legal issues like regulatory compliance, auditing need to be resolved (Kaufman, 2009).

In recent years data storage has enjoyed a period of unprecedented growth. According to Hype Cycle of BigData, the survey showed the major challenges identified in recent years. Nearly 47% of enterprises face data growth as a major challenge (Lapkin, 2012). Several security issues have been identified in the current cloud era. It is necessary to secure data at rest. Hence, storage encryption is proposed to invoke confidentiality of large data encryption. This being

a time consuming process is controlled by an efficient application of the process in parallel mode (Abadi, 2009). In this regard Hadoop's MapReduce an open source implementation seems to be an attractive cost effective solution for processing large scale data (Nicolae *et al.*, 2010).

This study proposes a method to secure large data deployed in cloud through encryption using MapReduce and standard XTS-AES mode (Martin, 2010). The contribution of the study is summarized as (1) Appropriate algorithm for encrypting large datasets, (2) Analyzing the best possible mode that support the encryption process to run parallely and (3) Introducing Map Reduce paradigm for performing the encryption process.

MATERIALS AND METHODS

Encryption algorithm: Towards the end of the 20th century many backdoors and flaws were found in the existing symmetric algorithm like DES (Data Encryption Standard) which is vulnerable to brute force attacks because of its relatively small 56 bits key size. To meet better security standard, the US Government agency National Institute of Standards and Technology (NIST) selected Rijndael's Algorithm as Advanced Encryption Standard (AES) and it has now become the industry standard. AES is typically designed to accept three different key sizes like 128,192 or 256 bits. The algorithm is capable of encrypting bulk data on top-end 32 bit and 64 bit CPUs. It has been proved as an efficient algorithm for encrypting all sorts of data deployed in cloud from text to audio and video (Schneier and Whiting, 2000). The performance of AES algorithms vary on various CPUs based on the key size (which is our forthcoming work) and its performance in the current scenario is evaluated in this study. Parallelizing the block contents involves various modes of operation that handles fixed block and variable block encryption through single and different keys.

Encryption mode: The efficiency of cryptographic algorithm that run in parallel mode can be improved by the introduction of several modes that are intended to use with symmetric block cipher like AES. The Electronic Code Book (ECB) (Dworkin, 2005) and XTX (Keller and Hall, 2011) modes are widely supported parallel encryption mechanisms. ECB supports the kind of encryption in which the plaintext consists of a sequence of bit blocks like b1, b2,.....bn that is converted into corresponding cipher text blocks such as c_1, c_2,.....cn through the same key that acts on all the blocks. ECB mode suffers from the fact that if the message has repetitive elements in different blocks all of them produce the same cipher text which is possible to substitute and find the actual plaintext. The other widely used IEEE 1619-2007 standard is XEX-TCB-CTS (XTS) mode in which key material for XTS-AES consists of an encryption key as well as a tweak key that is used to incorporate the logical position of the data block into the encryption. XTS outputs tend to produce independent outputs which lead to the parallelization of such methodology. XTS is a concrete instantiation of the class of tweak able block ciphers. The XTS mode allows parallelization and pipelining in cipher implementations. It enables the encryption of the last incomplete block of data while other modes are not having this facility. Figure 1 depict AES-XTS mode with two different keys which differ based on the block it resides.

Hadoop's MapReduce paradigm: Google proposed MapReduce to simplify data processing on large clusters (propriteary). Hadoop is the most popular open source implementation of the MapReduce framework developed by Yahoo! and the Apache software foundation (Nicolae *et al.*, 2010). Map Reduce is deployed as a powerful data processing service over open source systems and has become increasingly popular for its parallel programming framework. Hadoop's

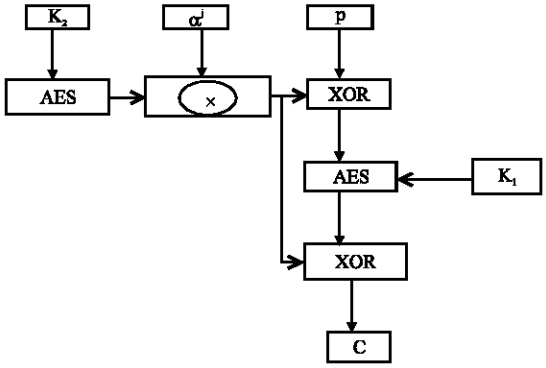

Fig. 1: Depicts AES-XTS mode with two different keys which differ based on the block it resides

MapReduce seems to be an attractive cost effective solution for large scale data processing services like securing data in cloud through block encryption (Pavlo *et al.*, 2009). MapReduce can fit in any kind of environment like closed or open systems. The framework is designed for writing applications that rapidly process vast data during runtime on compute clusters. The code automatically partitions input data, performs scheduling, monitoring and have the fault tolerance mechanism through which it re executes the failed tasks in a commodity of large cluster machines (Dean and Ghemawat, 2008; Pavlo *et al.*, 2009).

Formal definition of MapReduce programming: MapReduce is a simple model a with two keys such as map and reduce derived from functional programming languages (Nicolae *et al.*, 2010). For any kind of application to run using map reduce model, the input should be a set of key value pair. The mapper produces an intermediate key pair value that is sent to the reducer to carry out the rest of the application process. The explanation includes defining mappers and reducers. A description of how the system executes these two functions is also presented. The fundamental unit of data in map reduce computations is the <key; value> pair, where keys and values are always just binary strings.

Definition 1: A mapper is a randomized function that takes as input one ordered <key; value> pair of binary strings. As output, the mapper produces a finite multiset of new <key; value> pairs. It is important that the mapper operates on one <key; value> pair at a time.

Definition 2: A reducer is a randomized function that takes as input a binary string which is the key, and a sequence of values v_1, v_2 which are also binary strings. As output, the reducer produces a multiset pair of binary strings <k; $v_{k,1}$>, <k; $v_{k,2}$>, <k; $v_{k,3}$>,. The key in the output tuples is identical to the key in the input tuple.

Parallelizing encryption through map reduce process: As stated before, the main benefit of this programming paradigm is the ease of parallelization. Since each mapper μr only operates on only one tuple at a time, the system can have many instances of μr operating on different tuples in U_{r-1} in parallel. After the map step, the system partitions the set of tuples output throughout the instances of μr which are created based on their key. That is, part i of the partition has all key; value pairs that have key ki. Since reducer ρr only operates on one part of this partition, the system

based on the application create many instances of ρr running on different parts in parallel. The data will be stored as contiguous blocks and every block is represented by an unique block id ($I_0,I_1,I_2.....I_{n-1}$). A MapReduce includes set of mappers ($M_1,M_2.....M_r$) and reducers ($R_1,R_2.....R_r$).The input is given to mapper in the form of <block id, object>.This object is the content of the HDFS (Hadoop Distributed File System)block or the data stored in the corresponding block id. During encryption the mapper and the reducer function based on the <key; value> pair as discussed above.

Execution of mapper: Block $<I_{r-1}$,object ds1> is given to mapper M_r. The mapper will generate the corresponding encrypted output I'_R and send it to the reducer $R_r$let.

$W = \{<I_0,\text{object1}>,<I_1,\text{object2}>,<I_2,\text{object3}>.....<I_{n-1},\text{object n}>\}$ then general format is represented as.
$I'_r = I_{r-1}\text{-}W_{<\text{blockid,object}>}\ M_r\ (<\text{blockid,Enc-comp}_{\text{object}}>)$

Execution of reducer: The collected outputs from various mappers are written to the disk in the sequential order ($I'_1,I'_2.....I'_n$). In the proposed work the algorithm is designed in such a way for securing large data sets by performing block encryption followed by compression under each mapper and combining the result and storing it on HDFS (Hadoop Distributed File System). The Mapper reads block of equal size which can be optimized based on the available free nodes in the cloud environment as shown below:

$$\text{Block size} = \frac{\text{Input data size}}{\text{No. of nodes configured for mapper}}$$

Figure 2 shows how HDFS contents got assigned to mapper where mapper performs encryption followed by compression and the reducer is used to write the contents on to HDFS (i.e., output).

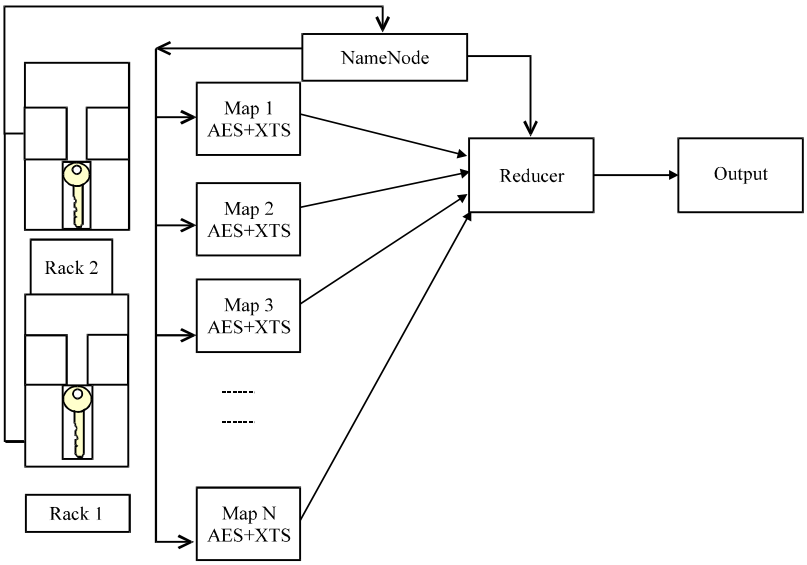

Fig. 2: Encryption using map reduce framework

Though HDFS (Hadoop Distributed File System) supports record level and block level compression on input data, in the proposed method after encryption under each mapper, a compression through gzip is done in order to reduce the storage space.

RESULTS

Experimental setup: The AES algorithm running under MapReduce was tested on a campus environment that was created as a HadoopTestbed. This testbed comprised of a 32 node cluster, each of which has an Intel Xeon 1.6 Ghz processor with 500 GB of local storage running on Hadoop 0.20. The initial tests were made with earlier versions of Hadoop.19. The testbed for protecting data in cloud environment was experimented with series of small datasets. The experimental results showed that encryption followed by compression mechanism using such parallel techniques required less time in the order of seconds for large datasets. Experimental results were explained with the time lapsed for running the same process with reducer and without reducer. A comparison graph was generated for encrypting different datasets and for different types of files like audio, video, text and image. The testbed is being utilized to perform encryption followed by compression while storing different datasets as in the cloud environment.

Experiment 1: Comparison of modes that support parallelism with AES: Figure 3 shows the time taken for symmetric AES encryption with ECB mode and XTS mode. The results prove that the time difference between the two modes was very less which implies that AES with XTS was adaptable for protecting data in cloud. While performing encryption using XTS mode, it produced a different output for the same data from two different mappers whereas EBC produced the same cipher the text from different mappers. Such a concept was found to be relatively insecure. The time difference for was 15 GB text data was 3 min which indicates that XTS with AES suits for securing large datasets in cloud environment.

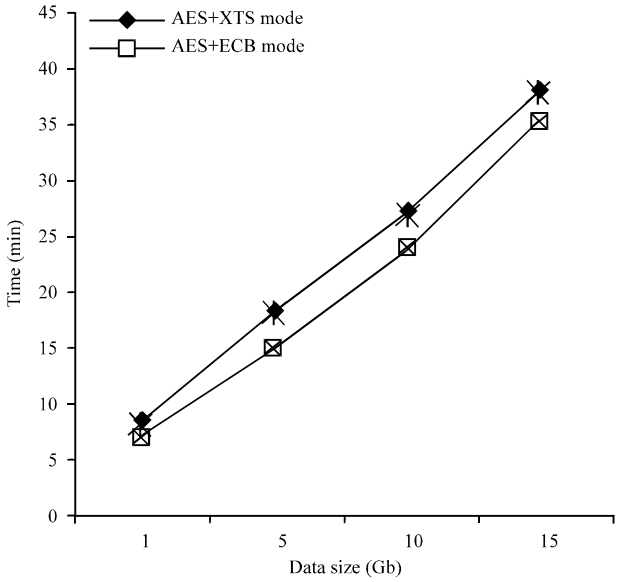

Fig. 3: Time taken for symmetric AES encryption with ECB mode and XTS mode

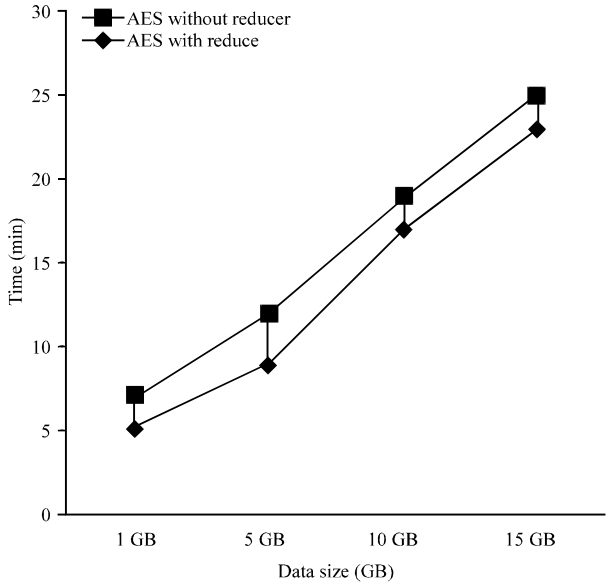

Fig. 4: Performance of the algorithm with reducer and without reducer

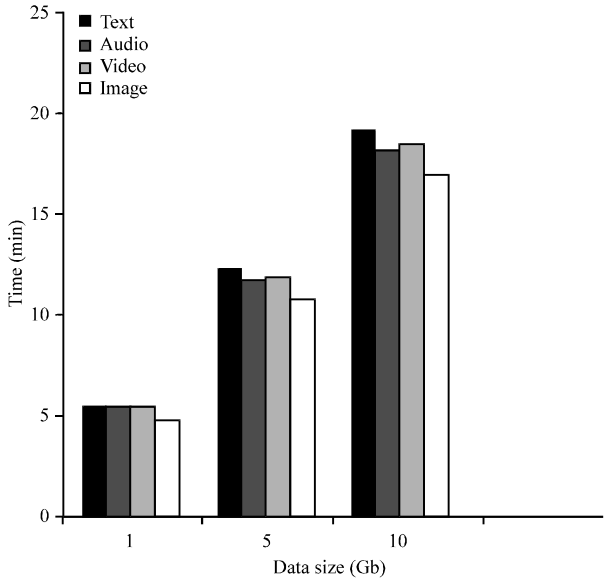

Fig. 5: Evaluation of the proposed approach for different datasets

Experiment 2: Execution of MapReduce for encryption with reducer and without reducer: Figure 4 shows the performance of MapReduce Algorithm using AES (under XTS Mode) with reducer and without reducer. When it is used without reducer all the nodes of the cluster were assigned as mapper and hence it increases the speed of the encryption followed by compression process.

Experiment 3: Evaluation of AES+XTS encryption using mapreduce process for different datasets: Figure 5 gives the evaluation of the proposed approach for different datasets. It was

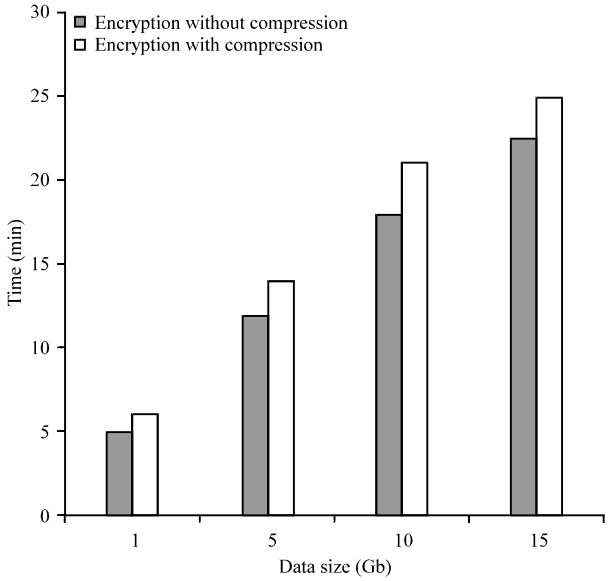

Fig. 6: Performance evaluation of compression technique on different datasets

inferred from the results generated that the proposed approach shows remarkable difference for large size of text, audio, video and image files than for small size user generated content files. It was identified that the time taken for encrypting text was relatively high but less for image files. Audio and video files required relatively equal time for performing the process.

Experiment 4: Performance evaluation of compression on different dataset without reducer: From the previous experiment 2 it was concluded that without reducer the performance of the algorithm was better. So compression on large dataset was experimented with a setup where all the nodes of the cluster act as mapper. Figure 6 shows the result that encryption with compression takes relatively less time. From the experiment with various datasets the compression ratio obtained for text and image data were in the order of 1:10 and 1:2. But for audio and video files no remarkable difference in the ratio was noticed.

DISCUSSION

Many security issues related to cloud were identified during large datastore, data reduction technique, effective storage and data archiving (Foster *et al*., 2008). Several programming models have been introduced to solve the issues related to large data in cloud through parallelize computation (Chaiken *et al*., 2008; Miceli *et al*., 2009). In this study protecting cloud data through encryption and to efficiently utilize the resources of the cloud, compression using MapReduce has found to be model is proposed and implemented. The model has found to be adaptable for different datasets like audio, video, text etc., MapReduce is considered as a simplified model designed for compute cloud have and Grids. MapReduce and its derivatives have the ability to operate on large dat (Abadi, 2009; Dean and Ghemawat, 2008). Parallel Algorithms like Message Passing Interface (MPI) and Bulk Synchronous Programming provide a high level of abstraction that allows users to write parallel programs (Valiant, 1997; Gropp *et al*., 1999). When compared with other parallel

code, Hadoop's MapReduce is designed to parallelize user program automatically. It has an additional feature of providing fault-tolerant mechanism. The Hadoop's MapReduce implementation proposed in our study allows performing encryption in parallel. Among the various modes of encryption, XTS proved to be slower than narrow block mode. But narrow block requires the more time when dealing with large datasets (Dworkin, 2005; El-Fotouh and Diepold, 2008). So the present study reveals that XTS mode which has a special tweak key is found to be secure than the other modes of encryption. It is suggested that compressing encrypted data, tends to be efficient in terms of storage space (Johnson *et al*., 2004; Ziv and Lempel, 1977).

In the present study, improved level of security is achieved through MapReduce framework which is being used in cloud world in recent years. The model was designed with three objectives. Firstly, ECB and XTS mode that supports parallelism, second XTS mode with AES running under mapper and reducer inorder to provide security to the user's data and thirdly the performance evaluation of the entire framework with and without using reducer. Finally the hybrid framework is tested for different datasets and the evaluation is done. It is clear from the results that AES with XTS mode running under mapper gives better result than using mapper and reducer together. The experimental result of the proposed technique shows that encryption methodology requires relatively less time for large datasets in a cluster of machines. The compression is done at the expense of time in order to utilize the resources of storage environment. Future enhancements of the work needs to focus on evaluating the performance under different configuration parameters and improve the kind of encryption through a selective mode. Moreover, compression of audio and video files based on the type of codec which is our future area of research.

ACKNOWLEDGMENT

We acknowledge Yahoo! India, Banglore for giving valuable suggestions to complete this study.

REFERENCES

Abadi, D.J., 2009. Data management in the cloud: Limitations and opportunities. IEEE Data Eng. Bull., 32: 3-12.

Chaiken, R., B. Jenkins, P.A. Larson, B. Ramsey, D. Shakib, S. Weaver and J. Zhou, 2008. Scope: Easy and efficient parallel processing of massive data sets. Proceedings of the VLDB, August 24-30, 2008, Auckland, New Zealand, pp: 12.

Chen, Y., V. Paxson and R.H. Katz, 2010. What's new about cloud computing security? Berkeley University of California, Report No. UCB/EECS-2010-5, January 20, 2010.

Dean, J. and S. Ghemawat, 2008. Mapreduce: Simplified data processing on large clusters. Comm. ACM, 51: 107-113.

Dworkin, M., 2005. Recommendation for block cipher modes of operation: The CMAC mode for authentication. National Institute of Standards and Technology, Special Publication 800-38B. csrc.nist.gov/publications/nistpubs/800-38C/SP800-38B.pdf.

El-Fotouh, M.A. and K. Diepold, 2008. A new narrow block mode of operations for disk encryption. Proceedings of the 4th International Conference on Information Assurance and Security, September 8-10, 2008, IEEE Computer Society, Napoli, Italy, pp: 126-131.

Foster, I., Y. Zhao, I. Raicu and S. Lu, 2008. Cloud computing and grid computing 360-degree compared. Proceedings of the Grid Computing Environments Workshop, November 16, 2008, Austin, Texas, pp: 1-10.

Gropp, W., E. Lusk and R. Thakur, 1999. Using MPI: Portable Parallel Programming with the Message-Passing Interface. MIT Press, ISBN: 0-262-57133-1, Cambridge, MA.

Johnson, M., P. Ishwar, V. Prabhakaran, D. Schonberg and K. Ramchandran, 2004. On compressing encrypted data. IEEE Trans. Signal Process., 52: 2992-3006.

Kaufman, L.M., 2009. Data security in the world of cloud computing. IEEE Security Privacy, 7: 61-64.

Keller, S.S. and T.A. Hall, 2011. The XTS-AES validation system. NIST-National Institute of Standards and Technology-XTS-AES(2011).

Lapkin, A., 2012. Hype cycle for big data. Gartner Inc. http://www.hadoopconsultant.nl/wp-content/uploads/hype_cycle_for_big_data_2012_235042.pdf

Martin, L., 2010. XTS: A mode of AES for encrypting hard disks. IEEE Secur. Privacy, 8: 68-69.

Miceli, C., M. Miceli, S. Jha, H. Kaiser and A. Merzky, 2009. Programming abstractions for data intensive computing on clouds and grids. Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, July 21, 2009, IEEE Computer Society, pp: 478-483.

Nicolae, B., D. Moise, G. Antoniu, L. Bouge and M. Dorier, 2010. BlobSeer: Bringing high throughput under heavy concurrency to hadoop map-reduce applications. Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium, April 19-23, 2010, Atlanta, GA., pp: 1-11.

Pavlo, A., E. Paulson, A. Rasin, D.J. Abadi, D.J. DeWitt, S. Madden and M. Stonebraker, 2009. A comparison of approaches to large-scale data analysis. Proceedings of the International Conference on Management of Data, June 29-July 02, 2009, Providence, RI, USA., pp: 165-178.

Schneier, B. and D. Whiting, 2000. A performance comparison of 5 AES finalists. Proceedings of the 3rd AES Candidate Conference, April 13-14, 2000, Hilton, New York, pp: 231-249.

Valiant, L.G., 1997. A bridging model for parallel computation. Communi. ACM, 33: 103-111.

Ziv, J. and A. Lempel, 1977. A universal algorithm for sequential data compression. IEEE Trans. Inform. Theory, 23: 337-343.