Intelligent Secure Storage Mechanism for Big Data

K.R. Remesh Babu
Department of Information Technology, Government Engineering College, Palakkad, Sreekrishnapuram, Kerala, India.
E-mail: remeshbabu@yahoo.com

K.P. Madhu
Department of Computer Science and Engineering, Government Engineering College, Idukki, Painvu, Kerala, India.
E-mail: kpmadhu@gmail.com

Received October 30, 2020; Accepted November 30, 2020
ISSN: 1735-188X
DOI: 10.14704/WEB/V18SI01/WEB18057

Abstract

The management of big data became more important due to the wide spread adoption of internet of things in various fields. The developments in technology, science, human habits, etc., generates massive amount of data, so it is increasingly important to store and protect these data from attacks. Big data analytics is now a hot topic. The data storage facility provided by the cloud computing enabled business organizations to overcome the burden of huge data storage and maintenance. Also, several distributed cloud applications supports them to analyze this data for taking appropriate decisions. The dynamic growth of data and data intensive applications demands an efficient intelligent storage mechanism for big data. The proposed system analyzes IP packets for vulnerabilities and classifies data nodes as reliable and unreliable nodes for the efficient data storage. The proposed Apriori algorithm based method automatically classifies the nodes for intelligent secure storage mechanism for the distributed big data storage.

Keywords

Big Data, Cloud, Secure Storage, Intelligent Mechanism, Quality of Service.

Introduction

The developments in technology, science, human habits, etc., generates massive amount of data. This rapid growth of data volume, diversity and veracity and speed of data-intensive applications demands a competent storage management system. It is also important that these storage locations should be reliable and free from external attacks. So there is exists need for an intelligent storage mechanism to efficiently and effectively

place the data in reliable locations. Now, most of the research works are mainly focused on memory management and virtual machine placement strategies. Like in virtual machine (VM) management scheme, the storage array's algorithms are needed to place data efficiently in secure locations. It is a cumbersome and challenging task to maintain Quality of Service (QoS) requirements (thereby to meet Service Level Agreements/Service Level Objectives) for above data intensive applications. All these factors demands for an intelligent mechanism with efficiency for the management of big data storage infrastructure.

Today, there are many service providers offers data storage facility all over the world. In cloud paradigm the storage locations i.e., data centers are located in different geographical locations. The service provider decides where to store users' data and type of storage device. Also, they have to decide the geographical location and whether data is to be stored on a co-located storage owned by others as per users' request. The green computing standards in power consumption makes it more complex. These challenging problems inspire this research work, which proposes a novel model, based on Apriori algorithm, which helps in the proper management of data and its storage infrastructure in a secure manner.

The proposed technique analyses and optimizes the multi-tiered data storage systems for a cloud data center. Here data nodes are classified according to its reliability. The main purpose of this method is to develop a novel modelling method to describe and help in the most suitable and reliable data placement. The main aim of this method is to place data on the most efficient target nodes in tiered cloud data storage architecture for better response time and reliability. In the proposed method, the allocation algorithm will classify the data nodes as reliable and unreliable for the effective and efficient data placement. The method triggers whenever necessary for data migrations for the optimized data storage. The model considers response time for providing better QoS to the customers.

In the cloud environment MapReduce framework proved its effectiveness for the distribute data processing [2]. This framework is simpler but powerful and efficient in handling large quantities of data. Now more sophisticated services are offered by commercial cloud providers in this environment, which made this framework more suitable for distributed computing services to all. Different machine learning algorithms can be created and applied in the cloud environment by Mahout framework provided by the Apache Foundation [3][16]. It provides effective tools for the analysis of large amount of data sets on clusters within a short period of time. Distributed big data processing and its storage in commodity hardware is the main highlight of this framework thus.

MapReduce concept can be applied to different applications, such as machine learning [4], document clustering, web log mining, distributed sorting, pattern based searching. The paper [5] successfully used map reduce framework for dynamic cloud environment. This framework can be used for multi and many core processor systems [6][7][8]. Another reason to take MapReduce platform is that, it proved effectiveness in computational grids [9] volunteer computing environments [10] and mobile environments [11].

A. Apriori Algorithm

Apriori algorithm is meant to determine frequent or large item set which in turn helps in discovering association rules [3][13][14][17]. The researchers have incorporated several improvements in the basic Apriori algorithm [24]. One of the significant advancements is the development of a protocol for secure mining of association rules in databases which is distributed horizontally [15]. This protocol is substantially more efficient in terms of communication rounds involved, communication cost incurred and computational cost experienced. Though Apriori algorithm is basically intended for data mining, this could be used in large networks for intrusion detection [12].

The pseudo code for basic steps in Apriori algorithm is given in Fig. 1. Here, a frequent *k-itemset* is an itemset with support greater than the user-specified threshold. Let this be denoted as L_k, where *k* is the size of the itemset. A candidate *k-itemset* is the set of potentially frequent itemset of size *k*. It can be denoted as C_k, where *k* is the size of the itemset. The algorithm proceeds in an iterative manner. The algorithm will execute a minimum of *k* passes to extract frequent *k-itemsets*.

Pass 1
1. Beget the candidate *1-itemsets* in C_1
2. Extract the frequent *1-itemsets* into L_1

Pass *k*
1. Beget the candidate *k-itemsets* in C_k from the frequent *(k-1)-itemsets* in L_{k-1}
   1. Join L_{k-1} *p* with $L_{k-1}q$, as follows:
      insert into C_k
      select $p.\text{item}_1, p.\text{item}_2, \ldots, p.\text{item}_{k-1}, q.\text{item}_{k-1}$
      from L_{k-1} *p*, $L_{k-1}q$
      where $p.\text{item}_1 = q.\text{item}_1, \ldots p.\text{item}_{k-2} = q.\text{item}_{k-2}, \ p.\text{item}_{k-1} < q.\text{item}_{k-1}$
   2. Produce all the possible (*k*-1)-subsets from the candidate *itemsets* in C_k
   3. Prune all candidate *itemsets* from C_k if any (*k-1*)-subset of the candidate *itemset* is not in the frequent *itemset* L_{k-1}
2. Scan the input transaction database to determine the support for each candidate *itemset* in C_k
3. Save the frequent *itemsets* into L_k

Fig. 1 Basic steps in Apriori Algorithm

The Apriori algorithm was successfully used in dynamic cloud computing area [25][28], virtual database design [26], and in distributed database design [27]. This shows it power in distributed processing.

This paper proposes a novel method based on Apriori algorithm that creates a model for generating rules for discovering reliable data nodes. Here breadth-first search and hash tables are used to count candidate nodes for reliable data storage. The detailed working of this method is given in the section 5.

The rest of this paper is organized as follows. The section 2 gives a detailed overview of the various advancements in distributed data processing. The proposed method is described in section 3. The architecture is given in section 4. Section 5 illustrates the working of the proposed method. The performance evaluation is given in section 6. The paper concludes with the findings in section 7.

Related Works

The data storage mechanisms are changed due to the huge amount of data generation in today's world. The most widely used distributed data storage framework is Hadoop. It makes use of HDFS for the distributed storage. There are many techniques that can be used with Hadoop MapReduce jobs to boost performance. The paper [1] briefs about Hadoop MapReduce and how it can be used for Big Data processing. The authors also described about different data management techniques.

The paper [18] analyzes the Infrastructure as a Service (IaaS) cloud's performance analysis, power consumption and resource utilization of applications on virtual Hadoop clusters. They have evaluated the performance based on the above parameters using different VM placement methods. Finally, reached a conclusion on optimal VM placement strategy based on the experiments conducted using different benchmark tools on virtual clusters.

When large number of small files stored in Hadoop nodes, memory management is a major concern. The paper [19] focusing on reduce read latency and heavy load problem of metadata server while handling small file workloads.

The big data storage and its processing is a cumbersome task in the business world. The researchers proposed several storage management mechanisms to improve the performance. To address this, the paper [29] proposed a distributed storage systems consists of workload balancing module with adaptive resource management framework

for cloud. Here the overloaded and under loaded nodes in the cluster are identified by the workload monitoring module. To regulate and balance workload among nodes, split, merge and pair algorithms called are used. In order to regulate VMs they also designed a Resource Reallocate Algorithm (RRA) which also improves performance.

The data storage in dynamic cloud was a cumbersome task. In paper [30] an analysis of dynamic social cloud was done to study and improve the data availability. The paper [20] presented a novel approach for storage optimization of small size files on HDFS. Their experimental result shows that it reduces the burden of Name Node and improves the throughput of accessing small files. This method uses a combination of file merging method with two-level pre fetching mechanism to mitigate small file problems of HDFS. It is not a general purpose solution and limited to power point (PPT) courseware files.

The paper [22] proposed a method for measuring the performance of nodes in Hadoop in cloud environment. This gives us an insight to assign tasks to each node according its performance, which will maintain good QoS for the customer. The paper [31][35][36] presents some of the basic principles and methodology to build scalable data models in a distributed environment. In mobile edge computing where data size is crucial which that affects overall performance of the system [32]. Load balanced data placement in HDFS is another intelligent method [33]. This method is tested in simulated environment in homogeneous environment. So for time-critical data-intensive applications [34] in the IoT-AI paradigm efficient storage mechanisms needed.

The Aprori algorithm can be used for finding association rules for a large scale processing [23]. It has a great influence in finding frequent item sets using candidate generation process. It focuses on how to improve the performance in processing of big data on high performance cluster. It also deals with the processing of big data in parallel on large cluster of computer nodes. This paper proposes a method that uses the above principle of Apriori algorithm that can be used to determine the reliable nodes in the Hadoop cluster based on the number of attacks to nodes.

Proposed Method

In the proposed method, the data packets to the nodes are captured and analyzed. Then classifies these nodes as reliable and unreliable nodes based on the number of attacks to the nodes. This classification of nodes is based on the reliability for efficient data storage. Here the performance is based on the time to store or retrieve data, data losses in nodes

and number of request handled per second. The proposed system consists of following modules.

1. Network packet capturing
2. Packet Feature extraction engine
3. Apriori algorithm for reliable node detection
4. Automatic Rule Generator System

The overall view of the proposed method is given the Fig 2.

A. Network Packet Capturing

Network packet capturing module captures network packets to the nodes. The user level packet capturing tool *libcap* is available in Linux environment is used as packet capturing tool. Also python *Pcapy* library supports extension to the native *libpcap* environment. The streaming data information captured by the capture system fed into NoSQL db. MongoDB with *pymongo* interface is used for data storage.

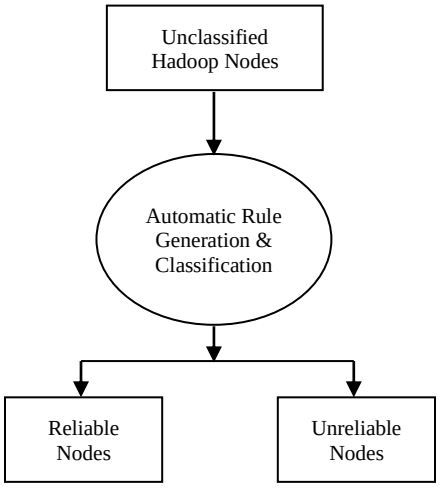

Fig. 2 Node Classification Procedure

B. Packet Feature Extraction Engine

The packet features such as TOS, TTL, Flags, checksum, source and destination addresses etc., are extracted by the Packet Feature Extraction Engine (PFEE). PFEE is based on python parser libraries. This engine is responsible for checking digital signature in packet in accordance with the normal packet data values which stored in mongoDB. The structure of the engine is given in Fig. 3.

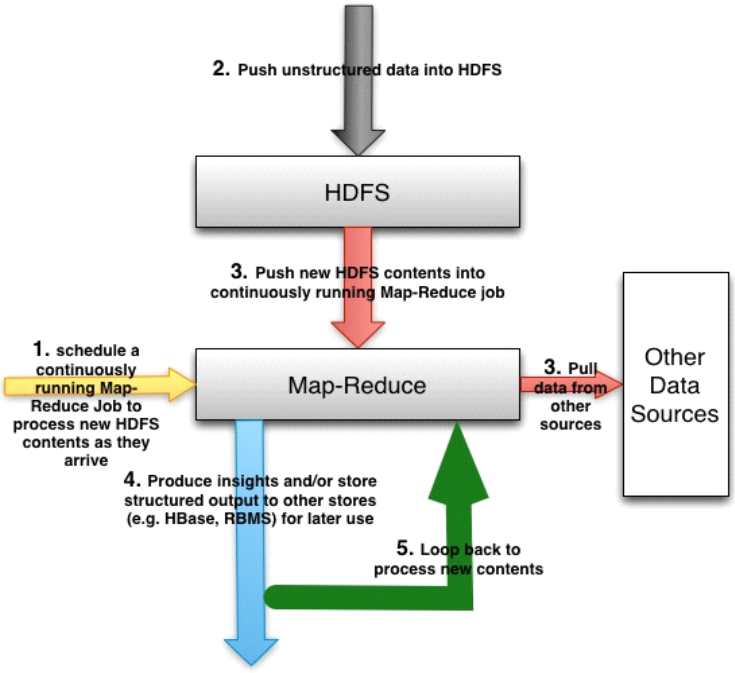

Fig. 3 Packet Feature Extraction Engine

In order to find the vulnerabilities in the packets to the nodes the proposed method checks basic signature of the packets. The following are the methods are used to in find and check the signature of packets in different conditions.

1. *Attacks from reserved IP address*: This can be identified by examining the source address field in the IP header.
2. *Handling of Illegal Packets:* The illegal TCP flags with different combination are identified. This is done by matching the flags set in a TCP header with identified good or bad flag combinations.
3. *Denial of service (DoS) attack:* This is attack is usually done by submitting the same command several times. In this kind of attack one of the signatures would be to maintain the number of times a command is issued and to alert the system when this number exceeds the threshold limit.
4. *FTP server attack:* To tackle this kind of attacks tracking signature is used. For this the method monitors FTP traffic for all successful login to the FTP server. If the user didn't authenticated it will alert before logging.

C. Apriori Algorithm based Rule Generator

The traffic to the different nodes are analyzed and rules updated based on vulnerabilities in the packets. This is module is based upon Linux packet filtering firewall environment.

The illustration of the Apriori algorithm based node detection is given in the Fig. 5.

System Design

The architecture of the proposed method is given in Fig. 4. The map reducing is done with the help of Hadoop MapReduce () function. The results are fed into the MongoDB for further use.

The Apriori algorithm is used for automatic rule generation. The packet capture module captures the packets for anomaly detection. This checks the signature part of the packets for any alterations. Then this information is passed to apriori mechanism for determining the most reliable nodes.

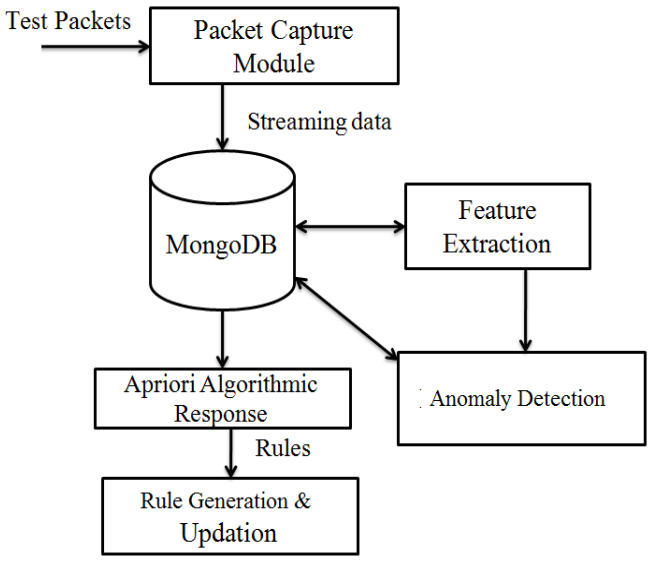

Fig.4 Architecture of Proposed System

The node addresses along with their unreliable routes are analyzed using the Apriori algorithm to determine the most vulnerable nodes. After identifying the most vulnerable nodes, the tables are updated to notify the system about these vulnerabilities.

An example of the above process is shown below: The example in fig. 5 shows the classification of data nodes. Set of nodes along with their number of attacks to the nodes are given**.** At first a candidate set C_1 of nodes are considered for observation *{1}, {2}, {3}, {4}, {5}*. From this frequent item set of nodes (L_1) generated. By analyzing the different node combinations, the candidate item set of nodes C_2 and frequent item set for nodes L_2 are generated. Finally in third step the candidate item set of nodes C_3 and frequent item

set of nodes L_3 is decided by the algorithm. Thus in the above example, the proposed system detects the most vulnerable nodes in three cycles. Here the most vulnerable nodes are {2, 3, 5}.

So in general, after every iteration the combinations of the resultant large item set of data nodes are again fed as an input to the *Apriori* algorithm so as to obtain the frequently attacked node set. This process is repeated till we reach the nodes list that had been attacked mostly or having large vulnerabilities.

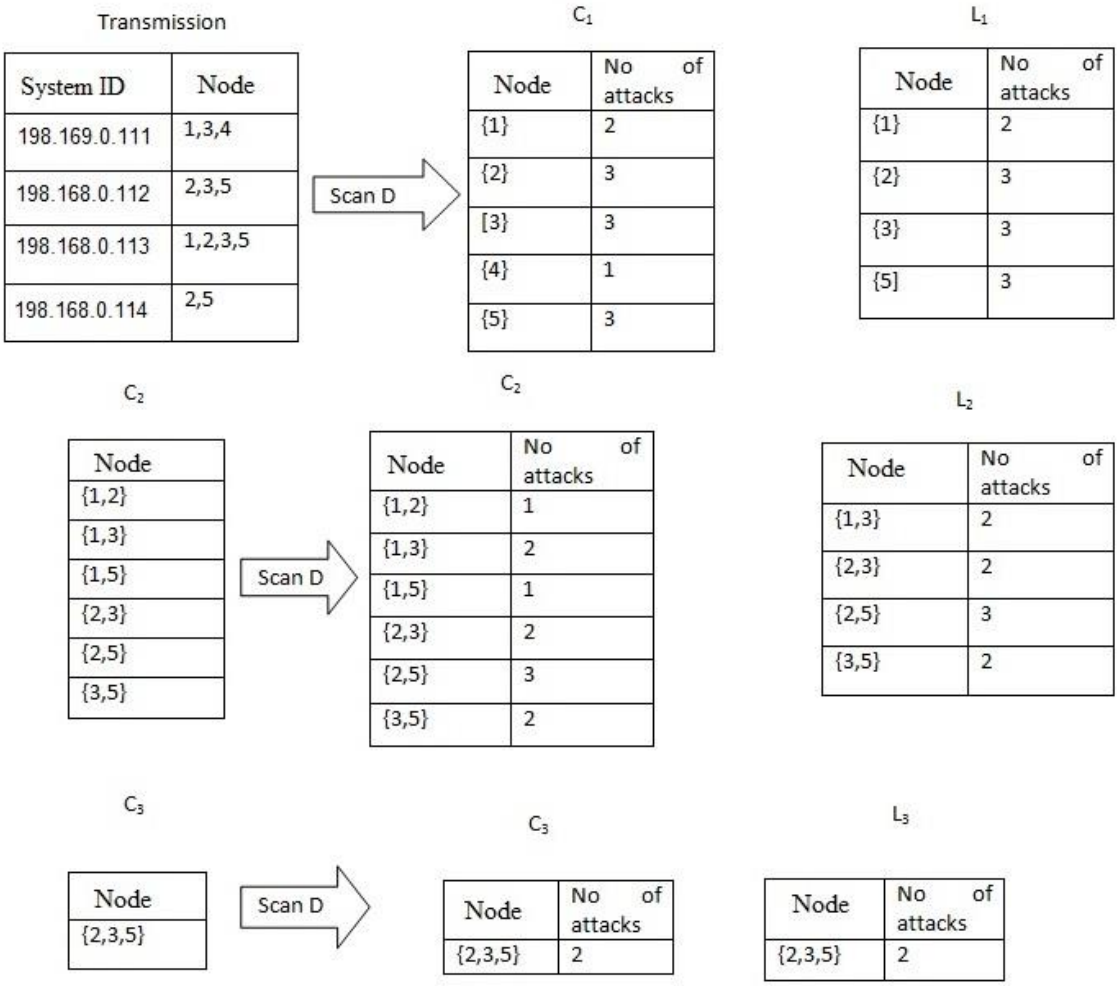

Fig. 5 Working of the Apriori algorithm for reliable node detection

Experimental Setup and Example

The system is implemented in a Hadoop cluster with 5 nodes. A pictorial representation on how it is done is shown below in figure 6. Suppose user A needs to store some data to B.

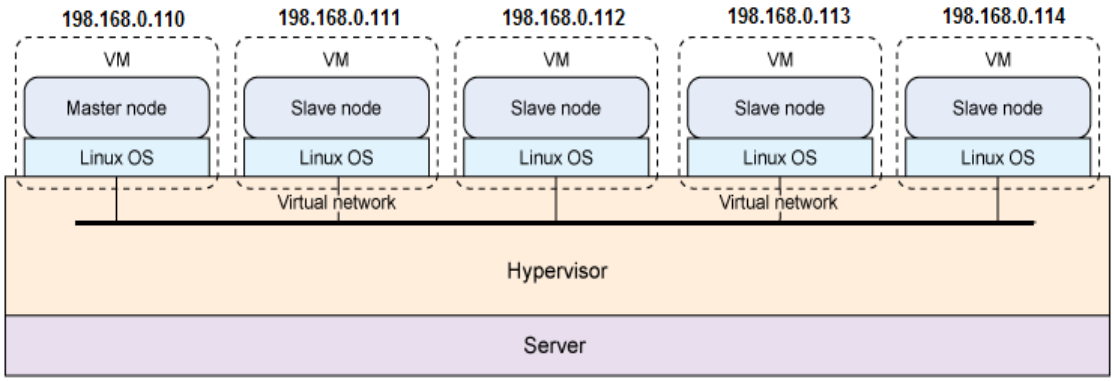

Fig. 6 HDFS with Five Nodes

A set of nodes 1, 2, 3, 4, 5 and 6 is shown in the Fig 7. The user needs to store packets to these nodes. The test packets are passed to these nodes. During each transmission different paths are followed by these packets. The paths or transmissions are said to be reliable if the receiver receives the packets without any error. After each transmission the table for nodes gets automatically updated as reliable and unreliable nodes. These details are further analyzed by the Apriori algorithm for automatic rule generation for data storage.

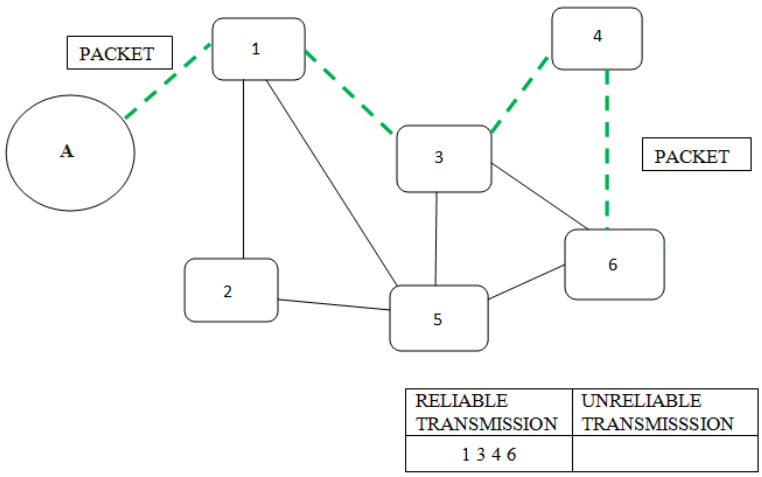

Fig.7 Reliable node detection

The Fig. 7 depicts a reliable node detection process. Since an errorless packet is delivered to B via the path 1→3→4→6 it can be considered as a reliable transmission. Thus this path is appended in the reliable transmission column of the table for future use. The reliable path is marked using green dotted lines in the above figure.

Fig. 8 illustrates an unreliable node detection scenario. The node 6 receives a packet with error when it follows the path 1→2→5→6. Thus the path 1→2→5→6 can be considered as

an unreliable path for test packet and hence it is an unreliable transmission. The unreliable path is represented using red dotted lines in the figure. From this figure we can infer that one or more nodes are in the unreliable set of nodes. These nodes are more vulnerable to external attacks and thus can notify our system.

For every test packet transmitted, the proposed mechanism checks the packet for reliability. Then it classifies the unclassified nodes as reliable nodes and unreliable nodes. This information is used by the system for further processing. After *n* transmissions there will be a set of classified nodes.

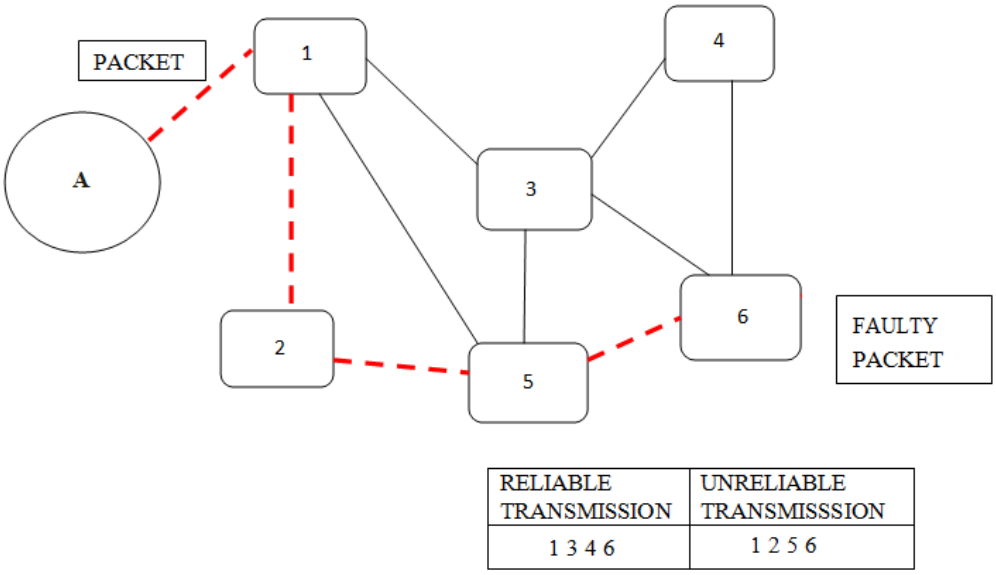

Fig.8 Unreliable node detection

The nodes are analyzed and classified based on the number of vulnerabilities happened in it. This classification helps to store the most important data to the more reliable nodes.

Performance Analysis

The performance analysis of the above method is measured based on the time for effective storing data. Times measured to store the files with different size and with different number of data nodes. The performance measured for smaller files are shown in the figure 9.

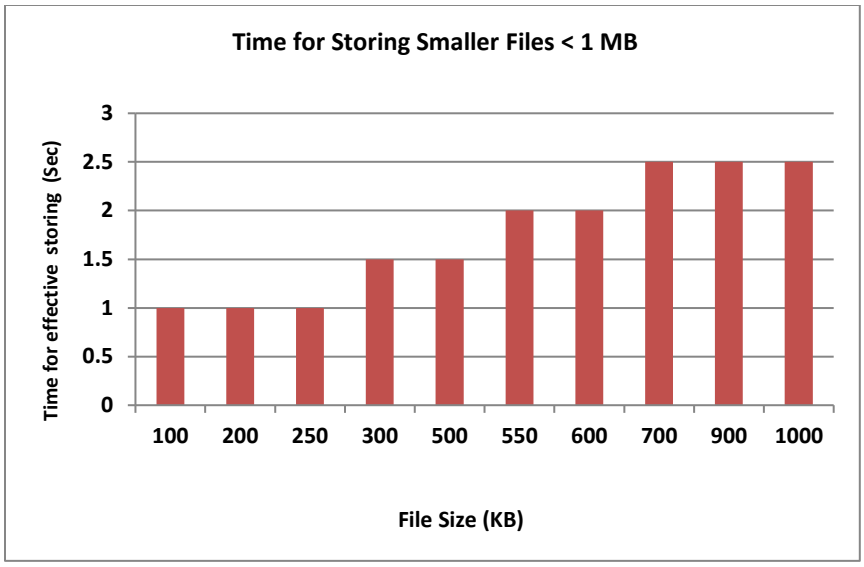

Fig.9 Storage time for smaller files

The Storage time for files with size greater than 1 MB is measured using single data node. The results are plotted in Fig. 10.

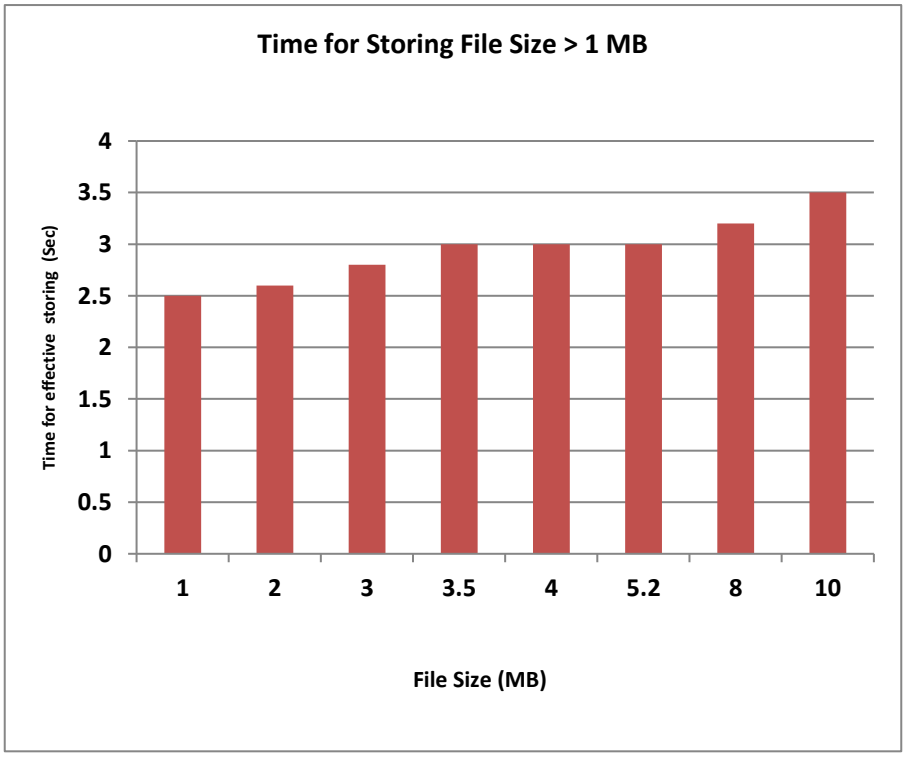

Fig.10 Storage time for files with size > 1MB

The figure 11 shows the performance comparison with files with size 1 to 10 MB and with 4 data nodes.

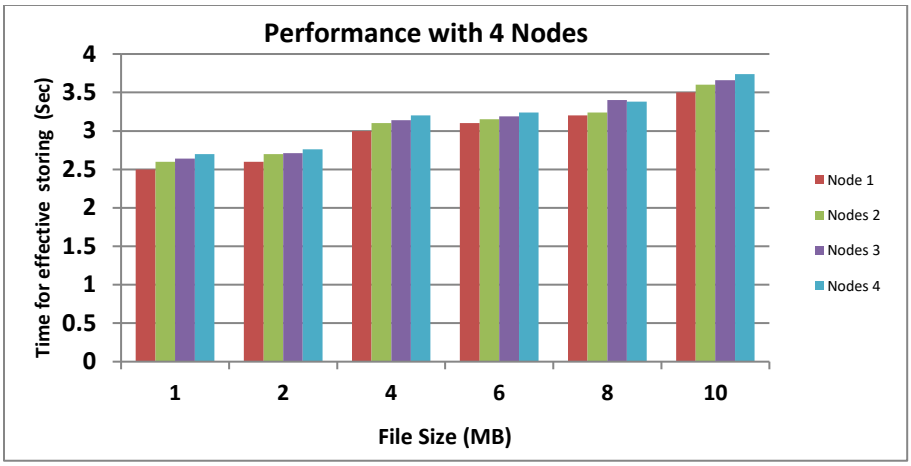

Fig. 11 Time for reliable storage

The performance of the proposed method is again tested based on the average number of request handled per second in four different conditions. The test results are shown in the figure 12. Case 1: in this environment all the server nodes are actively available and working properly. For the testing purpose we have employed four server nodes. In case 2, two simultaneous servers are used in the environment, so that if one of the server nodes fails, the system should continue working by running the second backup node. There will be certain delay in this condition due to synchronization of the system with new master node. Third scenario is similar to case 2, but here both servers not running concurrently, so that if the server node fails, new master node has to be selected. This causes potential delay in request handling, which adversely affects the performance of the system. But after this synchronization the system performance is normal. And in the fourth case, only one server node is provided. When this server node crashes, the entire system collapses.

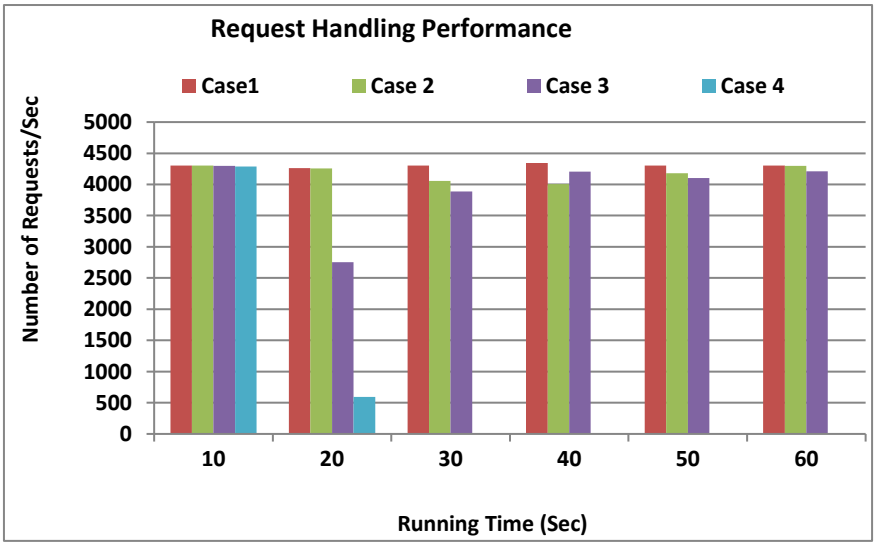

Fig. 12 Performance analysis

Conclusion and Future Works

This paper proposed an innovative model for the classification of nodes in the distributed Hadoop system. This method uses Apriori algorithm for classifying nodes as reliable and unreliable. Then this information will use to update the IP tables so as to store the data in these reliable nodes. This proposed method was tested and implemented in Hadoop cluster. This intelligent storage system provides easy management of data. This method can be extended in future to consider the quantitative impact of this storage mechanism and power efficient technologies to provide an integrated model for data storage methodologies. Also, automatic rule generator based on machine learning can be considered in future enhancement.

References

Tom, W. (2012). *Hadoop: The Definitive Guide, O'Reilly Media*. 3rd Edition.

Jeffrey, D., Sanjay, G. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM - 50th anniversary issue: 1958 - 2008, ACM New York, NY, USA, 51*(1), 107-113.

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, 1215, 487-499.

Chu, C.T., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G., Olukotun, K., & Ng, A.Y. (2007). Map-reduce for machine learning on multicore. *In Advances in neural information processing systems*, 281-288.

Marozzo, F., Talia, D., & Trunfio, P. (2012). P2P-MapReduce: Parallel data processing in dynamic Cloud environments. *Journal of Computer and System Sciences*, *78*(5), 1382-1402.

Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., & Kozyrakis, C. (2007). Evaluating mapreduce for multi-core and multiprocessor systems. *In IEEE 13th International Symposium on High Performance Computer Architecture*, 13-24.

Bingsheng, H., Wenbin, F., Qiong, L., Govindaraju, N.K., & Wang, T. (2008). Mars: a MapReduce framework on graphics processors. *Proceedings of the 17th ACM international conference on Parallel architectures and compilation techniques – PACT*, 260-269.

Chen, R., Chen, H., & Zang, B. (2010). Tiled-MapReduce: optimizing resource usages of data-parallel applications on multicore with tiling. *In Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, 523-534.

Tang, B., Moca, M., Chevalier, S., He, H., & Fedak, G. (2010). Towards mapreduce for desktop grid computing. *In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 193-200.

Lin, H., Ma, X., Archuleta, J., Feng, W.C., Gardner, M., & Zhang, Z. (2010). Moon: Mapreduce on opportunistic environments. *In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 95-106.

Dou, A., Kalogeraki, V., Gunopulos, D., Mielikainen, T., & Tuulos, V.H. (2010). Misco: a mapreduce framework for mobile systems. *In Proceedings of the 3rd international conference on pervasive technologies related to assistive environments*, 1-8.

Saboori, E., Parsazad, S., & Sanatkhani, Y. (2010). Automatic firewall rules generator for anomaly detection systems with Apriori algorithm. *In 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, *6*, V6-57-V6-60.

Nayak, A., & Stojmenovic, I. (2008). *Data Mining Algorithms II: Frequent Item Sets.* Handbook of Applied Algorithms: Solving Scientific, Engineering, and Practical Problems, Wiley-IEEE Press.

Agarwal, R., & Shafer, J.C. (1996). Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering*, *8*(6), 962-969.

Tassa, T. (2014). Secure Mining of Association Rules in Horizontally Distributed Databases. *IEEE Transactions on Knowledge and Data Engineering*, *26*(4), 970-983.

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The hadoop distributed file system. *In IEEE 26th symposium on mass storage systems and technologies (MSST)*, 1-10.

Yang, X.Y., Liu, Z., & Fu, Y. (2010). MapReduce as a programming model for association rules algorithm on Hadoop. *In the 3rd International Conference on Information Sciences and Interaction Sciences*, 99-102.

Conejero, J., Caminero, B., & Carrión, C. (2014). Analysing Hadoop performance in a multi-user IaaS Cloud. *In International Conference on High Performance Computing & Simulation (HPCS)*, 399-406.

Hendricks, J., Sambasivan, R., Sinnamohideenand, S., & Ganger, G. (2006). *Improving small file performance in object-based storage, Technical report, Tech. Report CMU-PDL-06-104*.

Dong, B., Qiu, J., Zheng, Q., Zhong, X., Li, J., & Li, Y. (2010). A novel approach to improving the efficiency of storing and accessing small files on hadoop: a case study by powerpoint files. *In IEEE International Conference on Services Computing*, 65-72.

Lin, W., & Liu, J. (2013). Performance analysis of MapReduce program in heterogeneous cloud computing. *Journal of networks*, *8*(8), 1734-1741.

Steinmetz, D., Perrault, B.W., Nordeen, R., Wilson, J., & Wang, X. (2012). Cloud computing performance benchmarking and virtual machine launch time. *In Proceedings of the 13th annual conference on Information technology education*, 89-90.

Chai, S., Yang, J., & Cheng, Y. (2007). The research of improved apriori algorithm for mining association rules. *In International Conference on Service Systems and Service Management*, 1-4.

Rui, C., & Zhiyi, L. (2011). An improved Apriori algorithm. *IEEE International Conference on Electronics and Optoelectronics (ICEOE)*, V1-476 - V1-478.

Marozzo, F., Talia, D., & Trunfio, P. (2010). A peer-to-peer framework for supporting mapreduce applications in dynamic cloud environments. In *Cloud Computing*, 113-125. http://doi.org/10.1007/978-1-84996-241-4_7.

Sathya, S., & Jose, M.V. (2011). Application of Hadoop MapReduce technique to Virtual Database system design. *In IEEE International Conference on Emerging Trends in Electrical and Computer Technology*, 892-896.

Cryans, J., Ratte, S., & Champagne, R. (2010). Adaptation of A priori to MapReduce to Build a Warehouse of Relations between Named Entities across the Web. *Proceeding of 2nd IEEE International Conference on Advances in Databases, Knowledge, and Data Applications*, 185-189.

Li, L., & Zhang, M. (2011). The Strategy of Mining Association Rule Based on Cloud Computing. *Proceedings of IEEE International Conference on Business Computing and Global Normalization*, 475-478.

Zhenhua, W., Haopeng, C., Ying, F., Delin, L., & Yunmeng, B. (2015). Workload balancing and adaptive resource management for the swift storage system on cloud. *Future Generation Computer Systems*, *51*, 120-131.

Raul, G.T., Marc S.A., Aleix, R., Adrian, M.M., Xavier, L., & Pedro, G.L. (2014). Giving form to social cloud storage through experimentation: Issues and insights. *Future Generation Computer Systems*, *40*, 1-16.

Jayesh, P. (2019). An Effective and Scalable Data Modeling for Enterprise Big Data Platform. *IEEE International Conference on Big Data (Big Data)*, 2691-2697.

Zhu, Y., Chevalier, K., Wang, N., Wang, X., Palacharla, P., & Ikeuchi, T. (2020). Efficient Mobile Edge Computing with Different Memory Capacities for Mobile Internet of Things. *In International Conference on Computing, Networking and Communications (ICNC)*, 826-832.

Ibrahim, A.I., Wei, D., & Mustafa, B. (2016). Intelligent Data Placement Mechanism for Replicas Distribution in Cloud Storage Systems. *IEEE International Conference on Smart Cloud (SmartCloud)*, 134-139.

Dautov, R., & Distefano, S. (2020). Stream Processing on Clustered Edge Devices. *IEEE Transactions on Cloud Computing*, 1-1. http://doi.org/10.1109/TCC.2020.2983402

Mazumdar, S., Seybold, D., Kritikos, K., & Verginadis, Y. (2019). A survey on data storage and placement methodologies for cloud-big data ecosystem. *Journal of Big Data*, *6*(1), 15.

Ullah, S., Awan, M. D., & Sikander Hayat Khiyal, M. (2018). Big Data in cloud computing: a resource management perspective. *Scientific Programming*. https://doi.org/10.1155/2018/5418679.