HadoopSec: Sensitivity-aware Secure Data Placement Strategy for Big Data/Hadoop Platform using Prescriptive Analytics

Revathy P[1] and Rajeswari Mukesh[2]

[1] OCBC Bank Limited, Singapore
[2] Hindustan Uniersity, Chennai, India
[1] revamadhankr@gmail.com, [2] rajeswarim@hindustanuniv.ac.in

Abstract—**Hadoop has become one of the key player in offering data analytics and data processing support for any organization that handles different shades of data management. Considering the current security offerings of Hadoop, companies are concerned of building a single large cluster and onboarding multiple projects on to the same common Hadoop cluster. Security vulnerability and privacy invasion due to malicious attackers or inner users are the main argument points in any Hadoop implementation. In particular, various types of security vulnerability occur due to the mode of data placement in Hadoop Cluster. When sensitive information is accessed by an unauthorized user or misused by an authorized person, they can compromise privacy. In this paper, we intend to address the approach of data placement across distributed DataNodes in a secure way by considering the sensitivity and security of the underlying data. Our data placement strategy aims to adaptively distribute the data across the cluster using advanced machine learning techniques to realize a more secured data/infrastructure. The data placement strategy discussed in this paper is highly extensible and scalable to suit different sort of sensitivity/security requirements.**

Keywords-Big Data; Hadoop; Security Measures; Data Block Placement; Sensitive Data placement; Multi-tenancy in Hadoop

I. INTRODUCTION

A distributed system is a network of self-governing compute nodes connected by rapid networks that appear as a single workstation. In real world, solving complex problems means division of problem into sub tasks and each of which is solved by one or more compute nodes which communicate with each other. The current disposition towards Big Data analytics has led to many intensive compute tasks. Big Data, is termed for a collection of data sets which are large and complex and difficult to process using traditional data processing tools. The need for Big Data management is to ensure high levels of data accessibility for business intelligence and big data analytics. This condition needs applications capable of distributed processing involving terabytes of information saved in a variety of file formats. Hadoop is a well-known and a successful open source implementation of the MapReduce programming model in the realm of distributed processing. The Hadoop runtime system coupled with HDFS provides parallelism and concurrency to achieve system reliability. The major categories of machine roles in a Hadoop deployment are Client machines, Master nodes and Slave nodes. The Master nodes supervise storing of data and running parallel computations on all that data using Map Reduce. The NameNode supervises and coordinates the data storage function in HDFS, while the JobTracker supervises and coordinates the parallel processing of data using Map Reduce. Slave Nodes are the vast majority of machines and do all the cloudy work of storing the data and running the computations. Each slave runs a DataNode and a TaskTracker daemon that communicates with and receives instructions from their master nodes. The TaskTracker daemon is a slave to the JobTracker likewise the DataNode daemon to the NameNode. HDFS file system is designed for storing huge files with streaming data access patterns, running on clusters of commodity hardware. An HDFS cluster has two types of node operating in a master-slave pattern: A NameNode (Master) managing the file system namespace, file System tree and the metadata for all the files and directories in the tree and some number of DataNode (Workers) managing the data blocks of files. The HDFS is so large that replicas of files are constantly created to meet performance and availability requirements. A replica is usually created so as the new storage location offers better performance and availability for accesses to or from a particular location. In the Hadoop architecture, the replica is commonly selected based on storage and network feasibility which makes it fault tolerant so as to recover from failing Data Node. The rest of this paper is organized as follows. Section II, discusses related current default block placement algorithm in Hadoop; Section III discusses the related work on placement algorithms; Section IV discusses the proposed Sensitivity-aware Secure Data Placement Strategy for Hadoop (HadoopSec) – based on an advanced machine learning placement algorithm that caters to sensitivity of data and prescribes the best suitable placement mechanism and also relates the output of the algorithm with historical information to truly make the system prescriptive in its nature. Section V concludes and discusses the future scope of this work.

II. HDFS DEFAULT BLOCK PLACEMENT STRATEGY

In HDFS, a file is divided into small chunks of size (default 64MB) defined by the parameter dfs.block.size in the config file named hdfs-site.xml. These data chunks will be placed on a

DOI: 10.5176/2251-3043_6.1.109
ISSN: 2251 - 3043 ; Volume 6, Issue 1; 2018 pp. 116 - 121

different DataNodes. The number of these datanodes is configured through the parameter dfs.replication in file hdfs-site.xml which helps to achieve fault tolerance. Each copy is called as a replica. HDFS uses rack aware data placement strategy that means if the blocks are placed in one rack then their copy will be placed in another rack so as to achieve fault tolerance when there is a node failure or any switch failure.

Following is the default block placement policy present in HDFS [8]:

- Place the first replica on DataNode [either local or random node depending upon the HDFS client running in the cluster].
- Place the second replica on a rack other than first replica placement.
- Place the third replica on a different data node in the same rack where the second replica is placed.
- If there are replicas remaining; distribute them randomly across the racks present in network with the restriction that, in the same rack there are no more than two replicas

III. Related Work

Ashwin Kumar et al [4] proposed Sensitive Data Detection [SDD] Framework which aims at identifying sensitive information in a dataset in Hadoop. This paper automates finding of sensitive information based on the dataset itself and the related datasets when the data owner doesn't provide any information on how the dataset should be used. Data Similarity Analyzer (DSA) implemented using Markov's algorithm estimates the similarity between datasets by combining the context similarity and usage pattern similarity. The advantage is that sensitive data can be handled as per the requirements. The limitation is that the author does not take into consideration about data replication and the turn-around time because this will add an overhead in the execution time.

JiongXie et al. [5], describes the need of Hadoop for data-intensive nature systems in heterogeneous clusters such as data mining systems or web indexing. Data locality plays a key role in improving MapReduce performance. In a heterogeneous cluster, high computing power nodes compete with that of low computing power node so there is often data movement from low computing power node to high computing power node thereby reducing the performance and leads to issue of load balancing. This paper proposes data reorganization algorithm to support data placement in the cluster as a means to tackle this issue. The ratio of computing power across nodes called "computing ratio". Depending upon computing ratio, fragments of a file are distributed so that all nodes can complete the processing of local data around the same time. The advantage of this work is better utilization of computing power. The limitation is that the author does not take into consideration about data replication because of the higher disk space utilization.

Su-Hyun Kim et al. [6] discusses the disadvantages of the block access token method used in Hadoop to control the permission of the data blocks. This paper proposes a secret-sharing-based block access token for authentication which uses the secret value shared by NameNode, DataNodes, and clients. The block access token is encrypted with the secret key. As a result, the block access token is more secure, and MITM by attackers is prevented. The authors have used the Shamir secret-sharing method to share the token between NameNode, DataNodes and client. The limitation of this work is using extra measure of adding secrecy for the basic action which will create overhead in the cluster operation. The advantage is the additional security in accessing the data blocks.

Jeremy Stribling et al. [7] proposes a new wide-area file system WheelFS, which allows application control over the sharing/independence tradeoff, including consistency, failure handling, and replica placement. WheelFS provides a location-independent hierarchy of directories and files with a POSIX file system interface. At any given time, every file or directory object has a single "primary" WheelFS storage server that is responsible for maintaining the latest contents of that object. When a WheelFS client needs to use an object, it must first determine which server is currently the primary for that object. The limitation is that Hadoop has the SPOF (NameNode) which is the master for all the data blocks.

Xianglong Ye et al. [8], proposes advanced block placement strategy which considers the space available in the Datanode. The Proposed strategy mainly considers the load balancing. HDFS mainly considers the network bandwidth but doesn't consider the current disk space utilization while placing blocks. Also, it doesn't consider the real-time situation of node so there is need of extra balancing tool called balancer to achieve the load balancing. Based on these limitations this author proposes new block placement policy which takes care about disk space utilization. Load balancing is achieved through taking lowest utilization node as a priority node. The advantage of paper discussed above is, there is proper load balancing based on real time situation of Datanode and prior known disk space utilization before placing block. Hence, no load balancer is needed. The limitation is that there is control overhead if large numbers of datanodes are present in Hadoop cluster.

Madhu Kumar et al. [13] proposed "A Dynamic Data Placement Scheme for Hadoop Using Real-time Access Patterns" which analyzes the real-time access patterns of data that is consumed by users. Data is placed near to users so that access time is reduced and bandwidth utilization is proper. In real situation, ideal nodes are chosen based on who is accessing the data node frequently, and choosing most relevant location. The distance is found using ping's RTT (round trip time). The benefit of this work is choosing Datanode based upon real time condition. This paper proposes lightweight extension for Hadoop called CoHadoop [14]. Co-location is achieved by adding a configurable property called locator, locator table is maintained at master-node and data placement policy is modified so that it uses locator while placing blocks. A file which does not have any locator is placed with default block placement strategy.

IV. Sensitivity-aware Secure Data Placement Strategy for Hadoop (HadoopSec)

A. HadoopSec Architecture

Fig. 1 shows the design of Sensitivity Aware Data Placement Strategy for Hadoop. As per existing Hadoop, the input file will be split into block chunks of either 64mb or 128mb size. The process flow in the design is as below.

- *Client splits the input file into chunks of blocks of either 64MB or 128MB based on configuration.*
- *While sending the request, client provides the affinity levels of data optionally,*
- *For each block, the client requests NameNode for the set of datanodes into which the block chunk needs to be written.*
- *NameNode first consults the Rack Awareness Script to get the current datanode layout snapshot.*
- *Then it internally consults the prescriptive analytics program, which takes multiple inputs and suggests the datanodes that can hold that block chunk.*
- *The datanodes list is then sent to client by the namenode*
- *Client then writes the data to the datanodes sequentially.*
- *This process is followed for the other blocks as well.*

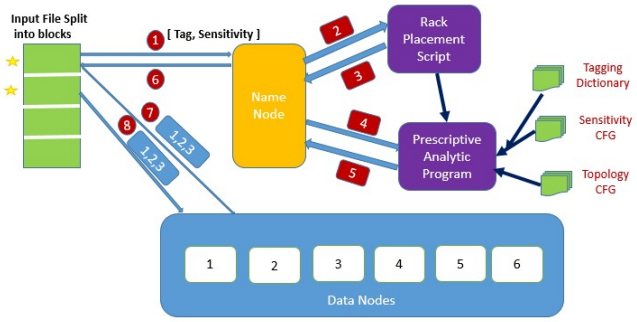

Figure 1. HadoopSec Architecture.

The Prescriptive Analytic Program Block is the key player in the overall design. It is used to classify the datanodes based on the sensitivity and quite a few other aspects. It uses unsupervised machine learning algorithm to predict the datanode which can host the data block. Unsupervised Algorithm takes the following configurations as input.

1) *Tagging Dictionary*

This configuration file will contain the client defined affinity levels between different groups whose data lies inside Hadoop.

2) *Sensitivity CFG*

This configuration file will contain the sensitivity levels of the file information that is to be placed inside Hadoop.

3) *Topology CFG (The presudo code can come after this section)*

This is the system configuration file of the cluster topology design.

In addition to these, the output of the rack awareness script is also used as an input by the algorithm. Using these, it deduces the datanodes affinity levels for the block which needs to be written into HDFS. The datanodes are formed into different clusters and the best datanodes are selected as the preferred location for the data block to be written.

B. Proposed HadoopSec Algorithmic Soultion

HadoopSec uses the below algorithm which takes the Tagging dictionary, Sensitivity CFG, Topology CFG and Rack-Awareness output as inputs and provides the output of best suitable data nodes for block placement.

HadoopSec Algorithm

Input: Tagging dictionary, Sensitivity CFG, Topology CFG and Rack-Awareness output (Matrix of feature vectors F – individual vectors are denoted by fi)
Parameters: Clustering similarity threshold γ
Output: Block-placement Output (a set of 'k' clusters)

Step1: Load the input files
Step2: Cleanse the data
Step2a: Impute the missing values (if any)
k-Means clustering (as an example) cannot deal with missing values. Any observation even with one missing dimension must be specially handled. If there are only few observations with missing values then these observations can be excluded from clustering. However, this must have equivalent rule during scoring about how to deal with missing values. Since in practice one cannot just refuse to exclude missing observations from segmentation, often better practice is to impute missing observations.
Step2b: Handle Categorical variables
If it's an ordinal variable (large, medium or small etc.,) it can be replaced by 5/10 etc.,). If it's a cardinal variable (like age-group, income group) they must be translated to binary format)
Step2c: Format the variables (date, numeric and string)
Step2d: Random initialization – k-Means clustering is prone to initial seeding i.e. random initialization of centroids which is required to kick-off iterative clustering process. Bad initialization may end up getting bad clusters.
Step3: Initialize by choosing Fsub , a subset of vectors from F
while un-clustered members of Fsub remain
Compute pairwise co-relations between all vectors in Fsub
Find the vector with the largest number of "close" neighbors,
with "close" defined as correlation > γ

Add this vector to the list of cluster centers, and remove it
and all its close neighbors from Fsub
end while

while un-clustered members of F remain
Choose a new Fsub, a subset of vectors from F
Compare Fsub, vectors to all previously found clusters:
- Compute correlation with all cluster centers
- Identify any vectors in Fsub that are "close" to previously found centers (correlation > γ)
- Remove these vectors from Fsub
while un-clustered members of Fsub remain
Continue clustering logic from:
//clustering step from Fsub
end while
end while

Step4: Allocate the blocks based on the output of clustering to ensure security
Step5: Repeat Step1-Step4 for future data and store the output
Step6: Compare the cluster information with the various inputs to come up with association for prescriptive analysis (i.e. compare associating clusters with the historical information classifications)

C. Desciption of the algorithm

The sensitivity and topology CFG files form the basis of the input files. The features from these 2 files are merged with the tagging information (whether the data is critical or not etc.,). Once the master input file is prepared, standard data transformation procedures are carried out (like missing data treatment, formatting of variables, creation of dummy variables etc.,). The information classification variable (example) is chosen as the dependent variable and the other features would be independent variables based on which the k-means clustering would be carried out. To kick off the clustering process, initial seeding is provided to set up the initialization of centroids. The clustering algorithm would eventually compute pairwise correlations between all the "Fsub" (subset of the entire feature list) number of independent variables or features to identify the vector with the largest number of "close" (defined by setting the threshold parameter) neighbors. This vector would now be added to the list of cluster centroids and then remove the same vector from its closest neighbors from Fsub. This logic is continued iteratively till the pre-specified set of "k" clusters is reached. The information blocks are allocated based on the output of clustering process to ensure security. The outputs of the successive clustering runs are stored. These historical associations between clusters (across multiple runs) would provide input to make the clustering engine intelligent and prescriptive in its nature.

V. Conclusion

As Hadoop turns out to be most sought out framework for data analytics and processing, it also brings apprehensions about access control, security and privacy along with it. Present security mechanisms in Hadoop are not adequate enough to protect the sensitive information from misuse. In addition, as Hadoop currently supports only basic data placement strategy, the issue of accessing sensitive information by misusers is highly possible. In this paper, the proposed Sensitivity Aware Data Placement Strategy for Hadoop (HadoopSec) framework brings down the risk level of placing sensitive data items in Hadoop. The sensitive information configuration is solely based on the information provided by data owners or by based on intuition. The proposed framework will work for any type of sensitivity data requirement. Our experimental results show that there is an overhead posed by the proposed framework to the existing Hadoop implementation. But this is a trade-off for protecting sensitive information in Hadoop.

References

[1] Madhan Kumar Srinivasan, K. Sarukesi, Paul Rodrigues, M. Saimanoj, P. Revathy, "State-of-the-art Cloud Computing Security Taxonomies – A classification of security challenges in the present cloud computing environment," ACM, Aug. 2012, pp. 470-476, DOI: 10.1145/2345396.2345474.

[2] Madhan Kumar Srinivasan, K. Sarukesi, K. Ashima, P. Revathy, "eCloudIDS – Design Roadmap for the Architecture of Next-generation Hybrid Two-tier Expert Engine-based IDS for Cloud Computing Environment," Springer CCIS, Springer Verlag-Heidelberg, USA, Sep. 2012, pp. 358-371, Service Vol. 335, DOI: 10.1007/978-3-642-34135-9_36.

[3] Madhan Kumar Srinivasan, K. Sarukesi, K. Ashima, P. Revathy, "eCloudIDS Tier-1 uX-Engine Subsystem Design and Implementation using Self-Organizing Map (SOM) for Secure Cloud Computing Environment," Springer CCIS, Springer Verlag-Heidelberg, USA, Sep. 2012, pp. 432-443, Service Vol. 335, DOI: 10.1007/978-3-642-34135-9_42.

[4] Ashwin Kumar TK, Hong Liu, Johnson P Thomas, Goutam Mylavarapu, "Identifying Sensitive Data Items within Hadoop" 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), and 2015 IEEE 12th International Conf on Embedded Software and Systems (ICESS).

[5] JiongXie; Shu Yin; Xiaojun Ruan; Zhiyang Ding; Yun Tian; Majors, J.; Manzanares, A.; Xiao Qin, "Improving MapReduce performance through data placement in heterogeneous Hadoop clusters," 2010 IEEE International Symposium on , vol., no., pp.1,9, 19-23 April 2010.

[6] Su-Hyun Kim, Im-Yeong Lee, "Data Block Management Scheme Based on Secret Sharing for HDFS" 10th International Conference on Broadband and Wireless Computing, Communication and Applications, 2015

[7] Jeremy Stribling, Yair Sovran, Irene Zhang, Xavid Pretzer,Jinyang Li, M. Frans Kaashoek, and Robert Morris, "Flexible, Wide-Area Storage for Distributed Systems with WheelFS", NSDI '09: 6th USENIX Symposium on Networked Systems Design and Implementation

[8] Xianglong Ye; Mengxing Huang; Donghai Zhu; PengXu, "A Novel Blocks Placement Strategy for Hadoop," Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on , vol., no., pp.3,7, May 30 2012-June 1 2012

[9] 7. Wang, J.; Xiao, Q.; Yin, J.; Shang, P., "DRAW: A New Data-gRouping-AWare Data Placement Scheme for Data Intensive Applications With Interest Locality," Magnetics, IEEE Transactions on , vol.49, no.6, pp.2514,2520, June 2013

[10] Krish, K.R.; Anwar, A.; Butt, A.R., "hatS: A Heterogeneity-Aware Tiered Storage for Hadoop," Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on , vol., no., pp.502,511, 26-29 May 2014

[11] Nishanth, S.; Radhikaa, B.; Ragavendar, T.J.; Babu, C.; Prabavathy, B., "CoHadoop++: A load balanced data co-location in Hadoop Distributed

File System," Advanced Computing (ICoAC), 2013 Fifth International Conference on , vol., no., pp.100,105, 18-20 Dec. 2013

[12] Shabeera, T.P.; Madhu Kumar, S.D., "Bandwidth-aware data placement scheme for Hadoop," Intelligent Computational Systems (RAICS), 2013 IEEE Recent Advances in , vol., no., pp.64,67, 19-21 Dec. 2013

[13] Poonthottam, V.P.; Madhu Kumar, S.D., "A Dynamic Data Placement Scheme for Hadoop Using Real-time Access Patterns," Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on , vol., no., pp.225,229, 22-25 Aug. 2013

[14] M. Y. Eltabakh, Y. Tian, F. Ozcan, R. Gemulla, A. Krettek, J. McPherson. "CoHadoop: Flexible Data Placement and Its Exploitationin Hadoop," In proceedings of 37th International Conference on Very Large Data Bases, 2011, Pages 575-585, Seattle, Washington.