ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 9, September 2016

ANALYSIS OF SECURITY AND PRIVACY FEATURES IN MAPREDUCE ON CLOUDS

M.KIRAN KUMAR[1], P. KRISHNA REDDY[2]

Associate Professor, Sree Venkateswara College Of Engineering, Nellore [1]
Assistant Professor, Sree Venkateswara College Of Enginnering, Nellore [2]

ABSTRACT:

MapReduce is a programming prototype that enables for huge scalability across hundreds or thousands of servers in a Hadoop. MapReduce is extensively used daily around the world as an efficient distributed computation tool for a huge class of problems such as search, clustering, log analysis, different types of join operations, pattern matching, matrix multiplication and analysis of social networks. Privacy and security of data and MapReduce computations are significant concerns when a MapReduce computation is implements in public or hybrid clouds. In order to perform a MapReduce functions in hybrid and public clouds, authentication of mappers-reducers, privacy of data-computations, Integrity and reliability of data-computations and freshness-correctness of the outputs are mandatory. Satisfying these necessities defend the operation from a number of types of attacks on data and MapReduce computations.

Keywords:

Cloud computing, Hadoop, HDFS, hybrid cloud, public cloud, private cloud, MapReduce algorithms, distributed computing, privacy, security

1. INTRODUCTION

Cloud computing infrastructure provides on-demand, easy, and scalable access to a shared pool of configurable resources, without worrying about managing those resources. Clouds provide three types of services, as follows:

(i) Infrastructure-as-a-service, IaaS, gives infrastructure in terms of virtual machines, storage space, and networks with some conditions. (ii) Platform-as-a-service, PaaS, provides a scalable software platform allocating the implementation of custom applications (iii) Software-as-a-service, SaaS, offers software organization in clouds as a service, for instance, emails and databases.

Fig.1: Cloud Services

Clouds can be classified into three types

(i) Public cloud: a cloud that make available services to a lot of users and is not beneath the control of a single exclusive user. (ii) Private cloud: a cloud that has its proprietary resources and is beneath the control of a single exclusive user (iii) Hybrid cloud: a grouping of public and private clouds.

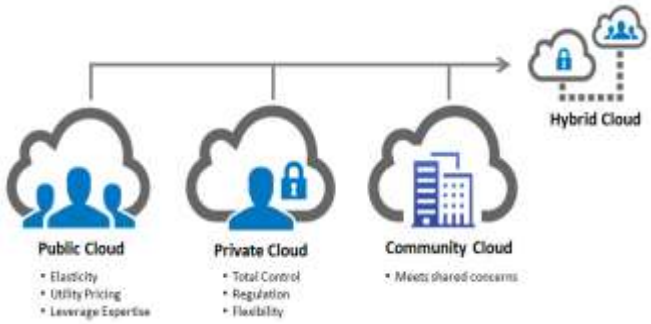

Fig.2: Cloud Types

The mainly frequent platform-as-a-service computational paradigm is MapReduce, introduced by Google in 2004. MapReduce presents a well-organized and fault tolerant parallel processing of significant data without any costly and dedicated computing node similar to a supercomputer.

MapReduce is being organized on both public and hybrid clouds, which are flat to several attacks and security threats. In the present days, a number of public clouds, e.g., Amazon Elastic MapReduce, IBM's Blue Cloud, Google App Engine, and Microsoft Azure, allow users to carry out MapReduce cloud computations without allowing for physical infrastructures and

software installations. Consequently, the deployment of MapReduce in public clouds facilitates users to process large-scale data space in a cost-effective approach and establishes a connection between two independent entities, i.e., clouds and MapReduce. As a negative aspect, the deployment of MapReduce in hybrid and public cloud desires to deal with various attacks on the communication networks and the cloud.

1.1. MapReduce Framework

MapReduce may be a process technique and a program model for distributed computing supported java. The MapReduce formula contains necessary tasks, specifically Map and scale back. Map takes a collection of information and converts it into another set of information, wherever individual components square measure lessened into tuples (key/value pairs). Secondly, scale back task, that takes the output from a map as associate degree input and combines those knowledge tuples into a smaller set of tuples. Because the sequence of the name MapReduce implies, the scale back task is usually performed once the map job. Parallel processing of large-scale knowledge provides outputs in a very timely manner. However, it constrains the computation thanks to node failure, ordering of outputs, system quantify ability, transparency, load equalization, fault tolerance, and synchronization among the nodes. A user-defined program forks a master method and employee processes at completely different nodes. The master method creates and assigns map tasks and reduces tasks to idle employee processes. An employee method deals with either a map task or a cut back task.

The Map Phase: The given computer file is processed within the map part, wherever the map perform is applied to knowledge and produces intermediate outputs (of the shape (key, value)), wherever the quantity of bits required to explain the worth in every (key, value) combine isn't essentially identical. the appliance of the map perform to one input (for example, a tuple of a on-line database or a node during a graph) is termed a plotter.

The Reduce Phase: The Reduce phase provides the ultimate output of MapReduce computations. The cut back part executes the cut back perform on intermediate outputs. The appliances of the cut back perform to one key and its associated list of values is termed a reducer.

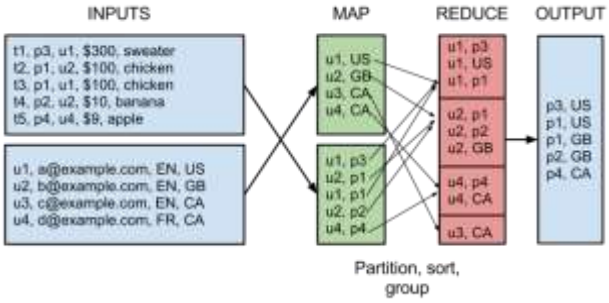

Fig 3: A general execution of a MapReduce algorithm.

Word count is a conventional example to demonstrate a MapReduce computation, where the assignment is to count the number of incidences of each word in a collection of documents. The unique input data is a collection of documents. Every mapper takes a document and implements a map task that results in a set of (key, value) pairs ({(*w*1, 1), (*w*2, 1), . . . , (*wn*, 1)}), Where each key, w_i, represents a word in the document, and each value is 1. The reduce task is executed subsequently where the reduce task adds up all the values equivalent to a key. Exclusively, a reducer for a key w_i takes all the (key, value) pairs equivalent to the key (or word) w_i and outputs a (w_i ,m_i) pair, where m is the total number of incidences of the word w_i in all the given documents.

Applications and models of MapReduce: A lot of applications in special areas are there already for MapReduce. Amongst them: matrix multiplication, detection of near-duplicates, interval join, similarity join, spatial join, pattern matching, data cube processing, graph processing, skyline queries, k-nearest-neighbors finding, star-join, theta-join, and image-audio-video-graph processing are a few applications of MapReduce in the real world.

1.2.Hadoop and HDFS

Apache Hadoop is the most known and widely used open-source software implementation of MapReduce for distributed storage space and distributed processing of large-scale information on clusters of nodes. Hadoop contains three most important components (i) Hadoop Distributed File System (HDFS): a

scalable and fault-tolerant distributed storage space system, (ii) Hadoop MapReduce (iii) Hadoop Common, the common utilities, which support the other Hadoop modules. Hadoop cluster contains a master node (that runs a JobTracker and a NameNode) and several slave nodes (where each slave node runs a TaskTracker and a DataNode); JobTracker and TaskTrackers present an environment for a MapReduce job implementation. Particularly, JobTracker accepts a MapReduce job from a user, executes the job on TaskTrackers, gets outputs from TaskTrackers, and presents outputs to the user. TaskTrackers carry out assigned jobs, provide outputs to JobTracker, and periodically throw heartbeat messages to JobTracker to show its presence and workload

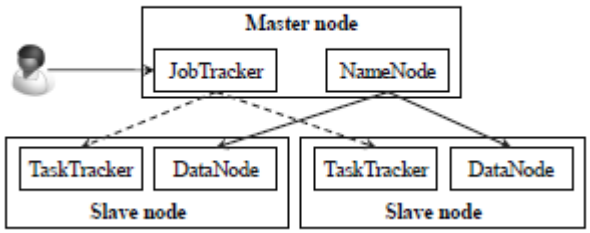

Fig 4: Structure of a Hadoop cluster with one master node and 2 slave nodes

NameNode and DataNodes present a distributed file system, named Hadoop Distributed File System, which supports write, read and delete operations on files, delete and create operations on directories. NameNode deals with the cluster metadata and DataNodes, which store up the data. NameNode keeps information of files and directories using Inodes. Inodes store several attributes of files and directories, for example modification and access times, permissions and disk space quotas. In HDFS, information is divided into small splits, called blocks, (64MB and 128MB are most frequently used sizes). Each block is separately replicated at multiple DataNodes, and block replicas are developed by mappers and reducers. Each DataNode stores the equivalent block replicas, and two files in the local host's resident file system are used to signify each block replica. The first file grasps the data itself, and the second file holds the block's metadata.

2. SECURITY AND PRIVACY CHALLENGES IN MAPREDUCE

Privacy and Security design challenges for MapReduce computations in the cloud are mainly size of input data and its storage, highly distributed nature of MapReduce computations, Data flow. MapReduce computations require a complex data flow among dissimilar storage space nodes, different computing nodes, and different clouds, The black-box nature of public clouds supporting MapReduce, Scalability, fault tolerance, and transparency, Economical issues, un trusted data access.

3. SECURITY ASPECTS IN MAPREDUCE

The security of information and computations plays a major role in MapReduce computations on each hybrid and public clouds. While not security, MapReduce computations similarly as MapReduce infrastructures are often tormented by many varieties of attacks. We have a tendency to gift security threats and security necessities for MapReduce computations. Notice that even if some security threats and security necessities area unit common for MapReduce and for generic cloud computing, we'll concentrate on security threats and security necessities within the context of MapReduce.

3.1. Security Threats in MapReduce

Distributed and replicated data processing in MapReduce open an opportunity for a wide range of attacks. While those attacks follow the same ideas as attacks in different cloud computation models, the exact application is different for MapReduce paradigm. Notice that most of these attacks are specific to MapReduce deployments in public clouds, as physical security and separation of private cloud deployments significantly reduce the risk of attacks and allow a physical separation of resources from attackers.

A) Impersonation attack.

Definition: An impersonation attack happens when an adversary successfully pretends to be a legitimate user of a system by a brute-force attack on weak passwords, pathetic encryption schemes, or other means.

MapReduce context: After a victorious impersonation attack an opponent can act on behalf of a officially permitted user and can execute MapReduce jobs that may effect in data leakage, data and computations tampering, or wrong computations on information. In addition, on public clouds under impersonation attack, an attacker might perform MapReduce computations

while the impersonated user is tariffed for data storage, the communication rate and computation time.

B) Denial-of-Service (DoS) attack.

Definition: A DoS attack happens once associate soul causes a system and therefore the network to become non-functional and non-accessible by legal users.

MapReduce context: A DoS attack happens once associate soul makes a node, mapper, or reducer to be non-functional and non-accessible by physical punishment undesirable and useless tasks. Moreover, a compromised node, mapper, or reducer could end in non-functionality of different non-compromised nodes, mappers, or reducers by repeatedly causing task requests for them to execute. Additionally, if associate aggressor compromises enough nodes during a Hadoop cluster, it should end in the failure of the full MapReduce framework and therefore the network overload.

C) Replay attack.

Definition: A replay attack occurs when an opponent resends (or replays) a captured valid message to the nodes.

MapReduce context: A replay attack occurs when an adversary assigns a number of old tasks to the nodes, building them constantly busy. In addition, an adversary can replay users qualifications to access the framework, and it may guide to impersonation and DoS attacks by spinning extreme amount of nodes.

D) Eavesdropping.

Definition: An eavesdropping attack occurs when an adversary monitors the network and the nodes without consent.

MapReduce context: An eavesdropping attack occurs when an adversary observes input data, intermediate outputs, the last outputs, and MapReduce computations without any permission from the data and computation's owner.

3.2. Security Requirements in MapReduce: Dynamic deployment and pattern of mappers and reducers for each MapReduce computation identify different security requirements in MapReduce as evaluated to the traditional parallel processing and cloud computing. Particularly, the allocation of different heterogeneous resources for dissimilar MapReduce computations, variability in the locations of resources, and different trust levels for dissimilar jobs are a only some parameters that complicate the security of MapReduce. To defeat security challenges in MapReduce computations, a secure MapReduce have got to provide authenticated and authorized access, privacy (a secure storage and computation), integrity (a fair execution and storage), and accounting-auditing of data, information and computations.

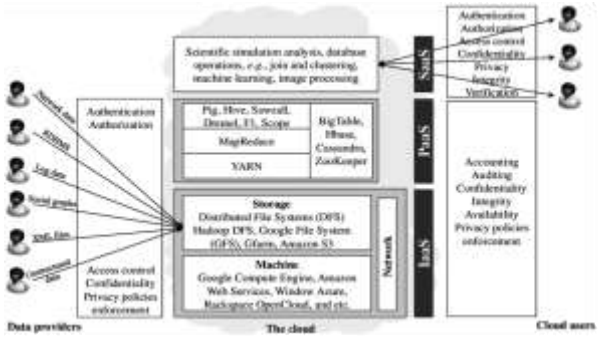

Fig 5: Security necessities in MapReduce setting on the cloud.

A) Authorization, Authentication, and access control of reducers and mappers:

Authentication presents a way to recognize an adversarial mapper, reducer, or user. Authentication is a procedure by which only those mappers and reducers that include rights to process information execute assigned tasks, and all the additional mappers and reducers who are not permitted to access information and the framework are denied. Previously mappers and reducers are authenticated; authorization of mappers and reducers permits them to access and process data by inspecting their right to use privileges to that data. Access control presents pre-configured policies that restrict an unauthorized user to access data and to right to use the framework. An adversarial mapper, reducer, or user mimics a authorized mapper, reducer, or user, while commit a breach authentication and authorization. Attacks on authentication and authorization methods are impersonation and replay attacks. The framework and information cannot be processed by any adversarial mappers, reducer, when authorization, authentication, and access control of mappers and reducers are functioning correctly.

B) Availability of data, mappers, and reducers:

Data, mappers, and reducers are supposed accessible to authenticated and authorized users without interruption. An adversarial code may build mappers-reducers and the network too active so that they cannot development data and available to

convey data, respectively. An attack on availability of data, mappers, and reducers be able to interrupt MapReduce computations. Particularly, an attack such as Denial-of-Service is an attack on accessibility of data, mappers, and reducers. For example, a single adversarial user in multi-users cloud-based MapReduce setting can considerably collision job completion times of the entire cluster even when using not as much of than 10% of cluster possessions.

C) Confidentiality of computations and data: Confidentiality of computations and information refers to the protection of computations and information, which can be in-transit from the user's location to the placement of computations or could also be hold on at public clouds, from unauthorized users and also the public cloud itself. An attack on confidentiality of computations and information is interception (man-in-the-middle attacks and eavesdropping). By making certain confidentiality, one cannot intercept computations and information throughout the transmission and following transmission on public clouds themselves. Verification of outputs is a not easy task in MapReduce due to a huge parallel processing and a huge amount of input/output data. Verification makes certain completeness, correctness, and freshness of outputs

4. ADVERSARIAL MODELS FOR MAPREDUCE SECURITY

There are numerous possible adversarial models in security, and thus, we will concentrate only on a limited set of adversarial models involved in MapReduce security.

A) Honest-but-Curious adversary:

An honest-but-curious human executes a MapReduce job properly and doesn't interfere with the duty furthermore as data; but, it performs some further computations for understanding the total information and also the job. For instance, a public cloud doesn't interfere a MapReduce job and information; however it will observe the duty and information.

B) Malicious adversary:

A malicious somebody will execute any computation for stealing, corrupting, and modifying knowledge still because the original MapReduce computation. for instance, a malicious mappers or reducer might not perform a computation or could offer wrong outputs Malicious adversaries in distributed settings, like MapReduce, area unit divided into 2 varieties, as follows: (i) non-collusive malicious somebody: a non-collusive malicious adversary works severally, and hence, provides wrong outputs while not consulting different malicious adversaries. during this case, if a homogenous task is allotted to 2 nodes and a minimum of one among them is non-collusive, then the malicious behavior of the node will be simply detected by scrutiny their outputs; (ii) conniving malicious somebody: a conniving malicious adversary communicates with all the opposite malicious adversaries before providing outputs. During this case, once a conniving somebody is allotted a task, it consults different conniving adversaries to seek out if they're allotted a homogenous task. If yes, then all the conniving adversaries offer a homogenous wrong output that creates it more durable to notice them.

5. PROPOSED SOLUTIONS FOR SECURING MAPREDUCE

5.1. Authentication, authorization, and access control based approaches:

An validation mechanism for Hadoop using Kerberos and three unusual types of tokens, namely delegation token, job token, block access token is presented. The communication between a user and HDFS is divided into two parts: (i) a user accesses NameNode using Hadoop's distant procedure call (RPC) libraries, and all RPCs attach using Simple Authentication and Security Layer that utilizes Kerberos, DIGEST-MD5, or a delegation token and (ii) a user accesses DataNodes by means of a streaming socket connection that is secured using a chunk access token. The working of the three types of tokens is as follows:

Delegation token: A delegation token is a secret key among a user and NameNode. After authenticating a user, a delegation token is produced by NameNode using Kerberos, and the token is worn for subsequent authentication of the user by NameNode exclusive of involving Kerberos.

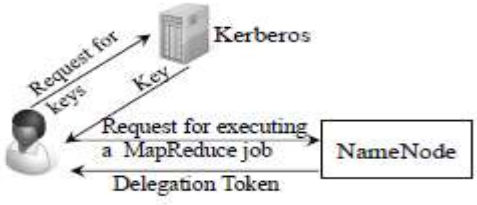

Fig 6.a: Token Delegation

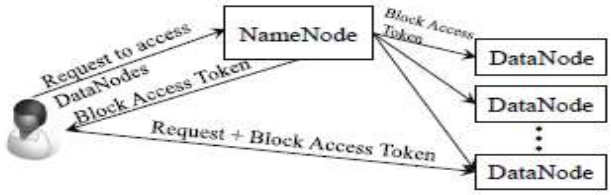

Fig 6.b: Block Access Token

Block access token: A block access token offers authentication policies to DataNodes by passing authorization data from NameNode to DataNodes and is generated by NameNode employing a symmetric-key theme wherever NameNode and every one of the DataNodes share a secret key; once a user desires to access a go in HDFS, it requests NameNode for block ids and locations of the file. In response, NameNode sends block ids and also the locations with a block access token for every block, if NameNode verifies credibleness and authority of the user. The user sends the block id with the block access token to the corresponding DataNode, and also the DataNode verifies the block access token before permitting access thereto block.

6. APPROACHES FOR SECURITY AND INTEGRITY OF STORAGE

A) iBigTable.

An development of BigTable, named iBigTable, guarantees the integrity of data using decentralized valid data structures. Two approaches are suggested for storing genuine data that is used to verify the integrity of data. Both the approaches build a Merkle Hash Tree (MHT) support authenticated data structure for the source table, the metadata tables, and the client tables. MHT permits capable and protected verification of contents of large-scale data, where non-leaf nodes are labeled with the jumble of the labels of their child nodes.

b) HDFS-RSA and HDFS-Pairing.

In order to store knowledge in a very confidential manner in HDFS, 2 advances supported hybrid cryptography. The hybrid cryptography approaches, use a block cipher and a stream cipher block ciphers and stream ciphers. Knowledge is split into fixed-sized blocks, d1, d2, . . . , dn, in keeping with a block cipher, and therefore the remaining a part of knowledge becomes block dn+1. The blocks d1, d2, . . . , dn square measure encrypted employing a random key k and a block cipher, and therefore the block dn+1 is encrypted mistreatment the key k and a stream cipher. The key, k, is, then, encrypted employing a public key scheme. the primary approach, known as HDFS-RSA, uses the RSA cryptography scheme and AES, and therefore the second approach, known as HDFS-pairing, uses a pairing-based cryptography.

7. CONCLUSIONS AND FUTURE SCOPE

Security attacks during MapReduce impersonation, denial-of-services, replay, and eavesdropping, man-in-the-middle, and repudiation attacks area unit presented. We have a tendency to believe four sorts of adversarial models, specifically honest-but-curious, hateful, knowledgeable, and network and nodes access opponents, and therefore the collision a MapReduce computation and therefore the Framework for making certain security and privacy within the scope of MapReduce. Existing algorithms and frameworks achieve success in resolution the precise security and privacy issues in MapReduce. The consequence verification is completed by replication of tasks; and privacy is ensured mistreatment information anonymization, differential privacy, and personal data retrieval. Privacy conserving analysis remains battling providing a high utilization of MapReduce framework. Extending MapReduce algorithms with privacy preserving support and these include support for secure computations between reducers and across clusters. Also, privacy policies, which define exactly what kind of aggregated or anonym zed data can be output, need to be defined.

8. REFERENCES

[1] Ren, Yulong, and Wen Tang. "A Service Integrity Assurance Framework For Cloud Computing Based On Mapreduce."Proceedings of IEEE CCIS2012. Hangzhou: 2012, pp 240 – 244, Oct. 30 2012-Nov. 1 2012

[2] N, Gonzalez, Miers C, Redigolo F, Carvalho T, Simplicio M, de Sousa G.T, and Pourzandi M. "A Quantitative Analysis of Current Security Concerns and Solutions for Cloud Computing.". Athens: 2011., pp 231 – 238, Nov. 29 2011- Dec. 1 2011

[3] Hao, Chen, and Ying Qiao. "Research of Cloud Computing based on the Hadoop platform.". Chengdu, China: 2011, pp. 181 – 184, 21-23 Oct 2011.

[4] Y, Amanatullah, Ipung H.P., Juliandri A, and Lim C. "Toward cloud computing reference architecture: Cloud service management perspective.". Jakarta: 2013, pp. 1-4, 13-14 Jun. 2013.

[5] A, Katal, Wazid M, and Goudar R.H. "Big data: Issues, challenges, tools and Good practices.". Noida: 2013, pp. 404 – 409, 8-10 Aug. 2013.
[6] Lu, Huang, Ting-tin Hu, and Hai-shan Chen. "Research on Hadoop Cloud Computing Model and its Applications.". Hangzhou, China: 2012, pp. 59 – 63, 21-24 Oct. 2012.
[7] Wie, Jiang , Ravi V.T, and Agrawal G. "A Map-Reduce System with an Alternate API for Multi-core Environments.". Melbourne, VIC: 2010, pp. 84-93, 17-20 May. 2010.
[8] K, Chitharanjan, and Kala Karun A. "A review on hadoop — HDFS infrastructure extensions.". JeJu Island: 2013, pp. 132-137, 11-12 Apr. 2013.