**Available Online at www.ijcsmc.com**

**International Journal of Computer Science and Mobile Computing**

**A Monthly Journal of Computer Science and Information Technology**

**ISSN 2320–088X**

**IMPACT FACTOR: 5.258**

*IJCSMC, Vol. 5, Issue. 6, June 2016, pg.355 – 360*

# A Security Framework for Big Data Computing through Distributed Cloud Data Centres in G-Hadoop

**Chaitanya P. Garware**
PG Student, Department of Computer Network, Flora Institute of Technology, Khopi, Pune, India
chaitanyagarware@gmail.com

**Prof. Bharat A. Tidke**
Guide, Department of Computer Network, Flora Institute of Technology, Khopi, Pune, India
batidke@gmail.com

***Abstract – Map Reduce is regarded as an adequate programming model for large scale data intensive applications. The hadoop framework is a well-known map reduce implementation that runs the map reduce tasks on cluster system .G-hadoop Is an extension of the hadoop map reduce framework with the functionality of allowing the map reduce tasks to run on multiple clusters. However, G-hadoop simply reuses the user authentication and job submission mechanism of hadoop, which is designed for a single cluster.***

***This Work proposes a new security model for G-hadoop. The security model is based on several security solutions such as public key cryptography and the SSL protocol, and is dedicatedly designed for distributed environments. This security framework provides a number of different security mechanisms to protect the G-hadoop system from traditional attacks.***

*Keyword: Big data, G-Hadoop, authentication, Security, Encryption.*

## 1. INTRODUCTION:

In today's business environment the use of information technologies to augment raditional business intelligence in order to gain a competitive advantage is an increasing focus. One of the more current applications of information technology is the use of data gathered from clients or customers in analytical activities in order to predict trends and/or monitor consumer behaviour. Collection of this big data is a big challenge, pun intended, just because of the sheer amount of data generated by users. Predictions of the amount of data that will have passed through the internet by the year 2020 is about 35 zettabytes, or 35 x 1021 bytes or 35 billion terabytes and the rate of increase will grow even higher as the years pass. Coupled with this, the emergence of the Big

Data phenomenon is attracting a large amount of attention and placing a higher imperative for Big Data analytics and storage. The International Data Corporation (IDC, 2013) now projects that the market for Big Data will increase through to $32.4 billion in the year 2017. With the increased market for Big Data, as well as the potential long term effects that Big Data will have on decision making and enterprise resource planning, this brings to mind the importance of safeguarding this data. Protecting the confidentiality, integrity, availability, and privacy of customer data and of data storage and transmission will become even more critical as the impact of a security breach can be severe.
Hadoop uses its own file system, the Hadoop Distributed File System (HDFS), to manage the input/output data of the MapReduce applications.

G-Hadoop is a MapReduce implementation targeting on a distributed system with multiple clusters, such as a Grid infrastructures , a Cloud , distributed virtual machines , or a multi-data centre infrastructure . In order to share data across multiple administrative domains, G-Hadoop replaces Hadoop's native distributed file system with the Gfarm Grid file system . MapReduce applications in G-Hadoop are scheduled across multiple clusters using a hierarchical scheduling approach. The MapReduce tasks are firstly scheduled among the clusters using Hadoop's data-aware scheduling policy and then among compute nodes using the existing cluster scheduler on the target clusters. G-Hadoop maintains the master/slave model of Hadoop, where the slave nodes are simple workers, while the master node accepts the jobs submitted by the user, splits them into smaller tasks, and finally distributes the tasks to the slave nodes. The current G-Hadoop system reuses the Hadoop mechanisms for user authentication and job submission, which is actually designed for singlecluster Environments. The mechanism applies the Secure Shell (SSH) protocol to establish a secure connection between a user and the target cluster. This kind of mechanism is not suitable for a distributed environment, like Grid, that contains several large-scale clusters. In G-Hadoop, for example, an individual SSH connection has to be built between the user and each single cluster. This approach does not handle the system as a whole; rather it works separately with the systems components. In addition, with the Hadoop security approach a user must log on to each cluster in order to be authenticated before being capable of using its resources for running MapReduce tasks. This is undoubtedly a tedious task for the users. Therefore, a more general, system-wide solution is required to hide the architecture details of the system as well as to free the user from the burden.

## G-Hadoop Architecture:

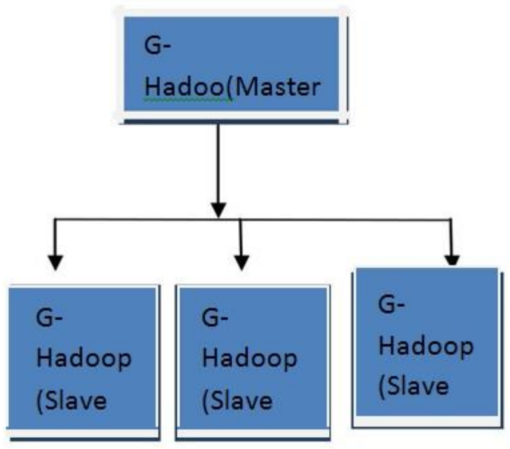

Fig shows an overview of the G-Hadoop high-level architecture and its basic components. As mentioned, the proposed architecture of G-Hadoop follows a master/slave communication model. The master node is the central entity in the G-Hadoop architecture. It is responsible for accepting jobs submitted by the user, splitting the jobs into smaller tasks, and distributing these task among its slave nodes. It is also responsible for managing the metadata of all files available in the system. The master node contains a metadata server that manages files distributed among multiple clusters, and a Job- Tracker that is responsible for splitting jobs into smaller tasks and scheduling these tasks among the participating clusters for execution. Tasks are submitted to the queue of the cluster scheduler (e.g. Torque) using a standard interface. The slave node of G-Hadoop enables the execution of MapReduce tasks on the computing nodes of the participating clusters. Its main component is a TaskTracker that is in charge of accepting and executing tasks from the JobTracker of the master node.

## 2. IMPLEMENTATION:

In this work, we designed a new security model for the G-Hadoop framework to meet the above challenges. The security framework is designed with the following properties:

1. A single-sign-on process with a user name and password: A user logs on to G-Hadoop once simply with his user name and password. In the following, all resources of the underlying system are available for the user and can be accessed by the user without additional self-participating security processes. The procedure of being authenticated by the different clusters of the underlying system is automatically performed by the security framework in the background.
2. Privacy of user information: The user information, such as authentication information, is invisible at the side of the slave nodes. Slave nodes take tasks, which are assigned by the master node, without being aware of user information, including the user name and password.
3. Access control: The security framework protects resources of the whole system from misuse or abuse of nonprofessional users. Users of the system only have the right to access the resource of a cluster that he can access with an SSH connection.
4. Scalability: A cluster can be easily integrated or removed from the execution environment without any change of the code on the slave nodes or any modification of the security framework itself.
5. Immutability: The security framework does not change the existing security mechanism of the clusters. Users of a cluster can still rely on their own authentication profiles to get access to the clusters.
6. Protection against attacks: The security framework protects the system from different common attacks and guarantees the privacy and security by exchanging sensitive information, such as information of authentication and encryption. It is also capable of detecting the fraudulent party of a fake entity to avoid abusing or illegal access to the resources by an attacker.

### G-Hadoop master node

The master node is the central entity in the G-Hadoop architecture. It is responsible for accepting jobs submitted by the user, splitting the jobs into smaller tasks and distributing these tasks Fig. 6. Software components of the G-Hadoop master node among its slave nodes. The master is also responsible for managing the metadata of all files available in the system. The G-Hadoop master node is composed of the following software components:

• **Metadata server**:

This server is an unmodified instance of the Metadata server of the Gfarm file system. The metadata server manages files that are distributed among multiple clusters. It resolves files to their actual location, manages their replication and is responsible for keeping track of opened file handles in order to coordinate access of multiple clients to files. The Gfarm metadata server is also responsible for managing users access control information.

• **JobTracker:**

This server is a modified version of Hadoop's original JobTracker. The JobTracker is responsible for splitting jobs into smaller tasks and scheduling these tasks among the participating clusters for execution. The JobTracker uses a data-aware scheduler and tries to distribute the computation among the clusters by taking the data's locality into account. The Gfarm file system is configured as the default file system for the MapReduce framework. The Gfarm Hadoop plug-in acts as glue between Hadoop's MapReduce framework and the Gfarm file system.

### G-Hadoop slave node:

A G-Hadoop slave node is installed on each participating cluster and enables it to run tasks scheduled by the JobTracker on the G-Hadoop master node.

The G-Hadoop slave node consists of the following software components:

• **TaskTracker**:

This server is an adapted version of the Hadoop TaskTracker and includes G-Hadoop related modifications. The TaskTracker is responsible for accepting and executing tasks sent by the DRMAA Gfarm Plugin.

• **JobTracker**:

Tasks are submitted to the queue of the cluster scheduler (e.g. Torque) using a standard DRMAA interface. A DRMAA Java library is used by the TaskTracker for task submission. Depending on the distributed resource manager used in the corresponding cluster, an adopted library is required. In order to access the files stored on the Gfarm file system, the Gfarm Hadoop plug-in is used.

• **I/O server**:

A Gfarm I/O server that manages the data stored on the G-Hadoop slave node. The I/O server is paired with the Metadata server on the G-Hadoop master node and is configured to store its data on the high performance file system on the cluster. In order to address performance bottlenecks, additional nodes with I/O servers can be deployed on the individual clusters.

• **Network share**:

The MapReduce applications and their configuration are localized by the TaskTracker to a shared location on the network. All compute nodes of the cluster are required to be able to access this shared location with the localized job in order to be able to perform the job's execution. In addition, the network share is used by the running map tasks on the compute nodes to store their intermediate output data. Since this data is served by the TaskTracker to a reduce task the performance of the network share is crucial and depends highly on the performance of the underlying network.

## 3. SECURITY MODEL:

As mentioned above, G-Hadoop was developed for a computing environment with large-scale distributed multiple clusters. The system components, including the CA server and the master node of G-Hadoop, may be distributed over the world and connected via a public network, for example the Internet, where various security threats exist. Hence, one component can trust another only when the authentication process is successfully conducted. However, the CA server has to be trusted by all components of the system. This is the basis condition of using certificates for authentication. In our security framework, the CA server is an important component that issues proxy credentials and slave credentials .Since the Gfarm file system, which is the underlying file system of G-Hadoop, applies the GSI security mechanism and has already a CA server, we simply reuse this server as the CA server of the proposed security framework. In addition, the slave nodes in G-Hadoop have their own GSI certificate for the Gfarm file system. This GSI certificate is also simply used as the slave credential.

**Hadoop Storage Security**

- encryption
- data-at-rest encryption
- encryption on the wire
- data in transit-protection
- isolation
- SELinux

## 4. TESTS AND PERFORMANCE EVALUATION:

This section discusses tests and performance evaluation for the G-Hadoop implementation.

4.1. Test 1: performance comparison of Hadoop and G-Hadoop :

4.1.1. Test setup

We have configured a cluster of test bed managed by Torque. Each node is equipped with 2 Dual Core AMD Opterontm Processor 270, 4 GB system memory, 1 GB ethernet and 160 GB IDE Ultra ATA133. The operating system is CentOS 5.5 64 bit (kernel 2.6.18-194.26.1.el5).

For comparison purposes, we have performed the same experiments on the following deployments of Hadoop (Scenario A) and G-Hadoop (Scenario B):

- Scenario A is a deployment of an unmodified release of Hadoop version 0.21.0, basically in its default configuration:

– 1 master node with a JobTracker and a Name Node (A.1),– 8–64 slave nodes with a Data Node and Task Tracker per slave node,
– 2 slots per Task Tracker, (map reduce.task tracker.map. tasks. maximum = 2)
– Hadoop installed and executed from a NFS share,

–No additional optimization is applied.

- Scenario B is deployed using our prototype implementation of G-Hadoop with the following configuration:
  - 1 G-Hadoop master node (B.1) with 1 JobTracker and 1 Gfarm metadata server,
  - 1 G-Hadoop slave node (B.2) with:

_ 1 TaskTracker configured to run 16– 128 tasks simultaneously,
_ 1 Gfarm DataNode,
_ 1 Torque master (pbs_server),
_ 1 Torque scheduler (default: pbs_sched),
  - 8–64 Torque compute nodes each with 1 Torque slave (pbs_ mom),
  - Hadoop installed and executed from a NFS share.

We execute the Map Reduce implementation of the Bailey– Borwein – Plouffe (BBP) [44] algorithm, which is part of the example benchmarks distributed within Apache Hadoop software. Before the job starts executing, the input splits for the configured number of tasks are calculated in order to reach a fair distribution of the workload. We executed this benchmark on scenarios (A) and (B) with cluster sizes from 16 to 64 compute nodes. The number of map tasks is set depending on the cluster size to the number of cores/slots (twice the number of nodes) and eight times the number of cores/slots on the deployed system.

## 5. CONLUSION:

The goal of this research is to advance the Map Reduce framework for large-scale distributed computing across multiple data centers with multiple clusters. The framework supports distributed data-intensive computation among multiple administrative domains using existing unmodified MapReduce applications.
In this work, we have presented the design and implementation of G-Hadoop, a Map Reduce framework based on Hadoop that aims to enable large-scale distributed computing across multiple clusters.

The architecture of G-Hadoop is based on a master/slave communication model. In order to support globally distributed data-intensive computation among multiple administrative domains, we use the traditional HDFS file system with the Gfarm file system, which can manage huge data sets across distributed clusters.
With these security mechanisms the designed security framework has the ability to prevent the most common attacks, such as MITM attack, replay attack, and delay attack, and ensures a secure communication of G-Hadoop over public networks. In addition, it adopts different mechanisms to protect the resources of G-Hadoop from abusing or misusing. On the whole, it provides a trustful and complete solution of the single-sign-on process for the user to access G-Hadoop. For further improvement, job execution in Phase V as well as the encryption algorithms and keys will be designed as changeable to increase the difficulty of cryptographic analysis by an attacker.

## REFERENCES

1. Archer, Glenn & Australia. Department of Finance and Deregulation (2013). Big data strategy issues paper. Dept. of Finance and Deregulation, [Canberra, A.C.T.] Retrieved from:http://www.finance.gov.au/files/2013/03/Big-Data-Strategy-Issues-Paper1.pdf
2. Biehn, N. (2013) The Missing V's in Big Data: Viability and Value. WIRED.com Retrieved from: http://www.wired.com/insights/2013/05/the-missing-vs-in-big-data-viability-and-value/

3. Big Data Analytics in Cyber Defense (2013) Ponemon Institute Retrieved from: http://www.ponemon.org/local/upload/file/Big_Data _Analytics_in_Cyber_Defense_V12.pdf
4. Celi, Leo Anthony, MD,M.S., M.P.H., Mark, Roger G,M.D., PhD., Stone, D. J., M.D., &Montgomery, R. A. (2013). "Big data" in the intensive care unit: Closing the data loop. American Journal of Respiratory and Critical Care Medicine, 187(11), 1157-60. Retrieved from http://search.proquest.com/docview/1446322712?accountid=13380
5. Christen, T. (2009) OSA Case Study: Applying OSA to Construct a High-Security B2C Internet Service. CTO DSwiss Ltd. Retrieved from: www.opensecurityarchitecture.org/cms/media/OSA_Case_Study_DataInherit_2009-Oct.pdf
6. Demchenko, Y., et al. (2013). "Architecture Framework and Components for the Big Data Ecosystem."
7. Globus. Overview of the Grid Security Infrastructure, Available at: http://toolkit.globus.org/toolkit/docs/3.2/g si/key/index.html, 2014.
8. Bingsheng He, Wenbin Fang, Qiong Luo, Naga K. Govindaraju, Tuyong Wang, Mars: a MapReduce framework on graphics processors, in: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, PACT'08, ACM, New York.