Volume 6, Issue 6, June 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Mandatory Factors to Implement Secure Data Processing for Big Data

[1]Reetu, [2]Dr. Dinesh Chaudhary
[1] Scholar, CSE, [2] HOD, CSE
[1, 2] Shri Ram College of Engineering & Technology, Maharshi Dayanand University, Rohtak, Haryana, India

Abstract: ***Big data is a term used to describe the collection of large and complex data sets. This large data sets are difficult to process using traditional data processing applications. In Big data, the software packages provide a rich set of tools and options where an individual could map the entire data landscape across the company, thus allowing the individual to analyze the threats he/she faces internally. This is one of the main advantage of big data keeps the data safe. With this an individual can be able to detect the potentially sensitive information that is not protected in an appropriate. For this purpose we use Apache Hadoop. Hadoop is a Programming framework used to support the processing of large data sets in a distributed computing environment. Hadoop is a software framework where an application break down into various parts. The Current Appache Hadoop ecosystem consists of the Hadoop Kernel, Mapreduce, HDFS and numbers of various components.. MapReduce is a programming framework used divide and conquer method used to break the large complex data into small units and process them . MapReduce have two stages which are : Map is the master node takes the input, divide into smaller subparts and distribute into worker nodes. A worker node further do this again that leads to the multi-level tree structure. The worker node process the m=smaller problem and passes the answer back to the master Node. Other is Reduce:- The Master node collects the answers from all the sub problems and combines them together to form the output.***

KeyWords: KV1, KV2, and KV3.

I. INTRODUCTION

Big Data

Big data is a term used to describe the collection of large and complex data sets that are difficult to process using on hand database management tools or traditional data processing applications. There are some characteristics of Big data: volume, velocity, variety, value, veracity, volatility, complexity. There are several **challenges** related to security for Big Data. These security issues can be classified into four distinct aspects of the Big Data Ecosystem.

1. Infrastructure Security
2. Data Privacy
3. Data Management
4. Integrity & Reactive Security

Big Data Security Framework are the following:

1. Data Management
2. Identity & Access Management
3. Data Protection & Privacy
4. Network Security
5. Infrastructure Security & Integrity

Big Data consists of extensive datasets, primarily in the characteristics of volume, velocity, and/or variety that require a scalable architecture for efficient storage, manipulation, and analysis.

To prevent the Big Data from different types of challenges we use Hadoop. Hadoop consists of MapReduce framework and HDFS. MapReduce is also a component of HDFS. Working of HDFS as shown in fig.

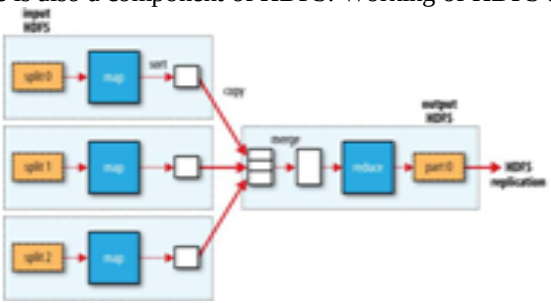

Fig. 1: Working of HDFS

"Apache Hadoop MapReduce is a framework for processing large data sets in parallel across a Hadoop cluster. Data analysis uses a two step map and reduce process. The job configuration supplies map and reduce analysis functions and the Hadoop framework provides the scheduling, distribution, and parallelization services.

The top level unit of work in MapReduce is a job. A job usually has a map and a reduce phase, though the reduce phase can be omitted. For example, consider a MapReduce job that counts the number of times each word is used across a set of documents. The map phase counts the words in each document, then the reduce phase aggregates the per-document data into word counts spanning the entire collection.

During the map phase, the input data is divided into input splits for analysis by map tasks running in parallel across the Hadoop cluster. By default, the MapReduce framework gets input data from the Hadoop Distributed File System (HDFS). Using the MarkLogic Connector for Hadoop enables the framework to get input data from a MarkLogic Server instance

The reduce phase uses results from map tasks as input to a set of parallel reduce tasks. The reduce tasks consolidate the data into final results. By default, the MapReduce framework stores results in HDFS. Using the MarkLogic Connector for Hadoop enables the framework to store results in a MarkLogic Server instance.

Although the reduce phase depends on output from the map phase, map and reduce processing is not necessarily sequential. That is, reduce tasks can begin as soon as any map task completes. It is not necessary for all map tasks to complete before any reduce task can begin. MapReduce operates on key-value pairs. Conceptually, a MapReduce job takes a set of input key-value pairs and produces a set of output key-value pairs by passing the data through map and reduce functions. The map tasks produce an intermediate set of key-value pairs that the reduce tasks uses as input. The diagram below illustrates the progression from input key-value pairs to output key-value pairs at a high level:

Though each set of key-value pairs is homogeneous, the key-value pairs in each step need not have the same type. For example, the key-value pairs in the input set (KV1) can be (string, string) pairs, with the map phase producing (string, integer) pairs as intermediate results (KV2), and the reduce phase producing (integer, string) pairs for the final results (KV3). Example: Calculating Word Occurrences.

When the user submits a MapReduce job to Hadoop:

1. The local Job Client prepares the job for submission and hands it off to the Job Tracker.
2. The Job Tracker schedules the job and distributes the map work among the Task Trackers for parallel processing.
3. Each Task Tracker spawns a Map Task. The Job Tracker receives progress information from the Task Trackers.
4. As map results become available, the Job Tracker distributes the reduce work among the Task Trackers for parallel processing.
5. Each Task Tracker spawns a Reduce Task to perform the work. The Job Tracker receives progress information from the Task Trackers.

All map tasks do not have to complete before reduce tasks begin running. Reduce tasks can begin as soon as map tasks begin completing. Thus, the map and reduce steps often overlap.

II. SETTING PROPERTIES IN A CONFIGURATION FILE

Configuration files are best suited for static configuration properties. By default, Apache Hadoop looks for configuration files in $HADOOP_CONF_DIR. You may override this location on the Hadoop command line. Job configuration files are XML files with the following layout:

```
<?xml version='1.0'?>
<?xml-stylesheet type='text/xsl' href='configuration.xsl?>
<configuration>
  <property>
    <name>the.property.name</name>
    <value>the.property.value</value>
  </property>
  <property>...</property>
</configuration>
```

Setting Properties Using the Hadoop API

You can use the Apache Hadoop API to set properties that cannot be set in a configuration file or that have dynamically calculated values. For example, you can set the InputFormat class by calling org.apache.hadoop.mapreduce.Job.setInputFormatClass.

Set properties prior to submitting the job. That is, prior to calling org.apache.mapreduce.Job.submit or org.apache.mapreduce.Job.waitForCompletion.

To set an arbitrary property programmatically, use the org.apache.hadoop.conf.Configuration API. This API includes methods for setting property values to string, boolean, numeric, and class types. Set the properties prior to starting the job. For example, to set the MarkLogic Connector for Hadoop marklogic.mapreduce.input.documentselector property, at runtime, you can do the following:


```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import com.marklogic.mapreduce.MarkLogicConstants;
...
public class myJob { ...
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs =
       new GenericOptionsParser(conf, args).getRemainingArgs();
    Job job = new Job(conf);

    // Build up the document selectory dynamically...
    String mySelector = ...;

    conf = job.getConfiguration();
    conf.set(MarkLogicConstants.DOCUMENT_SELECTOR, mySelector);
    ...
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Setting Properties on the Command Line

You can set properties on the hadoop command line using the -D command line option. For example: $ hadoop -**Dproperty_name=value** jar myJob.jar myClass

III. CONCLUSIONS

Hence there are several **challenges** related to security for Big Data as Big Data means large amount of Data. To prevent the Big Data from these types of challenges we use Hadoop. Hadoop consists of MapReduce framework and HDFS. The top level unit of work in MapReduce is a job.

A job usually has a map and a reduce phase, though the reduce phase can be omitted. During the map phase, the input data is divided into input splits for analysis by map tasks running in parallel across the Hadoop cluster. By default, the MapReduce framework gets input data from the Hadoop Distributed File System (HDFS). The reduce phase uses results from map tasks as input to a set of parallel reduce tasks. The reduce tasks consolidate the data into final results. By default, the MapReduce framework stores results in HDFS.

Although the reduce phase depends on output from the map phase, map and reduce processing is not necessarily sequential. MapReduce operates on key-value pairs. Conceptually, a MapReduce job takes a set of input key-value pairs and produces a set of output key-value pairs by passing the data through map and reduce functions. The map tasks produce an intermediate set of key-value pairs that the reduce tasks uses as input.

ACKNOWLEDGMENT

I am very much thankful to **Dr. Dinesh Kumar, HOD**, Department of Computer Science & Engineering, Shri Ram College of Engineering and Management, Palwal, for his expert advice and inspiration from time to time which helped me to concentrate more on our project work and execute it on time. I would like to take this opportunity to acknowledge the valuable help and guidance received from **Dr. S.K Garg (Director-Principal, SRCEM,** in spite of his extremely busy schedule. He not only gave me his valuable time but also provided the details regarding the project. I am thankful to the staff members of Department of Computer Engineering at Shri Ram College of Engineering and Management, Palwal for their valuable suggestions. Although it is not possible to Reetu, I cannot forget my well-wishers for their persistent support and cooperation.

REFERENCES

[1] http://www.sintef.no
[2] http://www.sciencedaily.com/releases/2013/05/130522085217.html
[3] http://www.cisco.com/web/solutions/sp/vni/vni_mobile_forecast_highlight/index.html
[4] Xiaoxue Zhang, Feng Xu, "Survey of Research on Big Data Storage", 2013 12th International Symposium on Distributed Computing and Applications to Business, Engineering & Science
[5] http://www.darkreading.com/views/dont-get-ha-duped-by-big-data-security-p/240144305
[6] Sachchidanand Singh, Nirmala Singh, "Big Data Analytics", 2012 International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, India
[7] Advantech. (2013). *Enhancing Big Data Security*. Retrieved from http://www.advantech.com.tw/nc/newsletter/whitepaper/big_data/big_data.pdf
[8] Agrawal, D., Das, S., & El Abbadi, A. (2011). Big data and cloud computing.

