# Crisis Analysis from Social Media Using Hadoop MapReduce

Setting up a Multi-Node Hadoop Cluster

This part explains the setup of Hadoop in a distributed environment using three systems (1 master & 2 slaves); given below are their IP addresses.

* Master: 192.168.1.4
* Slave1: 192.168.1.5
* Slave2: 192.168.1.6

Installing Java

Java is the main pre-requisite for Hadoop. Check whether Java is installed by using the command:

$ java -version

If everything works fine, it will tell the version and other details like this:

java version "1.8.0\_111"

Java(TM) SE Runtime Environment (build 1.8.0\_111-b13)

Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)

Otherwise follow the steps mentioned below to install it.

1. Download java (JDK) by visiting the following link: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
2. Then jdk-8u111-linux-x64.tar.gz will be downloaded into your system.
3. Copy downloaded file to home:

$ cp ~/Downloads/jdk-8u111-Linux-x64.gz ~/

1. Extract the tar file :

$ tar zxf jdk-8u111-Linux-x64.gz

1. New Java folder will be there :

$ ls

jdk1.8.0\_111

1. Setup Java Environment variables in bashrc file:

export JAVA\_HOME=/home/ubuntu/jdk1.8.0\_111

export PATH=$PATH:$JAVA\_HOME/bin

1. Verify the version and its all set:

$ java -version

Installing Hadoop

Apache Hadoop: It is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users.

This step involves setting up Hadoop in a private cloud (in Thoth Lab) and getting it running. We need to edit multiple configuration files and since Hadoop uses SSH, we need to create and setup SSH certificates. Below mentioned is the detailed process.

1. Rename the nodes as [master@192.168.1.4](mailto:master@192.168.1.4) , [slave1@192.168.1.5](mailto:slave1@192.168.1.5) & [slave2@192.168.1.6](mailto:slave2@192.168.1.6) .
2. Edit the etc/hosts file on all nodes to mention IP address of each node with hostname.

# vi /etc/hosts

192.168.1.4 master

192.168.1.5 slave1

192.168.1.6 slave2

1. Password-Less login through SSH:

* Generate keys using the following command: ssh-keygen –t rsa

$ ssh-keygen –t rsa

* Leave the file name and passphrase when prompted
* Public keys will be generated in ~/.ssh/id\_rsa.pub
* Copy keys using the command:

$ cat ~/.ssh/id\_rsa.pub >> ~/authorized\_keys

* Copy the keys to slave1 & slave2:

$ ssh-copy-id -i ~/authorized\_keys ubuntu@slave1

$ ssh-copy-id -i ~/authorized\_keys ubuntu@slave2

* Change the mode for this file :

$ chmod 0600 ~/authorized\_keys

* Now you can try to login to another node in the cluster using ssh:

$ ssh ubuntu@slave1

1. Install Hadoop on master node

* Download Hadoop-2.7.3 :

$ wget <http://apache.claz.org/hadoop/common/hadoop-2.7.3/hadoop-2.7.3-src.tar.gz>

* Create a directory for Hadoop :

$ mkdir /home/ubuntu/hadoop

* Copy downloaded file to that directory :

$ cp ~/Downloads/hadoop-2.7.3-src.tar.gz ~/hadoop/

* Extract the tar file :

$ tar –xvf ~/hadoop/hadoop-2.7.3-src.tar.gz

* Change the mode of folder:

$ chown –R ~/hadoop/hadoop

* Set up Hadoop environment variables in **bashrc file**

export HADOOP\_HOME = /home/ubuntu/hadoop/hadoop

export PATH = $PATH:$HADOOP\_HOME/bin

1. Configure Hadoop by making changes in following configuration files:

* Open core-site.xml to enter details of master node used for Hadoop instance:

$ sudo gedit ~/hadoop/hadoop/etc/hadoop/core-site.xml

<property>

<name>fs.default.name</name>

<value>hdfs://master:9000</value>

</property>

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

* Open hdfs-site.xml to enter details about replication, namenode and datanode

$ sudo gedit ~/hadoop/hadoop/etc/hadoop/hdfs-site.xml

<property>

<name>dfs.data.dir</name>

<value>/home/ubuntu/hadoop/hadoop/dfs/name/data</value>

<final>true</final>

</property>

<property>

<name>dfs.name.dir</name>

<value>/home/ubuntu/hadoop/hadoop/dfs/name</value>

<final>true</final>

</property>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

* Open **yarn-site.xml** to enter details about resource manager and node manager to configure yarn into Hadoop

$ sudo gedit ~/hadoop/hadoop/etc/hadoop/yarn-site.xml

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>master:8025</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>master:8030</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>master:8050</value>

</property>

* Open **mapred-site.xml** to enter details about job tracker and mapreduce framework

$ sudo gedit ~/hadoop/hadoop/etc/hadoop/yarn-site.xml

<property>

<name>mapred.job.tracker</name>

<value>master:9001</value>

</property>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

* Open **Hadoop-env.sh** to enter environment variables

$ sudo gedit ~/hadoop/hadoop/etc/hadoop/hadoop-env.sh

export JAVA\_HOME**=**/home/ubuntu/jdk1.8.0\_101

export HADOOP\_CONF\_DIR**=**/home/ubuntu/hadoop/hadoop/etc/hadoop

export HADOOP\_OPTS**=**"$HADOOP\_OPTS -Djava.net.preferIPv4Stack=true"

1. Transfer Hadoop files from master to all slave nodes using **scp**

$ scp -r ~/hadoop/hadoop slave1:~/hadoop

$ scp -r ~/hadoop/hadoop slave2:~/hadoop

1. Create masters and slaves file in etc/hadoop to enter hostnames

$ vi ~/hadoop/hadoop/etc/hadoop/masters

master

$ vi ~/hadoop/hadoop/etc/hadoop/slaves

slave1

slave2

1. Format namenode on master

$ hadoop namenode -format

1. Start all the services and Hadoop is all set to go

$ start-all.sh

1. Check all the services: $ jps

Tweets Collection Using Apache FLUME

Flume is a distributed and reliable service available for efficiently collecting, aggregating and moving large amounts of streaming data into Hadoop Distributed File System. In our case Flume will help us to collect the tweets and moving them to the Hadoop Distributed File System. It is a simple and flexible architecture.

FLUME was setup on our Master node, it pushed the tweets into the hdfs path we mention in its configuration files.

FLUME connects to the Steaming API of twitter and pushed tweets in real-time as they are published on Twitter.

Access Tokens for using Twitter API’s:

Create an app on **apps.twitter.com** and get the Authentication tokens. There are 4 tokens Consumer key, Consumer Secret, Access Token, Access Secret Token.

Installing FLUME:

1. Download Flume-1.6.0 : $ wget <http://apache.mirrors.hoobly.com/flume/1.6.0/apache-flume-1.6.0-bin.tar.gz>
2. Make flume directory : $ mkdir ~/flume
3. Copy flume tar to flume directory: $ cp -r ~/Downloads/apache-flume-1.6.0-bin.tar. gz ~/flume/
4. Extract Flume tar: $ tar –xvf ~/flume/apache-flume-1.6.0-bin.tar. gz
5. Open flume.env.sh.template file: $ gedit ~/flume/apache-flume-1.6.0-bin/conf/flume.env.sh.template
6. Mention the Java path: export JAVA\_HOME**=**/home/ubuntu/jdk1.8.0\_101
7. Save file as **flume.env.sh**
8. Add Flume environment variables in bashrc file:

export FLUME\_HOME = /home/ubuntu/flume/apache-flume-1.6.0-bin

export PATH = $PATH:$FLUME\_HOME/bin

1. Create **twitter.conf** file in the conf folder: $ cat > twitter.conf
2. Add details about source, channel and sink
3. Place **FLUME-sources-1.0SNAPSHOT.jar**, **twitter-4j-core-4.0.5.jar**, twitter4j-stream-4.0.5.jar, twitter4jmedia-support-4.0.5.jar in the lib folder inside the flume directory
4. Download TwitterSource.java from git (this file is used by flume)
5. Modify the TwitterSource.java file to get only the required data from tweets separated by a custom delimiter

public void onStatus(Status status) {   
        // The EventBuilder is used to build an event using the headers and   
        // the raw JSON of a tweet   
    // shouldn't log possibly sensitive customer data   
        logger.debug("tweet arrived");   
           
        headers.put("timestamp", String.valueOf(status.getCreatedAt().getTime()));   
   
    String lang = status.getLang();   
    String text = status.getText();   
    String url = status.getURLEntities().toString();   
    String lat, lon;   
    if (status.getGeoLocation()!=null)   
    {   
    lat = String.valueOf(status.getGeoLocation().getLatitude());   
    lon = String.valueOf(status.getGeoLocation().getLongitude());       
    }   
    else   
    {   
    lat = "null";   
    lon = "null";   
    }       
    boolean stat = status.isRetweet();   
    String ret;       
    if(stat)   
    ret = "true";   
    else   
    ret = "false";   
    String place = status.getPlace().getFullName();   
       
    String out = lang + "~%~" + text + "~%~" + url + "~%~" + lat +         "~%~" + lon + "~%~" + ret + "~%~" + place;    
        Event event = EventBuilder.withBody(out.getBytes(), headers);   
   
        channel.processEvent(event);   
      }

1. Download the necessary libraries and compile the java file
2. Replace the .class files in the **FLUME-sources-1.0SNAPSHOT.jar** placed inside flume lib folder with newly generated files
3. Run Flume: flume-ng agent -n TwitterAgent --conf /flume/apache-flume-1.6.0-bin/conf/ -f /home/ubuntu/flume/apache-flume-1.6.0-bin/conf/twitter.conf
4. You can track the status on “master:50070” port on your browser

Working on HDFS

**Designing Mapper and Reducer**

In this task Mapper and Reducer which are a part of Hadoop framework needs to be developed to preprocess the tweets collected. Mapper and Reducer are working on top of the text file generated by flume.

**Mapper:**

The mapper class reads data line by line from text file generated by flume.

It then extracts all the parameters language, text, url, latitude, longitude, retweet\_status and place.

It then filters the tweets as per the following conditions:

*  Language should be English
*  Retweet\_status should be false
*  Word count should be less than 10
*  No url should be there in text
*  No words like prayer, RIP etc should be present

If a tweet satisfies all these conditions, then it will consider it for analysis.

It will check what crises are mentioned in that tweet using a dictionary given and give the location and name of crisis as key value pair to output collector

**Output format: <location + crisis name,1>**

**Reducer:**

The reducer will use the output generated by mapper.

It will take key as a location and iterate over all the values(crisis) happened at that location.

It will finally give location name, crisis name and count to output collector.

**Output format: <location + crisis name, count>**

The output generated tells us for each location, how many times a particular crisis has happened there.

Steps:

1. Create a java file for MapReduce: $ cat > CrisisAnalysis.java
2. Add the required code and functions
3. Compile it using the command: $ hadoop com.sun.tools.javac.Main CrisisAnalysis.java
4. Make jar: $ jar cf ca.jar CrisisAnalysis\*.class
5. Run MapReduce using the command: $ hadoop jar ca.jar CrisisAnalysis /input /output
6. You can check the generated output in hdfs: $ hadoop fs –cat /output/part-r-00000

To run the Application: With Swing GUI

ccproject folder contains all the files required for the application

1. Create a folder: $ mkdir ~/ccproject
2. Copy all the required files, ca.jar, and make all new java files required for UI application
3. Five new java files are there: HomePage.java, NewJFrame.java, MyFusionTable.java, visualization.java and CcProject.java
4. Make all the required shell script files in this folder only
5. Compile all the files using the command: $ javac HomePage.java NewJFrame.java MyFusionTable.java visualization.java CcProject.java
6. Run the application using the command: $ java HomePage
7. Provide the location, name and select the visualization from the UI.