Project Report

On

“Descriptive Analysis of Weather Data”

CPSC 531-01 13480

Advanced Database Management

Spring, 2023

Under Guidance of

Prof. Tseng-Ching James Shen

Department of Computer Science

California State University

Fullerton, CA - 92831

May 2023

Prepared By:

Anirudh Venkatesh (885174318) anirudh.venkatesh@csu.fullerton.edu

Vedita Deshpande (885177469) vedita@csu.fullerton.edu

Revision History

Version Number	Date of release	Remarks
Version 1.0	14 May 2023	First Version

Intended Audience

This document is intended for:

1. System Engineers
2. Solution Integrators

Table of Contents

1) Introduction

Weather is an integral part, having a significant effect on our day-to-day activities. Due to the vast array of tasks and operations ranging from personal tasks to more large-scale industrial tasks that get affected by weather, weather analysis has been a highly significant and impactful field of study.

There has been intense research conducted to perform weather-related analysis and judgements that can act as guides for taking actions in the future.

Weather experts have systems that collect huge amounts of data about everyday weather, consisting of information on factors such as rain, humidity, day length, wind speed, and direction, etc. This data is collected throughout the day and fed into the data storage systems designed to store it. This generates the need for more efficient systems to store and analyze this huge amount of data over a long period of time and for a large number of areas under analysis.

The project aims to solve this problem by using the big data technologies like Hadoop, Hive and Spark SQL for weather-analysis tasks. The project proposes two methods for the analysis from the point of view of a weather expert –

Graphical Analysis and Command Line analysis are developed in a way so that the weather-expert can use both the tools to gain meaningful insights from a huge amount of weather data for a long time period for a wide range of attributes under consideration.

2) Functionalities

The design includes input from online transaction processing systems in different cities in the United States. This implementation mimics the actual data storage systems in the real world which have independent storage servers that collect data specific to a city.

The data from multiple such storage servers are gathered and injected into the Hadoop Distributed File System using the tool 'Sqoop.' Sqoop collects data from different cities and creates a secure connection to deliver the daily data collection into the standard HDFS cluster for analyzing this weather data.

The example used in the project uses data collected from 5 different cities – San Francisco, New York City, Austin, Miami, and Chicago. This model can be extended to multiple cities to be implemented in the real world further to store the big data into a Hadoop distributed file system.

After the data is stored into the Hadoop Distributed File System, Hive queries can be run on the system to get the desired output for analyzing the required weather data.

The project consists of two implementations for this –

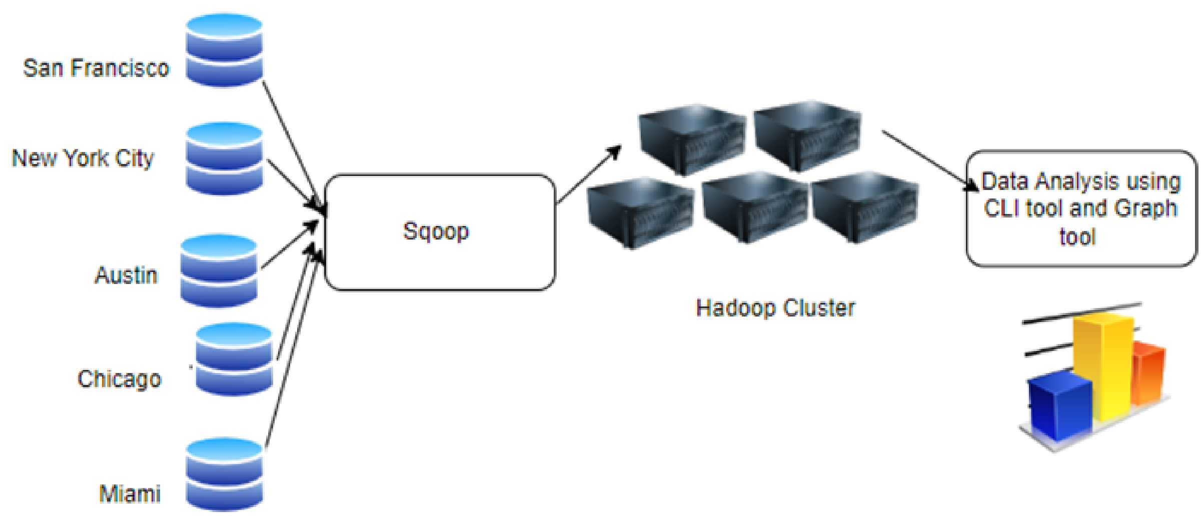
1. Hive CLI & Spark CLI

    a. The custom CLI tool developed, uses Hive queries in the background that will generate MapReduce jobs that are automatically on top of HDFS and Yarn. Thus the user can run custom queries and have more flexibility with the data analysis. The output from these queries can generate meaningful data that can be further used to gain good insight into the weather pattern and can also be used for prediction activities.

    b. The application also offers Spark CLI as a tool for running Hive queries on top of HDFS and Yarn. This is implemented using Hive Integration offered as part of the Spark framework used to run SQL queries for the data stored in Hadoop. Queries running on spark are faster than the queries running using the Hive CLI which can give the additional advantage of speed to the user to perform a quicker analysis.

2. Graphical Analysis

    The application consists of a python app that can be directly triggered by the weather expert to generate graphical reports of the required weather-related fields stored in Hadoop. This is done using MatPlot library in python which is a powerful tool to generate a variety of graphs depending on the type of analysis required by the weather-expert using the application.



3) Dataset

Reference Link - https://www.wunderground.com/history

- For testing purposes we used the data collected from - https://www.wunderground.com/history to mimic independent OLTP systems that can be used as input for the project. This design can be easily scaled to support multiple cities further.
- The database systems for the cities include several weather parameters collected throughout the day that are stored in the form of rows and column. Parameters like high temp , low temp, dew point, day length, humidity , wind etc are used for the project.
- We used independent datasets for a sample of 5 cities (New York City, Chicago, Miami, Austin and San Francisco) for the project that are used as input for Sqoop to inject the data into the Hadoop systems

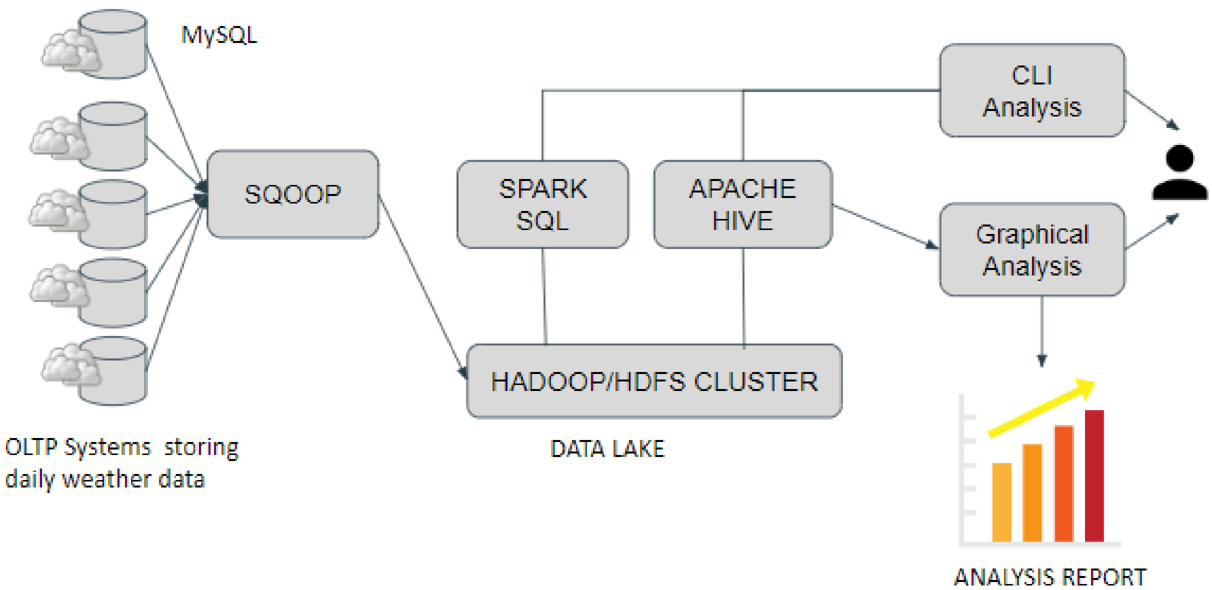
Sample Dataset used -

Date	High Temp	Low Temp	Precipitation	Dew Point	Wind	Humidity	Sea Level Pressure	Day Length
3/1/2022	48	31	0	38	18	76	30.3	680
3/2/2022	52	41	0	35	16	74	30	681
3/3/2022	47	26	0.03	36	25	73	30.4	685
3/4/2022	38	22	0	12	15	48	30.6	687
3/5/2022	45	31	0	32	14	67	30.5	688
3/6/2022	68	39	0	56	18	84	30.2	689
3/7/2022	74	51	0.05	55	32	67	29.9	700
3/8/2022	47	40	0.05	28	33	48	30.2	703
3/9/2022	40	33	0.01	32	15	92	30.2	705
3/10/2022	49	32	0.57	34	10	92	30.2	705

3) <u>Architecture & Design</u>

Tools and technologies used -

- Apache Hadoop 2.6.5
- Apache Hive 2.3.5
- Sqoop 1.4.7 for Hadoop 2.6.0
- MySQL 8.0.32
- MySQL connector JAVA 8.0.17
- OpenJDK 11.0.18
- Python 3.10
- Paramiko
- Matplotlib
- Spark 2.4.6 for Hadoop 2.6
- OpenSSH
- Oracle Virtualbox
- Scala



The system consists of input data sources as part of independent OLTP systems running using MySQL databases. Using Sqoop as an ingestion tool, these data sources are combined in a uniform manner in order to be injected into a common cluster comprising the Hadoop nodes.

Once the data is injected into the hadoop data nodes, Apache Hive and Spark SQL can be used to run SQL-like queries on top of the running Hadoop Cluster to get the weather-related data for analysis.

We initially developed our application using Apache Hive to run the SQL-like queries but later found out about the implementation use case using Spark SQL, which improved the query time significantly making it easier to get timely weather reports. Spark SQL can be easily integrated as part of an existing Hadoop System to leverage the Spark features without the need for a complete Apache Spark setup.

After getting the results from Apache Hive and Spark SQL we gather the data to present it in the form of graphs that can be used by weather experts to do further analysis on the weather data.

This feature enables non-technical people to leverage the advantages of big data technologies using the graphical analysis tool developed primarily in Python, for analyzing the weather data.

5) Minimum System Requirements

Every system that is planned to be a part of the cluster must satisfy the following hardware requirements:

- 1.5 GB RAM (2GB recommended)

- 20 GB Disk Space
- 2 CPU Cores (3 CPU Cores recommended)
- Ethernet
- Hypervisor to support virtualization

6) <u>GitHub Location of Code</u>

[Descriptive Analysis of Weather Data (GitHub)]

7) <u>Deployment Instructions</u>

The following steps are to be followed to deploy the application:

1. Setup Virtual Machines with requirements specified in Section 4. Number of virtual machines depends on the number of Datanodes required.

2. Fetch the disc Image file of ubuntu 20.04 LTS from the official Ubuntu website.

3. Load the disk image file onto the virtual machine's and run the VMs. Start the VMs and follow instructions on screen to install Ubuntu on the machine.
   Note: Basic Installation is sufficient. Complete Installation is not required.

4. Obtain the ip address of each of the VM by using the following command:
   ifconfig

5. Modify the /etc/hosts file with the ip address of each of the VMs. Format is:
   ip-address Hostname-of-that-vm

   A sample is given below:
   192.168.4.11 Namenode
   192.168.4.12 Datanode-1
   192.168.4.13 Datanode-2

6. Test if each machine is able to ping each other.
   ping <ip-address>

   If machines are able to ping each other, then the machines are able to communicate with each other.

If ping is not installed, then the same can be downloaded by using the command:
sudo apt-get install -y iputils-ping

7. Disable firewall on each of the VM
   ufw disable

8. Install openssh-server and wget on each machine. Command is:
   sudo apt-get install -y openssh-server wget

9. Create directories for Hadoop, Java, Hive, Spark. You may choose your own path but for the purposes of guidance, the instructions shall use the following paths for installation.

Hadoop	/usr/local/
Spark	/usr/local/
Hive	/usr/local/
Java	/usr/lib/

10. Download the Java SE Development kit 1.8 from the official Oracle website. Also download the Apache Hadoop, Apache Hive and Apache Spark .

11. Untar all the downloaded files onto the respective paths. First, navigate to the target directory and execute the following command to untar the files:
    sudo tar -xvf <your-download-path>/jdk-8u201-linux-x64.tar.gz

    Follow the same for Hadoop, Hive and Spark as well.

12. Create a link folder for all the folders. This will help in changing versions in the future.
    sudo ln -s /usr/lib/jdk1.8.0_201 java

    Repeat for Hadoop, Spark and Hive.

13. Change ownership of all the directories
    sudo chown -R <your-user>:<you-user> java
    sudo chown -R <your-user>:<your-user> jdk1.8.0_201

    Repeat for Hadoop, Hive and Spark.

14. Open the .bashrc file (located in /home/<user> directory) and append the following lines in it:
    export JAVA_HOME=/usr/lib/java
    export PATH=$PATH:$JAVA_HOME/bin
    export PATH=$PATH:$JAVA_HOME/sbin

    export HADOOP_HOME=/usr/lib/hadoop

export PATH=$PATH:$HADOOP_HOME/bin

export HIVE_HOME=/usr/lib/hive
export PATH=$PATH:$HIVE_HOME/bin

export SPARK_HOME=/usr/lib/spark
export PATH=$PATH:$SPARK_HOME/bin

15. Run the bashrc file so that Ubuntu is aware of the paths.
source .bashrc

16. MySQL Server Configuration
    a. Install MySQL-Server on the Namenode
    sudo apt-get install -y mysql-server

    b. Login to the MySQL-Server
    sudo mysql -u root -p

    c. Configure MySQL to run without sudo
        I. use mysql;
        II. update user set plugin='mysql_native_password' where user='root';
        III. flush privileges;

    d. Configure MySQL Password
        I. use mysql;
        II. alter user 'root'@'localhost' IDENTIFIED BY 'NewPassword';

    e. Configure Timezone
    SET GLOBAL time_zone = '+3:00';

    f. Configure remote access of MySQL Server
        I. GRANT ALL ON *.* to root@'%' IDENTIFIED BY 'root';
        II. flush privileges;

17. Import data in csv into MySQL Server
    a. Copy csv file into the secure mysql path
    sudo cp <your-csv-file>.csv /var/lib/mysql-files

    b. Login to MySQL server
    mysql -u root -p

    c. Create new database
    create database <your-database-name>;

d. Use database
Use <your-database-name>;

e. Create MySQL table
create table <table-name>(<columns and datatypes>);

f. Load data into the table
LOAD DATA INFILE '/var/lib/mysql-files/<your-file-name>.csv' INTO TABLE <table-name> FIELDS TERMINATED BY ',' IGNORE 1 LINES;

18. Download mysql-jdbc driver from [MySQL]

19. Configure Hadoop ($HADOOP_HOME/conf directory)

a. hdfs-site.xml
For Namenode:

```
<property>
<name>dfs.replication</name>
<value>3</value>
</property>


<property>
<name>dfs.namenode.name.dir</name>
<value>/adbms/name</value>
</property>

<property>
<name>dfs.datanode.data.dir</name>
<value>/adbms/data1</value>
<final>true</final>
</property>

<property>
<name>dfs.namenode.http-address</name>
<value>Namenode:50070</value>
</property>

<property>
<name>dfs.namenode.secondary.http-address</name>
<value>Datanode-1:50090</value>
</property>
```

For Datanode:

```
<property>
<name>dfs.replication</name>
<value>3</value>
</property>

<property>
<name>dfs.datanode.data.dir</name>
<value>/adbms/data2</value>
<final>true</final>
</property>

<property>
<name>dfs.namenode.http-address</name>
<value>Namenode:50070</value>
</property>

<property>
<name>dfs.namenode.secondary.http-address</name>
<value>Datanode-1:50090</value>
</property>
```

b. yarn-site.xml

```
<property>
<name>yarn.resourcemanager.address</name>
<value>Namenode:9001</value>
</property>

<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>Namenode:8031</value>
</property>

<property>
<name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>

<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
```

```
<property>
<name>yarn.nodemanager.pmem-check-enabled</name>
<value>false</value>
</property>

<property>
<name>yarn.nodemanager.vmem-check-enabled</name>
<value>false</value>
</property>
```

c. core-site.xml

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://Namenode:9000</value>
</property>
```

d. mapred-site.xml

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

e. slaves

```
Namenode
Datanode-1
Datanode-2
```

20. Configure HIVE
    a. Copy mysql-jdbc driver to $HIVE_HOME/lib
    b. hive-site.xml

<u>For Namenode:</u>

```
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.cj.jdbc.Driver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>root</value>
</property>
```

<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>root</value>
</property>

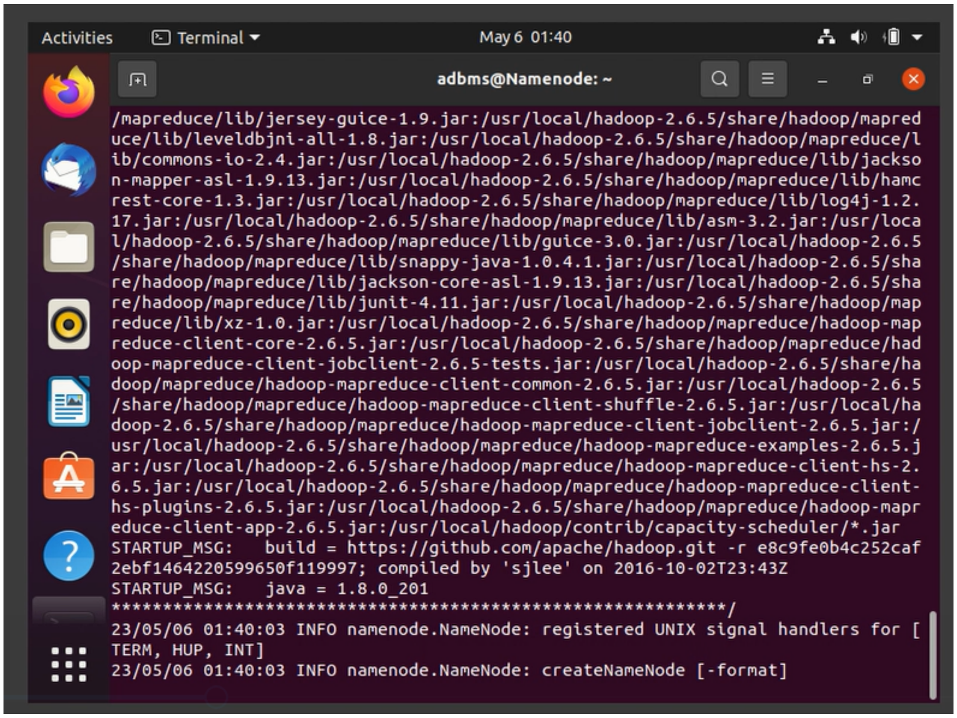
<property>
<name>hive.metastore.schema.verification</name>
<value>false</value>
</property>

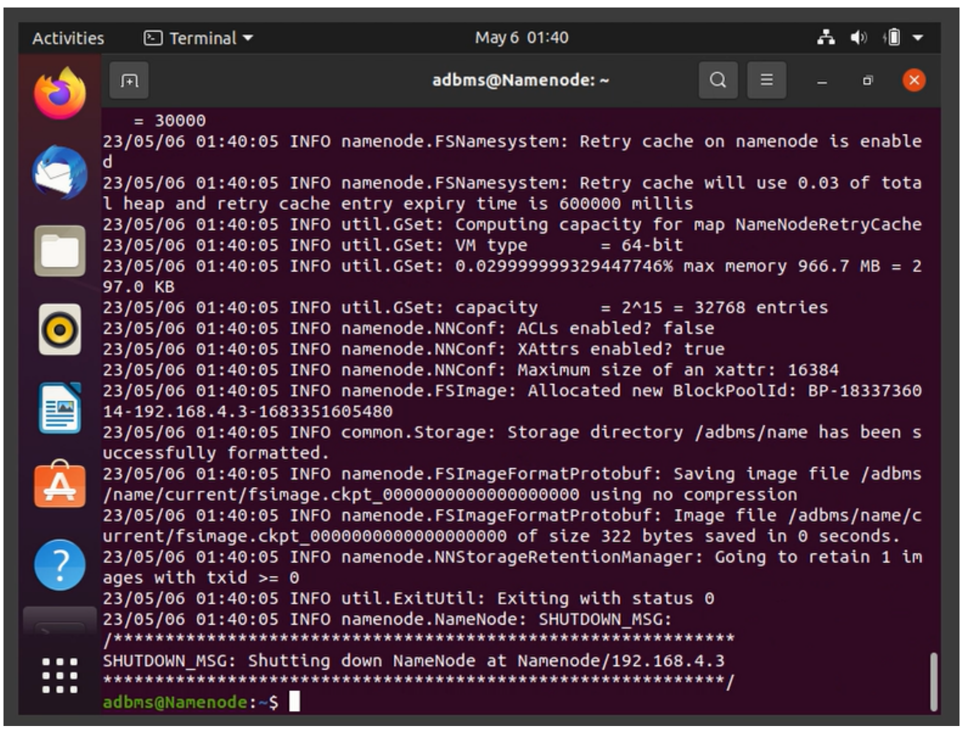
For Datanode (add all above configurations and add additional configuration below)

<property>
<name>hive.metastore.uris</name>
<value>thrift://Namenode:9083</value>
</property>

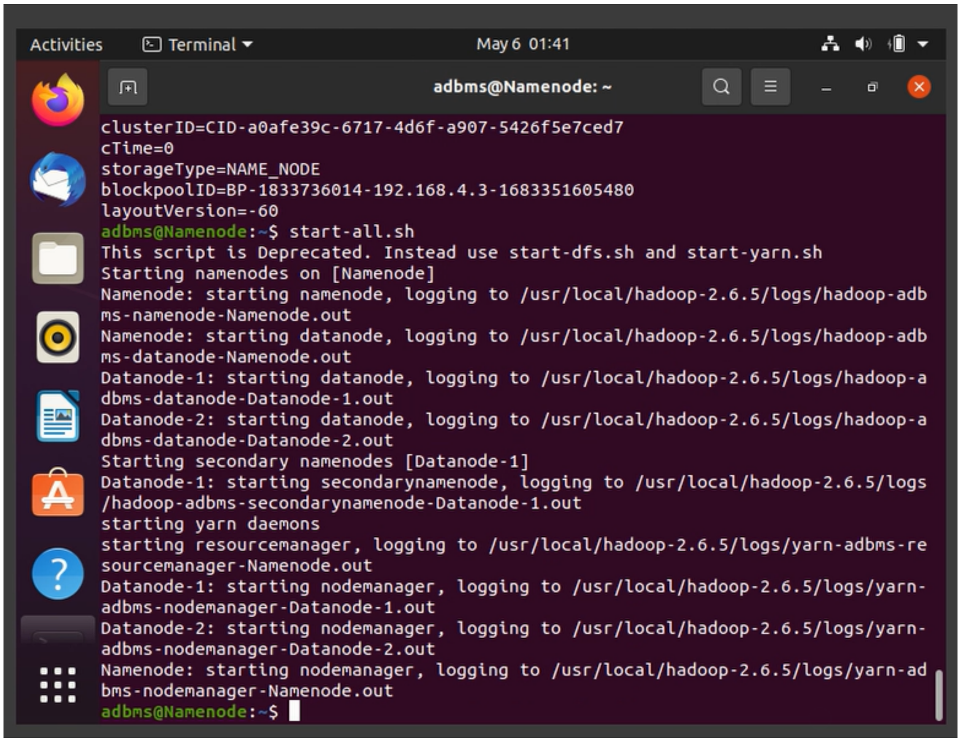
21. Configure SPARK ($SPARK_HOME)
    a. Copy mysql-jdbc driver to $SPARK_HOME/lib
    b. Copy hive-site.xml from $HIVE_HOME/conf to $SPARK_HOME/conf
    c. Slaves (inside $SPARK_HOME/conf)
       Namenode
       Datanode-1
       Datanode-2
    d. spark-env.sh
       export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
       export SPARK_HOME=/usr/local/spark
       export JAVA_HOME=/usr/lib/jdk1.8.0_201
       export SPARK_MASTER_IP=Namenode

22. Perform HDFS Namenode format
    Note: To be done only on Namenode and must be done only once.
    hdfs namenode -format

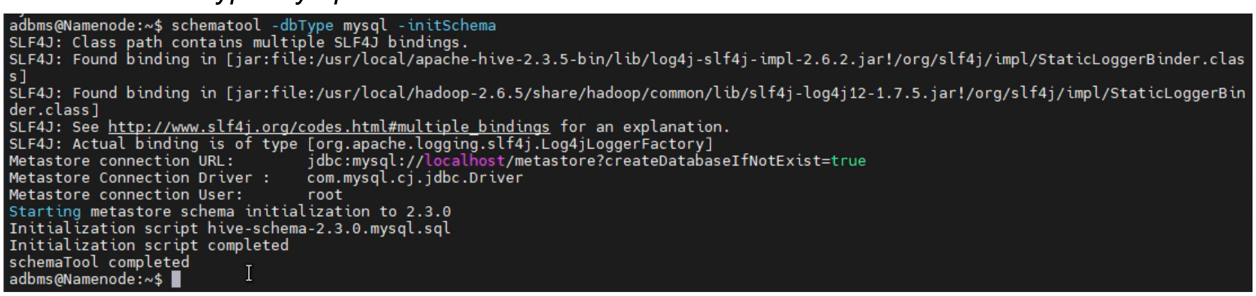




23. Start Hadoop for the first time. This will create data files as mentioned in the hdfs-site.xml configuration file.
    start-all.sh



24. Perform Hive Schematool Initialization
    schematool -dbType mysql -initSchema



25. On a new Terminal, run the Hive Metastore service
    hive --service metastore

26. Run Spark Services
    a. Navigate to the Spark conf directory ($SPARK_HOME/conf)

    b. Enter the Hadoop, Spark, Java paths and configure Spark Master IP on the spark-env.sh file

```
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export SPARK_HOME=/usr/local/spark
export JAVA_HOME=/usr/lib/jdk1.8.0_201
export SPARK_MASTER_IP=Namenode
adbms@Namenode:/usr/local/spark/conf$ S
```

c. Enter the worker nodes on the slaves file
   Namenode
   Datanode-1
   Datanode-2

d. Navigate to the Spark sbin directory
   cd $SPARK_HOME/sbin

e. Run the start-all.sh
   ./start-all.sh

27. Install Python libraries
    pip install paramiko
    pip install matplotlib

8) <u>Steps to Run the Application</u>

Once deployed, running the application is fairly simple.

1. Start Hadoop
   start-all.sh

2. Start Hive metastore service (to be run on new terminal)
   hive - -service metastore

3. Navigate to spark home directory and start spark
   cd $SPARK_HOME/sbin
   ./start-all.sh

4. Launch the Resource Manager GUI
   http://namenode:8088/cluster/apps

5. Launch Hadoop GUI
   http://namenode:50070/

6. Graphical Tool can be launched remotely by entering the machine ip address, username, password and entering the desired parameters.

7. Custom HiveUI sessions can also be created by entering ip address, username and password.

9) Steps to safely shutdown the system

The following steps are to be followed to safely shutdown the application:

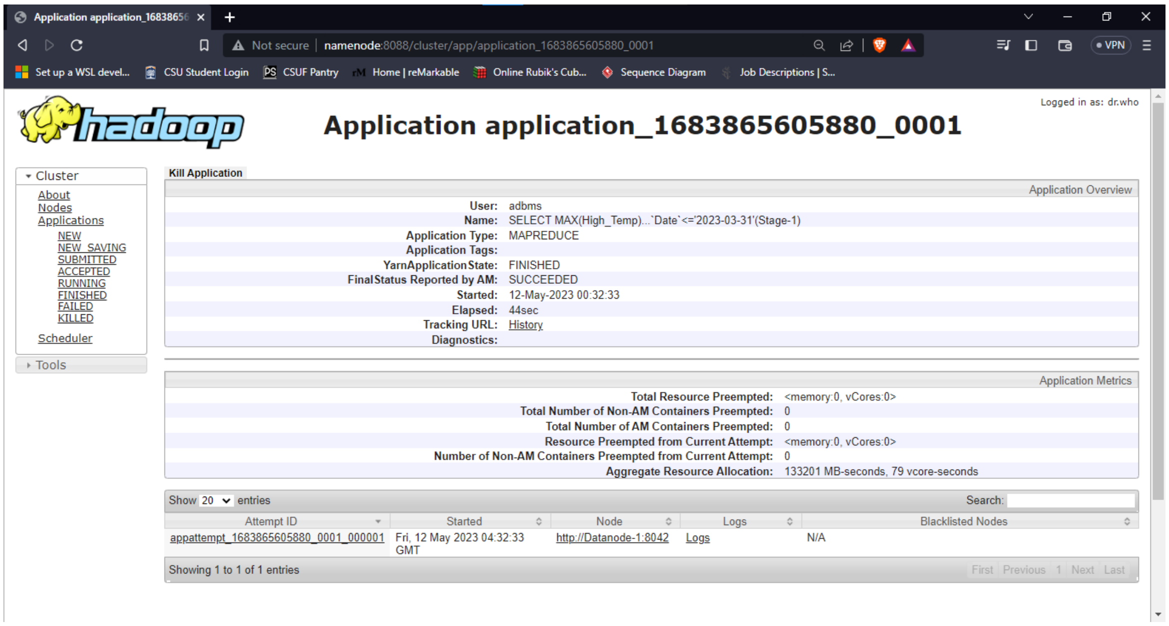
1. Stop Spark Services
   cd $SPARK_HOME/sbin
   ./stop-all.sh

2. Terminate HIVE metadata services
   Navigate to the terminal running the hive metadata service and press CTRL + C to terminate the session.

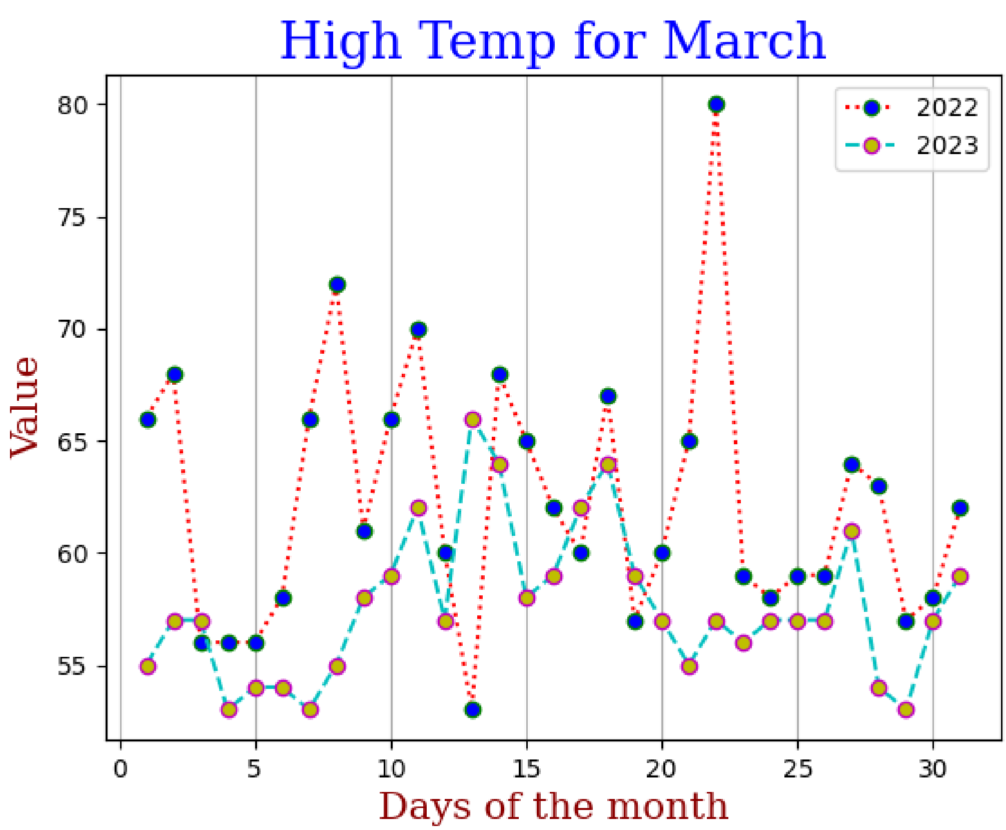
3. Stop Hadoop
   stop-all.sh

10) Test Results

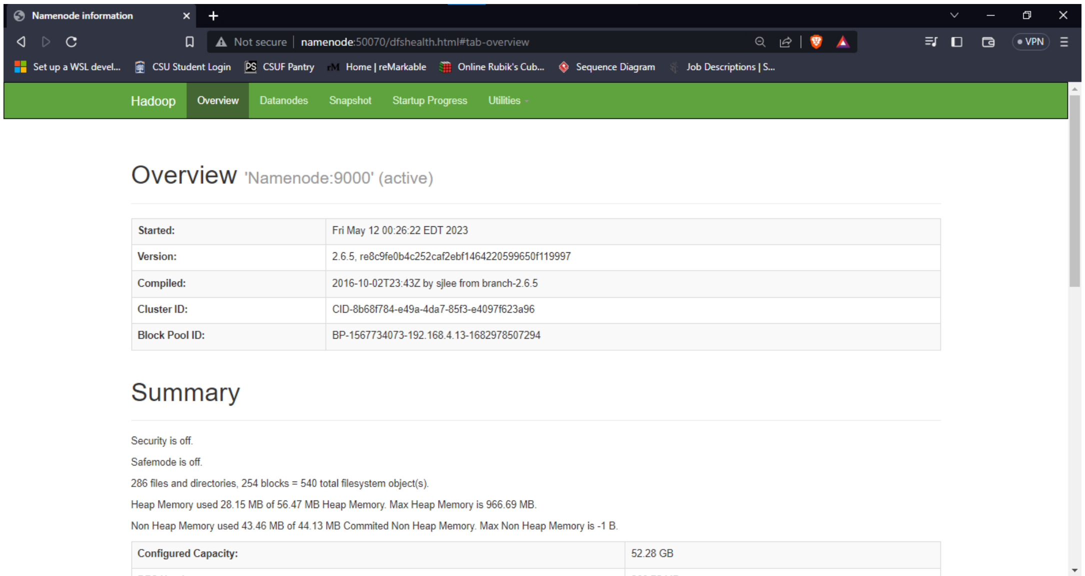
HIVE QL running a successful Hadoop MapReduce job



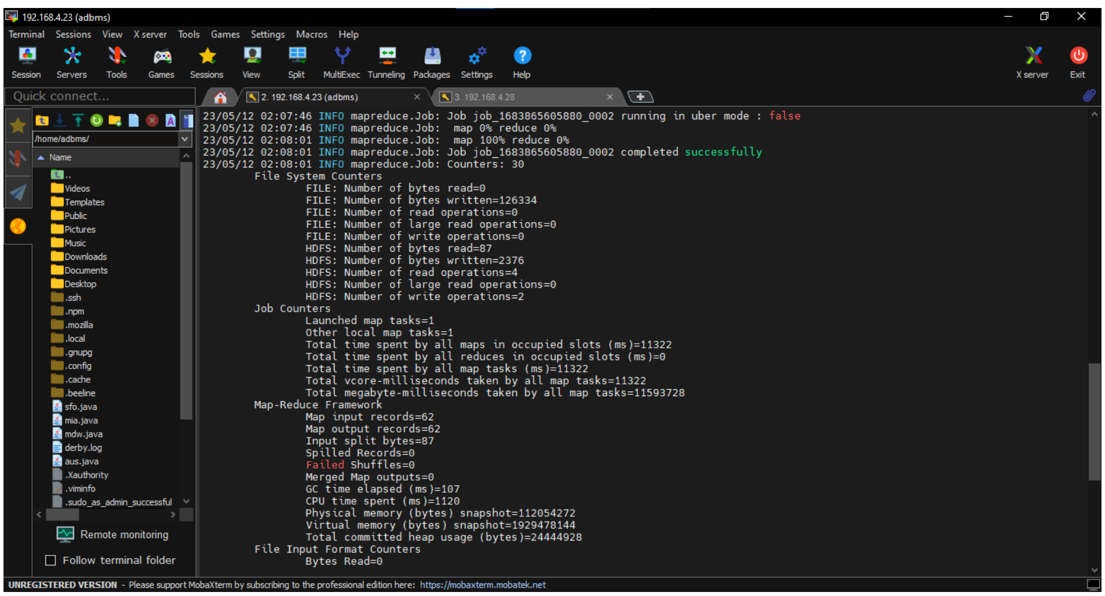
Graphical Analysis using Matplotlib



Hadoop Node Status



Sqoop Data Ingestion



11) <u>Conclusion</u>

Thus the weather analysis tools (CLI and Graphical Analysis tool) developed as part of the project can be used by a weather expert to perform descriptive analysis on the weather data collected from various sources.

This application is scalable for adding more cities and weather fields to the storage and leverages parallel processing as part of the MapReduce framework to run queries to analyze a huge amount of data to gain meaningful insights into the weather patterns for the cities used.

12) References

https://spark.apache.org/sql/

https://hadoop.apache.org/

https://www.wunderground.com/history

https://www.youtube.com/@dataengineeringvideos

https://www.youtube.com/@SimplilearnOfficial