Weather Data Analysis

VISHWANATH KODRE[1], SABYASACHI ROY CHOUDHURY[1], AND ABHIJIT THAKRE[1]

[1]School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[1]Corresponding authors: sabyasachi087@gmail.com, vkodre@gmail.com, athakre@gmail.com

project-S17-IR-P013, May 15, 2017

The project aims to analyze climate data with focusing on use of Hadoop Framework for data analysis and Ansible for automating deployment and monitoring.

Keywords: Cloud, I524

Report: https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IR-P013/report/report.pdf
Code: https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IR-P013/code

1. INTRODUCTION

The study of environmental science and climatic changes around has been done for decades, the study has always been predictive based on the past experiences and forecasting of the weather conditions around us. With use of modern days technologies it determining the climatic changes and with analysis done around it has helped human being to prepare and face the natural calamities. Though with current equipment weather department has strengthen their arms but has not been able to be full proof and many time its not been able to predict/ forecast the climatic changes effectively. The study of the whether data and geo graphical changes is ongoing evolving process. Thus more and more researcher needs modern days tools and technologies to leverage it and forecast more accurately.

1.1. Objective

The goal of this is to study the weather data and analyze the relationship between the geo graphical changes such change in geo magnetic field and/or natural disaster. With use of Hadoop for distributed data analysis aims to finds any pattern that might exists between these parameters. The course of the analysis will also provides visualization of these parameters in order to identify any pattern in a more intuitive way. By leveraging the power ansible for application deployment over cluster and monitoring the application performance to determine scalability and throughput. The conclusion will be determine by establishing any existing pattern, analysis done over it and by visualizing it.

2. DATA SOURCES

Weather data has been recorded since 19th century. This data can be used to estimate climate changes and forecasting. The same data can be can be used to find any existing pattern with natural disasters. Following sources has been compiled for weather, natural disaster and geo magnetic fields.

- Weather-Data[1]
- Natural Disaster[2]
- Geo Magnetic Field[3]

3. HIGH LEVEL DESIGN

The design of the application is thought of leveraging power of Hadoop as main processing unit of analysis with deployment on the cluster environment where application requires multiple processing units for execution, database for persistence and visualization tools for graphical outputs. The project is divided into following steps:

- Data cleaning and persistence - The raw data cannot be use directly for analysis. First data has to be parsed and required parameters will be extracted. Then this extracted data will be dumped into a NoSql database.
- Core Analysis Program - Core analysis program will be responsible for figuring out any hidden patterns between aforesaid parameters. Program will compare natural disasters occurred, geo-magnetic orientation and climate data set on a given location and duration and compute relationship between them. The program will be an MapReduce implementation and is the heart of the application. The program will be executed through Hadoop framework. Hadoop will execute the program in a distributed manner.
- Deployment and Monitoring - The application needs multiple processing units and monitoring system. Ansible will be used for deployment and manage nodes for program execution.Ansible will be responsible for following tasks i) Deployment and configuration of Hadoop on the multiple nodes. ii) Starting Hadoop servers, inserting/reading data. iii) Execution of the commands to run the analysis using

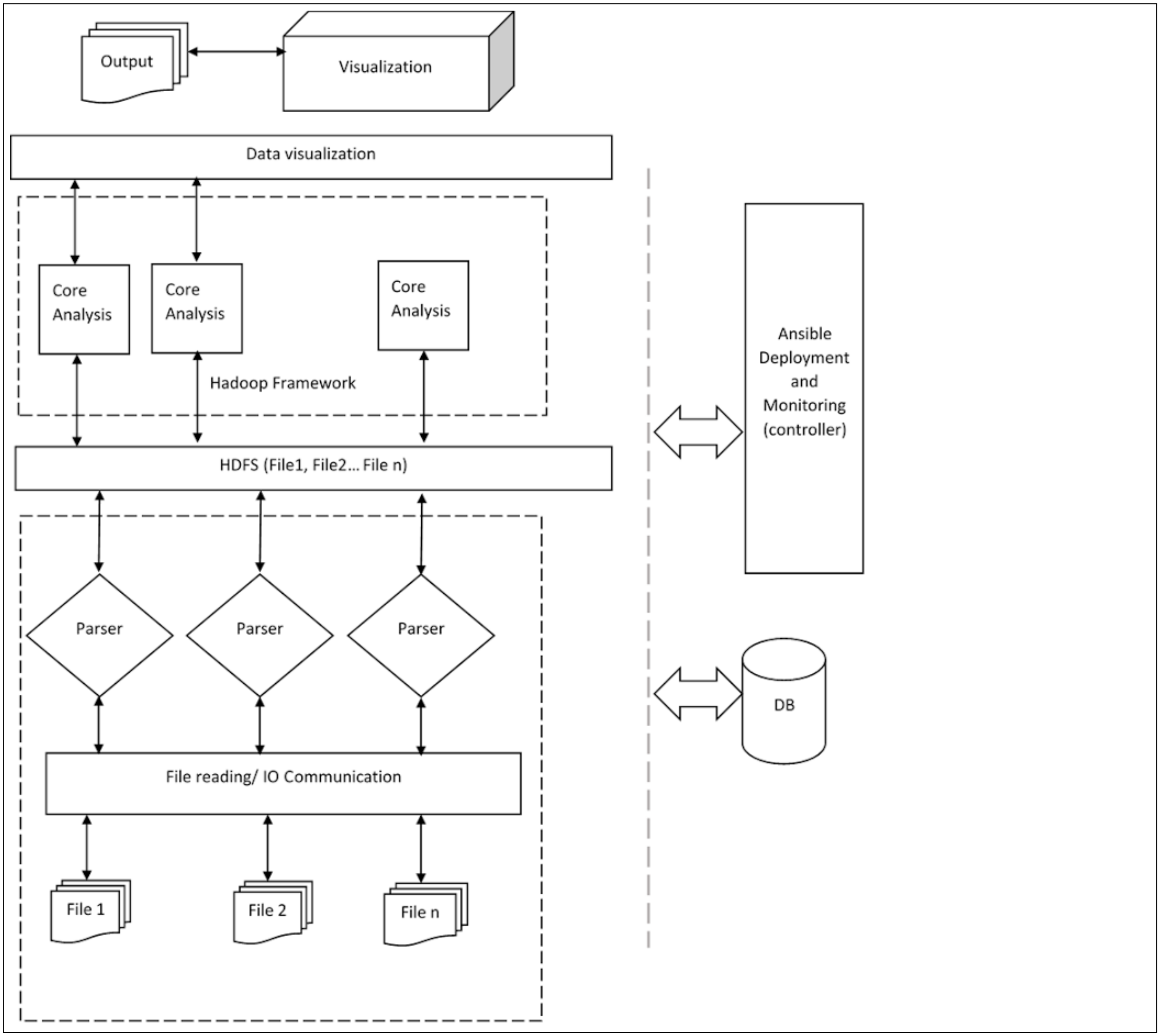

Fig. 1. Architecture

Hadoop to filter the input data and write response to HDFS or some output file. iv) This output can be then passes to the visualization step as the input data.

- Visualization - Finally once the programs completes execution, using the scikit-tool or other visualization tool kit and the output file, graphs and patterns depicting the relationship can be plotted more intuitive representation.
- BenchMarking - The application can be benchmarked for the scalability by addition more nodes and checking the performance for strong scaling. The report will be represented in tabular format.

4. DATA CURATION

Getting data ready is the very first and basic step for analysis. We have chose NCDC [4] as our source of data. NCDC [4] exposes few rest apis for accessing weather data. Following will give you a brief understanding on the apis used for getting the required data.

1. Datasets : This groups data into monthly daily , yearly pattern. There are eleven different datasets. We will be choosing GSOM (Global Summary Of Monthly) as our primary datasets. (URL : https://www.ncdc.noaa.gov/cdo-web/api/v2/datasets ,Attributes : GSOM). For this project we will use GSOM only.
2. Data Categories: This groups data into data category like Temperature, Pressure etc. We will consider only few. (URL : https://www.ncdc.noaa.gov/cdo-web/api/v2/datacategories, Attributes : "TEMP" (Air Temperature) , "PRES" (Pressure) , "EVAP" (Evaporation) and "VELOCITY" (Velocity)). For our project we will be using PRCP, SNOW, TMAX, TMIN and TAVG.
3. Data Types: This group Data Categories into further smaller sub types. (URL : https://www.ncdc.noaa.gov/cdo-web/api/v2/datatypes?datasetid=GSOM&datacategoryid=TEMP , Attributes : "TAVG" (Average Temperature), "TMAX" (Maximum Temperature) and "TMIN" (Minimum Temperature)) Other Data Categories does not have sub type.
4. Location Categories: This groups data in terms of location. (URL : https://www.ncdc.noaa.gov/cdo-web/api/v2/locationcategories , Attributes : "CITY" , "CLIM_DIV" (Climate Division), "CLIM_REG" (Climate Region) , "CNTRY" (Country) ,"ST" (State)). For this project we will be downloading data for India only.
5. Location: Groups data in terms of country, state etc. (URL : https://www.ncdc.noaa.gov/cdo-web/api/v2/locations?locationcategoryid=CNTRY, Attributes : "FIPS:IN" (INDIA) , "FIPS:IO" (Indian Ocean))
6. Station : This collects stations details based on the location id. (URL : https://www.ncdc.noaa.gov/cdo-web/api/v2/stations?locationid=FIPS:IN , Attributes: ids , mindate and maxdate). All of the stations details will be collected.
7. Data : This collects actual weather data for the given stationId, startdate and enddate. (URL : https://www.ncdc.noaa.gov/cdo-web/api/v2/data ?datasetid=GSOM&stationId=GHCND:IN1NCBC0005 &startdate=1970-10-10&enddate=1971-10-10). Since the project only requires only 5 attributes, further filter is applied while invoking the API. The data received will be saved into database.

4.1. Collecting Data

NCDC uses token based authentication for security and with each token , no more than 10,000 hits are allowed per day. Well 10,000 seems huge but truly its not. We have targeted one country (India) and 5 attributes chiefly precipitation, snow, maximum temperature, minimum temperature and average temperature for each month. Total number of weather data stations in the country is around 3500 and each station has a data range of 30 to 50 years. This leads to a total of 100,000 hits or more. To handle this, we need to load data incrementally, i.e. the curation program should have the ability to resume the download from the last save point. Before we dive more into logical section , lets first see the technology stack for data collection.

4.2. Technology Stack

We have considered python , Apache thrift and Apache Hbase for data curation step. We will be covering a short introduction of the steps to configure the system, before diving into the core logic of curation. Since we are going to use Apache Hadoop for our analysis purpose, Hbase comes as a natural choice of NoSql database. Java is the default language for both Hadoop and Hbase. So now the question is how to connect to Hbase using python and the solution is Apache Thrift or thrift. Thrift is a library from apache which generates hbase client. Thrift supports many third party languages including python. The following steps will be required to install thrift first.

4.2.1. Install Apache Thrift [5]

1. Download Thrift from "http://redrockdigimark.com/apachemirror/th 0.10.0.tar.gz".
2. Extract the file and run ./configure.
3. Execute sudo make install or simply make to generate the binaries. If using make, the thrift binaries has to put into path manually (can be found under compiler/cpp/).
4. After thrift is available in path , it can be tested by running "thrift –version".
5. Then the python module has to be created to be used within python. To do this download "https://github.com/apache/hbase/blob/master/hbase-thrift/ src/main/resources/org/apache/hadoop/hbase/thrift/Hbase.thrift" file.
6. Then execute "thrift –gen py <path/to/Hbase.thrift>". This will generate "gen-py" folder.
7. Add this folder to python path by export PYTHONPATH=<path/to>/gen-py/:$PYTHONPATH.

For details you can check Installation steps.Installation of Hbase will be discussed later along with Hadoop. Now thrift is ready to use. Execute "/hbase thrift start" to start the thrift server. Hbase has to be started separately.

4.2.2. Install Happybase [6]

Happybase is a wrapper program written on thrift to facilitate data access layer in python in a more readable and swift way. Rather than using thrift client directly we will be using happybase python module. To install happybase run "pip install happybase". Once done test the python library by running the following test code

Listing 1. Connect-Hbase

```
import happybase as hbase

connection = hbase.Connection('localhost')
print connection.tables()
```

5. HBASE AND TABLE STRUCTURE

Hbase [7] is a column based key-value database. We have considered two tables for weather data 'wda_stations' and 'wda_weather'. The structure of the tables are as follows

Listing 2. WDA_STATIONS Table

```
[{station-id-key : {
    station : {name : {} , id : {}},
    date_range : {
        min_date : {} ,
        max_date : {}
     },
    location : {
        latitude : {},
        longitude : {}
    }
}}]
```

Listing 3. WDA_WEATHER Table

```
[{yyyy-mm-key : {
    weather : {station-id : {
        data-type : {value}
}}}}]
```

5.1. Table Definition

Stations table will consist of all the weather stations with its location and the date range within which it is was active. Location will have the latitude and longitude. The station id will be the key for the table. NCDC uses a uniquely identifiable key for the stations across the globe. This makes the best attribute for the primary key irrespective of country and region. Weather table is a more completed table and has dynamic column family. The key for the table is the year and month in (YYYY-MM) format. Each key contains all the stations with their weather parameter. For example say a key 1967-01 (meaning January 1967) is having 10 stations with station id sid-1 to sid-10. Each station id will having 4 parameters or data types (TAVG,TMAX,TMIN and PRCP). And finally each parameter have their values.

6. DATA PERSISTENCE LAYER

The project is structured with python packages. We have a specific package for persistence as 'iu.i524.S17IRP013.dao'. The dao has the scripts to connect hbase and performs db operations. The base script is 'hbase_connect.py' which does the connection. The connection is made through Apache Thrift Rest Api. It has the connection object , which is used by the other two dao classes.

6.1. Station Dao

This script does all CRUD operations on stations table. On startup it will create the table if not available. It has 2 main functions insert and retrieve. The insert function takes key value pair as argument and put it into hbase table. Another function and the important one is 'get_station_data' which takes the start row as argument and returns next ten records. This is done via scan command of hbase which supports record limitation.

Listing 4. Stations Dao

```
def get_station_data(start_row=''):
    st_list = dict()
    count = 0
    if(start_row == ''):
        for key, data in table.scan(limit=10):
            st_list[key] = {'min_date':data['date_range:
                 min_date'], 'max_date':data['date_range
                 :max_date'] , \
                                'station_id': data['station:
                                    id'], \
                                'latitude':data['location:
                                    latitude'], \
                                'longitude':data['location:
                                    longitude']}
    else:
        for key, data in table.scan(limit=11, row_start=
             start_row):
            count = count + 1
            if(count > 1):
                st_list[key] = {'min_date':data['
                     date_range:min_date'], 'max_date':
                     data['date_range:max_date'], \
                                'station_id': data['station:
                                    id'], \
                                'latitude':data['location:
                                    latitude'], \
                                'longitude':data['location:
                                    longitude']}
    return st_list
```

6.2. Weather Dao

Weather dao is similar to station in dao in all respect except it deals with weather table. Apart from the getting records from the weather table, there is another method to get all keys from the table. We will see in later sections why we need the entire key list from the table.

7. WEATHER DATA PERSISTENCE

Once happybase, thrift and hbase is functional, we are ready to download our weather data. Weather data download is divided into two steps 1) Getting weather stations details for a given country and 2) Getting Weather data for a given station. Lets see them individually :

- Download Weather Stations - To consume rest services we have used python's inbuilt request response module. Let us walk you through the code. We have two main python script one for data access and another for rest consumption. "stations_dao.py" is for accessing table "wda_stations" in hbase. The functions are self explaining and hence will not be repeated here. "weather_services.py" is for consuming NCDC rest services. To load all stations , we will be using the code *'FIPS:IN'* i.e the region code for India.

Listing 5. Get Weather Stations

```
def get_stations(country='FIPS:IN', offset=0):
    url = NCDC_API + NCDC_SERVICES.STATIONS + \
    '?locationid=' + country + '&offset=' + str(
         offset)
```

```
    response = requests.get(url, headers=HEADERS)
    return response.json()

def load_stations():
    limit = 25
    offset = 0
    nbr_of_records = 0
    result = Services.get_stations(offset=offset)
    while(result != {}):
        nbr_of_records = nbr_of_records \
        + insert_station(result['results'])
        offset = offset + limit
        result = Services.get_stations(offset=
            offset)
    print str(nbr_of_records) \
    + ' stations loaded successfully !!'
```

The function 'load_stations' calls 'get_stations' till all the available stations are downloaded. 'insert_station' is the *DAO (Data Access Object)* call and inserts the dataset into hbase table.

- Download Weather Data - With all stations in the table, we invoke the data api of NCDC for downloading weather data. The argument for getting weather data is station id and date range. The two main methods for loading weather data are as follows

Listing 6. Get Weather Stations

```
def load_weather_data():
    start_row = ''
    st_list = stations.get_station_data(start_row=
        start_row)
    while st_list != {}:
        for key, value in st_list.items():
            get_weather_data(startDate=value['
                min_date'], \
            endDate=value['max_date'], stationId=
                key, station_details=value)
            start_row = key
        print 'Weather Data loaded till station id 
            = ' + key
        st_list = stations.get_station_data(
            start_row=start_row)

def get_weather_data(station_details, startDate='
    1968-01-01', endDate='1970-01-01', stationId='
    GHCND:IN001011001'):
    date_range_list = AppUtil.get_year_list(
        start_date=startDate, end_date=endDate)
    for date_range in date_range_list:
        limit = 25
        offset = 0
        result = Services.get_weather_data(
            startDate=date_range['min_date'],\
            endDate=date_range['max_date'],
            stationId=stationId, offset=offset)
        while(result != {}):
            # print result['results']
            insert_result(result['results'],
                station_details)
            offset = offset + limit
            result = Services.get_weather_data(
                startDate=date_range['min_date'],\
                endDate=date_range['max_date'],
                stationId=stationId, offset=offset)
```

The 'load_weather_data' function retrieves station data from the station table in a batch of ten. For each station id, it is then passed to 'get_weather_data' function to get weather data for the given station id. This data is again persisted into wda_weather table for future analysis and usage. These are the main functions to download weather data. All other functions are helper functions to enable the download.

8. INSTALL PYTHON PROGRAM

Our python programs are structure and packaged. So to use it, we need to install the packages , in order to make them available in the classpath or pythonpath. There is a setup.py script provided with the main source folder. This script installs the entire package into python core folder. To execute this *pip* must be pre-installed. Once pip is available run the following command

Listing 7. Install python packages

```
sudo pip install  <path_to_src_folder>/src
```

Use sudo or super user to install as other may have permissions issues. Once they are installed , we are ready to run our map reduce program.

9. EXECUTION

The application expects all the data in hbase are prefetched and ready. For downloading the datasets into hbase follow the steps

- Install the python packages. Check all installations hbase, hadoop , thrift and happybase.
- Start Hadoop and Hbase and Apache Thrift.
- Open up python cli and run the command 'iu.i524.S17IRP013.hadoop.init.DataSetup.py'.
- The above command will take time and will dump data into hbase table.
- It also created hdfs input folder as '/wda/input' and write 'wda_row_keys.txt' file into it. This file will have all the keys (year-month format) from weather table.

9.1. Run MapReduce

We need hadoop streaming api to execute our ma reduce program. Hadoop streaming api reads from hdfs and write it onto the standard io. This data is then read by our map program *wda_mapper.py* and further sent to reducer program *wda_reducer.py*.

Listing 8. Run MapReduce

```
bin/hadoop jar <path_to_streaming_jar>{share/hadoop/
    tools/lib}/hadoop-streaming-2.7.3.jar \
-file <path_to_mapper>/run/wda_mapper.py -mapper
    wda_mapper.py \
-file <path_to_reducer>/run/wda_reducer.py -reducer
    wda_reducer.py \
-input /wda/input -output /wda/output
```

The above will execute the map reduce analysis program and write the output into hdfs '/wda/output' folder. Once the program is finished check the output with following command :

Listing 9. Run MapReduce

```
hdfs dfs -ls /wda/output
hdfs dfs -cat /wda/output/<name_of_the_file>{part-r
    -00000}
Output >>
{'TAVG': ('GHCND:IN001020700', '35.02'),
'TMAX': ('GHCND:IN001020700', '43.08'),
'PRCP': ('GHCND:IN001011000', '863.7'),
'TMIN': ('GHCND:IN001020700', '14.08')}
```

10. DEPLOYMENT USING ANSIBLE

Ansible is open source automation tool. It can be used for deployment of software, configuration management and automation in the execution of application. It also serves for monitoring of the state of the application. As per current state of our project we have used ansible for deployment of software in our project. The script deploys Java, Hadoop, Hbase on the independent clusters. It also configures the properties in the key files. This script can further be extended to build the analysis , deploy and run the analysis module. Monitor the progress of the execution and display the reports, however currently it is work to be done.

In the project a separate directory structure is created for deployment. This folder consists of some key files and folders. We can dig into the details for each of them in below section.

1. Inventory.
2. Playbook.yml.
3. Roles —> Roles is a directory and we can going to explore the roles used in the project as we proceed.

10.1. Inventory Configuration

Inventory file contains the list of hostname or nodes that can be accessible by ansible. These nodes are then used in the script for deployment and configuration. In the current project chameleon server nodes were created and configured. These IP address can be changed dynamically. This file can also contain the group inside Which multiple ip can be configured. The deployment of hadoop was in clustered mode where we have on master and multiple slaves. In the inventory.configuration file there are multiple section with header which mentioned the list of nodes/IP address those needs to be included in the group. For example in the below file we have two groups.

Listing 10. Inventory

```
[weatherClusterMaster]
    <ipaddress1> ansible_ssh_user=cc
[weatherClusterSlave]
    <ipaddress2>  ansible_ssh_user=cc
    <ipaddress3> ansible_ssh_user=cc
```

1. WeatherClusterMaster : Node/Instance with ipaddress1 falls under the master cluster.
2. WeatherClusterSlave: Node/Instance with ipaddress2, ipaddress3 falls under the slave cluster.

The mapping of IP address to the group from the inventory file is used in the playbook.yml to represent them as a group. There are some script which needs to be part of the master cluster deployment however it don't need to be executed on the slave cluster. This problem is address using the playbook.yml roles

10.2. Playbook

Playbook works on mentioned host of group. It mentions the roles those will be operated on each of the host. The main.yml inside the task folder within each roles will be applied to The cluster.

Listing 11. Playbook Modules

```
---
- hosts: weatherClusterMaster
 remote\_user: root roles:
     - java
     - hadoop
     - hbase
     - master
- hosts: weatherClusterSlave remote\_user: root
roles:
     - java
     - hadoop
     - hbase
```

As mentioned above the hosts mentions the instance on which the scripts are going to be executed. So all the ipaddress those are mentioned under the weatherClusterMaster will undergo the execution of scripts those are mentioned in each of the roles for that task. These script will be executed using the remote_user mentioned in the script. In above example from project, both the cluster will execute the script mentioned inside java, hadoop and hbase role however master cluster additionally will execute the script from the master role folder as well. These could be the addition scripts like configuration of slave addresses in the slaves file and many such task which needs to be executed only for master instance.

10.3. Roles

Roles folder contains the list of the roles mentioned in the playbook.xml. There could be multiple roles which can be executed on node or group of nodes. In current project there are four roles i.e. java, hadoop, base and master. The name are very much self explainatory. Java : — Java role installs java on the remote machine . Roles contains multiple directory. The task directory is the main directory which mentions the list of task / script those needs to be executed on the nodes.

Current project mentions 4 different task in java. All these task takes care of installing java on the remote node.

Hadoop — Hadoop along with Hbase is the major role. The task file mentioned the below list of task

1. Download hadoop.
2. Extract hadoop.
3. Set java home/ Hadoop home configuration in bashrc.
4. Set Java home/Hadoop home path to env.rc
5. Create and Configure logging file.
6. Update core-site.xml
7. Configure slaves
8. Update other configuration files like core-site.xml, hdfs-site.xml, yarn-site, mapper- site.xml.
9. Replace hosts.

Sources fIle
The sources folder consists of other key folders like

1. Vars This is yml file which maintains the mapping for placeholder names used in the configuration files.
2. Meta : This file can contain it dependency roles and other details.
3. Templates : This contains the template configuration file.

After Hadoop, HBase, Thrift is installed and configured using similar way. You can refer to the main.yml inside the task folder in for the script details.

11. BENCHMARKING

The aforesaid program has been executed on futuresystems. The execution time has been collected by increasing nodes from 1 (single) to 3 (multiple) with 2cpus each. The graph indicates the

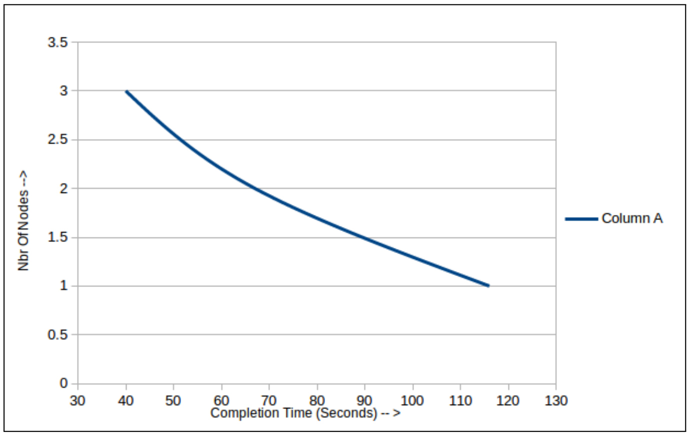

Fig. 2. Benchmarking

reduction in time with increase in nodes. Since the data used for the sample is not big enough , the time taken for starting the framework is considerably high as compared to total time taken by the analysis program itself. So only the map reduce time or

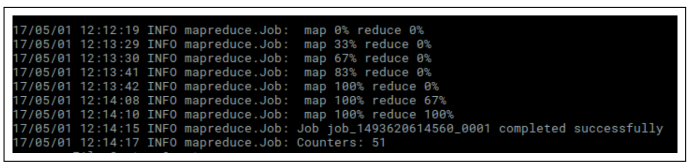

Fig. 3. Map Reduce Snapshot

the program time has been considered and not the ancillary time which includes time to start the hadoop framework.

12. SCOPE OF EXTENSIONS

The application is only distribute the data load from the hbase table. But fetching data from NCDC is sequential. Though its only a one time job , it takes considerable amount of time to download. The same application can be extended to use Hadoop MapReduce for downloading and persisting the raw data from Rest APIs. The same fundamentals can be used for this purpose. Initial weather stations can be downloaded sequentially and then the keys are to be distributed in HDFS for further weather data download.

12.1. Limitation

There is a limitation in download imposed by NCDC. NCDC uses token bases authentication which allows only 10000 invocation per day. To distribute the download among multiple nodes requires multiple tokens.

13. PROBLEM AND WORKAROUND

Usually Hadoop is run with java and connects directly with the Hbase to retrieve the data from database. However in the current implementation we have python script which is used for the analysis. The challenge was to fetch the data from Hbase by running the Hadoop program in python. There were some of the library available which connects Hadoop to Hbase however those were not reliable and supported by Apache.

Workaround for Connecting Hadoop to Hbase

In order to execute the python script on Hadoop in cluster, there was a work around considered. In this work around, all the keys were stored in the hadoop file system. These key were then divider as per the nodes and passed to the mapper method. For each key, mapper method called the hbase to fetch the data. This call to Hbase was via apache thrift and happybase

Let drill down more on what is Apache thrift.

Apache thrift is mechanism or framework which can be talk between two services which may or may not been written in same language. Apache thrift has its own Interface definition language, its compiler can generate client code and server code in same or different language and hence can be used as a bridge to talk between the two different language.

Apache thrift is very similar to SOAP. The way SOAP uses UDDI for publishing and discovering the service, thrift uses zoo keeper for finding services. Apache thrift was connecting to the Base with the help of Apache

HappyBase is a developer-friendly Python library to interact with Apache HBase. Below the surface, HappyBase uses the Python Thrift library to connect to HBase.

13.1. Security

This project has been deployed and tested on FutureSystems. To create instances on futuresystems follow the link. It uses nova client for managing instances. Now to run the Ansible playbook the key file has to be provided with –private-key or the default id_rsa will be used. There are two ways to handle this

1. The key used for creating the instances needs to be downloaded into your local file and pass the same key for running the ansible playbook. This can be done through scp command within india.futuresystems.org.
2. Copy your local keys (/.ssh/id_rsa & /.ssh/id_rsa.pub) into india.futuresystems.org. Use the same key to create the instances. In this process you don't have to provide the key file explicitly as it would be using /.ssh/id_rsa by default.

The aforesaid process is manual. We have used the first option to carry out the project.

14. CONCLUSION

MapReduce (MR) with Hadoop is an efficient framework for distributed computing. It can be run on any commodity hardwares and virtual machines. It has also some useful plugins available for shared computing which can share dataset without doing any IO operations,ex - Twister and Spark. Python with Hadoop on other hand is not so great combination as the communication requires additional layer i,e Standard IO. We have seen earlier that there is no support for Hbase communication within the hadoop as well. FutureSystems and Jetstream works well with the framework. Chameleon VMs did not have all ports opened (except 22 for ssh) which is a must for Hadoop to work in cluster as Hadoop uses IPC protocol for inter node communication.

REFERENCES

[1] "Weather data," Web page. [Online]. Available: https://www.ncdc.noaa.gov/

[2] “Natural disaster,” Web page. [Online]. Available: http://www.emdat.be/
[3] “Geo magnetic field data,” Web page. [Online]. Available: https://geohazards.usgs.gov/mailman/listinfo/geomag-data
[4] “Ncdc rest api,” Web page, Jan. 2017, accessed: 2017-04-25. [Online]. Available: https://www.ncdc.noaa.gov/cdo-web/webservices/v2
[5] BRUNDESH, “Install apache thrift with python,” Web Page, Jan. 2017, accessed: 2016-01-25. [Online]. Available: https://acadgild.com/blog/connecting-hbase-with-python-application-using-thrift-server/
[6] Happybase, “Install happybase,” Web Page, Apr. 2017. [Online]. Available: https://happybase.readthedocs.io/en/latest/installation.html
[7] A. Hbase, “Apache hbase,” Web Page, Apr. 2017, accessed: 2017-04-27. [Online]. Available: https://hbase.apache.org/