EAST WEST UNIVERSITY

Investigating Scope of NoSQL approach using Hadoop platform

Submitted by:

Rohul Amin Shuvo

ID: 2011-3-60-011

Fatema Tuz Zohora

ID: 2011-3-60-010

Supervised by:

Dr. Mohammad Rezwanul Huq

Assistant Professor

Department of Computer Science and Engineering

East West University

A project submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering to the Department of Computer Science and Engineering

Spring 2016

Abstract

With the current emphasis on huge data storing, NoSQL database have surged in popularity. Big data is used for massive data sets having large and complex structure with difficulties of storing, analyzing and visualizing for further results. These databases are claimed to perform better than SQL databases. The primary purpose of this project is to independently investigate the performance of some NoSQL and SQL-based approach. We used maximum, minimum and average operations on key-value stores implemented by NoSQL and SQL databases. In these project we used synthetically generated temperature data for experimental purpose.

[Rohul Amin Shuvo]

[Fatema Tuz Zohora]

Declaration

We hereby declare that, this project was done under CSE497 and has not been submitted elsewhere for requirement of any degree or diploma or any purpose except for publication.

Signature of the Candidates

_____

Rohul Amin Shuvo

ID: 2011-3-60-011

_____

Fatema Tuz Zohora

ID: 2011-3-60-010

Letter of Acceptance

This thesis is submitted by Rohul Amin shuvo, Id: 2011-3-60-011 and Fatema Tuz Zohora, Id: 2011-3-60-010 to the department of Computer Science and Engineering, East West University, Dhaka Bangladesh is accepted as satisfactory for the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering on May 5th, 2016.

Board of Examiners

1. ----------

Dr. Mohammad Rezwanul Huq

Assistant Professor (Supervisor)

Department of Computer Science and Engineering

East West University, Dhaka, Bangladesh

2. ----------

Prof. Dr. Md. Mozammel Huq Azad Khan

Professor & Chairperson

Department of Computer Science and Engineering

East West University, Dhaka, Bangladesh

Acknowledgement

We would like to thank Almighty Allah for giving us the strength and patience.

We would like to thank our honorable teacher and supervisor, Dr. Mohammad Rezwanul Huq, Assistant Professor, Dept. of CSE at East West University, who guided us to proper analysis of the system with support and feedbacks.

We also thank the researchers around the world for their work that helped us to learn and implement Hadoop MapReduce model.

Finally, we would like to thank our parents whom has given us support and love. Without them we would not be able to achieve this far.

Rohul Amin Shuvo

Fatema Tuz Zohora

Table of Contents

Chapter 3:

Theoretical Comparison 15-18

Chapter 4:

Working Procedure 12-16

Chapter 5:

Conclusion & Future Work 17

Chapter 1

Introduction

1.1 Problem Specification

NoSQL is one of the another type of data storage other than databases that is used to store huge amount of data storage. NoSQL is a non-relational database management system and is portable. On the other-hand traditional databases are relational and are well suited to meet the ACID criteria for data. Data is held in tables connected by relational algebra, and transactions are performed in a way that is consistent with ACID principles. But for non-relational databases ACID has always been sacrificed for other qualities, like speed and scalability. NoSQL and SQL both have their own features. But for future purpose and huge data collection and also unstructured data, NoSQL is the light. SQL can only work with structured data, so when we will be in need of storing and analyzing unstructured data, we can only depend on non-relational database.

1.2 Motivation

As today there is an enormous volume of data, examining these large sets contains structure and unstructured data of different types and sizes. Data Analytics allows the user to analyze the unusable data to make a faster and better decision. Now-a-days organizations are used to deal with huge data, which provokes various new ways of thoughts regarding how the data are produced, ordered, controlled and analyzed. Big Data can be given an all-encompassing term for any collection of data sets so large or complex. It becomes difficult to process those data using traditional data processing application. This paper represents the comparison of processing data using SQL and Big Data. In future the traditional databases can’t hold the enormous volume of data. So it’s better to change our thoughts about database before time.

1.3 Purpose of the Project

We have developed two applications to measure the execution time for SQL and NoSQL database. We have worked with some synthetically generated data of Bangladesh's seven cities from year 2001 to 2014 for 365 days of year. The application is developed in a manner where user can choose year, month, day and the aggregate functions maximum and minimum to see the results and performance of both SQL and NoSQL database. With these we want to prove that only NoSQL can hold the future of the enormous volume of data and execute it faster than any SQL.

1.4 Technology stack

These SQL and NoSQL applications are developed by java programming language. We have used a processing technique for NoSQL application named MapReduce which is a program model for distributed computing based on java on Hadoop platform. The SQL application has been developed with java and MySQL.

Programming Language: Java

Database: MYSQL

Graphical User Interface: Java Swing

Chapter 2

Terminology

2.1 Overview of SQL

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database. It can store and analyze structured data. SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix and SQL Server use SQL as standard database language. By writing queries we can manipulate data stored in database. SQL database is available in Basic, Standard, and Premium service tiers. Each service tier offers different levels of performance and capabilities to support lightweight to heavyweight database workloads. We can build an application on a small database, then change the service tier manually or programmatically at any time as our application needs, without downtime of our application. MySQL is an open source relational database management system. It is more powerful elixir database; it has gradually evolved to support higher-scale needs as well. It is still most commonly used in small to medium scale single server deployments, either as a component in web application or as a standalone database server.

2.1.1 Characteristics of SQL

- SQL is very simple and easy to learn.
- SQL allows the user to create, update, delete, and retrieve data from a database.
- SQL is an ANSI and ISO standard computer language for creating and manipulating databases.

- SQL works with database programs like DB2, Oracle, MS Access, Sybase, MS SQL Server etc.
- Applications written in SQL can be easily ported across systems. Such porting could be required when DBMS needs to upgrade.

2.1.2 Limitations

- Works with structured data only
- Relations must have a fixed set of columns
- High cost for data processing

2.2 Overview of NoSQL

NoSQL means Not Only SQL, implying that when designing a software solution or product, there are more than one storage mechanism that could be used based on the needs. Developers are working with applications that create massive volumes of new, frequently changing data types structured, semi-structured, unstructured and polymorphic data. Relational databases were not designed to handle semi-structured or unstructured data that face modern applications, nor were they built to take advantage of the processing power available today.

2.2.1 Characteristics of NoSQL

- NoSQL does not use SQL language.
- NoSQL stores large volume of data.
- In distributed environment we use NoSQL without any inconsistency.
- If any faults or failures exist in any machine, then there will be no discontinuation of any work.
- NoSQL is open source database, its source code is available to everyone and is free to use it without any overheads.
- NoSQL allows data to store in any record not having any fixed schema.
- NoSQL does not use concept of ACID properties.
- It has more flexible structure.
- NoSQL is scalable leading to high performance in a linear way.

2.2.2 Limitations

- NoSQL database demands a lot of technical skill with both installation and maintenance.
- In present, it is easier to find an RDBMS expert than a NoSQL expert.

2.2.3 Overview of Hadoop and MapReduce

Hadoop is an open-source software framework. It is used for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power and the ability to handle virtually limitless concurrent tasks or jobs [10]. The reason behind organizations turn to Hadoop is its ability to store and process huge amounts of data and any kind of data quickly. Currently, four core modules are included in the basic framework from the apache foundation. In this project we worked with the software programming model MapReduce. It is the programming paradigm that allows executing operations on massive amount of data across hundreds or thousands of servers in a Hadoop cluster. MapReduce refers to two separate and distinct tasks that Hadoop programs perform [16]. One is map job and second one is reduce job. The map job takes a set of data and converts it into another set of data, where individual elements are broken down into tuples. The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. The reduce job is always performed after the map job that's why it's called MapReduce.

2.2.4 Role of Data Architecture in NoSQL

There are four components in NOSQL's building block.

1. **Modelling language:** It describes the structure of the database and also defines schema on which it is based.
2. **Database Structure:** Each and every database while building uses its own data structures and stores data using permanent storage device.
3. **Database Query language:** All the operations are performed on the database that are create, update, read and delete (CURD).
4. **Transactions:** During any transaction in the data, there may be some types of faults or failure. But the machine will not stop working.

Chapter 3

Theoretical Comparison

3.1 Comparison Between Databases

In this project for NoSQL we have used Hadoop which is basically a distributed file system, not a database. It let us store large amount of file data on a cloud of machines, handling data redundancy. For the other part of this project we have used MySQL as SQL Database. It is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language(SQL). SQL is the most popular language for adding, accessing and managing content in a database. It is most noted for its quick processing, proven reliability, ease and flexibility of use.

3.1.1 MySQL

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. Now we will discuss about how MySQL components works together.

The topmost layer of MySQL architecture contains the services that aren't unique to MySQL. They're services most network-based client/server tools or servers need: connection handling, authentication, security etc.

In second layer the brain of MySQL is here, including code for query parsing, analysis, optimization, caching and all the built in functions. Stored procedures, triggers and views all functions provided across storage engines lives in this level.

The third layer contains the storage engines, which are responsible for storing and retrieving all data stored in MySQL. Each storage engine has its own advantages and limitations. The low level functions that performs operations like beginning of transaction or fetching row that has primary key are in the API. The storage engines don't parse SQL or communicate with each other; they simply respond to requests from the server [12].

The applications fetch the data from database through a driver. Then the driver manager requests the connector to connect with the server and with the query the application gets the data it was expecting.

3.1.2 Hadoop

Hadoop provides an API for processing all that stored data - Map-Reduce. The basic idea is that since the data is stored in many nodes, it's better off processing it in a distributed manner where each node can process the data stored on it rather than spend a lot of time moving it over the network. Hadoop can handle data in a very fluid way. Hadoop is more than just a faster, cheaper database and analytics tool. Hadoop doesn't insist to structure data. Data may be unstructured and schema less. Users can dump their data into the framework without needing to reformat it [15]. By contrast, relational databases require that data be structured and schemas be defined before storing the data. Hadoop has two parts. One is MapReduce and another is HDFS. The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has a master/slave architecture. Advantages of this architecture is backups of the entire database has relatively no impact on the master. Analytic applications can read from the slave without impacting the master. MapReduce is the original framework for writing applications that process large amounts of structured and unstructured data stored in the Hadoop Distributed File System (HDFS). Apache Hadoop YARN opened Hadoop to other data processing engines that can now run existing MapReduce jobs to process data in many different ways at the same time. MapReduce has a simple model of data

processing: inputs and outputs for the map and reduce functions are key-value pairs. The map and reduce functions in Hadoop MapReduce have the following general form:

map: (K1, V1) → list (K2, V2)

reduce: (K2, list(V2)) → list (K3, V3)

The first phase of a MapReduce program is called mapping. A list of data elements is provided, one at a time, to a function called the Mapper, which converts each element individually to an output data element. Reducing lets user aggregate values together. A reducer function receives an iterator of input values from an input list. It then combines these values together, returning a single output value. Reducing use to turn a large volume of data into a smaller data. In MapReduce, no value stands on its own. Every value has a key associated with it. Keys identify related values. The mapping and reducing functions receive not just values, but pairs of key with value. The output of each of these functions is the same. How MapReduce works with word count is given below in figure-3.1.2.

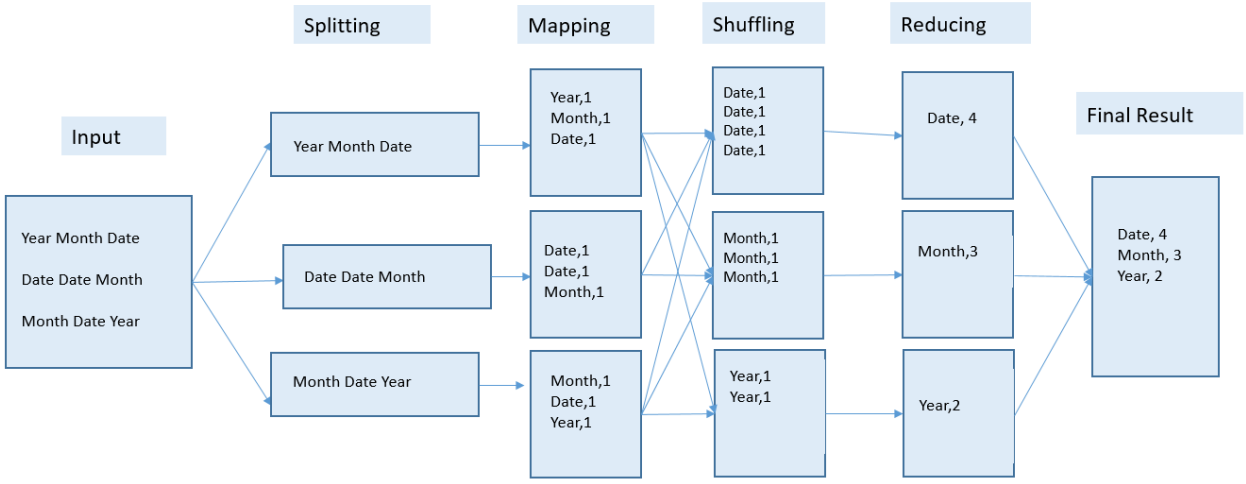

Figure-3.1.2 MapReduce word count problem

MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts

the outputs of the maps which are then input to the reduce tasks. Typically, both the input and output file are stored in Hadoop distribute file system. Reduce gathers its output while all the tasks are processing. Reduce can't begin before all mapping is done. The output of reduce is also a key and value. In figure-3.1.2 is a data-set of some words. The MapReduce job will split the input into some chunks. Then in the map task it will generate the chunks as some key value pair. Then in shuffling task it will collect the key value pair in alphabetic order. At last reducer will gather the output.

3.2 Functional Programming instead of Declarative Queries

SQL is fundamentally a high-level declarative language. Users query data by stating the result they want and let the database engine figure out how to derive it. Under MapReduce user specify the actual steps in processing the data, which is more analogous to an execution plan for a SQL engine. Under SQL user has query statements; under MapReduce user has scripts and codes. MapReduce allows to process data in a more general fashion than SQL queries. For example, one can build complex statistical models from data or reformat image data. SQL is not well designed for such tasks.

On the other hand, when working with data that do fit well into relational structures, some people may find MapReduce less natural to use. Those who are accustomed to the SQL paradigm may find it challenging to think in the MapReduce way. But note that many extensions are available to allow one to take advantage of the scalability of Hadoop while programming in more familiar paradigms. In fact, some enable to write queries in a SQL-like language, and the query is automatically compiled into MapReduce code for execution [13].

3.3 Scale-out Instead of Scale-up

Scaling commercial relational databases is expensive. Their design is more friendly to scaling up. To run a bigger database one need to buy a bigger machine. In fact, it's not unusual to see server vendors market their expensive high-end machines as "database-class servers." Unfortunately, at some point there won't be a big enough machine available for larger data sets. More importantly, the high-end machines are not cost effective for many applications. For example, a machine with four times the power of a standard PC costs a lot more than putting four such PCs in a cluster. Hadoop is designed to be a scale-out architecture operating on a cluster of commodity PC machines. Adding more resources means adding more machines to the Hadoop cluster. Hadoop clusters with ten to hundreds of machines is standard. In fact, other than for development purposes, there's no reason to run Hadoop on a single server [13].

Chapter 4

Working Procedure

To work with this project, first we learnt about Java, NoSQL, Hadoop, MapReduce, HDFS. For this we have collected many articles, paper works, videos on these keywords. The first thing we have learnt is what is Hadoop, MapReduce and HDFS. MapReduce is a programming model based on Java. So we had to learn Java first. At first we have solved some basic problems in java. After that we tried the MapReduce word count problems. We have worked on Linux operating system. At first Linux was difficult to handle. But after some time it is actually very easy to work in. The best feature of Linux operating systems is its low susceptibility to virus and malware infestation. We have installed Hadoop on Ubuntu. Then we have installed JDK the java developers kit and have installed eclipse by giving some command in the terminal. Then we created a new project and added the jar files. Without jar files the MapReduce will not work. Then we have solved a problem with word count using MapReduce. It's the basic concept of learning how MapReduce works. We have solved many other problems including the one we have described in this research. Now we are going to show how did we go through the installation part.

At first we checked whether there is any JDK installed or not by giving a command in the terminal. After seeing JDK not installed we gave a command which is- *sudo apt-get install openjdk-7-jdk*. After pressing enter button the terminal asked for the password. So we simply gave the password.

After giving the password it asked for the confirmation of installing 109MB of JDK package. We needed internet connection to download and install the JDK. So after checking the internet connection we typed Y to begin the download and installation.

After installation, to confirm if the JDK installed successfully we gave a command in the terminal- *java -version*. If JDK is installed it will show the current version of java.

The second pre-requisite was ssh. So to install ssh we gave a command – *sudo apt-get install ssh*. Again it asked for confirmation. And we typed y.

After installing all the pre-requisite we downloaded the apache Hadoop. We browsed in the Google for apache mirror and there were many links. We choose the top most link and then from the apache.bytenet.in site we typed Hadoop to find the folder. After clicking the Hadoop folder, we could see three folders. We clicked the core folder and from that folder we clicked the stable1. After that we clicked Hadoop-1.2.1.tar.gz to download the file.

After download we created an own directory named shanta. Then we copied the tar file into the new directory and extract that file there. After that we entered into the folder. After that we entered the conf folder. In the conf folder there are the main files we were going to install. The core-site.xml, Hadoop-env.sh, hdfs-site.xml, mapred-site.xml, masters and slaves these are the six files we are going to install. We opened the files in the text editor. Then we edited some text and saved it.

Then we need to update the hadoop environment variables or java environment variables in the bashrc file. For that we gave a command- *gedit ~/.bashrc*. It opened up the bashrc file. Then we edited that file.

Then we typed *ssh-keygen –t dsa –p ‘’ –f ~/.ssh/id_dsa* in the terminal. Then after executing the command we typed *cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys*. After executing it we typed *ssh localhost*. It showed us the last login time.

Then to format the namenode we typed *hadoop namenode –format*. We can execute this command only one time. The namenode will be formatted. If we execute the command again it will reformat the namenode.

After that we need to start the hadoop. But at first we started the hdfs. To start that we typed *start-dfs.sh*. To confirm it has started we typed *jps*. If hdfs is running, then after executing the command it will show the datanode, namenode, jps and secondaryNamenode.

Then to start mapreduce we typed *start-mapred.sh.* After executing this command, the job tracker and task tracker will run. We *jps* to confirm all the required things are running.

To stop all the tasks, we typed *stop-dfs.sh.* Then all the programs stopped.

By executing these commands, we have set the environment to execute mapreduce. Then we started eclipse and created new java project. We right clicked on that project then clicked build path. And then clicked on libraries and then clicked the add external libraries. Then we added the jar files. After that we have solved a word

count problem in that project. We created an input text file with random sentences. Then we showed the output in the console. After that we modified the code and showed the output in an output file. We have solved many problems till now. And our final work was to calculate the maximum and minimum temperature of a specific date. We have used composite group key to send it into reducer. While we select the year, month and day it sends the individual dates as one key. The key that reducer will take as input will be a specific date then. After that it will calculate the minimum and maximum temperature of that specific date from the entire input file. So it will show only two results. The max and the min from the entire input text file.

1. This is the input files we used in this project. There are 14 files for 14 years. All the temperatures are synthetically generated.

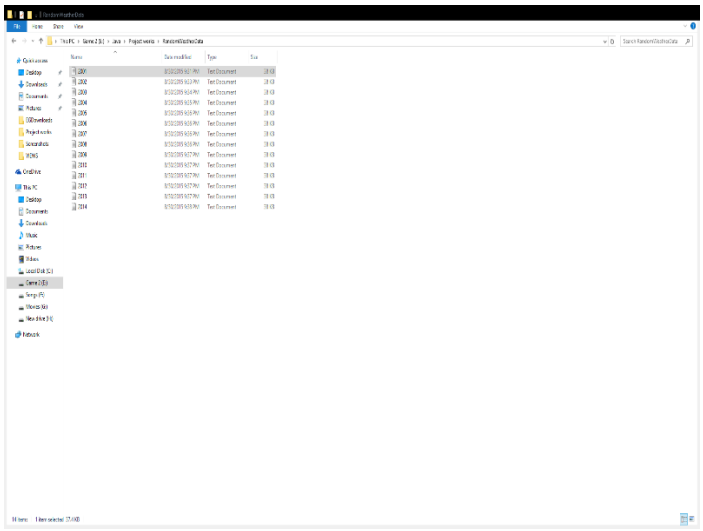

Figure 4.1: Input files used in this demo project

2. These files include year, city, month, day and temperature.

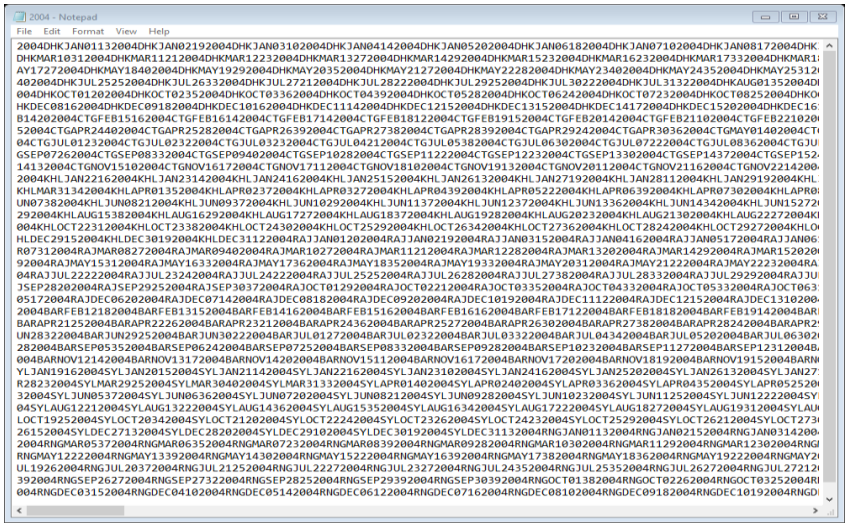

Figure 4.2: Data from year 2004

3. Then after running the program we can see a dialogue box. User will select date, month and year from the dropdown list and to see result user will select maximum temperature or minimum temperature and then press submit button.

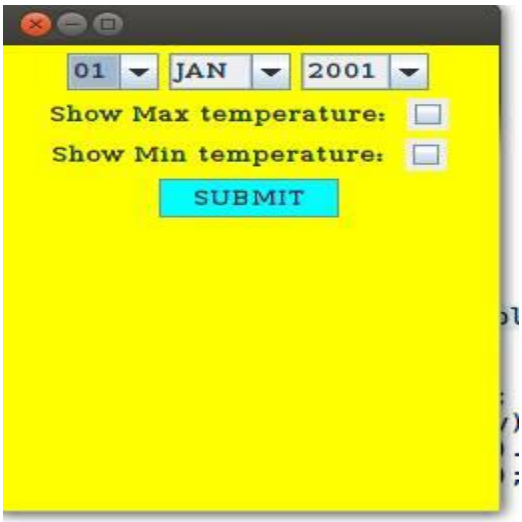

Figure 4.3: Demo project interface

4. After Pressing the Submit button user will show the directory of the file where the data is located. And user will write the name of the file where the output is going to be store.

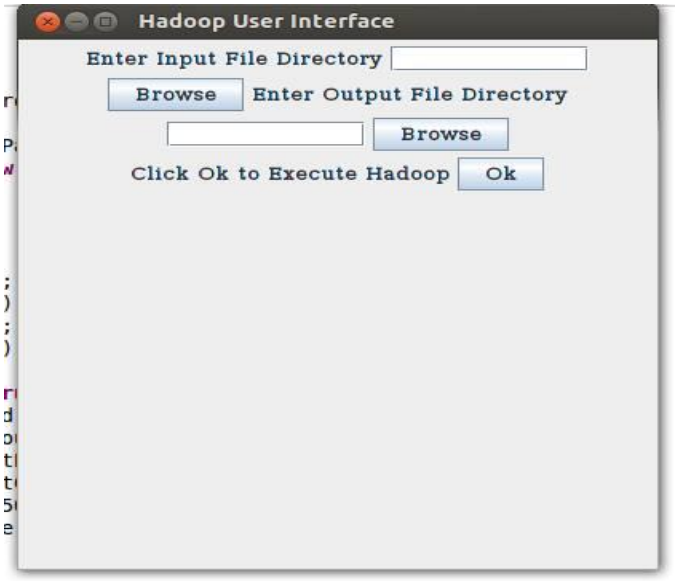

Figure 4.4 Demo project interface

5. Finally, the output will be shown in a dialogue box for the group by attributes user had chosen.

Figure 4.5 output of the demo project

Chapter 5

Conclusion & Future Work

5.1 Conclusion

In this project we have shown the comparison between SQL and NoSQL. The purpose of this work was to prove better performance of NoSQL than SQL and to investigate the scope of NoSQL approach using Hadoop environment. From this work we have learnt many things. The architecture of Hadoop MapReduce and MySQL. By implementing two java based applications we have found that all in all NoSQL's performance is better than the traditional one. Hadoop MapReduce reads data from text files so there is no problem of connection or request to the server. But if we use SQL server management system or MySQL database and try to execute any operation from an application it will first search for the connection to be build. That will kill time. These are the overheads of using SQL databases. It also consumes memory space to store data. But as NoSQL reads from a text file it doesn't need much space. And it also doesn't need to make rows or columns which is required in any SQL database. So we can say that using NoSQL database it makes life easier and time saving. For the large volume of data NoSQL is well suited. It will save our time and ease the way of handling large volume of data.

5.2 User Friendliness

In organizations they do not hire people to do just a specific task. There are many ranks from lower to upper grade. Normally lower grade officers are given the task to store data in database and execute query on them. For this they need to know about SQL. So they need to hire those who knows about SQL queries or have to

train them. From that perspective storing and analyzing data is not user friendly. But if organizations use applications then it is much easier to handle those data without knowing SQL. And it will also lessen the cost of training one employee.

5.3 Future Work

NoSQL is the most updated type of non-relational database. Developers are still working on it. This database works with unstructured data also, so it's a big advantage for many organizations who wants to store and manipulate that types of data. Many big organizations are using NoSQL database to store their data automatically. Such as Google use Bigtable. Netflix, LinkedIn and twitter also use NoSQL as their database. The use of NoSQL will increase when they will face problems using the traditional database and will find NoSQL more comfortable than the traditional ones.

5.4 References

[1] B. Tudorica and C. Bucur, "A comparison between several NoSQL databases with comments and notes," in Roedunet International Conference (RoEduNet), 2011 10th, june 2011, pp. 1 –5

[2] D. Bartholomew, "SQL vs. NoSQL," Linux Journal, no. 195, July 2010

[3] S. Tiwari, Professional NoSQL. Wiley/Wrox, August 2011

[4] www.researchgate.net https://www.researchgate.net/publication/261079289_A_performance_comparison_of_SQL_and_NoSQL_databases [Accessed: January 24, 2016].

[5] www.digitalocean.com https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models [Accessed: March 05, 2016].

[6] Springer Open

http://journalofbigdata.springeropen.com/articles/10.1186/s40537-014-0008-6 [Accessed: March 08, 2016].

[7] IEEE Xplore

http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6567202&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6567202 [Accessed: March 08, 2016].

[8] http://www.christofstrauch.de/nosqldbs.pdf [Accessed: March 08, 2016].

[9] sas.com

http://www.sas.com/en_my/insights/big-data/hadoop.html [Accessed: April 02, 2016].

[10] www.ijarcsse.com/docs/papers/8_August2012/...2.../V2I800154.pdf

[11] Digital Ocean https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models [Accessed: March 17, 2016].

[12] Safari

https://www.safaribooksonline.com/library/view/high-performance-mysql/9781449332471/ch01.html [Accessed: March 18, 2016].

[13] Google

https://sites.google.com/site/hadoopintroduction/home/comparing-sql-databases-and-hadoop [Accessed: March 20, 2016].

[14] Hadoop

https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html [Accessed: March 25, 2016].

[15] MAPR https://www.mapr.com/products/apache-hadoop [Accessed: March 29, 2016].

[16] IBM https://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/ [Accessed: March 24, 2016].

Appendix:

Main.java

```
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;
import javax.swing.border.LineBorder;



public class Main extends JApplet{

        String month[]={"JAN","FEB","MAR"};
```

```
        JButton button = new JButton("SUBMIT");
        JFrame frame = new JFrame();

        JComboBox d = new JComboBox();
        JComboBox m = new JComboBox();
        JComboBox y = new JComboBox();
        JCheckBox maxcheck = new JCheckBox();
        JCheckBox mincheck = new JCheckBox();

        JLabel showmax = new JLabel("Show Max temperature: ");
        JLabel showmin = new JLabel("Show Min temperature: ");

        String day,mon,year;

        Container ct = new Container();

public static void main(String[] args){
         Main obj = new Main();
         obj.combo();

}

public void combo()
        {

                for(int i=1;i<=31;i++)
                {
                        if(i<=9)d.addItem("0"+String.valueOf(i));
                        else d.addItem(String.valueOf(i));
                }
                for(int i=0;i<3;i++)m.addItem(month[i]);

                for(int i=2001;i<=2016;i++)y.addItem(String.valueOf(i));
```

```
day = d.getSelectedItem().toString();
        d.addActionListener(new ActionListener()
        {

                public void actionPerformed(ActionEvent ae)
                {
                        day = d.getSelectedItem().toString();
                }
        });


        mon = m.getSelectedItem().toString();
        m.addActionListener(new ActionListener()
        {

                public void actionPerformed(ActionEvent ae)
                {
                        mon = m.getSelectedItem().toString();
                }
        });


        year = y.getSelectedItem().toString();
        y.addActionListener(new ActionListener()
        {

                public void actionPerformed(ActionEvent ae)
                {
                        year = y.getSelectedItem().toString();
                }
        });


        maxcheck.addItemListener(new ItemListener() {
```

```
		@Override
		public void itemStateChanged(ItemEvent e)
		{
			// TODO Auto-generated method stub
			if(e.getStateChange()==1){
				Constants.maxchecked = true;
			}
			else {
				Constants.maxchecked = false;
			}
		}
	});

	mincheck.addItemListener(new ItemListener() {

		@Override
		public void itemStateChanged(ItemEvent e) {
			// TODO Auto-generated method stub
			if(e.getStateChange()==1){
				Constants.minchecked = true;
			}
			else {
				Constants.minchecked = false;
			}
		}
	});



button.addActionListener(new ActionListener()
{
  public void actionPerformed(ActionEvent ae)
  {
			try
			{
```

```
                    MaxTemperatureMapper mtm= new MaxTemperatureMapper();
                        BufferedWriter writer = new BufferedWriter(new
                    FileWriter("/home/shanta/Documents/Data/data.txt"));
                        writer.write(year+" "+mon+" "+day+"\r\n");

                        writer.close();


                        Runnable a = new Event();
                        Thread t = new Thread(a);
                        t.start();

                    }
                    catch (IOException e)
                    {

                        e.printStackTrace();
                    }
  }
});


        button.setBackground(Color.CYAN);

        ct = getContentPane();
        ct.setLayout(new FlowLayout());
        ct.add(d);
        ct.add(m);
        ct.add(y);

        ct.add(showmax);
        ct.add(maxcheck);
        ct.add(showmin);
        ct.add(mincheck);
```

```
                ct.add(button);
                ct.setVisible(true);
                ct.setBackground(Color.YELLOW);
                frame.setBackground(Color.BLUE);
                frame.getContentPane().add(ct);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setSize(250, 300);
                frame.setVisible(true);
        }
}
```

Event.java

```
import javax.swing.*;
import java.awt.Component;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
public class Event extends JFrame implements Runnable
{
        private JTextField txtPath;
        private JTextField txtPath2;
        JButton button1;
        JButton button2;
        JButton button3;
        JLabel label1;
        JLabel label2;
        JLabel label3;
        static String args0;
        static String args1;
```

```
public static void main(String[] args){

        new Event();
        }

public void Event(){

        this.setSize(400, 400);


        this.setLocationRelativeTo(null);

        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        this.setTitle("Hadoop User Interface");

        JPanel thePanel = new JPanel();

JLabel label1=new JLabel("Enter Input File Directory");
thePanel.add(label1);


        txtPath = new JTextField();
        txtPath.setBounds(10, 10, 414, 21);
        thePanel.add(txtPath);
        txtPath.setColumns(10);

        JButton btnBrowse = new JButton("Browse");
        btnBrowse.setBounds(10,41,87,23);
        thePanel.add(btnBrowse);

        btnBrowse.addActionListener(new ActionListener() {
```

```
                public void actionPerformed(ActionEvent e) {
                        JFileChooser fileChooser = new JFileChooser();

                        fileChooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);

                        Component parent = null;

                        int rVal = fileChooser.showOpenDialog(parent);
                        if (rVal == JFileChooser.APPROVE_OPTION) {
                                 args0 = fileChooser.getSelectedFile().getAbsolutePath();
                                System.out.println("one twoooo....."+args0);
                                  System.out.println("Please Choose Your Input Directory: " + args0);
                                  System.out.println("one....."+args0);
                        }else {
                    System.out.println("No Selection ");
                   }

                        System.out.println("one....." + args0);

                                txtPath.setText(fileChooser.getSelectedFile().toString());
                        }
                }

            );

JLabel label2=new JLabel("Enter Output File Directory");
thePanel.add(label2);
                txtPath2 = new JTextField();
                txtPath2.setBounds(200, 20, 514, 221);
                thePanel.add(txtPath2);
                txtPath2.setColumns(10);

                JButton btnBrowse2 = new JButton("Browse");
                btnBrowse2.setBounds(10,41,87,23);
                thePanel.add(btnBrowse2);
```

```
btnBrowse2.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		JFileChooser fileChooser = new JFileChooser();

		fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
		Component parent1 = null;
		int rVal1 = fileChooser.showOpenDialog(parent1);
		if (rVal1 == JFileChooser.APPROVE_OPTION) {

		args1 = fileChooser.getSelectedFile().getAbsolutePath();

		BufferedWriter writer;
		try {
			writer = new BufferedWriter(new
FileWriter("/home/shanta/Documents/Data/finalOutputDir.txt"));
			writer.write(args1);
			writer.close();
		} catch (IOException e1) {
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}

		System.out.println("Please Choose Your Output Directory: " + args1);
		} else {
	  System.out.println("No Selection ");
	 }

		System.out.println("two....."+args1);
		txtPath2.setText(fileChooser.getSelectedFile().toString());
		}

} );
```

```
JLabel label4=new JLabel("Click Ok to Execute Hadoop");
thePanel.add(label4);
                button1 = new JButton("Ok");

                ListenForButton lForButton = new ListenForButton();

                button1.addActionListener(lForButton);

                thePanel.add(button1);

                this.add(thePanel);

                this.setVisible(true);

        }

        private class ListenForButton implements ActionListener{


                public void actionPerformed(ActionEvent e){

                        if(e.getSource() == button1){

                                MaxTemperature mt = new MaxTemperature();
                                try {

                                        mt.exec(args0, args1);

                                } catch (Exception e1) {
                                        e1.printStackTrace();
                                }

                        }
```

```
                }
        }

        @Override
        public void run() {

                // TODO Auto-generated method stub
                Event();
        }
}
```

MaxTemperature.java

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxTemperature
{

        public void exec(String args0, String args1) throws Exception
    {
                                                        System.out.println(args0);
                                                        System.out.println(args1);

            if (args0 == null && args1 == null)
             {
                System.err.println("Please Enter the input and output parameters");
                System.exit(-1);
             }
```

```
        Configuration conf = new Configuration();

        Job job = new Job(conf, "min,max and average");

        job.setJarByClass(MaxTemperature.class);
        job.setJobName("Max temperature");

        FileInputFormat.addInputPath(job,new Path(args0));
        FileOutputFormat.setOutputPath(job,new Path (args1));

        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);

        job.setMapOutputKeyClass(CompositeGroupKey.class);
        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(CompositeGroupKey.class);
        job.setOutputValueClass(Reducer.class);


        System.exit(job.waitForCompletion(true)?0:1);
    }
}
```

MaxTemperatureMapper.java

```
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class MaxTemperatureMapper extends Mapper <LongWritable, Text, CompositeGroupKey, Text>
{
```

```
int year;
String mnth;
int date;

List<Integer> n = new ArrayList<Integer>();
List<String> m = new ArrayList<String>();
List<Integer> d = new ArrayList<Integer>();

public void map(LongWritable key, Text value, Context context) throws IOException,  InterruptedException
{
        BufferedReader br = null;

        try
        {
                String sCurrentLine;

                br = new BufferedReader(new FileReader("/home/shanta/Documents/Data/data.txt"));

                while ((sCurrentLine = br.readLine()) != null)
                {
                        String line = sCurrentLine.toString();
                        String[] arr=line.split(" ");
                        year = Integer.parseInt(arr[0]);
                        mnth = arr[1];
                        date = Integer.parseInt(arr[2]);

                        n.add(year);
                        m.add(mnth);
                        d.add(date);
                }

                                }
```

```
                                catch (IOException e)
                                {
                                        e.printStackTrace();
                                }

                                String line = value.toString();
        int year1 = Integer.parseInt(line.substring(0,4));
                                String mnth1 = line.substring(7,10);
         int date1 = Integer.parseInt(line.substring(10,12));

          int temp= Integer.parseInt(line.substring(12,14));
          String city = line.substring(4,7);
\

          if(year1==year && mnth.equals(mnth1) && date==date1)
          {
          CompositeGroupKey cntry = new CompositeGroupKey(year1,mnth1,date1);


                                        context.write(cntry, new Text(city + "_" + temp));
          }

                }
}
```

MaxTemperatureReducer.java

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;
import java.io.IOException;

import javax.swing.JFrame;

public class MaxTemperatureReducer extends Reducer <CompositeGroupKey, Text, CompositeGroupKey, Text >{

        public void reduce(CompositeGroupKey key, Iterable<Text> values, Context context) throws IOException,InterruptedException
```

```
{
    long max = Long.MIN_VALUE;
    long min = Long.MAX_VALUE;

Text maxCity = null;
Text minCity = null;
System.out.println(maxCity);

for(Text value : values){

        String compositeString = value.toString();
        System.out.println(compositeString);
        String[] compositeStringArray = compositeString.split("_");
        System.out.println(compositeStringArray);

        Text tempCity = new Text(compositeStringArray[0]);
        System.out.println("Temperature of City "+tempCity);
        long tempValue = new Long(compositeStringArray[1]).longValue();
        System.out.println("Temperature Value "+tempValue);

        //TODO

        if(tempValue > max){
                max = tempValue;
                System.out.println("max is"+max);
                maxCity = tempCity;
                System.out.println("maxcity: "+maxCity);
        }
        if(min > tempValue){
                        min = tempValue;
                System.out.println("min is "+min);
                minCity = tempCity;
                System.out.println("mincity: "+minCity);
        }
}
```

```
                        System.out.println("max "+max);
                        String keyText = new String(" max" + "(" + maxCity.toString() + "):" + max + " min" + "(" + minCity.toString() + "):" +
min);

                        if(Constants.maxchecked && Constants.minchecked)keyText+=" *";
                        else if(Constants.maxchecked)keyText+=" +";
                        else if(Constants.minchecked)keyText+=" -";
                        else keyText+=" ";

                        System.out.println("value is"+ keyText);
                        context.write(key,new Text(keyText));
    }


}
```

finalOutput.java

```
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

import javax.swing.JApplet;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingConstants;


public class finalOutput extends JApplet{

        public static String dateline=null;
        public static String maxtempline=null;
        public static String mintempline=null;
```

```
public static String flag=null;
public void swing()
{
		JFrame frame = new JFrame("Output");
		Container ct = new Container();
		JLabel first = new JLabel();
		first.setText(dateline);
		JLabel second = new JLabel();
		second.setText("Maximum temperature : "+maxtempline);
		JLabel third = new JLabel();
		third.setText("Minimum temperature : "+mintempline);
		ct = getContentPane();
		ct.setLayout(new FlowLayout());
		ct.add(first);
		if(flag.equals("+"))ct.add(second);
		else if(flag.equals("-"))ct.add(third);
		else if(flag.equals("*")){
			ct.add(second);
			ct.add(third);
		}

		ct.setVisible(true);
		ct.setBackground(Color.YELLOW);
		frame.getContentPane().add(ct);
		frame.setSize(300, 300);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setVisible(true);
}
public static void main(String[] args){

		BufferedReader br = null;

		try
		{
```

```
                String sCurrentLine=null;
                String outputdir=null;

                br = new BufferedReader(new FileReader("/home/shanta/Documents/Data/finalOutputDir.txt"));

                while ((sCurrentLine = br.readLine()) != null)
                {
                        outputdir = sCurrentLine.toString();
                }
                br.close();

                sCurrentLine=null;
                br = new BufferedReader(new FileReader(outputdir+"/part-r-00000"));
                while ((sCurrentLine = br.readLine()) != null)
                {
                        String[] arr = sCurrentLine.split(" ");
                        dateline = arr[0];
                        maxtempline = arr[1];
                        mintempline = arr[2];
                        if(sCurrentLine.contains("+"))flag="+";
                        else if(sCurrentLine.contains("-"))flag="-";
                        else if(sCurrentLine.contains("*"))flag="*";
                }
                br.close();
        }
        catch (IOException e)
        {
                e.printStackTrace();
        }

        finalOutput output = new finalOutput();
        output.swing();
    }
}
```