Weather Data Analysis using Hadoop Map-Reduce

(Aradhya Tulsyan, Aakash Bhambhani, Prateek Khamesra, Shalaka Patel)

Abstract

Weather analysis generates a large amount of information as data is gathered across various domains such as temperature, pressure, humidity, sea-level, wind speeds etc. and at the same time, for statistical concreteness, data is gathered redundantly across a lot of locations, several times everyday. This data is generated for thousands of locations across the globe. As a result, analysing this information to give an efficient trend change globally, or even locally, requires processing of such substantial datasets. This project involves the processing of publicly available weather data to generate trends of weather change for multiple locations and various durations of time to propound some beneficial analysis patterns. To do so, superior computing power is required and we have chosen Hadoop's MapReduce as our parallel processing tool to processes this enormous amount of data with high efficiency.

Background & Motivation

The primary motivation of this project is to develop a tool that shows the changes in the climate for any particular location. It should be possible to generate these results and trends arbitrarily for any location, any duration (within the limits of the gathered data) in a form that is comparison ready so that useful conclusions can be derived with ease. For e.g., over the last 2 decades that, we as a society have been made aware of the fact that our actions cause observable changes to the Earth's biosphere, the general rise in temperatures and erratic climatic changes being the patent proof. The project involves the processing of freely available data to generate graphs and comparisons which we think could be used to clearly show the global changes. The National Climatic Data Center (NCDC) has been gathering global data which is publicly accessible. The summaries of 9000+ locations for a period of 100 years is available which contains the location, year-month-date, average temperature, dew-point, pressure, average wind speed, max wind speed, max temperature, min temperature, visibility and precipitation for each day. This dataset is known as the Global Summary of the Day (GSOD) and contains weather measurements from 1929 - 2014. Since the data is in a consistent format, it will allow us to run Map-Reduce jobs on certain keys efficiently with minimal preprocessing.

Data Analysis

Pre-processing

The data downloaded from NCDC is in the following format; every year has files for different stations for which weather data was recorded. Each file had 365 records i.e. for each day for a particular station a single file exists and this file has 365 entries. Since there are more than 9000 stations on record, there is a sizable heap of files. Our initial test run with these many files as input resulted in a very slow execution of the map-reduce job. As a result, we decided to pre-process the data and combine the files for each year. Therefore instead of thousands of files every year, we will just have one file for each year. This speeds up the I/O from HDFS because it now has to deal with considerably less number of files, thus reducing random access and increasing continuous access.

To accomplish this, we wrote a python script which takes the raw zipped folders of data and processes all the files in the folder, extracts the files for each year and stations and finally concatenates the set of data into one single file for each year. The script also takes care of unwanted characters and lines in the input dataset like headers and semicolons etc. Therefore, our final map-reduce program doesn’t have to deal with cleaning the input data. After pre-processing, we end up with 19GB of clean data ready to be analyzed.

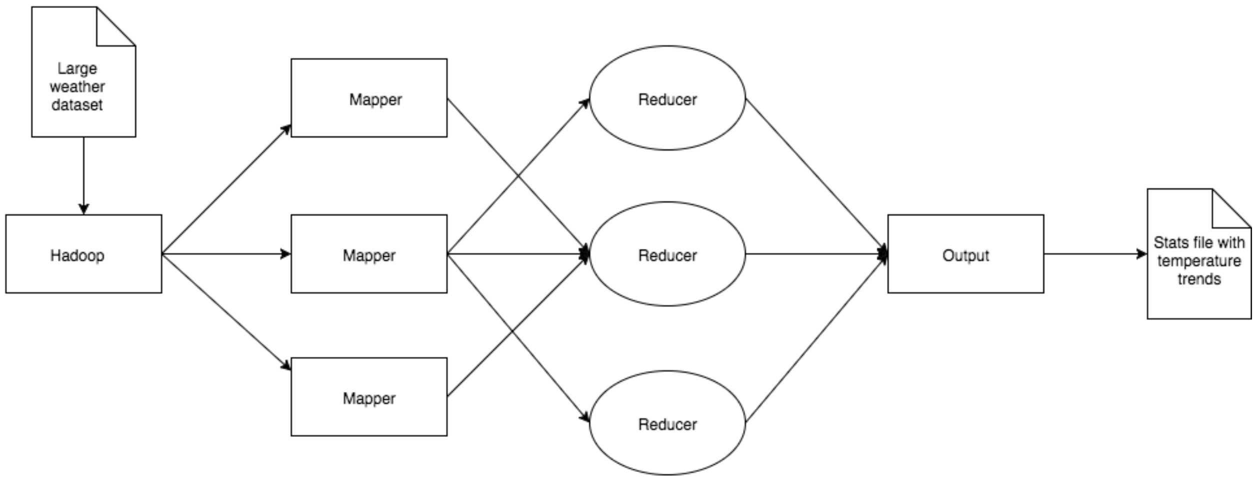
Processing

Each file in the NCDC dataset contains the weather measurements for parameters such as temperature, humidity, precipitation, wind-speed etc. for each day in a particular year. As a demonstration we have picked average temperature as the quantity we wish to analyze and we wish to generate a yearly aggregate of the daily average temperature. In other words we wish to have one entry per year per location which provides the temperature. A detailed explanation of this approach is provided in the implementation section.

Framework-Hadoop Map Reduce

Hadoop MapReduce is a framework using which we can write applications to process humongous amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a processing technique and a programming model for distributed computing based on java. The major advantage is that it is easy to scale data processing over multiple computing nodes.

The MapReduce algorithm consists of two important phases, namely Map and Reduce. The Map phase takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce phase takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the name MapReduce implies, the reduce task is always performed after the map job.



Architecture

Implementation

The implementation is deceptively simple but makes use of some powerful concepts. The main class weather behaves as a container class and a point of entry for the execution. It contains 2 classes TokenizerMapper and SumReducer which represent the Mapper and Reducer respectively (as can be seen from their parent classes). The Mapper implementation receives single lines of input when set up with the default Hadoop configuration. These lines are then trimmed and split into strings which allows easy access to individual features. The Station ID, year is the key and the average temperature is the value that will be written by the Mapper and sent to the shuffle phase.

The Reducer receives a set of values for each unique key (365 values for every key to be more concise), for which it generates an average and then prints the average as the result. In this way, each reducer sends out a single entry which corresponds to the year's average temperature.

Result and Evaluation

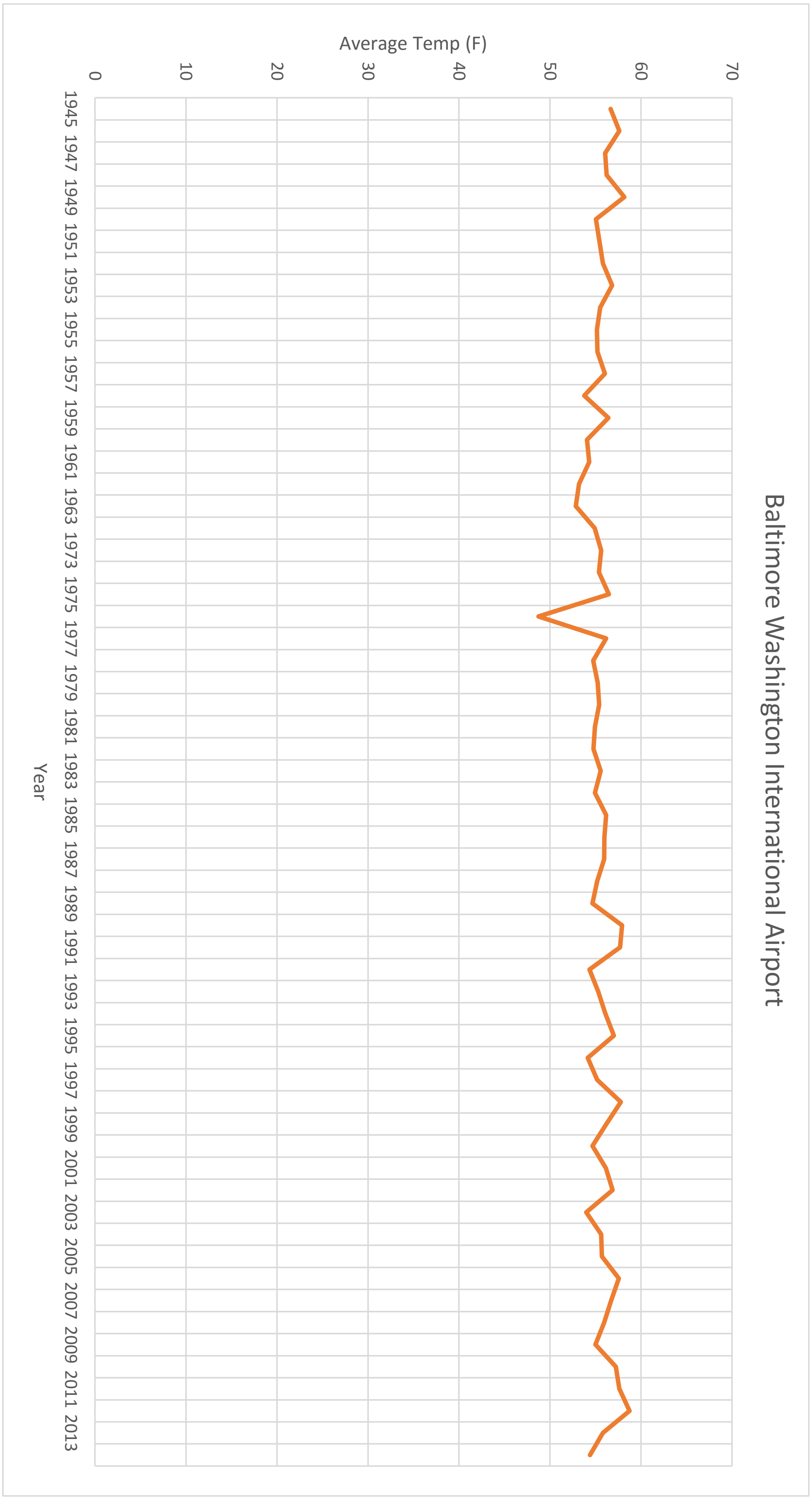
So far we have discussed the implementation of the framework and how it works with the data we have obtained. Following is a sample of the result we obtained.

724064 99999	1997	53.81624
724064 99999	1998	55.80994
724064 99999	1999	54.05781
724064 99999	2000	50.52547
724065 93733	1959	39.92449
724065 93733	1960	53.49071
724065 93733	1961	53.84712
724065 93733	1962	52.29452
724065 93733	1963	52.33452
724065 93733	1964	53.29645
724065 93733	1965	52.71973
724065 93733	1966	53.55973
724065 93733	1967	54.57973
724065 93733	1968	57.70955
724065 93733	1969	57.18877
724065 93733	1970	57.90301
724065 93733	1971	50.13807
724065 93733	1973	58.86768
724065 93733	1974	59.15897
724065 93733	1975	64.59927
724065 93733	1976	57.01307
724065 93733	1977	58.4779
724065 93733	1978	57.19091
724065 93733	1979	57.60885
724065 93733	1980	58.84039
724065 93733	1981	57.80071
724065 93733	1982	58.26293
724065 93733	1983	58.80101
724065 93733	1984	58.12047
724065 93733	1985	59.42053

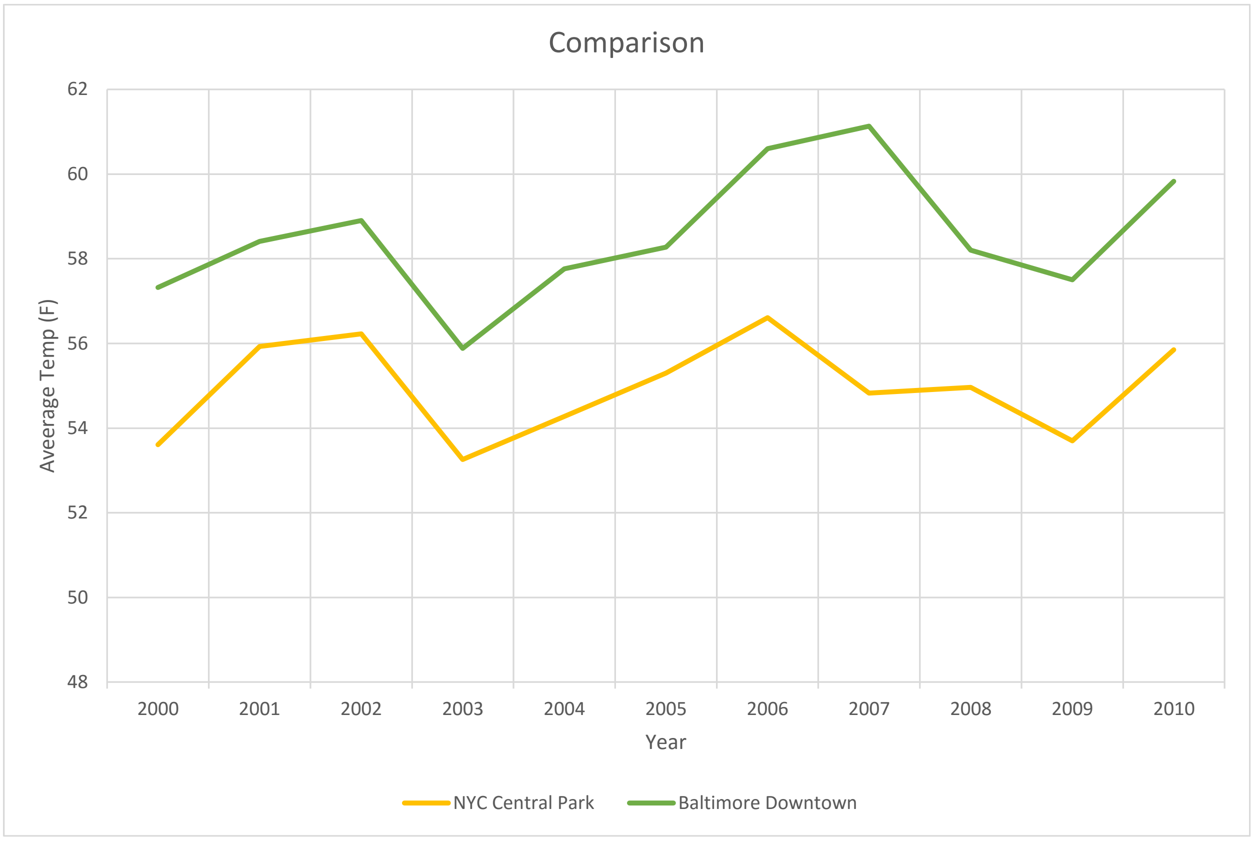
724066 99999	1985	55.86027
724066 99999	1986	55.79014
724066 99999	1987	56.29918
724066 99999	1988	55.92268
724066 99999	1989	56.05123
724066 99999	1990	58.71616
724066 99999	1991	58.73205
724066 99999	1992	54.99153
724066 99999	1993	56.92192
724066 99999	1994	56.8526
724066 99999	1995	56.30446
724066 99999	1996	53.62301
724066 99999	1997	54.8411
724066 99999	1998	57.76813
724066 99999	1999	54.20495
724066 99999	2000	52.52082
724066 99999	2001	54.67652
724066 99999	2002	55.12658
724066 99999	2003	52.07644
724066 99999	2004	53.65464
724066 99999	2005	54.03508
724067 93744	2006	59.43205
724067 93744	2007	57.92438
724067 93744	2008	57.16721
724067 93744	2009	55.68306
724067 93744	2010	57.62849
724067 93744	2011	57.61154
724067 93744	2012	59.12077
724067 93744	2013	56.23863
724067 93744	2014	55.12637

The first column represents the station ID, the second column represents the year and the third column the corresponding average temperature.

A very clean visual representation can be obtained by charting the data. For the station at BWI Airport we obtain the following chart. Note that the historic low achieved in the year of 1976 is clearly visible.


Baltimore Washington International Airport
Average Temp (F)
0
10
20
30
40
50
60
70
Year
1945
1947
1949
1951
1953
1955
1957
1959
1961
1963
1973
1975
1977
1979
1981
1983
1985
1987
1989
1991
1993
1995
1997
1999
2001
2003
2005
2007
2009
2011
2013

Comparisons are also made very intuitively when data is presented in such a graphical manner.



Conclusion

The 2 graphs shown above were easily generated and were just a sample of the possible potential of using MapReduce to analyze weather data. Once the dataset has been processed, it can provide trends in a particular format for any location and any duration as needed by the user. For any new trends (precipitation instead of temperature or a monthly average instead of a yearly average) the processing needs to be done again to generate results but once done it can be stored for later use.

MapReduce has been demonstrated to be a powerful tool for large scale data analysis. It has its own set of flaws; High volume of disk I/O in this case but is a good framework if used correctly (fixing data access trends).

Future Work

Taking the data on current weather, climate and atmospheric conditions, we can use machine learning algorithm to help forecast the weather for the coming days.

References

[1] - https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[2] - https://hadoop.apache.org/

[3] - http://www7.ncdc.noaa.gov/CDO/cdoselect.cmd?datasetabbv=GSOD