Big Data Engineering and Analytics

Project 1 NoSQL Proof of Concept

Storage Warriors

Abhinay Agrawal

Ambuj Nayan

Nitilaksha Halakatti

Rahul R Sharma

Analyzing weather, flight data, and their correlation

**Introduction**

The goal of this project was to build an application which could ingest, store, analyze, and extract meaningful insights from two different massive data stores. The first of these sources was NOAA (National Oceanic and Atmospheric Administration) and it provided us with hourly synoptic weather observations from station networks around the world. The second data source was UBTS (US Bureau of Transportation Services) and it provided us with flight history and delays.

During preliminary data analysis, we found that the volume of data was very huge. The respective sizes of weather and flight datasets were 750 GB and 225 GB approximately. The huge data volume and the NoSQL DB use case pushed us towards building a scalable and distributed system to store and process this data.

We decided to use Apache HBase which is a database associated with the Hadoop ecosystem. It is a distributed and scalable column family data store. The datasets in their raw form were not conducive for analysis and needed considerable amount of pre-processing. Custom python scripts were used to pre-process the data. After completion of pre-processing, we needed a scalable and distributed process which could perform bulk upload to HBase. Apache Spark was a good fit here because of its unique in-memory processing capabilities which allow it to process large-scale data at very high speeds.

Per our use case, this application had to provide ease of access to its users. Since most of the users in the current world are already familiar with SQL and SQL reduces the amount of code that needs to be written, we decided to integrate HBase with Apache Phoenix. Phoenix is an open source SQL skin for HBase and saves us the hassle of writing application code to query data using HBase APIs. Integrating Apache Phoenix with HBase, and using it to query huge volume of data from HBase had its fair share of problems which are described in greater detail later in this document. For visualization, we selected Apache Zeppelin. It is a web-based notebook application that enables data-driven, interactive data analytics, and collaboration with SQL.

One of the biggest challenges of this project was to find correlation between weather and flight data. These two datasets had different granularities and finding a spatiotemporal relationship between these two proved to be the trickiest problem. Since flight datasets did not have latitude and longitude information, we had to mine the internet to get the coordinates of all US airports. The next step was to find the weather station closest to each airport while ensuring that the station was close enough to have an impact on airport operations. This was achieved by a Java program which calculated the geographical distance between an airport and all the weather stations and picked the weather station closest to the airport. We also had to make sure that the observed weather phenomenon was in the same time window as the flight operation.

Once the relationship between weather and flight was established, we used machine learning to predict whether forecasted weather information will have an impact on flight schedules.

**Infrastructure setup**

*Hardware used for POC:*

Since this project involved downloading and processing huge data files, we selected a computer with the following specification:

Hard-disk: 1 TB (approximately 800 GB free)

Memory: 8 GB

Processor: Intel i7 quad core

Operating system: Ubuntu 16.04

*Hadoop ecosystem setup*:

Instead of using any cloud service provider, we decided to invest time in learning the setup and configuration of Hadoop ecosystem. Also, most of the cloud service providers had high tariff for our storage requirements. Following are the steps:

Installation of Hadoop involved downloading the distribution from [Apache](http://www.apache.org/dyn/closer.cgi/hadoop/common/) and modifying the configuration files as mentioned [here](http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html#Pseudo-Distributed_Operation).

We downloaded HBase from [Apache](http://www.apache.org/dyn/closer.cgi/hbase/) and configured it as per the guidelines provided [here](https://hbase.apache.org/book.html#quickstart).

Upon the completion of installation and configuration, the services related to DFS, YARN, and HBASE were started.

*Apache Phoenix setup:*

Downloaded the installation [tar](https://phoenix.apache.org/download.html) and copied the phoenix server jar into HBase lib directory.

Restarted HBase and after successful restart, we started the query server.

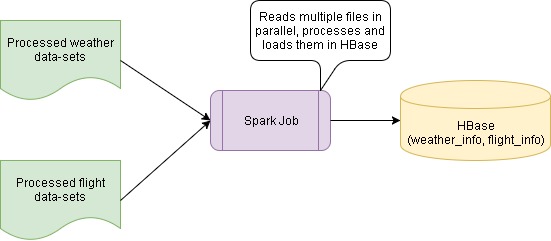
*Apache Zeppelin setup:*

Followed the steps at this [site](https://zeppelin.apache.org/docs/0.7.3/install/install.html) to download and install Zeppelin.

Configured Zeppelin as a thin client and used a JDBC connection to connect it to Apache Phoenix.

**Big Data Lifecycle of this project**

**Data pipeline for injection into HBase**



Language used for Spark job: Java

Number of processor cores used: 4

Default parallelism for spark job: 8

Serialized used: KyroSerializer