Scalable Computing Project Report -Group 4

Water demand forecasting based on weather forecast

Jarvin Mutatiina (S3555631)
Chriss Santi (S3795748)
Hilary Oodo (S3878708)

March 31, 2019

1 Introduction

Water demand forecasting is essential in predicting timely demand of water in a given area using weather forecast model. Our software would be capable of predicting the quantity of water that can be on demand automatically in a particular zone with minimal error record. Water demand forecasting is one of the methodologies used in management of water demands and supplies of a given area. Why should there be need for water demand forecasting? Water demand forecast is important in the area of determining how weather can influence the usage of water so that government or whosoever is in charge of water can take proactive measures in ensuring that there is no shortfall in water supply chain. In the Netherlands for instance, in 2012, 57percent of all supplied water is controlled based on a short term water demand forecast (Bakker et al., 2013a).
Water demand forcasting is basically focused on evaluating the existing providers of water resources like the private providers, the municipality, states or federal government. By doing so, it can then determine if there is need for improvement or adjustment. It can also makes recommendation if there is need for improvement of future systems. However, there are records of erroneous water demand forcasting model. So, our model is to improve on the existing models in ensuring that records of error are minimal and the software is scalable.

The application is built to demonstrate high performance and scalable forcasting of water demand using different technologies like apache spark, Cassandra, Kafka, Hadoop HDFS and other distributed data processing services. Docker technology is used in the deployment of different applications used for this project in local and cloud by running multiple containers and providing a backend database. The forecasting model used in implementation of this task is decision tree regression.

2 Methodology

2.1 Batch data

The data used in implementing this project was sourced from Royal Netherlands Meteorological Institute (KNMI). KNMI is the Dutch weather data centre. The dataset generated was water usage column based on the temperature column that exist at KNMI dataset. The results of what we got at KNMI was used as an input data to access the accuracy of water demand forecasting prediction model.The process involves generating the water data based on the temperature distribution in the KNMI dataset. The final batch data contains data attributes of; Year, temperature, Hour of the day, Wind direction (in degrees),Hourly average wind speed , Duration of sunshine, Air pressure, Horizontal visibility during observation and Relative humidity

2.2 Prediction Algorithm

We are using the Decision Tree Regressor algorithm from the Spark ML library to develop a model based on a given set of features - Wind direction, Hourly average wind speed, Temperature, Duration of sunshine, Air pressure and Relative humidity and the target attribute- gallons of water used. The choice of decision tree regressor is in order to use its attributes of breaking down dataset into small-scale subsets and that is better option for the batch data we are using.

2.3 Batch processing

The batch processor includes importing data from our online repository, writing it to HDFS storage. The predictive algorithm - Decision Tree Regressor is later trained with this data and the resulting model is stored in HDFS as well. Storing the model enables us to reuse the model for predicting the water for the streaming data.

2.4 Streaming

Streaming is done with Kafka and java producer. The producer sequentially picks data from a data repository, and broadcasts it to Kafka container. The Kafka consumer that resides on top of Apache Spark, receives this data by making a connection to the same running Kafka container.

3 Infrastructure

The process of implementing this project involves deployment of several technologies to ensure a complete, scalable running system. The performance and scalability of this application is well arranged to address some of the lapses that are experienced in the existing systems such as unrelaibility and predictive uncertainty. That is why we chose the Decision Tree Regressor algorithm in making our prediction to ensuring that all the factors such as temperature, wind direction, wind speed etc are all incorporated . For the diagram of the architecture and data flow, kindly refer to figure 1. However, here are the list of components used in the implementation of this project;

- Spark (Master)
- Spark (Worker)
- Hadoop Namenode
- Hadoop datanode
- Hadoop Distributed File System (HDFS)
- Apache Kafka
- Cassandra
- Flask
- Zookeeper
- Docker

The first process was to source for dataset which was generated using KNMI water usage column based on the temperature column structure. The generated batch data was then stored in the datanodes of HDFS. Then apache spark was used for the processing and distribution of the batch data across the streams using python programming language. Also kafka and java producers play role of messaging the spark streaming consumer whenever there is an input data.The next process is the algorithm stage which entails training the decision tree regressor so that it can predict the water usage as a result of weather variation and the computed model is stored in the hadoop HDFS. Next stage, we tried to fetch the streaming data using java script and the generated data is saved in the Cassandra database. Then, another thing we did was to predict the water consumption using the model and

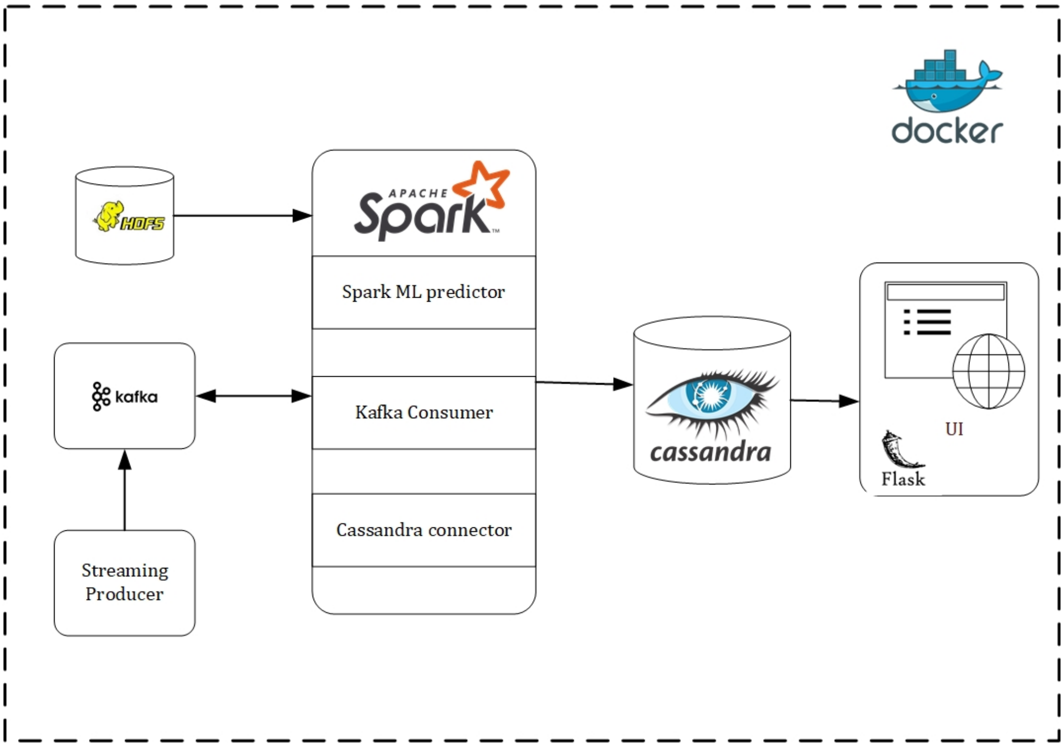

Figure 1: Architecture of the system

the streamed data. The prediction is achieved by reading the batch data model in hadoop HDFS and then computing it with spark. Finally the computed model is saved in Cassandra database. The entire processes need to be visualized. Hence, flask and HTML are used to put the application on the browser. At this point, end users can view it locally through user interface (UI) using Hue browser. All these processes were computed in docker containers. Docker plays major role in smooth deployment and running of most of these services. The technologies used for the implementation of this project are defined individually from section 3.1.Below, we have an architecture diagram representing the entire processes involved in the implementation of this project.

3.1 Apache Spark

Apache spark is a multi purpose distributed data processing machine.In implementing this application, spark master and two workers were used. The choice of using spark is to ensure scalability and to achieve high data processing tasks of the system.Spark was used in local mode for the development of the system and the programming language that was used to implement this is python. Then for the deployment, the cluster manager of spark that was used is spark standalone comprising of master and worker nodes. The choice of standalone mode is because of its flexibility with hadoop and ease to use attribute. When the job is being sent to the master which is the spark driver, the master would distribute it across the workers and then after processes are achieved, the master fetches the result from workers. The job of the cluster manager is to get instructions from the master for the resources that will be launched to the executors.

3.2 Hadoop Distributed File System (HDFS)

HDFS is fault tolerant file system designed specifically to run on the commodity hardware. For the implementation of this application, HDFS was used to store and manage the batch data across the distributed clusters. Also, it was used to store the computed model (algorithm) in the executor's in-memory of the apache spark. The choice of hadoop at this point is to ensure scalability which HDFS functions in different server machines.

3.3 Cassandra

Apache Cassandra is a well known peer to peer database which runs on cluster of uniform nodes. The choice of using Cassandra for database is in respect to its established functions for linear scalability, fault tolerance on commodity hardware or cloud infrastructure and decentralization. Secondly, it is easier to write spark's RDDs to Cassandra table. Connecting spark to cassandra using pyspark in a standalone python script was also a better way of wrapping pyspark libraries to the cassandra.

3.4 Flask

Flask is one of the python programming language's frameworks. It has functions of using other application components along with flask's features. For this project, Flask is used in the synchronization of the libraries for application of Web Service Gateway Interface. In addition to flask, Html is also used to provide the link to the browser.

3.5 Kafka

Apache kafka is a distributed publish-subscribe messaging system. The role of kafka here as a streaming platform is to convey messages across the streams of data for processing. It can also permit storage and processing of batch data in its platform.So, the choice of using kafka for this system is because, writing the Cassandra source connector into kafka through configuration file would allow saved data to easily return to an event stream.

3.6 Zookeeper

Apache zookeeper is one of the services used in this system in providing group services and distributed synchronization. The choice of Zookeeper is because of its open source server which enables smooth interaction with other distributed applications.

3.7 Docker Containerization

Docker provides platform for the development, deployment and smooth running of applications through its containers and images. For this project, docker containers are used to develop and deploy various technologies used in the building of this application. There are about eleven services (Containers) used in this system, each of them domiciled in docker. The docker containers are networked in such that data can be shared within the containers. In figure.2, we try to illustrate graphically how docker is used in the deployment of different services used in the implementation of this software.

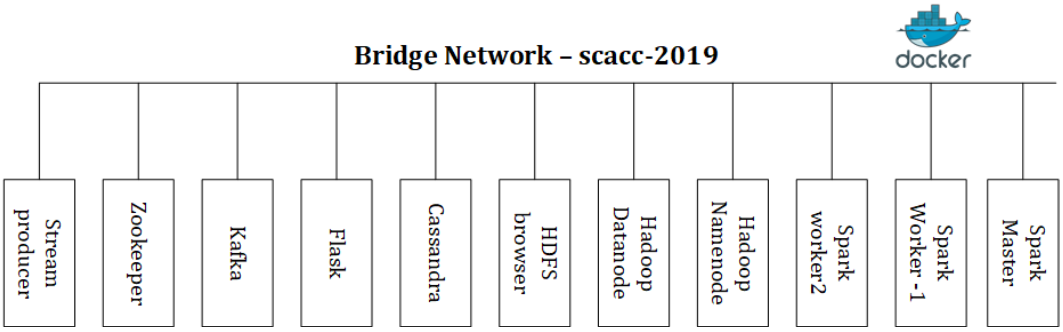

Figure 2: fig: The diagram of docker containers used in deploying different services

4 Conclusion

The general process of starting and completing this project were all covered here. It started with description of water demand forcasting through collection of dataset at KNMI to the implementation of the system and putting it on the browser.

In the collection of batch data, we stated the data attributes that was collected from KNMI. The prediction algorithm used which is Decision Tree Regressor was listed. In order to make the algorithm complex, we tried to introduce some noise into the model which brings in variation in the prediction algorithm. The streaming data was tested and everything is working fine. In all these, the entire components used for this project was deployed through docker containers.

Overall, our aim was to build a scalable water demand forcasting softare that runs an algorithm which predicts water usage base on the temperature. At this point, we can say that our aim has been accomplished.