Comparative Study of Big Data Predictive Analytics Frameworks

Nay Aung Soe Win, Mie Mie Su Thwin
University of Computer Studies, Yangon
nayaungsoewin@gmail.com, miemiesuthwinster@gmail.com

Abstract

Big data in terms is the huge volume of data that is difficult to be processed, handled and managed. The term big data is an emerging trend where a number of researches and data scientists are carried out for data analytics. One of the most interesting thing in big data analytics is about the predicting for future by using the data. Predictive analytics is the use of data and machine learning (ML) techniques to identify the future outcomes based on historical data. With predictive analytics a company can meaningfully leverage that business data to diagnose and solve business problems. But choosing a proper predictive analytics framework need to be consider many things, and it is also vital for every data analysis project. The best and the simplest practical way is to compare the response time of each framework. In this paper, we will investigate and compare two big data predictive analytics frameworks, Apache Mahout and Spark MLlib, from performance point of view. This comparative study will make things easier to the researcher and data scientists in the selection of big data analytics frameworks according to their analytics areas.

1. Introduction

The term “Big Data” has recently been applied to datasets that grow so large that they become very difficult to process with using traditional database management systems, and these dataset cannot be processed or stored using the resources of a single machine. With the fast development of networking, data storage, and the data collection capacity, big data is now rapidly expanding in all science and engineering domains. These are data sets whose size is beyond the ability of commonly used software tools and storage systems to capture, store, manage, as well as process the data within a tolerable elapsed time [5].

The challenges associated with big data are commonly referred to as the 3 V’s of Big Data - Volume, Velocity and Variety [3]. Volume refers to the amount of data, variety refers to the number of types of data and velocity refers to the speed of data processing. The 3 V’s provide a guide to the largest outstanding challenges associated with working with big data systems.

Big data analytics is the process of examining large data sets to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful business information. Big data analytics is the use of advanced analytic techniques against very large data sets that include different types such as structured/unstructured and streaming/batch, and different sizes from terabytes to zeta bytes. Big data comes from sensors, devices, video, audio, networks, log files, transactional applications, web, and social media.

2. Related Work

There are many types of existing big data analytic platforms for predictive analytics. Most of them based on MapReduce, distributed file system, and no-SQL indexing. Many areas such as intelligent transportation systems, health care system, telecommunication, banking, insurance, and other complicated areas related to economy need to develop big data analysis due to the complex characteristics of the industries themselves.

Weishan Zhang, Liang Xu, Zhongwei Li, and Qinghua Lu proposed a processing framework called, "Big Data Pre-Processing: A Quality Framework" [3]. In this framework, they propose a big data pre-processing quality framework, which aims at solving the numerous data quality issues that occur when attempting to apply data quality concepts to large data sets.

Juwei Shi, YunjieQiu, Umar Farooq Minhas, Limei Jiao, Chen Wang, Berthold Reinwald, and Fatma O¨ zcan assess the major compositional segments in MapReduce and Spark systems including: merging, execution time, and storing, by utilizing an arrangement of critical investigative workloads [7].

Sara Landset, Taghi M. Khoshgoftaar, Aaron N. Richter and Tawfiq Hasanin gives a rundown of criteria to making determinations of devices for Big Data Analytic alongside an investigation of the focal points and downsides of each [9].

3. Big Data Analytics

Analyzing big data allows analysts, researchers, and business users to make better and faster decisions using data that was previously inaccessible or unusable. Using advanced analytics techniques such as text analytics, machine learning, predictive analytics, data mining, statistics, and natural language processing, businesses can analyze previously untapped data sources independent or together with their existing enterprise data to gain new insights resulting in significantly better and faster decisions.

3.1. Apache Hadoop

Apache Hadoop is the de-facto framework for processing and storing large quantities of data, which is often referred to as "big data". The Apache Hadoop ecosystem consists of dozens of projects providing functionality ranging from storing, querying, indexing, transferring, streaming, and messaging [6]. Hadoop is a large-scale distributed file processing system designed for a cluster consisting of hundreds or thousands of nodes, each with multi-processor cores. Hadoop is designed to distribute and process large quantities of data across the nodes in the cluster. Hadoop does not require any special hardware. And also Hadoop may be used with any type of data, structured or unstructured. Hadoop is not a database, nor does Hadoop replace traditional database systems. Hadoop is designed to cover the scenario of storing and processing large-scale data, which most traditional systems do not support efficiently. Hadoop can join and aggregate data from many different sources and deliver results to other enterprise systems for further analysis. Hadoop has two main components: the Hadoop Distributed File System (HDFS) and MapReduce [2].

3.1.1. HDFS

The HDFS is a data storage and data processing file system and is designed to store and provide parallel, streaming access to large quantities of data (up to 100s of TB). HDFS sits on top of the native file system for an operating system. HDFS storage is spread across a cluster of nodes and a single large file could be stored across multiple nodes in the cluster. A file is broken into blocks, which is an abstraction over the underlying file system, with default size of 64MB. HDFS is designed to store large files and lots of them [2].

3.1.2. MapReduce

MapReduce is a distributed data processing framework for processing large quantities of data, distributed across a cluster of nodes in parallel. MapReduce processes input data as key/value pairs and is designed to process medium-to-large files. MapReduce has two phases: the map phase and the reduce phase. The map phase uses one or more mappers to process the input data and the reduce phase uses zero or more reducers to process the data output during the map phase. The input files is converted to key/value pairs before being input to the map phase. During the map phase, input data consisting of key/ value pairs is mapped to output key/value pairs using a user-defined Map() function . The map output data is partitioned and sorted for input to the reduce phase. The map output data is partitioned these values associated with the same key are partitioned to the same partition and sent to the same reducer. During the reduce phase, the data output from the map phase is processed to reduce the number of values associated with a key, or using some other user-defined function [2].

3.2 Predictive Analytics

Prediction is the key to the success of any business. Predictive analytics is the practice of extracting information from existing data sets in order to determine patterns and predict future outcomes and trends. It uses technology to predict the future and influence it. Organizations can use historical performance data to extrapolate and make predictions about the future and take actions that would affect those results. Predictive Analytics also offers a unique opportunity to identify future trends and allows organizations to act upon them, and it forecasts what might happen in the future with an acceptable level of reliability, and includes what-if scenarios and risk assessment.

The purpose of this paper is to compare two big data analytics frameworks, Mahout and Spark MLlib for predictive analytics from performance point of views.

3.3. Big Data Analytics Tools and Methods

With the evolution of technology and the increased multitudes of data flowing in and out of organizations daily, there has become a need for faster and more efficient ways of analyzing such data. Such data sets can no longer be easily analyzed with traditional data management and analysis techniques and infrastructures. Therefore, there arises a need for new tools and methods specialized for big data analytics.

Big Data analytics is the process of applying algorithms in order to analyze sets of data and extract useful and unknown patterns, relations, and information [1]. Furthermore, big data analytics are used to extract previously unknown, useful, valid, and hidden patterns and information from large data sets, as well as to detect important relationships among the stored variables.

3.3.1. Apache Spark

Apache Spark is a fast, in-memory data processing engine with elegant and expressive development APIs to allow data workers to efficiently execute streaming, machine learning or SQL workloads that require fast iterative access to datasets. With Spark running on Apache Hadoop, developers can now create applications to exploit Spark's power, derive insights, and enrich their data science workloads within a single, shared dataset in Hadoop. The Hadoop based architecture provides the foundation that enables Spark and other applications to share a common cluster and dataset while ensuring consistent levels of service and response [8]. Apache Spark consists of Spark core and a set of libraries. The core is the distributed execution engine and the Java, Scala, and Python APIs offer a platform for distributed application development.

3.3.2. Spark MLlib

Spark also includes MLlib, a library that provides a growing set of machine algorithms for common data science techniques: classification, regression, clustering, and dimensionality reduction. Spark's machine learning pipeline is a high level abstraction to model an entire data science workflow. Spark MLlib is a production-ready distributed machine-learning library developed on top of Spark. Since Spark's 0.8 release, MLlib has come packaged with Spark and has dramatically simplified machine-learning for Spark users. The ML pipeline package in Spark models a typical machine learning workflow and provides abstractions like transformer, estimator, pipeline and parameters [8]. This is an abstraction layer that makes data scientists more productive.

3.3.3. Apache Mahout

Apache Mahout is a powerful, scalable machine learning library that runs on top of Hadoop MapReduce, and using the MapReduce paradigm. Machine learning is a discipline of artificial intelligence focused on enabling machines to learn without being explicitly programmed, and it is commonly used to improve future performance based on previous outcomes [7]. Once big data is stored on the HDFS, Mahout provides the data science tools to automatically find meaningful patterns in those big data sets. The Apache Mahout project aims to make it faster and easier to turn big data into big information. Mahout supports four main data science use cases: collaborative filtering, clustering, classification and frequent itemset mining [7]. Spark allow to process multiple iterations on same input data because of in-memory caching. On the other hand, Mahout is ten times slower than Spark because Mahout's map and reduce operation need to read and write on physical hard disk frequently which causes extra delay in execution.

Spark's Resilient Distributed Dataset (RDD) provides the functionality where programmer can write a program in a distributed manner that offers a restricted form of distributed shared memory because RDD helps to persist the data in RAM and help for effectively process these data. On the other hand, multiple iterations on same input data is not possible in Mahout, and it slow due to data replication and disk I/O than Spark.

3.4. Apache Mahout vs. Spark MLlib

Apache Mahout was built to run on top of Hadoop, which does not handle iterative jobs very well. Hadoop writes to disk on every iteration. Since Machine Learning algorithms generally use many iterations, this makes Mahout run very slowly.

MLlib is a collection of high-level algorithms that runs on Spark. Spark is often noted as running iterative algorithms much faster than Hadoop since it tries to avoid writing to disk as much as possible. MLlib was built on top of Spark to take advantage of Spark's efficiency when running iterative machine learning algorithms.

The other major benefit is that MLlib is one of several libraries built on top of Spark: MLlib, Streaming, SparkSQL, and GraphX. Having this suite of libraries which easily interoperate makes development much easier.

Mahout has proven capabilities that Spark's MLlib still haven't touched. While Mahout is mature and comes with many ML algorithms to choose from, and therefore constrained by disk accesses it is slow and does not handle iterative jobs very well. Since Machine Learning algorithms generally use many iterations, this makes Mahout run very slowly. In contrast, MLlib is built on top of Spark, making it much faster than Mahout. The Spark MLlib comes in two packages: MLlib and ML. While MLlib contains the original API built on top of RDD, ML provides higher-level API built on top of DataFrames for constructing ML pipelines which is much faster than MLlib.

Spark MLlib library, which is different from Mahout, makes use of the DataFrame format which is fairly new in Spark, however it does not allow all types of algorithms, therefore although Spark ML may be faster than Mahout, you might find yourself needing Mahout anyways.

4. Telecom Churn Prediction Model

We tested our comparison by using a churn prediction model for a telecom and, we built telecom customer churn prediction models which are built on corporate consumer data. Churn prediction aims to detect customers intended to leave a service provider. Retaining one customer costs an organization from 5 to 10 times than gaining a new one. Churn prediction process is a highly debated research area for more than ten years [4]. Researchers from different disciplines have tried to analyze this problem from their own perspectives to figure out a clear understanding and to recommend an effective solution for churners in many business areas.

Predictive models can provide correct identification of possible churners in the near future in order to provide a retention solution. The models assess all customers and aim to predict churn and loyalty behavior based on the analysis of demographic data, customer purchases history, and service usage and billing data. All characteristics and transactions are analyzed, ranked and modelled to create customer or segment loyalty profiles [7].

The proposed model is composed of six steps which are identify problem domain, data selection, investigation the data set, classification, clustering and knowledge usage. The data set contains 19 attributes of randomly selected 100000 customers subscribed to a prepaid service for a time interval of three months. The attributes cover outgoing and incoming calls statistics. The data were provided with an indicator for each customer whether the customer churned (left the company) or still active.

4.1. System Overview Architecture

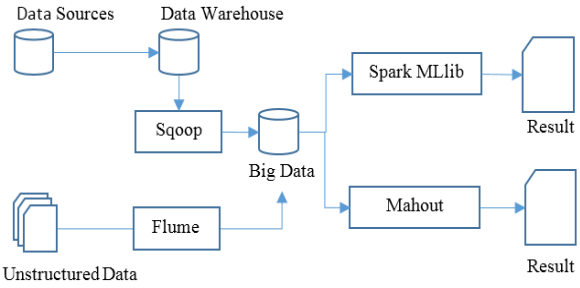


Figure 1. System Overview Architecture

Normally in a telecom company, customers' structure data such as usage histories are stored in different databases, and unstructured data such as web logs, mobile device activity, and social media activities are stored in the flat files like csv. Then structured data are moved into an enterprise data warehouse, and unstructured data move into big data system directly using Apache Flume. Data from data warehouse moved into big data system using Sqoop. Then we made analysis using MLlib and Mahout Data analytics frameworks. Then the analysis results are shown.

In this paper, we used data mining models for predicting customer churn in a telecommunication company. All models are composed of a clustering algorithm and a classification one. First, the clustering algorithm is applied to filter the data from outliers and unrepresentative behaviors. The resulted filtered data of the clustering algorithm become the input to the classification algorithm. Then the classification algorithm uses these data in the learning process and builds the final churn prediction model. For clustering, we use hierarchical clustering, *k*-means clustering, and while for classification we use the Multilayer Perceptron Artificial Neural Networks.

The proposed churn prediction model in the study is based mainly on two processes. In the first process a clustering algorithm is applied on the collected customer's data in order to separate customers to a number of groups that represent

different behaviors. We clustered the customers on their Average Revenue per User (ARPU). Largest two clusters that represent churners and non-churners are merged again and considered as an input to the next process. On the other hand, the small clusters are neglected as they represent outliers and unrepresentative behaviors. We first tested this analysis on Spark MLlib framework and it took 17.46 minutes to get the result. The business problem is to predict the likelihood of customers churning based on customer information such as customer demographic information, revenue per month, service quality, last tariff plan, calling usage etc. With trained mining model and new customer data, the mining model can be applied to new customer to predict who is more likely to churn, therefore customers are classified into two groups, churners and non-churners. In this report, customers are first grouped according to ARPU value last month. ARPU is a measurement used primarily by telecommunications and banking, defined as the total revenue divided by the number of subscribers. In each ARPU group, number of predicted churners is given. Users should pay more attention to those high ARPU groups with high predicted churn rate.

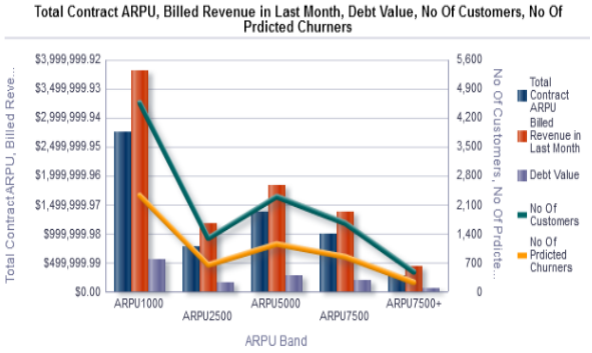


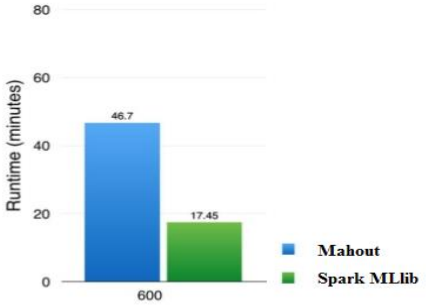
Figure 2. Prediction Output

This analytic report shown in Figure-2 provides year-level subscription performance based on churn statistics relating to a customer, such as high churn rate analysis for a subscription. According to the figure, the subscribers with low ARPU have high percentage to change their provider to a new one offering same service.

We also tested the same problem with Mahout Framework, and this takes about 46.7 minutes to receive the output result as shown in Figure-3 that is slower than Spark MLlib.

Table 1. Average Revenue per User Band data

ARPU Band	Total Contract ARPU	Billed Revenue in Last Month	Debt Value	No of Customers	No of Predicted Churners
ARPU1000	$ 2,742,197.00	$ 3,807,081.00	$ 570,403.00	4,565	2,333
ARPU2500	$ 772,472.00	$ 1,171,763.00	$ 159,442.00	1,291	646
ARPU5000	$ 1,367,811.00	$ 1,838,149.00	$ 289,213.00	2,287	1,142
ARPU7500	$ 1,002,035.00	$ 1,385,883.00	$ 209,574.00	1,668	856
ARPU7500+	$ 283,090.00	$ 435,377.00	$ 58,563.00	471	240



Data size: 200 GB, 600 Million Records

Figure 3. Performance Evaluation

4.2. Application to Customers' Usage Data Set

We used three month of data with over 600 million transactions as input data. In these data, customer demographics (age, gender, marital status, location, etc.), call statistics (length of calls like local, national and International, etc.), billing information (what the customer paid for), voice and data product (broadband services, special data tariffs, etc.), complaints and disputes (customer satisfaction issues and the remedial steps taken), and credit history will be included.

Table 2. Input Data Set Columns

FIELDS	DESCRIPTION
TRAN_DATE	Transaction Date
USAGE_LOCATION	Location of most usage happened
GENDER	Gender of Subscriber
AGE	Age of Subscriber
MARITAL_STATUS	Marital status of Subscriber
SUBSCRIBER_NO	Subscriber's Phone Number
INTERNET_USAGE_RVN	Revenue for using Internet
ONNET_VOICE_USAGE_RVN	Revenue for calling same operator
OFFNET_VOICE_USAGE_RVN	Revenue for calling cross operator
ISD_VOICE_USAGE_RVN	Revenue from calling to another country
ONNET_SMS_USAGE_RVN	Revenue for sending SMS to same operator
OFFNET_SMS_USAGE_RVN	Revenue for sending SMS to cross operator
ISD_SMS_USAGE_RVN	Revenue for sending SMS to another country
NO_OF_ONNET_CALL_PER_MON	Total number of call per month with same operator
NO_OF_OFFNET_CALL_PER_MON	Total number of call per month with cross operator
NO_OF_ONNET_SMS_PER_MON	Total number of call per month with same operator
NO_OF_OFFNET_SMS_PER_MON	Total number of call per month with cross operator
INTERNET_USAGE_MB_PER_MON	Total Internet usage MB per month
ROAMING_USAGE_RVN	Revenue for using Roaming service

4.3. Experiment Environment

We tested two frameworks for predictive analysis based on Spark MLlib and Apache Mahout. We created one NameNode, one Secondary NameNode, and four DataNode. The NameNode in Hadoop is the node where Hadoop stores all the location information of the files (metadata) in HDFS. This information is required when retrieving data from the cluster as the data is spread across multiple machines. The secondary name node is responsible for performing periodic housekeeping functions for the NameNode. It only creates checkpoints of the file system present in the NameNode. The DataNode is responsible for storing the files in HDFS. It manages the file blocks within the node. It sends information to the NameNode about the files and blocks stored in that node and responds to the NameNode for all file system operations. JobTracker is responsible for taking in requests from a client and assigning TaskTrackers with tasks to be performed. The JobTracker tries to assign tasks to the TaskTracker on the DataNode where the data is locally present (Data Locality).

TaskTracker is a daemon that accepts tasks (MapReduce and Shuffle) from the JobTracker. The TaskTracker keeps sending a heartbeat message to the JobTracker to notify that it is alive. The software and hardware components used in our development, and data set used processing are described in Table-2.

4.4. Experiment and Result Discussion

The specifications of hardware and necessary software components used for the experiment are described in Table-3.

Table 3. Experimental Environment

Server Machine	Dell PowerEdge T430 Server
Server OS	Oracle Enterprise Linux Server 6.8 64 bit
Client OS	Windows 8.1 64 bit
Host Specification	Intel Xeon processor E5-1410 64 GB RAM 12TB HDD (2 6TB HDD)
VMs Specification	8 GB RAM, 500 GB HDD
Number of Data Nodes	4
Big Data Software Components	Apache Hadoop 2.7.1 Hive 2.0 Apache Spark 2.0.2
Data Set	Telecom Dataset from Yahoo Labs

Testing used to verify the predictive relationship obtained in the training phase. The data was separated into training and testing sets with a 70:30 ratio. In the case of the given data set, the data for the first two months were used for training while that for the third month was used for testing. In the testing phase, we achieved comparative results as well. The bar chart in Figure-3 shows the results for test phase. According to Figure-3, Mahout takes about 46.7 minutes to get the result, and Spark MLlib takes about 17.45 minutes to get the result. Apache Spark is faster than MapReduce Mahout for this predictive analytics case.

And also we found that MapReduce is inefficient for multi-process applications that require low-latency data sharing across multiple parallel operations, and it is slower than Spark because intermediate result is stored in hard disk. Spark is faster than MapReduce especially in iterative operations because the intermediate data and result are persist in memory.

5. Conclusion and Further Work

Although most of the algorithms on Mahout have still been based on MapReduce, Spark's consistent improvements and increasing user base has lead Mahout. Today there are many misconception about the comparison of Spark and MapReduce framework. Many researchers are claiming that Spark is about 10 times faster than MapReduce framework because Spark offers an abstraction called Resilient Distributed Datasets (RDD) to support these applications efficiently and RDD can be stored in memory between queries without requiring replication. Churn prediction and management is critical in mobile telecom markets. In order to be competitive in this market, mobile service providers have to be able to predict possible churners and take proactive actions to retain valuable customers. In this paper, we tested both big data predictive analytics framework from performance point of view and the experimental result indicated that Spark is about 3 times faster than Mahout Framework. This paper also contains the chart comparison between these two tools. Each tool has its own advantages and disadvantages. By employing this comparative study, this concluded that Spark has better than MapReduce in some use cases. This comparative study will make things easier to the data scientists in the selection of big data analytics tools according to their areas. In future, we will find out the solution how to improve

the accuracy by applying machine learning algorithm on the Spark and MapReduce.

References

[1]. A.B. Smith, C.D. Jones, and E.F. Roberts, "Article Title", *Journal*, Publisher, Location, Date, pp. 1-10.

[2]. Alex Holms, "Hadoop in Practice", 2012, Manning Publications co.

[3]. [3] D. Laney, "3d data management: Controlling data volume, velocity and variety," META Group Research Note, vol. 6, 2001.

[4]. Kang, C. and Pei-Ji, S. (2008) Customer Churn Prediction Based on SVM. ISBIM'08. IEEE International Seminar on Business and Information Management, Wuhan, 19 December 2008, 306-309. Kubick, W.R.: Big Data, Information and Meaning. In: Clinical Trial Insights, pp. 26–28 (2012).

[5]. Pete Warden, "Big Data Glossary a guide to the new generations of data tools", 2011, O'Reilly.

[6]. Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman, "Mahout in Action", 2012, Manning Publications co.

[7]. Spark. https://spark.apache.org/, 2016. [Online; accessed 25-March- 2016].

[8]. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Technical Report UCB/EECS-2011-82, EECS Department, University of California, Berkeley, 2011.