Performance Analysis of a Scalable Naïve Bayes Classifier on MapReduce and Beyond MapReduce

Myat Cho Mon Oo[1], Thandar Thein[2]
University of Computer Studies, Yangon[1], University of Computer Studies, Maubin[2]
myatchomonoo@ucsy.edu.mm[1], thandartheinn@gmail.com[2]

Abstract

Many real world areas from different sources generate the big data with large volume of high velocity, complex and variable data. Big data becomes a challenge when they are difficult to process and extract knowledge using traditional analysis tools. Therefore the scalable machine learning algorithms are needed for processing such big data. Recently Hadoop MapReduce framework has been adapted for parallel computing. MapReduce may not fit for most of the real world data applications. For large scale machine learning on distributed system, Spark has finally become much more viable beyond MapReduce. Although both of these frameworks are Apache-hosted data analytic framework, their performance varies significantly based on the use case under their implementation. This paper aims to analyze the performance of scalable Naïve Bayes classifier (SNB) which is implemented on MapReduce and Beyond MapReduce over different real world datasets. The comparison results show that SNB on Beyond MapReduce provides minimal processing time than SNB on MapReduce for efficiently big data classification.

***Keywords**: Big Data, Beyond MapReduce, MapReduce, scalable Naïve Bayes, Spark*

1. Introduction

The massive growth of big data is increasing overwhelmingly during the past decades. The IBM Big Data Flood Infographic shows that 2.5 quintillion bytes of data are created every day [1]. Big data doesn't only refer to massive data, but also a series of techniques which turn a flood of data into valuable information. Traditional data mining techniques are not well suited to process the full value of big data. Therefore, an efficient and distributed processing model is needed to process and analyze such data. For processing huge amount of data, Hadoop is becoming the core technology to solve the huge data problems for large organizations with cloud storage. A commonly used architecture for Hadoop consists of Hadoop Distributed File System (HDFS) for big data storage and MapReduce for distributed parallel computing.

Machine learning is also ideal for exploiting the opportunities hidden insight in big data. Apache Mahout is an open source machine learning library built on top of Hadoop to provide distributed analytics capabilities. Although it was originally developed with MapReduce based algorithm, MapReduce was inefficient for most of the scalable machine learning that Mahout pioneered because of its limitation. Also, alternative frameworks such as Spark have finally become much more viable.

Spark absorbs the advantages of Hadoop MapReduce, unlike MapReduce, the intermediate and output results of the Spark jobs can be stored in memory, which is called Memory Computing. Memory Computing improves the efficiency of data computing. So, Spark is better suited for iterative applications, such as Data Mining and Machine Learning [2]. Starting from the release 0.10.0, a new generation of mahout was born for building backend independent programming environment, also called the code name, "Samsara". Mahout Samsara is backend-agnostic and uses a Scala-based programming environment to support writing parallel mathematical languages.

In this paper, the performances of SNB on MapReduce and Beyond MapReduce are compared and analyzed for efficiently big data classification. The remainder of this paper is organized as follow. First, the related work of this paper is described in section 2. Then, some background information is presented in section 3. And then architecture of SNB on MapReduce and Beyond MapReduce is presented in section 4. After that the performance evaluation and result discussions are shown in section 5. Finally, the conclusion of this paper is summarized in section 6.

2. Related Work

There are many types of existing big data processing model for large scale data processing. Most of them are open source big data technologies and different performance on different data natures. To analyze the big data and exact useful information, choosing the efficient model from performance analysis is important. The authors [3] analyzed the performance of K-mean algorithm on MapReduce and Spark. They compared the Map and Reduce phases of K-mean on these frameworks. They showed that Spark has faster processing time than MapReduce for clustering data.

To identify and explain the impact of different architectural choices, the authors [4] proposed a methodology by comparing the performance of Spark and Flink for big data analytics. Their key finding was that there none of the two framework outperforms the other for all data types, sizes and job patterns.

With the rapid growing amount of data, the need for scalable model to store and process these data is increasing. The authors [5] performed the comparison on scalability for batch big data processing on Spark and Flink. They used Spark MLlib and Flink for their comparative study. Their experimental results show that Spark has better performance and overall lower runtime than Flink.

This paper intends to compare the performance of SNB on MapReduce and Beyond MapReduce for efficiently big data classification. By using SNB on Beyond MapReduce, it performs efficiently processing on big data in distributed environment and provides good performance results than SNB on MapReduce.

3. Preliminaries

This section provides the preliminaries of the paper. First, section 3.1 introduces Apache Hadoop framework. Section 3.2 and 3.3 present MapReduce and Apache Spark, highlighting their architectural differences. Then section 3.4 describes Apache Mahout and Mahout Samsara.

3.1. Apache Hadoop Framework

Apache Hadoop [6] is an open-source software framework for storing and processing large data in a distributed manner. It supports data intensive distributed applications on large clusters built of commodity hardware. This framework is designed to be scalable, which allows the user to add more nodes as the application requires. Hadoop consists of the Hadoop Distributed File System (HDFS) for big data storage and MapReduce framework for distributed processing. Hadoop cluster consists of a single master node and many worker nodes. HDFS is based on master/slave architecture. The development of Hadoop-based data mining techniques has been widely spread, because of its fault-tolerant mechanism and its ease of use.

3.2. MapReduce Framework

The MapReduce framework [13] is the core of Apache Hadoop. This programming paradigm provides for massive scalability across hundreds or thousands of servers in a Hadoop cluster. It consists of two main phases: map and reduce. In the map phase, it takes a set of data and converts it into another set of data, where individual elements are broken down into tuples as key-value (< k, v >) pairs. In the reduce phase, the reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job. Although MapReduce had higher performance while comparing with traditional data mining algorithm, it has some important short comings. It may inefficient for application that share data across multiple steps.

3.3. Spark Framework

Apache Spark [7] is a fast and general engine for large-scale data processing. It has an advanced DAG (Directed Acyclic Graph) execution engine that supports acyclic data flow and in-memory computing. The core abstraction of Spark is Resilient Distributed Dataset (RDD) which has a better ability of computing and fault tolerance. So, Spark can allow us to store a data cache in memory, to perform computation and iteration for the same data directly from memory. It saves huge amount of disk I/O operation time. Therefore, it is more suitable for developing scalable machine learning algorithms.

3.4. Apache Mahout

Apache Mahout [8] is an environment for creating scalable, performant, machine learning applications. It is a machine learning library that runs over the Hadoop system for solving the clustering and classification problems. It born a new generation of mahout, linear algebra environment, known as the code name "Mahout Samsara"(release 0.10.0 or later). MapReduce was not a very good fit for most of the machine learning that mahout pioneered. In place of Hadoop MapReduce, Mahout has been focusing on implementing flexible and backend-agnostic machine learning environment.

Mahout Samsara is a new generation of mahout. It is also known as "Beyond MapReduce" because it is the part of mahout that deals with more advanced back-ends, post-MapReduce generation: Spark, Flink, and H2O. These backends extend the set of distributed paradigms beyond just MapReduce. Therefore, machine learning algorithms built with the mahout Samsara DSL are better served for iterative nature of applications.

4. Scalable Naïve Bayes Classifier (SNB)

Naïve Bayes algorithm is a popular algorithm in text classification field. It is a simple probabilistic classifier based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. It can be used for binary and multiclass classification. With the rapid growth of online documents on the web, the ability to automatically mine useful information from massive data has been a common challenge for traditional Naïve Bayes classifier. Therefore scalable Naïve Bayes classifier has been developed on distributed computing frameworks to meet the needs of big data classification in terms of memory requirements and computing power.

4.1 SNB on MapRedcue

On top of Hadoop MapReduce and HDFS, Naïve Bayes classifier can scale up to classify millions of data. The issue of scalability was seldom solved using actual scaling in machine learning. Therefore, SNB on MapReduce can process the vast amount of data with parallel and distributed processing. Figure 1 shows a SNB on MapReduce architecture.

Despite its popularity, there are some limitations of MapReduce to develop for scalable machine learning algorithms. For distributed parallel computing on large scale data, MapReduce consists of two main phases: map and reduce. Map takes a set of data and converts it into another set of data, where individual element are broken down <key, value> pairs and Reduce takes the output from the map as input and process further. The intermediate and output results of each phase are stored in HDFS. MapReduce requires a lot of time to perform these tasks thereby increasing latency. This may become overhead and implementing iterative map reduce jobs is expensive due to the huge space consumption by each job.

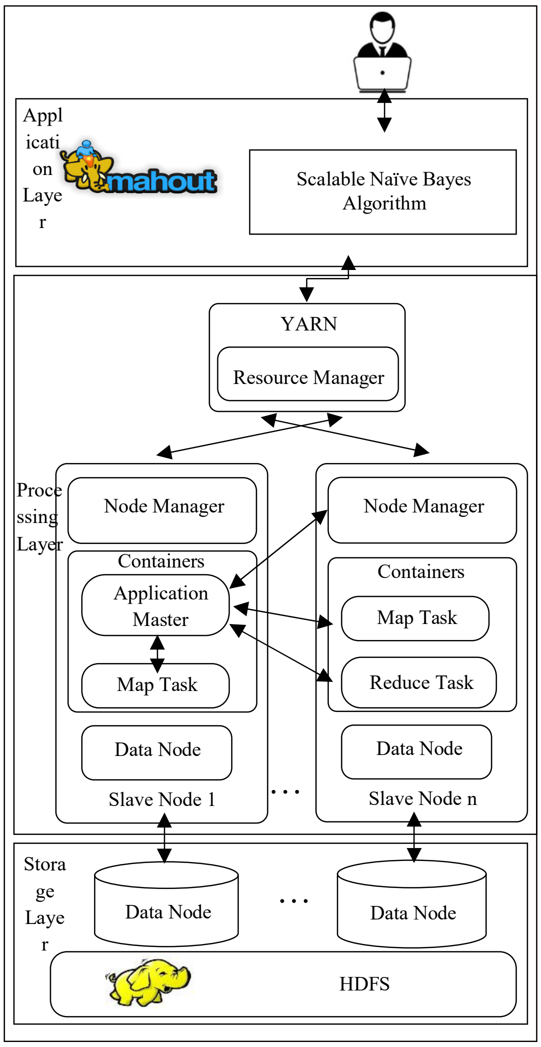

Figure 1. SNB on MapReduce

4.2. SNB on Beyond MapReduce

Samsara currently has two flavors of Naïve Byes implemented its distribution. The first is standard Multinomial Naïve Bayes ("MNB" or "Bayes") and the latter is a variation on Transformed Weight-normalized Complement Naïve Bayes

("TWCNB" or "CBayes") [9]. In this paper, Samsara's MNB is used for evaluation and Apache Spark is used as backend processing engine.

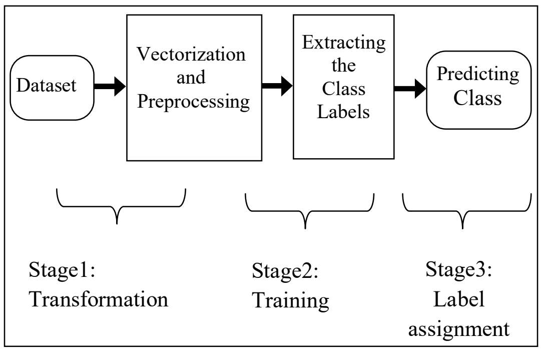

Figure 2. Processing Stages of SNB on MapReduce

SNB on Beyond MapReduce is divided into three stages: transformation, training and label assignment. The processing stages of SNB on Beyond MapReduce are shown in Figure 2. In the transformation stage, the dataset is acquired and vectorized the document. In order to make good use of the computing resources, Hadoop cluster is implemented. Hadoop is a framework that admits the data in sequence file format. Mahout Samsara over Hadoop also admits the data in the sequence file directory. The input data is converted to sequence file format to parse the <text> element of each document. After taking the sequence file conversion phase, the documents are vectorized using ***mahout seq2sparse***. It converts from the sequence file directory to vector format. The sequence file will be accepted as input and produce the output as vector using a weighting factor like TF-IDF (Term frequency-Inverse Document Frequency) scheme.

In training stage, SNB on Beyond MapReduce uses the Spark random Split API to split the training and testing sets. Since Spark backend environment is chosen, the algorithm in Mahout Samsara can take advantages of Spark native function. SNB stores the class label of each vectorized document as the row keys. And then, it extracts the all possible document identifier for each document.

In label assignment stage, SNB assigns a label to a vectorized document using a classification function. It predicts a classification of the document by assigning a class with the largest posterior probability. Figure 3 shows a SNB on Beyond MapReduce architecture.

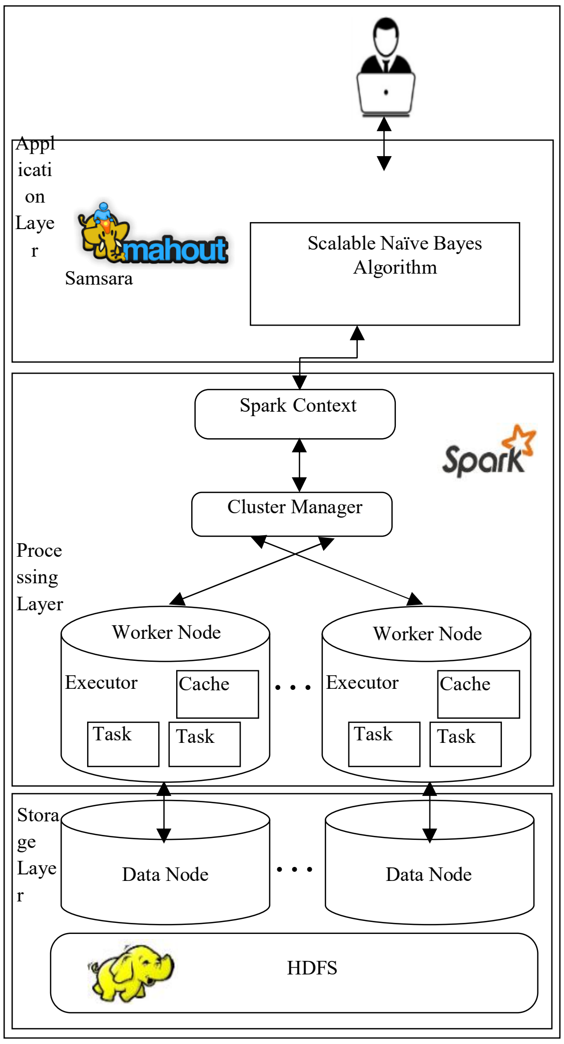

Figure 3. SNB on Beyond MapReduce

5. Performance Evaluation

This section compares and analyzes the performance of SNB on MapReduce and Beyond MapReduce. Instead, "scalable" machine learning is almost always based on finding more efficient algorithms, and most often, approximations to the original algorithm which can be computed much more efficiently. SNB on Beyond MapReduce takes the advantages of Spark which support in-memory computing. Therefore it provides faster computing with saving resource consumptions. Then, the performance analysis of SNB on MapReduce and Beyond MapReduce will be evaluated using confusion matrix which records correctly and incorrectly recognized examples for each class.

The four matrices of performance that measure the classification quality for the positive and negative classes independently are:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$TP_{rate} = \frac{TP}{TP+FN} \quad (2)$$

True Positive rate TPrate is the percentage of positive cases correctly classified as belonging to the positive class.

$$TN_{rate} = \frac{TN}{FP+TN} \quad (3)$$

True Negative rate TN_{rate} is the percentage of negative cases correctly classified as belonging to the negative class.

$$FP_{rate} = \frac{FP}{FP+TN} \quad (4)$$

False Positive rate FP_{rate} is the percentage of negative cases misclassified as belonging to the positive class.

$$FN_{rate} = \frac{FN}{TP+FN} \quad (5)$$

False Negative rate FN_{rate} is the percentage of positive cases misclassified as belonging to the negative class.

5.1. Experimental Environment

The two frameworks are implemented on distributed Hadoop cluster for performance evaluation. To build a storage cluster, 3 VMs are created for each cluster. All the experiments have been carried out over a Hadoop cluster of three computing nodes having configuration machine of Intel CORE i7 processor, 8 GB of RAM, 1 TB HDD on Linux Ubuntu-16.04 system.

The specific details of the software used and its configuration are open-sour Apache Hadoop distribution (Hadoop 2.6.0), apache spark (1.5.2), the latest release of Mahout Samsara (0.12.3) and Mahout (0.9). The descriptions of datasets used in this paper are presented in table 1.

Table 1.Experimental datasets

Dataset	Description
D1	Enron Data set [10] -Number of email (ham) : 52340 -Number of email (spam) : 30659
D2	Movie Reviews Data set [11] - Rotten Tomatoes movies review dataset -contain 1000 positive and 1000 negative reviews
D3	Twitter Data set [12] -Sentiment data for product reviews -contain 554470 positive and 494105 negative reviews

5.2. Performance Evaluation and Result Discussion

The performances analysis of SNB on MapReduce and Beyond MapReduce is presented in this section. For comparing with different datasets, the datasets are partitioned into 60/40 ratio for training and testing. The accuracy comparison of each dataset is shown in Figure 4. The results clearly show that SNB on Beyond MapReduce performs the large amount of data in minimal processing time than SNB on MapReduce.

In Dataset D1, SNB on Beyond MapReduce can correctly classify the 82999 emails within 39 seconds while SNB on MapReduce took 119 seconds. In dataset D2, SNB on Beyond MapReduce processed the movie review data within 40 seconds and SNB on MapReduce took 150 seconds. In dataset D3, SNB on Beyond MapReduce only took 93 seconds for processing tweet data although SNB on MapReduce took 280 seconds.

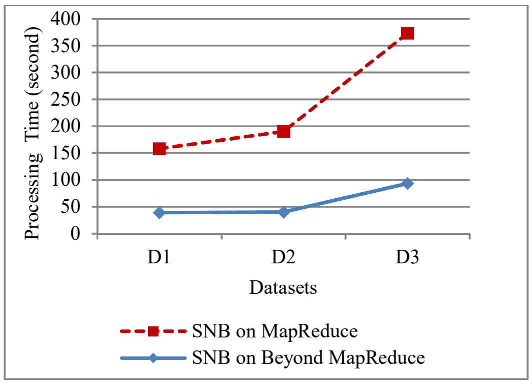

Figure 4. SNB on MapReduce vs. SNB on Beyond MapReduce (time)

The advantage of SNB on Beyond MapReduce is preserved over large datasets as well. Because it takes the full advantages of Spark's in-memory computing and other wonderful native function. Moreover, it uses the Samsara's text vectorization pipeline which is slightly different TF and IDF transformation of SNB on MapReduce. Therefore the label extraction and summation of TF_IDF observations per label require some shuffling and take a few times for efficiently big data classification.

According to the comparison results, SNB on Beyond MapReduce provides the minimal processing time than SNB on MapReduce over the different experimental datasets.

In this paper, the issue of scalability was seldom solved using actual scaling in machine learning. Figure 5 shows the accuracy comparison of SNB on MapReduce and Beyond MapReduce. A scalable Naïve Bayes classifier means having a learning algorithm which can deal with any amount of data, without consuming ever growing amounts of resources like memory and providing the accurate results.

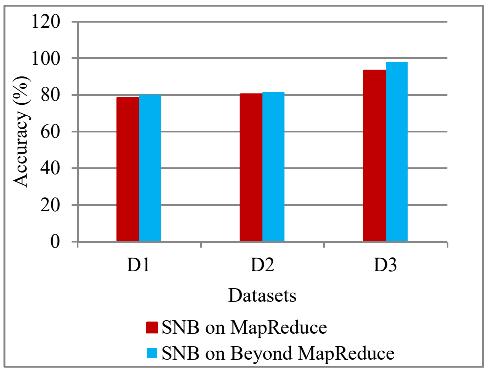

Figure 5. SNB on MapReduce vs. SNB on Beyond MapReduce (accuracy)

With the rapid development of information technology, faster is better in computing. Even though the accuracies of both of which are not slightly different, SNB on Beyond MapReduce can provide not only faster computing but also good performance results in term of accuracy.

6. Conclusion

Scalability has become one of the core concept slash buzzwords of big data. Big data requires a scalable and parallel machine learning algorithm for efficiently processing. Analytical processing time is required to take minimal time to get results. In this paper, the performance of a scalable naïve Bayes classifier on MapReduce and Beyond MapReduce is compared and analyzed. SNB on Beyond MapReduce can parallel process on big data with efficient computing. Our comparative study can show that SNB on Beyond MapReduce has a better accuracy and faster processing time than SNB on MapReduce. It also provides the good scalable performance on distributed environment. As future work, the issues in big data classification with scalability will be considered. Regarding the performance analysis of SNB on MapReduce and Beyond MapReduce, the techniques to deal with scalability of big data will be studied.

References

[1] Big Data Flood Infographic website. [Online] Available: https://www.ibm.com/, [Accessed:15-Sept-2017]

[2] J. Fu, J. Sun, K. Wang, "SPARK—A Big Data Processing Platform for Machine Learning", in Proceedings of IEEE International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration, 2016, pp. 48-51

[3] S. Gopalani, R. Arora, "Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means", in Proceedings of the International Journal of Computer Applications (0975 – 8887) Volume 113 – No. 1, March 2015

[4] O. Marcu, A. Costan, G. Antoniu, M. Hernandez, "Spark versus Flink: Understanding Performance in Big Data Analytics Frameworks", in Proceedings of IEEE International Conference on Cluster Computing, 2016, pp. 433-442

[5] D.Gil, S. Gallego, S. Garcíal, and F. Herrera, "A comparison on scalability for batch big Data processing on Apache Spark and Apache Flink", in Proceedings of Big Data Analytics, Springer International Publishing, BioMedCentral, March 2017.

[6] Apache Hadoop website. [Online]. Available: http://hadoop.apache.org/,[Accessed: 12-Aug-2017]

[7] Apache Spark website. [Online]. Available: http://Spark.apache.org/, [Accessed: 12-Sept-2017]

[8] Apache Mahout website. [Online]. Available: http://mahout.apache.org/, [Accessed: 12-Sept-2017]

[9] D. Lyubimov, A. Palumbo. Apache Mahout: Beyond MapReduce, 1st ed., 2016.

[10] W. Cukierski. The Enron Email Datasets on kaggle. [Online] Available: https://www.kaggle.com/wcukierski/enron-email-dataset, [Accessed: 15-Aug-2017]

[11] Pang and L. Lee. The Movie Review Data on kaggle. [Online] Available: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews, [Accessed: 15-Aug-2017]

[12] W. Debki. Twitter Data on Data world. [Online] Available: https://data.world/datasets/twitter, [Accessed: 12-Aug-2017]

[13] Hadoop MapReduce website. [Online] Available: https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm [Accessed: 12-Aug-2017]