International Journal of Engineering & Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-3444

International Journal of Engineering and Technology Volume 3 No. 2, February, 2013

Building Machine Learning Algorithms on Hadoop for Bigdata

Asha T, Shravanthi U.M, Nagashree N, Monika M
Department of Information Science & Engineering
Bangalore Institute of Technology, Bangalore, INDIA

ABSTRACT

Machine Learning (ML) is at the core of data analysis. Machine Learning Algorithms (MLA) are sequential and recursive and the accuracy of MLA's rely on size of the data (i.e., greater the data more accurate is the result). Absence of a reliable framework for MLA to work for bigdata has made these algorithms to cripple their ability to reach the fullest potential. Hadoop is one such framework that offers distributed storage and parallel data processing. The Existing problem to implement MLA on Hadoop is that the MLA's need data to be stored in single place because of its recursive nature, but Hadoop does not support data sharing. In this paper we propose an approach to build Machine Learning models for recursive MLA's on Hadoop so that the power of Machine Learning and Hadoop can be made available to process Bigdata. We compare the performance of ID3 decision tree algorithm, K-means clustering algorithm and K-Nearest Neighbor algorithm on both serial implementation and parallel implementation using Hadoop.

Keywords: *Machine Learning, Hadoop, Mapreduce, Parallelism, Bigdata*

1. INTRODUCTION

Widespread success of machine Learning is in addressing an even broader range of tasks for actionable business insights and optimizations. But amount of data being produced today is so much that, 90% of the data in the world today has been created in the last two years alone. The Internet provides a resource for compiling enormous amounts of data, often beyond the capacity of individual disks, and too large for processing with a single CPU. Hadoop an open source framework developed by Dough Cutting [2], is built on top of the Hadoop Distributed File System (HDFS) and Mapreduce paradigm [1] provides a parallelization framework which has garnered considerable acclaim for its ease-of-use, scalability, and fault-tolerance.

This article proposes to use Hadoop, an efficient and scalable platform for MLA's to process bigdata. It helps to solve the performance overhead caused by legacy system for MLA to process bigdata. Section A-D introduces the Apache Hadoop framework, Section II outlines related work on these problems, Section III illustrates a few Machine Learning algorithms and Section IV and V discusses our improvements to the Hadoop implementation.

1.1 Bigdata

The amount of digital information produced is exceeding day by day. The majority of this data will be "unstructured" complex data poorly suited to be managed by structured storage systems like relational databases. Unstructured data comes from many sources and takes many forms, it could be web logs, text files, sensor readings, user generated content like product reviews or text messages, audio, video and still imagery and more. All these kind of unstructured data which cannot be handled efficiently using existing approaches constitute a "big-data".

Dealing with big data requires two things:

- Inexpensive, reliable storage; and
- New tools for analyzing unstructured and structured data.

1.2 Hadoop

Apache Hadoop is a powerful open source software platform that addresses above mentioned problems of Bigdata. This is the platform used for cloud computing by some of the pioneers like Yahoo!, Amazon, Facebook, eBay, etc.

Two major components of Hadoop are:

1. Hadoop Distributed File System (HDFS) for distributed storage
2. Mapreduce for parallel processing

Thus, Hadoop offers a reliable and scalable environment for distributed computing, which involves many clusters that houses huge data upon which necessary computing need to be carried out.

1.3 HDFS

Hadoop includes a fault tolerant storage system called the Hadoop Distributed File System. HDFS is able to store huge amounts of information, scale up incrementally and survive the failure of significant parts of the storage infrastructure without losing data. Hadoop creates clusters of machines and coordinates work among them. Clusters can be built with inexpensive computers. If one fails, Hadoop continues to operate without losing data or interrupting work, by shifting computation to the remaining machines in the cluster.

HDFS manages storage on the cluster by breaking incoming files into pieces, called "blocks," and storing each of the blocks redundantly across the pool of servers as indicated in Figure 1. In the common case, HDFS stores three complete copies of each file by copying each piece to three different servers.

In Hadoop Distributed File System (HDFS),

- Data is organized into files and directories
- Files are divided into uniform sized blocks (default 128MB)
- Blocks are replicated (default 3 replicas) and distributed to handle hardware failure
- Replication for performance and fault tolerance (Rack-Aware placement)
- HDFS exposes block placement so that computation can be migrated to data

Checksum for detecting corruption

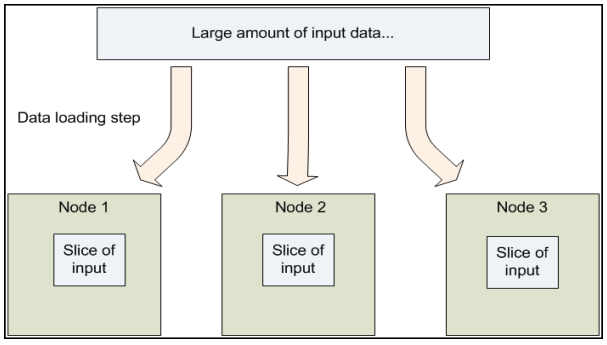

Figure 1: Data distribution

HDFS has several useful features. In the very simple example shown any two servers can fail, and the entire file will still be available. HDFS notices when a block or a node is lost, and creates a new copy of missing data from the replicas it manages. Because the cluster stores several copies of every block, more clients can read them at the same time without creating bottlenecks. The one obvious objection to HDFS – is consumption of three times the necessary storage space for the files it manages – is not so serious, given the plummeting cost of storage.

1.4 Mapreduce Paradigm

Mapreduce is a programming model and software framework for writing application that rapidly process vast amount of data in parallel on large clusters of compute nodes in a reliable and fault tolerant manner.

Map/Reduce works in following manner:

1. Identifies the repetition of input data.
2. Identifies the selection criteria of each input record.
3. For each input record, determine how many entries to be emitted and how the emitted entries should be grouped and processed together.
4. Determine the aggregation logic of grouped entries.

Identify the selection criteria of each aggregated result.

2. RELATED WORK

Machine Learning [3] has enormous application in the field of data mining. As it is extensively used for large sets of data, it is very necessary to parallelize the algorithms to run them with no time. There are many parallel programming languages such as Orca, Occam ABCL, SNOW, MPI and PARLOG, but none of these approaches make it obvious how to parallelize a particular algorithm. Caregea et.al. [7] give some general data distribution conditions for parallelizing machine learning, but restrict the focus to decision trees; Jin and Agarwal [8] give a general machine learning programming approach, but only for shared memory machines. This doesn't fit the architecture of cellular or grid type multiprocessors where cores have local cache, even if it can be dynamically reallocated.

There was also a general and exact technique for parallel programming of a large class of machine learning algorithms for multicore processors based on a map-reduce paradigm. Again that was a way to achieve speedup in the multi-core system and this cannot be used on a single core system. Using Hadoop, Machine learning can be parallelized on single core systems achieving a linear speedup in execution and performance.

3. MACHINE LEARNING ALGORITHMS

KNN classifier is an instance-based learning algorithm that is based on a distance function for pairs of

observations, such as the Euclidean distance and the Manhattan distance. The basic idea of KNN is very simple. In the classification paradigm, k nearest neighbors of a test sample are retrieved first. Then the similarities between the test sample and the k nearest neighbors are aggregated according to the class of the neighbors, and the test sample is assigned to the most similar class.

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques, for example, cross validation. The special case where the class is predicted to be the class of the closest training sample (i.e., when k = 1) is called the nearest neighbor algorithm.

In **Centroid-based clustering,** clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, *k*-means clustering gives a formal definition as an optimization problem: find the K cluster centers and assign the objects to the nearest cluster center, such that the squared distances from the cluster are minimized.

ID3 Decision tree builds a tree from a fixed set of examples. The resulting tree is used to classify future samples. ID3 builds a decision tree from a fixed set of examples. The resulting tree is used to classify future samples. The example has several attributes and belongs to a class. The leaf nodes of the decision tree contain the class name whereas a non-leaf node is a decision node. The decision node is an attribute test with each branch being a possible value of the attribute. ID3 uses information gain to help it decide which attribute goes into a decision node. The advantage of learning a decision tree is that a program, rather than a knowledge engineer, elicits knowledge from an expert.

4. PROPOSED METHODOLOGY

In the proposed work the aforementioned MLA's performance is compared without hadoop using serial architecture and with hadoop using parallel architecture as shown in Figure 2. Each of the three algorithm's workflow has been depicted in Figure 3. It can be seen that the task is accomplished by a sequence of jobs.

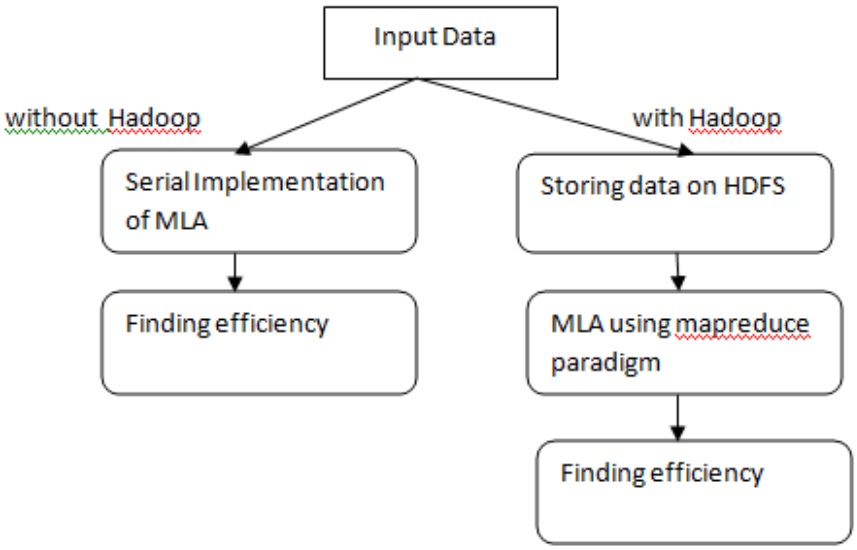

Figure 2: MLA architecture with and without Hadoop

As shown in the Figure 3, the input data for the KNN algorithm is taken from the HDFS. The overall task is divided into four jobs. The first[t] job separates the missing value sets. The second job generates a separate sequence file for each of the missing rows. The third job updates the missing value by calculating the distance. The last job is to merge the missing and non-missing rows and generate the output with no missing values.

The input data for the K-means algorithm is taken from the HDFS. The overall task is divided into three jobs. The first job generates a file of centroids and data object pairs from the input file. The next job associates centroids and the data objects to form a cluster and finally mean is calculated and a new centroid list is obtained. The jobs are recursively called until the previous centroids become equal to the newly generated centroid list.

The ID3 algorithm is implemented by a sequence of four jobs. The Total entropy is calculated in the first job, followed by information gain in the next job. The third job determines the attribute with highest information gain and the input data is divided to get the sub-node data. The jobs are recursively executed on the sub-node data until entropy becomes 0.

5. PERFORMANCE ANALYSIS

For about a decade it was considered that distributed computing is more complex to handle than expanding memory of single node cluster. This is because the inter-process communication (IPC) to be used to communicate with the nodes was tedious to implement as the code would run longer than the computation procedure itself. But now apache hadoop offers a more scalable and reliable platform to implement distributed computing. Performance comparison chart in Figure 4 shows the efficiency between two architectures. Through this it can be analyzed that Map/Reduce algorithm run on hadoop influences the performance significantly while handling huge data set stored on different nodes of a multi-node cluster.

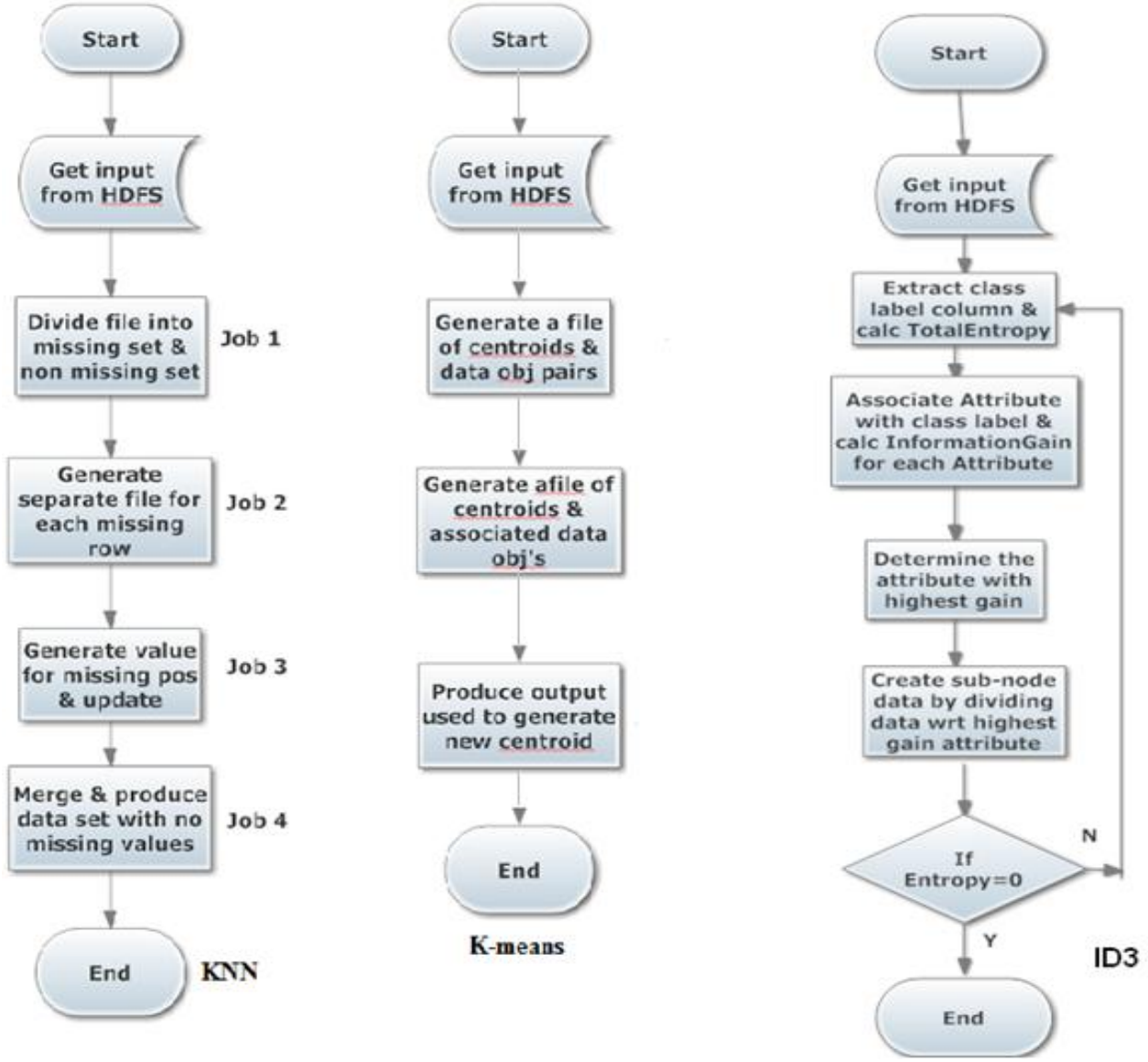

Figure 3: Three MLA algorithm implementation on Hadoop

6. CONCLUSION

In this paper we propose an approach for the implementation of recursive Machine learning algorithms in parallel fashion using a distributed framework like Hadoop. Performance analysis of MLA's on legacy system is compared with that of Hadoop distributed system. It was found that the algorithms perform productively for Bigdata. The application aims at providing efficient implementation of machine learning algorithms. The key attributes of the system was successfully achieved by improving the timing efficiency. The system provides opportunity to further career growth by moving across technology, domain and streams.

Figure 4: Comparison of timing for Bigdata

Algorithm	Data size	Serial implementation (without hadoop)	Parallel implementation (with hadoop)
K-NN	4kb	30s	30s
	1mb	10mn	3min
	2mb	15min	3min
	10mb	1hr	3min
K-Means	4kb	30s	1min
	1mb	4min	3min
	2mb	5min	3min
	10mb	Crashed	10min
ID3 Decision tree	4kb	Takes longest time even for small data	Takes longest time even for small data
	1mb		
	2mb		
	10mb		

REFERENCES

[1] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters". In Proc. ACM OSDI, 2004

[2] Doug Cutting et al. "About hadoop", http://lucene.apache.org/hadoop/about.html.

[3] Tom M. Mitchell: Machine Learning McGraw Hill, 1997.

[4] Yang Liu, Xiaohong Jiang, Huajun Chen, Jun Ma, and Xiangyu Zhang, "MapReduce-Based Pattern Finding Algorithm Applied in Motif Detection for Prescription Compatibility Network", APPT 2009, LNCS 5737, pp 341-355, Springer 2009.

[5] Seung-Jin Sul, Andrey Tovchigrechko, "Parallelizing BLAST and SOM algorithms with MapReduce-MPI library", In Proc. of IEEE International Parallel & Distributed Processing Symposium, 2011.

[6] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst, "HaLoop: Efficient iterative dataprocessing on large clusters", PVLDB, Volume 3, issue 1, pp 285–296, 2010.

[7] Jiawei Han and MichelineKamber, 'Data Mining Concepts and Techniques', Second Edition MorganKaufman Publishers 2006.

[8] Shoshani A, Agarwal D. A, Draney B. R, Jin G, Butler G. F "Deep scientific computing requires deep data", IBM Journal of Research and Development, Volume 48 ,Issue 2, pp 209-232, ACM, March 2004.

[9] Mapreduce: A flexible data processing tool, ACM Communications, Volume 53, pp 72-77, 2010.