Multiuser Job Scheduling for Distributed Machine Learning

K. Sireesha
M. Tech student CSE Department,
JNTUCE Anantapur
Anantapur, India

Abstract— **Machine Learning (ML) is part of computer science which explores the study and construction of algorithms that can learn from and make prediction on data. Emerging trends in machine learning is integration of big data in multi user environment. Multi user environments deal with a meta data or big data. For processing big data in multi user environments requires more efforts. Big data processing needs to be done in a distributed environment where thousands of nodes work together along with data centers, cloud computing resources and distributed file systems. Hadoop and spark are the best examples of multi user environments in distributed machine learning. The programming approach used in such environment is known as MapReduce which is best used to let multiple nodes participate in processing given job. It exploits the power of Graphic Processing Unit (GPU) and supports parallel processing of data. Machine learning algorithms focused on processing huge amount of data. However, there needs to be a framework that can improve the support for big data processing using machine learning algorithms. This paper presents the implementation of a framework that can support information retrieval and natural language processing. The framework takes big data as input and provides simulated environment for demonstration of functionalities of mapper and reducer. The application demonstrates the proof of concept.**

Keywords— ***Big data, distributed programming framework, MapReduce, machine learning.***

I. INTRODUCTION

Traditional data processing applications deals with a large or complex data sets, such data sets terms are referred in a single word called as "Big data". Big data has certain characteristics namely "Volume, Variety and Velocity". The first attribute is related to the quantity of data which is very huge in nature. The second attribute refers to the fact that data is available in different formats like structured, semi-structured and unstructured. The third attribute that is Velocity refers to the data that is in transit. When live data is being collected from different sources, such data can exhibit velocity characteristic. Big data when accumulated becomes a valuable source for extracting business intelligence. Unfortunately the process of big data cannot be directly done in the local machines due to resource constraints. It is possible with distributed programming frameworks that are associated with huge data centres and cloud computing paradigm.

Mining of big data is important for processing of data or for scheduling of data for multiple users. Mining is a process of extracting trends or patterns that are latent in the given dataset. The 'Biased conclusion' condition occurs if the data processing is done without mining.

A. Need for BigData Mining

When big data is not considered for processing (when partial data is mined), it may not result in comprehensive intelligence. Big data is dynamic in nature and it is streamed from multiple sources. In such case, processing some part of data or not including the live data can provide conclusions that are not accurate.

B. Bigdata Evolution

Big data evolution started in early 1968 in the form of hierarchical database where Online Transactional Processing(OLTP) is performed. Online Transactional Processing is performed and the analysis was manually made on historical data. Later in 1983 data warehousing concept came into existence which paved way for having historical data in a way that can be mined easily through programmable interfaces.

With Online Analytical Processing the concept of obtaining business intelligence and making well informed decisions came into existence. It continued till 1990. Later in the year 2000 to 2010 gradually steam computing came into existence and transferred into big data mining. Now the real time analytical processing is possible using big data. The big data when harnessed can provide comprehensive business intelligence.

C. MapReduce Programming

Map Reduce Programming is a new programming paradigm which is widely used in the distributed environment. The applications like Gmail, Face book are some of the examples where huge amount of data is being processed in distributed environment. The new programming paradigm used in such environments is MapReduce. This programming model can leverage distributed environment and the presence of Great Point Average (GPA) and data centres. It can exploit the power of multiple computers working together in distributed environment.

MapReduce programming supports user programs to interact with a master node which in turn assigns job to multiple worker nodes. The worker nodes perform the task of Map, Reduce and provide required output. Map functionality is done by worker nodes. It is based on the application specific work. The input files are split into multiple files and given to mappers which are running in multiple worker nodes. The output of the mappers will written to local storage and then the Reducer nodes take the output of mappers and perform further processing and write final output to files in a distributed file system.

Machine learning algorithms that are existing may not support MapReduce programming and therefore it is important to have them in the new programming model. This paper includes building a framework that provided simulated environment to support map and reduce programs in order to have machine learning on big data. It focused on processing big data which is in the form of documents. Information retrieval and natural language processing concepts are used for machine learning and produce Term Frequency/Inverse Document Frequency (TF/IDF) and other intermediary results. The framework needs much improvement to be more useful. The next sections of the paper are as follows. Section II provides review of literature. Section III represents multiuser job scheduling and Section IV explains experimental results while section V concludes the paper.

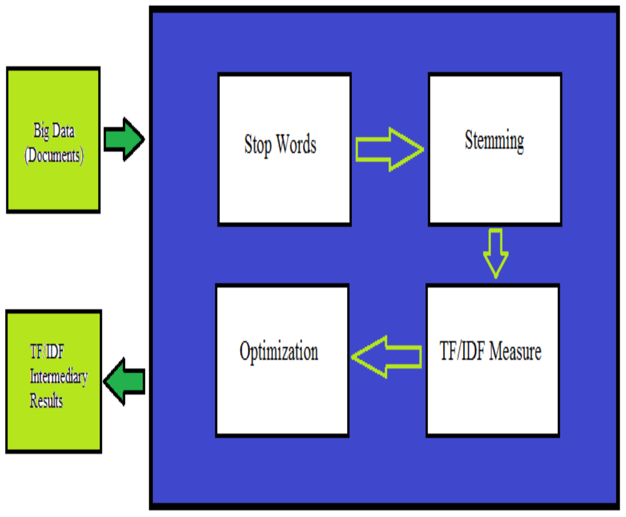
II. REATED WORK

This section provides review of literature on machine learning algorithms that support data mining. Several contributions came into existence on machine learning, explored in [1, 2, 3, 4, 5, 6]. The technologies explored in those researches are not widely visible in Natural Language Processing(NLP) and data mining in general. They are not visible in big real world applications. Scalability and accurate learning process issues might be the possible reason. Transformation of ML algorithms is needed for academic implementation to big real world usage. For making ML algorithms for future Big learning and scalability are to be kept in mind. It is essential to have such scalable algorithms to process big data.

Hadoop [7] is one of the distributed programming frameworks that can support MapReduce programming and it is being used widely by enterprise applications, which makes use of distributed file system and disk support for scalable operations. Therefore performance can exceeded when compared with its in-memory counterparts explored in [8, 9]. Some other systems that are explored in [10, 11] that provide low level programming interfaces which are powerful and versatile. They cannot provide high-level and general purpose building blocks needed in distributed programming approach such as scheduling, model partitioning, and managed communication. One of the graph-centric platforms is Pregel [12] that can help in efficient partitioning of models that are based on graph with several in-built scheduling and consistency mechanisms. Here in this paper framework for supporting machine learning algorithms in order to big data processing. As forth it supports information retrieval and natural language processing in a multi-user environment.

III. FRAMWWORK FOR DISTRIBUTED MACHINE LEARNING

The machine learning framework is meant for supporting machine learning approaches on big data. It provides multi-user environment for scheduling algorithms which takes care of scheduling jobs in distributed systems. Any machine learning algorithm can support by this framework. Till now, it supports natural language processing and information retrieval that can help improve. Figure 1, represents the framework for job scheduling over multi user environments in distributed systems. It set of documents as input and performs operations



natural language processing. Before going to NLP operations the dataset is subjected to pre-processing in the form of stemming and stop words. When they are completed, it makes use of measures to find out TF/ IDF values and presents results. Different similarity measures are supported in order to have accurate results.

Algorithm: Algorithm for Multi-user job scheduling

```
Algorithm: Multi-user job scheduling algorithm
Inputs: Jobs
Outputs: Scheduled jobs

01  Initialize job j
02  Initialize jobs vector J
03  Initialize node n
04  Initialize t1 to 15 milliseconds
05  Initialize t2 to 15 milliseconds
06  IF node n is busy THEN
07      Set j.wait=t1
08  END IF
09  IF n has free map slot THEN
10      Populate jobs to J
11      For j in J
12          IF j.wait=0 an n has still free map slot THEN
13              Assign j to n
14          END IF
15          IF j.wait =0 and n has no free map slot THEN
16              j.wait=t2;
17          END IF
18      END FOR
19  END IF
```

The algorithm firstly initializes the jobs vector and time. And it starts the if condition for nodes 'n' and sets the waiting time t1. Then it starts for map slots and starts the scheduling of jobs over the distributed system. If all the jobs are allotted then the algorithms are going to start calculation of Term Frequency (TF), and it shows the results that how many terms are present in the particular documents and how many words are there in the document. Scheduling that is based on waiting time optimizes performance of the environment in case of multiple users giving jobs simultaneously.

IV. EXPERIMENTAL RESULTS

The prototype application to demonstrate the MapReduce simulation implemented using Java is in the paper. For simulating distributed programming environment Mapper and Reducers are implemented. It supports multi-user job scheduling and processing of big data. NLP techniques are used to process complex of documents. TF/IDF are metrics used to process documents. The user can choose the location of sample document to be processed. Large numbers of text documents are provided as input. The application loads documents and perform natural language processing. The work proceeds using Map and Reduce programs. The process is split into multiple modules and the mappers are able to process the splitted data. Then the reducer gives the results and produces final output.

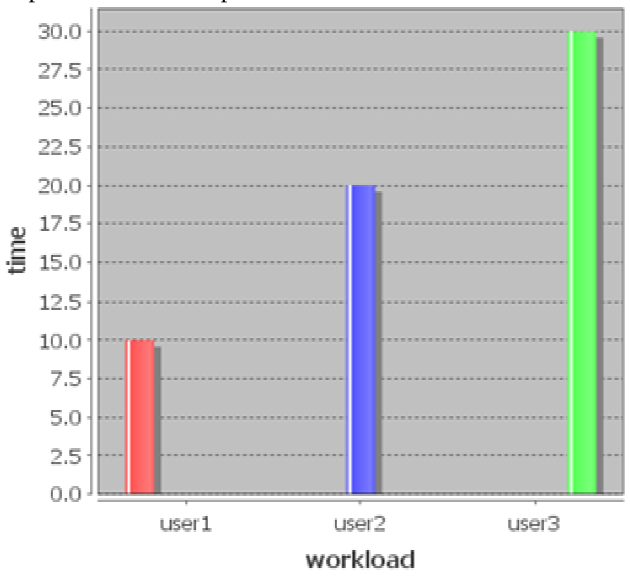


Figure 2. Workload of multiple users vs. time

Figure 2, shows the workload of multiple users and the time taken for job scheduling. The horizontal axis represents user's work load, the vertical axis shows the time taken to process the workload in the presence of multiple concurrent users.

V. CONCLUSIONS AND FUTURE WORK

The paper presents the machine learning approaches for big data in muti user environments. A frame work for job scheduling of multi user environments gives a benefit for big data analysis. MapReduce programming paradigm is a frame work for parallel data processing in multi user environments. Therefore it is essential to have machine learning algorithms in MapReduce approach. The framework takes big data as input and provides simulated environment demonstrating the functionalities of mapper and reducer. The empirical results represents the performance of mapreducer in multiple environments. The work can be extended for optimization of scheduling cost .

REFERENCES

[1] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin, "Parallel coordinate descent for l1-regularized loss minimization," in Proc. Int. Conf. Mach. Learn., 2011, pp. 321–328.

[2] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in Proc. Adv. Neural Inf. Process. Syst., 2011, pp. 873–881.

[3] M. Zinkevich, J. Langford, and A. J. Smola, "Slow learners are fast," in Proc. Adv. Neural Inf. Process. Syst., 2009, pp. 2331–2339.

[4] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1232–1240.

[5] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," J. Mach. Learn. Res., vol. 14, pp. 1303–1347, 2013.

[6] S. A. Williamson, A. Dubey, and E. P. Xing, "Parallel Markov chain Monte Carlo for nonparametric mixture models," in Proc.Int. Conf. Mach. Learn., 2013, pp. 98–106.

[7] T. White, Hadoop: The Definitive Guide. Sebastopol, CA, USA: O'Reilly Media, 2012.

[8] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, "Distributed GraphLab: A framework for machine learning and data mining in the cloud," in Proc. VLDB Endowment, vol. 5, pp. 716–727, 2012.

[9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in Proc. 2nd USENIX Conf. Hot Topics Cloud Comput., 2010, p. 10.

[10] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in Proc. 11th USENIX Conf. Operating Syst. Des. Implementation, 2014, pp. 583–598.

[11] R. Power and J. Li, "Piccolo: Building fast, distributed programs with partitioned tables," in Proc. USENIX Conf. Operating Syst. Des. Implementation, article 10, 2010, pp. 1–14.

[12] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 135–146.