A Study on MapReduce – Application and challenges

Dr. Manju Sharma[1], Ms. Ashmita[2]
[1]*Assistant Professor, Department of Computer Science, College of Engineering & Computer Science, Jazan University, Jazan, Kingdom of Saudi Arabia,*
[2]*Sushant University, Gurugram, Haryana-India*

***Abstract*—MapReduce has Its own Classification Association Rules, Functional Point i9s a method used (CARs). There ars some completion of the rules of mining, in that distributed rule trimming was mainly popular. The technique is used in handling large datasets if we have only few set of hardware. Artificial Neural Network (ANN) was main and most popular used in category of sentiment. There are Comparisons that are made between - Back propagation Neural Network (BPN), Probabilistic Neural Network (PNN) and Homogeneous Ensemble of PNN (HEN). The method of Vote is designed to use the characteristics of three classifiers which is: the Naive Bayes (NB), Support Vector Machine (SVM) and the Bagging. There are other techniques available for sentiment and parameter optimization for individual classifiers. We mainly implement a three-step methodology.**

***Index Terms*—Genetic Algorithm, ANN, Synonyms-Based Term (SBT), Stemming Algorithm.**

I. INTRODUCTION

Map Reduce can be compared to generic language for dataset twhich is in existence in for emotion carrying words. The new system Synonyms-Based Term (SBT) weighting implemented. It is a used to analyze the messages for sending and receiving on social media. It is used to receive messages from Facebook to seriate them for the polarity of the sentiments it contains. For separating the feelings, researcher's implements a way which is cross between lexical analysis and machine learning. The sentiment included in the tweets was the assistance of a lexicon-enhanced sentiment analysis which is an, method that made a rule-based categorization system. To determine its strategy it's a successful and as per the user perspectives evaluated for three other areas. It is found that it was demonstrated the suggested method to avoid the methodologies that were previously being used and demonstrate performing in it.

II. LITERATURE REVIEW

Big Data Analytics is technologies that can store and analyze data. After having the interest in understanding and exploring the ever-increasing data it has been conducted on the uses of big data. On the subjective and speculator aspects of Big Data and Analytics it is observed in this paper. It provides information on getting data in the organization. It provides a potential result of the study which is been carried out in various centers. We will discuss the possibilities and challenges for researchers and IT analysts to make informed decisions that will ultimately improve the performance and productivity of organizations.

The Algorithm
Generally MapReduce paradigm is based on sending the computer to where the data resides.
MapReduce program consist of three stages which is - Map stage, Shuffle stage, and Reduce stage.

In Map stage − the map or mapper's job is to process the input data. Generally the input data is in the form of file or directory which is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage is the combination of the Shuffle and the Reduce stage. The job of Reducer is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

Programming Models

'The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the Map Reduce library expresses the computation as two functions: Map and Reduce. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The Map Reduce library groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function. The Reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges together these values to form a possibly smaller set of values. Typically just zero or one output value is produced per Reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory [1]

Example: 'Consider the problem of counting the number of occurrences of each word in a large collection of documents. The user would write code similar to the following pseudo-code: map(String key, String value): // key: document name // value: document contents for each word w in value: EmitIntermediate(w, "1"); reduce(String key, Iterator values): // key: a word // values: a list of counts int result = 0; for each v in values: result += ParseInt(v); Emit(AsString(result)); The map function emits each word plus an associated count of occurrences (just '1' in this simple example). The reduce function sums together all counts emitted for a particular word. In addition, the user writes code to fill in a mapreduce specification object with the names of the input and output files, and optional tuning parameters. The user then invokes the MapReduce function, passing it the specification object. The user's code is linked together with the MapReduce library (implemented in C++). Appendix A contains the full program text for this example.'[1]

More Examples

'Distributed Grep: The map function emits a line if it matches a supplied pattern. The reduce function is an identity function that copies the supplied intermediate data to the output.

1. Count of URL Access Frequency: The map function processes logs of web page requests and outputs (URL, 1). The reduce function adds together all values for the same URL and emits a (URL, total count) pair.
2. Reverse Web-Link Graph: The map function outputs (target, source) pairs for each link to a target URL found in a page named source. The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: (target, list(source)).
3. Term-Vector per Host: A term vector summarizes the most important words in a document or a set of documents as a list of (word, frequency) pairs. The map function emits a (hostname, term vector) pair for each input document. The reduce function adds these term vectors together, throwing away infrequent terms, and then emits a final (hostname, term vector) pair.
4. Inverted Index: The map function parses each document and emits a sequence of (word, document ID) pairs. The reduce function accepts all pairs for a given word, sorts the corresponding document IDs, and emits a (word, list(document ID)) pair. The set of all output pairs forms a simple inverted index.'[4]

III. SCHEDULING

'MapReduce uses a block-level runtime scheduling with a speculative execution. A separate Map task is created to process a single data block. A node which finishes its task early gets more tasks. Tasks on a straggler node are redundantly executed on other idle nodes. Hadoop scheduler implements the speculative task scheduling with a simple heuristic method which compares the progress of each task to the average progress. Tasks with the lowest progress compared to the average are selected for re-execution. However, this heuristic method is not well suited in a heterogeneous environment where each node has different computing power. In this environment, even a node whose task progresses further than others may be the last if the node's computing power is inferior to others. Longest Approximate Time to End(LATE) scheduling is devised to improve the response time of Hadoop in heterogeneous environments'[2] [3].

IV. MAPREDUCE VARIANTS

'Map-Reduce-Merge is the first that attempts to address join problem in the MapReduce framework [5]. To support binary operations including join, Map-ReduceMerge extends MapReduce model by adding

Merge stage after Reduce stage. Map-Join-Reduce is another variant of MapReduce framework for one-phase joining [7]. The authors propose a filtering-join-aggregation model that adds Join stage before Reduce stage to perform joining within a single MR job. Each mapper reads tuples from a separate relation which take part in a join process. After that, the mapped outputs are shuffled and moved to joiners for actual joining, then the Reduce() function is applied. Joiners and reducers are actually run inside the same reduce task. An alternative that runs MapJoin-Reduce with two consecutive MR jobs is also proposed to avoid modifying MapReduce framework. For multi-way join, join chains are represented as a left-deep tree. Then previous joiners transfer joined tuples to the next joiner that is the parent operation of the previous joiners in the tree. For this, Map-Join-Reduce adopts one-to-many shuffling scheme that shuffles and assigns each mapped outputs to multiple joiners at a time.'[2]

It is a user-provided approach that processes a key or value to generate collection of intermediate key or value of pairs. A user can be provided a reduced function to merges intermediate values related to the same intermediate key.

'MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real-world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.'[8]

V. CONCLUSION

We can say that it is a programming model related to implementation of process that generate large amount of data sets. The MapReduce programming model has been successfully used at Google for many different purposes. The model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing. Second, a large variety of problems are easily expressible as MapReduce computations. For example, MapReduce is used for the generation of data for Google's production web search service, for sorting, for data mining, for machine learning, and many other systems. Third, we have developed an implementation of MapReduce that scales to large clusters of machines comprising thousands of machines.

REFERENCES

[1] MapReduce: Simplified Data Processing on Large Clusters Jeffrey Dean and Sanjay Ghemawat jeff@google.com, sanjay@google.com Google, Inc.

[2] Parallel Data Processing with MapReduce: A Survey Kyong-Ha Lee Yoon-Joon Lee Department of Computer Science KAIST bart7449@gmail.com yoonjoon.lee@kaist.ac.kr Hyunsik Choi Yon Dohn Chung Department of Computer Science and Engineering Korea University hyunsik.choi@korea.ac.kr ydchung@korea.ac.kr Bongki Moon Department of Computer Science University of Arizona bkmoon@cs.arizona.edu, SIGMOD Record, December 2011 (Vol. 40, No. 4)

[3] M. Zaharia et al. Improving mapreduce performance in heterogeneous environments. In Proceedings of the 8th USENIX OSDI, pages 29–42, 2008.

[4] Insights from papers — Google MapReduce: Simplified Data Processing on Large Clusters, Hemant Gupta.

[5] H. Yang et al. Map-reduce-merge: simplified relational data processing on large clusters. In Proceedings of the 2007 ACM SIGMOD, pages 1029–1040, 2007

[6] M.Carro, Alvaro Ortigosa Jose M.Martín Rosa, "Sentiment analysis in Facebook and its application to e-learning," Computers in Human Behavior, vol. 31, pp. 527 - 541, 2014.

[7] D. Jiang et al. Map-join-reduce: Towards scalable and efficient data analysis on large clusters. IEEE

Transactions on Knowledge and Data Engineering, 2010.

[8] https://research.google.com/archive/mapreduce.html.MapReduce: Simplified Data Processing on Large Clusters Jeffrey Dean and Sanjay Ghemawat