A Survey on Parallel Rough Set Based Knowledge Acquisition Using MapReduce from Big Data

Sachin Jadhav, Shubhangi Suryawanshi

Abstract— **Nowadays, the volume of data is growing at an nprecedented rate, big data mining , and knowledge discovery have become a new challenge in the era of data mining and machine learning. Rough set theory for knowledge acquisition has been successfully applied in data mining. The MapReduce technique, received more attention from scientific community as well as industry for its applicability in big data analysis. In this paper we have presented working and execution flow of the MapReduce Programming paradigm with Map and Reduce function.This paper also describes the various Mapreduce implementation with their pros and cons such as Google's MapReduce, ,YARN, Twister, phoenix etc. In this work we have also briefly discussed different issues and challenges that are faced by MapReduce while handling the Big data. And lastly we have presented some advantages of the Mapreduce Prograaming model.**

Index Terms—**: Big Data, Big Data Acquisition ,MapReduce, Online Processing, Privacy, Rough Sets ,Security**

I. INTRODUCTION

In todays Internet world, recent developments in the Web, social media, sensors and mobile devices have resulted in the large explosion of real time data . For example, today on social networking sites such as Facebook ,Twitter has more than one billion users. Everday with over 618 million active users generating more than 500 terabytes of new data . Nowdays, With the fast increase and update of big data in real-life applications, it brings a new challenge to quickly extract the useful information from real time data with the help of data mining techniques. Traditional data processing and storage approaches are facing many challenges in addressing Big Data demands.The term "Big Data" consist of large and complex amount of data sets made up of a variety of structured and unstructured data which are too big, too fast, and too hard to be managed by traditional techniques.

With the development of information technology, large amount of data are collected in different multiple formats from various multimedia devices. For the purpose of processing big data, Google developed a software framework called MapReduce to support large distributed data sets on clusters of computers which is effective to analyze large amounts of data. MapReduce is one of the most important cloud computing techniques.

MapReduce is a highly scalable programming model which is capable for processing large volumes of data by parallel execution on a large number of commodity computing nodes. MapReduce was popularized by Google , but today it has been implemented in many open source projects (for example. Apache Hadoop). The popularity of MapReduce is increased due to its high scalability, fault-tolerance, simplicity and independence from the programming language or the data storage system.

In the Big Data community, MapReduce has been seen as one of the key enabling approaches for meeting the continuously increasing demands on computing resources imposed by massive data sets.

.

II. WORKING OF MAPREDUCE PROGRAMMING MODEL

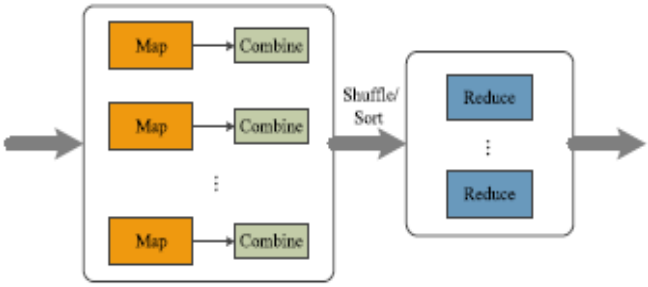

Figure 2.1 MapReduce programming model

MapReduce is a programming model, above figure 2.1 shows the system architecture of Mapreduce programming model . The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the MapReduce library expresses the computation by means of two necessary functions Map and Reduce, and one optional function Combine. Here, the parallel execution of each primitive is managed by the system runtime.

Sachin Jadhav *ME (Computer Engineering) Savitribai Phule University of Pune, Maharashtra(INDIA) Pune, Maharashtra(INDIA), 07588229311*

Shubhangi Suryawanshi*, M.E.(Computer). Asst.Prof. GHRIET pune, Maharashtra(INDIA)09970082182.*

.

ISSN: 2278 – 1323 All Rights Reserved © 2014 IJARCET

MAP
Map: (k1 , v1) → (k2 , v2)*.

The Map function takes an input key/value pair (k1 ,v1) and it produce outputs as a list of intermediate key/value pairs (k2 ,v2). The MapReduce library groups together all intermediate values associated with the same intermediate key and transforms them with the Combine/Reduce function.

COMBINE
Combine (optional) is local Reduce. It accepts a key and a set of values for that key from local Map. Then, it merges together these values to form a possibly smaller set of values and transforms them with the Reduce function.

REDUCE
Reduce: (k2 , v2 *) → (k2 , v3)*.

The Reduce function takes all values associated with the same key and produces
a list of key/value pairs. 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.

A. MapReduce Flow
MapReduce is a programming approch for processing large data sets in distributed environments . In the MapReduce paradigm, the Map function performs filtering and sorting of the data and the Reduce function perform the opertaion on the data such as grouping and aggregation

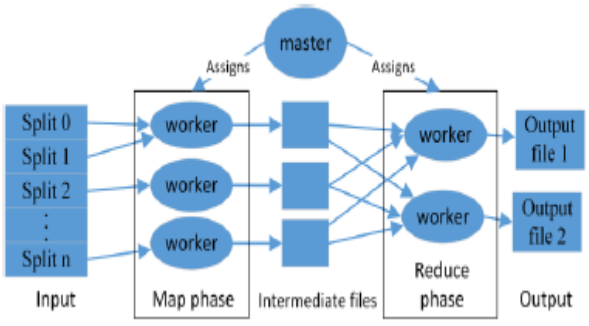

Figure 2.2: MapReduce flow

Figure 2.2 describe the working flow MapReduce programming approch . Here, One node is elected to be the master node which is responsible for assigning the work, while the rest of the nodes are worker nodes. The input data is divided into n splits and the master assigns splits to Map workers. Each worker processes the corresponding input split, generates key/value pairs and writes them to intermediate files (on disk or in memory).
The master node identifies the Reduce workers about the location of the intermediate files and the Reduce workers read data, process it according to the Reduce function, and finally, write data to output files.

The major contribution of the MapReduce programming paradigm is to achieve the scalability .In the MapReduce programming paradigm, the Map or Reduce task is divided into a high number of jobs which are assigned to nodes in the network.

Reliability is achieved by reassigning any failed node's job to another node. A well known open source MapReduce implementation is Hadoop which implements MapReduce on top of the Hadoop Distributed File System (HDFS).

III. DIFFERENT IMPLEMENATATION OF MAPREDUCE:

A. Google's MapReduce:

For handling the large clusters of networked machines Google developed MapReduce programming model. The MapReduce library originally handles parallelization and data distribution. Here , Data is distributed and saved on local disks of networked machines. Google File System (GFS) is a distributed file system which is used to supervise the data stored across the cluster. It creates duplicate of data blocks on multiple nodes for enhanced reliability and fault tolerance.MapReduce programming paradigm is highly scalable, and therefore it can be run on clusters consist of thousands of low-cost machines, built on untrustworthy hardware.

B. Hadoop:

Hadoop is a MapReduce implementation by Apache. Hadoop is an open-source implementation.
The architecture of Hadoop is same as Google's implementation. In Hadoop, data is distributed across the various machines in network using the Hadoop Distributed File System (HDFS). It distributes data on computers approximately to the cluster, and creates numerous replicas of data blocks for providing reliability and fault tolerance.
There are two types of nodes in HDFS:
1.DataNodes
2. NameNode.

Typically, a Hadoop deployment consist of a single NameNode, which is the master node and a set of DataNodes, which serve as slaves.
DataNodes are used to store the blocks of data and to serve them on request over the network. By default, data blocks are replicated in HDFS, for fault-tolerance and higher chance of data locality, when running MapReduce applications.

The NameNode is unique in an HDFS cluster and it is responsible for storing and managing metadata. It stores metadata in memory, thus we can limit the number of files that can be stored
by the system, according to the node's available memory.

Generally, a Hadoop application consists of one or more Hadoop jobs[4]. When running a Hadoop job, it allocates resource on the basis of the Hadoop scheduler. Then Map workers and Reduce workers are launched and start to work.

First, the intermediate data generated by Map function is written to a local file system . These processes increase the running time on Hadoop. Even if the input data is empty, it would run a system for a few seconds. Therefore, the application on Hadoop is always slow if the data is not enough big.

Advantages of HADOOP:

Hadoop has a better speedup as the size of data set increases. Haddop provide good fault tolerance , and it can also help to process very large-scale data. We can easily deploy Hadoop on local cluster and Public Cloud, such as Amazon EC2 and Microsoft Azure both support Hadoop by using Amazon Elastic MapReduce and HadoopOnAzure .

C. YARN(Yet Another Resource Negotiator)

In order, to improve the cluster utilization and scalability Hadoop community redesigned the architecture of Hadoop. The new design is called YARN [4].

YARN is included in the latest Hadoop release and its main aim is to allow the system to serve as a general data-processing framework. It supports programming models other than MapReduce, it also improve the scalability and resource utilization.

YARN makes no changes to the programming model or to HDFS. It consists of a re-designed runtime system, aiming to eliminate the pitfalls of the master-slave architecture. In YARN, the responsibilities of the JobTracker are divided into two different processes, the ResourceManager and the ApplicationMaster.

ResourceManager

The ResourceManager handles resources dynamically, using the notion of containers, instead of static Map/Reduce slots. Containers are configured based on information about available memory, CPU and disk capacity. It also has a pluggable scheduler, which can use different strategies to assign tasks to available nodes.

ApplicationMaster

The ApplicationMaster is a framework-specific process, it allows other programming models to be executed on top of YARN, such as MPI or Spark . It supervises the scheduled tasks as well as negotiates resources with the ResourceManager and

D. TWISTER

In Twister [7] , users first partition the data manually or automatically by a specified script, and send them to different compute nodes. Then, it generates a configuration file thta inform to the compute nodes to process the local data in Map phase. Inorder to achieve the better performance, Twister handles the intermediate data in the distributed memory of the worker nodes.

Twister also provides fault tolerance support only for iterative MapReduce computations rather than non-iterative MapReduce computations. These handling methods in Twister gives us better performance than Hadoop but with worse fault tolerance. The program execution on Twister is fast as compared to the Hadoop and Phoenix . It also has a Public Cloud version: Twister4Azure .

E. Phoenix:

The implementations of Phoenix are basically the same as that of original MapReduce programming paradigm. But instead of large clusters, it is designed for shared-memory systems. Here, the runtime in Phoenix uses P-threads that generate parallel Map or Reduce tasks, and schedules tasks dynamically to available processors .

In Phoenix[8], addition to Map and Reduce functions, the user provides a function that partitions the data before each step, and a function that implements key comparison. The programmer calls phoenix scheduler to start the MapReduce process. The function takes arrangement structure as an input, in which the user specifies the user-provided functions, pointers to input/output buffers and other options. The scheduler controls the runtime, and manages the threads that run all the Map and Reduce tasks.

The application running on Phoenix is fast, and it has an excellent speedup when the size of data size is not large. Because of its characteristic, it is more suitable for smaller data sets on single multi-cores compute node. If the size of data set is larger than memory, Phoenix would be failed with an out of memory error.

IV.ISSUES AND CHALLENGES OF MAPREDUCE

Here, we have described various issues and challenges that are faced by MapReduce when handling Big Data.

A. Issues

1. Performance Issues

Even though MapReduce provides scalability, fault-tolerance and capability of processing large amounts of data, it requires more query execution time . Normally, this query execution time is higher than what modern DBMSs offer and prevents interactive analysis.

In MapReduce programming approch performance is highly dependent on the nature of the application, but is also influenced by inherent system characteristics and design choices.

Implementation of Mapreduce spent more time in task initialization, scheduling, coordination and monitoring. MapReduce programming paradigm does not support data pipelining or overlap of the Map and the Reduce phases.

2. Programming Model Issues

In order to develop efficient MapReduce applications we need advanced programming skills and deep understanding of the system architecture.

In MapReduce data need to be uploaded to the file system and if the same dataset needs to be analyzed multiple times, it has to be read every time.

The computation steps are fixed and applications need to perform the map-shuffle sort-reduce sequence. Since MapReduce operators are stateless, MapReduce implementations of iterative algorithms require manual management of state and chaining of iterations.

3. Configuration and Automation Issues

There are different configuration parameters have to set when deploying a MapReduce cluster.

ISSN: 2278 – 1323 All Rights Reserved © 2014 IJARCET

In order to implement the MapReduce we need to include the different parameter such as number of parallel tasks, the size of the file blocks and the replication factor. Proper tuning of these parameters requires knowledge of both available hardware and workload characteristics, while misconfiguration might lead to inefficient execution and under utilization of resources

B. Challenges:

1. Data Storage

-MapReduce programming model is schema-free and index-free, this provides great flexibility and it enables MapReduce to work with semi-structured and unstructured data.

As soon as data is loaded MapReduce start to run . However, the lack of indexes on standard MapReduce may result in poor performance as compare to relational databases.

-Another challenge is the lack of a standardized SQL-like language. Therefore , one direction of research is concerned with providing SQL on top of MapReduce. An example of this category is Apache Hive which provides an SQL-like language on top of Hadoop.

2. Security

Accountability and Auditing are major security issues that may create a problem while working with MapReduce and Big Data.

Accountability[3] is the ability to know when someone performs an action and to hold them responsible for that action and is often tracked through auditing. In MapReduce programming approch accountability is only provided when the mappers and reducers are held responsible for the tasks they have completed .

2.Another security challenge presented to MapReduce and Big Data is that of providing access control, which can be shown through volume, variety and velocity of Big Data's properties.

When dealing with a large volume of information, work performed on that information is likely to require access to multiple storage locations and devices. Therefore, multiple access requirements will be required for any one task.

When dealing with data that has a large variety, semantic understanding of the data should play a role in the access control decision process .

Lastly, the velocity requirement of MapReduce and Big Data requires that whatever access control approach is used must be optimized to determine access control rights in a reasonable amount of time.

3. Privacy

Privacy must be considered whenever large amounts of information are used. Processes such as data mining and predictive analytics can discover or deduce information linkages.

Privacy protection requires an individual to maintain a level of control over their personal information. This control can be provided through transparency and allowing input from the data provider.

Transparency[3] is provided to an individual user by the knowledge of how private information is collected, what private information is collected, how the private information is being used, and

who has access to it. This can be very difficult when dealing with a large number of mappers and reducers that MapReduce often requires.

4. Online Processing

The Velocity dimension, as one of the Vs used to define Big Data property,it brings many new challenges to traditional data processing approaches and especially to MapReduce. Handling Big Data velocity often requires applications with online processing capabilities, areas such as financial fraud detection and algorithmic trading have been highly interested in this type of solutions.

The MapReduce programming approch[3] is not an appropriate solution for this kind of low-latency processing because, MapReduce computations are batch processes that start and finish, while computations over streams are continuous tasks that only finish upon user request.

- The inputs of MapReduce computations are snapshots of data stored on files, and the content of these files do not change during processing. Conversely, data streams are continuously generated and unbounded inputs .
- In order to provide fault tolerance, most of MapReduce implementations [3], such as Google's and Hadoop , write the results of the Map phase to local files before sending them to the reducers. In addition, these implementations store the output files in distributed and high-overhead file systems such as Google File System or HDFS respectively). This extensive file manipulation adds significant latency to the processing pipelines.

- Not every computation can be efficiently expressed using the MapReduce programming paradigm, and the model does not natively support the composition of jobs.

V.ADVANTAGES OF MAPREDUCE PROGRAMMING APPROCH:

The MapReduce programming 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.

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.

The implementation of MapReduce scales to large clusters of machines comprising thousands of machines. The implementation makes efficient use of these machine resources and therefore is suitable for use on many of the large computational problems encountered at Google.

VI.CONCLUSION:

Data mining from big data has been a new challenge in recent years. Traditional rough sets based methods for knowledge discovery fail to deal with the enlarging data in applications.

Google developed a software framework called MapReduce which is used to process big data and acquire the knowldege from it.
MapReduce also support large distributed data sets on clusters of computers which is effective to analyze large amounts of data. MapReduce is one of the most popular computing model for cloud computing platforms .
We can easily extend the MapReduce to be support for iterative programs in many applications such as data mining, web ranking and graph analysis .
By identifying MapReduce issues and challenges in Big Data, this paper also provides an
overview of the different Mapreduce implementation approches with their advantages and limitation.

REFERENCES

[1]. Junbo Zhang, Tianrui Li,Yi Pan,"Parallel rough set based knowledge acquisition using MapReduce from big data"**,**Proceeding in, BigMine'12 Proceedings of the 1st International Workshop on Big Data, Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Streams and Applications ,pages 20-27.

[2]. Junbo Zhang , Jian-Syuan Wong , Tianrui Li , Yi Pan ," A comparison of parallel large-scale knowledge acquisition using rough set theory on different MapReduce runtime systems " International Journal of Approximate Reasoning 55 (2014) 896–907

[3]. K. Grolinger, M. Hayes, W. Higashino, A. L'Heureux, D. S. Allison, M. A. M. Capretz, "Challenges for MapReduce in Big Data", to appear in the Proc. of the IEEE 10th 2014 World Congress on Services (SERVICES 2014), June 27-July 2, 1014, Alaska, USA.
[4]. Vasiliki Kalavri,Vladimir Vlassov, "mapreduce: Limitations, Optimizations and Open Issues" Proceeding in, TRUSTCOM '13 Proceedings of the 2013 12th IEEE nternational Conference on Trust, Security and Privacy in Computing and Communications ,Pages 1031-1038

[5]. J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[6] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1): 107–113, Jan. 2008.

[7] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox. Twister: a runtime for iterative mapreduce. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC'10, pages 810–818, New York, NY, USA, 2010. ACM.
[8] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, C. Kozyrakis, Evaluating mapreduce for multi-core and multiprocessor systems, in: Proceedings of
the 2007 IEEE 13th International Symposium on High Performance Computer Architecture, HPCA '07, IEEE Computer Society, Washington, DC, USA,
2007, pp. 13–24.

[9] X. Xu, J. J¨ ager, and H.-P. Kriegel. A fast parallel clustering algorithm for large spatial databases. Data Min. Knowl. Discov., 3(3): 263–290, Sept. 1999.

[10] J. Zhang, T. Li, D. Ruan, Z. Gao, and C. Zhao. A parallel method for computing rough set
approximations. Information Sciences, 194(0): 209–223, July 2012.

[11] W. Ziarko. Discovery through rough set theory. Commun. ACM, 42(11): 54–57, Nov. 1999.

[12] Hadoop: Open source implementation of MapReduce, < http : //hadoop.apache.org/mapreduce/ >.

ISSN: 2278 – 1323 All Rights Reserved © 2014 IJARCET