ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

HADOOP FILE SYSTEM AND FUNDAMENTAL CONCEPT OF MAPREDUCE INTERIOR AND CLOSURE ROUGH SET APPROXIMATIONS

A.Pradeepa[1], Dr. Antony Selvadoss Thanamani [2]

Research scholar, Computer science (Aided), NGM College, Coimbatore, India [1]

Associate professor & Head, Computer science, NGM College, Coimbatore, India [2]

Abstract: A massive data mining and knowledge discovery present a huge challenge with the volume of data growing at an unprecented approximation value. MapReduce has been implemented in manage many large-scale computation. The recently introduced MapReduce technique has received much consideration from both scientific community and industry for its applicability in big data analysis. The effective computation of approximation is essential improving the performance of data mining and other related task. For the purpose of data mining for massive data, parallel computing modes and algorithms are typical methods in these research fields. To mine knowledge from big data, we present consequently algorithm corresponding to the MapReduce based on roughest theory are put forward to deal with the massive data, in this paper comprehensive to evaluate the performances on the large data sets show that the proposed demonstrated can effectively process of big data.

Keywords: Data mining, MapReduce, HDFS (Hadoop Distributed File System), Rough sets

I. INTRODUCTION

Data processing and knowledge discovery for massive data has always been an active research area in data mining [1]. There are many application associated with massive data, such as association rule mining[2], sequential pattern mining[3], text mining[4] and temporal data mining[5], among the many algorithms based on rough set theory. Since the emergence of Hadoop implementation from various areas into the MapReduce, Leveraging this technology [6] forget at organization like Google, Yahoo (Hadoop) and social media it has now broken into the enterprise to solve the challenges that many organization are faced with, along with the coming cloud computing era, there are urgent requirements (e.g.) business intelligence, energy and utilities for data mining on the massive data by cloud computing platforms. Dean and Ghemawat from Google firstly presented a parallel programming model in MapReduce, which was a framework for processing huge data sets on certain kinds of distributable problems using a large number of computers(nodes), collectively referred to as a cluster[7,8]. MapReduce is a popular computing model for cloud computing platforms. There are many research papers related to MapReduce combined with the traditional methods [9, 10, 11]. In [12] Berlinska et al. analysed MapReduce computations as a divisible load scheduling problem and proposed a mathematical model of the application [13, 14, 15, and 16]. Yuan et al. presented a new distributed data integration system called VDB-MR, which was based on the MapReduce technology, to efficiently integrate data from heterogeneous data sources [14]. The significant of three trends: a shift to scalable, elastic computing infrastructure, an explosion in the complexity and verity of data available and the power and value that come from combining disparate data for comprehensive analysis make Hadoop a critical important platform for data-driven
Enterprise [16]. MapReduce framework offers clean abstraction between data analysis task and the underlying systems challenges involved in ensuring reliable large-scale computation. Finally, the MapReduce runtime system can be transparently explore the parallelism and schedule these components to distribute resource for execution. All problems formatted in this way can be parallelized automatically deploying the newest version of Hadoop (2.0, aka,'yarn') on Amazon EC2 [17].
The remaining of this paper is organized as follows and implementation of HDFS and MapReduce in section 2 includes the fundamental concept to MapReduce in

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

section 3. The methods for computing rough set approximation based on MapReduce are presented in section 4. The paper ends with conclusion in section 5.

II. AN IMPLEMENTATION OF HDFS AND MAPREDUCE

Hadoop Distributed File System is the primary storage system used by Hadoop applications. HDFS creates multiple replicas of data blocks [18] and distributes them on compute nodes, throughout a cluster to enable reliable extremely rapid computations. In this component of cost-effectiveness, scalability and streamlined architecture of Hadoop will make the technology more and more attractive. Hadoop [19] consist of two major components as follows: File storage and Distributed processing system. The first component of file storage is called "HDFS (Hadoop distributed file system)". It provides scalable, reliable, low cost storage. HDFS stores files across a collection of servers in a cluster. HDFS ensures data availability by continually monitoring the servers in a cluster and the blocks that they manage.

The second major components of Hadoop, is the parallel data processing system called "MapReduce". The MapReduce framework and the Hadoop distributed file system are running on the same set of nodes. The applications specify the input and output location and supply map and reduce function via implementations of appropriate interfaces and/or abstract classes. MapReduce allows the execution of java code and also uses software written in other languages. (e.g. C, C++, PHP, Python and Pearl). MapReduce and Hadoop were initially to solve an engineering problem core to the businesses, building an index of more than 1 billion web pages, but the technology is found to be proven effective in other unexpected areas like improving pagelayout, advertisement selection, spell-checking, maprendering and so on.

III. FUNDAMENTAL CONCEPT OF MAPREDUCE

MapReduce allows for distributed processing of the Map/Reduction functions. The MapReduce processing infrastructure includes an abstraction called an "input split" that permits each block to be broken into individual records. There are special processing built into reassemble records broken by block boundaries.

A) Input Reader

The *input* reader divides the input into appropriate size 'splits' and the framework assigns one split to each map function. The input reader reads data from stable storage and generates key/value pairs.

B) ***Map Function***

The *map* function process a key/value pair to generate another key/value pair. A number of such map functions running in parallel on the data that is partitioned across the cluster, produce a set of intermediate key/value pairs.

$$Map(k, v) \rightarrow < k^{'}, v' >^{*}$$

C) ***Compare Function***

The input for each reduces is pulled from the machine where the map ran and stored using the applications *comparison* function.

$$Compute(k', v') \rightarrow < k^{'}, v' >^{*}$$

D) Partition Function

The *partition* function is given the key and the number of reducers and returns the indexes of the desired reduce. It is important to pick a partition function that gives an approximately uniform distribution of data reducers assigned more than their share of data for load-balancing operation to finish.

E) Reduce Function

The *partition* function is given the key and the number of reducers and returns the indexes of the desired reduce. It is important to pick a partition function that gives an approximately uniform distribution of data reducers assigned more than their share of data for load-balancing operation to finish.

F) Reduce Function

The *reduce* function then merge all intermediate values that are associated with the same intermediate key.

$$Reduce(k', v') \rightarrow < k^{'}, v' >^{*}$$

G) Output Writer

The *reduce* function then merge all intermediate values that are associated with the same intermediate key.

$$Reduce(k', v') \rightarrow < k^{'}, v' >^{*}$$

MapReduce allows developers to write and deploy code that runs directly on each data-node server in the cluster. That code understands the format of the data stored in each block in the file and can implement simple algorithms and much more complex ones. It can be used to process vast amounts of data in-parallel on large clusters in a reliable and fault-tolerant fashion. Consequently, it renders the advantages of the Map/Reduce available to the users.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

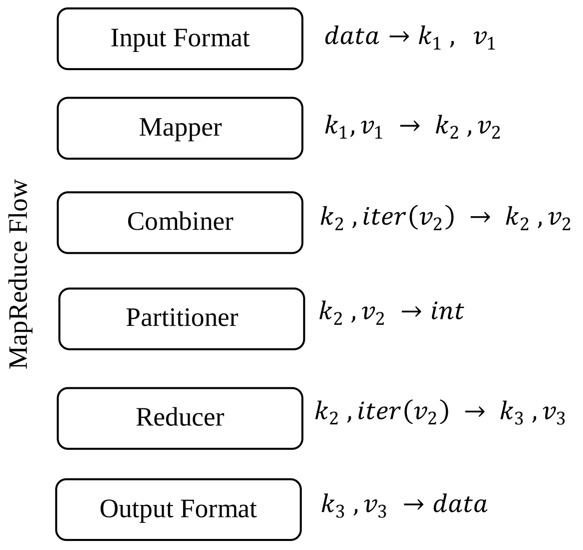
MapReduce Flow
Input Format
$data \rightarrow k_1, v_1$
Mapper
$k_1, v_1 \rightarrow k_2, v_2$
Combiner
$k_2, iter(v_2) \rightarrow k_2, v_2$
Partitioner
$k_2, v_2 \rightarrow int$
Reducer
$k_2, iter(v_2) \rightarrow k_3, v_3$
Output Format
$k_3, v_3 \rightarrow data$

Fig.1 MapReduce programming model

IV. COMPUTATION OF ROUGH SET THEORY APROXIMATION BASED ON MAPREDUCE

Rough set theory (RST) was proposed by Pawlak in the early 1980s. It is a powerful mathematical tool to describe the dependencies among attributes, evaluate the significance of attributes, and derive decision rules. Nowadays, many rough sets based-approaches have been successfully applied in machine learning and data mining. They have been found to be particularly useful for rule induction and feature subset selection. Rough set theory proposes a new mathematical approach to imperfect knowledge. In this approach, vagueness is expressed by a boundary region set. Rough set concept can be defined by means of topological operations, interior and closure, called approximations. This application of Rough set has resolved complex problems.

- Implementations of Map Reduce.
- Traditional methods combined with Map Reduce.

Rough sets can be also defined by using, instead of approximations, a rough membership Function. We briefly summarize the algorithm for computation of rough set approximation and analyze the MapReduce functions.

$$R_*(X) = \{x \in U : \mu_{X(x)}^{R} = 1\}$$
$$R^*(X) = \{x \in U : \mu_{X(x)}^{R} > 1\}$$
$$RN_R(X) = \{x \in U : \mu_{X(x)}^{R} > 1\}$$

The rough membership function expresses conditional probability that x belongs to X given R and can be interpreted as a degree that x belongs to X in view of information about x expressed by R. Rough sets can be also defined by using, instead of approximations, a rough membership function.

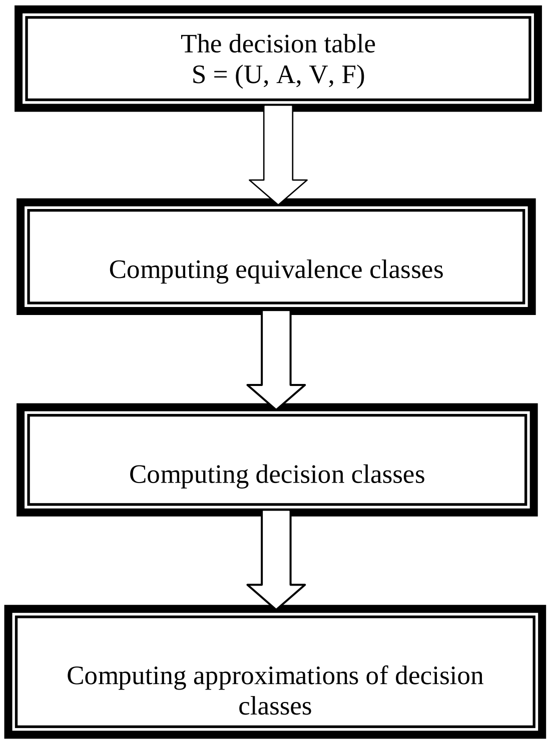
The decision table
S = (U, A, V, F)
Computing equivalence classes
Computing decision classes
Computing approximations of decision classes

Fig.2 Evaluate For Computing Rough Set Approximations

•In classical set theory, either an element belongs to a set or it does not.

•The corresponding membership function is the characteristic function for the set, i.e. the function takes values 1 and 0, respectively. In the case of rough sets, the notion of membership is different.

•The *rough membership function* quantifies the degree of relative overlap between the set X and the equivalence class $R(x)$ to which x belongs. It is defined as follows.

$$\mu_X^R : U \rightarrow \langle 0,1 \rangle$$
$$\mu_X^R(x) = \frac{|X \cap R(x)|}{|R(x)|}$$

A decision system expresses all the knowledge about the model.

•This table may be unnecessarily large in part because it is redundant in at least two ways.

•The same or indiscernible objects may be represented several times, or some of the attributes may be superfluous.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

Essentially, the MapReduce model allows users to write map/reduce components with rough set functional. The components are then composed as a data flow graph with fixed dependency relationship to explicitly specifies its parallelism.

V. CONCLUSION

Many rough sets based algorithm for data mining have been proposed in the past decades. However, the enlarging data in applications make algorithms based on rough sets a challenging task. Since the computation of rough set approximations is the necessary step, the development of its efficient algorithms becomes an important task. In this paper, we produce a MapReduce based on computing rough set approximations. This algorithms corresponding to the rough set method were successfully designed. It is indeed an inexpensive and simple, yet powerful solution for exploiting the potential of parallel processing. This may involve exploring the scheduling mechanisms of Hadoop or any other implementation of Map/Reduce. Future work will involve the parallel frequent pattern mining exploration of an alternative method that calculate the attribute space, so that information systems with a large number of attributes, such as those used in mathematical, may be analysed effectively.

REFERENCES

[1] J. Han, M. Kamber, Data Mining: Concepts and Techniques, second ed., Morgan Kaufman, San Francisco, 2006.
[2] P.Y. Hsu, Y.L. Chen, C.C. Ling, Algorithms for mining association rules in bag databases, Information Sciences 166 (2004) 31–47.
[3] Y.C. Hu, G.H. Tzeng, C.M. Chen, Deriving two-stage learning sequences from knowledge in fuzzy sequential pattern mining, Information Sciences 159 (2004) 69–86.
[4] K. Lagus, S. Kaski, T. Kohonen, Mining massive document collections by the websom method, Information Sciences 163 (2004) 135–156.
[5] Y. Li, S. Zhu, X.S. Wang, S. Jajodia, Looking into the seeds of time: discovering temporal patterns in large transaction sets, Information Sciences 176 (2006)1003–1031.
[6] Mine Olson, "Hadoop scalable, flexible data storage and analysis", Vol 1 no 3.
[7] J. Dean, S. Ghemawat, and MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (2008) 107–113.
[8] J. Ekanayake, S. Pallickara, G. Fox, MapReduce for data intensive scientific analyses, in: Proceedings of Fourth IEEE International Conference on eScience,
Indianapolis, Indiana, USA, 2008, pp. 277–284.
[9] J. Berlin´ ska, M. Drozdowski, Scheduling divisible MapReduce computations, Journal of Parallel and Distributed Computing 71 (2011) 450–459.
[10] J. Ekanayake, S. Pallickara, G. Fox, MapReduce for data intensive scientific analyses, in: Proceedings of Fourth IEEE International Conference on eScience,
Indianapolis, Indiana, USA, 2008, pp. 277–284.
[11] Z. Lv, Y. Hu, H. Zhong, J. Wu, B. Li, H. Zhao, Parallel k-means clustering of remote sensing images based on MapReduce, in: F. Wang, Z. Gong, X. Luo, J.Lei (Eds.), Web Information Systems and Mining, Lecture Notes in Computer Science, vol. 6318, Springer, Berlin/Heidelberg, 2010, pp. 162–170.
[12] J. Wan, W. Yu, X. Xu, Design and implement of distributed document clustering based on MapReduce, in: Proceedings of the Second Symposium International Computer Science and Computational Technology (ISCSCT), Huangshan, PR China, 2009, pp. 278–280.
[13] Y. Yang, Z. Chen, Z. Liang, G. Wang, Attribute reduction for massive data based on rough set theory and MapReduce, in: J. Yu, S. Greco, P. Lingras, G.Wang, A. Skowron (Eds.), Rough Set and Knowledge Technology, Lecture Notes in Computer Science, vol. 6401, Springer, Berlin/Heidelberg, 2010, pp. 672–678.
[14] Y. Yuan, Y. Wu, X. Feng, J. Li, G. Yang, W. Zheng, VDB-MR: MapReduce-based distributed data integration using virtual database, Future Generation
Computer Systems 26 (2010) 1418–1425.
[15] W. Zhao, H. Ma, Q. He, Parallel k-means clustering based on MapReduce, Lecture Notes in Computer Science 5931 (2009) 674–679.
[16] Y. Yuan, Y. Wu, X. Feng, J. Li, G. Yang, W. Zheng, VDB-MR: MapReduce-based distributed data integration using virtual database, Future Generation
Computer Systems 26 (2010) 1418–1425.
[17] Hadoop wiki - powered by. http://wiki.apache.org/hadoop/PoweredBy.
[18] Hadoop: open source implementation of MapReduce, <http://hadoop.apache.org/mapreduce/>.
[19] Pawlak, Z. (1991). Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, ISBN 0-79231472, Norwell-USA.
[20] Wei, J. M. (2003). Rough Set based Approach to Selection of Node, International Journal of Computational Cognition, Vol. 1, No. 2, pp. 25–40, ISSN 1542-8060.