International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS)
ISSN (Online) 2349-6967 *Volume 2 , Issue 1(Jan-Feb 2015), PP1-5*

Dimension reduction using RST and Map Reduce

Komal Awalekar[1], Reshma Aga[2]
[1 2] *(Rajarambapu Institute of Technology, Sangli, Maharashtra, india)*

Abstract:- Nowadays, handling with massive data at big rate or large-scale data mining has become new task. Rough set theory for knowledge learning or developing is applied in data mining. In this project we are going to present a serial as well as a parallel method [1]. These serial and parallel methods are used for computing rough set approximations. We will implement the parallel algorithm on Hadoop Map Reduce platform [3, 5]. We will evaluate the performance of three parallel algorithms with respect to its speedup, scale up and size up[7]. The Hadoop Map Reduce platform is a programming model and also it is a software framework for developing applications. The Hadoop Map Reduce platform works on massive data and also it the data gets processed rapidly in parallel on large clusters of compute notes. In this project, we will evaluate only the efficiency of the algorithms and not its accuracy. By using parallel method on Hadoop platform, due to transparent spilling, the data size limit will be get removed. However, algorithms based on rough set theory are quite a challenging task for the applications of enlarged data.

Keywords: Rough Set, Data Mining, Sequential Algorithm, Parallel Algorithm, HDFS (Hadoop Distributed File System)

I. INTRODUCTION

Massive data mining and knowledge discovery present a tremendous challenge with the data volume growing at an unprecedented rate. Rough set theory has been successfully applied in data mining. The lower and upper approximations are basic concepts in rough set theory. The effective computation of approximations is vital for improving the performance of data mining or other related tasks. The recently introduced Map Reduce technique has gained a lot of attention from the scientific community for its applicability in massive data analysis. This paper proposes a parallel method for computing rough set approximations. On sequentially, algorithms corresponding to the parallel method based on the Map Reduce technique are put forward to deal with the massive data. An extensive experimental evaluation on different large data sets shows that the proposed parallel method is effective for data mining.

II. LITERATURE SURVEY

In this section we will introduce our starting point, i.e. Rough Set Theory and Hadoop concept.

2.1 Rough Set Theory:-

Rough Set Theory[5] is one of the mathematical method which was developed by Pawlak in 1982. It deals with the classificatory analysis of data tables. The main goal of the rough set analysis is to synthesize approximation of concepts from acquired data. The fundamental concept behind Rough Set Theory is the approximation of lower and upper spaces of a set, the approximation of spaces being the formal classification of knowledge regarding the interest domain.

Rough Set Theory represents different mathematical approach to vagueness, uncertainty. Rough Set Theory is applied in data mining for knowledge acquisition. Rough Set Theory is proposed to process with big data. When the available data is insufficient to determine the value of a given set then at that time we can use Rough Set Theory for the representation of the concerned set. The basic concepts in Rough Set Theory are upper approximation and lower approximation.

The subset generated by lower approximations is characterized by objects that will definitely form part of an interest subset, whereas the upper approximation is characterized by objects that will possibly form part of an interest subset. Every subset defined through upper and lower approximation is known as Rough Set.

Over the years Rough Set Theory has become a valuable tool in the resolution of various problems, such as: representation of uncertain or imprecise knowledge; knowledge analysis; evaluation of quality and availability of information with respect to consistency and presence not of date patterns; identification and evaluation of date dependency; reasoning based an uncertain and reduct of information data.
The Rough Set Theory has been applied in several fields some of them are data mining, image processing, medical informatics, pattern recognition, knowledge discovery and expert systems.

Basic concepts of the rough sets theory Information system:-
Formally, an *information system[5]*, IS can be seen as a system IS=(U,A) where U is the universe (a finite set of objects, U={x1,x2,…..,xm}) and A is the set of attributes. Each attribute a∈A defines an information function *fa*: U->V*a* where V*a* is the set of values of a, called the domain of attribute a.

Indiscernibility relation:-

For every set of attributes $B\subset A$, an indiscernibility *relation* Ind(B) is defined in the following way: two objects, *xi* and *xj* are indiscernible by the set of attributes *B* in *A*, if *b*(xi)=*b*(xj) for every $b\subset B$. The equivalence class of Ind(*B)* is called *elementary set* in .

Lower and upper approximations:-

The rough sets approach to data analysis hinges on two basic concepts, namely the *lower* and the *upper approximations* of a set .

Let *X* denote the subset of elements of the universe U(X$\subset$*U).* The lower approximation of *X* in B(B⊆A),denoted as $\underline{BX}$, is defined as the union of all these elementary sets which are contained in *X.*
The above statement is to be read as: the lower approximation of the set X is a set of objects xi, which belong to the elementary sets contained in X (in the space B)
The upper approximation on the set X, denoted as BX, is the union of these elementary sets, which have a non-empty intersection with X:

BX={x*i*∈U|[x*i*]Ind(B) ∩ $X\neq 0$}.

For any object xi of the lower approximation of X (i.e., xi∈ $\underline{BX}$), it is certain that it belongs to X. For any object xi of the upper approximation of X (i.e., xi∈BX), we can only say that xi may belong to X. The difference
BNX=BX-$\underline{BX}$
Is a boundary of X in U.
Accuracy of approximation
An accuracy measure of the set *X* in *B*:*A* is defined as:
μ*B*(*X*) =card($\underline{BX}$)/card(BX).
The cardinality of a set is the number of objects contained in the lower (upper) approximation of the set *X.* As one can notice, 0 ≤ μ*B*(*X*) ≤1. If *X* is definable in *U* then μ*B*(*X*)=1, if *X* is indefinable in *U* then μ*B*(*X*) <-1.

Core and reduct of attributes:-

If the set of attributes is dependent, one can be interested in finding all possible minimal subsets of attributes, which lead to the same number of elementary sets as the whole set of attributes (*reducts*)[5] and in finding the set of all indispensable attributes (*core*)[5].
The concepts of core and reduct are two fundamental concepts of the rough sets theory. The reduct is the essential part of an IS, which can discern all objects discernible by the original IS. The core is the common part of all reducts. To compute reducts and core, the *discernibility matrix* is used. The discernibility matrix has the dimension *n* x *n*, where *n* denotes the number of elementary sets and its elements are defined as the set of all attributes which discern elementary sets [x]*i* and [x]*j*.

2.2 Hadoop:

Hadoop includes fault-tolerant storage system called Hadoop Distributed File System. HDFS is used to store large amount of information, scale up incrementally without losing data..

Hadoop distributes the data by pushing the work involved in an analysis out to many different servers. Each of the servers runs the analysis on its own block from the file. Hadoop is an open-source software framework written in Java and based on Google's MapReduce[1] and distributed file system work. It is built to support distributed applications by analyzing very large bodies of data using clusters of servers, transforming it into a form that is more usable by those applications. Hadoop is designed to be deployed on commonly available, general-purpose infrastructure.

2.3 HDFS:-

The Hadoop Distributed File System[2] is a scalable and reliable distributed storage system that aggregates the storage of every node in a Hadoop cluster into a single global file system. HDFS stores individual files in large blocks, allowing it to efficiently store very large or numerous files across multiple machines and access individual chunks of data in parallel, without needing to read the entire file into a single computer's memory. Reliability is achieved by replicating the data across multiple hosts, with each block of data being stored, by default, on three separate computers. If an individual node fails, the data remains available and an additional copy of any blocks it holds may be made on new machines to protect against future failures.

Hadoop Distributed File System (HDFS)achieves fault tolerance and high performance by breaking data into blocks and spreading them across large numbers of worker nodes. Hadoop MapReduce Engine accepts jobs from applications and divides those jobs into tasks that it assigns to various worker nodes.

Hadoop coordinates work among *clusters* of machines which are created by hadoop itself. Clusters can be built with inexpensive computers. If one of the machine fails, without losing data or interrupting work hadoop continues the work by shifting work to remaining machine in the cluster. By breaking incoming files into pieces called "blocks", and storing each of the blocks redundantly across the pool of servers, HDFS manages storage in cluster. HDFS stores three complete copies of each file.

We can see that even though any two servers fails then entire file will still be available. HDFS creates a new copy of missing data from the replicas which it manages when it notices a block or a node is lost. As the cluster stores several copies of every block, clients can read them at the same time without creating bottlenecks.

There are various redundancy techniques such as RAID machine. HDFS has two key advantages over RAID that,

i) It requires no special hardware because it can be built from commodity servers.

ii)It can survive more kinds of failure.

2.4 MapReduce:-

Hadoop operates on massive datasets by horizontally scaling the processing across very large numbers of servers through an approach called MapReduce, MapReduce is the programming model that allows Hadoop to efficiently process large amounts of data. MapReduce breaks large data processing problems into multiple steps, namely a set of Maps and Reduces, that can each be worked on at the same time (in parallel) on multiple computers.

MapReduce is designed to work with of HDFS. Apache Hadoop automatically optimizes the execution of MapReduce programs so that a given Map or Reduce step is run on the HDFS node that contains locally the blocks of data required to complete the step. Explaining the MapReduce algorithm in a few words can be difficult, but we provide an example in Appendix A for the curious or technically inclined. For the rest, suffice it to say that MapReduce has proven itself in its ability to allow data processing problems that once required many hours to complete on very expensive computers to be written as programs that run in minutes on a handful of rather inexpensive machines. And, while MapReduce can require a shift in thinking on the part of developers, many problems not traditionally solved using the method are easily expressed as MapReduce programs.

2.5 Hadoop for Big Data Analysis:-

Hadoop aims at problems that require examination of all the available data. Hadoop uses a technique called MapReduce to carry out the exhaustive analysis quickly such as Text Analysis and Image Processing which requires that every single record to be read.

III . RELATED WORK

3.1 Sequential Method:-

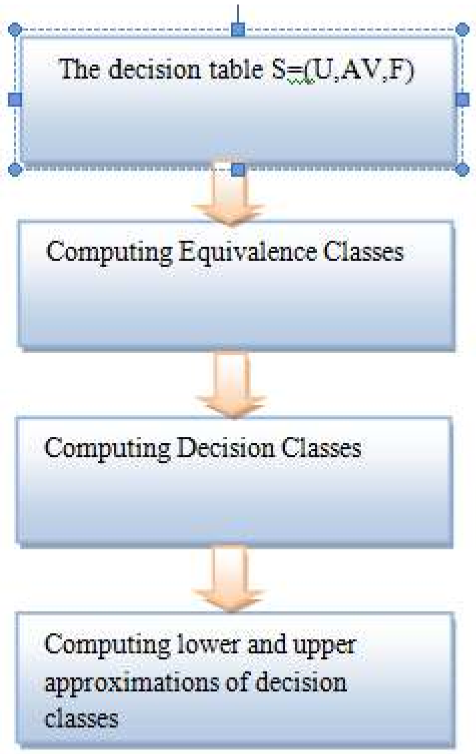

The decision table S=(U,AV,F)
Computing Equivalence Classes
Computing Decision Classes
Computing lower and upper approximations of decision classes

Fig 1 . Sequential Method

.

IV. PROPOSED SYSTEM

Parallel Method:-
In parallel method we overcome the time complexity problem of sequential method.

In this we are going to use hadoop mapreduce which is a programming model and software framework for developing applications that rapidly process massive data in parallel on large clusters of computer nodes, each of which has four 2.13 GHz cores and 12 GB of memory. Hadoop version 0.20.2 and Java 1.6.0.01 are used as the MapReduce system for all experiments.

In parallel method we calculate the efficiency of algorithm not the accuracy because both the method i.e parallel and sequential method produces same result.

We may design the corresponding parallel algorithms for computing equivalence classes, decision classes and associations between equivalence classes and decision classes. We present a parallel method for computing rough set approximations. Our parallel algorithms have all been implemented on Hadoop MapReduce platform.

The proposed parallel method scales up very well and has excellent speedup and sizeup behavior. Specially, as the size of the data set increases, the speedup performs better.

The sequential method performs in-memory processing ,it cannot deal with large data set the proposed parallel method removes this data size limit.

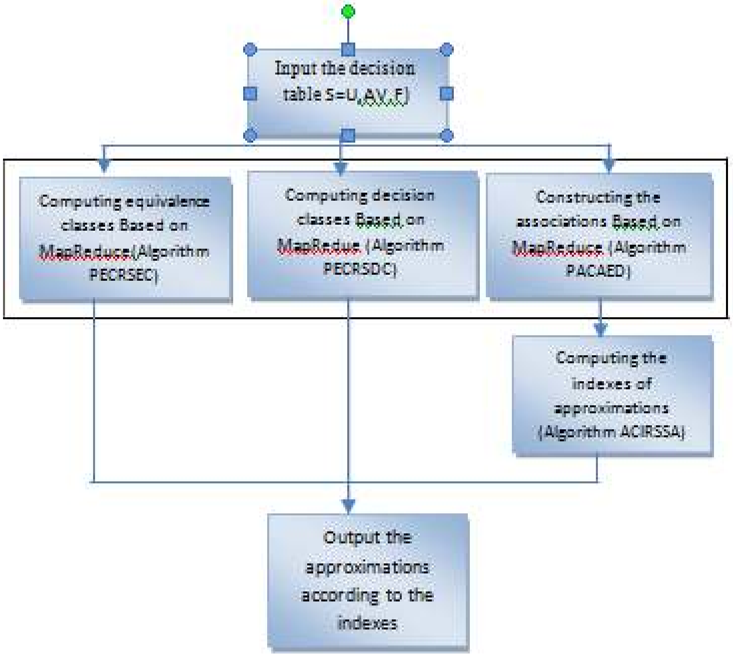

Input the decision table S=U,AV,F)
Computing equivalence classes Based on MapReduce(Algorithm PECRSEC)
Computing decision classes Based on MapRedue (Algorithm PECRSDC)
Constructing the associations Based on MapReduce (Algorithm PACAED)
Computing the indexes of approximations (Algorithm ACIRSSA)
Output the approximations according to the indexes

Fig 2. Parallel Method

V. OBJECTIVES

5.1 Scale up:-

Scaleup[4] is ability of larger system to perform larger job in the same execution time.

Scaleup(D, p)=$T_{Di/}T_{Dp}$

Where D is the data set, TDi is the execution time for D on i^{th} node, TDp is the execution time for p *D on p nodes.

5.2 Size up:-

Sizeup[4] analysis holds the number of computers in the system constant, and grows the size of the data sets by the factor p.

Sizeup(D, p)=$T_{Sp/}T_{S1}$

where TSp is the execution time for p*D, TS1 is the execution time for D.

5.3 Speedup:-
To measure the speedup[4], we keep the data set constant and increase the number of nodes in the system.
Speedup(p)=T_{1}/T_{p}

VI. RESULTS

We have tested existence system on .NET framework.
This section contains the results got after the implementation of sequential method for rough set approximation.

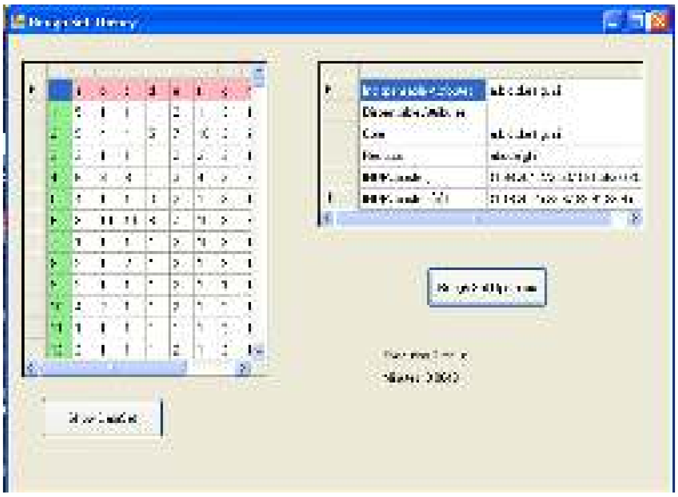
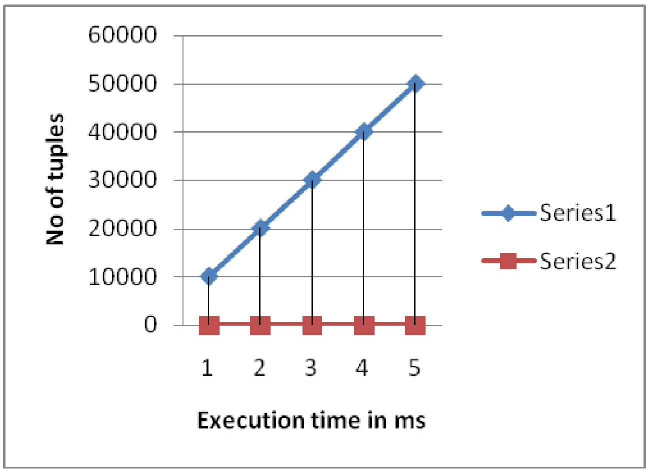


Fig 3. Sequential Algorithm

TABLE 1:

No.Of Tuples	*Execution Time(Milisecond)*
10000	*0.0843*
20000	*0.1886*
30000	*0.2529*
40000	*0.3372*
50000	*0.4215*

Graph No.1 : Volume Vs Velocity



VII. CONCLUSION

In this paper we explored the concept of Rough Set approximations using Hadoop Map reduce. The sequential algorithm tested on different size of datasets. Also we have identified equivalence class, rough set reduct and core using sequential algorithm which is applied on different size dataset.

REFERENCES

[1]"*Scheduling divisible MapReduce computations*", Journal of Parallel and Distributed Computing 71 (2011) 450–459.

[2]Hadoop: Open sourse implementation of MapReduce- ''http://hadoop.apache.org/mapreduce/".

[3] ''*A Parallel method for computing Rough Set Approximation''*- from Information Sciences journal-194 (2012) 209–223 By Junbo Zhang, Tianrui Li, Da Raun, Zizhe Gao, Chengbin Zhao.

[4] Zhang, J.. "A parallel method for rough set approximations", from Information Sciences journal-194 (2012) 209–223 By Junbo Zhang.

[5]Z. Pawlak, Rough Sets. Theoretical Aspects of Reasoning about Data, Kluwer Academic Publisher, Dordrecht, Nether-lands, 1991.

[6]W.P. Ziarko Ed. , Rough Sets, Fuzzy Sets and Knowledge Discovery, Springer, New York