Journal of Theoretical and Applied Information Technology
20th February 2016. Vol.84. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

INPUT SPLIT FREQUENT PATTERN TREE USING MAPREDUCE PARADIGM IN HADOOP

[1]GREESHMA L, [2]PRADEEPINI G

[1]CSE Department of K L University, Vaddeswaram, Andhra Pradesh, India

[2] CSE Department of K L University, Vaddeswaram, Andhra Pradesh, India

E-mail: [1]greeshma243@gmail.com, [2]pradeepini.gera@gmail.com

ABSTRACT

Big data has been attracted in information industry and in the society in the recent years, due to the wide availability of huge amount of data in the Internet and the complexity of data is growing every day. Hence distributed data mining algorithms has decided to exploit big data adaptable to current technology. Since there exist some limitations in traditional algorithm for dealing with the massive volume of data set which degrades the performance. So, thereby we require fast and efficient scalable frequent item sets for storing and processing large data sets. Existing algorithm like apriori algorithm performs a multiple scans from external storage, which leads to heavy burden for I/O devices. In this paper, we proposed Association Rule Mining based on Hadoop Distributed File System for storing huge amount of data and implemented using MapReduce object oriented programming paradigm for processing of a data.

Keywords: *Big Data, Hadoop Distributed File System, MapReduce, Association Rule Mining, Distributed Frequent Item set Mining.*

1. INTRODUCTION

Big data is handling complex data sets where the challenges are not only to save the data but also to read, browse, cache and also in order to do meaningful analytics on top of it [23]. Big data plays a vital role in both private and public organizations for storing massive amount of domain-specific which also provides the essential information about outlier analysis, predictive analysis and also it helps to predict the future trends in stock market and decision for making stock investments where huge amount of data is unsupervised and unlabeled. Industries are embracing Big data in Retail, Financial services, Manufacturing, Government, Advertising public relations, Media and Telecommunications, healthcare and life sciences [10]. The challenge in big data analytics includes retrieving patterns from massive volume of data, fast processing and simplifying discriminant description, which are in the form of a rule.

In traditional approach, an enterprise will get a powerful computer and it will speed in whatever data available in this computer. The computer will do a good job unless there exist a certain point where the computer will not be able to do processing any more. Hence it is not scalable. Since big data is growing rapidly. Thus a traditional enterprise approach has its own limitations. The limitation of an existing solution is data warehouse is build on top of RDBMS. Data warehouse has fixed schema of RDBMS, the cost of databases like Oracle, MySQL has a commercial aspect and if the file size is of 1Gigabytes the problem of saving huge file and accessing becomes difficult [6] [32].

Data warehouse performs analytics only on small portion of data but not on huge data sets. As compared to traditional solutions, hadoop turns out to be a good solution to the existing data analytics problem because it does storage and processing at same location. Hadoop is framework of tools that is an open source. Big Data analytics and hadoop open source projects are emerged as preferred solutions to address business and the current technology trends. The keyword behind hadoop is big data. The big data creates challenging points that hadoop address and the challenges are at 3 levels velocity, volume and variety. Lot of data coming at very high speed, big volume of data is gathered and data is all of heterogeneity. Hadoop is a combination of Distributed File System and Analytics algorithm. Distributed File System saves the big data files and where as MapReduce algorithm helps us to do analytics in quick time on commodity machines as a set of clusters. The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

hadoop takes different approach than the enterprise approach [4]. It breaks the computational data into smaller pieces. Thus why it is able to deal with big data and these computations will be computed in equal amount of time and the results are combined together and sent back to application. Therefore, we can say that hadoop has mainly two components. First one is MapReduce and the second component is file system that is termed as called Hadoop Distributed File System (HDFS). Hadoop is a framework of tools for managing and handling distributed applications. Hadoop is a specially designed File System for storing huge amount of datasets with clusters of commodity hardware with streaming access patterns which implies write once, read any number of times without changing the content of file once stored in HDFS. The applications are executed using object oriented programming paradigm called Map Reduce [2] [4][5][6].

Map Reduce uses the parallel processing of large data sets. The main aim is to build distributed association rule mining for huge datasets but not for a single portion of data. But in traditional algorithm like Apriori [1] which suffers mainly from two problems. One of the problem is main memory has to handle multiple candidate item sets for massive volume of data which will be a heavy weight component for processor. Thereby degrades the performance of an algorithm. Second problem, requires multiple scans from huge amount of data set [17]. Hence number of iterations will be more and time complexity of an algorithm increases. So, these reasons made researchers to be more concerned with the parallel processing of massive volume of a data set and implemented using Map Reduce technique through a java based framework. The Association Rule Mining using Map Reduce paradigm and storing the data in HDFS. So this distributed framework helps us to identity frequent item sets from large data set. Therefore the framework provides the efficiency and scalability even if the size of data increases in database as cloud resources are sourced from different service providers. Related work is discussed in Section 2. Section 3 outlines necessary prerequisites for storing and processing using hadoop distributed file system architecture, MapReduce and proposed methods. Section 4 describes about experimental results. Conclusion is given in Section 5.

2. RELATED WORK

Yahoo [25], Google [14], Facebook [7] and Oracle [34] are introducing new tools for fast storage and parallel processing of massive volume of data. Apache hadoop has proposed MapReduce 2.0 (Y ARN)[18]. In Cloud computing, big data provides data as service in the context of cost, efficiency and scalability. In Cloud computing environment big data analysis plays a vital role for storing and processing the data [40]. Since the data stored in hadoop is of unstructured format [33]. There are various issues for managing the data in cloud computing environment. HDFS and MapReduce provide efficient storage and processing of big data. Teragen and tera sort provides their performances in terms of storage and parallel of data. In recent years Oracle has published a white paper to provide big data solution that incorporates hadoop, Oracle datawarehouse and Oracle NoSQL [12]. In order to run multiple applications in heterogeneous environment by implementing MapReduce programming model requires certain scheduling task such as LATE and SAMR scheduler [9][27]. Nykiel and Potamisas proposed an MRShare approach to share information across job tracker in MapReduce [23]. Valvag and Oivos presented a system that integrates storage distributed file system along with its runtime environment [11][35]. However, none of the above-proposed systems failed to provide parallel data processing along with distributed computing solution. This mechanism is termed as static routing. Pasquier first proposed a classified searching approach with the help of Apriori properties over the subset of itemset termed as CLOSE [29]. Pei proposed CLOSET using the denser unit of data known as Frequent Pattern- Tree [30]. Grahne and Zhu proposed subsequent version of CLOSET+ [16]. J.Han proposed a Frequent Pattern growth algorithm for mining frequent patterns without generating a candidate keys [22]. J.Han and Holt proposed scalable and parallel mining of association rules on clusters of workstations [19][21]. Yu and Zhou proposed parallel transactional identifiers based on FP mining algorithm on PC cluster and grid computing system [37]. The above-mentioned algorithms work efficiently only on small or medium scale data sets but not for large scale data sets. Our method provides a solution to decrease the conflicts in MapReduce, time and cost and also provides fast processing and storing data using hadoop architecture and MapReduce techniques even for large volume of data sets.

Journal of Theoretical and Applied Information Technology
20th February 2016. Vol.84. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. PREREQUISITES AND PROPOSED METHOD

Figures In this section, we discuss the concept of Big Data [5], Hadoop Ecosystem [3], architecture of Hadoop Distributed File System [2] and Map Reduce programming paradigm [22], which are required for proposing an algorithm.

3.1 Big Data and Hadoop Ecosystem

Sections Big Data is about terabytes or petabytes of file. Day-by-day the size of a data is increasing from megabytes to petabytes and also increasing the data heterogeneity and the complexity [6]. IBM has defined the Big Data in three parameters.

(i) Volume: The scalability of data size.

(ii) Variety: Different forms of data such as video, audio, posting the information, images, structured, unstructured or unsupervised data etc.,

(iii) Velocity: Analysis of streaming data in terms of speed.

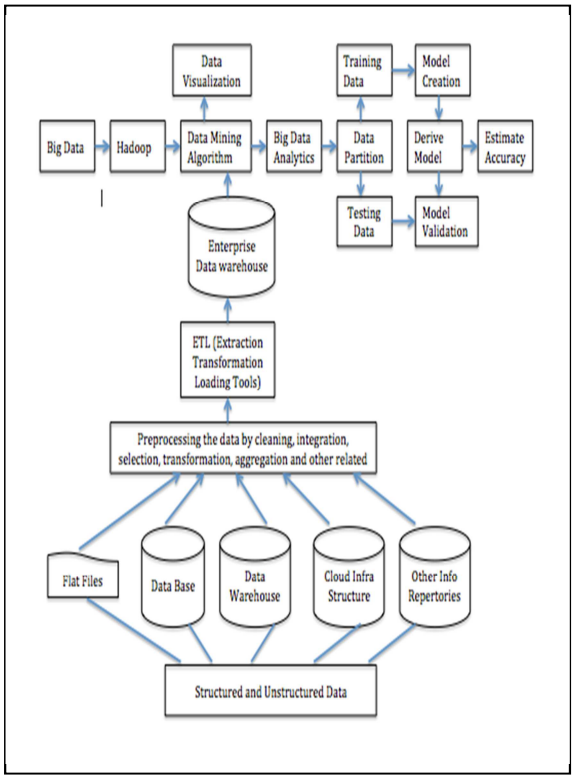


Figure 1: A Holistic View of a Big Data System

Flat Files, Database, Data warehouse, Cloud Infrastructure, other information repositories are set of databases Thee information is collected from structured or an unstructured data. Collected data may be represented as an inconsistent format. So there is need to convert an inconsistent data into a consistent data by using certain preprocessing techniques like data cleaning, data integration, data transformation, data reduction and concept hierarchy generation. Data cleaning as a process one of a data migration tool is ETL (Extraction, Transformation, Loading). This tool allows simple transformation through Graphical User Interface (GUI).

Enterprise Data warehouse gathers all information by spanning the entire organization [32]. It collects the data from multiple data sources especially from one or more operational systems or from external providers. The data is summarized and the size ranges from gigabytes to terabytes or beyond. It is implemented using parallel architecture or on mainframes. It is essential to mine the data using various data mining algorithms like characterization, classification, association and correlation analysis, clustering, outlier analysis, prediction and evolution analysis. One of data mining algorithm used is association mining. By using big data analytics the data is partitioned into training and testing data set. Holdout, random subsampling, cross validation and bootstrap are the techniques that are used to estimate accuracy.

Hadoop is scalable fault tolerant distributed system for data storage and processing. Hadoop works on distributed model, which are low cost computers [2]. Hive is similar to data warehouse but the difference between Hive and data warehouse is Hive uses entire population of data where as data warehouse uses only a single portion of data. Pig Latin Data Analytics is an abstraction on top of an underlined Map Reduce implementation. Mahout is an artificial intelligence [3]. For example websites like amazon uses some kind of search runs on top of it. Influences the end customer to buy that product which are not done in any previous website. These three components are underlined above Hadoop Distributed File System and Map Reduce. Apache Oozie is a workflow helps to start and stop the jobs and run jobs at specified intervals of time [2]. It also helps to schedule Map Reduce, Hive, Pig Latin, Mahout jobs are specified intervals of time [3]. The frameworks include flume and sqoop. Sqoop pulls the data from Online Transactional Processing

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(OLTP) or RDBMS and puts it into Hadoop implementations. The Flame is more about unstructured or semi-structured data, which helps to pull as well as push data from system to put into Hadoop Distributed File System environment. Hadoop consist of mainly two components HDFS and Map Reduce.

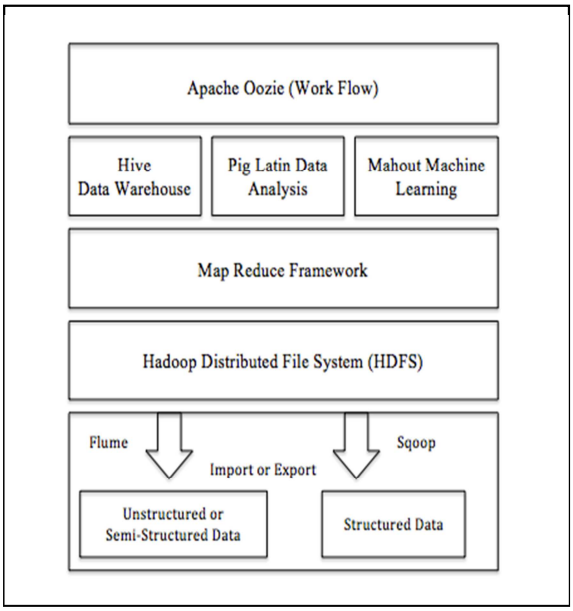


Figure 2: Hadoop Ecosystem

3.2 HDFS Architecture and Map Reduce

Hadoop Distributed File System (HDFS) part is typical cluster with collection of machines and their will be one heavy due server called as Admin Node and their will be collection of commodity machine. Commodity machine is like Personal Computer, Laptops of 8GB RAM, 1tetrabytes of hard disk and Intel i7 processing power. A typical hadoop implementation could have one Admin Node and 10 to 15 commodity machines are spread across the clusters. HDFS contains Master and Slave. Master (admin) Node runs on two daemons. One is Job Tracker and another is Name Node. Name Node directly correlates to HDFS part of it.

The Name Node and Admin Node manages everything with respect to storing, managing and accessing huge data files. In contract to that Job Tracker is more about Map Reduce Algorithm implementation [8] [9]. They both run at admin node levels and they both are daemons. The role of Job Tracker component is to break higher and bigger task into smaller pieces and to send each small computation to task tracker and send result back to Job Tracker as it combines the results and sends final result to application and Node Name running on master is responsible for storing metadata information about name and replications. A typical admin node is a heavy-duty machine with 64GB RAM kind of machine. Slave consist of set of the commodity machines are those machines which save data and trying to save on Distributed System which always have two daemons running on it. One is data node and another daemon could be task tracker. Data Node is going to take care about the distributed aspect of an implementation and Task Tracker is going to take care about Map Reduce Algorithm implementation part of it. Data Node is interacting with Name Node and where as Task Tracker is going on interacting with Job Tracker. Data is replicated on multiple nodes. Hadoop solves the limitations of an existing system of RDBMS/ Data warehouse. HDFS helps to solve entire file a gigabytes of size [2] [6]. Unlike in data warehouse, no need to cut the portion of it and do the analytics. So the Map Reduce framework does the analytics but in hadoop takes very less time [8]. If we increase the size of the commodity machine the amount of time it takes for processing will be less than an hour which scales out an efficient architecture.

For example if client sends file size of 200MB request to node name then the file is divided into four block which are termed as input splits because the default block size of hadoop is 64MB. So, their exist four input splits for a given file size. First three blocks consist of 64MB size each and the last block consist of 8MB and the remaining 56MB free space can be given another file size. But the client does not now what are free spaces available in the data node. Suppose if a system is crashed then one of the input split file will be lost. To overcome this HDFS maintains three replications (or a back up files) by default [2]. It also gives the acknowledgement to the client that the first input split is stored in 1, 2 and 4 commodities. The data node gives block report and heart beat to Name Node for every small period of time such that the blocks are live or dead. If the Meta data is lost then there is no useful with hadoop and the entire cluster is inaccessible. It is always recommended to maintain Node Name with a high reliable hardware in order to overcome the data loss. So the Node Name is called as single point of failure. Assume that for processing data of 200MB we require program size of 10KB. Client

Journal of Theoretical and Applied Information Technology
20th February 2016. Vol.84. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

sends the 10KB of data in HDFS environment there comes the existence of Job Tracker. Job Tracker sends Meta data operations to get the block information that is available in Name Node.

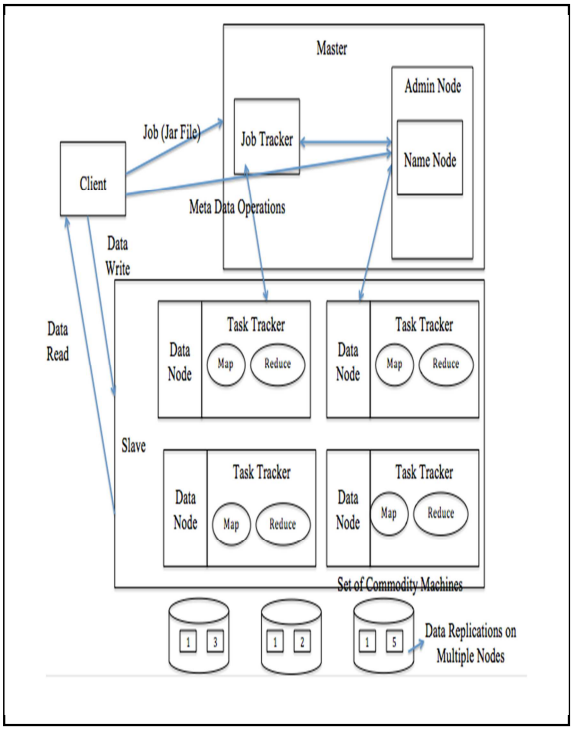


Figure 3: Hadoop Distributed File System Architecture

Job Tracker finds the nearest Task Tracker so that the information can be sent in less amount of time. The processing of program in Task Tracker is defined as a Map. If Map working in a system is crashed then the data node is lost [10]. The Job Tracker will reallocate task to another Task Tracker based replicated data available in a block. For every 3seconds of time the Task Tracker will give the heart beat to Job Tracker to intimate that they are alive. Job Tracker waits for 10 heart beats i.e., 30 seconds of time. Thereby concluding that either Task tracker is working slowly or it is dead. Along with heart beat Task Tracker sends the available slots to Job Tracker. It is recommended to maintain a high reliable hardware for Job Tracker. It is also defined as single point of failure. The Task Tracker generates output file. The number of output file is based on number of reducer. Finally the reducer combines the entire output file and it is sent back to an application.

The object oriented programming paradigm used in hadoop is Map Reduce. This Model consist of two primitives (i) Map and (ii) Reduce [8] [10]. The Map Reduce is based on (key, value) pair where key should not contain any duplicates but there is no restriction for value. For example input file of size 200MB is divided into four input splits. Record Reader reads only one record at a time. Record Reader is a predefined interface in hadoop. It records (byte offset, records the entire transaction) which is in the form of (key, value) pair. They are represented in four different formats that are predefined classes in Hadoop. Each Mapper contains number of lines in which input split appears to be parallel processing.

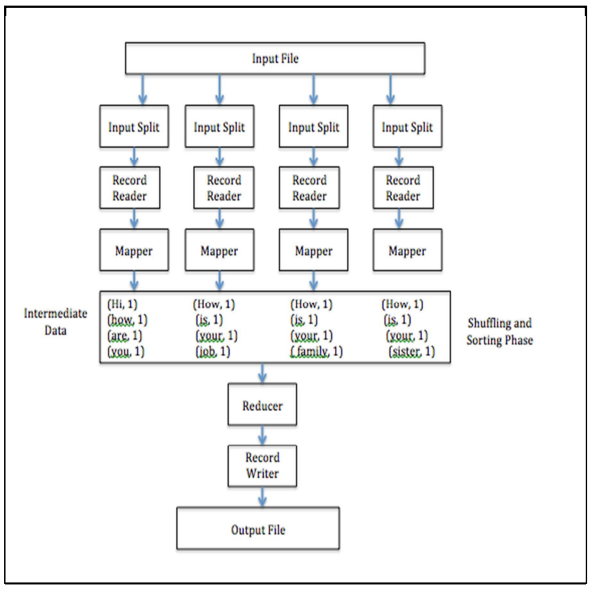


Figure 4: Map Reduce Flowchart

The default one is Text Input Format, Key Value Text Input Format, Sequence File Input Format and Sequence File As Text Input Format. The number of mapper depends on the input splits. The Mapper generates the intermediate data which is also in the form of (key, value) pair contains duplicates keys. In order to prune redundant keys we require two phases shuffling and sorting phase. In shuffling phase it combines all values, which are identical to key. Due to Writable Comparable Interface the sorting is done automatically. Therefore the Record Writer writes only 1 (key, value) pair at a time. Finally the output file contains part-00000.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Our Unique Approach using Map Reduce Implementation:

```
Map (String Item, String Value)
// Item: Item Name in a Stream data
// Value: Support count value of an item satisfying threshold value.
While (has more Data Stream)
  Value= Get Cached Data Stream ()
For each word in value
  Obtain Intermediate Data (w,"1")

Reduce (String key, Iterator values)
//Key: a word
// Values: a list of count
While (has More Intermediate data)
int result= get result from HDFS
For each value V
 result=result + parseInt (V)
Obtain the result as a String
```

3.3 Proposed Method

In distributed computing, association rule mining is more concerned with efficient and scalable design of algorithm and requires less communication of data in network. HDFS provides efficient and scalable data that is processed in location itself instead of transferring the data to the computation section. In HDFS, only information required for data processing is sent through network and handling of nodes is done effectively in distributed environment.

3.3.1 Construction and Mining of Input Split FP-Tree

The Proposed algorithm Mining Association Rule using Map Reduce is simple and flexible because of hadoop framework that is implemented using Map-Reduce object oriented programming paradigm. FP-Tree is reduced form of stored data that requires two times scanning of database. But if the threshold value is very less for huge amount of data set then the number of levels in a tree will be a large number that increases complexity. Root node of tree is labeled as null. Second time it scans a database in order to create a branch node for each transaction. The item set are arranged in a descending order based on their support count values and the list is denoted as L={I1:8, I2:7, I3:6, I4:5, I5:2}. By scanning the first transaction which contains three item sets are arranged in L order i.e., <I1, I2, I3, I4, I5>.It leads to the construction of first branch node of tree with three nodes where I1 is the child of root node and I2 is linked to I1. In second transaction T2 contains I2, I3, I4 item sets, which are arranged in L, order Here another branch is created to tree I2 is child to root, I3 is linked to I2 and I4 is linked to I3 by assigning a support count as 1 to each item sets. But in third transaction, however the branch could share common prefix I1 with an existing path for Transaction T1 and by incrementing the count value of each item by 1where <I3:1> is linked to <I1:2>. Finally FP-Tree is obtained by considering all the transactions. Therefore considering I4 as a suffix , the prefix path for I4 is <I1,I2, I3:1>, <I1,I3:1>, <I1:1>, <I2,I3:1> and <I1,I2:1> which forms a conditional pattern-base. The conditional FP-Tree for I4 contains all paths that are mentioned-above as its support count is less than the minimum support threshold value. The FP-Tree contains the parameters like prefix path subtree and header table. The header table has three attributes namely Node Name, support count and node link. In mining process having header table information will be useful. But it will be burden for distributed and massive data. In this paper we proposed input split FP-tree helps us to store only minimum information required for processing. It is represented as follows: <bucket addressing, bucket item, bucket count, index of parent node>.

Table 1: Transactional Data Set

TID	Item sets
T1	I1 I2
T2	I2 I3 I4
T3	I1 I3 I4 I5
T4	I1 I4 I5
T5	I1 I2 I3
T6	I1 I2 I3 I4
T7	I1
T8	I1 I2 I3
T9	I1 I2 I4
T10	I2 I3 I5

Journal of Theoretical and Applied Information Technology
20th February 2016. Vol.84. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The input split FP-Tree is mined by calling FP-growth (ISFP-Tree, l_1) which is implemented as follows: Map function operates on <Item set, supp_count> which is a key, value pair and produces the intermediate data in the form of <Item set, supp_count> as conditional pattern base. Since the intermediate data contains duplicates item sets that are to be pruned using conditional input split FP-tree handled by Mapper Job. For performing this task we require shuffling phase eliminates redundant keys and thereby by combing all the support count which are the identical item sets. Finally Reducer job produces frequent pattern are constructed from Conditional Input Split FP-Tree that produces another (key, value) pair as <item sets, support_count>.

Table 2: Input Split Data Structure Corresponding to FP-Tree

Bucket Address	0	1	2	3	4	5	6
Item	root	I1	I2	I2	I3	I4	I3
Count	0	8	2	5	1	1	2
Parent	null	0	0	1	1	1	2

Bucket Address	7	8	9	10	11	12	13	14
Item	I3	I4	I4	I5	I4	I5	I4	I5
Count	3	1	1	1	1	1	1	1
Parent	3	3	4	5	6	6	7	9

The procedure MRIS_DataStructure illustrates to retrieve input split 'S' by scanning database 'D' with set of items I={I_1, I_2,……..}. If there are no input splits. It will discover the buffer size 'b' by an analytical processing and deficiency between the successive transactional identifier is computed (TID_i and TID_{i+1}). The deficiency indicates the irrelevant itemsets. If the deficiency is less than min_supp threshold value (1%) then increases the buffer size to store TID_{i+1}.

Algorithm: Mining Association Rule using Map Reduce paradigm
Input: D, a transactional database with a set of transactional identifiers TID={TID_1, TID2, …, TIDn} and min-sup, the minimum support count threshold value.
Output: Input Splits Frequent Pattern 'S' with <key, value> pairs.

Method:
1. The FP-tree is constructed in the following steps:
// Store the transactional database in HDFS.

a. Huge amount of data is divided into input splits are based on data set size and HDFS block size.
b. For each input split the record reader scans only one record at a time and computes the support count and itemsets. Prefix path subtree is constructed from the distributed input split FP-tree.

2. MapReduce Input Split Data Structure by calling the procedure (MRIS_DataStructure)

a. if (! Scan D contains (S))
b. {
c. def:=0,b:=0
d. i:=S's index in TID
e. while((def<min_supp)and(b+def+length(TID_i) < number of transactions))
f. {
g. b:=b + def + length(TID_i)
h. def:= ref (TID_{i+1})-ref(TID_i)-length(TID_i)
i. i:=i+1
j. }
k. starting from ref(S), identify Itemsets from D
l. }
m. return scan 'D'.get_Input Splits (S)

3. The input split FP-Tree is mined by calling the procedure (ISFP-tree, l_1)

a. if input split tree consist of single path
b. {
c. for each aggregation (denoted as l_2) of Name Node in the path produces l_2 U l_1 with supp_count= min_sup of Name Node in l_2.
d. }
e. else
f. for each link l_i in header table of ISFP-Tree
g. {
h. Generates pattern l_2=l_iUl_1 with supp_count =l_i.supp_count.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

i. Generate l_2's conditional pattern base and l_2's conditional ISFP-Tree
j. If $ISTree_{l2} \neq \Phi \neq \emptyset$not null then
k. Call FP-growth ($ISTree_{l2}$, l2)
l. }

4. For optimizing the output file more specifically the additional operation ⊕ is to be performed for reducer that is in the form of (key, value) pair and the function 'f ' is defined as

$$f(\{V_0,V_1,V_2,\ldots\ldots..V_n\})=V_1 \oplus V_2 \oplus V_3 \oplus \ldots\ldots\ldots \oplus V_n$$

To perform this task, the programmer should use additional reducer by declaring it as an interface. The optimized Input Split FP-Tree is measured by using Gain Ratio per Sensitive Loss data (GRSL). The highest value of GRSL is considered as more relevant one. The GRSL of specific: base ⟶ derive (base) is formulated as follows:

$$GRSL(S_i) = \frac{GR(S_i)}{(SL(S_i)+1)} \quad (1)$$

The term GR (S_i) is the Gain Ratio of a particular Input Split FP-Tree S_i and SL (S_i) is the sensitive lossy data. Both the terms can be calculated by using certain statistical measurements.

$$GR(S_i) = \frac{Gain(S_i)}{Input\ Split(S_i)} \quad (2)$$

$$Input\ Split_{S_i}(T) = -\sum_{d \in drive(b)} \frac{|T_d|}{|T_b|} \log_2 \frac{|T_d|}{|T_b|} \quad (3)$$

$$Gain(S_i) = I(T) - I_{S_i}(T) \quad (4)$$

$$I(T) = -\sum P_b \log_2(P_b) \quad (5)$$

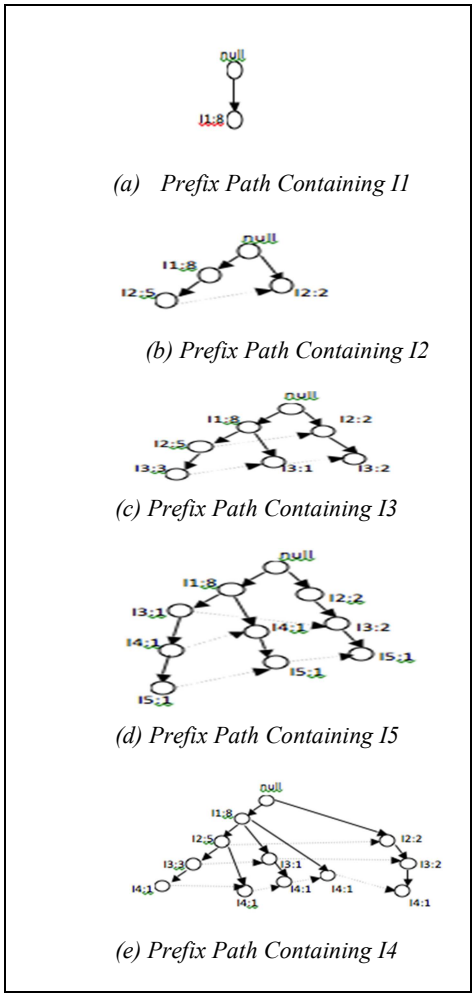
$$I_{S_i}(T) = \sum_{d \in drive(b)} \frac{|T_d|}{|T_b|} \times I(T_d) \quad (6)$$

Let T_x denotes set of training dataset containing items which are generalized to 'x' where $|T_x|$ is the number of tuples in T_x. I (T_x) denotes the entropy of T_x. Let $|(T_x, sv)|$ describes number of tuples with a sensitive value SV in T_x and Information gain of training dataset I (T_x) is formulated as follows:

$$I(T_x) = -\sum_{sv \in SL} \frac{|T_x, SV|}{|T_x|} \log_2 \frac{|T_x, SV|}{|T_x|} \quad (7)$$

A data vector of split point of all itemsets in a transactional database is defined as anonymization hierarchy (AH).
AH = < S_1, S_2, …… ……,S_n> where S_i $1 \leq i \leq n$ is the split point of Input Split FP-Tree. Let A_b indicates anonymity of before forming Input Split and A_d (S_i) indicates anonymity of after forming Input Splits sensitive lossy data is measured as follows:

$$SL(S_i) = A_b - A_d(S_i) \quad (8)$$



(a) Prefix Path Containing I1

(b) Prefix Path Containing I2

(c) Prefix Path Containing I3

(d) Prefix Path Containing I5

(e) Prefix Path Containing I4

Figure 5: Mining Input Split FP-Tree based on Input Split Data Structure corresponding to FP-Tree by considering prefix path subtree ending with each itemset.

Journal of Theoretical and Applied Information Technology
20th February 2016. Vol.84. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. EXPERIMENTAL RESULTS

To Calculate the performance of mining the association rule using Map Reduce Object Oriented programming paradigm the data sets are generated by conducting the experiments on multi-node hadoop commodity machines and each node with the configuration 320GB of hard disk, Intel core i7 2.5GB and 4 GB of RAM. Hadoop is open source software, which requires JDK to be installed on Linux system. The hadoop can be verified using jps command. To verify the performance of MapReduce copy files from local drive to HDFS. As the name node records the metadata for each file.

To compare the performance of Input Split Frequent Pattern Tree using Map Reduce algorithm with Tidset based Parallel FP-Tree algorithm that includes Apriori and FP-growth algorithm. But TPFP-Tree uses divide and conquer approach and produce a better result in PC clusters and grid computing. Parallel mining of Association Rule from text database on the set of commodity machines as clusters. But the data set generated using IBM's Quest Synthetic, TPFP-Tree does not perform effectively as it distributes the data in cloud environment among the set of nodes [31] [36] [37]. The results are analyzed by conducting experiments.

Table 3: The Datasets that are taken for ARM

Dataset	Items	Transaction	Size (MB)
Chess	90	3495	0.45
Yeast	850	150000	4.96
FoodMart	43000	1000005	35.60

Figure 6, 7 and 8 shows experimental results conducted for Chess, Yeast and FoodMart dataset that illustrate execution time (in minutes) of Input Split FP-Tree and Tidset Parallel FP-Tree. For large-scale datasets Input Split Frequent Pattern Tree using MapReduce shows more efficient algorithm when compared to other three algorithms like Apriori, FP-Tree and Tidset parallel FP-Tree especially when minimum support threshold value is very low. In Table 4 MapReduce Time and Runtime for the various datasets. In HDFS, the number of mapper plays a vital role in the runtime of Job Tracker and Task Tracker. The number of reducer determines how many input split files are required to produce the desired output file. If there exist only one reducer, then the output file will be larger in size. This output file becomes a heavy weight component that has to be transferred to various data node over network. As the experiments are conducted on various dataset like Chess, Yeast and Foodmart datasets by considering 1% minimum support threshold value.

Table 4: MapReduce Time and Runtime for the Various Datasets

Transactions	Map Time	Reduce Time	Runtime (in minutes)
3495	3.57	23.81	4.3
150000	21.92	45.91	12.2
1000005	41.21	86.7	22.5

Figure describes the execution time in minutes and number of reducer. Table 5 shows runtime between two algorithms. Input Split FP-Tree performs better than Tidset based Parallel FP-Tree because the proposed algorithm uses MapReduce paradigm for larger dataset as it generates input split files. Like Input Split FP-Tree algorithm constructs subtree in map phase and mining Input Split FP-Tree in reducer phase.

Table 5: Runtime/Execution Time between the two Algorithms

Dataset	Input Split FP-Tree	Tidset Parallel FP-Tree
Chess	4.3	37.5
Yeast	12.2	87.2
FoodMart	22.5	97.5

The Tidset Parallel Frequent Pattern Tree performs a good processing for medium size data set. As day-by-day it increases the data heterogeneity and complexity. For such kind of a data the TPFP-Tree algorithm increases exponentially [20] [21]. But for mining association rule using Map Reduce algorithm gives fast and efficient scalability. The Input Split Frequent Pattern-Tree construction is based on each individual by dividing the data set based on Hadoop block size and merging the obtained result performs

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

lot of information sharing among data nodes and Node Name.

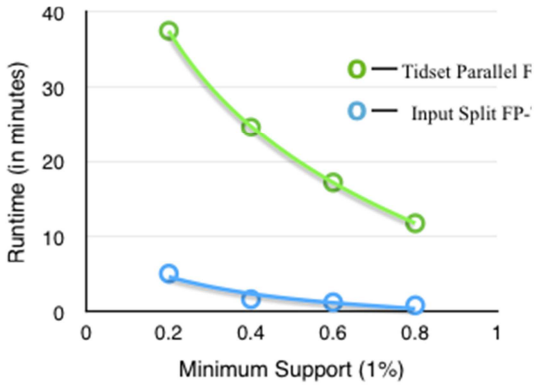


Figure. 6: Runtime in Minutes for Chess Dataset and Considering 1% Minimum Support ThresholdVvalue.

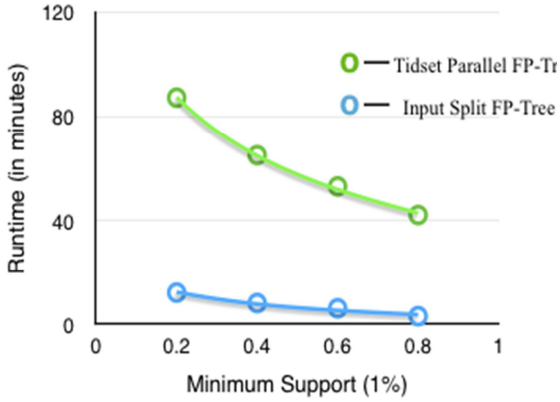


Figure 7: Runtime in Minutes for Yeast Dataset and Considering 1% Minimum Support Threshold Value.

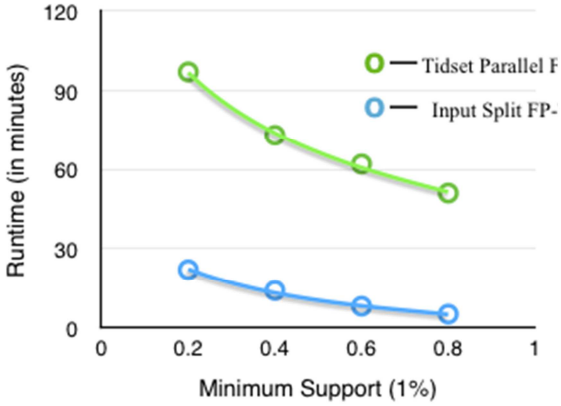


Figure 8: Runtime in Minutes for FoodMart Dataset and Considering 1% Minimum Support Threshold Value.

In mining association rule using Map Reduce algorithm, as execution time decreases as the number of nodes increases. Thus storage and processing of a data in hadoop gives a better performance.

5. CONCLUSION

In this paper we proposed Input Split Frequent Pattern Tree algorithm using Map Reduce paradigm that uses hadoop cluster efficiently in order to retrieve the frequent item sets from huge data sets. In this algorithm data is stored in hadoop cluster inside the hadoop distributed file system and it is processed or implemented using Map Reduce object oriented programming paradigm. Unlike in the existing algorithm Tidset Parallel Frequent Pattern Tree algorithm uses divide and conquer approach but the data storage and processing is done efficiently only for medium data set in cloud environment but for large data set. In this algorithm also it requires a lot of information sharing between name node and data node in hadoop master and slave architecture. Because of the implementation of Map Reduce model in hadoop and the presence of intermediate data the processing has became very easy. Based on the experimental results Input Split Frequent Pattern Tree is fast and scalable than Tidset Parallel Frequent Pattern Tree algorithm.

REFRENCES:

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of the 20th International Conference on Very Large Data Bases, 1994, pp. 487–499.
[2] Apache Hadoop, http://hadoop.apache.org.
[3] Apache Mahout, http://mahout.apache.org.
[4] bigdata@csail, http://bigdata.csail.mit.edu/.
[5]‘Big Data’ has Big Potential to Improve Americans’ Lives, Increase Economic Opportunities, Committee on Science, Space and Technology (April 2013). URL http://science.house.gov/press-release.
[6] D. Borthakur, J. Gray, J.S. Sarma, K. Muthukkaruppan, N. Spiegelberg, H. Kuang, K. Ranganathan, D. Molkov, A. Menon, S. Rash, R. Schmidt, A. Aiyer, Apache Hadoop Goes Realtime at Facebook, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2011),ACM, New York, USA, 2011, pp. 1071–1080.
[7] D. Borthakur, “Facebook has the worlds largest

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

hadoop cluster," *Re- trieved April*, vol. 20, p. 2012, 2010.

[8] Y. Chen, S. Alspaugh, R. Katz, Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads, Proceedings of the VLDB Endowment 5 (12) (2012) 1802–1813.

[9] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "Samr: A self- adaptive mapreduce scheduling algorithm in heterogeneous environ- ment," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 2736–2743.

[10] Y. Chen, S. Alspaugh, D. Borthakur, R. Katz, Energy Efficiency for Large-Scale MapReduce Workloads with Significant Interactive Analysis, in: Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys 2012), ACM, New York, USA, 2012, pp. 43–56.

[11] "Cogset: A unified engine for reliable storage and parallel processing," in *Network and Parallel Computing, 2009. NPC'09. Sixth IFIP International Conference on*. IEEE, 2009, pp. 174–181.

[12] J. P. Dijcks, "Oracle: Big data for the enterprise," *Oracle White Paper*, 2012.

[13] D. Fisher, R. DeLine, M. Czerwinski, S. Drucker, Interactions with Big Data Analytics, Interactions 19 (3) (2012) 50–59.

[14] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[15] L.Glimcher, R.Jin , G.Agarwal, Middleware for data mining application on clusters and grids, Journal of Parallel and Distributed Computing 68 (1) (2008). 37-53.

[16] GrahneG, ZhuJ(2005)Fast algorithms for frequent itemset mining using FP-trees. IEEETransKnowl Data Eng 17(10): 1347–1362

[17] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Journal of Data Mining and Knowledge Discovery (8) - 1 (2004) 53 -87.

[18] A. Hadoop, "Apache hadoop nextgen mapreduce (yarn)," 2013.

[19] J.D.Holt, S.M.Chung, Parallel mining of association rules from text databases on a cluster of workstations, Proceeding of 18th International Symposium on Parallel and Distributed Processing, 2004.

[20] E.H.S Han, G.Karypis, Scalable parallel data mining for association rules, IEEE Transaction on Knowledge and Data Engineering 12 (3) (2000).

[21] E.H.S Han, G. Karypis, V.Kumar, Scalable parallel data mining for association rules,IEEE Transaction on Knowledge and Data Engineering. 12 (3) (2000). 352-377.

[22] HanJ,PeiJ,YinY(2000)Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data (SIGMOD '00), pp 1–12

[23] The Intel science and technology center for big data, http://istc-bigdata.org.

[24] A.Javed , A. Khokhar, Frequent pattern mining on message passing multiprocessor systems, Distributed and Parallel Database 16 (3) (2004) 321 – 334.

[25] R. T. Kaushik and M. Bhandarkar, "Greenhdfs: Towards an energy- conserving storage-efficient, hybrid hadoop compute cluster," in *Pro- ceedings of the USENIX Annual Technical Conference*, 2010.

[26] Mannila H, Motwani R (eds) Proceedings of the second SIAM international conference on data mining. Arlington, VA, pp 457–473

[27] A.MateiZaharia, A. Joseph, and I. RandyKatz, "Improving mapreduce performance in heterogeneous environments," 2010.

[28] T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas, "Mrshare: sharing across multiple queries in mapreduce," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 494–505, 2010.

[29] PasquierN, TaouilR,BastideY,StummeG, LakhalL (2005): Generating a condensed representation for association rules. J Intell Inf Syst 29–60

[30] PeiJ, HanJ, MaoR (2000) CLOSET: an efficient algorithm for mining frequent closed itemsets. In: ACM SIGMOD workshop on research issues in, data mining and knowledge discovery, pp 21–30

[31] M.S.Perez , A. Sanchez , V.Robles , P.Herrero, J.M Peria, Design and implementation of a data mining grid – aware architecture, Future Generation Computer Systems 23 (1) (2007) 42-47.

[32] P. Russom, Big Data Analytics, TDWI best practices report, The Data Warehousing Institute (TDWI) Research (2011).

[33] Raymond Gardiner Goss Kousikan Veeramuthu , "Heading Towards Big Data Building A Better Data Warehouse For More

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Data, More Speed, And More Users", 978-1-4673-5007-5/13/ ©2013 IEEE

[34] S. Shankar, A. Choi, and J.-P. Dijcks, "Integrating hadoop data with oracle parallel processing–an oracle white paper," *Oracle Corporation*, 2010.

[35] S. V. Valvag and D. Johansen, "Oivos: Simple and efficient distributed data processing," in *High Performance Computing and Communica- tions, 2008. HPCC'08. 10th IEEE International Conference on*. IEEE, 2008, pp. 113–122. □

[36] C.H.Wu, C.C.Lai, Y.C.Lo, An empirical study on mining sequential patterns in a grid computing environment, Expert systems with Application 39 (5) (2012) 5748-5757.

[37] K.M.Yu, J.Zhou, Parallel TID- based frequent pattern mining algorithm on PC clusters and grid computing system, Expert system with Applications 37 (3) (2010) 2486-2494.

[38] J.Zhou, K.M.Yu, Parallel TID- based frequent pattern mining algorithm on PC clusters and grid computing system, Expert system with Applications 37 (3) (2010) 2486-2494.

[39] J.Zhou, K.M.Yu, Tidset - based parallel FP-tree algorithm for the frequent pattern mining problem on PC clusters, Lecture Notes in Computer Science 5036 (2008).18-28.

[40] Zibin Zheng, Jieming Zhu, and Michael R. Lyu , "Service-generated Big Data and Big Data-as-a-Service: An Overview" , 978-0-7695- 5006-0/13 © 2013 IEEE