International Journal For Technological Research In Engineering
Volume 2, Issue 4, December-2014 *ISSN (Online): 2347 - 4718*

AN EFFICIENT IMPLEMENTATION OF MINING FREQUENT ITEMSETS BASED ON HADOOP-MAPREDUCE MODEL

Sankar Rela
M.Tech, Sri Venkateswara University
Department of Computer Science & Engineering
Tirupathi, Andhra Pradesh, India.

ABSTRACT: Finding frequent itemsets is one of the most important fields of data mining. Apriori algorithm is the most established algorithm for finding frequent itemsets from a transactional dataset; however, it needs to scan the dataset many times and to generate many candidate itemsets. Unfortunately, when the dataset size is huge, both memory use and computational cost can still be very expensive. In addition, single processor's memory and CPU resources are very limited, which make the algorithm performance inefficient. Parallel and distributed computing are effective strategies for accelerating algorithms performance.

In this paper, we have implemented an efficient MapReduce Apriori algorithm (MRApriori) based on Hadoop-MapReduce model which needs only two phases (MapReduce Jobs) to find all frequent itemsets. And also using apache hive for data warehouse. we can generate reports by using hive.

Keywords: Hadoop, MapReduce, Parallel Computing, Distributed Computing, Apriori Algorithm, Frequent Itemset, Data Mining, Association Rules.

I. INTRODUCTION

Data mining is the efficient discovery of previously unknown patterns in large datasets. It has attracted a lot of attention from both research and commercial communities for finding interesting information hidden in large datasets. One of the most important areas of data mining is association rule mining; its task is to find all subsets of items which frequently occur and the relationship between them by using two main steps: finding frequent itemsets (core step) and generating association rules Apriori is the most established algorithm for finding frequent itemsets from a transactional dataset; however, it needs to scan the dataset many times and to generate many candidate itemsets. Unfortunately, when the dataset size is huge, both memory use and computational cost can still be very expensive. In addition, single processor's memory and CPU resources are very limited, which make the algorithm performance inefficient. Furthermore; because of the exponential growth of worldwide information, enterprises (organizations) have to deal with an ever growing amount of data. As these data grow past hundreds of gigabytes towards a terabyte or more, it becomes nearly impossible to process (mine) them on a single sequential machine. The solution for the above problems is parallel and distributed computing. Parallel and distributed computing offer a potential solution for the above problems if the efficient and scalable parallel and distributed algorithm can be implemented. Such easy and efficient implementation can be achieved by using Hadoop-MapReduce model which is a programming model for easily and efficiently writing applications that process vast amount of data in-parallel on large clusters of commodity hardware in a reliable, fault-tolerance manner.

II. MAPREDUCE MODEL

MapReduce is one of the earliest and best known models in parallel and distributed (cluster) computing area, created by Google in 2004, based on C++ language. It is a programming model and associated implementation for processing and generating large data sets in a massively parallel and distributed manner. One of the attractive qualities of MapReduce model is its simplicity: a MapReduce application (job) consists only of two functions, Map and Reduce functions. Developers write the map function that processes a key/value pair to generate a set of intermediate key/value pairs, and the reduce function that merges all intermediate values associated with the same intermediate key. Many data mining areas such as: association rule, clustering and classification algorithms have been implemented based on MapReduce model .

A. Hadoop Framework

Hadoop is an open source framework, created as an open source implementation of Google MapReduce architecture, based on Java language, sponsored by Apache Software Foundation. The Apache Hadoop software library is defined by Apache as "a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. It originally consists of two models: MapReduce; the programing model, Hadoop Distributed File System (HDFS); the distributed storage model which designed after Google File System (GFS). Now it supports additional models and systems such as: HBase; a distributed column-oriented database, Hive: a data warehouse system, Avro: a data serialization system, Chukwa: a data collection system, ZooKeeper: a high-performance coordination service for distributed application, and Pig; a high level data-flow

language. Hadoop-MapReduce is a programming model for easily and efficiently writing applications which process vast amount of data (terabyte or more data sets) in-parallel on large clusters of commodity hardware in a reliable, fault-tolerance manner. A MapReduce program (job) partitions the input dataset into independent splits which are processed by the map tasks (map task per split) in a completely parallel manner. The Hadoop framework combines and stores the maps output as a set of intermediate key/value pairs which are then fetched as an input to the reduce tasks.

III. RELATED WORK

As we mentioned in the previous two sections, the performance of sequential Apriori algorithm is inefficient, especially when dealing with huge data sets. Thus, parallel Apriori algorithms were proposed . But in general, parallel and distributed computing are wide and varied fields and come with many problems that were not exists in sequential computing; so, a lot of time and effort are needed to handle and solve those problems. Such problems are: load balancing, data partition and distribution, jobs assignment and monitoring, parameters passing between nodes…etc. As mentioned above, parallel and distributed computing are wide and varied fields but the key distinctions of Hadoop are its simplicity, scalability, and reliability which solve most (if not all) of the challenges and problems listed above easily and efficiently. Because of the benefits of MapReduce model, some parallel Apriori algorithms are implemented using Hadoop-MapReduce model. In general, we can classify those algorithms as two classes: one-phase and k-phases. In one-phase class; the algorithm needs only one phase (MapReduce job) to find all frequent itemsets.

IV. PSEUDO-CODE

Pseudo code for MRApriori Algorithm

```
Map Task: // one for each split
Input: Si // Split i, line = transaction
Output: <key, 1> pairs, where key is an element
of candidate itemsets.
1.  Foreach transaction t in Si
2.    Map(line offset,t) // Map function
3.      Foreach itemset I in t  /* I = all possible sub
      sets of t  */
4.        Out (I,1);
5.      End foreach
6.    End map
7.  End foreach
8.  End

Reduce Task:
Input: <key2, value2> pairs,
minimum_support_count, where key2 is an
element of the cand. Itemsets and value2 is its
occurrence in each split
Output: <key3, value3> pairs, key3 is an element
of frequent itemsets and value3 is its occurrence in
the whole dataset.
1.  Reduce (key2, value2) // Reduce fun.
2.    Sum=0;
3.    While (value2.hasNext())
4.      Sum+= value2.getNext();
5.    End while
6.    If (sum>= min_sup_count)
7.      Out (key2, sum);
8.  End if
9.  End reduce
10. End
```

Fig:1 Pseudo-code of One-phase Algorithm

In k-phases class (k is maximum length of frequent itemsets); the algorithm needs k phases (MapReduce jobs) to find all frequent k-itemsets [14, 15, 16]: phase one to find frequent 1-itemset, phase two to find frequent 2-itemset, and so on.

```
Map Task: // one for each split
Input: Si // Split i, line = transaction
Output: <key, 1> pairs, where key is an element
of candidate k-itemset
1.  Foreach transaction t in Si
2.    Map(line offset,t) // Map function
3.      Foreach item I in t // I= token
4.        Out (I,1);
5.      End foreach
6.    End map
7.  End foreach
8.  End

Reduce Task:
Input: <key2, value2> pairs,
minimum_support_count, where key2 is an
element of the candidate k-itemset and value2 is its
occurrence in each split
Output: <key3, value3> pairs, where key3 is an
element of frequent k-itemset and value3 is its
occurrence in the whole dataset.
1.  Reduce (key2, value2) // Reduce fun.
2.    Sum=0;
3.    While (value2.hasNext())
4.      Sum+= value2.getNext();
5.    End while
6.    If (sum>= min_sup_count)
7.      Out (key2, sum); // collected in Lk
8.  End if
9.  End reduce
End
```

Fig:2 Pseudo-code of k-phases Algorithm & k=1

```
Map Task: // one for each split
Input: Si, Lk-1
Output: <key, 1> pairs, key is an element of
candidate k-itemset
1.  Read Lk-1 from DistributeCache.
2.  Ck= ap_gen(Lk-1) // self-join
3.  Foreach transaction t in Si
4.    Map(line offset,t) // Map function
5.      Ct = subset (Ck, t);
6.      Foreach candidate c in Ct
7.        Out (c,1);
8.      End foreach
9.    End map
10. End foreach
11. End

Reduce Task:
The same reduce task as the previous phase
```

Fig: 3 Pseudo-code of k-phases Algorithm & k>=2

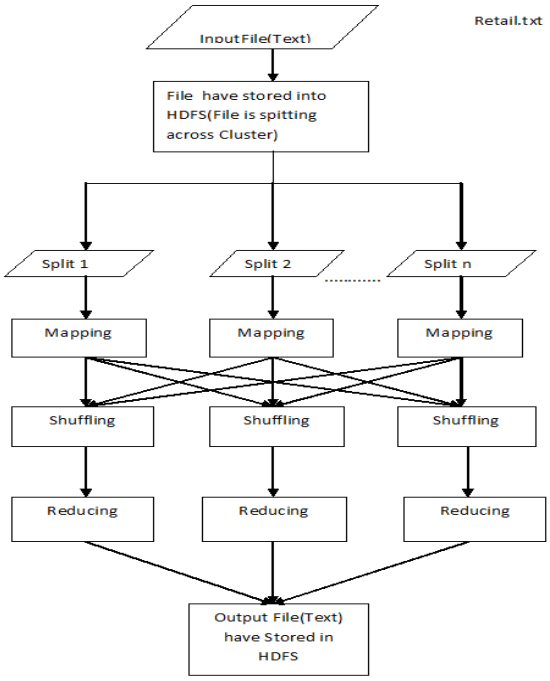

Fig 4: Flow diagram for mapreduce

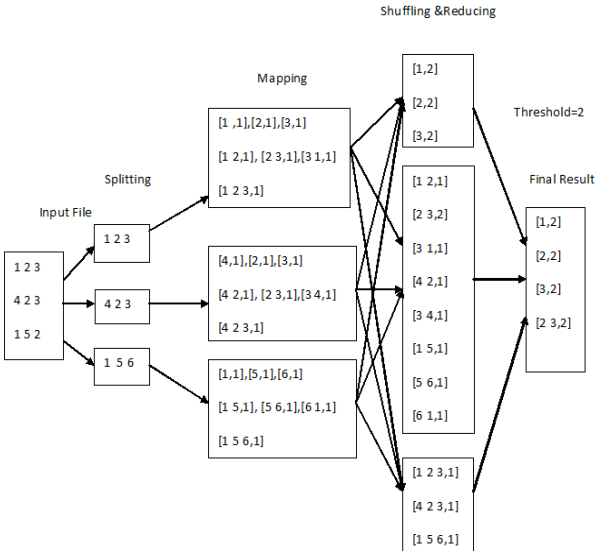

Fig 5: Flow diagram for MRApriori Algorithm

V. APACHE HIVE

The Apache Hive is data warehouse software facilitates querying and managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL. At the same time this language also allows traditional map/reduce programmers to plug in their custom mappers and reducers when it is inconvenient or inefficient to express this logic in HiveQL. After generating frequent itemsets,then we should load the itemsets into hive, which generating reports requested by client .In between we are using web server(tomcat) to communicate with hive.

VI. CONCLUSION

We have proposed a new implementation of Apriori algorithm based on Hadoop-MapReduce model, called MRApriori algorithm. we have implemented MRApriori algorithm on single machine and distributed environment(cluster environment which containing three nodes cluster). The results showed that: one-phase algorithm is inefficient and impractical; k-phases algorithm is effective and has execution time close to our proposed algorithm since, the experiments were done on a single machine and the combine output records did not move from map worker to reduce worker over the network.Apache hive is also used to generating reports.

REFERENCES

[1] http://hadoop.apache.org/
[2] http://en.wikipedia.org/wiki/MapReduce
[3] http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[4] http://bigdatauniversity.com/bdu-wp/bdu-course/hadoop-fundamentals-i-version-3/
[5] http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html
[6] http://www.cbtnuggets.com/it-training-videos/course/apache-hadoop
[7] Ye Y. & Chiang C. (2006). A Parallel Apriori Algorithm for Frequent Itemsets Mining. Proc. of the 4th International Conference on Software Engineering Research, Management and Applications (SERA ’06). Seattle, WA, IEEE: 87 – 94.
[8] Yu K. & Zhou J. (2008). A Weighted Load-Balancing Parallel Apriori Algorithm for Association Rule Mining. Proc. of the International Conference on Granular Computing (GrC ’08). Hangzhou, IEEE: 756 – 761.
[9] Yang X.Y., Liu Z. & Fu Y. (2010). MapReduce as a Programming Model for Association Rules Algorithm on Hadoop. Proc. of the 3rd International Conference on Information Sciences and Interaction Sciences (ICIS ’10). Chengdu, China, IEEE: 99 – 102.

