Volume 5, Issue 1, January 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Line of Attack to Accentuate FIM in Cloud Computing

Vijay Swaroop A
Dept. of CSE, Visvesvaraya Technological University
India

Abstract— Cloud computing can provide infrastructure to massive and complex data of data mining. Data mining is a process of extracting potentially useful information from raw data, so as to improve the quality of the information service. This paper discusses at brief, Apriori algorithm to obtain frequent itemsets and at surficial layer tells how FIM- Frequent Itemset Mining is used in cloud computing to analyse data. Then, the paper aims at extracting frequent itemsets using candidate generation with n nodes on Map-Reduce programming model and its development platform-Hadoop.

Keywords— Hadoop, Map-Reduce, Frequent Itemset Mining, Association Rule Mining

I. INTRODUCTION

The Internet is becoming a vital tool in our life, as its users are becoming more numerous. The Cloud, as it is often referred to, involves using computing resources – hardware and software – that are delivered as a service over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). It joins the ranks of terms including: grid computing, utility computing, virtualization, clustering, etc. The cloud is a virtualization of resources that maintains and manages itself.

Mining frequent Itemset is an active area of research. As the database size increases, the computational time and required memory also increases. This problem could be solved by using algorithms that read the input database more than once and count/store interesting parameters instead of keeping the database in the memory.

Apriori is one of the most popular data mining approaches for finding frequent itemsets from transactional datasets.

II. PRELIMINARIES

Applying frequent itemset mining to large databases is problematic. Apriori algorithm, which is a level wise breadth first search based algorithm, is a solution. But, the memory requirements for handling the complete set of candidate itemsets blows up fast and renders Apriori based schemes very inefficient to use on single machines.

Parallel programming is becoming a necessity to deal with massive amounts of data. Out of the two parallel programming architectures, distributed programming is advantageous as it does not have scalability issues. Map-Reduce model is a parallel programming framework proposed by Google, simplifies the programming for distributed data processing. Firstly, the model allow users to easily handle large-scale data; then all the procedures for computing are abstracted into two basic operations of Map and Reduce. In Map stage, data will be decomposed into smaller scale, and executed on different nodes of the cluster, and the results are integrated summary in the Reduce phase.

Map-Reduce model is a simple but very effective parallel programming model. Apache Hadoop Map-Reduce uses the HDFS distributed parallel file system

For data storage, which stores the data across the local disks of the computing nodes while presenting a single file system view through the HDFS API. Hadoop has an architecture consisting of a master node with many client workers and uses a global queue for task scheduling, thus achieving natural load balancing among the tasks. The Map Reduce model reduces the data transfer overheads by overlapping data communication with computations when reduce steps are involved.

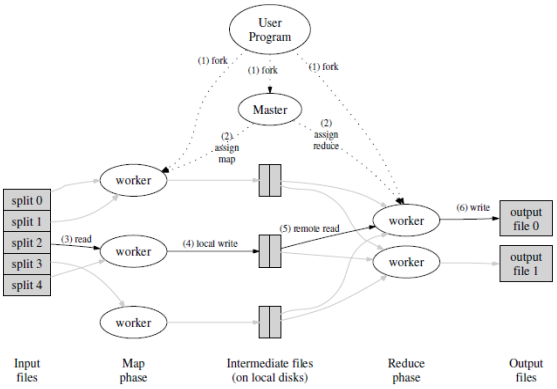

Fig. 1 Map – Reduce Model

III. GUIDELINES

Association Rule Mining (ARM)'s task is to find all the frequent subsets of items and relationship between them. ARM is performed in two phases: The establishment of frequent item sets and the establishment of association rules. Apriori algorithm generates Boolean association rules.

Apriori algorithm uses say, 'I' as a notation to denote a set of items and let a transaction defined by 'T'. T is defined as (tid,X), where tid is transaction identifier and X is a subset of items from I. An itemset is said to be frequent, only if, its support value 'Y' is greater than a given threshold ∞, which is called as minimum support or minsup in short.

$$Sup(Y) = |\{tid|Y \text{ (subset)} X , (tid,X) \text{ (belongs to)} D\}|$$

Map – Reduce paradigm requires key – value pairs for input and output, so first the data set has to be converted to this format. Transactional databases can be converted to key value pairs where the key is the transaction ID and value is the set of items from the transaction. The challenge is to find the right place to separate the internal data and find a good key – value representation for the data passing between the map and the reduce phase.

Apriori algorithm finds all frequent itemsets by scanning the database time after time. This algorithm waste lots of memory and takes more time, because it lacks in parallelization. So, we need to have a parallelism based Apriori algorithm to yield better results.

IV. P – ARM IN MAP - REDUCE

The parallelized Apriori algorithm has to be implemented in Hadoop Map Reduce model with the following phases:

a. Partitioning and distributing the data

Before the start of Map phase, the Hadoop framework will split the input database D into smaller chunks. The size of the split depends on the framework configuration and how the data is distributed across the file – systems on the machines in the cluster. In the sense, the transaction database needs to be converted into Boolean matrix. If the transaction database has 'm' items and 'n' transactions, the matrix will have 'm+1" rows and 'n+2' columns. The first column registers "items" and first row registers "TID "of the transactions. HDFS store the input data. HDFS also helps in data localization for Map/ Reduce task.

b. Formatting the data subsets

This step formats the data subsets into key, value pairs.

c. Mapper task execution

This step is used to count all the items and subsequently to count the frequency of each candidate. The candidate itemset C^p is generated

d. MinSup comparison

The minimum support needs to be compared with the support of item sets, if the support of item set is smaller than the min – support the row of the item sets will be deleted.

e. Combiner function

Combiner produces itemset and support count. Combiner then divides the intermediate pairs into r different partitions.

f. Reduce Task execution

Here, the key itemsets are first sorted. The support count of the same candidates needs to be added to get the actual support count in the whole transaction database. The frequent itemset is the actual support count, which is dentoted by L^p.

V. PSEUDO – CODE FOR APRIORI ALGORITHM

```
Ck : Candidate Itemset of size k
Lk : Frequent Itemset of size k
L1 = {Frequent items};
For(k=1;LK != Null; k++) do begin
CK+1 = Candidates generated from Lk ;
For each transaction t in database do
Increment the count of all candidates in Ck+1
That are contained in t
LK+1 = candidates in Ck+1 with minsup
End
Return UKLK
Apriori algorithm uses two steps: JOIN and PRUNE
Join : CK is generated by joining Lk-1 with itself
Prune: Any (k-1) itemset that is not frequent cannot be a subset of a frequent k – itemset
```


Frequent itemset can be generated by Apriori algorithm. Then. Association rules can be generated using minsup and minimum confidence.

VI. P – ARM FOR FIM

The algorithm to be implemented with the above phases is listed below. The following algorithm introduces parallel scan to the traditional Apriori algorithm in order to reduce the time wasted.

a. The database should first be horizontally divided into n data subsets and distribute it to m nodes.
b. Each node then scans its own data sets and generate candidate item set C^p. The support count of each candidate itemset is set to 1.
c. C^p is then divided in to r partitions and sent to r nodes with their support count.
d. The final practical support is produced and the frequent item set L^p is determined after comparing with minimum support count Minsup.
e. The output of r nodes will be merged to get the global frequency itemset L.

Here, we are traversing the transaction database only once, thereby reducing the time wasted.

VII. CONCLUSIONS

In this paper parallel association rule mining strategy in the cloud computing environment is studied. A parallelized Apriori algorithm on Map Reduce programming model on the Hadoop platform is also proposed and studied. The same needs to be implemented and tested on a large database and prove that the theoretical study is better than the sequential algorithm.

REFERENCES

[1] Jeffrey Voas and Jia Zhang, "Cloud Computing: New Wine or Just a New *Database Systems Journal* vol. III, no. 3/2012.
[2] K. Bhaduri, K. Das, K. Liu, H. Kargupta, and J. Ryan, Distributed Data Mining Bibliography, 2011.
[3] S. C. T. Chu, S. K. Kim, Y. A. Lin, Y. Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, Map-reduce for machine learning on multicore, Advances in neural information processing systems. 19 (2007) 281-287.
[4] Information on http://mahout.apache.org/.
[5] D. Talia and P. Trunfio, How distributed data mining tasks can thrive as knowledge services Communications of the ACM. 53(2010) 132-137.
[6] L. Yu, J. Zheng, W. C. Shen, B. Wu, B. Wang, L. Qian, B. R. Zhang, BC-PDM: data mining, social network analysis and text mining system based on cloud computing, Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. (2012) 1496-1499.
[7] R. Agrawal and J.C.Shafer, Parallel mining of Association Rules, IEEE Transactions on knowledge and Data engineering., 8(6): 962 – 969, December 1996.
[8] Gillick D Faria A DeNero J., MapReduce : Distributed Computing for Machine Learning, Berkeley (2006).
[9] "Top 10 algorithms in data mining" , Springer – Verlag London Limited 2007.
[10] Jongwook Woo and Yuhang Xu, "Market basket analysis algorithm with Map/Reduce of Cloud Computing ", The 2011 International Conference on Parallel and Distributed Processing techniques and Applications (PDPTA 2011), Las Vegas, July 18 – 21, 2011.

