International Journal of Computer Science and Engineering Communications,
Vol.3, Issue.4, Page.1224-1228, (2015)
www.scientistlink.com

Frequent Pattern Mining in Big Data

[1] Nagaraju.T, [2]Subhashini.P

[1] PG Scholar - Department of Computer Science, St. Peter's University, Chennai, India.
[2]Assistant Professor- Department of Computer Science, St. Peter's University, Chennai, India.
[1.]nagarajthulasimani@gmail.com, [2]subhait2k@rediffmail.com

ABSTRACT

Frequent Pattern Mining (FPM) is one of the most well-known techniques to extract knowledge from data. The combinatorial explosion of FIM methods become even more problematic when they are applied to Big Data. Fortunately, recent improvements in the field of parallel programming already provide good tools to tackle this problem. However, these tools come with their own technical challenges, e.g. balanced data distribution and inter-communication costs. In this paper, investigation applicability of FIM(Frequent Item-set Mining) techniques on the Map Reduce platform. The introduce two new methods for mining large datasets: Dist-Eclat focuses on speed while BigFIM is optimized to run on really large datasets. In our experiments show the scalability of our methods.

Mining frequent itemsets is one of the most investigated fields in data mining. It is a fundamental and crucial task. In order to improve the efficiency and processing the data parallel Map reduce algorithm is used, for mining frequent itemsets, is proposed. Firstly, the data structure binary string is employed to describe the database. Hadoop is used to process the big data parallel using Map reduce. Large files are stored in the Hadoop file system and processing the input files for finding the frequent patterns in the given input files. Here the system, which acts as a source can produce structured noise of its own, and hence the dependency on helpers may get reduced.

1. INTRODUCTION

Frequent Pattern Mining (FPM) is one of the most well-known techniques to extract knowledge from data. The combinatorial explosion of FIM methods become even more problematic when they are applied to Big Data. Fortunately, recent improvements in the field of parallel programming already provide good tools to tackle this problem. However, these tools come with their own technical challenges, e.g. balanced data distribution and inter-communication costs. In this paper, investigating the applicability of FIM techniques on the Map Reduce platform. introduced two new methods for mining large datasets: Dist-Eclat focuses on speed while BigFIM is optimized to run on really large datasets. In our experiments show the scalability of our methods.The data in Hadoop Distributed File System is scattered and needs lots of time to retrieve. MapReduce function on data sets of key & value pair is the programming paradigm of large distributed operation. The proposed work aims to minimize the data retrieval time taken by the MapReduce program in the HDFS. The major idea is to use in-memory caching in the map phase which shall give a fast and efficient way of searching frequent item set in MapReduce paradigm. For real time processing on Hadoop, a search mechanism is implemented in HDFS. The approach is used to improve its availability, performance and scalability for searching frequent item set.

Hadoop is a free, Java based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of apache project sponsored by the Apache Software Foundation. The below figure1 show the mapreduce system model which allows us to create datanodes in the cloud infrastructure. This AWS cloud is created in amazon web server this can be accessed by any where across the world. It consists of Namenode and Datanodes and together forms a cluster like infrastructure and all the nodes are interconnected. Thus the client send a query to namenode and searches the required datas with the namenode,This will perform the mapreduce operation using hadoop. All the files are stored in the hadoop distributed file system by Client. Thus the Hadoop system consists of HDFS and MapReduce. HDFS is highly fault.

Desynchronization errors. However, it is observed . that such a scheme is successful when the number of selected host signal samples is much less than the total number of host signal samples. In 3-D DWT domain is used to hide data. They use LL sub band coefficients and do not perform any adaptive selection. Therefore, they do not use error correction codes robust to erasures. Instead, they use BCH code to increase error correction capability.

System Models

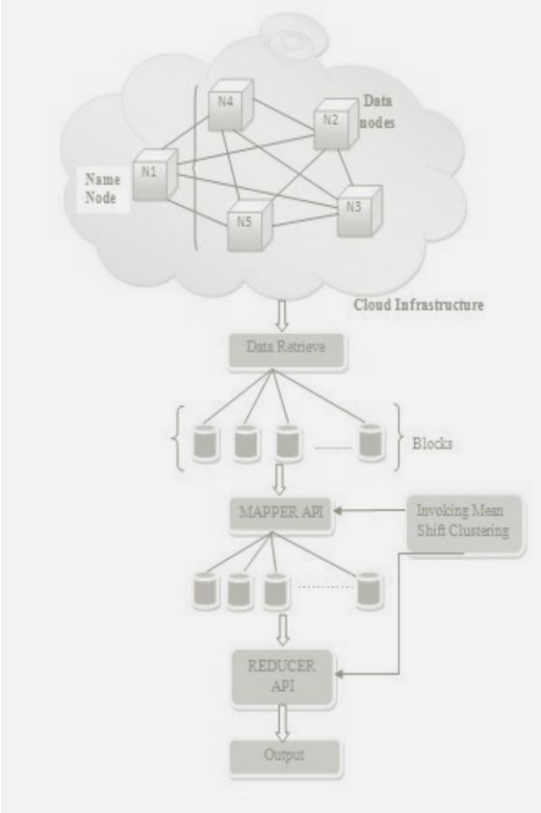

Figure 1.1 Map Reduce System Model

By means of simple rules applied to the frame markers, it introduces certain level of robustness against frame drop, repeat and inserts attacks. And it also utilizes systematic RA codes to encode message bits and frame marker bits. Each bit is associated with a block residing in a group of frames. Random interleaving is performed spatio-temporally; hence, dependency to local characteristics is reduced. Host signal coefficients used for data hiding are selected at four stages. First, frame selection is performed. Frames with sufficient number of blocks are selected. Next, only some predetermined low frequency DCT coefficients are permitted to hide data. Then the average energy of the block is expected to be greater than a predetermined threshold. In the final stage, the energy of each coefficient is compared against another threshold.

2. LITERATURE SURVEY

Hadoop is an open source cloud computing environment which supports the large data sets in a distributed computing environment, primarily in data warehouse systems and plays an important role in support of big data analytics to understand the user behaviour and their needs in web services. It stores application data and file system separately. Hadoop Distributed File System is highly fault-tolerant, which provides high throughput access to the application data and is a perfect application that has large data sets. It has the Master and Slave architecture. The cluster of Hadoop Distributed File System consists of DataNode and NameNode. NameNode which is a central server, also called as a Master server, is responsible for managing the file system namespace and client access to files and DataNode in the cluster node are responsible for files storage. Google introduced MapReduce to hold up large distributed computing, on large clusters of computers to implement huge amounts of data set.

MapReduce is a structure that processed parallel problems of big data set using a large number of computers that is collectively referred to as a cluster or a grid.

NameNode:

Keeps metadata in RAM.Each block information occupies roughly 150 bytes of memoryWithout NameNode, the file system cannot be usedPersistence of metadata: synchronous and atomic writes to NFS

Secondary NameNode:

- Merges the namespace with the edit log.
- A useful trick to recover from a failure of the NameNode is to use the NFS copy of metadata and switch the secondary to primary

DataNode:

- They store data and talk to clients.
- They report periodically to the NameNode the list of blocks they hold

JobTracker:

- Create an object for the job
- Encapsulate its tasks
- Bookkeeping with the tasks' status and progress

TaskTracker:

- TaskTrackers periodically send heartbeats to the JobTracker
- TaskTracker is alive
- Heartbeat contains also information on availability of the
- TaskTrackers to execute a task

Map Reduce:

Map Reduce is a programming model controlling a large number of nodes to handle a huge amount of data by combining node. A MapReduce application that needs to be run on the MapReduce system is called a job. The input file of a job, which resides on a distributed file system throughout the cluster, is split into even-sized block replicated for fault tolerance. A job can be divided into a series of tasks. After splitting input into block they first processed by a map task, which generate a list in the form of key-value pairs. Map outputs are split into buckets based on the key. After Mapping a reduce function is apply on the list of key generated by map function. In a cluster which runs MapReduce, there is only one Name Node also called master, which records information about the location of data chunks. There are lots of Data Nodes, also called workers, which store data in individual nodes.

There is only one Job Tracker and a series of TaskTrackers. Job Tracker is a process which manages jobs. Task Tracker is a process which manages tasks on a node. MapReduce is a programming model for data processing. The model is simple, yet not too simple to express useful programs in. Hadoop can run MapReduce programs written in various languages; in this chapter, look at the same program expressed in Java, Ruby, Python, and C++. Most important, MapReduce programs are inherently parallel, thus putting very large-scale data analysis into the hands of anyone with enough machines at her disposal. MapReduce comes into its own for large datasets.

FP Growth Algorithm:-

Apriori algorithm is found to be better for association rule mining. Still there are various difficulties faced by apriori algorithm. This algorithms suffer from the following two disadvantages

1) It as costly to handle large number of candidate sets.

2) It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching.

Keeping this in mind a new class of algorithm has recently been proposed which avoid the generation of large numbers of candidate sets. It describe one such method called the FP-Tree Growth algorithm. The algorithm involve 2 phases. In 1 phase, it constructs the FP-tree with respect to a given support. The construction of this tree requires 2 phases over the whole database. in phase 2 the algorithm does not use the transaction database any more, but it uses the FP-tree .this can be easily understood by the given example Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure highly condensed, but complete for frequent pattern mining avoid costly database scans.Develop an efficient, FP-tree-based frequent pattern mining method.

A divide-and-conquer methodology: decompose mining tasks into smaller ones. Avoid candidate generation: sub-database test only! Consider the same previous example of a database, D refer fig 1(a), consisting of 9 transactions. Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %).The first scan of database is same as Apriori, which derives the set of 1-itemsets & their support counts The set of frequent items is sorted in the order of descending support count.The resulting set is denoted as L = {I2:7, I1:6, I3:6, I4:2, I5:2}.First, create the root of the tree, labeled with "null".Scan the database D a second time. (First time the scanned it to create 1-itemset and then L).The items in each transaction are processed in L order (i.e. sorted order).A branch is created for each transaction with items having their support count separated by colon.Whenever the same node is encountered in another transaction, it just increment the support count of the common node or Prefix.To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links.Now, The problem of mining frequent patterns in database is transformed to that of mining the FP-Tree.

Steps for FP Tree processing:

1. Start from each frequent length-1 pattern (as an initial suffix pattern).
2. Construct its conditional pattern base which consists of the set of prefix paths in the FP-Tree co-occurring with suffix pattern.
3. Then, Construct its conditional FP-Tree & perform mining on such a tree.
4. The pattern growth is achieved by concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-Tree.
5. The union of all frequent patterns (generated by step 4) gives the required frequent itemset. Now, Following the above mentioned steps:

• Let's start from I5. The I5 is involved in 2 branches namely {I2 I1 I5: 1} and {I2 I1 I3 I5: 1}.

• Therefore considering I5 as suffix, its 2 corresponding prefix paths would be {I2 I1: 1} and {I2 I1 I3: 1}, which forms its conditional pattern base.

Out of these, Only I1 & I2 is selected in the conditional FP-Tree because I3 is not satisfying the minimum support count.

For I1 , support count in conditional pattern base = 1 + 1 = 2

For I2 , support count in conditional pattern base = 1 + 1 = 2

For I3, support count in conditional pattern base = 1

Thus support count for I3 is less than required min_sup which is 2 here

5. RESULTS COMPARISON

The existing system uses web server in the map phase using the jetty web server which shall give a moderate way of searching data in MapReduce paradigm. The load balancer is used to balance the workload across servers to improve the availability, scalability and performance. The existing work aims to minimize the data retrieval time taken by the MapReduce program in the cloud.

Limitations

- Installations of web servers across all nodes are difficult and complex to manage.
- The performance of the system is not enough and does not meet to the level if the data grows rapidly.
- The response time is minimal when it takes data through web server.

The data in Hadoop Distributed File System is scattered which makes searching and data retrieval is one of the most challenging tasks in HDFS. So to overcome this challenge the proposed system provides a searchable mechanism in HDFS. The MapReduce mechanism of Hadoop is implemented with in-memory caching so that the data retrieval will be faster and gives high performance.

Implemented FPGrowth algorithm for data retrieval from Hadoop HDFS. The task of map function, in Phase I FPGrowth, is employing Apriori algorithm on the whole split as an input and finding a set of partial frequent itemsets. Therefore, the Mapper's output is a list of intermediate key/value pairs, defined as <itemset, supportCount>, where itemset is an element of partial frequent itemsets and the supportCount is its partial count.

When all the map tasks are finished, the reduce task is started. The reduce task adds up the supportCount of the same partial frequent itemsets and gets the partial support count of them. Reducer also computes for each partial frequent itemset.

Advantages

This system is going to implement in-Memory Caching system which will gradually increase the performance of 10 x times higher and decrease the time taken to fetch the frequent Item sets data in a multi-node Hadoop cluster.

8 CONCLUSIONS

Frequent pattern mining is generated from the big data input file under hadoop environment based on the minimum support count. Frequent pattern are generated with efficient performance by using the MapReduce and FP-Growth algorithm. Generated pattern counts are verified for the performance and its efficiency successfully.In this experiment the performance of finding the frequent item set is executed in lesser time when compared with other existing algorithms. in experimenting the processing of big data input file using FP-Growth algorithm and generated the output files with Frequent Item sets based on the minimum support count. With support of map reduce in processing of big data using Hadoop system the analyses the difference in performance while processing the big data. This system works fine and give better performance even in high data load.

6. REFERENCES

[1] Ming-Yen Lin, Pei-Yu Lee, and Sue-Chen Hsueh. 2012. Aprioribased frequent itemset mining algorithms on MapReduce.In Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication (ICUIMC '12). ACM, New York, NY, USA, , Article 76 , 8 pages.

[2] Farzanyar, Z., & Cercone, N. (2013, August). Efficient Mining of Frequent itemsets in Social Network Data based on MapReduce Framework. In Proceedings of the 2013 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013) (pp. 1183-1188). IEEE Computer Society.

[3] Li L. & Zhang M. (2011). The Strategy of Mining Association Rule Based on Cloud Computing. Proceeding of the 2011 International Conference on Business Computing and Global Informatization (BCGIN '11). Washington, DC, USA, IEEE: 475- 478.

[4] Li N., Zeng L., He Q. & Shi Z. (2012). Parallel Implementation of Apriori Algorithm Based on MapReduce. Proc. of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD '12). Kyoto, IEEE: 236 – 241.

[5] 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.

[6] Othman Yahya, Osman Hegazy, Ehab Ezat (2012).An Efficient Implementation of Apriori Algorithm Based on Hadoop-Mapreduce Model, Proc. of the International Journal of Reviews in Computing 31st December 2012. Vol. 12:59-67.

[7] H. Li, Y. Wang, D. Zhang, M. Zhang, and E.Y. Chang: PFP: Parallel FP-Growth for Query Recommendation. In: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 107-114, 2008.

[8] L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng. Balanced parallel FP-Growth with MapReduce. In Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on, pages 243–246, nov. 2010. 598

[9] E. Ozkural, B. Ucar, and C. Aykanat. Parallel frequent item set mining with selective item replication. Parallel and Distributed Systems, IEEE Transactions on, 22(10):1632–1640, oct. 2011.

[10] Hadoop, http://hadoop.apache.org/