CHAPTER 3

ANALYSIS AND IMPLEMENTATION OF EXISTING APPROACHES OF APRIORI AND GRID MINING

3.1 INTRODUCTION

With the detailed literature survey conducted in chapter 2, the research is motivated towards the direction of frequent itemset mining. Apriori algorithm seems to be better suitable for association rule mining which makes it interesting to mine interesting patterns from data. Research work proceeds in such a way of analysing which algorithm can be chosen to suit the frequent itemset mining because numerous algorithms exist for association rule mining. Initially apriori algorithm is being tested in an eclipse environment to find out the working process. Then distributed data mining concepts are analysed to check the ability of apriori in distributed environment. Grid and cloud environments are being experimented and then cloud is chosen because of its less complexity. This chapter starts with exact discussion of what is frequent item set mining and then proceed with the experiments done for the analysis purpose.

This chapter explains frequent itemset mining and apriori algorithm at first instance. Then the benefits of moving onto distributed mining is discussed. Distributed environment is equipped with plenty of ways to work with, among them the first trial has been done with a grid environment and later drawback or inconvenience of grid environment is narrated. Since cloud technology is a latest emerging field for distributed environment and easy to implement, possibilities of
implementing this research work with cloud instead of grid is being discussed. The results of executing apriori and few other variations of apriori in eclipse environment are investigated. Then results of apriori algorithm executed in weka and distributed apriori executed in Globus grid toolkit are compared and discussed. Then this research proposes and outlines a model to proceed with a better distributed environment.

3.2 RESEARCH MOTIVE BEHIND FREQUENT ITEMSET MINING

Frequent itemsets (Jianyong Wang et al. 2007) and association rules, discovery of both are considered as synonyms. However association rule discovery is a difficult characterization which is performed on data and relies on extraction of frequent itemsets from the dataset. The “market-basket” model of data explains the concept of frequent itemsets since it is showing a relationship type of many to many among items and baskets and it is showing also conventions about the outline of data. All algorithms which work for the frequent itemset solution, are not up to the standards.

Market-basket model of data can be described as follows. It consists of items and transactions that is, baskets. Every basket consists of small group of items from the entire count of items. Impulsively, a set of items that appears in many baskets are called as “frequent”. A support threshold value s is assumed. The support value of set of items I, is calculated using number count of baskets which are all having I. If that value is more than s then I is said to be frequent. Analysis of true market baskets is said to be the original application of market-basket model. The numerous products, sold by the store are mentioned as “items” and every transaction is a basket. Everyday lakhs of transactions are taking place.
in retail or whole-sale markets. The research of frequent itemset analysis is not only applied for market baskets but also applied for other categories of data.

Collection of if–then rules are association rules. It takes the form \( A \rightarrow b \), where \( A \) is for set of items and \( b \) is for an item. The inference of this association rule is that if entire items in \( A \) are present in some basket, then \( b \) may also appear in that basket. The view, “likely” is enacted by the rule \( A \rightarrow b \) and the value of confidence is computed with respect to the value of support for \( A \cup \{b\} \) to the support for \( A \). Hence confidence of the rule is defined as the fraction of the baskets with all of \( A \) that also contain \( b \). That is an association rule \( A \rightarrow b \) is defined as the variance between its confidence and the fraction of baskets which contain \( b \). Zero interest rules are having no influence of \( A \) on \( b \), that means the fraction of baskets including \( A \) that contain \( b \) would be accurately the same as the fraction of all baskets that contain \( b \). But these rules are with high positive interest, if \( A \) is present in a basket which causes \( b \) to be present, or highly negative interest that means the presence of \( A \) is not having the presence of \( b \). The discussion proceeds such that, if association rule \( A \rightarrow b \) is applied to a group of baskets, then support value of \( A \) must be logically more.

The definitions are given as follows. A value to indicate the frequency of item-set \( X \) appears in the dataset is termed as Support. The support value of \( X \) with respect to \( T \) is described as the percentage of transactions in the dataset that contains itemset \( X \). A value to indicate frequency of the rule to be true is termed as Confidence. The confidence value of a rule \( X \rightarrow Y \), is said to be the ratio of transactions that consists of \( X \) which also contains \( Y \).
Suppose all itemsets are identified which meet a support threshold, all association rules having high support and high confidence are found out. It can be guessed that if there are not many frequent itemsets then there will not be many candidates for valid support and valid confidence association rules. The motive is to act upon each rule. It will not be much useful if the store manager is delivered with many association rules. The support threshold can be adjusted to get only limited association rules which can be acted upon.

3.3 IMPLEMENTATION OF APRIORI ALGORITHM

An important algorithm proposed by Agrawal R and Srikant R (1994) is Apriori to mine frequent itemsets and to find out boolean association rules. This algorithm applies prior knowledge based on frequent itemset properties. It uses an iterative approach, called as level-wise search, which is being experimented such that k-itemsets are used to explore (k+1) itemsets. The dataset is scanned to find out frequent 1 itemsets which accumulates count of each item, then items satisfying minimum support are collected. The result set is referred as L1, which is then applied to find L2 that is, the collection of frequent 2- itemsets. Then it is used to find L3. The iteration is repeated until no more frequent k- itemsets are found. One full scan is required to find out each Lk from the dataset. Efficiency of the level-wise generation of frequent itemsets is enhanced using Apriori property which is termed as all nonempty subsets of a frequent itemset must also be frequent, is used to reduce the search space. The two set process is followed which consists of join and prune actions.

**The join step:** To find Lk, which is a collection of candidate k- itemsets, is generated by joining L_{k-1} with itself. This set of candidates is denoted
as \( C_k \). Let \( l_1 \) and \( l_2 \) be itemset in \( l_{k-1} \). After a series of process the resulting itemset formed by joining \( l_1 \) and \( l_2 \) is \( l_1[1], l_1[2], \ldots l_1[k-2], l_1[k-1], l_2[k-1] \).

**The prune step:** \( C_k \) is a superset of \( L_k \), all frequent \( k \)-itemsets are included in \( C_k \). Dataset is scanned to determine each candidate’s count in \( C_k \) which helps to determine \( L_k \) (candidates having a count no less than the minimum support count).

To decrease the size of candidate set \( C_k \), the **Apriori property** is applied as follows. Any \((k-1)\) itemsets that is not frequent cannot be a subset of a frequent \( k \)-itemset. Hence, if any \((k-1)\) subset of a candidate \( k \)-itemset is not in \( L_{k-1} \). The candidate those are all not frequent will be removed from \( C_k \).

Then the research is aimed to experiment the apriori algorithm at the initial stage, since the algorithm is considered to be the source of the work which proceeds to find out an effective algorithm to investigate the large set of data items. Figure 3.1 shows that apriori algorithm which has been executed on an eclipse platform. Retail itemset has been used as the dataset with only 5 items. This has been done on an experimental purpose, because the research work is aimed to proceed with a discussion of implementation of an appropriate apriori in a java environment. Eclipse IDE is opted for this purpose. Apriori algorithm is coded in java and eclipse is identified as the best IDE for implementing the algorithm. Eclipse provides options for executing any algorithm as a stand-alone mode as well as in distributed mode. Initially it has been run in stand-alone mode. The figure 3.1 clearly depicts the way of frequent 1 itemsets, 2 itemsets, and so on to be separated. This is given as a sample of how large itemsets are split up and used for frequent itemset mining.
3.4 VARIATIONS OF APRIORI ALGORITHMS

The explanation on various forms of apriori algorithms are as follows.

3.4.1 Apriori-TID Algorithm

AprioritID algorithm is an extension of Apriori, which is not depending on raw dataset, instead it represents every transaction internally by the candidates present within it. It uses a function to find out the candidate itemsets, the dataset is not used for support count, instead set of candidate itemsets is used for k>1. If a transaction is not having single candidate k-itemset, the set of candidate itemsets will not access that transaction. This will gradually reduce the transaction count in the set containing the candidate itemsets as compared to the dataset. Value of k augments and each entry will be smaller compared to the
respective transactions. For initial passes Apriori works better but for final passes AprioriTid is working better. These two algorithms, Apriori and AprioriTID approaches are combined in AprioriHybrid algorithm.

### 3.4.2 FP-Growth Algorithm

FP-Growth (Han J et al. 2000) builds large frequent itemsets but don’t generate and test candidates. Here dataset projections are exploited by frequent itemsets, and frequent items in the projections grow patterns. FP-Growth algorithm draws FP-Tree, a data structure, with frequent items, to facilitate projections. This data structure is very compact so that it can be stored in main memory. FP-tree is built using two database scans so that mining can be done on FP-tree. In this method dataset need not to be scanned every time. Frequent pattern growth is a way for mining itemsets which are frequent, without candidate generation (Sheila A Abaya 2012). FP-tree data structure is constructed to compress real dataset. Figure 3.2 shows runtime of Apriori and FP-Growth algorithms executed on an eclipse platform on a trial basis.

![Figure 3.2 Apriori and FP-Growth](image-url)
FP-Growth algorithm is proved to be best option than apriori algorithm in many ways which are all listed out below. So FP-growth algorithm is also executed in the same eclipse environment and tested.

### 3.4.3 HDO Algorithm

HDO Apriori (High Demension Oriented) algorithm is developed by improving traditional algorithm. It adopts a novel method for reducing generation of redundant sub-itemsets in the pruning process and having efficient mining procedure, though the dimension of data is high. Experiments is done with KDD Cup 1999 datasets. The experiments done here in this research validate the work done for HDO algorithm.

### 3.4.4 WDPA Algorithm

The aim to find out solution for distributed data problem, leads to the invention of Weighted Distributed Parallel Apriori algorithm (WDPA) (Kun-Ming Yu. 2008). It stores metadata in TID forms. Only a single scan of the dataset is required. Load-balancing is better because of TID counts as well as reduction of idle time for processors. Experimental results show that, WDPA outperforms other algorithms.

### 3.4.5 HPA Algorithm

Parallel algorithms are performing with optimum cost. Hash function is used for execution of Hash Partitioned Algorithm which partitions candidate
itemsets among processors. Eui-Hong et al. (2000) says that HPA is good for parallel and distributed environment since it uses whole memory space of all the processors. Dynamic data allocation is also followed here in this application. Sujni Paul and Saravanan V (2008) says that HPA can be applied in parallel and distributed environments as a non-trivial task. They introduced method for satisfying challenges related with parallel and distributed data mining. The challenges are minimizing I/O, increasing processing speed and communication cost.

Grid environments offers data distribution techniques which significantly reduce the total execution time of a program effectively for data mining applications. This paper by Eui-Hong et al. (2000) describes Heuristic Data Distribution Scheme (HDDS), a linear programming formulation for solving data distribution problem on grids. It is a heuristic method and experiments are conducted using Globus toolkit. Experimental results show that data could be executed more efficiently than conventional schemes in this technique.

3.4.6 Experiments and Investigations

The experiments are being done on a PC with configuration of 2.2GHz Intel(R) Core(TM) 2 Duo PC and 2GB memory installed with Windows 7 OS. Application is coded using Java in NetBeans IDE. The first experiment is carried out in a purpose of checking the competence of original apriori algorithm, HDO algorithm and IAA algorithm. A synthesized dataset T10I5D10KN50 is generated by IBM's Quest Synthetic Data Generator. Figure 3.3 shows the result and it is understood that the method of count-based prune operation in Improved Apriori Algorithm show difference than other algorithms. The purpose of second
experiment is to compare entire runtime of traditional Apriori-Tid, Modified WDPA. Here only one process is applied and (k-1) dimensional frequent itemsets are considered, k-dimensional candidate itemsets are requested. The dataset T10I4D10KN100 is used for this experiment.

Though it is not used, creating structure <itemsets, TIDs> is taking more time in WPDA and IAA (Lei Ji et al. 2006), specifically in the final round. If the minimum support is more than 4%, the association rule mining completes in two or three rounds, and final round produces more frequent itemsets. The result in Figure 3.3 depict Apriori-Tid is speeder than both algorithms. The iterations are increased if the minimum support is decreased. WPDA and IAA algorithms save the running time of execution. On the whole, figure 3.3 shows the time comparison of various apriori algorithms such as Apriori-Tid, WPDA and Improved Apriori Algorithm (IAA). This experiment is done in Google App Engine with eclipse plugin.

![Figure 3.3 Variations of Apriori](image-url)
IAA performs well compared to other algorithms using the technique of count-based prune operation and candidate generation. IAA records a fixed length for (k-1)-dimensional frequent itemsets, hence WDPA proposes weighted load-balance strategy which is adopted when parallel or distributed data mining is using IAA. The output brings out certain facts which are all described as follows.

3.4.7 Comparison of Algorithms

- (k-1) dimensional itemsets are utilized by HDO, which are infrequent to minimize the operation time.
- The method proved to be better is one that is, Dynamic itemset counting.
- Apriori-Growth combines Apriori and FP-growth algorithms.
- Parallel/Distributed ARM – HPA, WDPA, Apriori-T, (Chao-Tung Yang, 2008) HDDS – Timing and memory usage experiments proves the drawbacks of each algorithm.

3.5 RESEARCH FOCUS ON DISTRIBUTED APRIORI ALGORITHMS

Distributed algorithm mines data in a distributed environment (Lamine M Aouad et al. 2007) and it should not exchange raw data between sharing sites. Hence Distributed Association Rule Mining (DARM), has become a new innovative research area supported by numerous distributed algorithms. These algorithms try to optimize the transmission of data between various sites to find out knowledge patterns. The setup of the knowledge grid helps for a better way of storing, acquiring, exchanging of information and converting the information into useful knowledge. Distributed data mining is analysing large data sets conserved over sites distributed geographically. Set of software components are available in Globus toolkit for building distributed systems. (Chao-Tung Yang et al. 2008).
3.5.1 Apriori Grid Service

The Apriori Grid Service which is introduced by Ansari E et al. (2007), observes OGSA (Open Grid Service Architecture) rules, conditions, interfaces and working behaviours. A standard interface and a discovery service to register details about Grid service instances in registry services are comprised in service data access. The method FindServiceData is called by client application to extract service information from instances of individual Apriori Grid Service. The service data access also defines an interface and semantics for vital service creation of the Apriori Grid Service which is located at the Service Data Element level. Sakshi Aggarwal et al. (2008) says about a platform to apply and deploy knowledge which is distributing management services and software to various places is termed as Grid technology. The paper also talks about MPICH-G2 which is a grid enabled message passing technique for communicating between different grid nodes. Service Oriented Architecture (SOA) is a programming model to construct adaptable, integrated, and interoperable software applications. OGCA reflects SOA along with Grid context. OGSA provides a basic interfaces to develop interoperable Grid systems and applications. It also approves Web Services to define methods, functions and discovery techniques.

This research work implements a simulated grid environment with Globus toolkit in a LAN setup and tries to implement apriori.

3.5.2 Assessment of Classical Apriori & Grid Based Apriori Algorithms

The research work carried out an experiment with classical apriori and grid based apriori algorithm using Globus toolkit. A sequence of experimental runs are done. Sequential centralized implementation of the Apriori algorithm is
carried out using weka library and a decentralized implementation of Apriori Grid Service using Globus toolkit are experimented. The dataset is in the format of $TxxIyyDzzzK$, where $xx$ refers the average number of items present per transaction, $yy$ refers the average support of each item in the dataset and $zzzK$ refers the total number of transactions in $K$ (1000 s). The tests are done for database sizes of 3000, 7000, 10000 and 50000 transactions respectively and the resulting rules had 31%, 26% and 21% support factors. Figure 3.4 describes Grid Apriori service and classical apriori algorithms comparison values which are all obtained from experiments.

![Figure 3.4 Comparison of Traditional Apriori and Grid Apriori service](image)

The effectiveness of the Grid Apriori Service algorithm compared with the traditional Apriori is explicitly exposed. Now the Grid version Apriori performance is proved to be greater than traditional Apriori by 24% for 31% support, and by 26% for 26% support and to 32% for 21% support. These experiments compared the performance for a count of 10000 transactions. The experiment can be extended to local intranet virtual organizations. This experiment is further extendable to virtual organization on the local intranet. The Grid Apriori Service test can be performed with different types of parameters.
The parameters are algorithm support, remote databases, and confidence factors. As a result, rules can be generated in real time. The speed of these applications are improved due to grid environment.

3.6 RESEARCH FOCUS ON WEIGHTED ASSOCIATION RULE MINING

The characteristics of general transaction database (Ramkumar G D et al. 1998) is computed as a numerical weight value. The traditional apriori algorithm mines association rules using a binary mapped dataset which shows occurrence of data or an item in one stretch of action. But WARM allows gathering and verifying good number of information related to the features of data, which helps in the formation of recurrent and weightage rules. In the new approach of weighted mining a subset is given high significance. High utility valued association rules are provided in this method by considering the weight factor, utility and diminution.

By controlling number of frequent item sets just by minimum supporting threshold and pruning techniques, there is possibility for meaningless frequent item sets. But here the procedure divides the items into many types and set up weighted value for each type. The new procedure computes weighted support and confidence, and do pruning and selection according to the minimum weighted support and confidence threshold. Hence the result will be new frequent item sets and association rules.

Weighted association-rule mining considers significance of transactions or items. Item weight or transaction weight are used to find out
different kinds of interesting models from a dataset. Standard support value is modified with WSupport in Weighted Association Rule Mining. These rules are in different forms like fuzzy, utility association rules and so on. Item weight, $w$, can have a value from 0 to 1. It gradually increases with importance of items from 0 to 1. For example if itemset $X$ is 0.95, and it tells us the importance of itemset in transaction $D$. A weight value of 0.1 indicates a less important set. In this paper by Ramkumar G D et al. (1998) it is said WSupport of binary weighted rule $X \Rightarrow Y$ is an adjusting ratio compared to support, or mathematically, $wsupport (X, Y) = ( w_j \cup y ) support (X, Y)$ where the weights of the items $\{i_1, i_2, ..., i_m\}$ are $\{w_1, w_2, ..., w_m\}$ respectively. If the weighted support of the itemset $X$ is larger than or equal to the weighted support threshold, that is $wsupport(X) \geq wminsupt$, then $X$ is termed as large weighted itemset.

3.7 RESEARCH ANALYSIS ON MAPREDUCE MODEL

A new programming model and software framework which is termed as MapReduce model is designed by Google for the processing of large sets of data. It is used to process records from user in a parallel way using Mappers and Reducers, finally it merged the outputs of these mappers. Two phases are used here and MapReduce becomes familiar in parallel as well as distributed programming model, its processing and generates huge data sets in distributed way.

As said in paper by Xin Yue Yang et al. 2010, MapReduce is a linear scalable programming model. The functions are Map and Reduce and they define a mapping from one set of key-value pairs to another and the respective function is obvious to the size of data or cluster. If size of the input data increases, a job will run as slowly. But if size of cluster is increased, a job will run with at most speed. Apache Hadoop provides a suitable infrastructure for MapReduce.
implementation. Apache Software Foundation provides Hadoop as an open source platform consists of MapReduce and its Hadoop Distributed File System HDFS. Additional services are Core and HBase. Hadoop Core is a set of components and interfaces for file systems and general I/O of distributed environment. A distributed data processing model and execution environment that runs on large clusters of commodity machines, is termed as Hadoop and Hadoop HDFS is a distributed file system that runs on large clusters of machines.

It is difficult to set up an enormous size cluster comprises of hundreds or thousands of nodes for measuring scalability of an algorithm. Environment setup is crucial for best performance. Amazon Elastic Compute Cloud EC2 can be used by Hadoop users where they can implement their applications on demand and pay for the time they use and it also can be combined with Sun Grid Engine. Simulators help users to compute job time and costs, then to submit to EC2. Hundreds of nodes can be constituted by Hadoop to process and compute Big Data. Global community of providers such as Yahoo, Facebook, Cloudera, and Twitters are using Hadoop. The paper by Xin Yue Yang et al. (2010), also talks about subprojects of Hadoop which consists of Hadoop Common, Avro, HBase and so on.

This research work proposes to choose MapReduce framework for distributed environment because of its familiarity in the recent trends.

3.8 RESEARCH ANALYSIS ON SECTOR SPHERE MODEL

One infrastructure that offers various resources and/or services in World Wide Web is sector-sphere (Robert Grossman et al. 2004) which
resembles MapReduce. In this model, Storage services are offered by a storage cloud, and compute services are offered by compute cloud. Compute clusters connected with networks of high performance are used by Sector and Sphere for analysis purpose. Sector is providing persistent storage to large datasets which are achieved as distributed indexes for a long term period. Sector manages various segments scattered throughout the distributed storage. Data is replicated by sector to ensure its durability, decline the latency of retrieval and opportunities for parallelism. Sphere is for executing functions of concurrent users by a pattern which is managing data by sector. Same user function is applied to every segment of dataset providing chance for automatic parallelism. Data need not be moved and being processed at its own place by Sector/Sphere model. The concept can be very well explained as sector handles data while sphere processes data with functions and Sector/Sphere is made suitable for high level wide area networks.

This research work made efforts to implement sector sphere model for the apriori algorithm distributed execution, but fails because of certain vulnerabilities caused due to overloading of the machine.

3.9 GOOGLE FILE SYSTEM

Subsequently this research considers about distributed proprietary system, termed as Google File System (GFS or GoogleFS) created for Google’s own applications (Ameya Daphalapurkar et al. 2014). Effective and reliable access to data is provided using product hardware. New version of GFS is Colossus 2010. A GFS cluster consists of multiple nodes and are divided into two types One Master node and many Chunk servers. File made up of uniform sized chunks and the servers store these chunks. A unique label is assigned to each chunk while creating as well as logical mappings of files are kept in track.
Replication is also done to avoid loss of data. This research also experiments apriori algorithms in Google app engine and finds out that algorithms can be well executed in a distributed environment.

3.10 CHOICE OF HADOOP DISTRIBUTED FILE SYSTEM

As discussed in section 3.7, the research on distributed data mining gives better outcome with Hadoop which is a Map/Reduce platform by Apache. It consists of hundreds of nodes which processes and computes Big Data. Yahoo, Facebook, Cloudera, and Twitters and so on are global community contributors. The sub projects of Hadoop are HDFS, MapReduce, Avro, Hadoop Common, Chukwa, HBase, Hive, Mahout, Pig, and ZooKeeper.

3.11 RESULTS INTERPRETATION

Initially, the experiments are conducted in weka. Then Apriori and FP Growth algorithms are implemented for specified datasets in eclipse platform. They conclude that Apriori performs less, compared to FP-Growth. FP-Growth shows better runtime which is less compared to Apriori. Second, the experiments conducted in Google App engine to compare three more various algorithms of apriori and it shows that Improved Apriori Algorithm gives a better result. Subsequently, the research is focussed on distributed environment. Third experiments are conducted with Apriori and grid based apriori using Globus toolkit. The results prove that distributed grid environment performs well in terms of time for a specified transaction.
3.12 CONCLUSION

This chapter mainly compares the execution results of various apriori algorithms on a sequential weka platform and Google app engine. As well as it compares the performance of apriori in a grid based environment and standalone environment. This chapter concludes that it is better to proceed distributed environment with cloud compared to grid services. Then compared to the complexities of grid environment, cloud hadoop environment is better because of the following notable points. Hadoop is created as an open source framework for the usage of different clients having different needs. GFS is implemented especially for meeting the ever growing demands of Google’s data processing. GFS chunks are divided into 64MB blocks, whereas HDFS is divided into 128MB blocks. Client based applications are distributed file systems where only authorized clients can access files stored in the central servers. The naming conventions and mapping schemes are managed by distributed file systems. Hadoop Distributed File System (HDFS) which is an open source software was developed by Yahoo and designed to store huge amounts of datasets for streaming data on client applications. MapReduce framework provided by hadoop helps the users with a programming model for processing data. Hence this research work proceeds to implement apriori algorithms in a distributed Hadoop environment, and to derive a novel hybrid algorithm.