Journal of Theoretical and Applied Information Technology
31st March 2016. Vol.85. No.3

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

PARALLEL IMPLEMENTATION OF APRIORI ALGORITHMS ON THE HADOOP-MAPREDUCE PLATFORM- AN EVALUATION OF LITERATURE

A.L.SAYETH SAABITH[1], ELANKOVAN SUNDARARAJAN[2], AND AZURALIZA ABU BAKAR[3]
[1,2]Centre for Software Technology and Management, [3]Center for Artificial Intelligence and Technology, Faculty of Information.
Science and Technology, Universiti Kebangsaan Malaysia, UKM Bangi, 43600, Selangor-DE, Malaysia.

A B S T R A C T

Data mining is the extraction of useful, prognostic, interesting, and unknown information from massive transaction databases and other repositories. Data mining tools predict potential trends and actions, allowing various fields to make proactive, knowledge-driven decisions. Recently, with the rapid growth of information technology, the amount of data has exponentially increased in various fields. Big data mostly comes from people's day-to-day activities and Internet-based companies. Mining frequent itemsets and association rule mining (ARM) are well-analysed techniques for revealing attractive correlations among variables in huge datasets. The Apriori algorithm is one of the most broadly used algorithms in ARM, and it collects the itemsets that frequently occur in order to discover association rules in massive datasets. The original Apriori algorithm is for sequential (single node or computer) environments. This Apriori algorithm has many drawbacks for processing huge datasets, such as that a single machine's memory, CPU and storage capacity are insufficient. Parallel and distributed computing is the better solution to overcome the above problems. Many researchers have parallelized the Apriori algorithm. This study performs a survey on several well-enhanced and revised techniques for the parallel Apriori algorithm in the Hadoop-MapReduce environment. The Hadoop-MapReduce framework is a programming model that efficiently and effectively processes enormous databases in parallel. It can handle large clusters of commodity hardware in a reliable and fault-tolerant manner. This survey will provide an overall view of the parallel Apriori algorithm implementation in the Hadoop-MapReduce environment and briefly discuss the challenges and open issues of big data in the cloud and Hadoop-MapReduce. Moreover, this survey will not only give overall existing improved Apriori algorithm methods on Hadoop-MapReduce but also provide future research direction for upcoming researchers.

Keywords: *Data mining big data ARM Hadoop-MapReduce Cloud Apriori*

1 INTRODUCTION

Data mining is the process of extracting useful, potential, novel, understandable, and concealed information from databases that are huge, noisy, and ambiguous[40,97]. Data mining plays a vital role in various applications in the modern world, such as market analysis, credit assessment, fraud detection, medical and pharma discovery, fault diagnosis in production systems, insurance and healthcare, banking and finance, hazard forecasting, customer relationship management (CRM), and exploration of science [6,47,69,71,87,95]. Many view data mining as synonymous to Knowledge Discovery from Data (KDD), while others consider data mining as an essential stage in the KDD process [25,47,128]. The outline of the KDD process is shown in Figure 1.

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

The first step is to define a problem from a particular domain that contains appropriate previous knowledge and particular application goals. The second process is choosing an appropriate dataset, which consists of a dataset or concentrates on a subset of variables or data samples on which discovery is to be accomplished. Pre-processing is the third step of the KDD process, which consists of data collecting, data cleaning, and data selection. It is the key step in the KDD process, and it removes noise, outliers, and redundant or irrelevant information, handles missing data fields, and determines DBMS issues, such as types of data, schema, and the managing of missing and unknown values.

The sixth step is choosing the proper data mining algorithm(s), which includes selecting technique(s) to be used to find the patterns of the data, such as deciding which models may be proper and matching a particular data mining technique with the KDD process. The seventh key step is data mining, which includes discovery of the interesting patterns in the particular assigned dataset, including classification rules, decision trees, regression, clustering, sequence modelling, dependency, and line analysis. The eighth step is interpretation, which consists of data mining techniques and finding out whether a good clustering or classifying approach must interpret the result of such an approach. If a result cannot be explained properly, it is useless for further

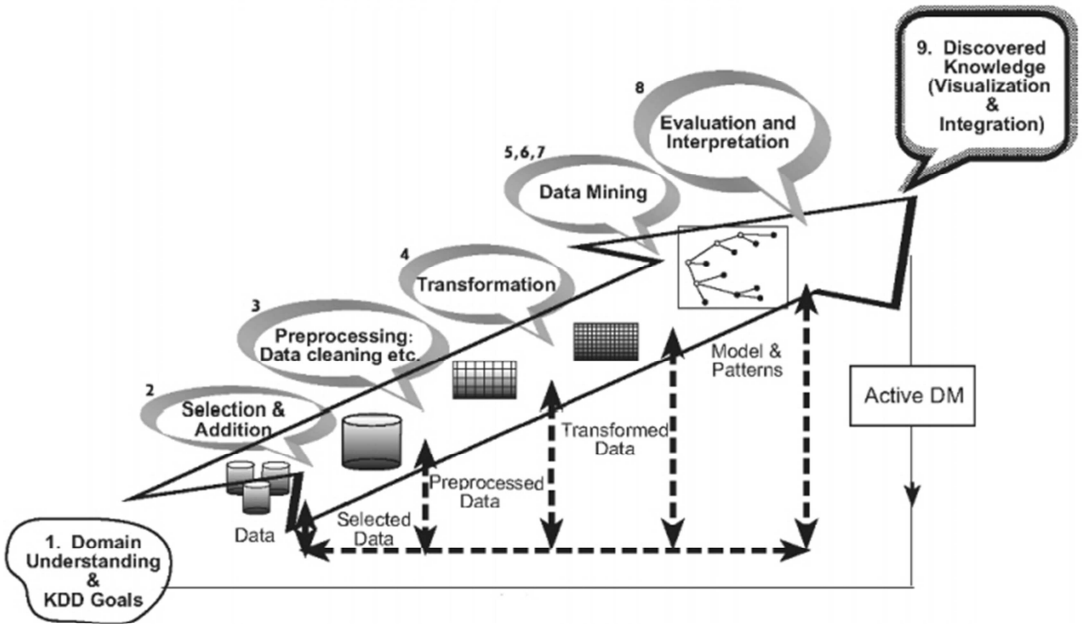

Figure. 1. Knowledge discovery of KDD process [32,63]

The fourth step is data transformation, which may refer to data reduction and projection; this process helps to discover the most valuable features of the data that is depending on the task and applies dimensionality reduction, such as reducing the number of attributes, attribute values, and tuples, or transformation methods, such as normalization, aggregation, generalization, and attribute construction, to reduce the effective number of variables under consideration or to find invariant representations for the data. The fifth step is one of the most important processes for selecting the appropriate function of data mining; it constructs a suitable model derived by the particular data mining algorithms (e.g. association rule mining, classification, summarization, clustering, and regression).

application. The last step is utilizing the discovered knowledge, i.e. using a newly discovered set of knowledge for future analysis and the prediction of new models [128].

Business intelligence has become an integral part of many successful organisations. Analysing data and making decisions based upon the analysis is very important for an organisation's growth. Data mining techniques help analyse the substantial data available to assist in decision-making. Association Rule Mining (ARM) or Frequent Itemset Mining (FIM) is one of the key areas of the data-mining paradigm. It is intended to extract interesting relationships, patterns, and associations among sets of items in the transaction database or other data repositories [12,20,43,128]. The most typical application of ARM is in market basket analysis, which analyses

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

the purchasing behaviour of customers by finding items that are frequently purchased together. In addition to the many business application, it is also applicable to telecommunication networks, web log mining, market and risk management, inventory control, bio-informatics, medical diagnosis and text mining [34,47].

The Apriori algorithm is one of the best classical algorithms for discovering frequent itemsets from a transactional database, but it has some drawbacks, such as that it scans the dataset many times to generate frequent itemsets and that it generates many candidate itemsets. When data mining mainly deals with large volumes of data, both memory usage and computational cost can be very high; also, a single processor's memory and central processing unit resources are restricted, which impacts the inefficient performance of the algorithm [5,74]. One way to improve the performance and efficiency of the Apriori algorithm is parallelizing and distributing the process of generating frequent itemsets and association rules. These versions of parallel and distributed Apriori algorithms improve the mining performance but also have some overheads, such as workload balancing, partitioning of input data, reduction of the communication costs and aggregation of information at local nodes to form the global information [5,60,115,124,125].

The problems with most distributed framework are the overheads of managing the distributed system and the lack of a high-level parallel programming language. Working with a large number of computing nodes in a cluster or grid, there is always the potential of node failures, which cause multiple re-executions of tasks. All of these pitfalls can be overcome by the Hadoop-MapReduce framework introduced by Google [26,27,44,99]. The Hadoop-MapReduce model is a Java-based programming model for readily and efficiently developing applications that process massive datasets in parallel on large clusters of commodity hardware in a trustworthy failure-resilient manner [57,91,110].

In this study, we give a detailed review of several improved Apriori algorithms in the Hadoop-MapReduce environment and the challenges and open issues of big data in the cloud and for Hadoop-MapReduce. The rest of this paper is organized as follows. Section 2 explains big data and cloud computing. Section 3 elaborates data mining techniques, the basic concept of ARM, and the Apriori algorithm. Section 4 provides an overview of the parallel discovery of the Apriori algorithm and related challenges. Section 5 describes the concepts of Hadoop, MapReduce, and HDFS and how the Apriori algorithm is implemented for the Hadoop-MapReduce model with an example. Section 6 presents analysis of several improved Apriori algorithms in the Hadoop-MapReduce environment. Section 7 briefly discusses the challenges and open issues of big data in the cloud and for Hadoop-MapReduce. Section 8 presents the discussion and conclusion.

2 BIG DATA AND CLOUD COMPUTING

Big data is the science of analyzing high volumes of diverse data in near-real time (volume, velocity, variety, veracity). Massive amount of data generated are generated daily due to technological growth, digitalization and by a variety of sources, including business application transactions, web pages, videos, images, e-mails, and social media. Typically it involves using NoSQL technology and a distributed architecture to analyze the data. The analysis can be done in the public cloud or on private infrastructure. Cloud computing provides IT resources such as Infrastructure, Platform, Software, Database, and Storage as service. It provides many features like: on-demand self-service, resource pooling, rapid elasticity, flexible scaling and high availability. Big data represents content and cloud computing is an environment that can be used to perform tasks on big data. Nonetheless, the two concepts are connected. In fact, big data can be processed, analyzed, and managed on cloud.

2.1 Big Data

Big data is the term used to delineate massive amounts of information of both structured and unstructured data types [113, 131]. Traditional database techniques and software applications are not suitable to process big data analytics. Big data is often characterized using four important aspects, namely data volume, data variety, data veracity and data velocity, sometimes referred to as the 4 V's of big data (volume, variety, veracity, and velocity) [45]. In addition, each of the four V's has its own consequences for analytics. Figure 2 shows the 4 V's characteristics of big data.

2.1.1 Volume

Volume refers to the generation and collection of different types of massive data. Most of the

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

datasets are too large to store and analyse using traditional database technology. Distributed systems are a new type of technology to overcome the above drawbacks. For instance, 40 zetabytes (43 trillion GB) of data will have been created by the year 2020, 6 billion people have mobile devices (85.7% of the total world population), and 2.5 quintillion bytes (2.3 trillion GB) of data are produced each day [45,50].

2.1.2 Velocity

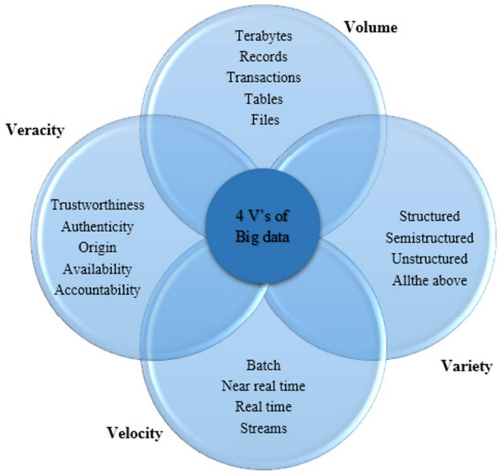

Figure. 2. 4 V's Characteristics of Big Data

Velocity represents the speed at which new data are generated, stored, analysed, and visualized. In the big data era, data are produced in real-time, or near real-time. Currently, the big data production speed from various perspectives is almost unbelievable. For instance, every minute, email users send 204 million messages, Facebook users share 2.46 million pieces of content, Google receives over 4 million search queries, and YouTube users upload 100 hours of video [45, 50].

2.1.3 Variety

Variety represents the various types of data users can now utilize. Previously, all data fell under categories of structured data. It was neatly stored in tables and relational databases. Currently, most of the world's data produced by many organisations and Internet service companies is unstructured data. Data can be categorized into four formats: structured data, semi-structured data, unstructured data, and complex structured data, such as audio, video, web pages, text, images, 3D models, simulations, and sensor data [50]. Big data requires different approaches and techniques to analyse and manage all of the raw data.

2.1.4 Veracity

Veracity refers to the uncertainty of data [18]. Different volumes of data come with different variety at high velocity from different sources. Organisations need to ensure the veracity of data, i.e., its accuracy, fidelity, and truthfulness, because improper data may cause significant problems for organisations as well as customers.

2.2 Cloud Computing

Cloud computing is a paradigm that is evolved from distributed processing, parallel computing, and grid computing. Cloud computing refers to "Outsourcing" and provides ubiquitous, expedient, and elastic services over the Internet or similar networks or both with access to a shared pool of computing resources, such as storage, memory, servers, network, applications, and services [33,52,65] The main features of cloud computing are resource pooling, rapid elasticity, self-service and on-demand capabilities, broad network access, virtualization, accessibility and scalability, and measured service [65]. Cloud computing delivers infrastructure, platform, and software as services via a cloud and delivers them over the Internet or a private network [29]. These cloud computing services can be generally categorized into three different service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [48,65,109].

2.2.1 Infrastructure as a Service (IaaS)

IaaS is one of three pillars of the cloud-computing model. This service model allows cloud users to use virtualized information technology resources for computing, storage, networking, and other infrastructure components on behalf of its cloud users [7,64,65,96]. The user can deploy and run his applications over his chosen operating system environment. The user does not have privilege to manage or control the underlying cloud infrastructure, but the user can control the operating system, computing storage, deployed applications and networking components [96].

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

Table 1: Advantages And Disadvantages Of Cloud

Advantages	Disadvantages
Cost efficiency	Security and privacy in the cloud due to multi-tenancy
Convenience and high availability	Performance problems due to the reliance on the Internet
Backup and recovery	
Quick deployment and ease of integration	
Resiliency and redundancy	
Device diversity and location independence	
Scalability and performance	
Increased storage capacity	
Automatic software integration	

Popular IaaS implementations include Amazon Web Service (AWS), GoGrid, Rackspace Cloud, Windows Azure, IBM Smart Cloud, and Google Compute Engine [28].

2.2.2 Platform as a Service (PaaS)

PaaS is the second pillar of the cloud-computing model. The platform cloud is an integrated computer system consisting of both hardware and software infrastructure. The user application can be deployed on this virtualized cloud platform using various software tools and programming languages supported by the cloud service provider. The cloud user cannot manage the underlying cloud infrastructure [101,106, 109]. The cloud provider supports user application development and testing on a well-defined service platform. This PaaS model permits a collaborated software platform for users from all over the world. This model also motivates third parties to provide software management, integration, and monitoring solutions. Common PaaS vendors include Google App Engine, Salesforce.com (CRM), Microsoft Azure, Amazon Elastic MapReduce, and Aneka [28].

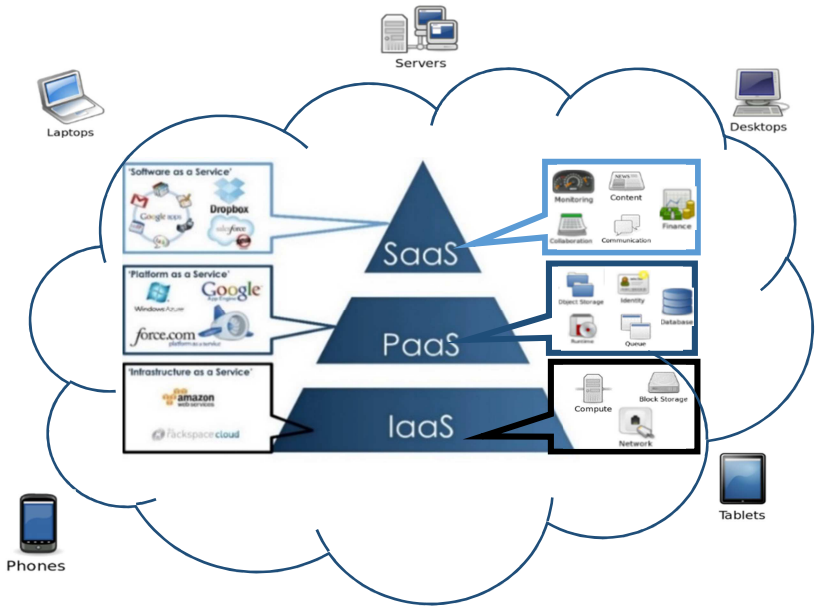

Figure. 3. Cloud computing service models with examples

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

2.2.3 Software as a Service (SaaS)

The third pillar of the cloud model is SaaS. SaaS is a way of delivering and managing software applications over the Internet by one or more cloud providers. SaaS application software (s/w) sometimes may refer to web-based s/w, on-demand s/w or hosted s/w. The providers have privileged access to the application, including security controls, availability, and performance. Suitable examples of SaaS services include Google Gmail and Docs, Microsoft SharePoint, and CRM software from Salesforce.com [28].

Figure 3 illustrates three cloud models at different service levels of the cloud. SaaS is applied at the application end using interfaces by users or clients, such as monitoring, communication, content management, finance, and collaboration. At the PaaS layer, the cloud platform must perform object storage, user identity, runtime, database and job queuing handling. At the bottom layer of the IaaS services, block storage, compute instances, the network, the file system, and storage must be provisioned to satisfy user demands.

Cloud computing has many advantages as well as some disadvantages. Table 1 describes the pros and cons of cloud computing.

3 DATA MINING TECHNIQUES

Data mining methods and tools are a set of well-defined procedures to create a data mining model from data repositories [47]. Typically, specified data mining tools analyse the user-defined data to create a model considering the exact types of patterns and trends of the data. The techniques reveal the solutions of this analysis to define the optimal parameters for creating the data mining model [12]. These parameters help to extract the patterns and detailed statistics from the entire dataset. Data mining methods involve seven common classes of data mining techniques according to function and application purpose, which are described as follows:

i. **Anomaly Detection or Outlier Detection**: Anomaly or Outlier detection refers to the identification of the items, events, and observations in a dataset that do not conform to normal behaviour. SVM, Fuzzy Logic, K-Means, Nearest Neighbour, and Outlier Count are the popular algorithms used in outlier detection [84]. The outlier detection techniques can be applied to numerous domains, such as network intrusion detection, telecommunication fraud detection, credit card fraud detection, fault detection, system health monitoring, and event detection in sensor networks [12,34,47]

.

ii. **Association Rule Mining (ARM):** ARM attempts to find frequent itemsets among large datasets and describes the association relationship among different attributes [85]. The most-used algorithms for ARM are Apriori, Eclat, FP-Growth, and partition. Market basket analysis, text mining, Web usage mining, protein sequences, and Bioinformatics are the application areas for ARM [12,13,34,47].

iii. **Classification:** Classification is the data mining function that assigns items in a collection to target categories or classes [34]. The goal of classification is to build a model that can accurately predict the target class for each case in the data. Decision Tree, Naïve Bayes, k-NN, GLM, and SVM are the well-known algorithms, and fraud detection, credit risk, stock market, DNA, and E-mail classification are well-established areas using classification techniques [12,34,47,85]

iv. **Clustering:** Clustering is the process of grouping a set of physical or abstract objects in such a way that objects in the same group are more similar to each other than to those in other groups. K-Means, canopy, DBSCAN, EM, Fuzzy, C-Means, CLOPE, and cobweb are popular implementations of clustering. The clustering techniques can be implemented in several fields, such as machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, crime analysis, and climatology (12,23, 34, 47].

v. **Regression:** Regression is commonly used to predict future trends based on past values by suitable points on the curve. Multivariate, and Adaptive Regression are two of the best algorithms used in regression analysis. The detection of fraud, and the minimization of risk assessments are suitable areas for the use of regression analysis [10, 47, 97].

vi. **Summarization:** Visualization and report generation is the main goal of the summarization. It provides a more compact representation of the dataset [47]. LexRank, TF–ISF, and TextRank are two of the best algorithms used in summarization. Multimedia documents, text summarization, and image collection are appropriate domains to

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

graphically represent data using summarization [3,69].

vii. **Mining Time-Series Data:** Many organisations and business industries utilize time-series or dynamic data. It is typically the case that all statistical and real-time control data used in process monitoring and control are essentially time series. A time-series database consists of sequences of values or events obtained over repeated measurements of time. The values are typically measured at equal time intervals (e.g. hourly, daily, and weekly). Time-series databases are popular in many applications, such as stock market analysis, economic and sales forecasting, budgetary analysis, utility studies, inventory studies, yield projections, workload projections, process and quality control, observation of natural phenomena (such as the atmosphere, temperature, wind, and earthquakes), scientific and engineering experiments, and medical treatments[35,97].

3.1 Association Rule Mining (ARM)

ARM is one of the key methods of data mining techniques, and it was introduced by Agrawal et al. in 1993. We elaborate some generic concepts of association rules of mining formally as follows:

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of m different literals or items. For instance, goods such as bags, pens, and pencils for purchase in a shop are items. X is a set of items such that $X \subseteq I$, a collection of zero or more items, is called an itemset. If an itemset contains k items, it is called a k-itemset. For example, a set of items for purchase from a super market is an itemset.

Let $D = \{t_i, t_{i+1}, \ldots\ldots, t_n\}$ be a set of transactions, where each transaction t has tid and $t - itemset$ $t = (tid, t - itemset)$.

The itemset X has in the transaction dataset D a support, denoted as S; if $S\%$ transaction contains X, here we called $S = Supp(X)$.

$$Supp(X) = \frac{|\{t \in D; X \subseteq t\}|}{|D|}$$

An itemset X in a transaction database D is called a large, or frequent, itemset if its support is equal to, or greater than, the threshold minimal support (*minsup*) given by users. The *negation* of an itemset X is $\neg X$.

The support of $\neg X$ is $supp(\neg X) = 1 - supp(X)$.

An association rule is an implication in the form of $X \to Y, where\ X, Y \subseteq I\ and\ X \cap Y = \phi$ [117,127]. The quality of an association rule can be represented as measurements, support and confidence.

$Support\ (S)$ determines how often a rule is applicable to a given dataset.

$S(X \to Y) = {\text{Supp}(X \cup Y)}/{D}$.

$Confidence(C)$ determines how frequently items in Y appear in transactions that contain X.

$C(X \to Y) = Supp(X \cup Y)/Supp(X)$.

The association rule mining task can be broken down into two sub-tasks [43, 116, 129].

I. Finding all of the frequent itemsets that have support above the user-specified minimum support, i.e. generating all frequent itemsets.

II. Generating all rules that have minimum confidence in the following simple way: For every frequent itemset X and any $B \subset X, let\ A = X - B$. If the confidence of a rule $A \to B$ is greater than or equal to the minimum confidence (or $supp(X)/supp(A) \geq minconf$), then it can be extracted as a valid rule.

The ARM performance typically depends on the first task. Usually, ARM generates a vast number of association rules. Most of the time, it is difficult for users to understand and confirm a huge number of complex association rules. Thus, it is important to generate only "interesting" and "non-redundant" rules or rules satisfying certain criteria, such as being easy to handle, control, understand, and increase the strength of. Dozens of algorithms have been developed to find the frequent itemsets and association rules in ARM, and some of the commonly used algorithms are the Apriori algorithm, partition algorithm, hash tree algorithm, dynamic item set counting algorithm, FP tree growth algorithm, Eclat and dEclat.

3.1.1 Apriori algorithm

The rapid advancement of information technology has resulted in the accumulation of tremendous amounts of data for organisations, and therefore, extracting needed information from huge amounts of data has been a significant challenge for researchers [79]. Apriori is a classic and broadly used ARM algorithm. It uses an iterative approach called breath-first search to generate $(k-1)$ itemsets from k item sets. The basic principal of this algorithm is that all nonempty subsets of a frequent itemset must be frequent [15,97]. There are two main steps in Apriori:

1. The prune step: remove an itemset if its support is less than min_sup, which is a value

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

predefined by the user, and abandon the itemset if its subset is not frequent.

2. The Join step: candidates are produced by joining among the frequent itemsets level-wise. The key drawback of this algorithm is the multiple dataset scans.

Algorithm 1 presents the pseudocode of the Apriori algorithm [97].

Algorithm 1: Frequent itemset generation of the Apriori algorithm

$k = 1$
$L_k = \{i | i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Find all frequent 1-itemsets}
repeat
$k = k+1$.
$C_k = apriori\text{-}gen(L_{k-1})$).
{Generate candidate itemsets}
for each transection $t \in T$ do
$C_t = subset(C_k, t)$
{Identify all candidates that belong to t}
for each candidate itemset $c \in C_t$ do
$\sigma(c) = \sigma(c)+1$. {Increment support count}
end for
end for
$L_k = \{c | c \in C_k \wedge \sigma(c) \geq N \, x \, minsup$ {Extract the frequent k-itemsets}
until $L_k = \phi$
$Result = \cup L_k$

Cloud could be a perfect platform for data mining algorithms because of the advantages of cloud such as availability, scalability, recovery, performance, and cost effective. However, the classical Apriori algorithm cannot be implemented in the parallel environment because it was intended for sequential processing. Numerous techniques have been proposed to improve the efficiency of the classical Apriori algorithm, such as direct hash and pruning (DHP), transaction reduction, partitioning, sampling, dynamic itemset counting (DIC), vertical layout techniques, and FP-Growth [53]. Table 2 describe some popular sequential Apriori algorithms that have improved the efficiency and performance of the original Apriori algorithm in a sequential manner.

4 PARALLEL DISCOVERY OF THE APRIORI ALGORITHM

The current parallel and distributed algorithms are based on the classical Apriori algorithm. Classical Apriori is a well-known algorithm for discovering frequent itemsets from a transactional database; however, it needs to scan the dataset repeatedly to find frequent itemsets, and it generates numerous candidate itemsets. Unfortunately, if the dataset is massive, both the memory usage and computational cost are more expensive. Moreover, a single machine processor's memory and central processing unit resources are inadequate, which makes the algorithm performance inefficient. Furthermore, because of the exponential increase in global information, many organisations must deal with big data. As these data grow past GBs towards TBs or more, it becomes infeasible to manage, store, and analyse such data on a single sequential machine.

Parallel and distributed computing offer a potential solution for the above problems when efficient and scalable parallel and distributed algorithms can be implemented. Count Distribution (CD), Data Distribution (DD) and Candidate Distribution are three parallel versions of the Apriori algorithm that have been developed by Agrawal and Shafer [5]. Ever since, many versions of parallel algorithms
have been proposed to improve the efficiency of the classical Apriori algorithm [1,5,11,20,21,22 ,36,37,51,72,73,78, 89,90,94,103,112, 118,119, 120,123,125]. These parallel algorithms can be implemented in the cloud-computing environment to reduce computation time, memory usage and I/O overhead for generating frequent itemsets. This can boost the performance of association rule-mining algorithms.

4.1 Count Distribution algorithm (CD)

This technique involves the data parallelism approach that splits the database into horizontal partitions and then scans individually to find the local counts of all candidate itemsets on each process. Finally, the local counts are summed up after every iteration to obtain the global count to find frequent itemsets. The main advantage of this method is the minimization of the communication cost, as data tuples are not exchanged among processors, only the counts are exchanged [5,125]. Table 3 describes the count distribution algorithm

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

Table 2: Methods for improving the Apriori algorithm

Method	Description	Storage data structure	Database	Algorithm proposed by
Direct hash and pruning (DHP)	This method attempts to generate large itemsets efficiently and reduces the transaction database size. When generating L_1, the algorithm also generates all of the 2-itemsets for each transaction, hashes them to a hash table and keeps a count.	array	Suitable for medium-size databases	[68,75,76,77,122]
Transaction reduction	A transaction that does not contain any frequent k+1-itemset may be marked and removed.	array	Suitable for small- and medium-size databases	[93,100,117]
Partitioning	The set of transactions may be divided into a number of disjoint subsets. Then, each partition is searched for frequent itemsets. These frequent itemsets are called local frequent itemsets.	array	More suitable for huge-size databases	[30, 80, 81, 125]
Sampling	A random sample (usually large enough to fit in the main memory) may be obtained from the overall set of transactions, and the sample is searched for frequent itemsets	array	Right fit for all sizes of databases	[42, 62, 102,123]
Dynamic itemset counting (DIC)	DIC allows for the counting of an itemset to begin as soon as we suspect that it may be necessary to count it.	array	Appropriate for small- and medium-size databases	[16,17,19,98]
Vertical layout technique (Eclat)	The Eclat algorithm is based on depth-first search algorithm techniques. It uses a vertical database layout instance of a horizontal layout, i.e., instead of explicitly listing all transactions, each item is stored together with its cover (also called tidlist), and the intersection-based approach is used to compute the support of an itemset.	array	Suitable for medium-size and dense datasets but not small-size datasets	[103, 121]

pseudocode and complexity analysis.

4.2 Data Distribution Algorithm (DD)

This method is helpful in using the average main memory of machines in parallel by partitioning both the database and the candidate itemsets. As each candidate itemset is counted by only one process, all processes have to exchange database

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

partitions during each iteration for each process to obtain the global counts of the assigned candidate itemsets. The key advantages of this method are that the total memory of the system is used more efficiently and that this algorithm is viable only on a machine with very fast communication [5,125]. Table 4 explains the data distribution algorithm pseudocode and complexity analysis.

4.3 Candidate Distribution algorithm

This method also partitions candidate itemsets but selectively uses replicas, instead of partitioning and exchanging database transactions, such that each process can proceed independently. The main advantage of this method is that this algorithm tries to do away with the dependence between processors; therefore, processors work separately without synchronizing. Table 5 explains the data distribution algorithm pseudocode.

4.4 Challenges of a parallel Apriori algorithm

Apriori parallel algorithms handle gigantic datasets on various platforms with different configurations. There are many major challenges that need to be considered, as mentioned below.

- How can efficacious load balancing be achieved, and which type of load balancing (static or dynamic) is suitable for a particular algorithm to use?
- How can the total memory system be used effectively?
- How can new algorithms be produced for different memory systems?
- Which data layout is more convenient to use (horizontal, vertical, hybrid, or projection)?
- How is the choice of which parallel technique (data or task parallelism) to use in a new algorithm decided?
- How can the communication cost among processors be reduced?
- How can synchronization be minimized?
- How can system failure and data recovery be managed?
- How can parallel programming issues be simplified?
- How can scalability and high availability be managed?

5 APRIORI ALGORITHM ON THE HADOOP-MAPREDUCE MODEL

To overcome the above-stated major challenges, the Hadoop-MapReduce framework is a suitable solution. Hadoop is an open-source programming framework that is capable of running applications for large-scale processing and storage on large clusters of commodity hardware [57,110]. Hadoop cluster characteristics include the partitioning or distributing of data, computation across multiple nodes, and performing computations in parallel [88]. It provides applications with both reliability and data motion. The Apache Hadoop framework consists of four core components: Hadoop common, Hadoop distributed file system, Hadoop YARN, and Hadoop-MapReduce. Hadoop common contains Java libraries and utilities required by other Hadoop components. These libraries contain the file system and essential Java files and scripts and operating system-level abstractions.

The Hadoop Distributed File System (HDFS) that stores data on the simple computer machines provides high-throughput aggregate bandwidth across the cluster [91]. Hadoop YARN is a resource-management framework for handling compute resources and job scheduling of user applications [105]. Hadoop-MapReduce is a programming model for parallel processing of large-scale datasets. Beyond the above four modules, the Apache Hadoop framework has many related projects, such as Oozie, HBase, Pig, Mahout, Hive, Sqoop, Flume, and Zookeeper. Figure 4 illustrates components of the Hadoop framework and their ecosystem [82].

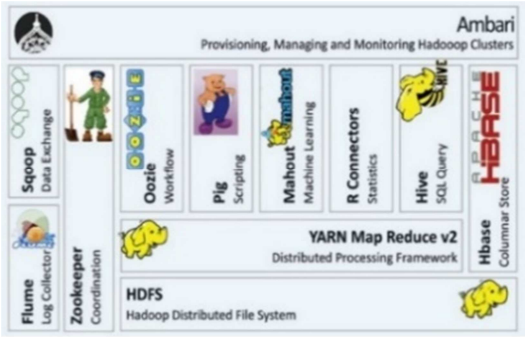

Figure 4. Hadoop framework and its ecosystem

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

Table 3: CD Algorithm Pseudocode And Analysis

Method	Pseudo code	Analysis
Count Distribution Algorithm (CD)	Pass>1 Generate complete C_k from L_{k-1} Count local data D_i to find support for C_k Exchange local count for C_k to find global C_k Compute L_k from C_k Pass k=1 Generate local C_1 from local data D_i Merge local candidate sets to find C_1	Input/output Time $O(N/P)$ CPU Time $O(count)/P + overhead$ Communication Volume $\sum_k O(\lvert C_k \rvert)\ per\ CPU$ Message Count $\sum_k \log P$

Table 4: DD Algorithm Pseudocode And Analysis

Method	Pseudo code	Analysis
Data Distribution Algorithm (DD)	Partition data + candidate set Generate C_k from L_{k-1} ; retain $\lvert C_k \rvert/P$ locally Count local C_k using both local and remote data Calculate local L_k using the local C_k and synchronize	Input/output Time $O(N/P)$ CPU Time $O(count)/P + overhead$ Communication Volume $\sum_k O(N)\ per\ CPU$ Message Count $\sum_k O(P)$

Table 5: Candidate Distribution Algorithm Pseudocode

Method	Pseudo code
Candidate Distribution Algorithm	Pass k < l: Use either Count or Data distribution algorithm Pass k = l: Partition L_{k-1} among the N processors such that L_{k-1} sets are "well balanced". Processor Pi generates C_k logically using only the L_{k-1} partition assigned to it. Pi develops global counts for candidate in C_k and the database is repartitioned into D_i at the same time. After P_i has processed all its local data and any data received from all other processors, it posts N-1 asynchronous receive buffers to receive L_k from all other processors. These L_k are needed for pruning C_{k+1} in the prune step of candidate generation. Processor P_i computes L_k from C_k and asynchronously broadcasts it to the other N-1 processors using N-1 asynchronous sends. Pass $K>1$: Processor P_i collects all frequent itemsets that have been sent to it by other processors. P_i generates C_k using the local L_{k-1}. P_i makes a pass over D_i and counts C_k.

Journal of Theoretical and Applied Information Technology
31st March 2016. Vol.85. No.3

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

5.1 Hadoop Common

The Hadoop software library is an Apache open-source framework written in the Java programming language that allows for the distributed processing of huge datasets across clusters of computers using simple programming models [8,44]. It is an Apache project released under Apache Open Source License v2.0, which is very commercially friendly. The Hadoop framework provides distributed storage and computation among computers on clusters. HDFS provides the distributed storage. MapReduce performs the computing process across the clusters of computers [44, 57,110]. Figure 5 illustrates the high-level architecture of Hadoop.

5.2 Yet Another Resource Negotiator (YARN)

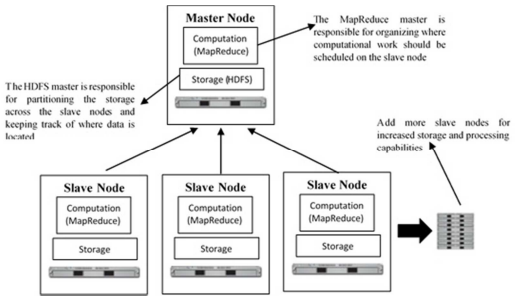

Figure. 5. High-level architecture of Hadoop

Apache Yet Another Resource Negotiator (YARN) is a next-generation compute and resource management infrastructure, and it delegates many scheduling functions (e.g. task fault tolerance) to per-application components in Hadoop[110]. YARN eliminates the scalability limitation of the first-generation MapReduce paradigm. YARN's basic idea is to split the two major functionalities of the JobTracker, resource management and job scheduling, into separate daemons.

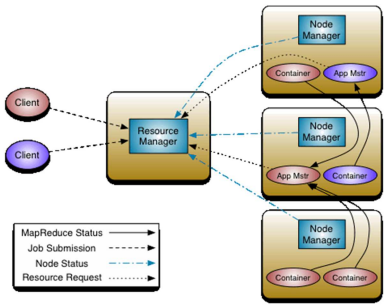

Figure. 6. YARN architecture [8]

The idea is to have a global ResourceManager and per-application ApplicationMaster. The ResourceManager arbitrates resources among all of the applications in the system and has two components: the Scheduler and Applications Manager [56]. The YARN architecture is shown in Figure 6.

5.3 Hadoop Distributed File System (HDFS)

HDFS is responsible for storing very large data reliably on a cluster in Hadoop. Files in HDFS are split into blocks before they are stored on the cluster. The typical size of a block is 64 MB or 128 MB. The blocks belonging to one file are then stored on different nodes. HDFS separately stores application data as well as file-system metadata. HDFS stores metadata on the NameNode, and application data are stored on DataNodes. All nodes are fully connected and communicate among themselves using TCP-based protocols. HDFS has many characteristics, such as its 'writing to one-reading to many' model, processing logic near to the data instead of moving the data to the processing logic, data access through MapReduce streaming, simple and robust model, scalability for processing huge data and storing data reliably, cost effectiveness for distributing and processing data across clusters of commodity hardware, efficient distribution of data and parallel processing, and reliability manifested in maintaining multiple copies of data on existing nodes [91,92]. Figure 7 describes a logical representation of the components in HDFS.

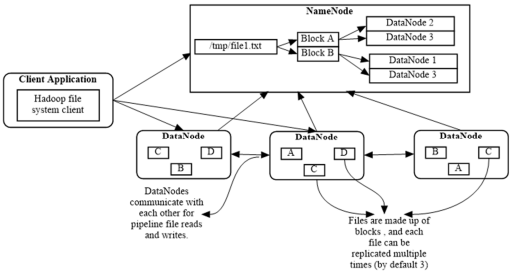

Figure. 7. HDFS shows how a client communicates with the master NameNode and slave DataNodes

5.4 MapReduce

MapReduce is a software framework designed to process and generate large amounts of data in-parallel across distributed clusters (thousands of nodes) of personal computer hardware in a trustworthy, fault-recoverable manner [74]. The

Journal of Theoretical and Applied Information Technology
31st March 2016. Vol.85. No.3

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

MapReduce paradigm is composed of the map task, which takes input pairs and produces intermediate key/value pairs, and the reduce task, which accepts the output key/value pairs from the map task and merges, shuffles, and combines those key/value pairs.

The map task is always performed before the reduce task. Generally, both the input and the output are stored in HDFS. The framework is responsible for scheduling tasks, monitoring them and re-executing the failed ones [44, 57, 110]. A logical view of the MapReduce programming model is shown in Figure 8.

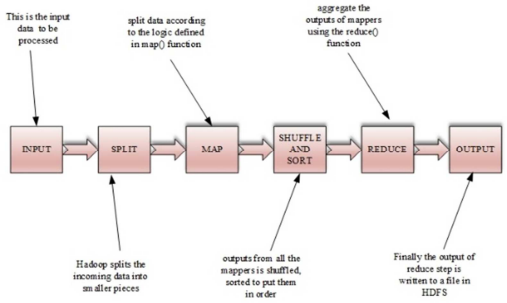

Figure. 8. Logical view of the MapReduce programming model

The map operation and reduce function need to be defined by the programmer. Figure 9 shows a logical view of the map function with respect to its input and output.

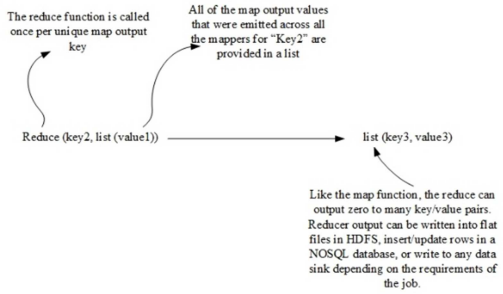

Figure 9. Logical view of the map function

The shuffle and sort phases are illustrated in Figure 10.

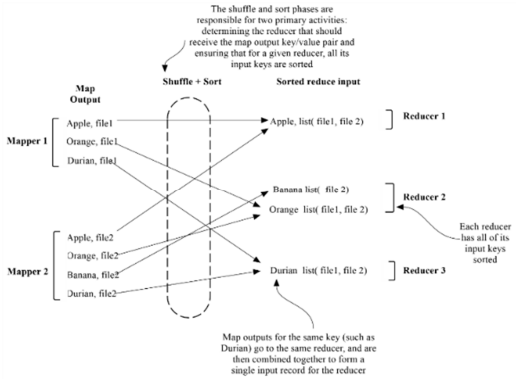

Figure. 10. Logical view of reduce function

Figure 11 illustrates the pseudocode definition of a reduce function.

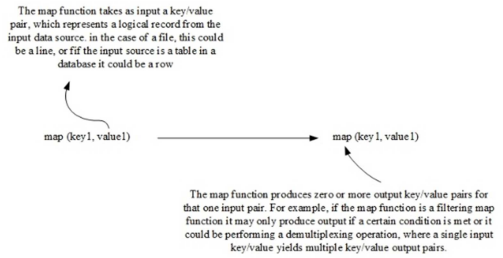

Figure. 11. Logical view of the reduce function

Hadoop's MapReduce architecture is based on the master-slave architecture model in HDFS. The overall logical architecture of the Hadoop-MapReduce framework with its main components is shown in Figure 12

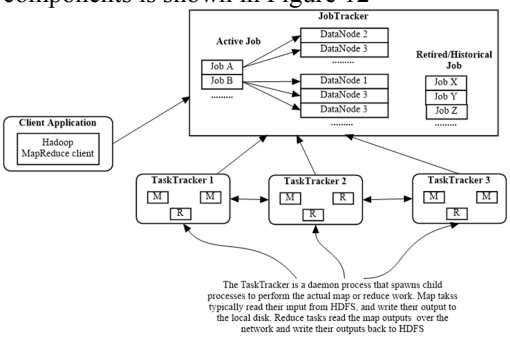

Figure. 12. Logical architecture of MapReduce

5.5 Apriori example on the Hadoop MapReduce model

The following example briefly describes how to implement the Apriori algorithm on the Hadoop-MapReduce model, as shown in Figure 13, using the Map, Shuffle, and Reduce processes.

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

6. ANALYSIS OF SEVERAL IMPROVED APRIORI ALGORITHMS IN THE HADOOP-MAPREDUCE ENVIRONMENT

In this section, the related works are analysed regarding four categories: the objective of the work, main theme of the researchers' concern, datasets that were used for the experiment, and machine configuration and platform used to execute the proposed studies. The following tables describe the above four categories as follows:

Table 6 presents several related studies that use a parallel Apriori algorithm to deal with big data through the use of the Hadoop and MapReduce distributed framework. This table provides the objectives of related papers that deal with a parallel Apriori algorithm based on Hadoop technologies and describes the techniques and technologies used in the Hadoop-MapReduce environment to address big data.

Table 7 summarizes the main theme of the related studies.

Table 8 demonstrates the datasets that were used to analyse the proposed studies. Some researchers have given very clear details of the dataset that they used.

Table 9 summarizes the hardware and software configuration and platform where the authors implemented their experiment.

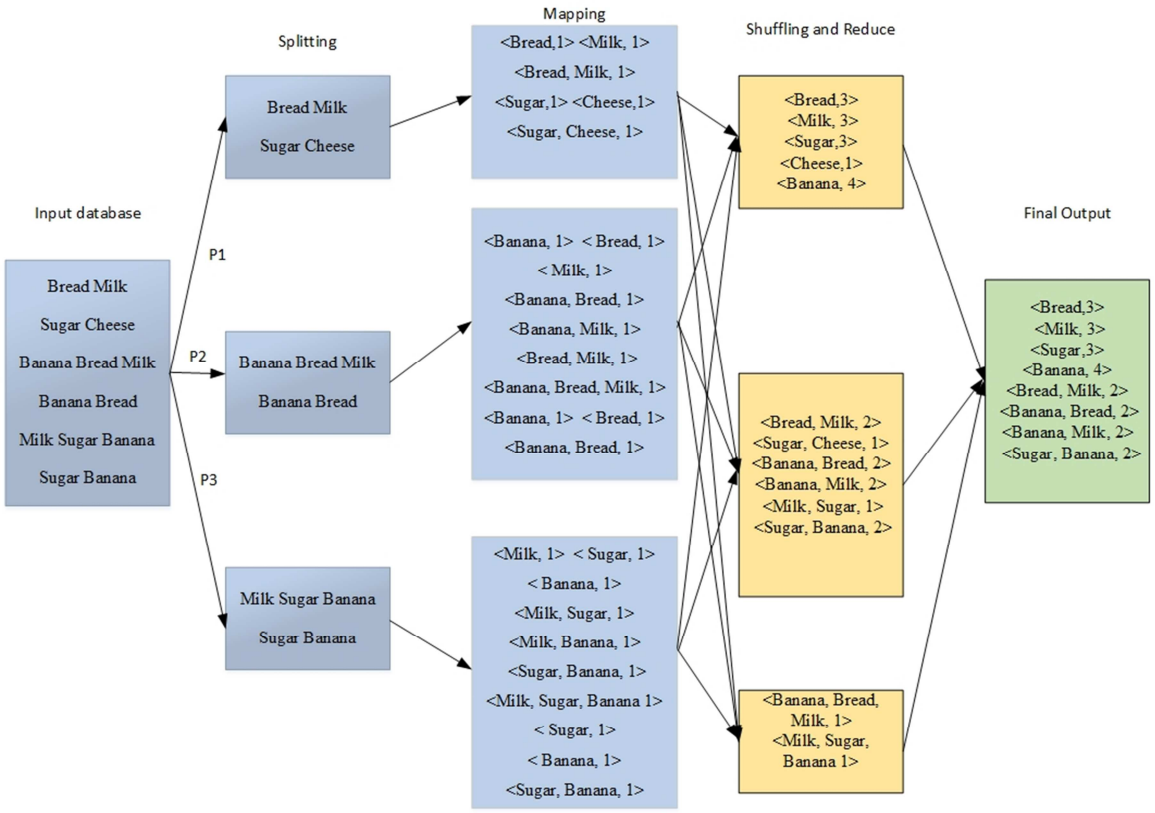

Figure. 13. Apriori example on MapReduce framework

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

Table 6: Review of several improved Apriori algorithms on Hadoop-MapReduce and their objectives

Reference	Objective
[60]	To evaluate the performance of their proposed Apriori algorithm in terms of size-up, speedup, and scale-up to address massive-scale datasets.
[117]	They proved their improved Apriori algorithm on a theoretical basis. First, they replace the transaction dataset using the Boolean matrix array; by this method, non-frequent item sets can be removed from the matrix, and there is no need to scan the original database repeatedly. It just needs to operate on the Boolean matrix using the vector operation "AND" and the random access characteristics of the array so that it can directly generate the k- frequent itemsets.
[74]	From this study, they suggested a count distribution algorithm as the best way to parallelize the Apriori algorithm. Hadoop has three modes of operation: as the Standalone, Pseudo-Distributed, and Fully-Distributed modes. The count distribution strategy was chosen to implement the Apriori algorithm on a Hadoop cluster.
[31]	The authors have improved the Apriori algorithm using the following steps: • Split the transaction database horizontally into *n* data subsets and distribute them to *'m'* nodes. • Each node scans its own datasets and generates a set of candidate itemsets C_p • Then, the support count of each candidate itemset is set to 1. This candidate itemset C_p is divided into *r* partitions and sent to *r* nodes with their support count. *r* nodes respectively accumulate the support count of the same itemset to produce the final practical support and determine the frequent itemset L_p in the partition after comparing with the min_sup. • Finally merge the output of r nodes to generate set of global frequent itemset *L*
[111]	To illustrate its time complexity, which theoretically shows that the algorithm has much higher performance than the sequential algorithm when the map and reduce nodes get added.
[114]	To compare and prove the good performance of the newly proposed 2-phase algorithm with previously existing 1-phase and *k*-phase scanning algorithms repeatedly changing the number of transaction and minimum support.
[59])	To deploy the revised Apriori algorithm on Amazon Elastic Cloud Computing (EC2) to assess the performance when varying the number of nodes and min_sup threshold.
[54]	To produce all of the subsets that would be generated from the given itemset, these subsets are searched against the datasets and frequency is noted. There are huge data itemsets and their subsets, hence they must be searched simultaneously so that the search time is reduced. An experiment evaluated this by changing the Hadoop mode with Fully configured multi-node Hadoop with Different System Configuration (FHDSC) and Fully configured multi-node Hadoop with Similar System Configuration (FHSSC) $$N = \frac{FHDSC}{FHSSC}$$ $$FHDSC = FHSSC = log_e N$$ where *N* is the number of nodes installed in the cluster

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

[83]	The Apriori algorithm is a famous algorithm for association rule mining. The traditional Apriori algorithm is not suitable for the cloud-computing paradigm because it was not designed for parallel and distributed computing. The current implementations have the drawback that they do not scale linearly as the number of records increases. Secondly, the execution time increases when a higher value of k-itemsets is required. Thus, the authors have attempted to overcome the above limitations, and they have improved the Apriori algorithm such that it now has the following features: • It will scale linearly as the number of records increases. • The time taken will be agnostic to the value k. That is, whatever k-itemsets run occurs, it will take the same amount of time for the given number of records.
[49]	The authors have produced a new map-reduce-based algorithm expressing the problem of mining frequent itemsets using dynamic workload management through a block-based partitioning. The block-based approach addresses memory constraints because the basic task of generating combinations may need a very large memory space depending on several parameters, including the support threshold. The features of the proposed algorithm are that it uses a properly tuned estimation to measure the correct itemset during the pass, and the redundancy is eliminated by handling duplicates carefully and because of the workload management.
[46]	In this study, the authors proposed a new parallel frequent pattern algorithm. The proposed algorithm has five steps: • Divide the complete transaction database Db into the number of available processors so that each individual processor works on the allocated DB in parallel • Construct the Global Frequent Header Table using a MapReduce task for the first time • Generate conditional frequent patterns using a map task the second time • Each reducer combines all conditional frequent patterns of the same key to generate frequent sets • All reducers' outputs are aggregated to generate the final frequent sets.
[66]	They proposed an algorithm that used a Trie structure and grouping methodology, but it will locally prune the dataset at the base level as well as at an intermediate level by grouping nodes into clusters. At the end, the final aggregation of the frequent itemsets is done at the global level. The proposed algorithm can reduce the message passing overhead. Like the gossip-based mining approach, the overall message passing overhead occurs because it follows a random approach for communication, and, if any update occurs, then the message passing should occur randomly for all nodes, while, in our proposed strategy, the global dataset generated is centralized, so it will take only one message for one node to communicate. Additionally, fault tolerance is automatically resolved by using the Hadoop distributed framework.

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

[107] The authors introduce a method named the Token-Based approach that is used to generate precise frequent patterns of data. The input of this module is the output from the previous module. The output displayed after the elimination of the threshold value is the frequent pattern. They evaluated the performance by using a confusion matrix; it provides the accuracy of the patterns.

Confusion Matrix	Frequent	Infrequent
Frequent	True Positive(TP)	False Negative(FN)
Infrequent	False Positive(FP)	True Negative(TN)

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$$Precision = \frac{TP}{(TP + FP)}$$

$$Sensitivity(Recall) = \frac{TP}{(TP + FN)}$$

[104] From this study, they introduced a method to measure the performance of the distributed algorithm. Measuring the spent time of an algorithm that runs on multiple machines in a parallel environment is not a trivial task. The Hadoop framework can tell how long a job was running, but their computations are using multiple jobs; they can sum up these jobs of course, but the framework does many things in the background, such as copying code and data between the nodes as well as ordering and shuffling the intermediate job data. Thus, they created a scaffolding framework for their implementation that is responsible for measuring the time spent on each task.

[55] Several Apriori-based algorithms were developed, but the modifications are based on the reduction of the number of the database scans. However, Apriori-based algorithms have some other drawbacks. The role of the 2 frequent itemsets is fundamental, therefore it is useful to take care of the second candidates. In this case, the itemset counters can be stored in a triangular matrix. This triangular matrix can be indexed by the itemset identifiers. Hence, this matrix contains counter values of each 2 itemsets, and the diagonal of the matrix can be the place holder of the 1-item set counters. The advantages of this solution are as follows:

- It is necessary to generate the 2 candidate
- It is possible to count the 1 and 2 itemset in one scan
- Each of the counters can be reached in *O(1)* time
- It is independent of the size of database, as it depends only on the number of items.

[41] The goal of the paper is to experimentally evaluate association rule-mining approaches using a vertical dataset layout and database partitioning on the Hadoop-MapReduce framework. They developed a new MapReduce-based association rule miner for extracting strong rules from benchmark datasets.

[67] They introduced two algorithms based on the MapReduce framework for which the frequency thresholds can be set low:

1. Dist-Eclat is a MapReduce implementation of the well-known Eclat algorithm, optimized for speed in case a specific encoding of the data fits into memory. This technique is able to mine large datasets, but it can be prohibitive when dealing with massive amounts of data.
2. BigFIM is optimized to deal with truly Big Data by using a hybrid algorithm, combining principles from both Apriori and Eclat.

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

[58]	They proposed a MapReduce-based parallel FP-Growth algorithm, which intelligently shards a large-scale mining task into independent parallel computational tasks and maps them onto MapReduce jobs. The proposed algorithm then uses the MapReduce model to take advantage of its recovery model. It can achieve near-linear speedup with the capability of restarting from computer failures. They experimented with their proposed algorithm on a massive dataset and verified the outstanding scalability of this algorithm. To make the algorithm suitable for mining Web data, which are usually of long tail distribution, they designed this algorithm to mine top-k patterns related to each item rather than relying on a user-specified value for the global minimal support threshold. They demonstrated that the proposed algorithm is effective in mining tag-tag associations and WebPage-WebPage associations to support query recommendation or related search.

Table 7: Main theme of the proposed approaches

Reference	Main theme
[60]	In this study, the authors implemented a parallel Apriori algorithm in the context of the MapReduce paradigm. MapReduce is a framework for parallel data processing in a high-performance cluster computing environment
[117]	The aims of this study were to find the frequent itemsets and association rule in the transactional database with the min_sup and min_conf predefined by the user on the Hadoop-MapReduce framework. Typically, the standard Apriori algorithm has many challenges to discover the frequent itemsets from the huge dataset efficiently and quickly.
[74]	ARM plays a key role in data mining techniques. Apriori is an extensively used algorithm in association rule mining. The classical Apriori algorithm runs on a single node. Thus, it is difficult to process large datasets on a single node using the classical Apriori algorithm. There have been various studies for parallelizing the algorithm. In this study, Apache Hadoop has been chosen as the distributed framework to implement the Apriori algorithm, and the performance of the algorithm was evaluated on Hadoop. The classical Apriori algorithm was modified to be run on Hadoop.
[114]	They introduced the ability to find all k-frequent itemsets within only two phases of scanning the entire dataset and implemented that in a MapReduce Apriori algorithm on the Hadoop-MapReduce model efficiently and effectively compared with the 1-phase and k-phase algorithms.
[59]	Big-data analysis is a very important research field in data mining strategies in the context of the cloud-computing paradigm. In this study, the authors developed a new parallel Apriori algorithm based on the existing sequential Apriori association rule algorithm for implementation on the Hadoop-MapReduce parallel computing platform.
[54]	Apriori is a broadly used algorithm for obtaining all frequent itemsets. Currently, data come from various domains in various formats with unbelievable velocity. Processing, managing, and analysing such huge amounts of data are impossible using simple computing machines. Hence, distributed and parallel environments undertake such scenarios efficiently. Apache Hadoop-MapReduce distribution is one of the cluster frameworks in a distributed environment that helps by distributing voluminous data across a number of nodes in the framework

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

[31] This study has implemented a revised Apriori algorithm to extract frequent pattern itemsets from transactional databases based on the Hadoop-MapReduce framework. They used the single-node Hadoop cluster mode to evaluate the performance. This redesigned algorithm can be easily parallelized and is easy to implement

[83] Cloud computing has become a big name in the present era. It has proved to be a great solution for storing and processing huge amounts of data. Data mining techniques implemented with the cloud-computing paradigm can be useful for analysing big data in the cloud. In this paper, the researchers used the Apriori algorithm for association rules in a cloud environment.

[49] From this study, the authors proposed one such distributed algorithm that is run on Hadoop, which is one of the most popular recent distributed frameworks and is mainly focused on the map/reduce paradigm.

[46] In data mining, research in the field of distributed and parallel mining is gaining popularity as there is a tremendous increase in the sizes of databases stored in data warehouses. Frequent pattern mining is one of the important areas in data mining for finding associations in databases. As FP-Tree is considered the best compact data structure for holding data patterns in memory, there have been efforts to make it parallel and distributed to handle large databases. However, it incurs large amounts of communication overhead during the mining. In this study, the authors proposed the Hadoop map/reduce framework as a parallel and distributed framework for frequent pattern mining algorithm as it shows the best performance results for large databases.

[66] In this study, the authors discussed different distributed Apriori algorithms proposed by different authors and compared those algorithms with a comparison matrix. They also proposed a new algorithm.

COMPARISON MATRIX

Algorithm	Complexity	Message passing overhead
Count Distribution (CD)	High, $O(\|C_k\|n)$, where C_k is a candidate itemset and n is the number of site	Less
Fast Distribution Mining (FDM)	High, less than CD $O(\|C_P\|n)$, where C_p is the union of all local candidate itemsets	Less
Optimized Distribution Association Mining (ODAM)	Low compared to CD,FDM $O(\|C_R+P(F_D\|*n)$, where C_R is the intersection of all local frequent itemsets, $P(F_D)$ is the total number of disjoint local frequent itemsets that have higher probability, and n is the total number of sites	High
Distributed Trie Frequent Itemset Mining (DTFIM)	Low compared to CD,FDM,ODAM ($O(n^2)$)	High
Gossip-based distributed frequent itemset mining	Low compared to all of the above *(O(nlogn))*	High

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

[107]	Data mining is a rapidly growing field, and it has become a popular research area. In this study, a system was proposed to find the frequent patterns in order to increase the profit by making the frequent items available consistently. A token-based approach is proposed in order to find the frequent pattern from the intermediate output. There are many existing methods for analysing big data and frequent patterns, but the proposed system is fault tolerant, time consuming and highly reliable.
[58]	Fault recovery is a typical problem in massive computing environments because the probability that none of the thousands of computers crashes during execution of a task is close to zero. The demands of sustainable speedup and fault tolerance require highly constrained and efficient communication protocols. Thus, they show that their proposed solution is able to address the issues of memory use and fault tolerance in addition to more effectively parallelizing computation.
[67]	The main challenge in adapting algorithms to the MapReduce framework is the limitation of the communication between tasks, which are run as batches in parallel. All of the tasks that have to be executed should be defined at start-up, such that nodes do not have to communicate with each other during task execution. Fortunately, the prefix tree that is used by Eclat can be partitioned into independent groups. Each one of these independent groups can be mined separately on different machines. The proposed algorithms exploit the inter-independence of sub-trees of a prefix tree and use longer FIs as prefixes for better load balancing.

Table 8: Details of the datasets used in experiments

Reference	Dataset
[60]	In this study, they used the transactional data for an all-electronics branch and the T1014D100K dataset. They replicated it to obtain 1 GB, 2 Gb, 4 GB, and 8 GB. For the T1014D100K dataset, they have replicated it into 2 times, 4 times, and 8 times and got 0.6 GB, 1.2 GB and 2.4 GB datasets, respectively. They denoted those datasets as T1014D200K, T1014D400K and T1014D800K. Additionally, they used some transactional logs from a telecommunication company.
[117]	Sample
[74]	Accident dataset- National Institute of Statistics (NIS) for the region of Flanders (Belgium) period from 1991-2000. 340,184 records, 572 attributes and they used synthetically generated dataset
[114]	In this study, the T1014D100K dataset was used to obtain the experiment results generated by IBM's quest synthetic data generator. The total number of transactions is 100000, and each transaction contains 10 items on average; the total number of items is 1000, and the average length of frequent itemsets is 4

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

[59] The datasets of chess, mushroom, and connect are real-life datasets downloaded from the UCI dataset repository, and T1014D100K was synthetically generated by the IBM Almaden Quest research group

Dataset	Total Instances	Total Attributes
Chess	3196	36
Mushroom	8124	22
Connect	67557	42
T1014D100K	100000	26

[54] Transactional Dataset

[31] Word count Example

[83] INA[1]

[49] The experiment is done using two real datasets that are publicly available and have different characteristics: one is generated by an IBM synthetic data generator and the other is the basket dataset.

Table I: Number of Items and Transactions in each Dataset

Dataset	σ	$\|F_1\|$	$\|F\|$	Max $\{k\|\|F_k\|>0\}$c
T2017D500K	700	804	550126	18
Retail Market	7	8051	285758	11

Table II: Lowest Min Threshold for each Dataset

Dataset	#Items	#Transaction	Min|T|	Max|T|	Avg|T|
T2017D100K	942	90000	4	77	39
Retail Market	16470	88163	1	51	13

[46] Several real and synthetic datasets are used to test the performance of the algorithm.

[66] Mushroom dataset from UCI (8124 number of records and 23 transaction length)

[58] They generated a tag transaction database and named it TTD and a URL transaction database, as shown in below

	TTD	WWD
URLs	802,939	802,739
Tags	1.021.107	1,021,107
Transactions	15.898.949	7,009,457
Total items	84,925,908	38,333,653

[1] INA-Information Not Available

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

[67] They have used the benchmark data, such as Abstract, T1014D100K, Mushroom, and Pumsb. Additionally, they used a scrape of the delicious dataset provided by DAI-Labor.

Dataset	Number of Items	Number of Transactions
Abstracts	4,976	859
T1014D100K	870	100,000
Mushroom	119	8,124
Pumsb	2,113	49,046
Tag	45,446,863	6,201,207

[107] The daily, monthly, and yearly sales of various shops are collected from various websites.

Table 9: Experiment platform and H/W & S/W Configuration

Reference	Hadoop mode	#phase	Introduced Algorithm
[59]	INA	k	PApriori
[74]	Fully distributed	k	Modified Apriori
[114]	Standalone	2	MRAPRIORI
[59]	Fully Distribution	INA	Modified Apriori
[54]	Stand alone, pseudo, and fully distributed	INA	Modified Apriori
[31]	Standalone	1	Modified Apriori
[83]	Fully distribution	2	Modified Apriori
[46]	Standalone	2	DPFPM
[66]	Standalone	2	Modified Apriori
[67]	Standalone	k	Dist-Eclat and BigFIM
[58]	INA	k	Parallel FP (PFP)
[107]	INA	INA	Modified Apriori
[117]	-	1	None
[111]	-	-	Apriori-Map/Reduce

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

6 CHALLENGES, AND OPEN ISSUES OF BIG DATA IN THE CLOUD AND HADOOP-MAPREDUCE

6.1 Big data in the Cloud: Challenges

- Scalability: How can the explosion growth of data be handled in an appropriate manner?
- Availability: How can the cloud service providers be trusted to continuously provide data that are stored in the cloud?
- Data integrity: How can it be ensured that a cloud user's data are not being viewed, altered, or analysed by authorized parties or data owner?
- Data Transformation: Big data has a variety of data formats, so how can data be transformed into a suitable format for big data analysis?
- Data Quality: Big data originates from various domains, so how can high-quality data be obtained from various sources?
- Data Heterogeneity: Big data comes from numerous sources with different types of data and they have incompatible format and inconsistency, so how can the challenge of big data analysis and management on a cloud platform be addressed?
- Privacy: How can the reliability of storing private data in cloud storage be ensured?
- Legal issues: How can the unique laws and regulations be established to achieve data privacy and protection?
- Data governance: How can a well-defined and acceptable data policy, according to the type of data to be stored, how quickly an individual data access must be, and how the private data are accessed, be created?
- Timeliness: How can interaction response times to complex queries at scale over high-volume event streams be provided?
- Data visualization: How can the visualization of big data be improved in an appropriate way?

6.2 Big Data in the Cloud: Open Issues

- Find the more feasible techniques to simplify the transformation and cleaning of unstructured big data in a cloud platform such as the Hadoop-MapReduce distribution platform.
- Improve the techniques to store, easily retrieved, and migrated big data between cloud servers and other devices.
- There is a door for efficiently improving the data analysis tools and technologies that are highly appreciated under the cloud platform.
- Data security is the keen open issue to solve the unresolved security threats that exist in cloud computing, such as privacy, confidentiality, data integrity, and availability of big data using the in-cloud environment.

6.3 Hadoop-MapReduce: Challenges

- The current architecture of the Hadoop-MapReduce programming model does not provide a fast, scalable and distributed resource infrastructure solution. Business and scientific organisations are required to solve a wide range of data-intensive analytic problems.
- Current implementations of the Hadoop-MapReduce programming model are not able to react quickly and grow or shrink to adjust to the current workload to reduce costs while maximizing results based on the application and/or user demand.
- The current architecture of the Hadoop-MapReduce programming model does not make it easy to handle multiple application integrations on a production-scale distributed system with automated application service deployment capability.
- Current implementations of the Hadoop-MapReduce programming model are not specifically optimized to take full advantage of multi-core servers.
- Current implementations of the Hadoop-MapReduce programming model only support the Hadoop Distribution File System (HDFS). Data

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

transformation is the key challenge to translate the data into an HDFS-suitable format.

6.4 Hadoop-MapReduce: Open Issues

- Optimization technique-related issues require study under the Hadoop-MapReduce architecture.
- MapReduce programs are based on data-intensive instead of computation-intensive tasks, and, in order to achieve good performance, it is vital to minimize disk I/O and communication. Therefore, many studies seek ways to enable in-memory processing and avoid reading from disk when possible.
- Another open issue is to find the mechanism by which Hadoop could offer capabilities of tuneable fault-tolerance to users or provide automatic fault tolerance adjustment mechanisms, depending on the cluster and application characteristics.

7 DISCUSSION AND CONCLUSION

7.1 Discussion

Hadoop-MapReduce is an open-source programming software framework introduced by Google in 2004, and since then, numerous methods have been proposed to improve the performance of the Apriori algorithm on the Hadoop-MapReduce framework. According to Tables 6 and 7, researchers' perspectives can be classified as follows:

I. Horizontal vs. vertical layout analysis
Most of the researchers analysed the performance of the Apriori algorithm based on a horizontal layout. Eclat is the vertical dataset layout algorithm that is more efficient than the Apriori algorithm in the sequential environment. Sandy Moens et al. [67] proposed two new parallel algorithms based on the Eclat algorithm: Dist-Eclat and BigFIM. Dist-Eclat is a MapReduce implementation of the well-known Eclat algorithm that focuses on speed, while BigFIM is optimized to deal with truly Big Data by using a hybrid algorithm with both Apriori and Eclat on MapReduce. Rahman et al. [41] implemented a parallel Eclat algorithm on Hadoop-MapReduce. The authors checked the performance, and they achieved good results.

II. 1-phase, 2-phase, and *k*-phase analysis
Researchers find all the frequent itemsets in three different ways: 1-phase, which requires a single iteration to find all frequent itemsets; 2-phase, which only needs two MapReduce phases to extract all frequent itemsets; and *k*-phase, which needs to find all frequent itemsets in k MapReduce phases. Othman et al. compared 2-phase with the 1-phase and k-phase types. They proved that the 2-phase algorithm is much better than the 1- and *k*-phase algorithms. Additionally, 1-phase is worse than *k*-phase, as it is inefficient and slow.

III. Hadoop mode performance analysis
Hadoop can typically be configured to run in three modes: Standalone mode (on the local computer, useful for testing and debugging), pseudo-distributed mode (i.e. on an emulated "cluster" of one computer, useful for testing), and fully-distributed mode (on a full cluster, for production purposes).

- Standalone mode: this is the default mode of the Hadoop framework, i.e., it is configured to run in the non-distributed mode. None of the Hadoop daemons (NameNode, DataNode, Secondary NameNode, Job Tracker, and Task Tracker) are running. The local file system is utilized instead of the Hadoop Distribution File System (HDFS). This mode is very useful for the testing and debugging process.
- Pseudo mode: the Hadoop daemons run on a local machine, therefore simulating a cluster on a small scale. Different Hadoop daemons run in different Java virtual machine (JVM) instances but on a single machine. Here, HDFS is exploited instead of the local file system.
- Fully distributed mode: this mode consists of multi-node clusters and is exploiting the actual power of Hadoop.

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

Researchers mostly implemented their work in the standalone and fully distributed modes (see Table 9) because the pseudo-distributed mode does not add value for most purposes. Koundinya et al. [54] tested the three-mode performance with an increasing number of transactions. From their study, the fully distributed mode is better than the standalone and pseudo modes

IV. Candidate and frequent itemset analysis

Most of the Hadoop-MapReduce-based modified algorithms focused on the reduction of the number of database scans. However, the Apriori algorithm suffers when generating and storing candidate itemsets specially focusing on 2-candidate itemsets. Kovacs et al. proposed an algorithm to handle 2 itemsets in a special way (see Table 6 and 7), thereby significantly reducing the response time of the Apriori algorithm, but the drawback of this method is that more memory is needed to store the itemset in a triangular matrix (spatial complexity of $O(n^2)$)

7.2 Conclusion

Currently, data are huge and exponentially increasing every day. As datasets grow past GBs to TBs or more, it has become infeasible to manage, store, and analyse them on a single sequential machine. Parallel and distributed computing offers a potential solution for the above problem when efficient and scalable parallel and distributed algorithms can be implemented. The Hadoop-MapReduce platform is more suitable for the parallel processing of huge data on many clusters using commodity hardware. Several survey studies have done by previous researchers but in this survey mainly concentrate on the parallelized Apriori algorithm on the Hadoop-MapReduce programming software framework with different new techniques such as 1-, 2-, and k-phase iteration techniques, of data layout (horizontal and vertical), and Hadoop modes and their suitability. We have presented a review of the various proposed parallel Apriori algorithm methods on the Hadoop-MapReduce platform. We analysed the related studies in several ways, such as by objectives of the studies, main themes of the proposed methods, which datasets were used to analyse the performance of the proposed algorithms, software and hardware configurations, which Hadoop mode was used, how many phases they used, etc. Moreover, we deeply discussed different categories, such as the types of data layout (horizontal and vertical), 1-, 2-, and k-phase iteration techniques and their performance, types of Hadoop modes and their appropriateness, and how to generate more candidate itemsets. Finally, we interpreted the challenges and open issues of big data in the cloud and the Hadoop-MapReduce framework. From this study, we can ultimately conclude that the Hadoop-MapReduce platform is an efficient and scalable platform for the analysis of any big data computational problem and that the Hadoop-MapReduce-based Apriori algorithm is more efficient than the sequential version without Hadoop-MapReduce platform algorithms.

ACKNOWLEDGMENT

We wish to thank Universiti Kebangsaan Malaysia (UKM) and Ministry of Higher Education Malaysia for supporting this work through the following research grants (ERGS/1/2013/ICT07/UKM/02/3 and DIP-2014-37).

REFERENCES

[1] Adamo, J.-M. (2001). Data Mining for Association Rules and Sequential Patterns: sequential and parallel algorithms. *Springer Science & Business Media.*

[2] Aggarwal, C. C., Zhao, Y., & Yu, P. S. (2010). On clustering graph streams. In *Sdm* ,Vol.2 , pp. 478–489.

[3] Aggarwal, C., & Zhai, C. (2012). *Mining text data*. Springer Science & Business Media.

[4] Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *ACM SIGMOD Record* ,Vol. 22, pp. 207–216. ACM.

[5] Agrawal, R., & Shafer, J. C. (1996). Parallel mining of association rules. In *IEEE Transactions on Knowledge and Data Engineering* ,Vol. 8(6), pp. 962–969. Springer.

[6] Antonie, M., Coman, A., & Zaiane, O. R. (2001). Application of Data Mining Techniques for Medical Image Classification. In *Proceedings of the second international Workshop on Multimida Data Mining (MDM/KDD'2001)* pp. 94–101.

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

[7] Antonopoulos, N., & Gillam, L. (2010). *Cloud computing: Principles, systems and applications*. Springer Science & Business Media.

[8] apache. (2014). Welcome to Apache™ Hadoop®!

[9] Azarmi, B. (2014). Knowing about the Hadoop ecosystem.

[10] Bakin, S. (1999). *Adaptive regression and model selection in data mining problems*.Thesis.

[11] Becuzzi, P., Coppola, M., & Vanneschi, M. (1999). Mining of association rules in very large databases: *A structured parallel approach. In Euro-Par'99: Parallel Processing*, Vol. 1685, pp. 1441–1450. Springer.

[12] Berry, M. J., & Gordon S. Linoff. (2011). Data mining techniques-for marketing, sales and customer support. *John Wiley & Sons, Inc*.

[13] Berry, M., & Linoff, G. (1999). Mastering Data Mining: The Art and Science of Customer Relationship Management. *John Wiley & Sons, Inc*.

[14] Bialecki, a, Cafarella, M., Cutting, D., & O'Malley, O. (2005). Hadoop: a framework for running applications on large clusters built of commodity hardware. h*ttp://Lucene. Apache. Org/Hadoop, 11*.

[15] Borgelt, C., Borgelt, C., Kruse, R., & Kruse, R. (2002). Induction of Association Rules: Apriori Implementation. In *15th Conference on Computational Statistics Physica Verlag, Heidelberg, Germany 2002*, Vol. 1, pp. 1–6. Springer.

[16] Brin, S., Motwani, R., Ullman, J. D., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD Record* , Vol. 26, pp. 255–264. ACM.

[17] Brin, S., Ramkumar, G. D., & Tsur, S. (2001). Method and apparatus for dynamically counting large itemsets. Google Patents.

[18] Buhl, H. U., Röglinger, M., Moser, D.-K. F., & Heidemann, J. (2013). Big data. *Business & Information Systems Engineering,* Vol. *5*(2), pp.65–69.

[19] Burdick, D., Calimlim, M., & Gehrke, J. (2001). MAFIA: a maximal frequent itemset algorithm for transactional databases. In *Proceedings 17th International Conference on Data Engineering*, pp. 443–452. IEEE.

[20] Chen, D. C. D., Lai, C. L. C., Hu, W. H. W., Chen, W. C. W., Zhang, Y. Z. Y., & Zheng, W. Z. W. (2006). Tree partition based parallel frequent pattern mining on shared memory systems. *In Proceedings 20th IEEE International Parallel & Distributed Processing Symposium* (p. 8 pp.). IEEE.

[21] Cheung, D. W., Ng, V. T., Fu, A. W., & Fu, Y. F. Y. (1996). Efficient mining of association rules in distributed databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol.*8*(6), pp.911–922.

[22] Cheung, D., & Xiao, Y. (1998). Effect of data skewness in parallel mining of association rules. In *Research and Development in Knowledge Discovery and Data Mining* pp. 48–60. Springer.

[23] Cios, K. J., Pedrycz, W., & Swiniarsk, R. M. (1998). Data mining methods for knowledge discovery. *IEEE Transactions on Neural Networks / a Publication of the IEEE Neural Networks Council*, Vol. *9*(6), pp.1533–1534.

[24] Coenen, F., Leng, P., & Ahmed, S. (2004). Data Structure for Association Rule Mining : *IEEE Transactions on Knowledge and Data Engineering*, Vol. *16*(6), pp.1–5.

[25] Crc, H. (2009). Data Mining and. In *Knowledge Creation Diffusion Utilization* pp. 27–61. Springer.

[26] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, Vol. *51*(1), pp. 107-113.

[27] Dean, J., & Ghemawat, S. (2010). MapReduce: a flexible data processing tool. *Communications of the ACM*, Vol. *53*(1), pp. 72-77.

[28] Dillon, T., Wu, C., & Chang, E. (2010). Cloud computing: issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on AINA*, pp. 27–33. IEEE.

[29] Domzal, J. (2011). *Securing the cloud: Cloud computer security techniques and tactics (Winkler, V.; 2011) [Book reviews]*. *IEEE Communications Magazine* , Vol. 49. Elsevier.

[30] Evfimievsky, a, Srikant, R., Gehrke, J., & Agrawal, R. (2002). Privacy preserving data mining of association rules. *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Databases and Data Mining*, Vol. *16*(9), pp. 217–228.

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

[31] Ezhilvathani, A., & Raja, K. (2013). Implementation of parallel apriori algorithm on hadoop cluster. *International Journal of Computer Science and Mobile Computing*. Vol. 2(4), pp. 513-516

[32] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, Vol. *39*(11), pp. 27–34.

[33] Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud Computing and Grid Computing 360-degree compared. In *Grid Computing Environments Workshop, GCE 2008*, pp. 1–10. IEEE.

[34] Frank, J. (2000). *Data mining. Nature biotechnology* (Vol. 18 Suppl). Morgan Kaufmann.

[35] Fu, T. C. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*, *24*(1), 164–181. Ganti, V., Gehrke, J., & Ramakrishnan, R. (1999). Mining very large databases. *Computer*, Vol. *32*(8), pp. 38–45.

[36] Han, E.-H., Karypis, G., & Kumar, V. (1997). Scalable parallel data mining for association rules. *ACM SIGMOD Record*, Vol. *26*(2), pp. 277–288.

[37] Han, J., Cheng, H., Xin, D., & Yan, X. (2007). Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery*, Vol. *15*(1), pp. 55–86.

[38] Han, J., & Kamber, M. (2006). *Data Mining, Concepts and Techniques*. Morgan Kaufmann.Elsevier.

[39] Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *In ACM SIGMOD Record* , Vol. 29, pp. 1–12. ACM.

[40] Hand, D. J. (2007). Principles of data mining. *Drug Safety (Vol. 30). MIT press.*

[41] Hazarika, M., & Rahman, M. (2014). Mapreduce Based Eclat Algorithm for Association Rule Mining in Datamining : Mr _ Eclat. *International Journal of Computer Science and Engineering*, Vol. *3*(1), pp. 19–28.

[42] Hidber, C. (1999). Online association rule mining. *SIGMOD '99 Proceedings of the 1999 ACM SIGMOD international conference on Management of data* Vol. 28(2). pp.145-156. ACM.

[43] Hipp, J., Güntzer, U., & Nakhaeizadeh, G. (2000). Algorithms for association rule mining --- a general survey and comparison. *ACM SIGKDD Explorations Newsletter*, Vol. *2*(1), pp. 58–64.

[44] Holmes, A. (2012). *Hadoop in Practices*. Manning Publications Co.

[45] IBM. (2013). four-vs-big-data. http://www.ibmbigdatahub.com/infographic/four-vs-big-data

[46] Itkar, S., & Kulkarni, U. (2013). Distributed Algorithm for Frequent Pattern Mining using HadoopMap Reduce Framework. *Proceeding of International Conference on Advances in Computer Science, AETACS*, pp.15-24.

[47] Jiawei, H., & Kamber, M. (2006). Data mining: concepts and techniques. *San Francisco, CA, Itd: Morgan Kaufmann*, Vol.5.

[48] Jula, A., Sundararajan, E., & Othman, Z. (2014). Cloud computing service composition: A systematic literature review. *Expert Systems with Applications*, Vol. *41*(8), pp. 3809–3824.

[49] Jyoti, L. D., & Kiran, B. D. (2014). A Novel Methodology of Frequent Itemset Mining on Hadoop. *International Journal of Emerging Technology and Advanced Engineering*, Vol. 4(7), pp. 851–859

[50] Kaisler, S., Armour, F., Espinosa, J. A., & Money, W. (2013). Big data: Issues and challenges moving forward. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on HICSS*, pp. 995–1004 . IEEE.

[51] Kamber, M. (2012). Mining Association Rules in Large Databases. *Knowledge and Data Engineering, IEEE Transactions on*, Vol. *11*(5), pp. 798–805.

[52] Kim, J., Seo, S., Jung, D., Kim, J. S., & Huh, J. (2012). Parameter-aware I/O management for solid state disks (SSDs). *IEEE Transactions on Computers*, Vol. *61*(5), pp. 636–649.

[53] Kotsiantis, S., & Kanellopoulos, D. (2006). Association Rules Mining : A Recent Overview Basic Concepts & Basic Association Rules Algorithms. *GESTS International Transactions on Computer Science and Engineering*, Vol. *32*(1), pp. 71–82.

[54] Koundinya, A., & Sharma, K. (2012). Map/Reduce Deisgn and Implementation of Apriori Alogirthm for handling voluminous data-sets. *Advanced Computing: An International Journal*, Vol. *3*(6), pp. 29–39.

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

[55] Kovacs, F., & Illes, J. (2013). Frequent itemset mining on hadoop., *2013 IEEE 9th International Conference on Computational Cybernatics*, pp. 241–245.

[56] Kulkarni, A. P., & Khandewal, M. (2014). Survey on Hadoop and Introduction to YARN. *International Journal of Emerging Technology and Advanced Engineering*, Vol. *4*(5), pp.82–87.

[57] Lam, C. (2011). *Hadoop in Action*. Manning Publications Co.

[58] Leclerc, M. (1991). An implementation of the McEliece-cryptosystem. In *ACM SIGSAC Review* (Vol. 9, pp. 1–4). ACM. Li, H., Wang, Y., Zhang, D., Zhang, M., & Chang, E. Y. (2008). Pfp. *Proceedings of the 2008 ACM Conference on Recommender Systems - RecSys '08*, pp. 107-114.

[59] Li, J., Roy, P., Khan, S. U., Wang, L., & Bai, Y. (2012). Data Mining Using Clouds : An Experimental Implementation of Apriori over MapReduce. *12th International Conference on Scalable Computing and Communications (ScalCom).*

[60] Li, N. (2013). Parallel Implementation of Apriori Algorithm Based on MapReduce Received 22 March 2012 Accepted 13 November 2012. In *Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012 13th ACIS International Conference on*, Vol. 1, pp. 89–96. IEEE.

[61] Liu, B., Hsu, W., Ma, Y., & Ma, B. (1998). Integrating Classification and Association Rule Mining. In *Knowledge Discovery and Data Mining*, pp. 80–86.

[62] Mahafzah, B. a., Al-Badarneh, a. F., & Zakaria, M. Z. (2009). A new sampling technique for association rule mining. *Journal of Information Science*, Vol. *35*(3), pp. 358–376.

[63] Maimon, O, & Rokach, L. (2005). Data mining and knowledge discovery handbook. Springer.

[64] Manvi, S. S., & Shyam, G. K. (2014). Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. *Journal of network and computer applications, 41*, 424-440

[65] Mell, P., & Grance, T. (2011). The NIST definition of cloud computing, NIST Special Publication 800, 145.

[66] Modgi, M. P. (2014). Mining Distributed Frequent Itemset with Hadoop. *International Journal of Computer Science & Information Technologies(IJCSIT)*, Vol*5*(3), pp. 3093–3097.

[67] Moens, S., Aksehirli, E., & Goethals, B. (2013). Frequent Itemset Mining for Big Data. *Big Data, 2013 IEEE International Conference on*, pp. 111–118.

[68] Najadat, H., Shatnawi, A., & Obiedat, G. (2011). A New Perfect Hashing and Pruning Algorithm for Mining Association Rule. *Communications of the IBIMA*, pp. 1–8.

[69] Neto, J. L., Santos, A. D., Kaestner, C. a. a., & Freitas, A. a. (2000). Document Clustering and Text Summarization.Citeseer.

[70] Ngai, E. W. T., Xiu, L., & Chau, D. C. K. (2009). Application of data mining techniques in customer relationship management: A literature review and classification. *Expert Systems with Applications*, Vol. *36*(2), pp. 2592–2602.

[71] Obenshain, M. K. (2004). Application of data mining techniques to healthcare data. *Infection Control and Hospital Epidemiology : The Official Journal of the Society of Hospital Epidemiologists of America*, Vol. *25*(8), pp. 690–695.

[72] Orlando, S., Palmerini, P., & Perego, R. (2001). Enhancing the Apriori Algorithm for Frequent Set Counting. In *Data Warehousing and Knowledge Discovery* pp. 71–82. Springer.

[73] Orlando, S., Palmerini, P., Perego, R., & Silvestri, F. (2003). An efficient parallel and distributed algorithm for counting frequent sets. In *High Performance Computing for Computational Science - Vecpar 2002*, Vol. 2565, pp. 421–435. Springer.

[74] Oruganti, S., Ding, Q., & Tabrizi, N. (2013). Exploring HADOOP as a Platform for Distributed Association Rule Mining. In *FUTURE COMPUTING 2013, The Fifth International Conference on Future Computational Technologies and Applications* pp. 62–67.

[75] Ozel, S. a., & Guvenir, H. a. (2001). An algorithm for mining association rules using perfect hashing and database pruning. In *10th Turkish Symposium on Artificial Intelligence and Neural Networks*, pp. 257–264. Citeseer.

[76] Park, J. S., Chen, M. S., & Yu, P. S. (1997). Using a hash-based method with transaction trimming for mining association rules. *IEEE Transactions on Knowledge and Data Engineering*, Vol. *9*(5), pp. 813–825.

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

[77] Park, J. S., Chen, M.-S., & Yu, P. S. (1995a). *An effective hash-based algorithm for mining association rules*. *ACM SIGMOD Record*, Vol. 24(2), pp. 175-186. ACM.

[78] Park, J. S., Chen, M.-S., & Yu, P. S. (1995b). Efficient parallel data mining for association rules. In *Proceedings of the fourth international conference on Information and knowledge management - CIKM '95* pp. 31–36. ACM.

[79] Patel, A. B., Birla, M., & Nair, U. (2012). Addressing big data problem using Hadoop and Map Reduce. In *3rd Nirma University International Conference on Engineering, NUiCONE 2012*, pp. 1–5. IEEE.

[80] Paul, S., & Saravanan, V. (2008). Hash Partitioned Apriori in parallel and distributed data mining environment with dynamic data allocation approach. In *Proceedings of the International Conference on Computer Science and Information Technology, ICCSIT 2008*, pp. 481–485. IEEE.

[81] Pei, J. P. J., Han, J. H. J., Lu, H. L. H., Nishio, S. N. S., Tang, S. T. S., & Yang, D. Y. D. (2001). H-mine: hyper-structure mining of frequent patterns in large databases. In *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 441–448. IEEE.

[82] Polato, I., Ré, R., Goldman, A., & Kon, F. (2014). A comprehensive view of Hadoop research—A systematic literature review. *Journal of network and computer applications, 46*, 1-25.

[83] Qureshi, Z., & Bansal, S. (2014). Improving Apriori Algorithm to get better performance with Cloud Computing. *International Journal of Software & Hardware Research in Engineerin, Vol. 2(2),pp. 33-37*.

[84] Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00* , Vol. 29, pp. 427–438. ACM.

[85] Romero, C., & Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, Vol. *33*(1), pp. 135–146.

[86] Said, A. M., Dominic, P. D. D., & Abdullah, A. B. (2009). A comparative study of fp-growth variations. *International Journal of Computer Science and Network Security*, Vol. *9*(5), pp. 266–272.

[87] Shaw, M. J. . B. C., Subramaniam, C. ., Tan, G. W. ., & Welge, M. E. . (2001). Knowledge management and data mining for marketing. *Decision Support Systems*, *31*(1), 127–137.

[88] Sheikh, N. (2013). Implementing Analytics. *Implementing Analytics*, Vol. *46*(5), pp. 3–20.

[89] Shintani, T., & Kitsuregawa, M. (1996). Hash based parallel algorithms for mining association rules. In *Fourth International Conference on Parallel and Distributed Information Systems*, pp. 19–30. IEEE.

[90] Shintani, T., & Kitsuregawa, M. (1998). Parallel mining algorithms for generalized association rules with classification hierarchy. In *ACM SIGMOD Record*, Vol. 27, pp. 25–36. ACM.

[91] Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010a). The Hadoop distributed file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010*, pp. 1–10. IEEE.

[92] Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010b). The Hadoop distributed file system. *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010*, Vol. *11*(2007), pp. 1–14.

[93] Singh, J., & Ram, H. (2013). Improving Efficiency of Apriori Algorithm Using. *International Journal of Scientific and Research Publications*, Vol. *3*(1), pp. 1–4.

[94] Srikant, R., & Agrawal, R. (1997). Mining generalized association rules. *Future Generation Computer Systems*, Vol. *13*(2-3), pp. 161–180.

[95] Srivastava, J., Cooley, R., Deshpande, M., & Tan, P.-N. (2000). Web usage mining: Discovery and applications of usage patterns from web data. *Acm Sigkdd*, Vol. *1*(2), pp. 12–23.

[96] Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, Vol. *34*(1), pp. 1–11.

[97] Tan, P. (2007). *Introduction To Data Mining*.,Vol. 1. Pearson Addison Wesley Boston.

[98] Tang, J. (1998). Using incremental pruning to increase the efficiency of dynamic itemset counting for mining association rules. In *Proceedings of the seventh international conference on* , pp. 273–280. ACM.

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

[99] Taylor, R. C. (2010). An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. *BMC Bioinformatics*, *11 Suppl 12*(Suppl 12), S1.

[100] Thevar, R. E., & Krishnamoorthy, R. (2008). A new approach of modified transaction reduction algorithm for mining frequent itemset. In *2008 11th International Conference on Computer and Information Technology*, pp. 1–6. IEEE.

[101] Tian, W., & Zhao, Y. (2015). 1 - An Introduction to Cloud Computing. In W. Tian & Y. Zhao (Eds.), *Optimized Cloud Resource Management and Scheduling*, pp. 1–15. Boston: Morgan Kaufmann.

[102] Toivonen, H. (1996). Sampling Large Databases for Association Rules. In *Proceedings of the 22th International Conference on Very Large Data Bases*, Vol. 96, pp. 134–145.

[103] V.MuthuLakshmi, N., & Sandhya Rani, K. (2012). Privacy Preserving Association Rule Mining in Vertically Partitioned Databases. In *International Journal of Computer Applications*, Vol. 39, pp. 29–35. ACM.

[104] Vajk, I. (2013). Performance Evaluation of Apriori Algorithm on a Hadoop Cluster. *Wseas.Us*, pp. 114–121.

[105] Vavilapalli, V. K., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Shah, H. (2013). Apache Hadoop YARN. In *the 4th annual Symposium*, pp. 1–16. ACM.

[106] Velte, T., Velte, A., & Elsenpeter, R. (2009). *Cloud computing, a practical approach*. McGraw-Hill, Inc.

[107] Vinodhini, S., & Vinoth, M. (2014). Frequent Pattern Identification using Map Reduce Paradigm. *International Journal of Engineering Research & Technology,* Vol. 3(3), pp. 1763-1768.

[108] Wang, K., Tang, L., Han, J., & Liu, J. (2002). *Top Down FP-Growth for Association Rule Mining*. *Pakdd*. Springer.

[109] Whaiduzzaman, M., Sookhak, M., Gani, A., & Buyya, R. (2014). A survey on vehicular cloud computing. *Journal of network and computer applications, 40*, 325-344.

[110] White, T. (2009). *Hadoop: the definitive guide: the definitive guide*. " O'Reilly Media, Inc."

[111] Woo, J. (2012). Apriori-Map/Reduce Algorithm. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*

[112] Wu, H., Lu, Z., Pan, L., Xu, R., & Jiang, W. (2009). An improved Apriori-based algorithm for association rules mining. In *6th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2009*, Vol. 2, pp. 51–55. IEEE.

[113] Wu, X., Zhu, X., Wu, G., & Ding, W. (2014). Data mining with big data. *Knowledge and Data Engineering, IEEE Transaction on*, Vol. *26*(1), pp. 97–107.

[114] Yahya, O., Hegazy, O., & Ezat, E. (2012). An efficient implementation of Apriori algorithm based on Hadoop-Mapreduce model. *International Journal of Reviews in Computing,* Vol. 12, pp.59-67.

[115] Ye, Y., & Chiang, C. C. (2006). A parallel apriori algorithm for frequent itemsets mining. In *Proceedings - Fourth International Conference on Software Engineering Research, Management and Applications, SERA 2006* ,pp. 87–94. IEEE.

[116] Yu, H., Wen, J., Wang, H., & Jun, L. (2011). An improved Apriori algorithm based on the Boolean matrix and Hadoop. *Procedia Engineering*,Vol.15, pp. 1827-1831, CEIS 2011.

[117] Yu, H., Wen, J., Wang, H., & Li, J. (2011). An improved Apriori algorithm based on the Boolean matrix and Hadoop. In *Procedia Engineering*, Vol. 15, pp. 1827–1831. IEEE.

[118] Yu, K. M., Zhou, J., Hong, T. P., & Zhou, J. L. (2010). A load-balanced distributed parallel mining algorithm. *Expert Systems with Applications*, Vol. *37*(3), pp. 2459–2464.

[119] Yu, K. M., & Zhou, J. L. (2008). A weighted load-balancing parallel apriori algorithm for association rule mining. In *2008 IEEE International Conference on Granular Computing, GRC 2008* (pp. 756–761). IEEE.

[120] Zaiane, O. R., El-Hajj, M., & Lu, P. (2001). Fast parallel association rule mining without candidacy generation. In *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 665–668. IEEE.

[121] Zaki, M. J., & Gouda, K. (2003). Fast vertical mining using diffsets. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge*

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

discovery and data mining - KDD '03, p. 326. ACM.

[122] Zaki, M. J. M., Parthasarathy, S., Ogihara, M., Li, W., & Others. (1997). New algorithms for fast discovery of association rules. In *Kdd*, Vol. 7, pp. 283–286.

[123] Zaki, M. J., Ogihara, M., Parthasarathy, S., & Li, W. (1996). Parallel Data Mining for Association Rules on Shared-Memory Multi-Processors. *Proceedings of the 1996 ACM/IEEE Conference on Supercomputing*, Vol. *3*(1), pp. 1–29.

[124] Zaki, M. J., Parthasarathy, S., Li, W. L. W., & Ogihara, M. (1997). Evaluation of sampling for data mining of association rules. In *Proceedings Seventh International Workshop on Research Issues in Data Engineering. High Performance Database Management for Large-Scale Applications*, pp. 42–50. IEEE.

[125] Zaki, M. J., & Zaki, M. J. (1999). Parallel and Distributed Data Mining: A Survey. *IEEE Concurrency*, Vol. *7*(4), pp. 14–25.

[126] Zaki, M., Parthasarathy, S., Ogihara, M., & Li, W. (1997). Parallel algorithms for discovery of association rules. *... and Knowledge Discovery*, Vol. *373*(4), pp. 343–373.

[127] Zhan, J., Matwin, S., & Chang, L. (2007). Privacy-preserving collaborative association rule mining. *Journal of network and computer applications, 30*(3), 1216-1227.

[128] Zhang, C., & Zhang, S. (2002). *Association rule mining: models and algorithms*. *Lecture Notes in Artificial Intelligence*, Springer-Verlag.

[129] Zhao, Q., & Bhowmick, S. S. (2003). Association rule mining: A survey. *Nanyang Technological University, Singapore*.Technical Report.

[130] Zheng, Z., Kohavi, R., & Mason, L. (2001). Real world performance of association rule algorithms. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, pp. 401–406. ACM.

[131] Zikopoulos, P., Eaton, C., DeRoose, D., Deutsch, T., & Lapis, G. (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media.