GROUPING ASSOCIATION RULES

USING CLUSTERING TECHNIQUES IN BIG DATA

BY

MOHAMED ALI ALASOW

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

MAY 2019

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **MOHAMED ALI ALASOW** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee

Dr. Salahadin Adem Mohammed
(Adviser)

Dr. El-Sayed M. El-Alfy (Member)

Dr. Sajjad Mahmood (Member)

for

Dr. Hamoud Aljamaan
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

17/6/2019
Date

©Mohamed Ali Alasow
2019

I dedicate this thesis with all my love to my beloved parents for their love, support and care. I also dedicate this work, to my daughter Afnaan and her mother Mariamo, to my sister rahma for support and encouragement. Thank You.

ACKNOWLEDGMENTS

All praise be to Allah alone, and might his peace and blessing be upon his messenger Mohamed, his family and companions. First and foremost, I would like to express my deep thanks to my supervisor Dr. Salahadin Adem Mohammed for his guidance, advice and patience. Secondly, it is my pleasure to thank my committee members Dr. El-Sayed M. El-Alfy and Dr. Sajjad Mahmood for their corporation and support. Furthermore, I would like to express my grateful feeling to King Fahd University of Petroleum and Minerals for giving me a chance to complete my master degree by providing full scholarship. Also I would like to send my especial thanks to the head of the Department of Information and Computer Science (ICS) for providing necessary facilities to accomplish my research. Lastly, I wish to express my sincere gratitude to my parents, family and friends for their encouragement and moral support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THESIS ABSTRACT

NAME: Mohamed Ali Alasow

TITLE OF STUDY: GROUPING ASSOCIATION RULES USING CLUSTERING TECHNIQUES IN BIG DATA

MAJOR FIELD: Computer Science

DATE OF DEGREE: May 2019

Mining association rules between data items is essential in the discovery of knowledge hidden in datasets. There are many efficient association rules mining algorithms. The problem is with the large number of rules they often discover. Large number of rules make the discovery of knowledge very challenging because too many rules are difficult to understand, interpret or visualize. To reduce the number of discovered rules, researchers proposed approaches such as meta rules, rules pruning, rules grouping, etc.

With the advent of the era of big data, the frequency and size of big datasets is growing by the day; and thus, the discovery of hidden knowledge from these datasets is becoming essential. So far the solutions to the large number of association rules are limited to the rules generated from traditional datasets. They can't

be applied to the huge number of rules discovered from big datasets. To bridge this gap, in this thesis, we are proposing a parallel rule grouping algorithm based on MapReduce. To the best of our knowledge, this is the first solution to the problem of huge number of rules generated from big datasets. We implemented the proposed algorithm in Hadoop and conducted many experiments to study its performance. To measure the performance of the proposed algorithm, we used elpased time, speedup, sizeup, and scaleup. We used benchmark datasets up to 4GB in size. The experimental results show that the proposed algorithm have high performance.

Add Arabic Abstract Page

CHAPTER 1

INTRODUCTION

Knowledge discovery from databases (KDD) is a non-trivial, nine step process of extracting valid and potentially useful knowledge from datasets. Data mining is a major step in KDD and refers to the techniques used to discover useful patterns hidden in datasets [7]. Association rule mining is a branch of data mining used to discover frequent patterns, associations, correlations, and other relationships between data items of a dataset [8]. In other words, for a given dataset, association rule mining aims to find the rules which enable us to predict the occurrence of a specific data item based on the occurrences of the other data items in the dataset.

An example of an association rule would be "If a customer buys bread, he is 75% likely to also buy milk" . An association rule (AR) is of the form $X \rightarrow Y$, where X and Y are sets of data items in the same dataset and $X \cap Y = \emptyset$. In a rule, the set of data items to the left of the arrow is called the antecedent and the one to the right of the arrow is called the consequent. For example, in the rule $X \rightarrow Y$, the antecedent is X and the consequent is Y. Not every discovered association

rule is interesting or strong. Strong association rules are mainly identified by two parameters, namely, *support* and $confidence$. A rule is considered strong, if its support and confidence are above pre-specified thresholds.

Association rule mining has many applications in marketing, medicine, finance, security, weather, bio-informatics, airline information systems, and many other fields [9]. Specially, it is extensively used in market basket analysis, stock market prediction, customer-behavior trend analysis, cross marketing, catalog design and advertising [8]. Due to the above mentioned applications and many others, it has attracted many researchers to propose different association rule mining algorithms that extract rules from datasets [10].

Often, the number of strong rules discovered by association rule mining algorithms is large, even from a moderately sized dataset. Large number of rules are difficult to inspect, analyze, visualize, or interpret, which reduces the usefulness of the discovered rules in decision making and other applications [11]. The number of rules can be reduced by using higher *support* and *confidence* thresholds but that will exclude some strong rules. To address the problem of large number of rules, different solutions have been proposed in the literature [10, 11, 12, 13, 14, 15]. Some of the proposed solutions include constraint-based mining, meta-rules, rules pruning, and rules grouping or clustering.

1.1 Problem Statement

The huge volume of data that is generated daily has motivated database researchers to propose many types of NoSQL and NewSQL database management systems. The number of such databases is growing very quickly. At the time of writing this thesis, there were more than 225 NoSQL and NewSQL database management systems [16]. Some of the most popular NoSQL database management systems are MongoDB, Cassandra, Redis, and HBase; and some of the top NewSQL databases are CockroachDB, VoltDB, ClustrixDB, and MemSQL. Many of these database management systems have currently been used by many companies to handle big datasets. The need to discover knowledge from this big data databases has attracted many researchers to propose knowledge discovery algorithms [17]. Some researches proposed modifications to the traditional knowledge discovery methods to handle big data whereas others proposed new ones. Similarly, for mining association rules from big datasets, a number of researches proposed new algorithms [18]. But no one has proposed a solution to deal with the huge number of rules generated from these new algorithms. The only available solutions are for rules generated from traditional datasets. Such solutions include rules pruning, rules grouping or rules clustering, and others. However, these traditional solutions cannot be used to prune,group,cluster or summarize the huge volume of rules discovered from big datasets. In this thesis, we are proposing the first association rules clustering algorithm that can handle huge volume of rules discovered from big datasets using Hadoop and MapReduce.

1.2 Motivation

We are in the era of big data. The volume, variety, and velocity of data generated every second by individuals, businesses and others is unprecedented and is growing by the day. As mentioned above, a large number of discovered rules is very challenging to process and benefit from. The number of association rules discovered from traditional datasets is often large, hence it is reasonable to assume that those extracted from big data will be much larger. The existing solutions to the problem of large number of rules are limited to those extracted from traditional datasets and they can't be applied to those extracted from big datasets [17]. Although there are solutions for mining association rules in big data, eg. [19, 20] to the best of our knowledge there is no work done for parallelizing the grouping of rules generated from big datasets. This motivated us to propose, in this thesis, the first solution to the huge number of association rules generated from big data using Hadoop and MapReduce.

1.3 Thesis Objectives and Contributions

As mentioned above, existing solutions to the problem of large number of association rules are limited to those discovered from traditional datasets. No researcher has proposed a solution to the challenges associated with the huge volume of association rules discovered from big datasets. Hence, the objective of this thesis is to propose a solution to this problem.

The main contributions of this thesis are:

- Comprehensive literature survey of existing solutions to the problem of large number of association rules.
- A new sequential algorithm which prunes strong association rules based on their structure.
- A new MapReduce algorithm which prunes the discovered strong association rules based on their structure.
- A new MapReduce algorithm which clusters the discovered strong association rules based on their lift values.
- Conducting comparison and performance analysis of the proposed algorithms.
- Publish and share our findings to the research community.

1.4 Methodology

We have followed the following phases in our methodology.

Phase 1: Conducting literature review

In the first phase, we extensively studied existing rule pruning, rule grouping, and rule clustering algorithms. We identified the limitations and strengths of each algorithm. This phase also enabled us to clearly state the scope and the contributions of the thesis.

Phase 2: Proposal of new algorithms

In the second phase, we proposed two new association rule pruning algorithms, one sequential and another parallel. We also proposed a new association rule clustering algorithm.

Phase 3: Setting up hadoop cluster

In this phase, we installed and configured a Hadoop cluster that we used to implement and run the proposed parallel algorithms. We also prepared the benchmark datasets used in our experiments.

Phase 4: Implementation of the proposed algorithms

Here, the proposed sequential and parallel algorithms were implemented using Java and Python respectively.

Phase 5: Evaluation of the proposed algorithms

During this phase, the proposed algorithms were experimented using benchmark datasets. From the collected experimental results, the performances of the proposed algorithms were analyzed and their strengths and limitations were identified.

Phase 6: Drawing conclusions and suggesting some future directions

At the end, conclusions from the research work were drawn and future research directions were highlighted for clustering association rules.

Phase 7: Thesis write up

The thesis writing process was done during all mentioned phases.

1.5 Thesis Organization

The rest of this thesis is organized as the follows. Chapter 2 presents basic terminology and background information on association rule mining and big data. We reviewed related works in Chapter 3. The proposed structure based pruning algorithms are explained in Chapter 4; and the lift based clustering algorithm is explained in Chapter 5. Chapter 6 presents performance analysis and comparisons of the proposed solutions. Finally, Chapter 7 concludes our thesis work and suggests some future directions.

CHAPTER 2

BACKGROUND

This chapter gives background information necessary to understand the ideas presented in this thesis. It starts by a brief discussion of data mining in Section 2.1. The concepts behind mining association rules, which is the main topic of this thesis, are discussed in Section 2.2. Section 2.3 gives a general overview of big data and Hadoop.

2.1 Data Mining Overview

This section gives a brief explanation of knowledge discovery and data mining. It then discusses association rules, association rule measures, and association rule mining with some examples of association rule mining applications.

Knowledge discovery from datasets (KDD) is a multi-step process of extracting valid and potentially useful knowledge from datasets. This process is illustrated in Figure 2.1. Data mining appeared in the 1990s as a step in KDD, and refers to the sophisticated tools and techniques used to discover useful but previously

unknown information hidden in large datasets [7, 21]. It is an interdisciplinary field of Computer Science which involves machine learning, statistics, pattern recognition, algorithms, databases and many others. Initially, the main goal of using data mining was limited to create operational reports; later, when new machine learning algorithms were developed, users started using it for knowledge discovery and decision-making. Through the data mining process, different kinds of knowledge, such as, classification rules, association rules, clustering and others can be discovered [8].

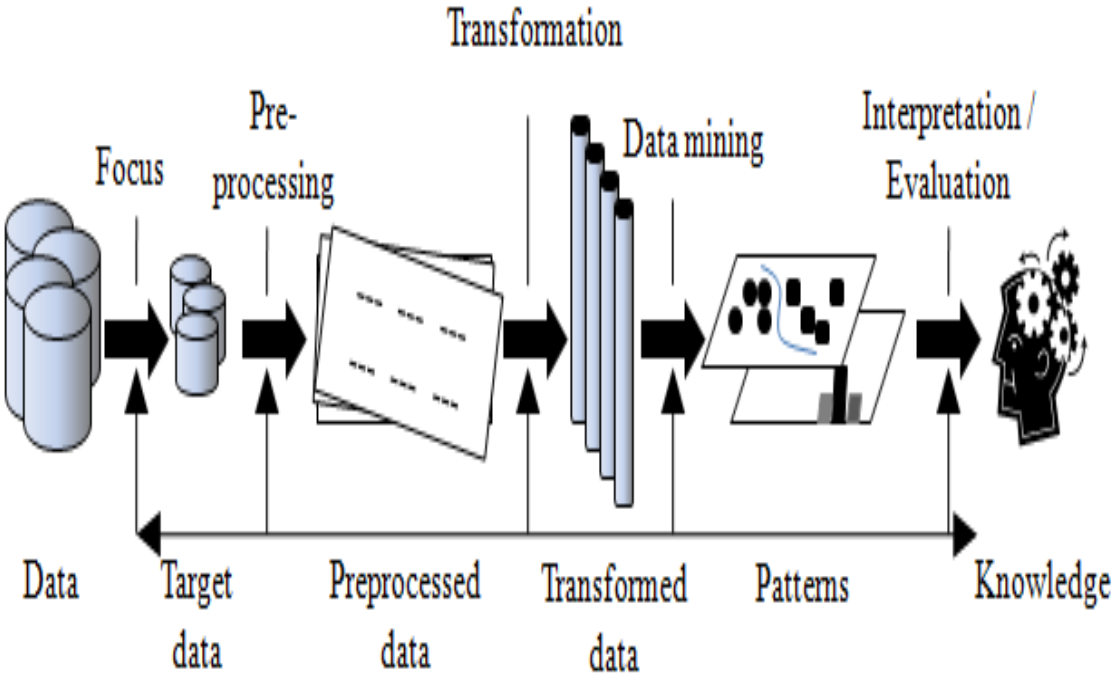

Figure 2.1: KDD steps[1]

2.2 Association Rules

The recording of very detailed data about customer's purchases in many supermarkets and other business centers spurred the need to further analyze and find

valuable information about the items frequently purchased together. This information helps businesses to understand customer behavior, design catalogs, and tune promotions. Association rule mining can help obtain such information in the form of if-then statements [11]. Association rule mining is a sub-field of data mining that aims to extract hidden knowledge from datasets in the form of frequent patterns, correlations, associations or causal structures [22]. Formally, Agrawal et al. [11] formulated the standard base line for the problem of association rules as follows: Let $I = \{I_1, I_2, \ldots, I_n\}$ be a set of items in a transactional database D and $|D|$ be the number of transactions in D. A transaction T in D contains a set of n items such that $T \subseteq I$. Let X and Y be two sets of items in D. Formally association rule is described in implication of $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \oslash$. X is named as antecedent and Y as consequent of the rule.

2.2.1 Strong Association Rules

Often, association rule mining algorithms discover many association rules. Not all discovered association rules are interesting or strong. To identify strong association rules, two types of measures, namely, subjective and objectives measures are used [23, 23, 24]. The objective measures (data-driven measures) are quantitative in nature and they rely mainly on statistical aspects of data, whereas the subjective measures (user-driven measures) rely on user-beliefs. In practice, both types of measures are used for finding strong rules [25]. In the literature, there are many objective measures but the three most popular measures are *support*, *confi-*

dence, and *lift*. There are some uncommon objective measures in literature such as *improvement*, *validity*, and $influence$ [26]. However, we only use support, confidence and lift which are intensively used in the literature and are explained more precisely below.

Support, denoted as S, measures the generality of a given association rule in a given dataset. In a transaction database D, the *support* of the rule $X \to Y$ is:

$$S\big(x \to Y\big) = \frac{|X \cup Y|}{|D|} \tag{2.1}$$

where $|X \cup Y|$ is the number of transactions in D which contain both X and Y , and $|D|$ is the total number of transactions in D. The confidence, denoted as C, of $X \to Y$ is:

$$C\big(X \to Y\big) = \frac{|X \cup Y|}{|X|} \tag{2.2}$$

In other words, $C\big(X \to Y\big)$ is the proportion of transactions containing X that also contain Y.

Support and confidence measures fail to filter out all uninteresting association rules. To address this deficiency, data mining scholars used another measure called $lift$ [26]. Lift computes the correlation between the consequent and antecedent of a rule and can be used to filter out more uninteresting rules. The advantage of lift is that it does not pose the downward closure or the problem of rare itemsets. Although the lift measure does not have the problem of rare itemsets, but still it has some drawbacks especially when filtering out some rules which are made up

of highly frequent itemsets. The lift of $X \rightarrow Y$ is:

$$Lift\big(X \rightarrow Y\big) = \frac{S\big(X \cup Y\big)}{S\big(X\big) * S\big(Y\big)} \tag{2.3}$$

According to the above mentioned objective measures, an association rule is strong if its *support*, *confidence*, and *lift* are above pre-specified thresholds (to be set by the decision maker).

Unlike objective measures, subjective measures embody the subjective factors which allow the integration of user participation in association rule evaluation. The rule evaluation in the subjective-based measures is based on some subjective emulation criteria such as simplicity, novelty, availability, unexpectedness, actionability, and so on. For example, novelty is one of the subjective indicators and it can be used for evaluation of quality and the similarity of discovered rules [27]. The novelty of given association rules can be measured by calculating the difference between every item of antecedent and that of the consequent. The novelty of a rule reflects the difference associated between rules of the knowledge base and discovered rules. The key concept behind novelty is that it increases the user knowledge by proving novel information about previously unknown interesting rules.

2.2.2 Frequent Itemsets

In a transactional database, a transaction is a set of items (itemset). For instance, in market basket analysis, an item can be sugar, milk, bread or others depending

on the database. A $k-itemset$ is a set of k items. The *supportscount* (frequency) of an itemset is n if it appears in n transactions of a given database. Mining association rules in a transactional database can be achieved in three main steps [28]. In the first step, all the frequent itemsets in a given database are extracted. An itemset is frequent, if it appears a pre-specified minimum number of times in the dataset (refered to as minsup). In the second step, all possible association rules are generated from each frequent itemset. In the last step, strong association rules are identified.

The number of frequent itemsets generated from a dataset can be large. However they can be compactly represented using two commonly used sets, namely, **closed itemset** and **maximal frequent itemset**. An itemset is closed, if its *support count* is higher than that of all its supersets [29]. An itemset X is superset of an itemset Y if all items of Y are in X. We can also say Y is a subset of X. From the *support count* of a closed itemset, the *support counts* of some of its subset itemsets can be generated [29, 30, 31, 3]. A maximal frequent itemset is a frequent itemset, and none of its supersets are frequent [32, 33, 34, 35, 36]. As shown in Figure 2.2, a maximal frequent itemset is a subset of closed frequent itemset.

2.2.3 Association Rule Mining Algorithms

Association rules are discovered using association rule mining algorithms. Basically, most association rule mining algorithms are quite similar and have many

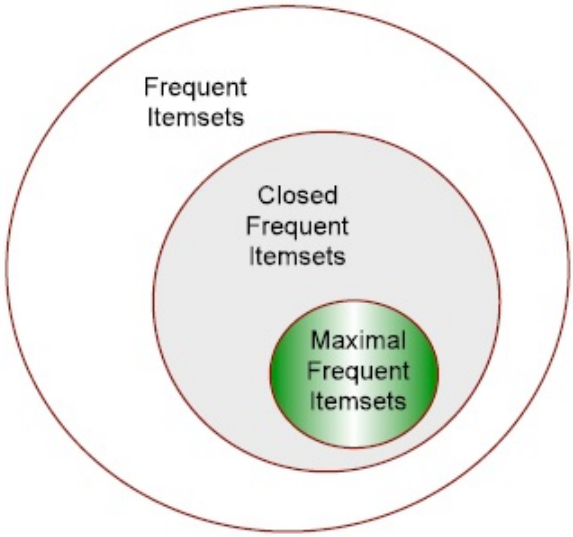

Figure 2.2: Closed and maximal frequent itemsets. [2]

common functionalities, however regarding to application-specific requirements, their implementation features could differ. Some algorithms add certain improvements and extra features to deal with specific application dependent requirements. Initially association rule mining algorithms emerged during the time of centralized database systems where sequential processing methods were the first choice in most applications. The process of applying association rule mining algorithms primarily comes after or while the step of generating frequent itemsets. Different algorithms follow different strategies to scan the dataset to extract the frequent itemsets. Commonly, all algorithms try to decrease complexity by reducing the number of database scans and by efficiently handling candidate itemsets. The two most commonly used association rule mining algorithms are Apriori and FP-growth.

Apriori Algorithm

The Apriori algorithm goes through many database scans to generate frequent itemsets [37]. First, it scans the database to extract frequent 1-itemsets. It then combines the frequent 1-itemsets to generate candidate 2-itemsets. It scans the database again to check which of the candidates 2-itemsets are frequent. It combines the frequent 2-itemsets to generate candidate 3-itemsets. It scans the database for the third time to find which of the candidate 3-itemsets are frequent and so on until it finds all the frequent itemsets. The number of database scans depends on the size of the largest frequent itemset. At last, Apriori generates rules from the frequent itemsets. Figure 2.3 shows an illustrative example of Apriori algorithm.

FP-growth Algorithm

The Frequent Pattern (FP) algorithm scans the database twice [38]. In the first scan, it finds the support-count of each 1-itemset. In the second scan, it builds a tree, FP-tree, out of all the transactions in the database as illustrated Figure 2.4. Each node of the tree represents an item. The root node is represented by a null item. The items in a transaction form a branch in the tree. The items in a branch are sorted in descending order of their support-count starting from the root-node. This sorting is done to minimize the size of the tree. Each node also keeps a count which represents the number of transactions which contain the node and the nodes above it in the branch. After the tree is built out of all the

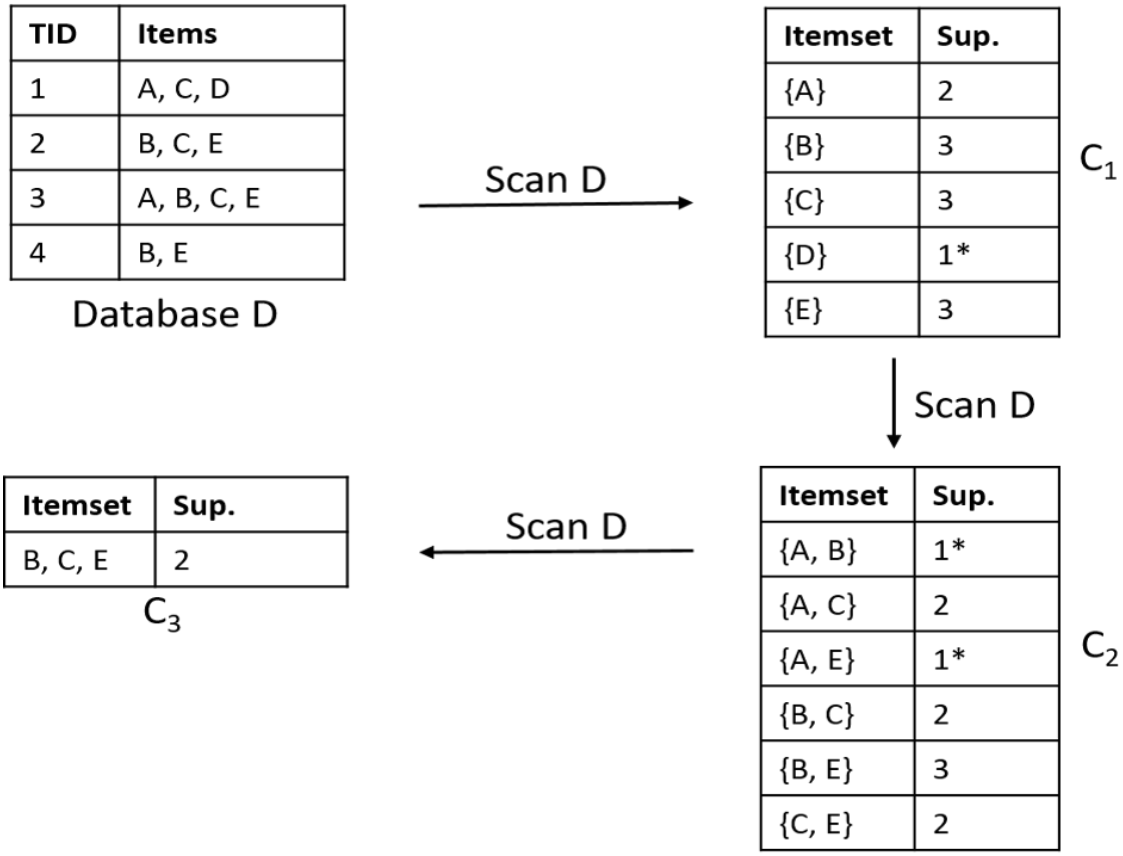

Figure 2.3: Steps of Apriori algorithm when minsup = 2; itemsets with * are non-frequent itemsets [3]

transactions in the database, the FP-growth algorithm systematically traverses the tree to generate the frequent itemsets. The FP-growth algorithms is faster than Apriori. Unlike Apriori, it doesn't generate candidate itemsets and it doesn't scan the database more than two times.

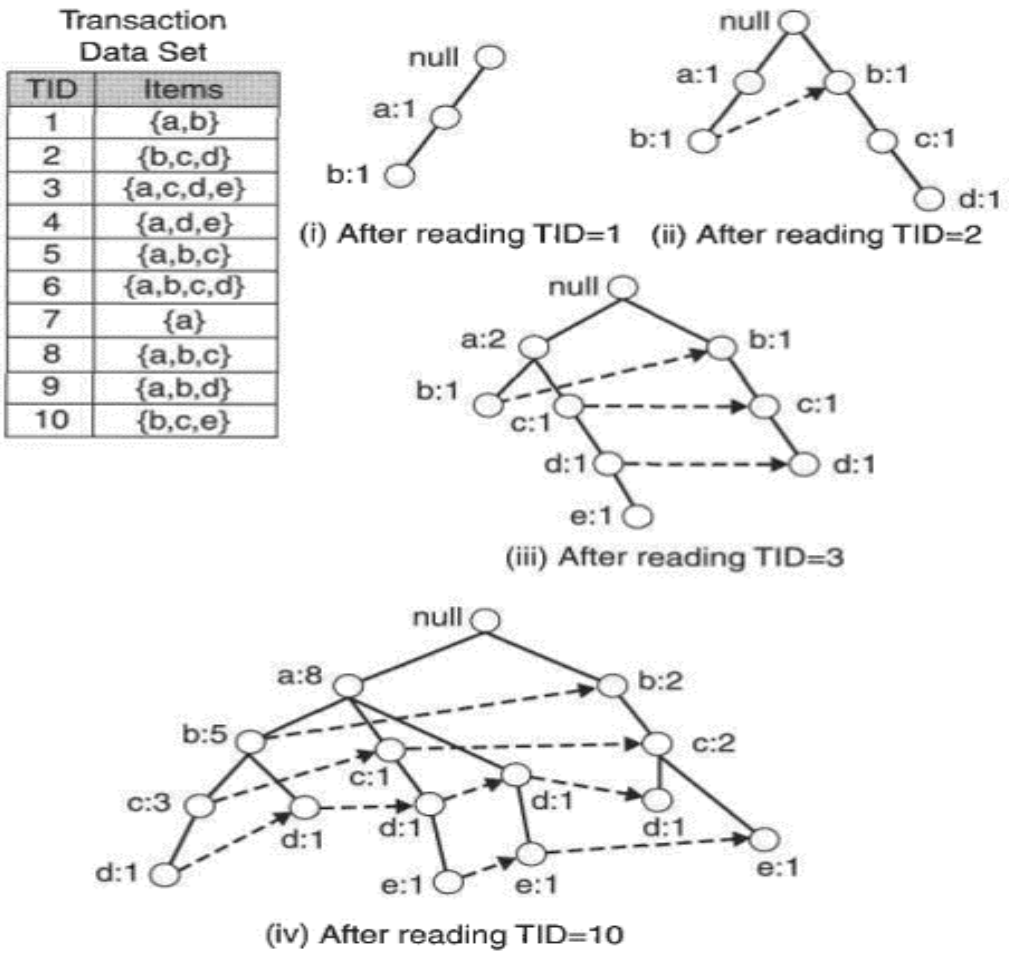

Figure 2.4: Construction of FP-tree [4]

With the advancement of technology, data is generated with huge velocity, variety and volume than ever before by various sources, such as sensors, social networks, global business, Internet of Things (IoT) and so on. Existing sequential rule mining approaches, such as Apriori and FP-growth, can not be applied to process such data; and thus parallel versions of Apriori and FP-growth algorithms have been proposed by many researchers [18, 39].

2.2.4 Association Rule Mining Applications

Beside the applications mentioned in Chapter 1, association rule mining techniques are also used to analyze data stored in Health Information Systems collected from various operations, such as, medical diagnostics, medical administrative tasks, and medical financial information, and so on [40]. Association rules mining is also used in information retrieval systems. Xin Li et al. [41] implemented association rule mining to extract key phrases that are useful for capturing saved contents in a document for retrieval purpose. They grouped frequencies that have strong co-occurrence in order to reduce redundancy in the obtained result. By adopting these two approaches they achieved higher performance of information retrieval.

2.3 BigData Overview

The past few years witnessed several technological advances in social networks, IoT, data storage, sensors, computing systems, such as, cloud computing, communication systems, financial and medical systems and others. This has consequently resulted in the era of big data analytic. It is in a new era of revolution in data analytics and data management that expands across several scientific fields. Although there is no consensus on the definition of big data [42], it is commonly used to describe data that cannot be stored or processed using the traditional computing approaches within a given time frame. In 2014, Dobre and Xhafa [43] reported that every day the world generates around 2.5 quintillion bytes of data.

Now in 2018, the amount of data generated is significantly more. Over the last two years alone, 90% of the data in the world was generated; and with the Internet of Things, the pace will increase dramatically. According to Gantz and Reinsel [44], by 2020, 40 zettabytes will be generated of which 90% is unstructured data. We are definitely in the era of big data; and thus, big data processing has considerably attracted a plethora of research efforts to address challenges pertaining to its 3V characteristics (volume, variety, and velocity) in many applications, Figure 2.5. We refer the reader to the work in [45] that provides a consolidated description of big data and analytic methods used for big data.

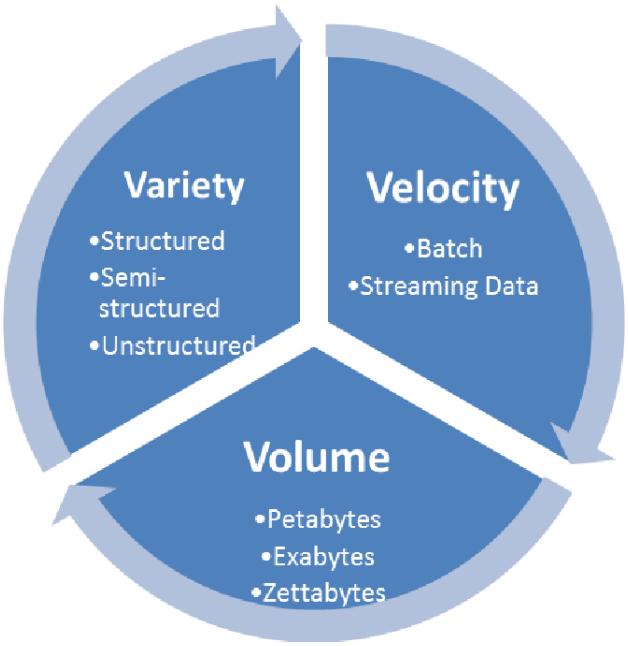

Figure 2.5: Big data characteristics [5]

2.3.1 Hadoop

The era of big data raised challenges on big companies, such as, Google, Facebook and Amazon [46]. Everyday they need to process petabytes of structured, semi-

structured and unstructured data to meet consumer demands. Traditional applications are incapable of processing big data. Fortunetly, Hadoop came in time as a solution to big data problems. Hadoop is a set of open-source software utilities that facilitate using clusters of many computers to solve problems involving massive amount of data and computation [47]. It provides a software framework for distributed storage and processing of big data using a simple programming model. Initially, Hadoop was developed by Apache Software Foundation using Java programming as a solution to data intensive distributed applications. Hadoop now is in its third version, Hadoop 3. The rest of this section discusses the Hadoop 2.8.1 but for more coverage of all Hadoop versions, we refer the reader to [48]. Hadoop 2.8.1 has two core components, namely, HDFS and MapReduce. HDFS is a distributed file system and MapReduce is a programming paradigm that enables big data to be processed across thousands of nodes in a Hadoop cluster. Figure 2.6 shows a simple Hadoop 2.8.1 architecture.

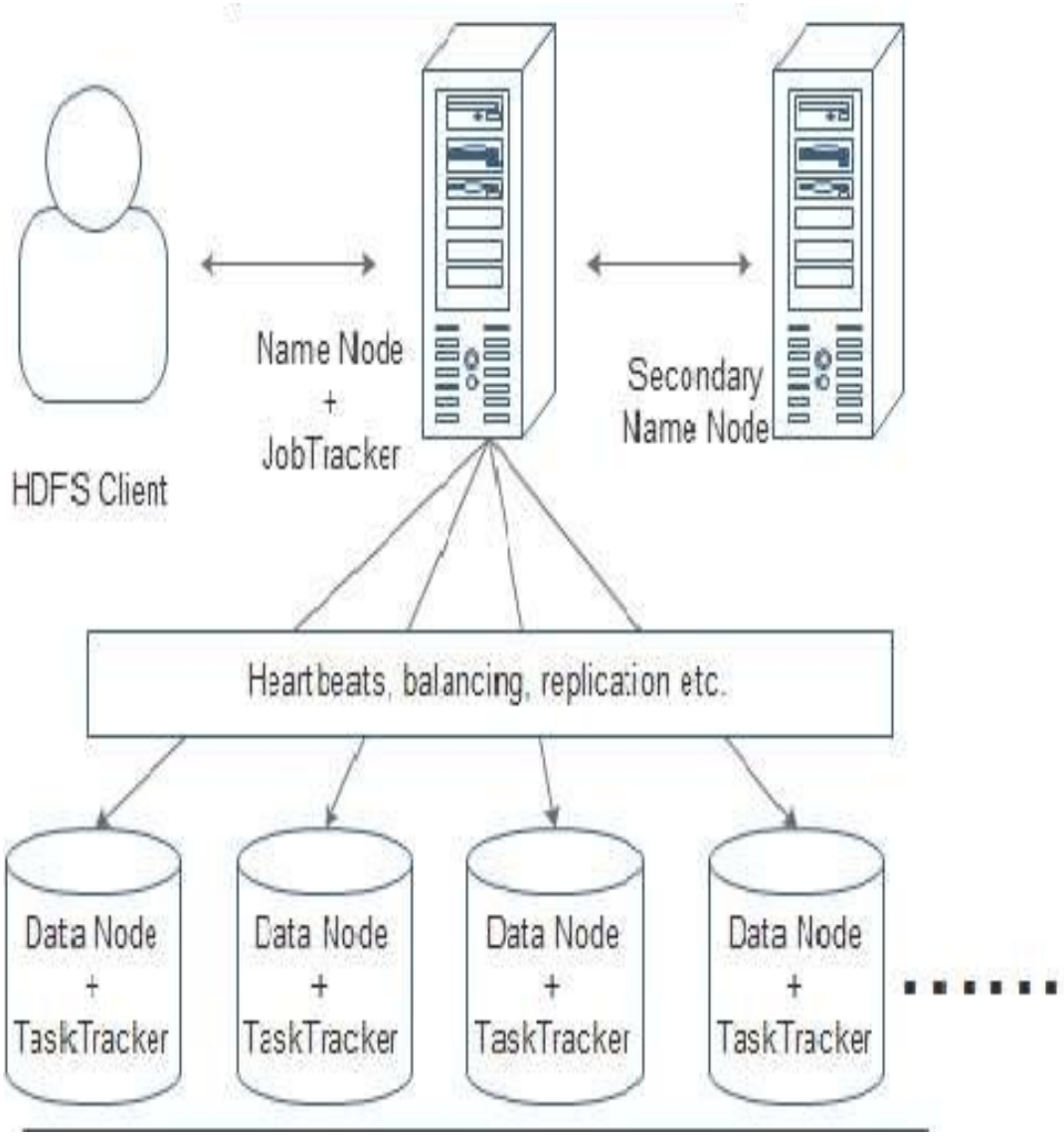

Figure 2.6: A simple Hadoop architecture [6]

HDFS

HDFS is a distributed block-structured file system where each file is divided into 128 MB blocks and stored across multiple nodes [46, 49]. HDFS automatically optimizes high bandwidth streaming and deals with all kind of data formats, such as, text, images, videos etc regardless of underlying architecture. To avoid loss of data and a single point of failure, HDFS replicates each file block in at least two nodes. HDFS is based on Master/Slave architecture. It has one master node and several slave nodes. HDFS has three main daemons, namely, name-node, secondary-name-node, and data-node. The name-node runs in the master-node to manage the distribution of file partitions across Hadoop nodes. It frequently checks the status of each data-node and instructs it what to do. Each data-node runs in a single slave-node and it manages the storage of file blocks in that node. It frequently updates the name-node about its file blocks. The secondary-name-node is used as a fast backup of name-node meta-data.

MapReduce

The term "MapReduce" actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples of the form key-value pairs. The second is the reduce job, which takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies,

the reduce job is always performed after the map job [50]. Practically it is not necessary that mapper and reducer interact sequentially. A reducer can start its processing as any of mappers completes its task without waiting for the completion of remaining mappers. The Hadoop framework takes the role of parallelization, scheduling and distribution services. Also, before a reducer receives the mapped data from a mapper, Hadoop framework performs some intermediate operations such as sorting and shuffling. During map and reduce phases, the input, output, and intermediate data are stored in the file system.

Like HDFS, MapReduce utilizes Master/Slave architecture [46]. It uses two other daemons, namely Job-Tracker and Task-Tracker. The Job-Tracker runs on the master-node and is responsible for various activities, such as, scheduling and dividing jobs into the map and reduce tasks. It also monitors the running of Task-Tracker on slave-nodes. Periodically the Task-Tracker sends heartbeat messages to the Job-Tracker to indicate that its alive and executing its assigned task. Within a stipulated time frame, if the Job-Tracker does not receive the expected heartbeat message from a Task-Tracker, it will consider it dead and schedule that specific job to another available Task-Tracker. In contrast to Task-Tracker, if the Job-Tracker goes down, all executing jobs will be stopped. This limitation is now solved in the new versions of Hadoop. The Task-Tracker runs on each slave-node of a cluster. Based on the capacity, each Task-Tracker is assigned a limited number of tasks by the Job-Tracker. The Job-Tracker monitors the number of tasks assigned to each Task-Tracker running on a particular slave-node through the heartbeat protocol.

CHAPTER 3

LITERATURE REVIEW

Several approaches have been proposed in the literature to solve the problem of large number of association rules. In this chapter we will briefly discuss each proposed approach.

Aijun et al. [10] proposed two domain knowledge-based methods to prune and summarize the generated strong association rules. In the first method, to group rules, the user defines the semantic distance between rules using data taxonomy. The second method groups all rules that share an item in the antecedent and consequent. The idea of domain ontologies is introduced to generalize association rules. This concept facilitates the representation of mined rules in the form of is-a hierarchy. Marinica et al. [51] proposed a post-processing method to prune and filter association rules. They integrated user knowledge modeled in the form of ontology connected to the data to improve the selection of interesting rules.

Martine Cadot and Alain Lelu [52] carried masive prunning to generate operational set of association rules. They proposed four principle meta-rule concept to

eliminate conflicting and redundant rules. The proposed method allows the users to combine rules for reasoning using common sense logic.

Liu et al. [53], Padmanabhan and Tuzhilin [54], and Silberschatz and Tuzhilin [55] employed user domain knowledge to find unexpected rules. Torvonin et al. [14] used a rule cover method to differentiate redundant rules from non-redundant. Brijs et al. [56] enhanced the rule cover method using integer programming techniques to maximize rule redundancy reduction amount in the generated rules. Although these approaches help to prune interesting rules, they are depending on domain knowledge to give users the ability to view interesting rules and discarding the uninteresting rules. Unlike domain knowledge-based approaches various subjective and objective interesting measures are introduced to segregate all interesting rules from the rest of rules based on a given measure .

Bayardo et al. [57] came up with minimum improvement concept to prune interesting rules. They measured the difference between the confidence of a rule and that of its proper sub-rules to perform the rule pruning action. In this method, choosing a low-minimum improvement threshold can loose sensitivity to catch many overlapping rules.

Bay and Pazzani [58] considered contrast sets, that contain the conjunction of meaningfully different attributes and values across their distribution in the set of classes or groups of interest, to prune and remove insignificant contrast rules. Huang and Webb [59] anatomically discarded the discovered insignificant rules in the discretized quantitative attributes using search algorithm called OPUS.

Liu et al [53] utilized the popular statistical test, chi-squared, as a base way to prune the rules. He evaluated the dependence of antecedent and consequent of a rule by measuring $\tilde{\chi}^2$ with respect to the whole data. To do pruning of the rule result set, a pre-specified $\tilde{\chi}^2$ at the significance level c is compared with computed $\tilde{\chi}^2$ -test of rule. In case chi-squared test value is 0 it indicates that the the attributes are statistically independent. Using statistical chi-squared test suffered from the issue of data sparsity to prune many rules. Liu et al. [53] enhanced the approach by summarizing for the mined rules into a special set called the direction setting rules. This set can preserve the general relationship in the domain.

Pruning can also be achieved using directed hypergraph and that is what Chawla et al. [60] did through an adaptive local pruning method. Deducing the structural strength of graph methods, association rules networks are represented in the form of a graph and rule pruning becomes much easier to generate from the graph nodes. The process of pruning in this approache is applied locally to keep the importance of non-local rules separate from pruning process. However, this method fails to maintain the global picture of association rules.

Unlike pre-processing algorithms, Huawen Liu et al. [61] proposed a post-analyze approach that facilitates eliminating redundancy rules generated in the association mining step using Galois Connection theory. The advantage of this method is that it avoids information loss in the pruning process. The comparison result showed that the computational cost of proposed method is much less than the well-known Apriori method.

There are couple of measures that are practiced to filter and obtain only interesting association rules based on given threshold. The following approaches use interesting measures to discover, from the large association rules, only interesting rules. H. Yun et al. [62] presented a technique that uses relative support value to discover rare data that tends to appear infrequent but are highly associated. G. Hrovat et al. [63] formed interesting measures from analyzing time series data over a long period of training. These interesting measures are used for sequential patterns. To enhance the rule quality in the frequency mining association rules Hyeoncheol Kim et al. [64] suggest a measure named surprisal that before association rule generation eliminates noisy data that can degrade the quality of the rule result. The capability of proposed pruning method is tested on question-response datasets and the result from the experiment shows this method produces good quality association rules. Solving the problem of single based support in association rule mining domain Hu. Ya-Han et al. [65] presented the concept of multiple minimum supports(MMS), the proposed method allows users to select multiple different nature items. They extended this algorithm to discover interesting sequential patterns using multiple minimum support values. The proposed method is compared with three traditional data mining algorithms named GSP, MSCP-growth, and PLWAPtree on several data sets.

Tayal et al. [66] used a fuzzy based algorithm for mining association rules for predicting diseases like cancer. In this method, after generating candidate itemsets, $K - means$clustering algorithm is employed to group itemsets around

mid-points. From given centroids fuzzy membership is calculated. Finally using fuzzy support and confidence value fuzzy rules are generated. This method reveals good prediction result compared to conventional method.

Fernandes et al. [67] presented a visualization technique to be used as a pre-processing step in the mining phase based on dual scale metric. This pre-processing technique leads to dimensionality reduction by allowing users to remove items from dataset without breaking the quality of extracted association rules.

Grouping and summarizing techniques are presented by An et al. [68] to solve the problem of very large number of association rules that are difficulty to manage and analyze easily. The first algorithm does grouping based on the structure of rules and creates a group of clusters recursively. The second algorithm uses the semantic tree-structure of items to group rules by their semantic distance. The evaluation result shows that the two algorithms effectively reduce the large number of association rules by grouping them.

Many researchers clustered rules based on some distance functions. Toivonen et al. [14] assumed that the distance of association rules can be estimated in term of the number of rules that are different in the overall rules to be grouped. Gupta et al. [69] improved distance metric method by normalization to cluster the rules. In contrast to that, Lent et al. [15] argued that it is possible to utilize geometric properties to cluster association rules. This approach is restricted to antecedents with only two attributes. Gupta et al. estimated the distance between two rules using conditional probability [69]. They use a special case of

the Agglomerative Chain algorithm named Dimensionless Agglomerative Chain Clustering to cluster association rules. the proposed algorithm allows association rule clustering without having dimensional information. Moreover, they utilize Self Organizing Feature Map (SOM) to cluster points into a Euclidean space and apply visualization technique.

Bo Li et al. [70] suggested alternative association rule method that can classify transaction databases using CFSFDP clustering approach in order to extract useful association rules. This technique allows a parameterized selection using predefined interesting measures.

To mine quantitative association rules of the high-dimensional telemetry data, recorded for satellite performance analysis, Xin Dong et al. [71] proposed a new improved APRIORI algorithm called QARC Apriori with partitioning. In order to filter and produce more meaningful and concise result. Two pruning techniques are employed after rule generating step. The proposed algorithm is suitable for quantitative association rule mining.

Aswani Kumar [72] applied $K - means$ clustering algorithm, before doing association rule mining on the data set, to reduce the large number of formal context outcome resulted from formal concept analysis mining process. S. Bedi et al. [73] presented two clustering techniques, Association Rule Hypergraph Partitioning and Principal Component Partitioning. They utilize the graph theory to atomically discover association rules and document their similarities without prompting any distance information. The two algorithms show that they are

effective even in occasions where the dimensionality degree is high.

Quan et al. [74] proposed a new technique called Formal Concept Analyze (FCA) to mine a special kind of association rules named concepts association rules, which covey more semantic information than traditional association rules. A clustering-based algorithm with distance metric is used to regionally group the similar conceptual association rules in this method. Koh et al. [75] generated rare association rules from database transaction records. He adopted clustering step, prior to mining process, in order to classify and express solid groups that form different patterns. The obtained result in this method is more informative compared to non-clustered data sets.

To reduce the significant overhead of finding association rules, Liu et al. [76] suggested to use clustering to minimize the required number of database scans, by partition all similar transactions under one group before looking for any association in the data set. They perform summarizing of clustered data to get more manageable result. The proposed algorithm requires only one pass of database scanning.

Focusing on the field of binary data analyze Francesco Palumbo et al. [77] illustrated a combined method of clustering and dimension reduction that can identify the association of non-trivial structures hidden in binary data. The proposed method gives analytical result and allows graphical representation of the result. The drawback of this method is the possibility of losing small information from the expected result.

Visualization based techniques such as parallel coordinate plot [78] and matrix-based visualization [79] are introduced as post-processing techniques to analyze the discovered association rules. These techniques help to visualize the interrelations between association rule categories in a great detail. Unfortunately, the most of visualization techniques cant display large sets of rules.

Classification approaches have been practiced to classify association rules based on classification classes (CBA). Liu et al. [80] created a class of attribute constraint to control the consequent of rules before generating from association rules. Similarly [80] introduced a novel method, to reduce the generated association classes, using rule consequent constraint based on (NECR-tree). Han et al [81] suggested an efficient classification method CMAR that classifies association rules. CMAR finds rules from prefix tree structure like CR-tree CMAR using FP-growth, and that leads finding more specific and lower level rules than CBA. In contrast to CBA, CMAR measures the weighted analyze of multiple rules to determine a class label which permits to have classification.

Omer M. Soysal [82] discovered mostly associated pattern (MAPS) from structured data by using heuristic approach. This method generates patterns, without considering all combination of an item, by using maximal association constraint. Unlike most of the data mining techniques, this method does not require pruning step. This method also creates a tree like patterns that can help decision makers to visual data easily. The proposed algorithm is tested against traffic accident

data set.

Cutbill et al. [83] used Jaccard similarity technique as a method to identify the redundancy constraints in optimization algorithms that can cause to steer the algorithm from the feasible solution . The rules produced by this method are found informative and correct.

Seol et al. [84] proposed wTabular-algorithm that removes unimportant rules discovered in big data sets by assigning each rule a weight. The proposed method employes Quine-Mccluskey method for rule reduction. The result of comparision with existing schemes such as Apriori and FP-growth algorithm gives that the proposed algorithm improves in the terms of processing time,support, credibility, and rule reduction rate.

Djenouri et al. [20] Proposed a metarules discovery-based approach that gives users the summary of rule space through a meta-rules representation. This method uses the bees swarm optimization concept. The proposed approach gives an option to prune rules to the users. The proposed method is extended using GPU-based parallel programming to test with large datasets.

Table 3.1: Literature Review Summary

Paper	Year	Algorithm	Strengths	Limitations	Domain	Perf. Measure
Zaki et al. [85]	2000	closed frequent item sets	Reduces redundant rules exponentially compared to old one. .	Deals with rules with single consequent. .	General	Time
B. Liu et al. [53]	1999	Direction setting(DS) and Non-direction setting with correlation	Good with highly correlated items. Generates small number of DS instead of huge number of associaton rules	Discovers huge rules	General	Time
H. Yun et al. [62]	2003	Relative Support Approach	Performs shorter time than Apriori by removing many rules containing unnecessary rare data. .	Generates Large Number of infrequent itemset. The algorithm is slow for large dataset.	General	Time
Kim and Kwak [64]	2005	Surprisal Measure	Eliminates noisy in the data to enhance the quality of data to be manned. Prunes uninteresting attributed from data set before mining process	Sensitive for bias on statistics aspects of attribute.	General	Rules
K.Tayal and V.Ravi [66]	2015	Improved Algorithm (Parallel)	Gives better predication for diseases compared to conventional methods. .	Generates large number of association rules.	Health	NA
Fernandes and Garc [86] a	2012	Visualization Technique	This approache provides dimensionality reeducation and improves the quality of extracted association rules. .	The mean shift used in this algorithm is sensible to bandwidth selection. uses iterative process which takes long time. .	General	Time
S. Khan and X. Huang [68]	2006	Network Semantic Technique	Uses grouping and summerization to reduce the un-manageable number of rules generated in association rule mining.	The semantic trees used do not exactly reflect the semantic relationship of the intended objects.	General	Time
O.M.Soysal [82]	2015	Heuristic Approach	Searches best solution by using heuristic technique. Prevents unneeded generation of rules in the first place.	Requires more knowledge acquisition for getting experience before taking single decision.	General	NA
Djenouri et al. [20]	2017	GSum-BSO	Employes parallel processing to process very large data	Requires user involvement and only gives meta-rule summary.	General	Time

CHAPTER 4

THE PROPOSED STRUCTURE BASED RULE PRUNING ALGORITHMS

The proposed association rules clustering approach consists of four MapReduce algorithms. The first MapReduce algorithm is called *PPrune*. It prunes strong association rules based one their structure. It is an optional step which is only done if the number of strong association rules is too large. The second MapReduce algorithm is called *Create-ACM*. This algorithm reads all the strong association rules and transforms them into a 2-dimensional array called ACM, which stands for Antecedent-Consequent Matrix. ACM is used to store the support count of strong association rules. The support counts are then used to compute the lift values of the strong association rules. Lift values of rules will be explained in the next chapter. The third MapReduce algorithm, *Compute-Lift*, takes as input ACM and

a transaction database and computes the lift values of the strong association rules. The last MapReduce algorithm is *Cluster-SAR* which groups strong association rules based on their lift values. Figure 4.1 show the block diagram of the proposed approach.

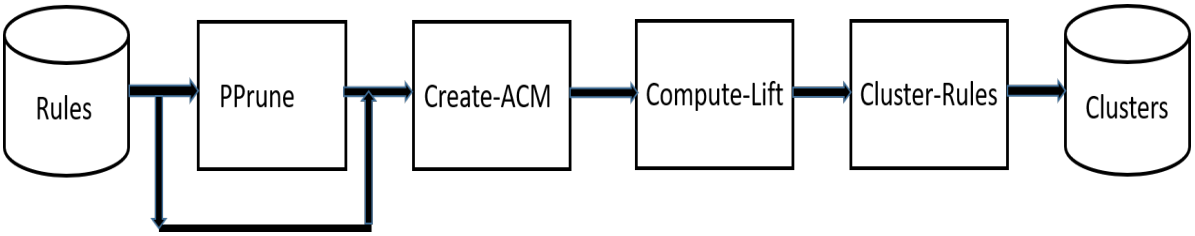

Figure 4.1: Block diagram of the proposed approach.

Rule structure based pruning is a fast way of reducing the number of strong association rules to be clustered. It is needed to speedup clustering rules that are extremely large. In its conservative form, it prunes the rules without reading the transaction database. The idea is based on structural rule cover which is explained in [14]. A *rule cover* states that if X, Y, and Z are three itemsets of a given transactional database D, then Φ(X,Y,Z) $\subseteq$ Φ(X,Y), *where* Φ(x) is set of transactions in D that match the itemset x. In other words, the transactions that match the rule $X, Z \rightarrow Y$ are a subset of the transactions that match $X \rightarrow Y$. If rules such as $X, Z \rightarrow Y$ are removed from a rule cover, then the remaining set which contains $X \rightarrow Y$ is a rule cover. A structural rule cover consists of the most general rules of the original set of rules.

In this chapter we propose two structure based association rules pruning algorithms called SPrune and PPrune. The other three MapReduce algorithms, namely, Create-ACM, Compute-lift, and Cluster-SAR are discussed in the next

chapter. SPrune is a sequential algorithm whereas PPrune is a parallel, MapReduce, algorithm. Sprune is needed to compute the speedup of PPrune. SPrune is explained in the next section and PPrune is explained in the one after.

4.1 Sequential Rule Pruning Algorithm

The proposed sequential association rules pruning algorithm, SPrune, goes through two phases to prune given rules as shown in Figure 4.2. In Phase 1, it reads a set of rules, $R = \{r_1, r_2, \ldots r_{|R|}\}$, and partitions the rules into a set of partitions; and in Phase 2, it clusters the rules one partition at a time. The output of SPrune is a set of general rules $G = \{g_1, g_2, \ldots g_{|G|}\}$. The partitioning phase (Phase 1) is necessary to reduce the number of disk accesses needed during the pruning, Phase 2.

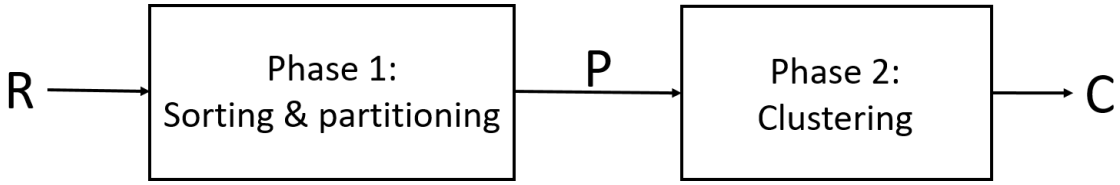

Figure 4.2: The two phase of SPrune.

In Phase 1, SPrune partitions the rules twice. First, it partitions the rules based on rule consequent. In other words, rules with the same consequent are put into the same partition. Let $P = \{P_1, P_2, \ldots P_{|P|}\}$ be the set of rule partitions created as a result, where $|P|$ is the total number of partitions which is also equal to the number of distinct rules consequents. During partitioning, the size of each rule antecedent (the number of items in a rule antecedent) is computed

and items in the rule sorted in alphabetical order. For example, if a rule antecedent, of size 3, is {milk, sugar, bread} before sorting, it will be {bread, milk, sugar} after sorting. Sorting each rule antecedent is done to speeds up Phase 2 processing. SPrune again partitions each P_i into sub-partitions according to the size of rule antecedent. Rules of the same antecedent size are put into the same sub-partition. For example, rules of antecedent size two are put in the same sub-partition and those of antecedent size 3 are put in another sub-partition. Let $\{P_{i,1}, P_{i,2}, \ldots, P_{i,|P_i|}\}$ be the set of sub-partitions created from P_i, where $|P_i|$ is their total number. Also, let the size of a rule antecedent in $P_{i,j}$ be less than that of a rule in $P_{i,j+1}$. Figure 4.3 shows an example of a hierarchy of partitions that can be created by Phase 1 of SPrune.

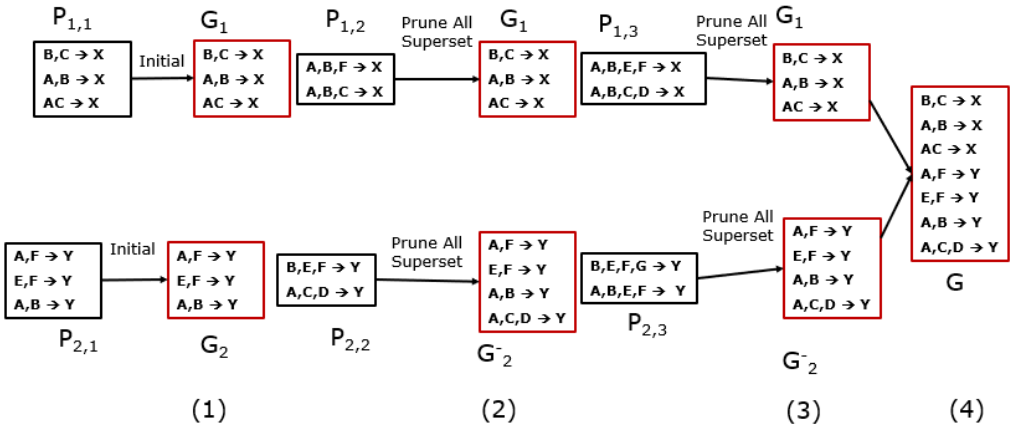

Figure 4.3: Partitions and sub-partitions of R created in Phase 1 of SPrune

Phase 1 of the SPrune algorithm is shown in Figure 4.3. The algorithm takes as input R and gives as output sub-partitions. In lines 3 and 4, the function computes the size of each rule antecedent and sorts it in alphabetical order. The function *partition*, Line 5, puts each rule in the right partition according to its consequent.

In Line 6, the function *sub-partition*, puts each rule in its sub-partition based on its consequent and the size of its antecedent. At last, the algorithm returns the sub-partitions.

Algorithm 1 Phase 1 of SPrune: Partitioning

Require: $R = \{r_1, r_2, \ldots, r_{|R|}\}$
Ensure: $sub - partitions$ r ∈ R

1: r.consequent ← get-consequent(r)

2: r.antecedent ← get-antecedent(r)

3: r.size ← get-size(r.antecedent)

4: r.antecedent ← sort(r.antecedent)

5: partition(r, r.consequent)

6: sub-partition(r, r.size)

7: **return** sub-partitions

In Phase 2, SPrune prunes the rules in each partition; but before we explain how SPrune prunes rules, let us define the terms *subset* and *superset* in the context of SPrune.

Definition 4.1 *A rule r_i is a subset of a rule r_j, if all the items of r_i are also items of r_j.*

Definition 4.2 *A rule r_j is a superset of a rule r_i, if all the items of r_i are also items of r_j.*

SPrune prunes the rules in each partition independently of those in the other partitions. This is because rules with different consequents can't be mapped to

the same general rule. Sprune starts with the first sub-partition of P_1 which is $P_{1,1}$. It considers each rule in $P_{1,1}$ as a general rule. Let $G = \{g_1, g_2, \ldots, g_{|G|}\}$ be the set of general rules after processing $P_{1,1}$. Then the processing moves to the rules in $P_{1,2}$. If a rule in $P_{1,2}$ is superset of any rule in G, then the rule is pruned, otherwise it is added to G and $|G|$ is incremented by 1. After all the rules in $P_{1,2}$ are processed, SPrune processes the rules in $P_{1,3}$, then those in $P_{1,4}$ and so on until all the rules in P_1 are completed. After SPrune is done processing all the rules in P_1, it will do the same with the rules of P_2, then P_3, and so on until it finishes processing the the rules in $P_{|P|}$. This is equivalent to processing all the rules in R.

Phase 2 of SPrune is depicted in Algorithm 2. The algorithms takes as input the sub-partitions created in Phase 1 and gives as output G which is a set of general rules. At Line 1, SPrune initializes the set of general rules G to null. From lines 4 and 9 it loops through each partition, p_i, and each of its sub-partitions, $P_{i,j}$. At Line 3 it initializes an empty set G_i of general rules of the current partition, P_i. Then for each rule in $P_{i,j}$, it will do the following. If the rule belongs to the first sub-partition, $P_{i,1}$, or the rule is not a superset of any of the rules in G_i, then the rule is added to G_i, Lines 6 to 9. At last, at Line 10, all the rules in G_i are added to G. Figure 2 shows how Phase 2 prunes strong association rules.

Algorithm 2 Phase 2 of SPrune: Pruning

Require: $P_{i,j}$ for $i = 1, 2, \ldots |P|$, $j = 1, 2, \ldots, |P_{|P_i|}|$
Ensure: G

1: $G \leftarrow \oslash$

2: **for** i = 1; i $\leq$ $|P|$; i++ **do**

3: $G_i \leftarrow \oslash$

4: **for** j = 1; j $\leq$ $|P_i|$; j++ **do**

5: **for all** r $\in$ $P_{i,j}$ **do**

6: **if** j == 1 **then**

7: $G_i \leftarrow G_i \cup r$
! Cover(G_i, r)

8: $G_i \leftarrow G_i \cup r$

9: **end if**

10: $G \leftarrow G \cup G_i$

return G

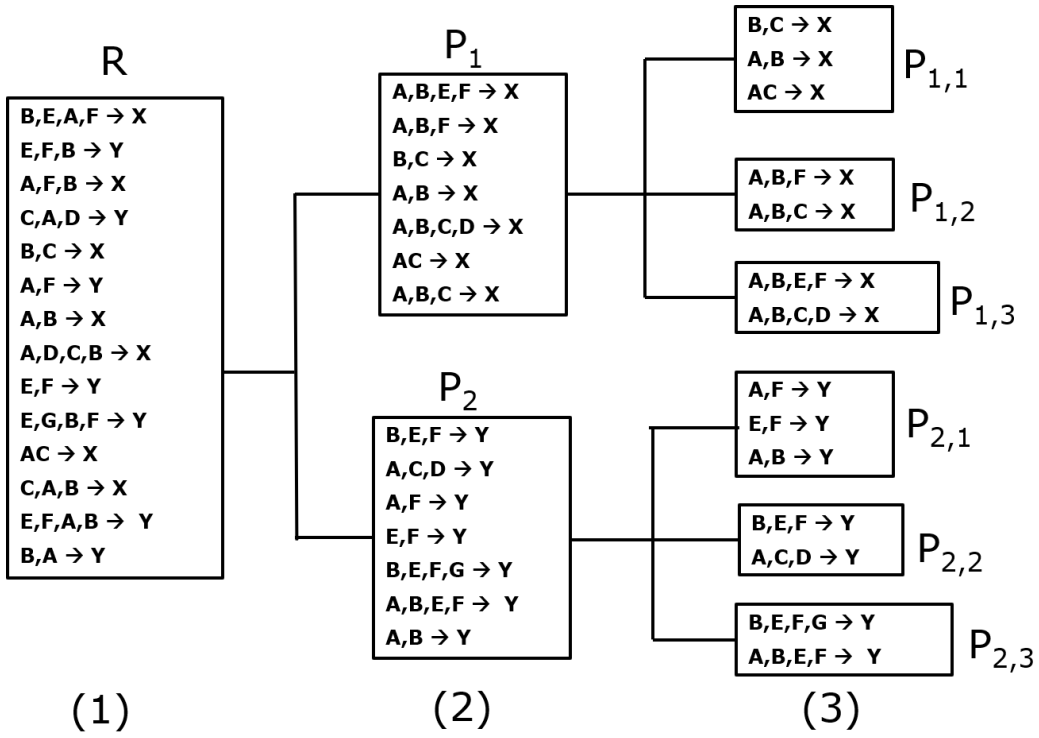

Figure 4.4: SPrune partitioning steps

Example 4.1

Figure 4.4 shows how SPrune partitions strong association rules in Phase 1. Sprune first reads the rules in R and partitions them according to their consequents in to Partitions P_1 and P_2. The consequent of the rules in P_1 is X and those in P_2 is Y. It again partitions the rules in P_1 according to the size of their antecedents in to three partitions, namely, $P_{1,1}$, $P_{1,2}$, and $P_{1,3}$. Each rule antecedent in $P_{1,1}$ has 2 items, those in $P_{1,2}$ has 3 items and those in $P_{1,3}$ has 4 items. Similarly, Sprune partitions the rules in P_2 into $P_{2,1}$, $P_{2,2}$, and $P_{2,3}$ for the same reason.

Example 4.2

Figure 4.5 shows the steps of how SPrune prunes rules in Phase 2. Let us consider the rules in R and assume that they are already partitioned in Phase 1

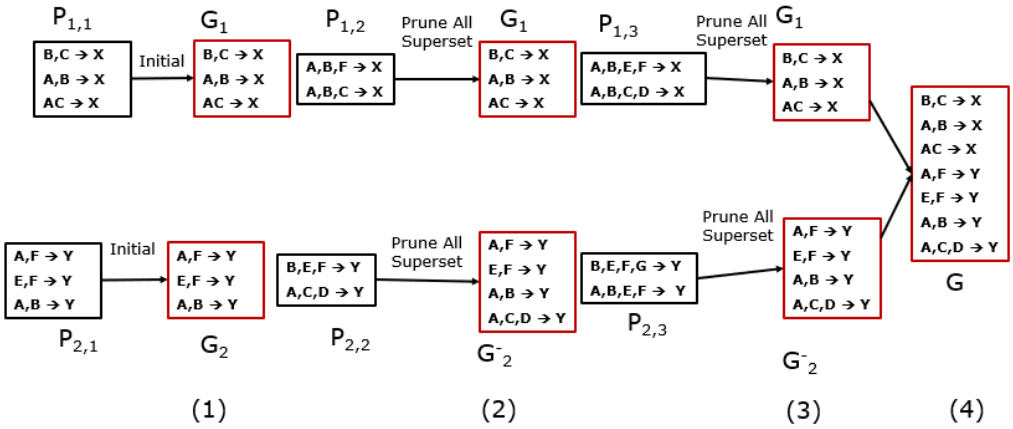

Figure 4.5: SPrune pruning steps

as shown in Figure 4.4. In Phase 2, SPrune starts with an empty set G_1 of general rules. In Step 1, it reads the rules in $P_{1,1}$ and puts all of them in G_1. So after Step 1, G_1 will consists of 3 rules, namely, $BC \Rightarrow X$, $AB \Rightarrow X$, and $AC \Rightarrow X$. In Step 2, SPrune will read each rule in $P_{1,2}$ and check if the rule is already covered by a rule in G_1. All the rules in $P_{1,2}$ are covered so none of them qualifies to be a general rule. In Step 3, it will repeat what it did in Step 2 but with the rules in $P_{1,3}$. All the rules in $P_{1,3}$ are also covered so none of them will be added to G_1. So by the end of Step 3, G_1 will consists of the same three rules it added in Step 1. After discovering all the general rules in P_1, it starts searching for general rules in P_2. It starts with an empty G_2 and the rules in $P_{2,1}$. After processing the rules in $P_{2,1}$, G_2 will consist of three rules, namely, $AF \rightarrow Y$, $EF \rightarrow Y$, and $AB \Rightarrow Y$. Then it will search for general rules in $P_{2,2}$. From the two rules in $P_{2,2}$, $BEF \rightarrow Y$ is covered by $EF \rightarrow Y$, but $ACD \Rightarrow Y$ is not covered by any rule and thus added to G_2. After processing $P_{2,2}$, G_2 will consist of $AF \rightarrow Y$, $EF \rightarrow Y$, $AB \Rightarrow Y$ and $ACD \rightarrow Y$. After SPrune is done processing all the rules in $P_{2,2}$

it will process the rules in $P_{2,3}$. All the rules in $P_{2,3}$ are covered so none of them will be added to G_2. At last SPrune puts all the rules of G_1 and G_2 in to G which is the set of all the general rules. Out of the 14 rules in R, SPrune found seven general rules and the remaining seven rules will be pruned.

4.2 The Proposed Parallel Association Rules Pruning Algorithm

The proposed parallel association rules pruning algorithm, which we named PPrune, is similar to SPrune but is implemented using MapReduce and runs in Hadoop. For a given R, the general rules discovered by PPrune are the same as those discovered by SPrune. The Map method of PPrune reads each $r \in R$ and identifies its antecedent and consequent, r.antecedent and r.consequent. It then sorts r.antecedent and computes its size, r.size, which is the number of items in r.antecedent. At last, it makes r.consequent as the key and r.antecedent and r.size as the value and sends them to the reducer.

The Map method of PPrune is depicted in Algorithm 3. It takes r as input and computes its antecedent, its consequent, and the size of its antecedent, lines 3 to 5. It then makes the consequent as the key and the antecedent and its size as the value and sends them to the reducer, lines 7 to 9.

The Reduce method of PPrune takes as input a rule consequent as a key, K1, and set of strong association rules with the same consequent as a value, V1. It

Algorithm 3 The Map method of PPrune

Require: key, r
Ensure: k1, v1

```
1: Class Mapper
2: Method Map(key, r)
3: r.consequent ← get-consequent(r)
4: r.antecedent ← get-antecedent(r)
5: r.size ← get-size(r.antecedent)
6: r.antecedent ← sort(r.antecedent)
7: k1 ← r.consequent
8: v1 ← r.size + "," + r;
9: Emit(k1, v1);  =0
```

gives as output a set of general rules as output value, V2. It goes through two phases. In Phase 1, it puts each r in $V1$ with the same r.consequent and r.size into the same sub-partition, S[r.size], lines 3 to 5. After it puts each rule in one of the sub-partition of S, it deletes the empty sub-partitions of S by resizing and re-indexing S, Line 6.

In Phase 2, PPrune prunes covered rules and starts at Line 7. First it initializes the set of general rules G to null, Line 7. In lines 8 to 13, it loops through each sub-partitions of S and each rule in a sub-partition and does the following. If a rule belongs to the first sub-partition, $S[1]$ or the rule is not a superset of any of the rules in G, then the rule is added to G, otherwise it is pruned, Lines 10 to 12. At last, at Line 16, it emits G.

Algorithm 4 The Reduce method of PPrune

Require: k1, V1
Ensure: $V2$

```
1: Class Reducer
2: Method Reduce(K1, V1)
3: for all r ∈ V1 do
4:     S[r.size] ← r
5: end for
6: S ← Re-index(S)
7: G ← ∅
8: for i = 1; i ≤ |S|; i++ do
9:     for all r ∈ S[i] do
10:        if i == 1 then
11:            G ← G ∪ r
   ! Cover(G, r)
12:            G ← G ∪ r
13:        end if
14:        k2 ← k1;
15:        V2 ← G;
16:        Emit(k2, V2)
17:    end for
```

CHAPTER 5

THE PROPOSED LIFT-BASED RULE CLUSTERING APPROACH

This chapter explains the proposed rule clustering approach. It consists of the remaining three MapReduce algorithms introduced in the previous chapter, namely, Create-ACM, Compute lift, and Cluster-SAR. The approach is based on association rules interest measure known as lift [87]. The lift of a rule $A \rightarrow C$ is defined as

$$lift(A \rightarrow C) = \frac{support(A \cup C)}{support(A)support(C)} \tag{5.1}$$

Lift measures the correlation of a rule with its antecedent and its consequent. A lift of value one indicates that a rule is not correlated to its antecedent and consequent; and a value much higher than one shows strong correlation. The proposed clustering approach is based on the assumption that rules with antecedents that

are highly correlated with the same set of consequents are similar and thus should be clustered together [87]. Unlike the existing clustering approaches, this approach clusters antecedents containing itemsets which are rarely purchased together. This is because of the high correlation they have with the same consequents.

To perform the clustering efficiently, we created a 2-dimensional array (a matrix), M. The size of M is $|A|$ by $|C|$, where $|A|$ is the number of distinct antecedents and $|C|$ is the number of distinct consequents of all the strong association rules that are going to be clustered. The element $m_{i,j}$ of M contains the lift value of the rule $A_i \rightarrow C_j$, where A_i is the antecedent which corresponds to the i^{th} row of M and C_j is consequent which corresponds to the j^{th} column of M. An element of M which doesnt belong to a strong association rule is assigned to a lift value of 1.

The distance between antecedents A_i and A_j is defined as:

$$dis(A_i, A_j) = \sqrt{\sum_{k=1}^{|C|} m_{i,k} - m_{j,k}} \tag{5.2}$$

To cluster antecedents into set of K groups, namely $G = {}_1, G_2, \ldots, G_k$, we use k-means algorithm and minimize the within cluster sum of squares, $sum_{i=1}^{K} sum_{A_j \in G_i} dis(A_j, \mu_i)$, where μ_i is the centroid of G_i.

5.1 The Create-ACM MapReduce Algorithm

The Create-ACM MapReduce algorithm takes as input strong association rules and initializes two 1-dimensional arrays called RHS and LHS, and a 2-dimensional array called ACM. LHS is indexed by the distinct antecedents of the strong association rules; and RHS is indexed by the distinct consequents of the strong association rules. Let $A = \{A_1, A_2, \ldots A_{|A|}\}$ be the set of all distinct antecedents and $C = \{C_1, C_2, \ldots C_{|C|}\}$ be the set of all distinct consequents in the strong association rules, where $|A|$ is the number of distinct antecedents and $|C|$ the number of distinct consequents. The size of ACM is $|A|$ by $|C|$. Its rows are indexed by the members of A and its columns are indexed by the members of C. Let $m_{i,j}$ be an element of ACM indexed by A_i and C_j and corresponds to the rule $A_i \rightarrow C_j$.

The Map function of Create-ACM extracts the antecedent and the consequent of each rule and emits them to the reducer function. The function also initializes two global arrays called LHS and RHS. The size of LHS is $|A|$ and the size of RHS is $|C|$. LHS and RHS elements are initially set to 0. The map function of the *Create-ACM* algorithm is depicted in Algorithm 5. At lines 3 and 4 the function extracts the antecedent and the consequent of a rule. It then adds the consequent of a rule to RHS and the antecedent to LHS, lines 5 and 6. At Line 7 the function emits the antecedent and consequent of a rule to the reducer.

The Reduce function of Create-ACM, creates ACM array. It is depicted in Algorithm 6. The function takes an antecedent and all its consequents from the

Algorithm 5 The Map method of Init-ACM

Require: LHS, RHS, k1, v1
Ensure: k2, v2

```
1: Class Mapper
2: Method Map(k1, v1)
3: K2 ← get-antecedent(v1)
4: V2 ← get-consequent(v1)
5: LHS ← AddLHS(v1);
6: RHS ← AddRHS(v2);
7: Emit(k2, v2);
```

Mapper as input. It also uses a global lists LHS and RHS initialized by the reducer. At Line 3, the reducer uses a function called ACM-Init to create a row of ACM which contains $|C|$ elements which are all initialized to -1. The row corresponds to one of the antecedents and each of its elements corresponds to one of the $|C|$ consequents. Each element in a row correspond to a rule. At Line 4, each element of ACM which corresponds to a strong rule is set to 0. At last, the Reducer emits the current antecedent with its corresponding ACM row, Line 8.

5.2 The Compute-Lift MapReduce Algorithm

The Compute-Lift algorithm takes as input transactions and gives as output the lift values of the strong association rules. It uses the three global arrays TXN-count, LHS, RHS, and ACM to compute the lift values. The Map function of the algorithm is depicted in Algorithm 7. It counts the number of input transactions

Algorithm 6 The Reducer method of Init-ACM

Require: LHS, RHS, ACM, k2, V2
Ensure: k3, v3

1: Class Reducer

2: Method Reduce(K2, V2)

3: ACM-Init(k2, RHS, ACM);

4: **for all** v $\in$ $V2$ **do**
$ACM[K2, v] = 0$;

6: K3 $\leftarrow$ k2;

7: V3 $\leftarrow$ ACM[K2];

8: Emit(k3, V3);

At Line 3. At Line 4, it generates all the possible rules from a transaction. It then, at Line 5, it extracts the antecedents and consequents of each rule generated in Line 4. If the antecedent is in LHS, then the corresponding element in LHS is incremented by 1, Line 10. Also If the consequent is in RHS, then the corresponding element in RHS is incremented by 1, Line 11. At last, the map function emits the current antecedent and consequent if they have a corresponding element in ACM, Line 12.

The reducer function of Compute-Lift is shown in Algorithm 8. It takes an antecedent and all its consequents. It also uses the global variables TXN-count, ACM, LHS and RHS. The reducer function receives from the reducer, an antecedent and all its consequents. For each consequent, it checks the corresponding ACM element if it belongs to a strong association rule, Line 4. If it belongs to a strong association rule, then the count of that element is incremented by 1, Line

Algorithm 7 The Map method of Compute-Lift

Require: ACM, k1, v1, TXN-count, LHS, RHS
Ensure: K2, v2

```
1: Class Mapper
2: Method Map(k1, v1)
3: TXN-count++;
4: R ← generate-rules(v1)
5: for all r ∈ R do
   c← r.consequent;
   a ← r.antecedent;
6:     update(LHS, a);
8:     update(RHS, c);
9:     if Exists(ACM, a, c) then
10:        k2 ← a;
11:        V2 ← c;
12:        Emit(K2, V2);
13:        =0
```

4. At last, the reducer computes the lift values using the Compute-lifts function At line 7, and emits the antecedent and the corresponding lift values.

Algorithm 8 The reducer method of Compute-Lift

Require: TXN-count, ACM, LHS, RHS, k2, v2

Ensure: K3, v3

1: Class Reducer

2: Method Reduce(k2, v2)

3: a $\leftarrow$ = k2;

4: **for all** c $\in$ $V2$ **do**
if (**then**$ACM[a, c] >= 0$)
$ACM[a, c] + +$;

6: k3 $\leftarrow$ a;

7: V3 $\leftarrow$ Compute-lifts(ACM, count, LHS, RHS, a, c);

8: Emit(K3, V3);

5.3 The Association Rules Clustering Algorithm, Cluster-SAR

The Cluster-SAR algorithm takes as input the rows of the 2-dimensional array ACM and returns as output the cluster of each strong association rule. Let us refer to each row of ACM as a sample. As explained above, each sample corresponds to a distinct antecedent and each element of a sample corresponds to a distinct consequent. Each element $m_{i,j}$ of ACM contains the lift value of the rule $A_i \rightarrow C_j$.

Cluster-SAR uses K (a pre-specified value) and global variables, called centroids, which are initialized by random values. The algorithm is a slight modification of the one proposed in [87]. The Map function of the algorithm is depicted in Algorithm 9. At Line 3, the function initializes the local variable index to -1 and the MinDistance to the highest real number. For each sample it reads, the map function computes the distance of the sample from each of the K centroids. It associates each sample with the index of the closest centroid, Lines 4 to 6. At last, at Line 9, the function emits each centroid with its corresponding samples.

Algorithm 9 The Map method of Cluster-SAR

Require: centroid, k1, v1
Ensure: K2, v2

```
1: Class Mapper
2: Method Map(k1, v1)
3: Init(index, minDistance);
4: for i = 0; i ¡ k; i++ do
5:     distance ← EuclideanDistance(V1, centroid[i])
6:     if  dis ≤ MinDistance then
   minDistance ← distance;
   index = i;
7:         K2 ← index;
8:         V2 ← to_string(V1);
9:         Emit(K2, V2);
```

The Reduce function of Cluster-SAR computes the new centroids. It takes

as input each centroids and associated samples. For each centroid, it computes the sum of the corresponding elements in its corresponding samples, Line 4. It also counts the number of samples associated with each centroid, Line 6. It then computes the average of the samples to generate the new centroids, Line 7.

Algorithm 10 The Reduce method of Cluster-SAR

Require: k2, v2
Ensure: K3, v3
1: Class Reducer

2: Method Reduce(k2, v2)

3: Init(SumV2, count);

4: **for all** v $\in$ $V2$ **do**
$ComputeSum(SumV2, 2)$;

6: count++

7: centroids $\leftarrow$ ComputeCentroids(SumV2, count);

8: K3 $\leftarrow$ k2;

9: V3 $\leftarrow$ to_string(centroids);

10: Emit(K3, V3);
=0

CHAPTER 6

EXPERIMENTAL EVALUATION

In this chapter, we present the experimental results and performance analysis of the proposed algorithms. In Section 6.1, the experimental settings, such as the computer system, the software tools, and the datasets used during the experiments are discussed. Also in the same section, the performance measures used to evaluate the proposed algorithms are discussed. The experimental results and performance analysis of the four MapReduce algorithms are presented in subsequent sections.

6.1 Experimental Settings

In this section, the computer system, the pieces of software, the benchmark datasets and the evaluation metrics used in the experiments are presented.

Node	Node Type	Cores	RAM	Processor
Name Node	Virtual	2	8	Intel i7-7600HQ CPU @ 2.80 GHz
Data Node 1	Virtual	2	4	Intel i7-7700HQ CPU @ 2.81 GHz
Data Node 2	Virtual	2	4	Intel i7-7700HQ CPU @ 2.81 GHz
Data Node 3	Virtual	2	4	Intel i7-7700HQ CPU @ 2.81 GHz
Data Node 4	Virtual	2	4	Intel i7-7700HQ CPU @ 2.81 GHz

Table 6.1: Configuration of Hadoop Cluster

6.1.1 The Computer System and Software Tools

The proposed algorithms were written in Java and Python. The experiments were conducted on a local Hadoop 2.8.1 cluster. The cluster consisted of three physical machines and 5 virtual nodes. The Name Node was in one machine, and the four data nodes were in two other machines. Each two data nodes were in a machine. Table 6.1 shows the specification of each node.

6.1.2 Experimental Datasets

To study the performance of the proposed algorithms, we experimented using five benchmark datasets, namely, Mushroom, Chess, T10I4D100K, Webdocs, and AllElectronics. The first four datasets were downloaded from the " Frequent Itemset Dataset Repositoris, https://fimi.ua.ac.be/data and http://fimi.uantwerpen.be/data/ ". AllElectronics dataset is used in some related work and it was copied from [88]. We chose these datasets because they are frequently used in related work and they have different characteristics. Table 6.2 shows the name, the number of items, the number of transactions, the average transaction size, and the number of association rules in each dataset.

Dataset	Size	Items	Transactions	Avg. Items per Transaction	Rules
AllElectronics	1 KB	5	9	2.6	52
Chess	335 KB	75	3196	37	108061
Mushroom	558 KB	119	8124	23	111790
T10I4D100K	3928 KB	870	100000	10	5608
Webdocs	1.48 GB	5,267,656	1,692,082	61	1,231,984

Table 6.2: Experimental Datasets

TID	Transaction
T100	I_1, I_2, I_5
T200	I_2, I_4
T300	I_2, I_3
T400	I_1, I_2, I_4
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3
T800	I_1, I_2, I_3, I_5
T900	I_1, I_2, I_3

Table 6.3: AllElectronics Datasets

AllElectronics dataset: The AllElectronics dataset is a small synthetic dataset of 9 transactions and 5 items, as shown in table 6.3. The first column represents transaction number and the second column lists the items of each transaction.

Chess dataset: This dataset is a chess endgames database. It is frequently used for machine learning experiments to classify positions. It contains game-theoretic values for legal positions. The stores game-theoretic values denote whether or not positions are won for either player, or the number of moves needed to win according to the minimax-optimal play.

Mushroom dataset: This dataset consists of the description of hypothetical samples of 23 gilled mushrooms species in the Agaricus and Lepiota Family. The dataset describes the edibility of each species as edible, poisonous, unknown, and

Dataset	Name	1 GB	2 GB	3 GB	4 GB
AllElectronics	D1	D1-1	D1-2	D1-3	D1-4
Chess	D2	D2-1	D2-2	D2-3	D2-4
Mushroom	D3	D3-1	D3-2	D3-3	D3-4
T10I4D100K	D4	D4-1	D4-2	D4-3	D4-4

Table 6.4: Dataset Labels

not recommended. They were drawn from The Audubon Society Field Guide to North American Mushrooms.

T10I4D100K dataset: T10I4D100K is one of the synthetic datasets which is frequently used by related work. It was generated by IBM Almaden Quest research group. The group named the datasets according to this convention: T: average number of items per itemset, I: average size of itemsets in a transaction, and D: the number of transaction in the dataset .

Webdocs dataset: The Webdocs is a real life transactions dataset. It is a collection of web html documents. It contains 1,692,082 transactions and the longest transaction contains 71,472 items.

As can be seen from Table 6.2, the first four datasets are very small; hence, we replicated them and created 16 files of sizes 1GB, 2GB, 3GB and 4GB. We also labeled each of these 16 files as shown in Table 6.4. For example, we refer to the 3 GB chess dataset as D2-3 and to the 2 GB Mushroom dataset as D3-2.

6.1.3 Evaluation Measures

To measure the performance of the proposed algorithms, we used four evaluation measures, namely elapsed time, speedup, scaleup, and sizeup.

Elapsed time refers to the amount of time a process takes from the time it was submitted until the time it was completed.

Speedup refers to how much faster is a parallel algorithm than the corresponding sequential algorithm; and is defined by Equation 6.1 as

$$Speedup = \frac{T_1}{T_p} \tag{6.1}$$

where T_1 is the elapsed time of a task done by a single CPU, and T_p is the elapsed time of the same task done in parallel by p CPUs. Ideally, we like the speedup to be p

Scaleup refers to the ability to keep the same elapsed time when both workload and resources increase proportionally; and is defined by Equation 6.2 as

$$Scaleup = \frac{T_1}{T_{n,n}} \tag{6.2}$$

where T_1 is the elapsed time a single CPU takes to finish a workload, and $T_{n,n}$ is the amount n CPUs takes to accomplish n times the workload. Ideally we like the scaleup to be 1.

Sizeup refers to how much longer a system takes if the workload increase n times; and is defined by Equation 6.2 as

$$Sizeup = \frac{T_n}{T_1} \tag{6.3}$$

where T_1 is the elapsed time to process a workload, and T_n is the elapsed time to

process n times the original workload. Ideally we like the sizeup to be less than or equal to n.

6.2 Performance Evaluation of Proposed Algorithms

The cost of a MapReduce algorithm is the sum of its I/O, CPU, and communication costs. Higher communication cost reduces the performance of a MapReduce algorithm. I/O cost scales well with the number of mappers so it has no negative effect on performance. If the percentage of CPU cost is high, then communication cost becomes less significant giving us better performance.

This section presents the experimental results and the performance analysis of the proposed four MapReduce algorithms. Each algorithm was experimented with 16 different datasets and up to four nodes. The elapsed time, speedup, sizeup, and scaleup figures of each algorithm are presented in the next subsection. In the subsequent subsection, the performance of the same MapReduce algorithms is presented using the Webdocs dataset and up to 16 data nodes.

6.2.1 Performance Evaluation of PPrune Algorithm

To study the performance of PPrune algorithm, we conducted many sets of experiments. The first set of experiments was done to study how the execution time of PPrune is affected by different datasets and different number of nodes. For this

experiment, we used D1-1, D2-1, D3-1, and D4-1 and we varied the nodes from 1 to 4. The results of this set of experiments are shown in Figure 6.1. As expected, as the number of nodes increases, the elapsed time of PPrune decreases. Also, the elapsed time increases with the number of items in a dataset. This is because transactions from datasets with more items require more time to compare. That is why D4-1 and D3-1 required more processing time than D1-1 and D2-1.

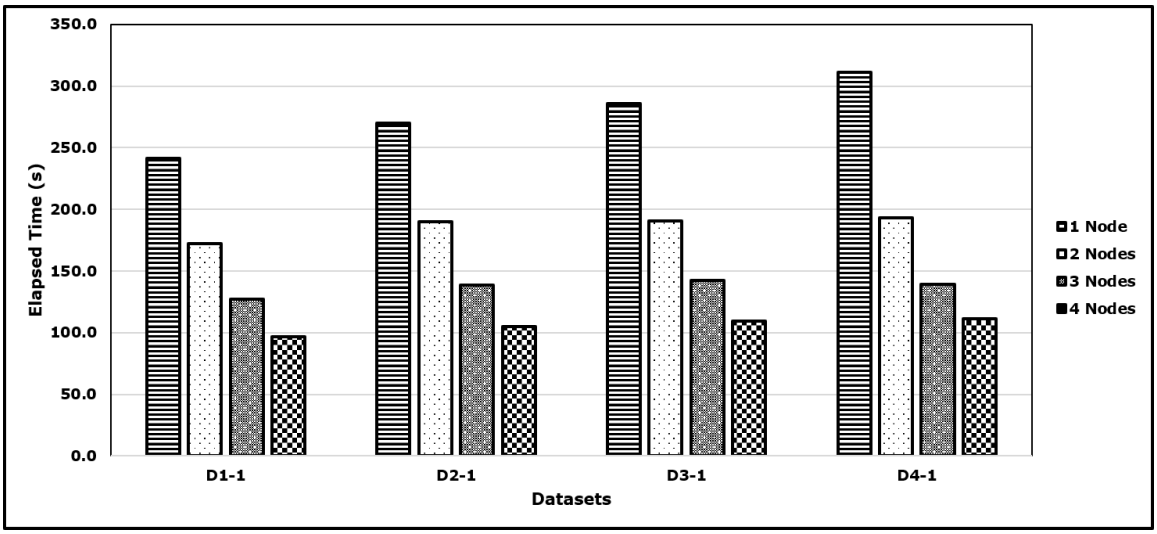

Figure 6.1: PPrune elapsed time.

To study the speedup, sizeup, and scaleup of PPrune, we conducted twelve sets of experiments. The first three sets were done to study the speedup, sizeup, and scaleup of PPrune using the AllElectronics dataset, D1. Figures 6.2 to 6.4 show the obtained results.

To measure the speedup of PPrune, we kept the size of D1 constant and varied the number of nodes from 1 to 4, as shown in Figure 6.2. The speedup of PPrune improved as the size of D1 increased. With small datasets, the proportion of the communication cost is higher to that of I/O and CPU. This is because the

percentage of the communication cost decreases to that of I/O and CPU costs when the dataset size increases.

To measure the sizeup of PPrune, we varied D1 size and the number of nodes as shown in Figure 6.3. The results show sublinear performance and it improved as the size of D1 increased. This is because higher percentage of time is spent on CPU and I/O than in communication.

The scaleup measure of PPrune was obtained by increasing the size of D1 in proportion with the number of nodes. Figure 6.4 shows the scaleup performance of the algorithm. The figure shows PPrune scales well. Scaleup decreases as the number of nodes and the size of D1 increase but it was always more than 80%.

From the experimental results, we can easily conclude that the performance of PPrune was very good when processing dataset D1 which is characterized by few items and short transactions or rules.

To study the performance of PPrune with a dataset with more items and longer transactions than D1, we conducted three sets of experiments with D2. The results of the experiments are shown in Figures 6.5, 6.6 and6.7. The performance of PPrune was even better than with D1 because the percentage of the communication cost was higher when processing D1 than when processing D2. Longer transaction result in longer rules which demand more CPU time than shorter rules.

Dataset D3 has more items but shorter transactions than D2. The performance of PPrune when processing D3 was similar to that of D2. Having more distinct

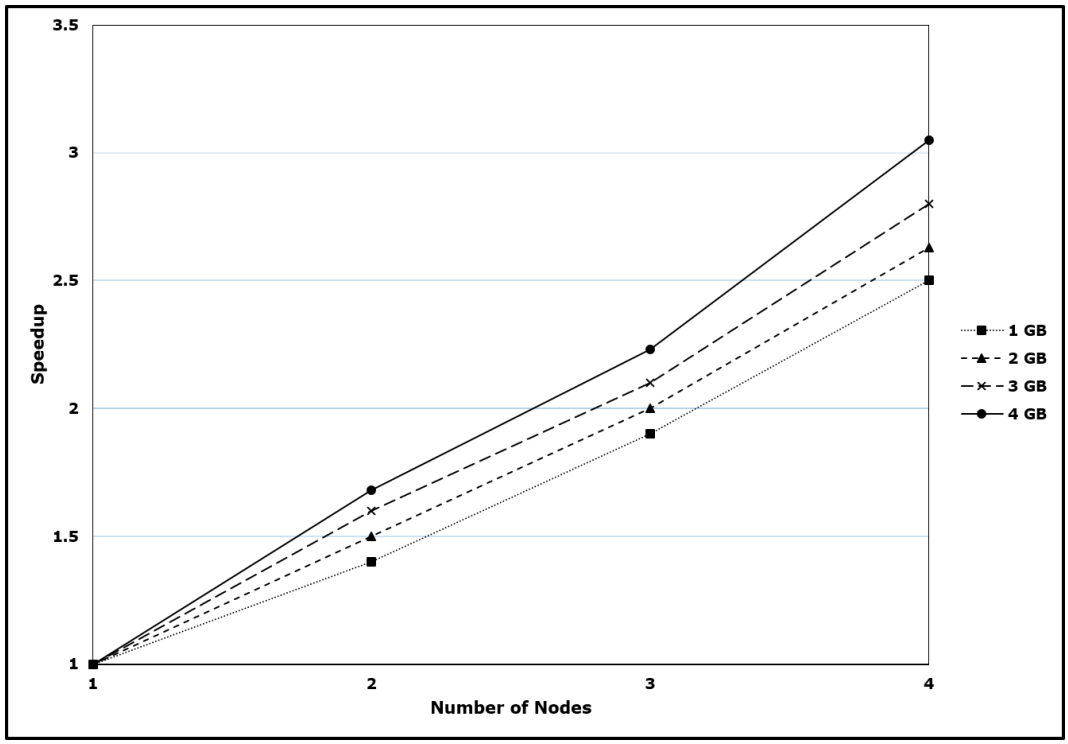

Figure 6.2: Speedup of PPrune on D1 Dataset.

items also increase the percentage of CPU cost of PPrune. This is because it adds to the number of rules which demands longer searches. To study the performance of PPrune with dataset D3. The results of the experiments are shown in Figures 6.8, 6.9 and 6.10. The performance of PPrune was also better than with D1 because the percentage of the communication cost was higher when processing D1 than when processing D2.

To study the performance of PPrune with a dataset with many number of distinct items, we conducted three sets of experiments using the dataset D4. D4 has 870 distinct items whereas D1 has only 5 distinct items. PPrune elapsed time is slightly with D4 than with D1 as shown in Figure 6.1. The speedup, scaleup, and sizeup measures of PPrune are similar or better when processing D4. This is

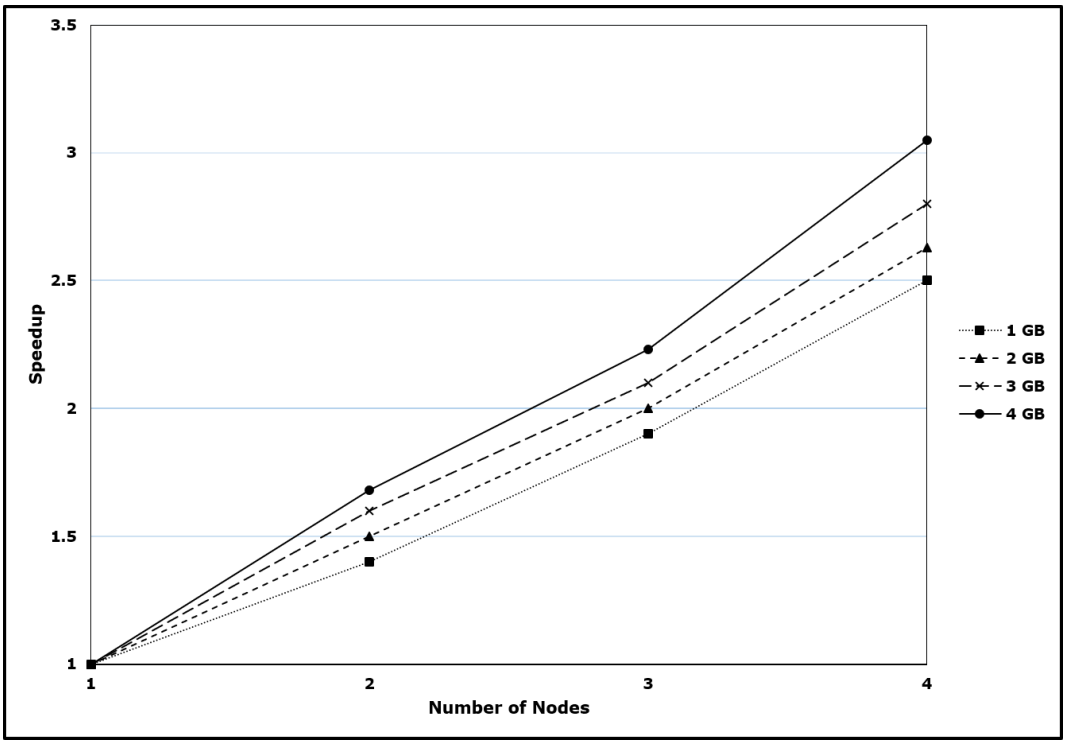

Figure 6.3: Sizeup of PPrune on D1 Dataset.

again more percentage of CPU time is needed to process D4 which has far more distinct items. The results of the experiments are shown in Figures 6.11, 6.12 and 6.13.

When computing speedup, any parallel algorithm must be compared with the best sequential algorithm. So we wrote the SPrune, a sequential algorithm, to compare it with PPrune. Before using SPrune to compute the speedup measure of PPrune, we compared the elapsed time of SPrune with that of PPrune which runs in 1 node. We found out that the elapsed time of SPrune was much higher in all the experiments we conducted using D1, D2, D3, and D4. This is because of the I/O cost of SPrune. Both algorithms have similar CPU and communication costs. The I/O cost of SPrune is higher because of the data block size. PPrune

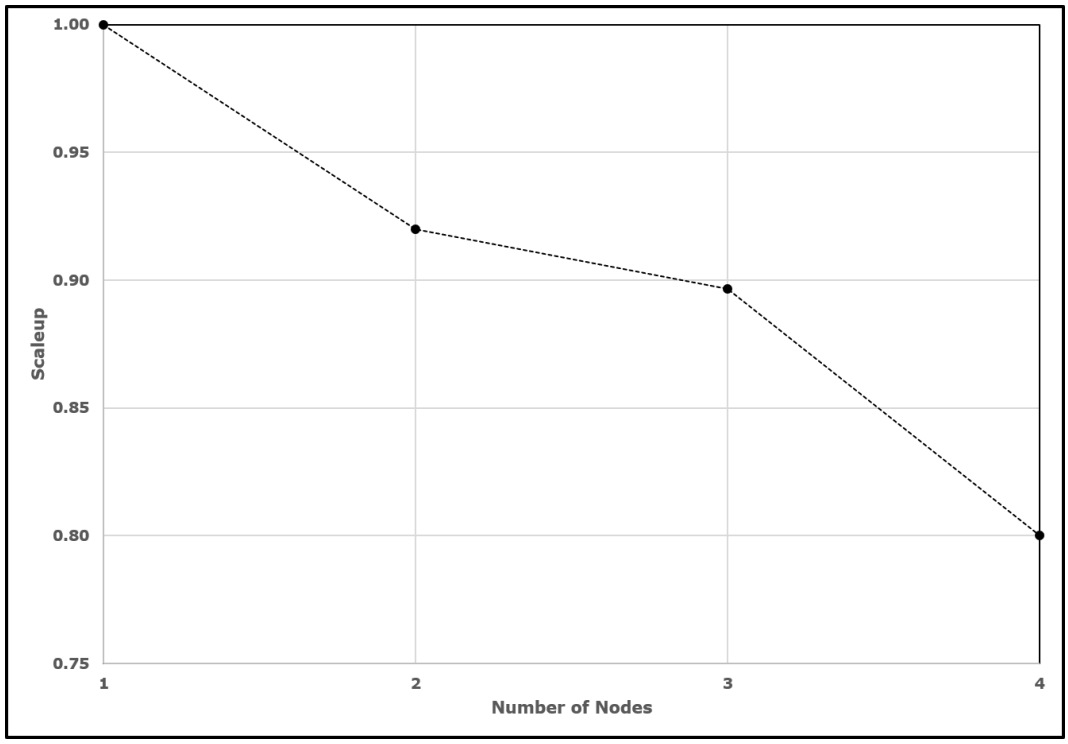

Figure 6.4: Scaleup of PPrune on D1 Dataset.

used a data block size of 128MB whereas SPrune used 4KB. So SPrune needed far more I/O than PPrune and that is why its elapsed time was much higher. As a result, we decided to use PPrune that runs in a single node than using SPrune.

6.2.2 Performance Evaluation of Create-ACM Algorithm

To study the performance of Create-ACM algorithm, we conducted similar experiments that we conducted for PPrune. The first set of experiments was done to study how the elapsed time of PPrune is affected by different datasets and different number of nodes. For this experiment, we used D1-1, D2-1, D3-1, and D4-1 and we varied the nodes from 1 to 4. The results of this set of experiments are shown in Figure 6.14. Again similar to that of PPrune, as the number of nodes

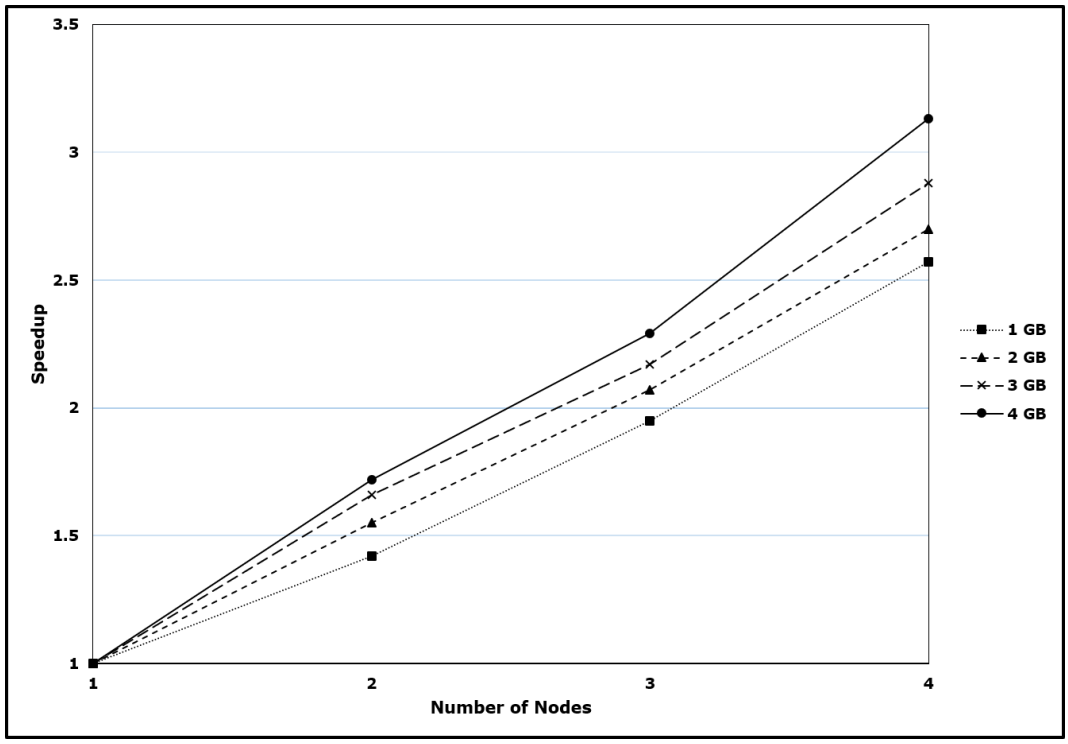

Figure 6.5: Speedup of PPrune on D2 Dataset.

increases, the elapsed time of Create-ACM decreases. Also, the elapsed time increases with the number of items in a dataset. This is because transactions from datasets with more items or longer rules require more time to compare. That is why D4-1 and D3-1 required more processing time than D1-1 and D2-1.

To study the speedup, sizeup, and scaleup of Create-ACM, we conducted many sets of experiments. The first three sets were done to study the speedup, sizeup, and scaleup of Create-ACM using D1. Figures 6.15, 6.16, and 6.17 show the obtained results. To measure the speedup of Create-ACM, we did the same as we did with that of PPrune. We kept the size of D1 constant and varied the number of nodes from 1 to 4, Figure 6.15. The speedup of Create-ACM improved as the size of D1 increased. This is because the percentage of the communication cost

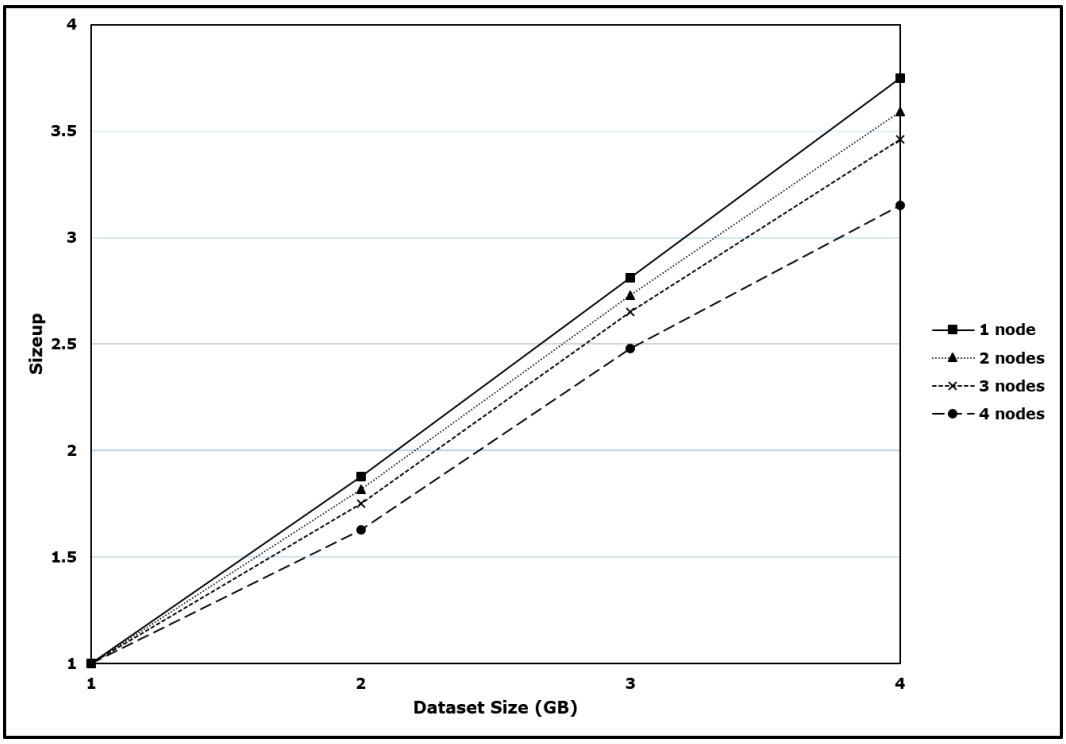

Figure 6.6: Sizeup of PPrune on D2 Dataset.

decreases when the dataset size increases. To measure the sizeup of PPrune, we varied D1 size and the number of nodes as shown in Figure 6.16. Again as that of PPrune, the results show sublinear performance and it improved as the size of D1 increased. This is because higher percentage of time is spent on CPU and I/O than in communication.

To evaluate the scaleup measure of Create-ACM, we increased the size of D1 in proportion with the number of nodes. Figure 6.17 shows the scaleup performance of the algorithm. The figure shows the algorithm scales well. Scaleup decreases as the number of nodes and the size of D1 increase but it was always more than 78%.

From the experimental results, we can safely conclude that the performance of

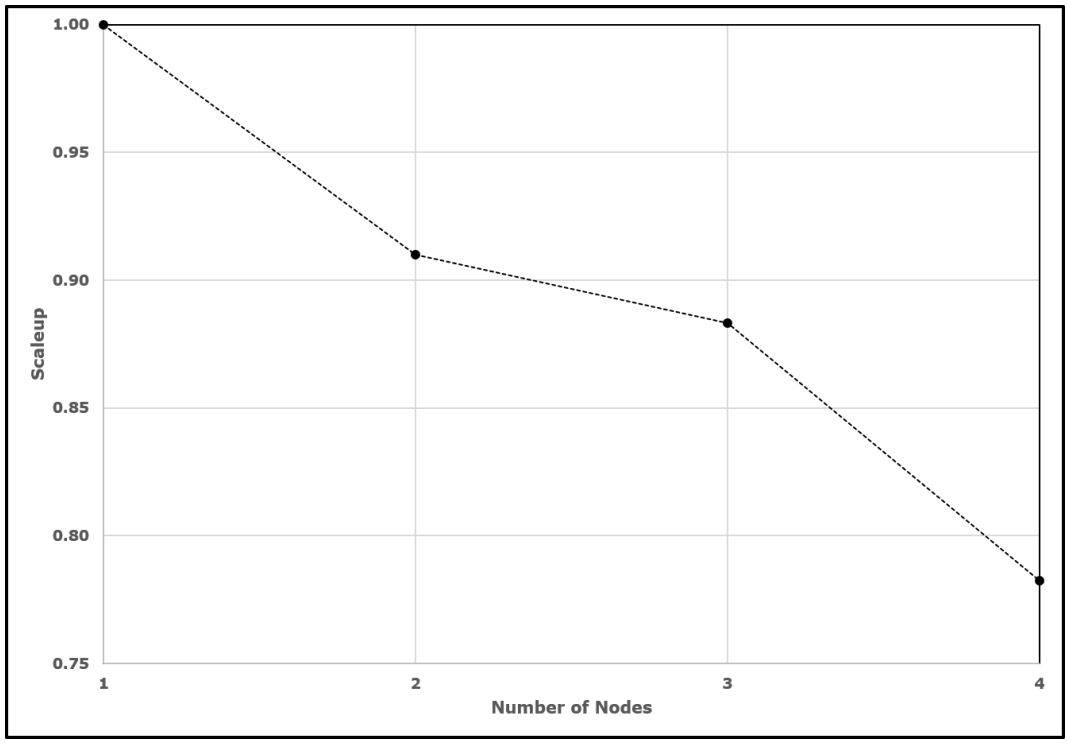

Figure 6.7: Scaleup of PPrune on D2 Dataset.

Create-ACM was very good when processing datasets characterized by few items and short transactions or rules such as D1.

Performance Evaluation of Create-ACM Algorithm With Dataset Of More Items

To study the performance of Create-ACM with a dataset with more items and longer transaction than D1, we conducted three sets of experiments with D2. The results of the experiments are shown in Figures 6.18, 6.19 and 6.20. The performance of Create-ACM was even better than with D1 because the percentage of the communication costs is less when processing D2 than when processing D1. This is because longer transaction or longer rules demand more CPU time.

As mentioned above, dataset D3 has more items but shorter transactions or

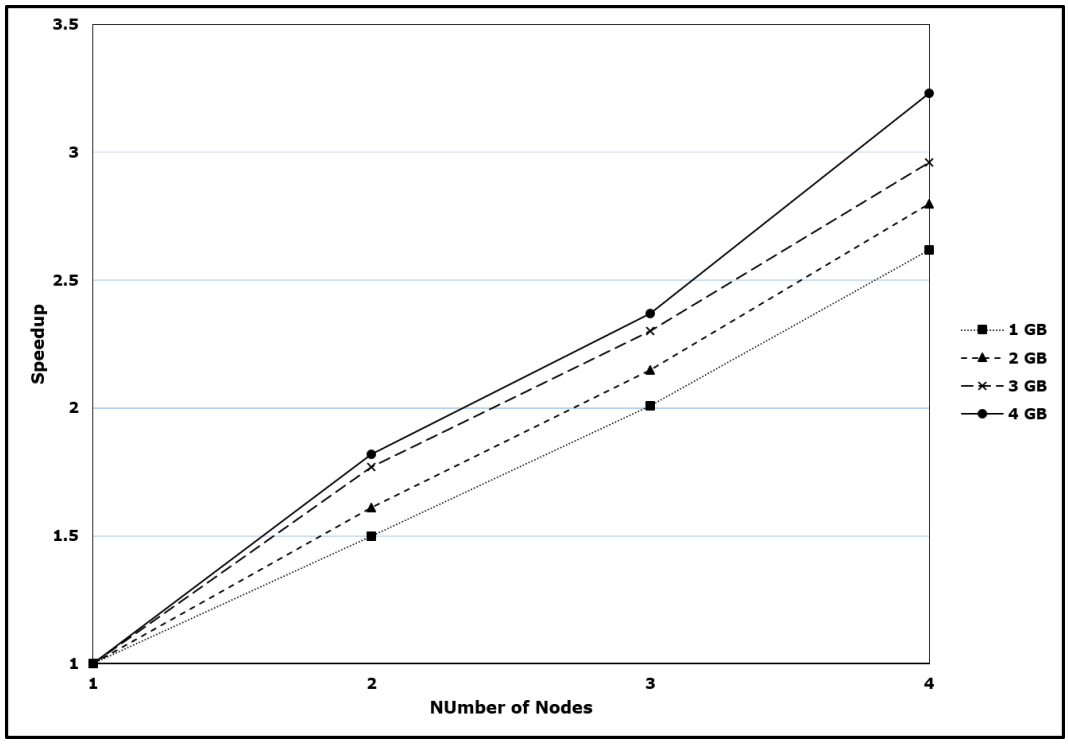

Figure 6.8: Speedup of PPrune on D3 Dataset.

rules than D2. Again, the performance of Create-ACM when processing D3 is similar to that of D2. This is because more items demands longer searches, hence more CPU cost. We conducted three sets of experiments to study the performance of Create-ACM with a dataset D3. The results of the experiments are shown in Figures 6.21, 6.22 and 6.23. The performance of Create-ACM is better when processing D3 than when processing D1 because the percentage of the communication cost is lower.

To study the performance of Create-ACM with a dataset with many number of distinct items, we conducted three sets of experiments using the dataset D4 which has 870 distinct items. As shown in Figure 6.14 the elapsed time of Create-ACM is higher with D4 than with D1 which has only 5 distinct items; but the

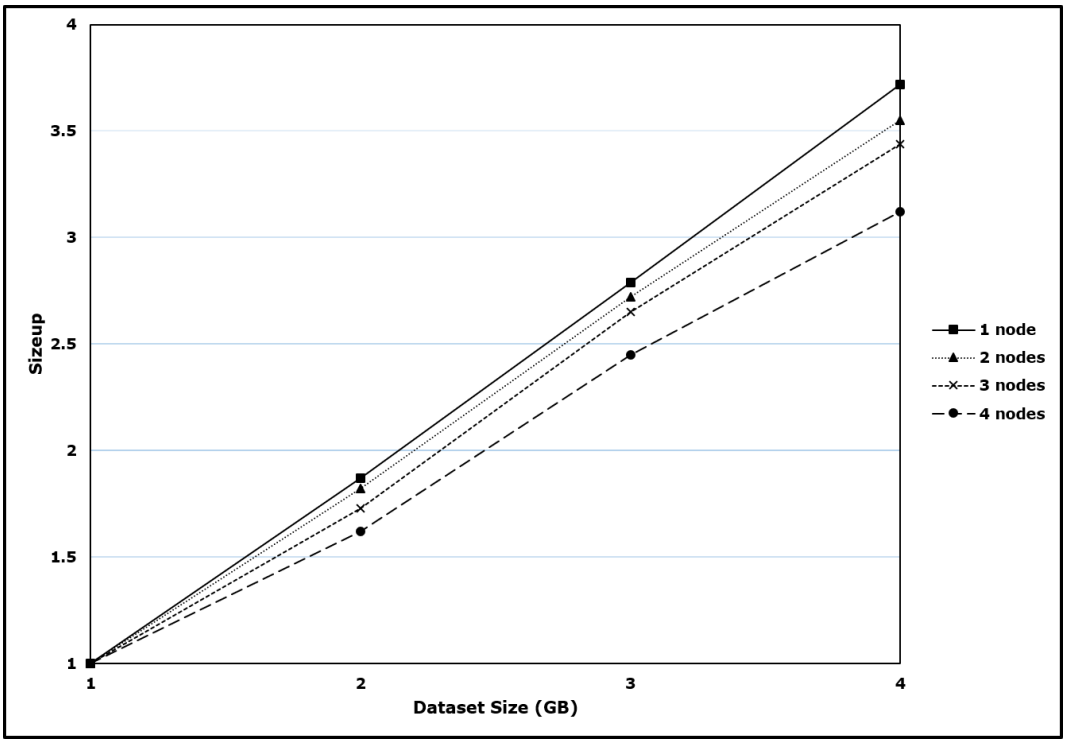

Figure 6.9: Sizeup of PPrune on D3 Dataset.

speedup, scaleup, and sizeup measures of Create-ACM are similar or better when processing D4. This is because more percentage of CPU time is needed to process D4 than to process D1. The results of the experiments are shown in Figures 6.24, 6.25 and 6.26

6.2.3 Performance Evaluation of Compute-Lift Algorithm

To study the performance of Compute-Lift algorithm, we conducted similar experiments that we conducted for PPrune and Create-ACM. To study how the elapsed time of Compute-Lift is affected by different datasets and different number of nodes, we experimented using D1-1, D2-1, D3-1, and D4-1 and we varied the nodes from 1 to 4. The results are shown in Figure 6.27. As expected, as

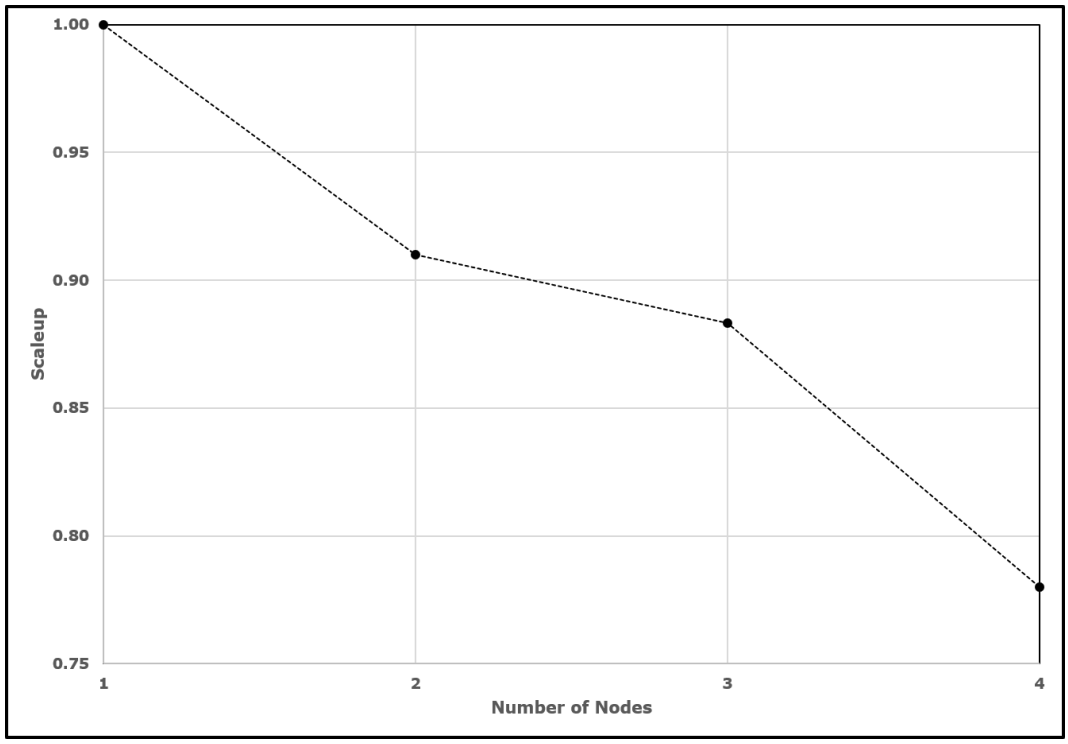

Figure 6.10: Scaleup of PPrune on D3 Dataset.

the number of nodes increases, the elapsed time of Compute-Lift decreases. Also, the elapsed time increases with the number of items in a dataset because more distinct items take longer time to compare. That is why processing D4 takes more time than processing D1 as shown by Figure 6.27.

To study the speedup, sizeup, and scaleup of Compute-Lift we conducted many sets of experiments. The first three sets of experiments were done using D1. Figures 6.28, 6.29, and 6.30 show the obtained results. To measure the speedup of Compute-Lift, we did the same as we did with that of PPrune and Create-ACM. We kept the size of D1 constant and varied the number of nodes from 1 to 4, Figure 6.28. The speedup of Compute-Lift improved as the size of D1 increased. This is because the percentage of the communication cost decreases when the

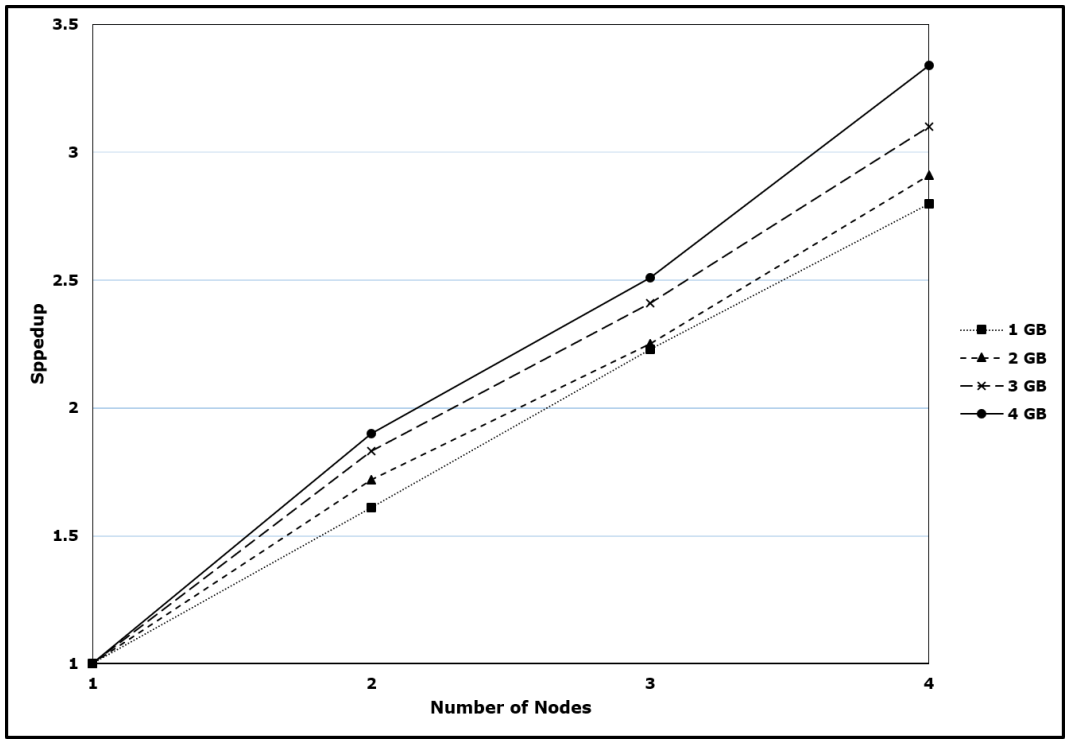

Figure 6.11: Speedup of PPrune on D4 Dataset.

dataset size increases. To measure the sizeup of Compute-Lift, we varied D1 size and the number of nodes as shown in Figure 6.29. Again as that of Create-ACM and PPrune, The results show sublinear performance and it improved as the size of D1 increased. This is because higher percentage of time is spent on CPU and I/O than in communication.

To find the scaleup measure of Compute-Lift, we increased the size of D1 in proportion with the number of nodes. Figure 6.30 shows the scaleup performance of the algorithm. The figure shows the algorithm scales well. Scaleup decreases as the number of nodes and the size of D1 increase but it was always more than 77%.

Again from the above experimental results, we can safely conclude that the

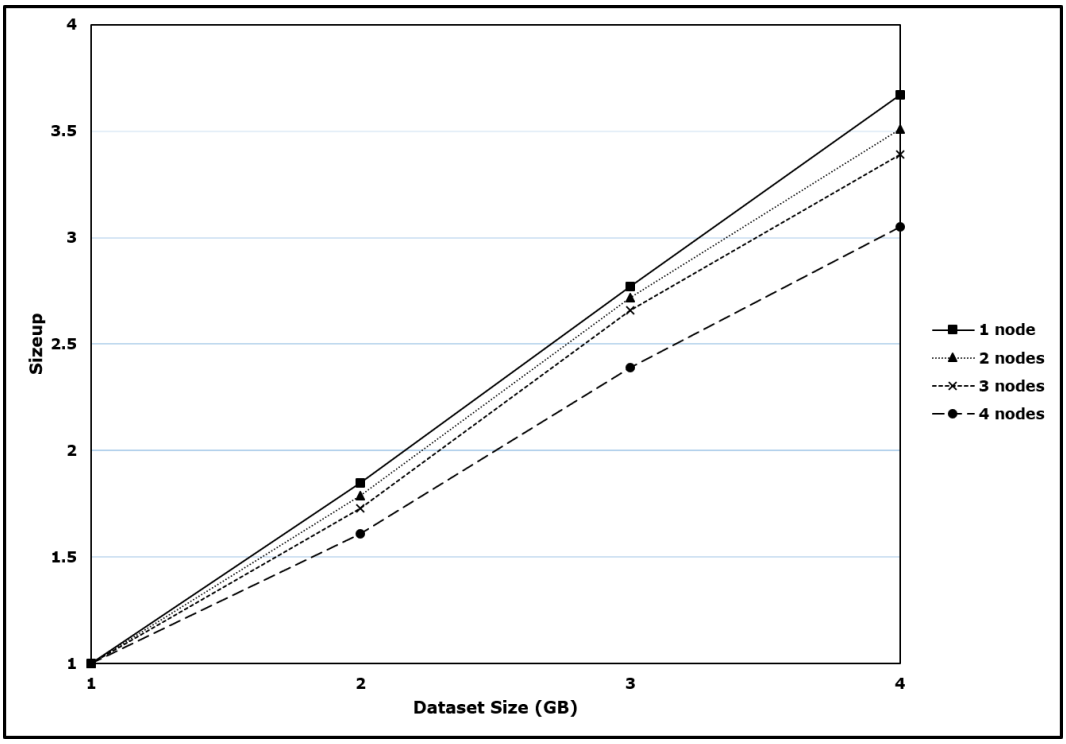

Figure 6.12: Sizeup of PPrune on D4 Dataset.

performance of Compute-Lift was very good when processing D1.

To study the performance of Compute-Lift with a dataset with more items and longer transaction than D1, we conducted three sets of experiments with D2. The results of the experiments are shown in Figures 6.31, 6.32 and 6.33. The performance of Compute-Lift was even better than with D1 because the percentage of the communication cost is less when processing D2 than when processing D1.

As mentioned above, dataset D3 has more items but shorter transactions or rules than D2. The performance of Compute-Lift when processing D3 is similar to that of D2. This is because D3 has more items but shorter rules than D2. We conducted three sets of experiments to study the performance of Compute-Lift with D3. The results of the experiments are shown in Figures 6.34, 6.35 and

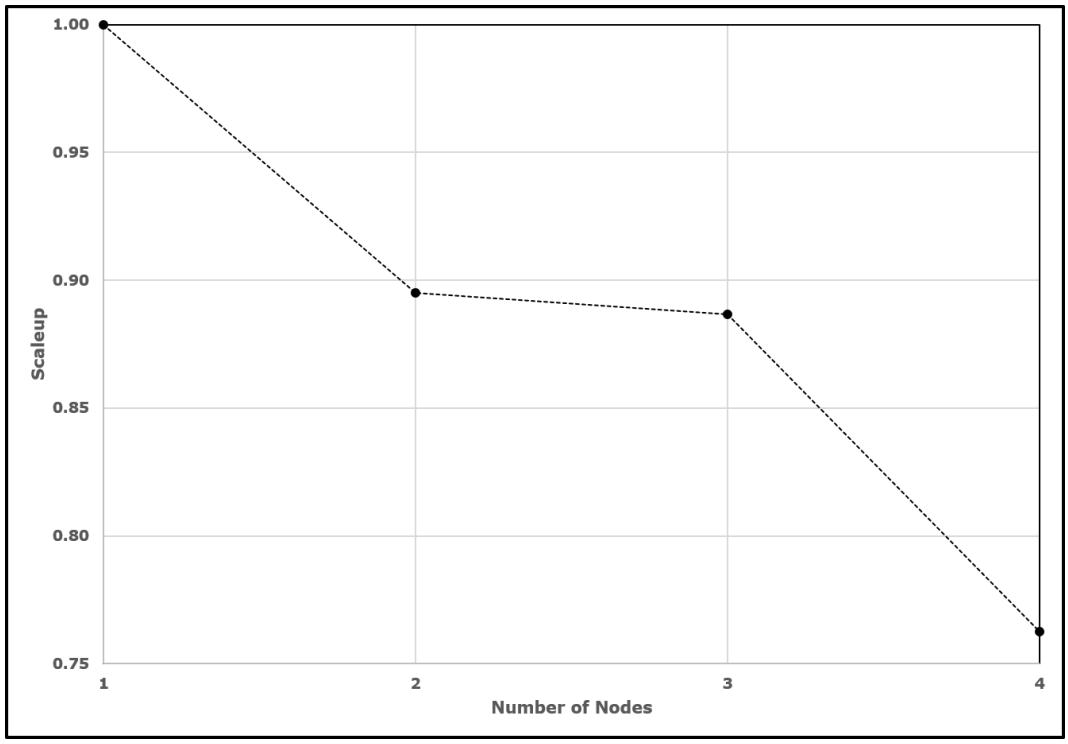

Figure 6.13: Scaleup of PPrune on D4 Dataset.

6.36. The performance of Create-Lift is better with D3 than with D1 because the percentage of the communication cost is lower in comparison to CPU and I/O costs.

To study the performance of Compute-Lift with a dataset with many number of distinct items, we conducted three sets of experiments using the dataset D4. As shown in Figure 6.27 the elapsed time of Compute-Lift is higher with D4 than with D1. but the speedup, scaleup, and sizeup measures of Compute-Lift are similar or better when processing D4. This is because more percentage of CPU time is needed to process D4 than to process D1. The results of the experiments are shown in Figures 6.37, 6.38 and 6.39

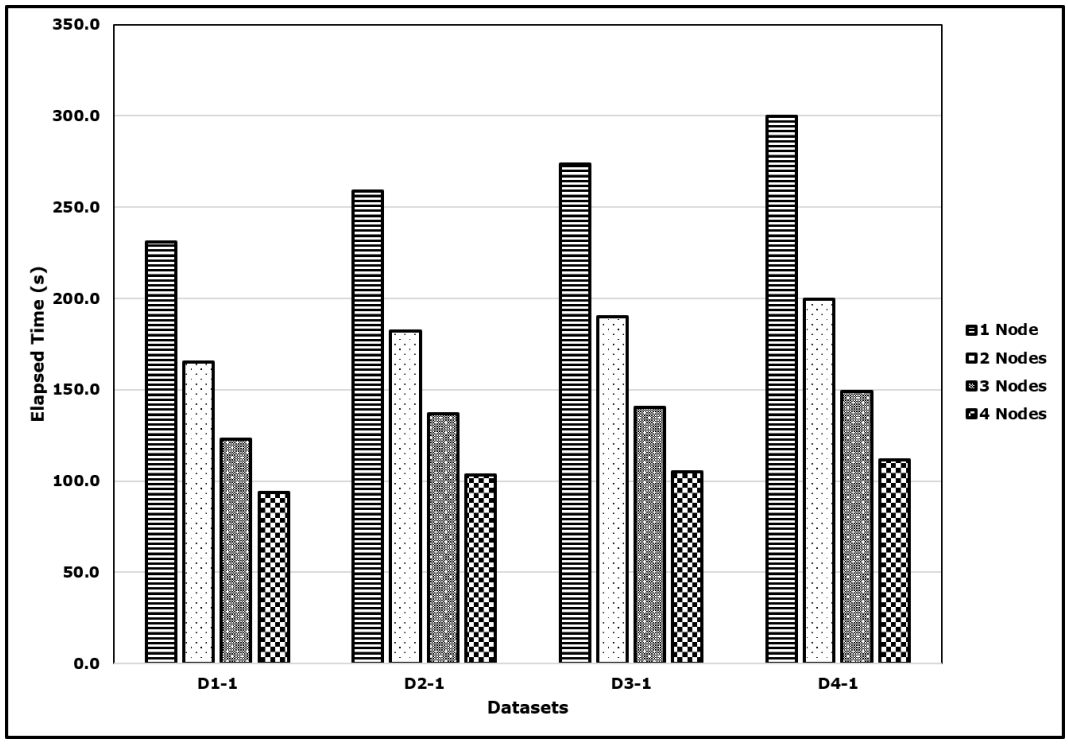

Figure 6.14: Create-ACM elapsed time

6.2.4 Performance Evaluation of Cluster-Rules Algorithm

Cluster-Rules algorithm takes as input the ACM array which contains the lift values of the strong association rules. The size of the ACM array produced as the result of processing D1, D2, D3, and D4 are very small. The elapsed times of the algorithm is in milliseconds. The Cluster-Rules algorithm is similar to PKmeans algorithm so it will have similar performance when given datasets above 1 GB [89]. We tested Cluster-rules and we found its performance to be similar to the one in [87].

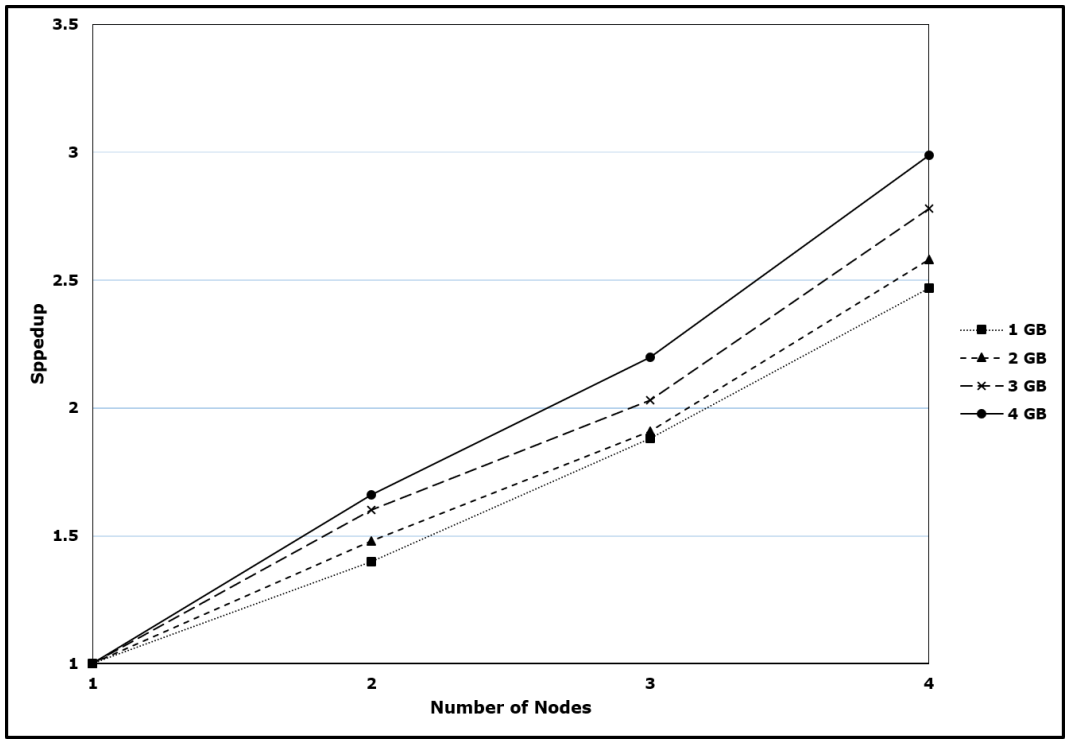

Figure 6.15: Speedup of Create-ACM on D1 Dataset.

6.2.5 Performance Evaluation Using Webdoc Dataset

In all the above mentioned experiments, each dataset was replicated many times. The number of rules generated from the replicated dataset is not huge as the existing huge benchmark datasets. To experiment with huge number of rules we used the Webdoc dataset. We minimized the minimum support so we can generate huge number of rules from the dataset. Some attributes of the dataset are shown in Table 6.2. We added another physical machine (with Intel i7-8750H Processor) to the three shown in 6.1 and we created up to 16 data nodes and then we tested the proposed MapReduce algorithms. The performance of each algorithm is shown in Figure 6.40. As expected, as the number of data nodes increased, the efficiency decreased. The efficiency of a multiprocessors is defined as shown in (6.4). Also,

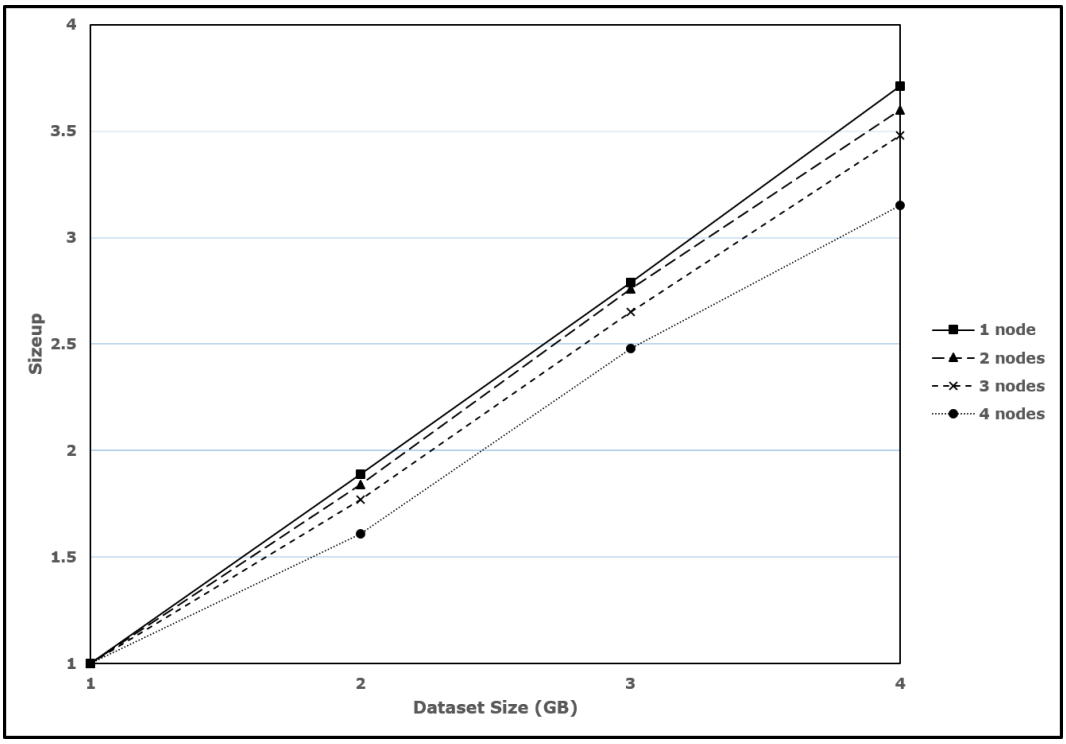

Figure 6.16: Sizeup of Create-ACM on D1 Dataset.

when the number of data nodes used exceeded 8, the speedup decreased. This is because the percentage of the communication cost was too high compared to the CPU and I/O costs. The main reason for the high communication cost is the size of the Webdoc dataset, 1.48 GB, which is very small for a Hadoop machine with more than 4 data nodes. In general, the speedup of a Hadoop cluster with many nodes improves with bigger datasets.

$$Efficiency = \frac{Speedup}{P} \tag{6.4}$$

where P is the number of processors.

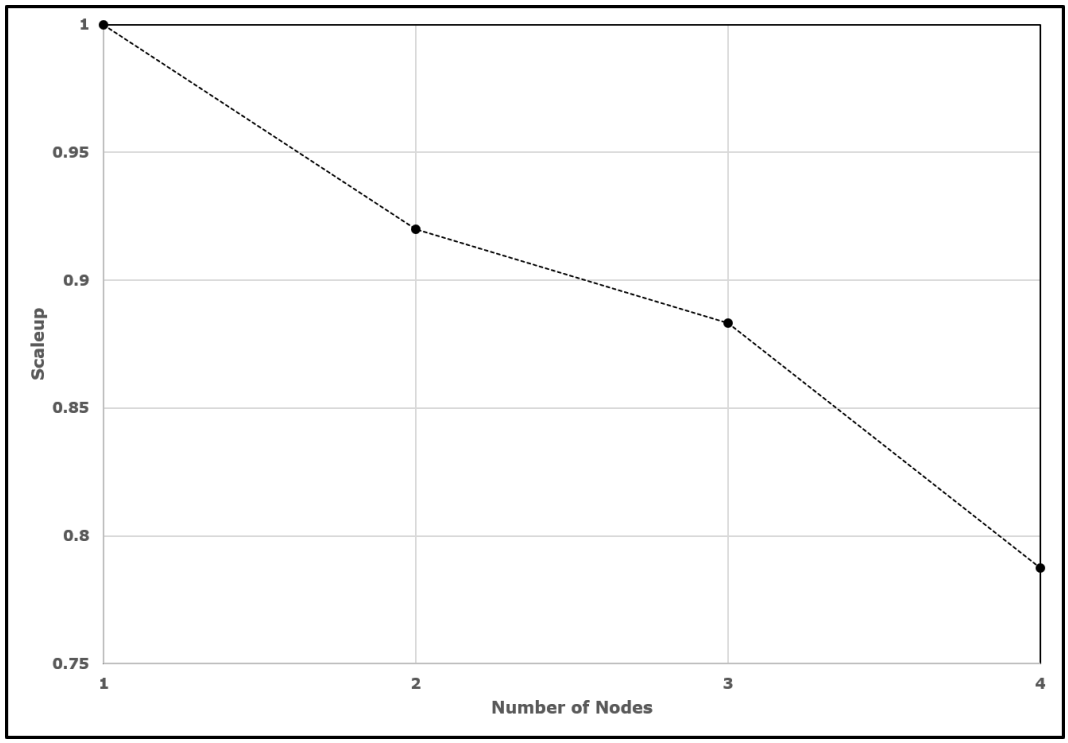

Figure 6.17: Scaleup of Create-ACM on D1 Dataset.

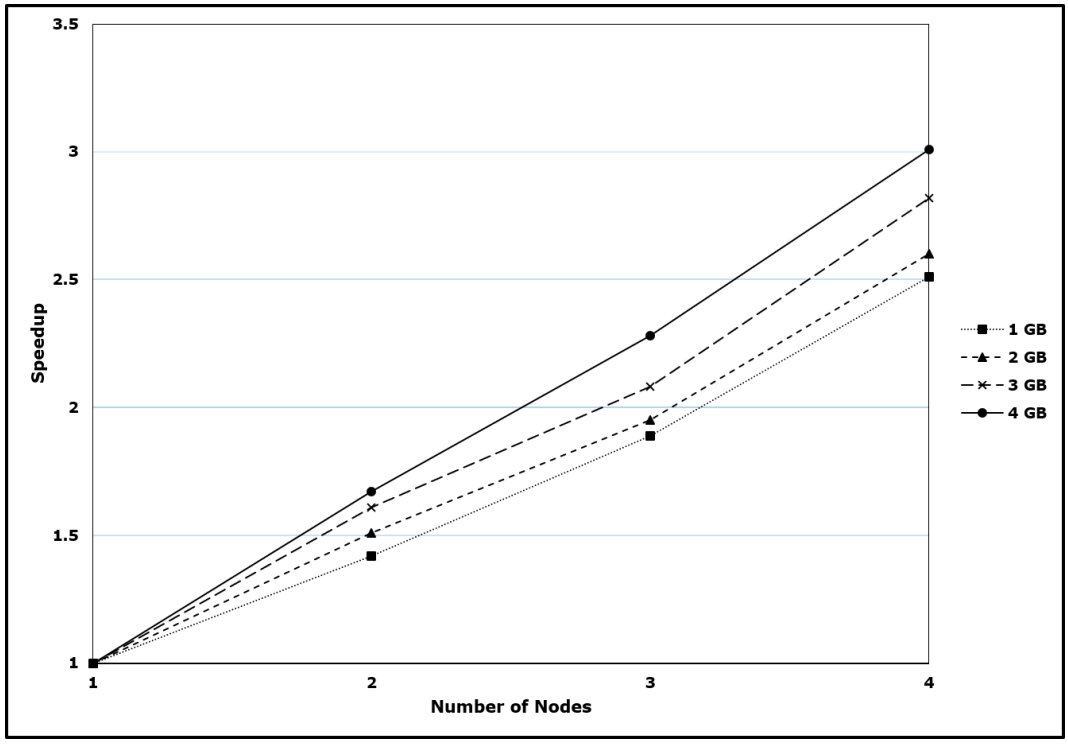

Figure 6.18: Speedup of Create-ACM on D2 Dataset.

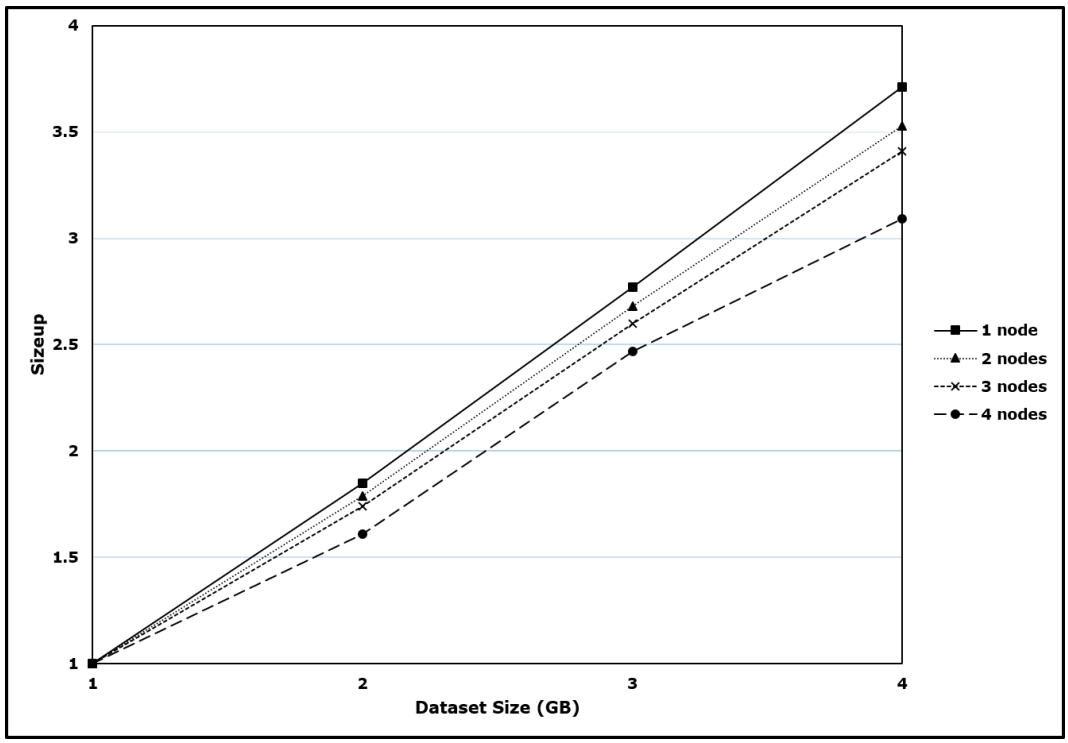

Figure 6.19: Sizeup of Create-ACM on D2 Dataset.

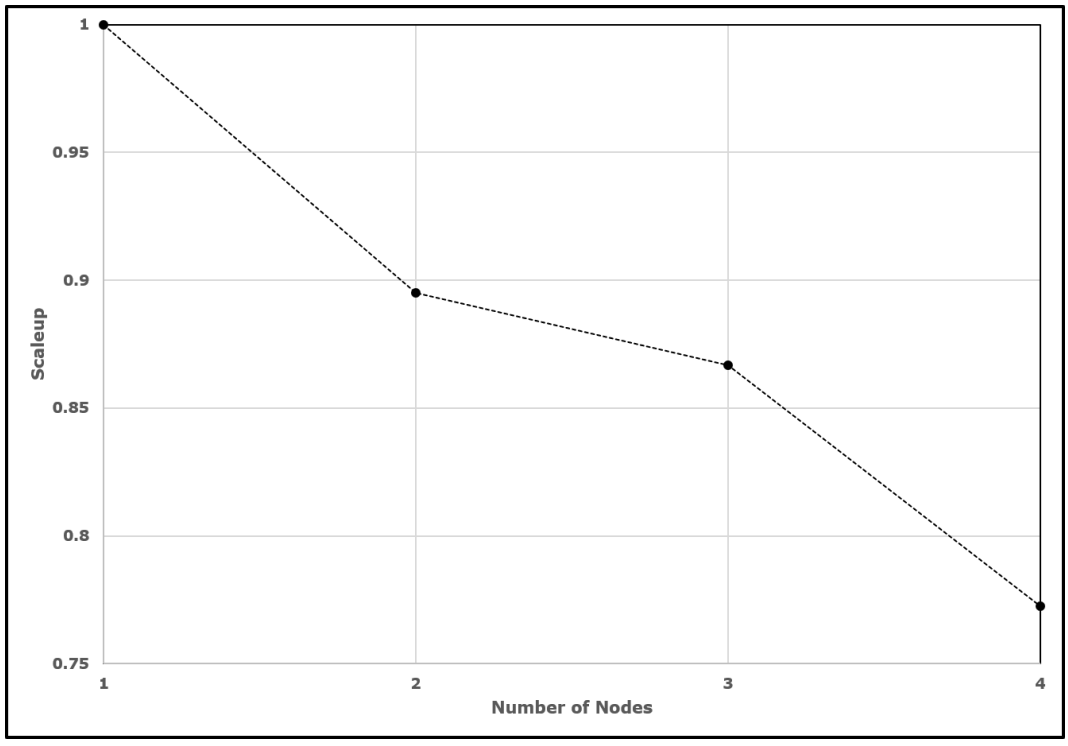

Figure 6.20: Scaleup of Create-ACM on D2 Dataset.

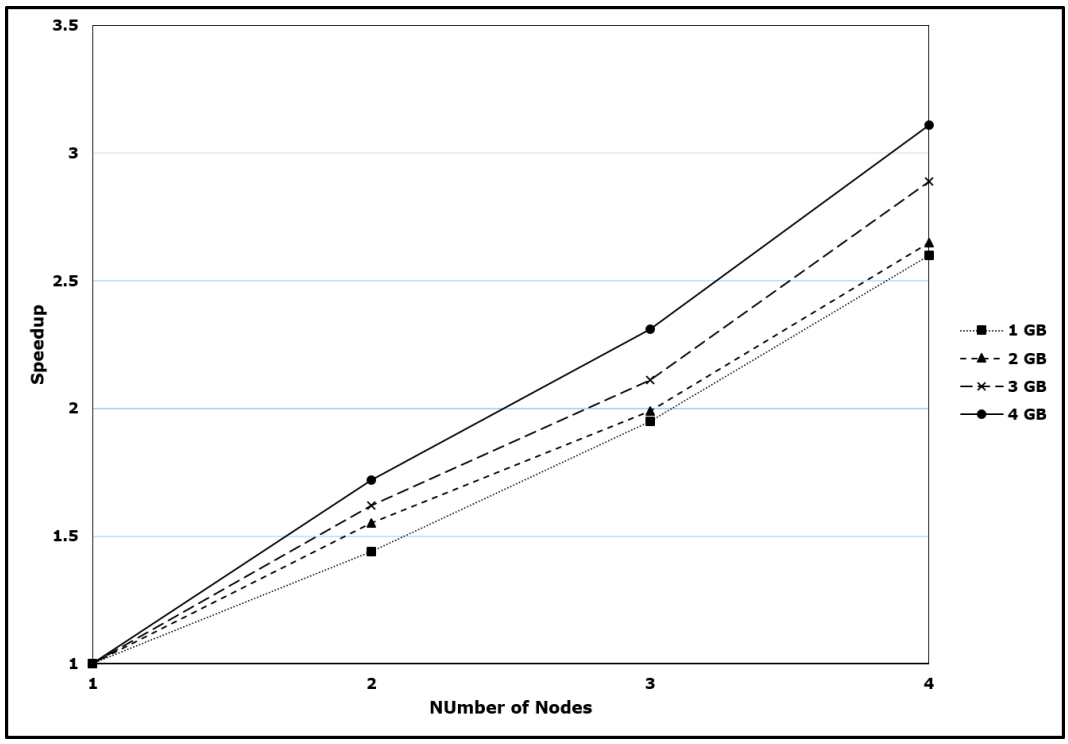

Figure 6.21: Speedup of Create-ACM on D3 Dataset

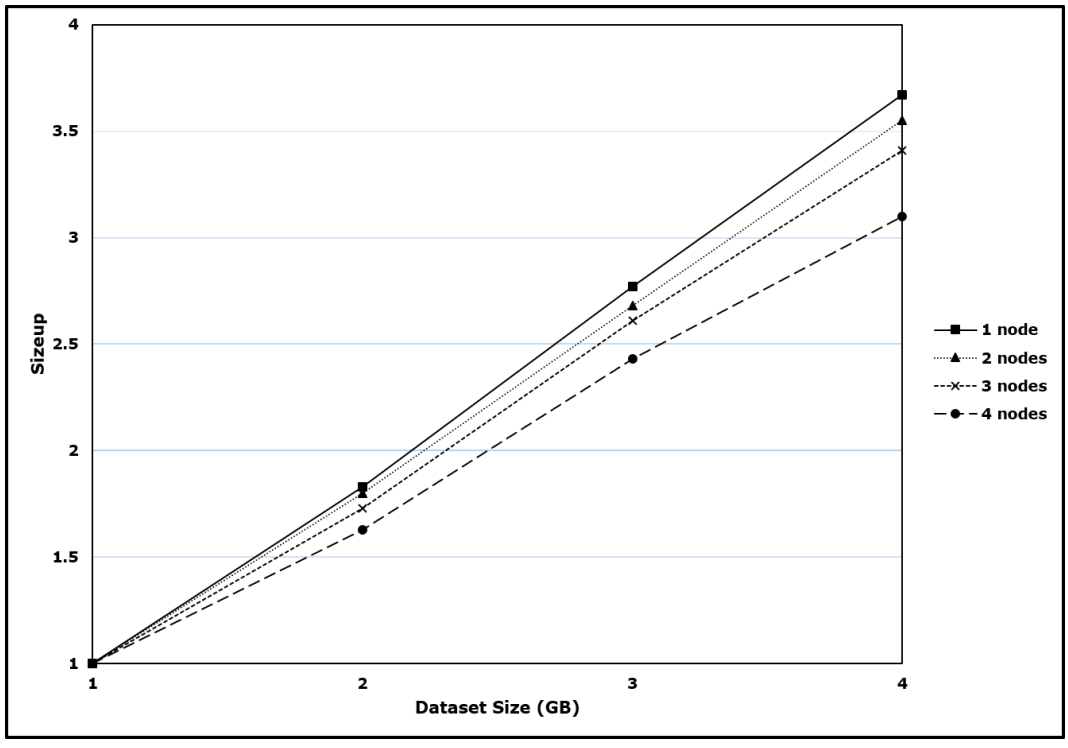

Figure 6.22: Sizeup of Create-ACM on D3 Dataset.

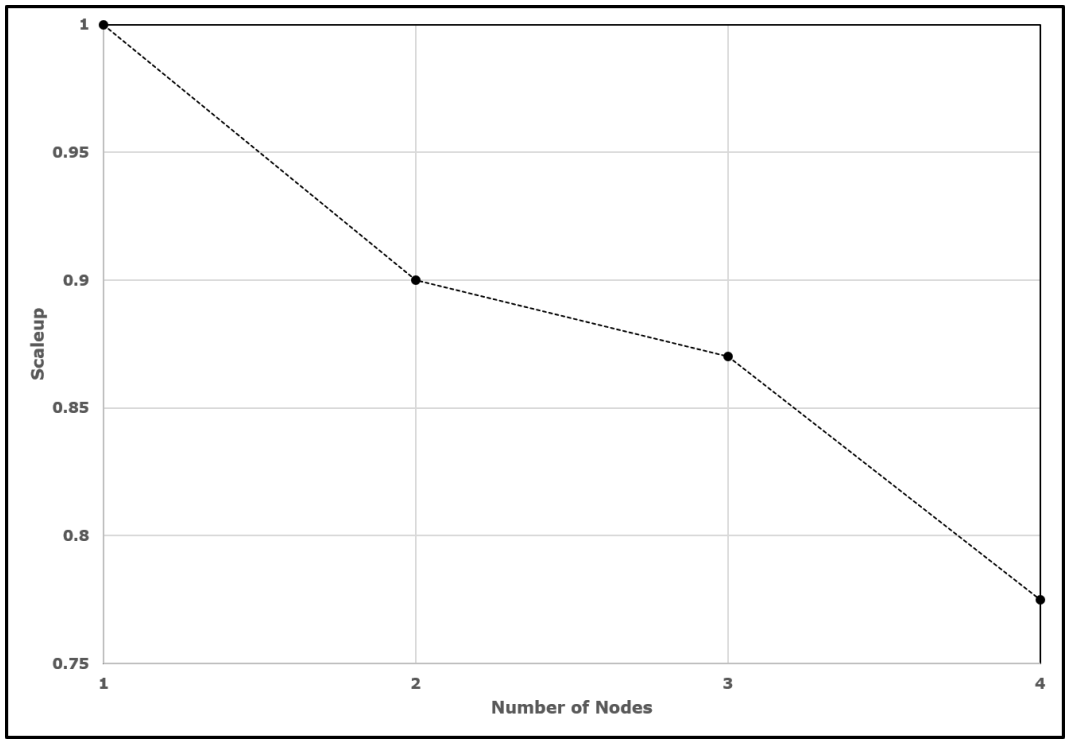

Figure 6.23: Scaleup of Create-ACM on D3 Dataset.

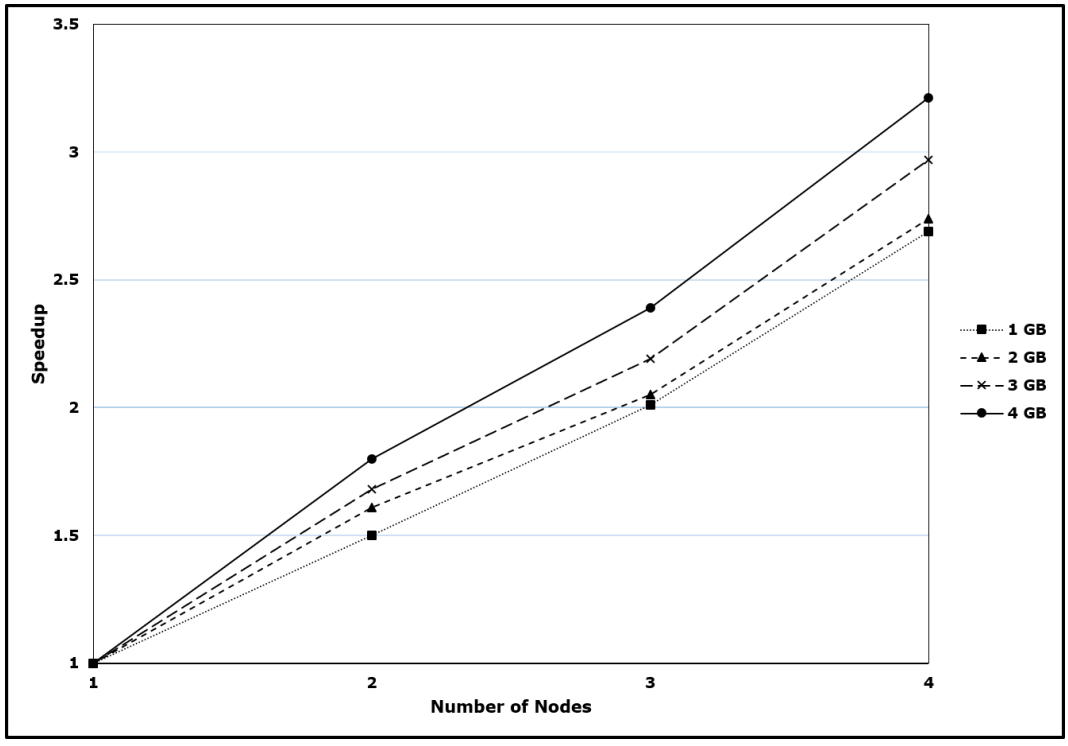

Figure 6.24: Speedup of Create-ACM on D4 Dataset.

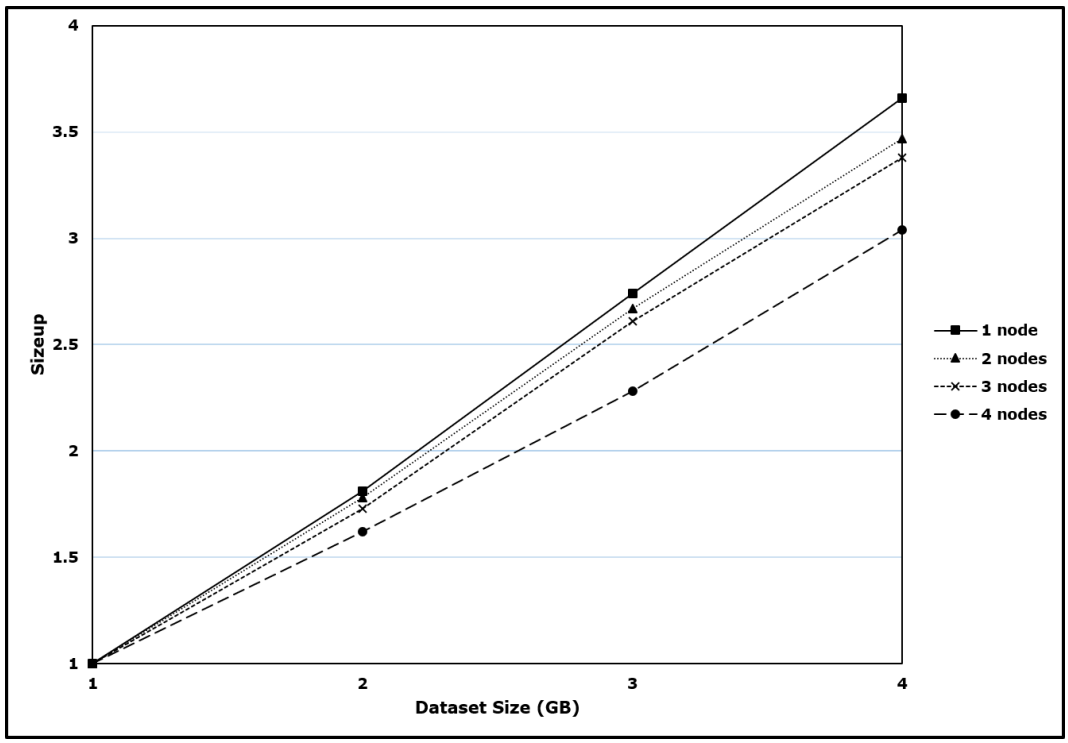

Figure 6.25: Sizeup of Create-ACM on D4 Dataset.

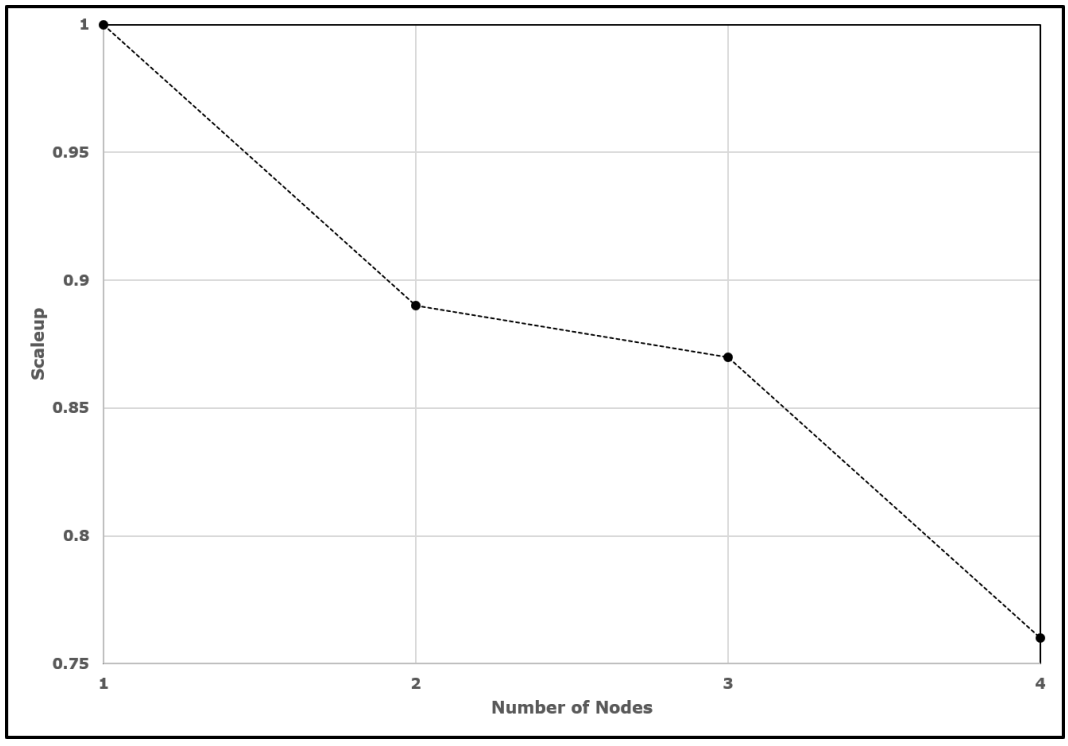

Figure 6.26: Scaleup of Create-ACM on D4 Dataset.

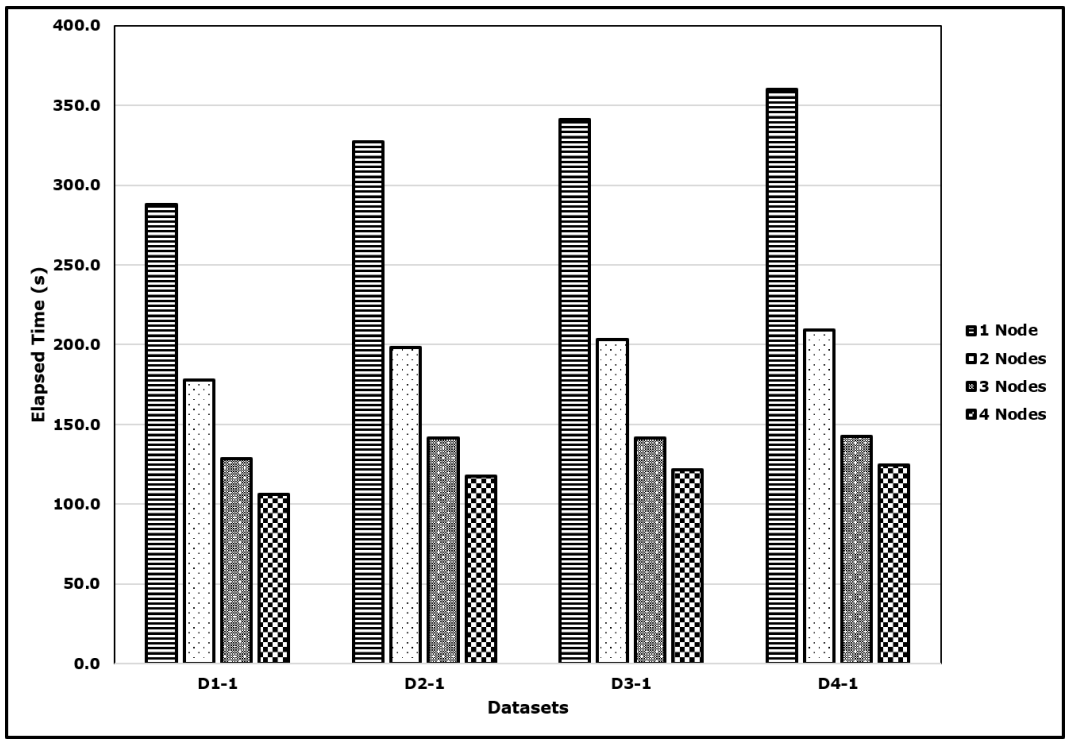

Figure 6.27: Compute-Lift elapsed time.

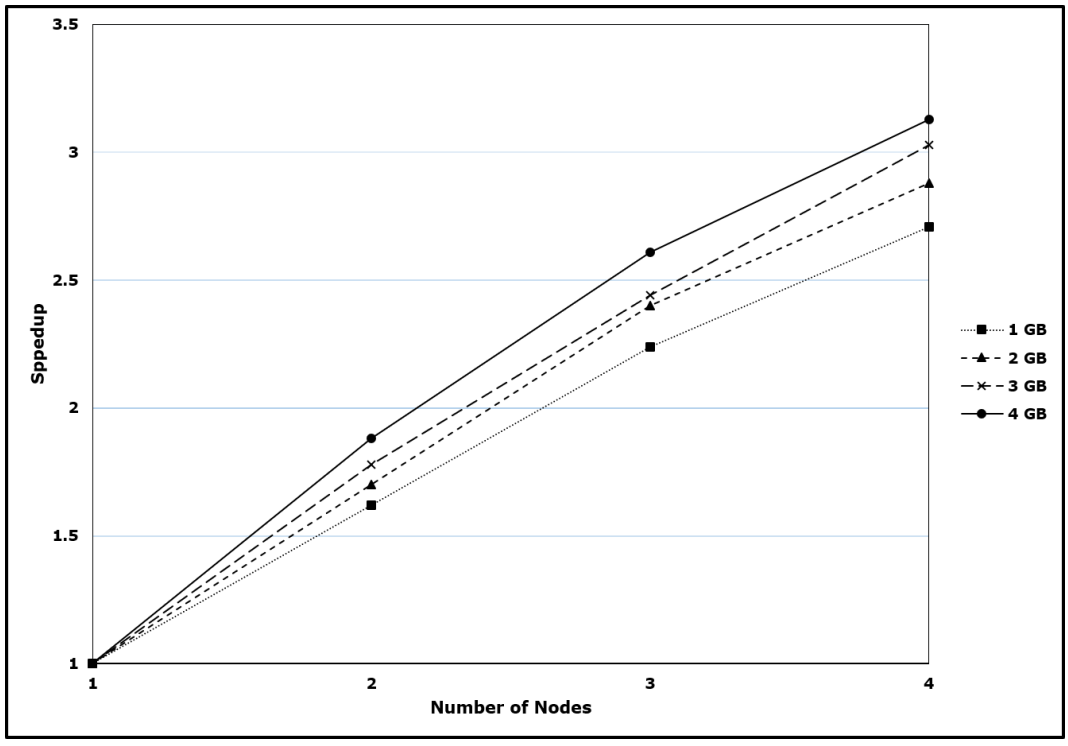

Figure 6.28: Speedup of Compute-Lift on D1 Dataset.

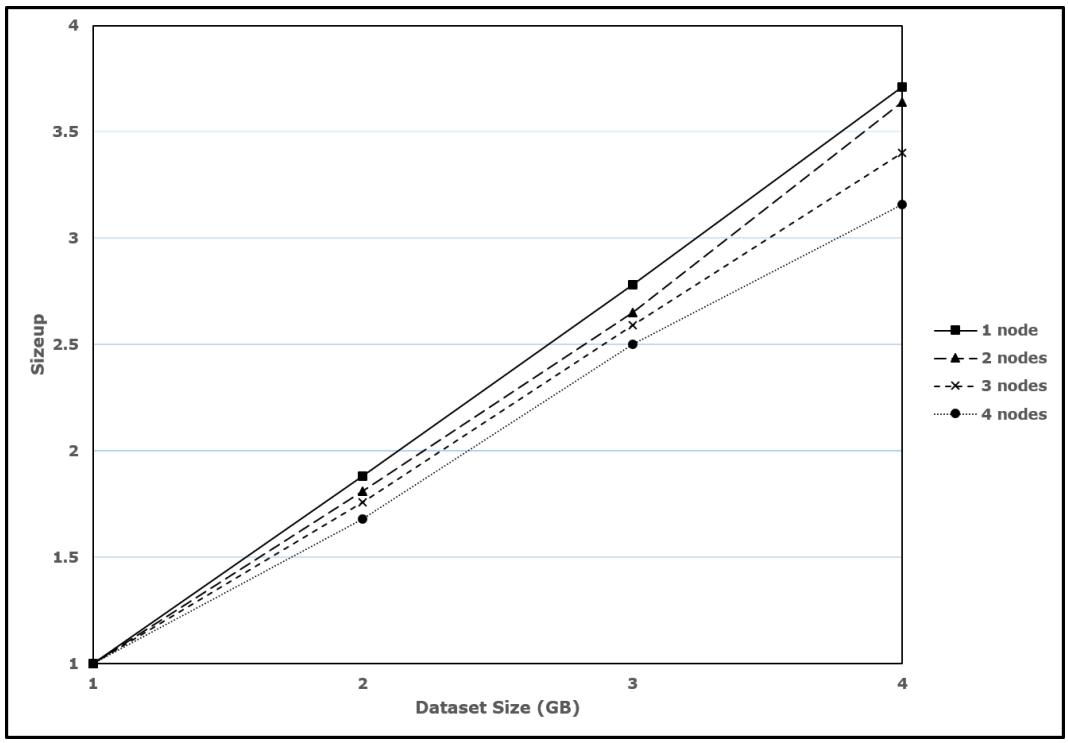

Figure 6.29: Sizeup of Compute-Lift on D1 Dataset.

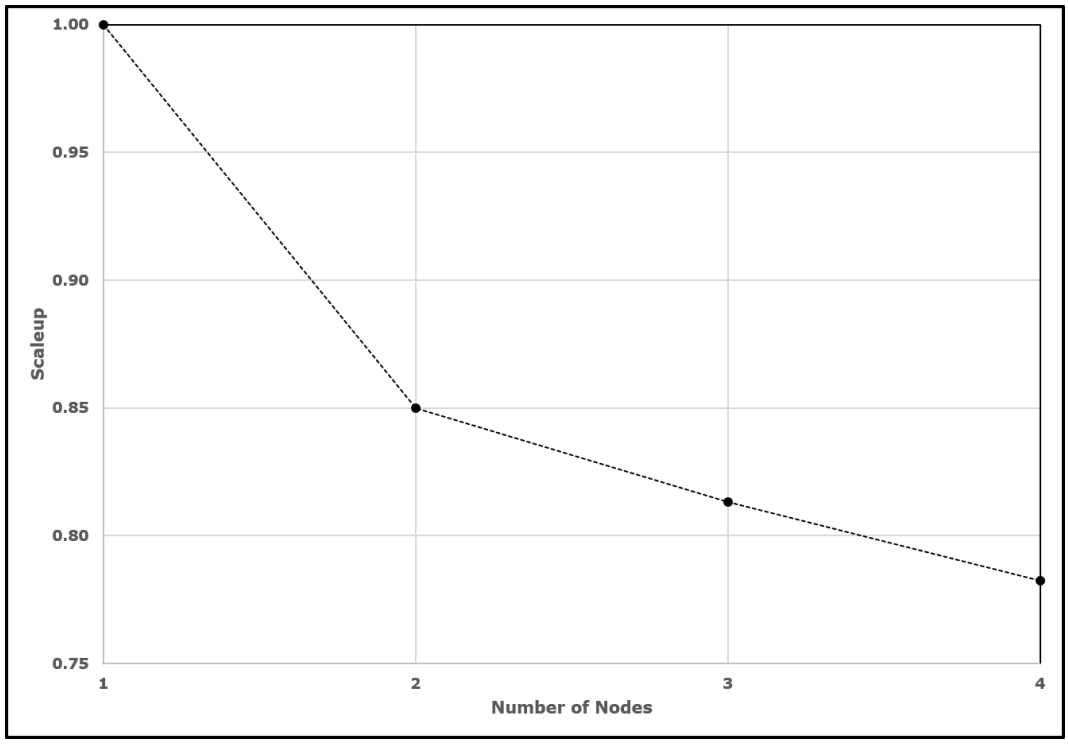

Figure 6.30: Scaleup of Compute-Lift on D1 Dataset.

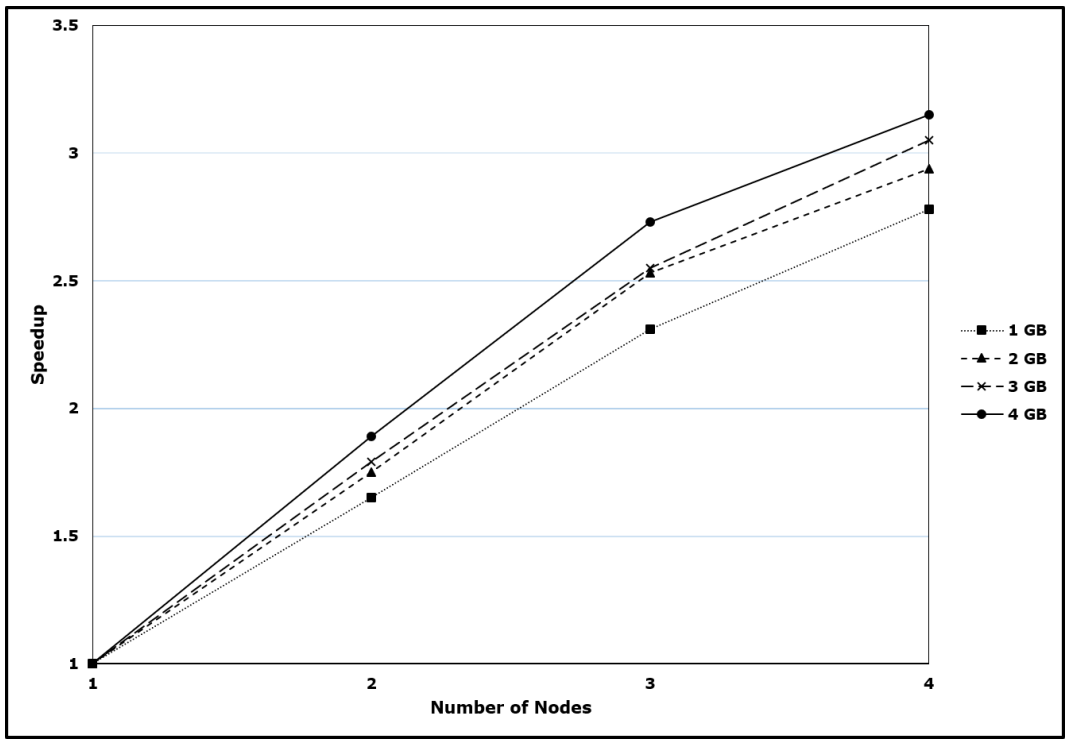

Figure 6.31: Speedup of Compute-Lift on D2 Dataset.

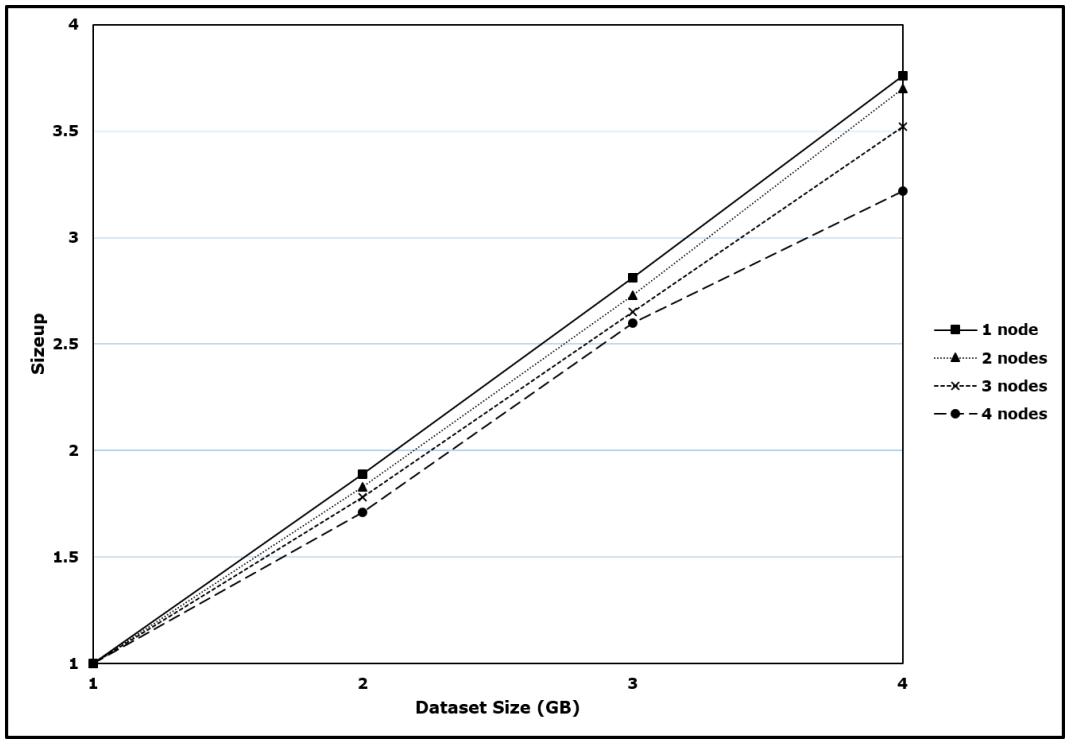

Figure 6.32: Sizeup of Compute-Lift on D2 Dataset.

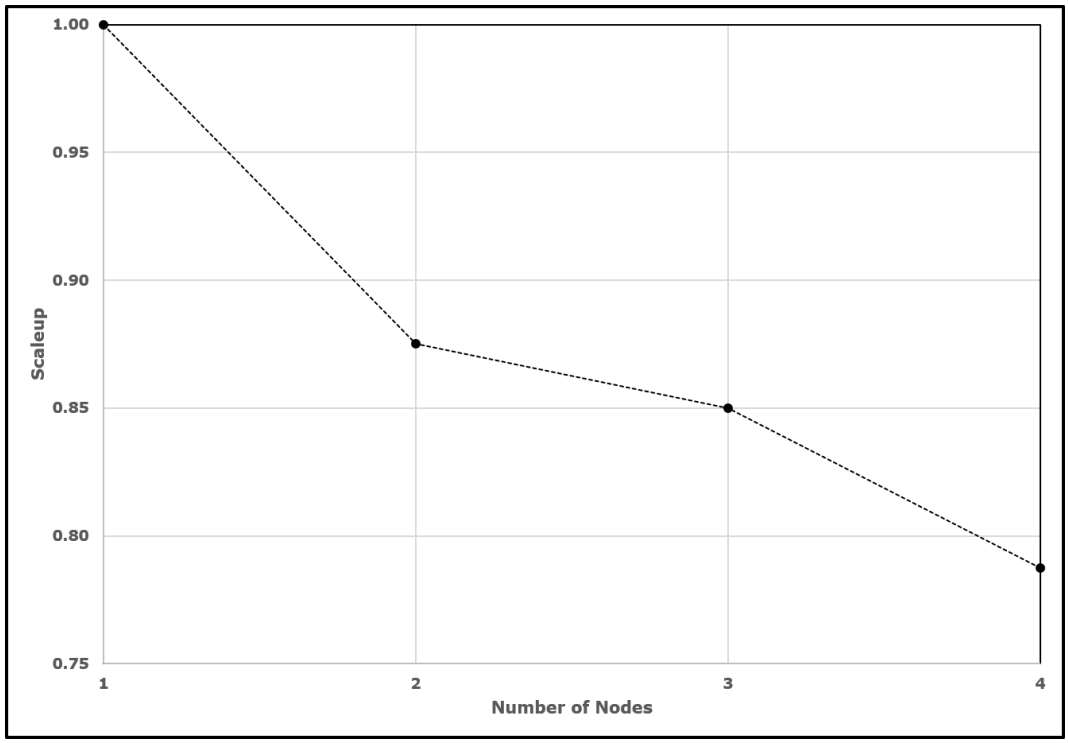

Figure 6.33: Scaleup of Compute-Lift on D2 Dataset.

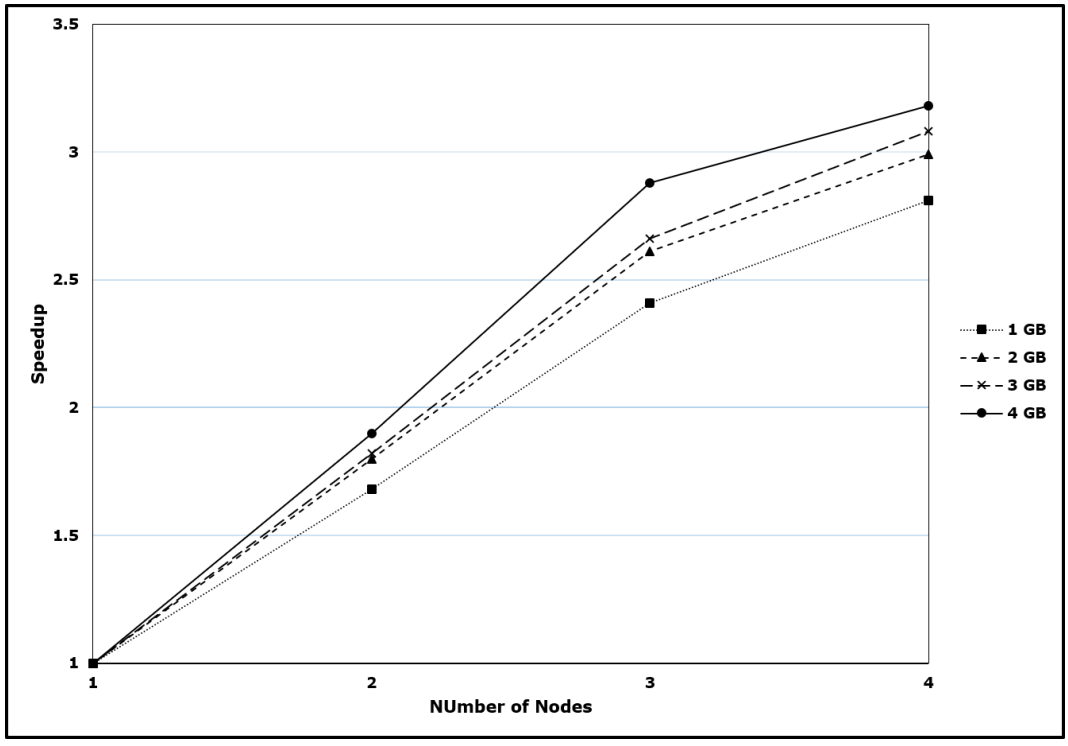

Figure 6.34: Speedup of Compute-Lift on D3 Dataset.

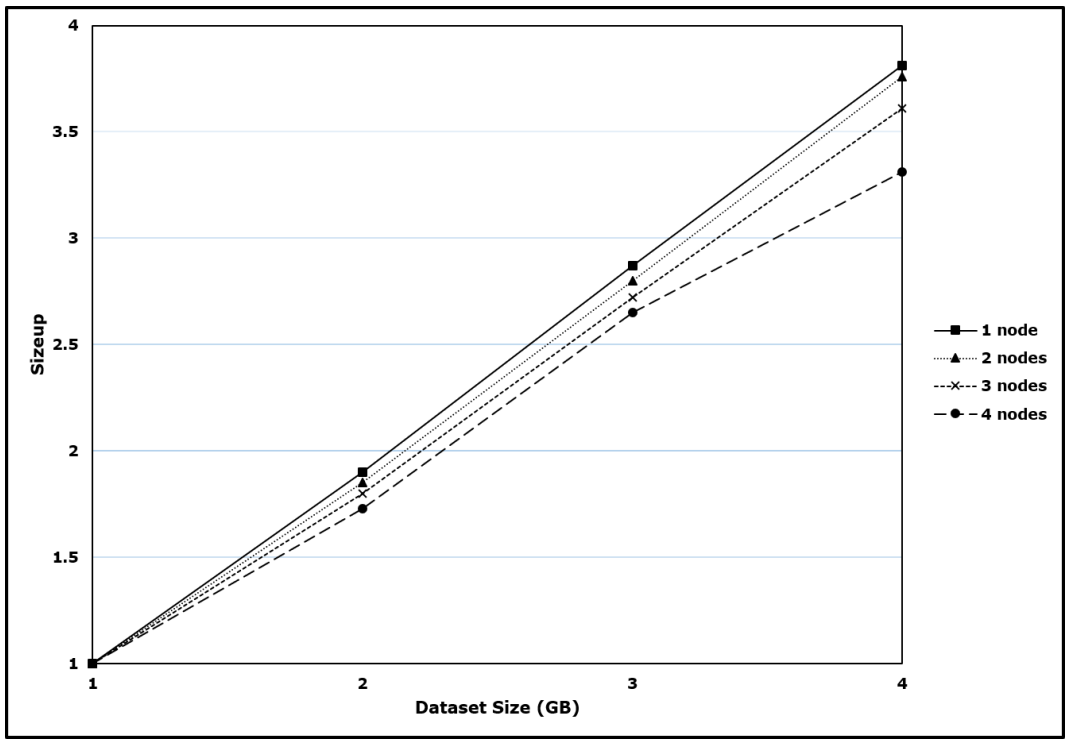

Figure 6.35: Sizeup of Compute-Lift on D3 Dataset.

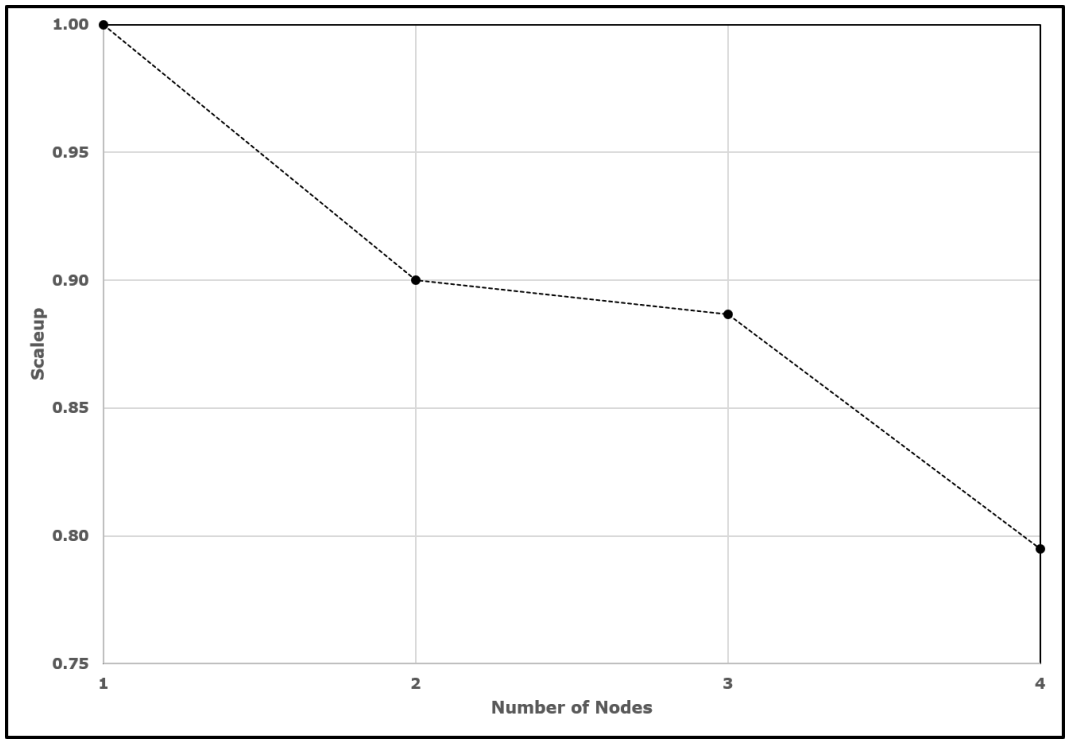

Figure 6.36: Scaleup of Compute-Lift on D3 Dataset.

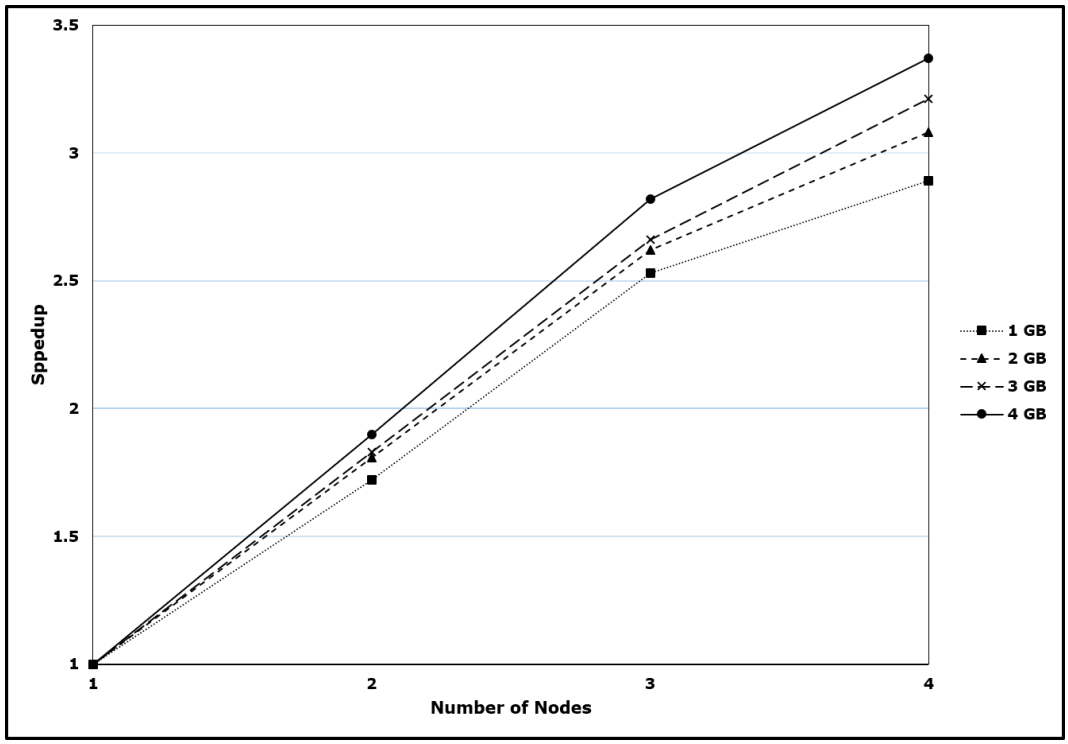

Figure 6.37: Speedup of Compute-Lift on D4 Dataset.

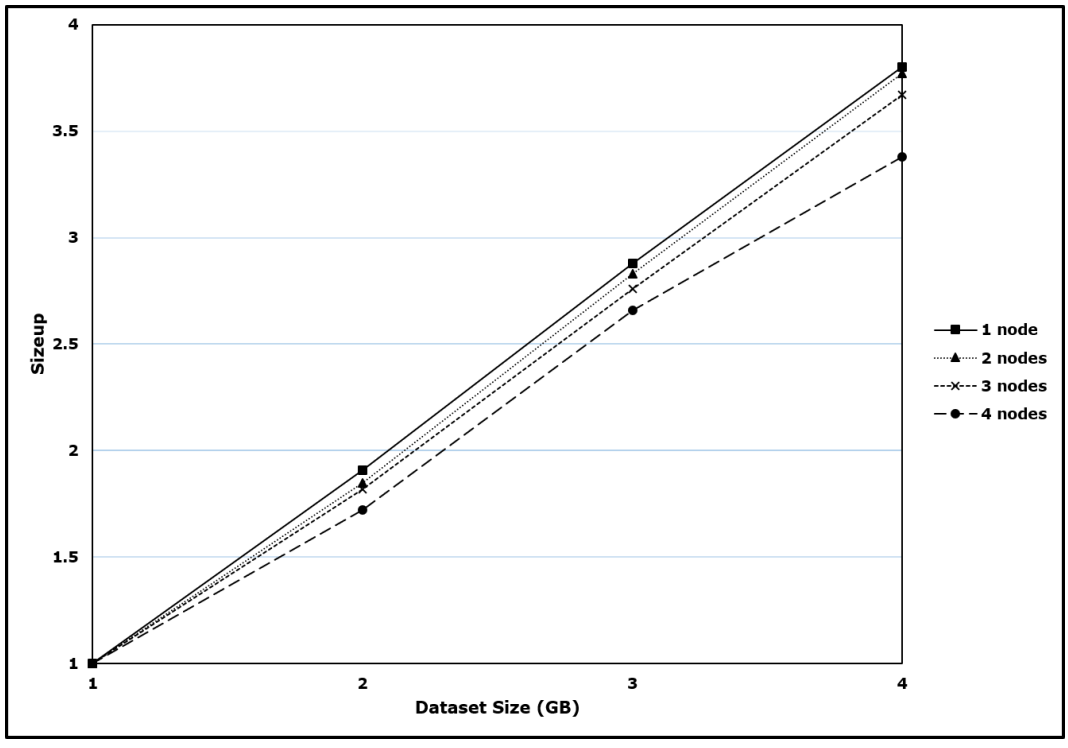

Figure 6.38: Sizeup of Compute-Lift on D4 Dataset.

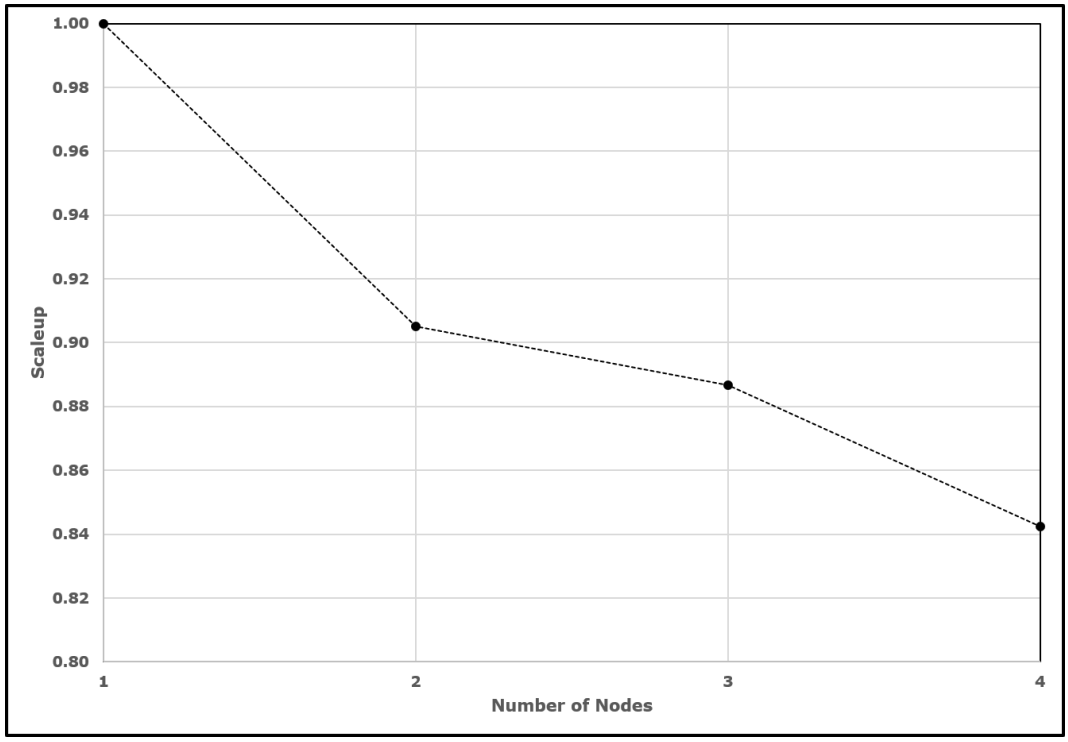

Figure 6.39: Scaleup of Compute-Lift on D4 Dataset.

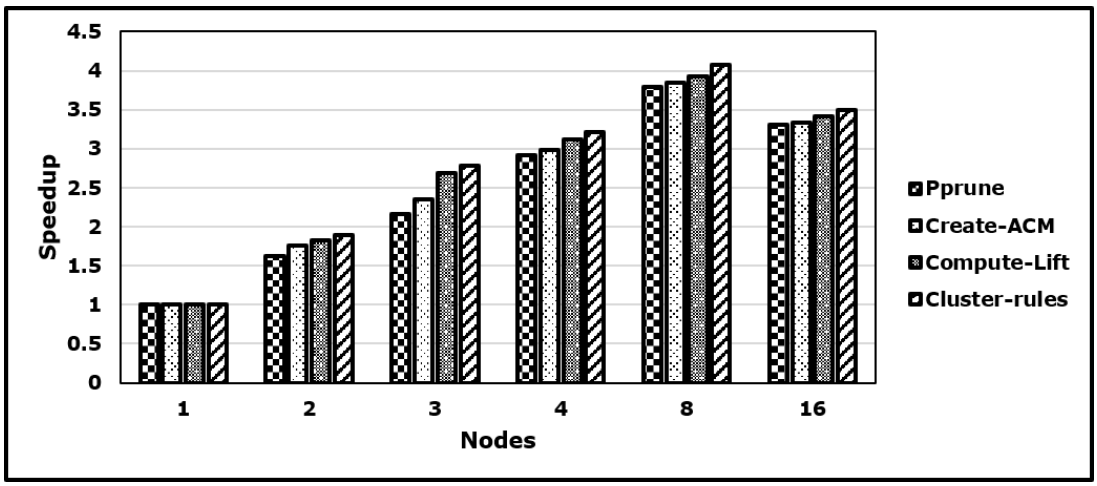

Figure 6.40: Speedup of the proposed algorithms. Dataset used: Webdoc.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Association rule mining is a branch of data mining used to discover frequent patterns, associations, correlations, and other relationships between data items of a dataset. An association rule is of the form $X \rightarrow Y$, where X and Y are sets of data items in the same dataset and $X \cap Y = \emptyset$. In a rule, the set of data items to the left of the arrow, such as X in the above rule, is called the antecedent and the one to the right of the arrow is called the consequent. Association rule mining has many applications in marketing, medicine, finance, security, weather prediction, bio-informatics, and many other fields.

Association rule mining algorithms often discover many rules; but many of these rules are not interesting. A rule to be considered interesting or strong, its

confidence and support must be above pre-specified thresholds. Often, the number of discovered strong association rules is still large, which makes them difficult to inspect, analyze, visualize, or interpret. To solve the problem of large number of association rules, researchers proposed pruning, grouping and clustering rules. Now with the advent of big data technology, the number of discovered association rules will increase drastically. Existing solution of handling large number of discovered association rules is expensive to be applied to the huge number generated from big data. To the best of our knowledge, there is no solution proposed to prune, group or cluster rules generated from big data. This motivated us to propose, in this thesis, the first solution to the huge number of association rules generated from big data.

The proposed solution clusters the rules into user defined number of groups. It consists of four MapReduce Algorithms. The first algorithm is called *PPrune* which clusters association rules based on their structure. The idea of this algorithm is based on structural rule cover which is explained in [14]. A *rule cover* states that if X, Y, and Z are three itemsets of a given transactional database D, then Φ(X,Y,Z) $\subseteq$ Φ(X,Y), *where* Φ(x) is the set of transactions in D that match the itemset x. In other words, the transactions that match the rule $X, Z \rightarrow Y$ are contained among the transactions that match $X \rightarrow Y$. If rules such as $X, Z \rightarrow Y$ are removed from a rule cover, then the remaining set which contains $X \rightarrow Y$ is a rule cover. A structural rule cover consists of the most general rules of the original set of rules. PPrune is an optional algorithm which runs before

the main clustering algorithm. It is needed if the given number of rules is too big to handle. The second MapReduce algorithm creates a matrix called ACM (Antecedent-Consequent Matrix) which is indexed by the distinct antecedents and consequents of the strong association rules. ACM is used to store lift values of the strong association rules. Let the set $A = \{A_1, A_2, \ldots A_{|A|}\}$ be the set of all antecedents and $C = \{C_1, C_2, \ldots C_{|C|}\}$ be the set of all consequents in the strong association rules, where $|A|$ and $|C|$ are the number of distinct antecedents and consequents respectively. Element $m_{i,j}$ of ACM contains the lift value of the rule $A_i \rightarrow \mathrm{C}_j$. The third MapReduce algorithm computes the lift value of each strong association rule and stores it in its corresponding ACM element. The last MapReduce algorithm clusters the association rules with different antecedents based on their lift values. Lift measures the correlation of a rule with its antecedent and its consequent. The proposed clustering approach is based on the assumption that rules with antecedents that are highly correlated with the same set of consequents are similar and thus they should be in the same cluster [87]. Unlike the existing clustering approaches, this approach clusters antecedents containing itemsets which are rarely purchased together. This is because of the high correlation they have with the same consequents.

The proposed algorithms were experimented in Hadoop cluster. In the experiments both real and systhetic benchmark datasets were used. The performance of the proposed algorithms was evaluated using four measures, namely elapsed time, scaleup, sizeup, and speedup. All the algorithms showed high performance. For

example, the lowest scalability obtained by the experiments was 77%.

7.2 Future Work

As future work, we plan to accomplish the following four ideas. First, we plan to test the proposed algorithms using more nodes and bigger dataset; Second, we plan to use a visualization software to see the distribution of clusters; Third, we plan to use a functions other than lift. Finally, we plan to extend our algorithms to handle rules with any number of items in their consequents.

REFERENCES

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The kdd process for extracting useful knowledge from volumes of data," *Communications of the ACM*, vol. 39, no. 11, pp. 27–34, 1996.

[2] N. Viriyarattanaporn, "Near closed frequent itemsets to accelerate the generation of association rules in a data stream environment," Ph.D. dissertation, Auckland University of Technology, 2010.

[3] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," in *International Conference on Database Theory*. Springer, 1999, pp. 398–416.

[4] P.-N. Tan, *Introduction to data mining*. Pearson Education India, 2018.

[5] T. Ivanov, N. Korfiatis, and R. V. Zicari, "On the inequality of the 3v's of big data architectural paradigms: A case for heterogeneity," *arXiv preprint arXiv:1311.0805*, 2013.

[6] A. Vyas, S. Ram, and M. E Scholar, "Comparative study of mapreduce frameworks in big data analytics," 12 2017.

[7] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Prentice hall Upper Saddle River, 2003, vol. 2, no. 9.

[8] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[9] F. J. V. Martín, J. L. C. Sequera, and M. A. N. Huerga, "Using data mining techniques to discover patterns in an airline's flight hours assignments," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 13, no. 2, pp. 45–62, 2017.

[10] A. An, S. M. Khan, and X. Huang, "Objective and subjective algorithms for grouping association rules." in *ICDM*, vol. 3, 2003, p. 477.

[11] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *ACM Sigmod*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[12] S. Chawla, J. G. Davis, and G. Pandey, "On local pruning of association rules using directed hypergraphs." in *ICDE*, vol. 4, 2004, pp. 832–841.

[13] A. Berrado and G. C. Runger, "Using metarules to organize and group discovered association rules," *Data mining and knowledge discovery*, vol. 14, no. 3, pp. 409–431, 2007.

[14] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hätönen, and H. Mannila, "Pruning and grouping discovered association rules," 1995.

[15] B. Lent, A. Swami, and J. Widom, "Clustering association rules," in *Data Engineering, 1997. Proceedings. 13th International Conference on.* IEEE, 1997, pp. 220–231.

[16] J. Pokorny, "Nosql databases: a step to database scalability in web environment," *International Journal of Web Information Systems*, vol. 9, no. 1, pp. 69–82, 2013.

[17] J. Sangeetha and V. S. J. Prakash, "A survey on big data mining techniques," *International Journal of Computer Science and Information Security*, vol. 15, no. 1, p. 482, 2017.

[18] A. A. Pandagale and A. R. Surve, "Hadoop-hbase for finding association rules using apriori mapreduce algorithm," in *Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference on.* IEEE, 2016, pp. 795–798.

[19] Y. Chen, F. Li, and J. Fan, "Mining association rules in big data with NGEP," *Cluster Computing*, vol. 18, no. 2, pp. 577–585, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10586-014-0419-3

[20] Y. Djenouri, A. Bendjoudi, D. Djenouri, A. Belhadi, and N. Nouali-Taboudjemat, "New gpu-based swarm intelligence approach for reducing big association rules space," in *2017 IEEE SmartWorld, Ubiquitous Intelligence*

& Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). IEEE, 2017, pp. 1–6.

[21] G. Piateski and W. Frawley, *Knowledge discovery in databases*. MIT press, 1991.

[22] S. Kotsiantis and D. Kanellopoulos, "Association rules mining: A recent overview," *GESTS International Transactions on Computer Science and Engineering*, vol. 32, no. 1, pp. 71–82, 2006.

[23] B. Liu, W. Hsu, S. Chen, and Y. Ma, "Analyzing the subjective interestingness of association rules," *IEEE Intelligent Systems and their Applications*, vol. 15, no. 5, pp. 47–55, 2000.

[24] P.-N. Tan, V. Kumar, and J. Srivastava, "Selecting the right objective measure for association analysis," *Information Systems*, vol. 29, no. 4, pp. 293–313, 2004.

[25] A. A. Freitas, "On rule interestingness measures," *Knowledge-Based Systems*, vol. 12, no. 5-6, pp. 309–315, 1999.

[26] G. Yongmei and B. Fuguang, "The research on measure method of association rules mining," *International Journal of Database Theory and Application*, vol. 8, no. 2, pp. 245–258, 2015.

[27] Q. Yan-xia, "Measurement of novelty: Factor of evaluation for the association rules [j]," *Application Research of Computers*, vol. 1, 2004.

[28] E. Duneja and A. Sachan, "A survey on frequent itemset mining with association rules," *International Journal of Computer Applications*, vol. 46, no. 23, pp. 18–24, 2012.

[29] J. Pei, J. Han, R. Mao *et al.*, "Closet: An efficient algorithm for mining frequent closed itemsets." in *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, vol. 4, no. 2, 2000, pp. 21–30.

[30] M. J. Zaki and C.-J. Hsiao, "Charm: An efficient algorithm for closed itemset mining," in *Proceedings of the 2002 SIAM international conference on data mining*. SIAM, 2002, pp. 457–473.

[31] S. S. Lodhi, P. Arya, and D. Vishwakarma, "Frequent itemset mining technique in data mining," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 5, pp. pp–395, 2012.

[32] R. J. Bayardo Jr, "Efficiently mining long patterns from databases," *ACM Sigmod Record*, vol. 27, no. 2, pp. 85–93, 1998.

[33] D. Burdick, M. Calimlim, and J. Gehrke, "Mafia: A maximal frequent itemset algorithm for transactional databases," in *Data Engineering, 2001. Proceedings. 17th International Conference on*. IEEE, 2001, pp. 443–452.

[34] R. C. Agarwal, C. C. Aggarwal, and V. Prasad, "A tree projection algorithm for generation of frequent item sets," *Journal of parallel and Distributed Computing*, vol. 61, no. 3, pp. 350–371, 2001.

[35] M. Rajalakshmi, D. T. Purusothaman, and D. R. Nedunchezhian, "Maximal frequent itemset generation using segmentation approach," *arXiv preprint arXiv:1109.2427*, 2011.

[36] Q. Zou, W. W. Chu, and B. Lu, "Smartminer: A depth first algorithm guided by tail information for mining maximal frequent itemsets," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on.* IEEE, 2002, pp. 570–577.

[37] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[38] Y. Zeng, S. Yin, J. Liu, and M. Zhang, "Research of improved fp-growth algorithm in association rules mining," *Scientific Programming*, vol. 2015, p. 6, 2015.

[39] Z. Wang, G. Luo, Y. Hu, and Z. Wang, "Association rule mining with parallel frequent pattern growth algorithm on hadoop."

[40] N. Rajkumar, K. R. Vimal, M. Nathiya, and K. Silambarasan, "Mining association rules in big data for e-healthcare information system," *Research*

Journal of Applied Sciences, Engineering and Technology, vol. 8, no. 8, pp. 1002–1008, 2014.

[41] X. Li and F. Song, "Keyphrase extraction and grouping based on association rules." in *FLAIRS Conference*, 2015, pp. 181–186.

[42] J. S. Ward and A. Barker, "Undefined by data: A survey of big data definitions arxiv: 1309," 2013.

[43] C. Dobre and F. Xhafa, "Intelligent services for big data science," *Future Generation Computer Systems*, vol. 37, p. 267281, 07 2014.

[44] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," (verified June 2018), http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf.

[45] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137 – 144, 2015.

[46] M. R. Ghazi and D. Gangodkar, "Hadoop, mapreduce and hdfs: a developers perspective," *Procedia Computer Science*, vol. 48, pp. 45–50, 2015.

[47] R. P. Padhy, "Big data processing with hadoop-mapreduce in cloud systems," *International Journal of Cloud Computing and Services Science*, vol. 2, no. 1, p. 16, 2013.

[48] A. S. Foundation, "Apache hadoop," (verified November 2018), http://hadoop.apache.org/.

[49] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on.* Ieee, 2010, pp. 1–10.

[50] C. Uzunkaya, T. Ensari, and Y. Kavurucu, "Hadoop ecosystem and its analysis on tweets," *Procedia-Social and Behavioral Sciences*, vol. 195, pp. 1890–1897, 2015.

[51] C. Marinica, F. Guillet, and H. Briand, "Post-processing of discovered association rules using ontologies," in *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on.* IEEE, 2008, pp. 126–133.

[52] M. Cadot and A. Lelu, "Massive pruning for building an operational set of association rules: Metarules for eliminating conflicting and redundant rules," in *2009 International Conference on Information, Process, and Knowledge Management.* IEEE, 2009, pp. 90–98.

[53] B. Liu, W. Hsu, and Y. Ma, "Pruning and summarizing the discovered associations," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 1999, pp. 125–134.

[54] B. Padmanabhan and A. Tuzhilin, "A belief-driven method for discovering unexpected patterns." in *KDD*, vol. 98, 1998, pp. 94–100.

[55] A. Silberschatz and A. Tuzhilin, "What makes patterns interesting in knowledge discovery systems," *IEEE Transactions on Knowledge and data engineering*, vol. 8, no. 6, pp. 970–974, 1996.

[56] T. Brijs, K. Vanhoof, and G. Wets, "Reducing redundancy in characteristic rule discovery by using ip-techniques," 1999.

[57] R. J. Bayardo, R. Agrawal, and D. Gunopulos, "Constraint-based rule mining in large, dense databases," *Data mining and knowledge discovery*, vol. 4, no. 2-3, pp. 217–240, 2000.

[58] S. D. Bay and M. J. Pazzani, "Detecting group differences: Mining contrast sets," *Data mining and knowledge discovery*, vol. 5, no. 3, pp. 213–246, 2001.

[59] S. Huang and G. I. Webb, "Discarding insignificant rules during impact rule discovery in large, dense databases," in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 541–545.

[60] S. Chawla and J. Davis, "On local pruning of association rules using directed hypergraphs," in *ICDE04–Proc. of the 20th International Conference on Data Engeneering, IEEE*. Citeseer, 2003.

[61] H. Liu, L. Liu, and H. Zhang, "A fast pruning redundant rule method using galois connection," *Applied Soft Computing*, vol. 11, no. 1, pp. 130–137, 2011.

[62] H. Yun, D. Ha, B. Hwang, and K. H. Ryu, "Mining association rules on significant rare data using relative support," *Journal of Systems and Software*, vol. 67, no. 3, pp. 181–191, 2003.

[63] G. Hrovat, I. Fister Jr, K. Yermak, G. Stiglic, and I. Fister, "nterestingness meaIsure for mining sequential patterns in sports," *Journal of Intelligent & Fuzzy Systems*, vol. 29, no. 5, pp. 1981–1994, 2015.

[64] H. Kim and E.-Y. Kwak, "Information-based pruning for interesting association rule mining in the item response dataset," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2005, pp. 372–378.

[65] Y.-H. Hu, F. Wu, and Y.-C. Liao, "Sequential pattern mining with multiple minimum supports: A tree based approach," in *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on*. IEEE, 2010, pp. 428–433.

[66] K. Tayal and V. Ravi, "Fuzzy association rule mining using binary particle swarm optimization: Application to cyber fraud analytics," in *Computational Intelligence and Computing Research (ICCIC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–5.

[67] S. J. Samuel, K. RVP, K. Sashidhar, and C. Bharathi, "A survey on big data and its research challenges," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 8, pp. 3343–3347, 2015.

[68] A. An, S. Khan, and X. Huang, "Hierarchical grouping of association rules and its application to a real-world domain," *International journal of systems science*, vol. 37, no. 13, pp. 867–878, 2006.

[69] A. Strehl, G. K. Gupta, and J. Ghosh, "Distance based clustering of association rules," in *Proceedings ANNIE*, vol. 9, no. 1999, 1999, pp. 759–764.

[70] B. Li, Z. Pei, and K. Qin, "Association rules mining based on clustering analysis and soft sets," in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 675–680.

[71] X. Dong and D. Pi, "An effective method for mining quantitative association rules with clustering partition in satellite telemetry data," in *Advanced Cloud and Big Data (CBD), 2014 Second International Conference on*. IEEE, 2014, pp. 26–33.

[72] C. A. Kumar, "Fuzzy clustering-based formal concept analysis for association rules mining," *Applied artificial intelligence*, vol. 26, no. 3, pp. 274–301, 2012.

[73] S. Bedi, H. Yadav, and P. Yadav, "Categorization, clustering and association rule mining on www," in *Multimedia, Signal Processing and Communication Technologies, 2009. IMPACT'09. International*. IEEE, 2009, pp. 173–177.

[74] T. T. Quan, L. N. Ngo, and S. C. Hui, "An effective clustering-based approach for conceptual association rules mining," in *Computing and Communication Technologies, 2009. RIVF'09. International Conference on*. IEEE, 2009, pp. 1–7.

[75] Y. S. Koh and R. Pears, "Rare association rule mining via transaction clustering," in *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*. Australian Computer Society, Inc., 2008, pp. 87–94.

[76] F. Liu, Z. Lu, and S. Lu, "Mining association rules using clustering," *Intelligent Data Analysis*, vol. 5, no. 4, pp. 309–326, 2001.

[77] F. Palumbo and A. I. DEnza, "Clustering and dimensionality reduction to discover interesting patterns in binary data," in *Advances in Data Analysis, Data Handling and Business Intelligence*. Springer, 2009, pp. 45–55.

[78] L. Yang, "Visualizing frequent itemsets, association rules, and sequential patterns in parallel coordinates," in *International Conference on Computational Science and Its Applications*. Springer, 2003, pp. 21–30.

[79] M. Hahsler and R. Karpienko, "Visualizing association rules in hierarchical groups," *Journal of Business Economics*, vol. 87, no. 3, pp. 317–335, 2017.

[80] B. L. W. H. Y. Ma and B. Liu, "Integrating classification and association rule mining," in *Proceedings of the fourth international conference on knowledge discovery and data mining*, 1998.

[81] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[82] Ö. M. Soysal, "Association rule mining with mostly associated sequential patterns," *Expert Systems with applications*, vol. 42, no. 5, pp. 2582–2592, 2015.

[83] A. Cutbill and G. G. Wang, "Mining constraint relationships and redundancies with association analysis for optimization problem formulation," *Engineering Optimization*, vol. 48, no. 1, pp. 115–134, 2016.

[84] W. S. Seol, H. W. Jeong, B. Lee, and H. Y. Youn, "Reduction of association rules for big data sets in socially-aware computing," in *2013 IEEE 16th International Conference on Computational Science and Engineering*. IEEE, 2013, pp. 949–956.

[85] M. J. Zaki, "Generating non-redundant association rules," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 34–43.

[86] L. A. Fernandes and A. C. B. García, "Association rule visualization and pruning through response-style data organization and clustering," in *Ibero-American Conference on Artificial Intelligence*. Springer, 2012, pp. 71–80.

[87] M. Hahsler, "Grouping association rules using lift," *C. Iyigun, R. Moghaddess, and A*, 2016.

[88] W. page., " https://www.kaggle.com/," NA.

[89] S. Kannan and R. Bhaskaran, "Association rule pruning based on interestingness measures with clustering," *arXiv preprint arXiv:0912.1822*, 2009.

Vitae

- Name: Mohamed Ali Alasow
- Nationality: Somali
- Date of Birth: 1988
- Email: *calasow028@hotmail.com*
- Permenant Address: calasow028@hotmail.com