Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A fast and resource efficient mining algorithm for discovering frequent patterns in distributed computing environments

Kawuu W. Lin *, Sheng-Hao Chung

Department of Computer Science and Information Engineering, National Kaohsiung University of Applied Sciences, Taiwan

HIGHLIGHTS

- The algorithm is *parameter-less* and able to determine the appropriate number of computing nodes *automatically*.
- The algorithm requires only 54.3% execution time of PSWS when using 20 nodes.
- The algorithm achieves better *load balancing* as compared with existing methods.
- The nodes do not need to exchange any transactions or sub databases with each other.

ARTICLE INFO

Article history:
Received 20 December 2014
Received in revised form
18 March 2015
Accepted 15 May 2015
Available online 28 May 2015

Keywords:
Data mining
Frequent pattern mining
Distributed mining
Parallel mining

ABSTRACT

The advancement of electronic technology enables us to collect logs from various devices. Such logs require detailed analysis in order to be broadly useful. Data mining is a technique that has been widely used to extract hidden information from such data. Data mining is mainly composed of association rules mining, sequent pattern mining, classification and clustering. Association rules mining has attracted significant attention and been successfully applied to various fields. Although the past studies can effectively discover frequent patterns to deduce association rules, execution efficiency is still a critical problem. To speed up execution, many methods using parallel and distributed computing technology have been proposed in recent years. Most of the past studies focused on parallelizing the workload in a high end machine or in distributed computing environments like grid or cloud computing systems; however, very few of them discuss how to efficiently determine the appropriate number of computing nodes, considering execution efficiency and load balancing. An intuition is that execution speed is proportional to the number of computing nodes—that is, more the number of computing nodes, faster is the execution speed. However, this is incorrect for such algorithms because of the inherently algorithmic design. Allocating too many computing nodes can lead to high execution time. In addition to the execution inefficiency, inappropriate resource allocation is a waste of computing power and network bandwidth. At the same time, load cannot be effectively distributed if there are too few nodes allocated. In this paper, we propose a fast, load balancing and resource efficient algorithm named FLR-Mining for discovering frequent patterns in distributed computing systems. FLR-Mining is capable of determining the appropriate number of computing nodes automatically and achieving better load balancing as compared with existing methods. Through empirical evaluation, FLR-Mining is shown to deliver excellent performance in terms of execution efficiency and load balancing.

1. Introduction

The advancement of electronic technology enables us to collect logs from various devices. Such logs require detailed analysis in order to be broadly useful. Data mining is a technique that has been widely used to extract hidden information from such data. For example, association rules mining is one of the most popular mining algorithms that can discover associated relationships among items; namely, frequent patterns [1,2], which can be used to determine a suitable goods placement scheme for helping the industry improve services and increase profits. Data mining has been successfully applied to various fields. With the rise of social networking, a user's preferences or likes can be known by analyzing the articles posted or shared by the user, by their 'likes' and by their subscriptions to fan pages. The advertising impact can

* Corresponding author.
E-mail address: linwc@kuas.edu.tw (K.W. Lin).

http://dx.doi.org/10.1016/j.future.2015.05.009
0167-739X/© 2015 Elsevier B.V. All rights reserved.

be improved by mining the attention of customers [3,4]. Some of the past studies developed mining algorithms for mining useful information from gene-expression data [5–7] to help with the new direction of human diseases.

Data mining is mainly composed of association rules mining, sequent pattern mining, classification and clustering. Apriori, proposed by Agrawal et al. [1], is the first algorithm for mining association rules and has attracted considerable discussion in the field. Although Apriori can effectively discover frequent patterns, the main drawbacks are the long execution time and the large required memory for storing 2-candidate itemsets. To speed up the execution, J. Han et al. [2] proposed a novel data structure (FP-Tree) and an associated mining algorithm (FP-Growth) to mitigate the shortcomings of Apriori. FP-tree is a prefix-tree based structure, which is intended to compress a database by merging transactions with common prefixes into a single path of the tree. Using the FP-Tree structure makes it possible to accommodate the 2-candidate itemsets in memory. In addition, the Apriori based algorithms need *n* scans of the database if the maximum length of frequent patterns is *n*. FP-Growth requires only two scans of the database, which significantly improves the execution performance. The FP-Growth algorithm for frequent pattern mining remains one of the most efficient methods.

Nowadays hundreds of millions of people generate various kinds of information through electronic equipment. The database is huge, ranging in size from terabytes to petabytes. Moreover, there is a rapid upward trend in database size, which makes mining difficult, leading to Big Data problems [8]. In practical applications, execution performance is a critical problem. To improve execution performance, several methods that use parallel and distributed computing technology like grid [9,10] systems have been proposed in recent years. These studies can be further divided into two types, (1) Apriori-like approach and (2) FP-Growth-like approach. For the Apriori-like methods, the authors in [11] are the first ones who used parallel computing to accelerate the mining efficiency. Although the counting and data are successfully distributed, the performance is still unsatisfactory. In [12], a revised algorithm was proposed to run on multi-processor machines [12]; this architecture is faster, but such machines are expensive. In large scale computing, the cloud computing architecture (Hadoop or MapReduce) [13] is used to speed up frequent pattern mining. Some studies use specific designs like Transaction Identifiers (TID) [14] to enhance execution performance. However, these algorithms have the same performance drawbacks as that of Apriori—they require a lot of time because they scan the database multiple times. For FP-Growth-like methods, most of the past studies tried to improve the FP-tree structure to enhance mining efficiency. G. Grahne et al. [15] discovered that FP-Growth spent about 80% of CPU time in scanning FP-trees and building conditional FP-trees. To reduce the time and resources required in scanning a repetitive structure, an algorithm called FP-growth* was proposed, which records certain information to reduce FP-tree scan time, thus effectively improving execution efficiency. B. Schlegel et al. [16] proposed the CFP-Tree, which is a ternary tree architecture that can be used to reduce tree size. S.J. Yen et al. [17] proposed a TFP-tree structure for decreasing memory usage and increasing efficiency. TFP-tree merges and prunes FP-tree to reduce recursive building of FP-trees. Some researchers used disk space to solve big data mining problems. M. Adnan et al. [18] proposed a DRFP-tree that can use disk space to store the entire FP-tree for solving the memory problem caused by big data. J. Han et al. [19] proposed a database projection method that builds multiple sub-databases for effectively improving scalability. In [20], G. Grahne et al. proposed aggressive projection to reduce the problem of too many sub-databases caused by database projection [19], and applied FP-growth* [15] to speed up mining. In [21–23], the authors used multi-processor machines or distributed computing resources to improve mining efficiency. Hadoop is a popular cloud computing technology and has been used for performance enhancement in [24,25]. Y. Qiu et al. [26] proposed QFP-growth, wherein the database is divided into equal sized and disjointed sub-databases. The FP-trees corresponding to the sub-databases are then built individually, and the FP-trees are then merged into a complete one; the time for building an FP-tree is thus reduced. J. Zhou et al. proposed the TPFP [27] algorithm that used the TidSet data structure to exchange information effectively in grid computing environments without requiring scanning of the entire database. In addition, they also proposed the BTP-tree [28] algorithm, which is an extension of TPFP [27], with the ability to use different machines with varying computing power and memory. As a result, it is effective for load balancing in a heterogeneous computing system. Kawuu W. Lin et al. proposed the CARM algorithm [29] and PSWS [30], which primarily applies distributed computing techniques to speed up recursive mining of conditional FP-trees, which resulted in an improvement in execution efficiency. Such algorithms compress the entire FP-tree; the FP-tree is transmitted to connection nodes and subsequently transmitted to computing nodes for achieving low network traffic in cloud computing environments. PSWS has developed a better work dispatch policy and load balancing, as compared with BTP-tree [28].

With the advent of Big Data, execution efficiency has become critical to mining. A slow mining algorithm is not capable of discovering important information from such big databases in a timely manner, thus leading to tangible or intangible losses. As a result, many researchers try to use parallel, distributed computing or cloud computing techniques to improve the execution efficiency. Most of the past studies focused on how the algorithms can be parallelized within a machine or in distributed computing environments, however, very few of them discussed about determining the number of computing nodes while considering load balancing at the same time. An intuition is that execution speed is proportional to the number of computing nodes—that is, more the number of computing nodes, faster is the execution speed. However, this is incorrect for distributed algorithms that mine frequent patterns because the algorithms cannot begin the mining until they receive needed data [30,29] and must exchange information among the computing nodes over the physical network [28]. Having too many computing nodes can lead to increased execution time. In addition to execution efficiency, inappropriate resource allocation is a waste of computing power and network bandwidth. At the same time, load cannot be effectively distributed if there are too few nodes allocated.

Consider the load balancing problem of frequent pattern mining. Since it is almost impossible to determine the mining time of a task in advance, developing a good algorithm that performs load balancing is challenging. In [29], the authors proposed CARM that divides a mining task into several sub-mining jobs by using the header item, corresponding to a conditional FP-tree, of header table as a working unit and sends the disjoint set of header items to the distributed nodes for mining the corresponding conditional FP-trees. PSWS is a CARM-based algorithm that focused on the load balancing problem and is one of the most efficient algorithms for mining frequent patterns in such environments. One of the main drawbacks of PSWS is the number of computing nodes that needs to be manually determined. Clearly, it is impractical to allocate a person to manually determine the number of computing nodes required when a mining task is submitted. Moreover, there are no guidelines for making an appropriate determination. Another drawback of PSWS is that the performance of load balancing degrades with the increase in the number of computing nodes.

Furthermore, the support has a big impact on execution time. In the field of association rules mining, the user specifies the support threshold to obtain hidden information [1] from a database. When