Abstract. Although the FP-Growth association-rule mining algorithm is more efficient than the Apriori algorithm, it has two disadvantages. The first is that the FP-tree can become too large to be created in memory; the second is the serial processing approach used. In this paper, a kind of parallel association-rule mining algorithm has been proposed. It does not need to create an overall FP-tree, and it can distribute data mining tasks over several computing nodes to achieve parallel processing. This approach will greatly improve efficiency and processing ability when used for mining association rules and is suitable for association-rule mining on massive data sets.

Keywords: Data mining, Association rules, Frequent pattern, FP-tree, Parallel algorithm.

1 Introduction

Association rules were first proposed by Agrawal in 1993 [1] and are an important data mining research topic. The Apriori algorithm was proposed in 1994 by Agrawal and Srikant as an original algorithm for mining Boolean-type association rules [2]. Many improved algorithms have been derived from the Apriori algorithm. However, all Apriori-like algorithms have the following deficiencies [3]: (1) they must spend a large amount of time processing a large number of candidate item sets, and (2) they must repeatedly scan the database to match candidate item sets to find frequent patterns. To avoid the deficiencies of the Apriori-like algorithms, the FP-growth algorithm was proposed by Han et al. [3] for mining association rules. Research has shown that the FP-growth algorithm is approximately one order of magnitude faster than the Apriori algorithm.

In recent years, along with the digitalization of society, massive data applications have become more and more prevalent. Although the FP-growth algorithm is more effective than Apriori, it is still unsuitable for performing association-rule mining tasks on massive data sets. The reasons for this are that the FP-growth algorithm performs serial processing and that the FP-tree may become too large to be created in memory.

Association-rule mining problems based on applications of distributed data structures have made apparent the practical need for a distributed data mining algorithm to avoid the transmission of large amounts of data in networks.
To achieve parallel association-rule mining or association-rule mining on distributed data structures, approaches reported in the literature include data division [4], multithreaded memory-sharing parallel algorithms [5-6], and division of the overall FP-tree into small FP-trees for parallel processing once the overall FP-tree has been obtained [7]. Reference [8] suggested that in the FP-growth algorithm, the processes of mining every conditional-pattern base are independent, and therefore frequent pattern mining of every conditional-pattern base can be regarded as an independent subtask, and the whole task of frequent pattern mining can be divided into a set of subtasks which can be assigned to different nodes of a computer cluster.

This paper proposes an improved algorithm based on the FP-growth algorithm. The algorithm is designed to operate in a distributed application data framework, does not need to create an overall FP-tree, and uses parallel processing in all its principal steps.

2 Description of the Problem

Let I=\{i_1,i_2,\ldots,i_n\} be the set of all items, and let transaction T be a set composed of several items from I, T \subseteq I. Each transaction has a unique Tid identifier.

Let D be a global database composed of multiple local databases, \(D = \sum_{j=1}^{m} d_j\), where \(d_j \ (j = 1,2,\ldots,m)\) are distributed in different storage nodes \(M_j\).

Let \(C_j \ (j=1,2\ldots,p)\) be a group of computing nodes with powerful calculation capabilities. In a physical sense, a storage node and a computing node can completely coincide, partially overlap, or be completely distinct.

The algorithm will attempt to make full use of the computing nodes \(C_j \ (j = 1, 2\ldots p)\) to discover the frequent patterns as quickly as possible from the overall transaction database D to obtain association rules.

3 Algorithm Description

(1) For each storage node \(M_j\): obtain the count of all items in \(d_j\).

(2) In the central computing node: collect and sum the counts of all the items in each \(d_j\) to obtain the count of all items in the overall database D. Order the items in descending order. Delete items for which the count is less than the specified minimum count supported. The result is the overall 1-item frequent set L.

(3) In the central computing node: to distribute the frequent pattern mining task over many computing nodes by items to realize parallel processing in the following step, the central computing node needs to assign a computing node to every item in L and to add its assigned results to L. Finally, L is obtained as shown in Table 1.