Parallel Algorithm for Mining Maximal Frequent Patterns

Hui Wang\textsuperscript{1,2}\textsuperscript{**}, Zhiting Xiao\textsuperscript{2}, Hongjun Zhang\textsuperscript{1}, and Shengyi Jiang\textsuperscript{1}

\textsuperscript{1}Computer School, Huazhong University of Science & Technology, 430074, WuHan, China
\textsuperscript{2}Wuhan Communication College, 430010, Wuhan, China
hustwanghui@mail.china.com

Abstract. We present a novel and powerful parallel algorithm for mining maximal frequent patterns, called Par-MinMax. It decomposes the search space by prefix-based equivalence classes, distributes work among the processors and selectively duplicates databases in such a way that each processor can compute the maximal frequent patterns independently. It utilizes multiple level backtrack pruning strategy and other novel pruning strategies, along with vertical database format, counting frequency by simple tid-list intersection operation. These techniques eliminate the need for synchronization, drastically cutting down the I/O overhead. The analysis and experimental results demonstrate the superb efficiency of our approach in comparison with the existing work.

1 Introduction

Mining frequent patterns is to discover all frequent patterns in a given database. It comprises the core of several data mining algorithms such as association rule mining and sequence mining, and dominates the running time of those algorithms. It has been shown to have an exponential worst case running time in the number of items, therefore much research [1,2,3,4] has been devoted to increasing the efficiency of the task.

Since both the data size and the computational costs are large, parallel algorithms have been studied extensively. Frequent pattern discovery has become a challenge for parallel programming since it is a highly complex operation on huge datasets demanding efficient and scalable algorithms. Most previous parallel algorithms [5,6,7,8,9,10] use complicated hash structures, make repeated passes over the database partition, have to exchange the partial results among all the processors during each iteration, resulting in additional maintaining overhead, high I/O overhead, expensive communication and synchronization cost.

\textsuperscript{*} This paper is supported by the National Natural Science Foundation of China under Grant No.60273075.

\textsuperscript{**} Hui Wang is a PhD candidate in computer school, Huazhong university of science and technology. Her research interests include data mining and parallel computing.
We present a novel and powerful parallel algorithm for mining maximal frequent patterns, called Par-MinMax, which is based on its serial version MinMax[11]. The new algorithm decomposes the original search space into smaller pieces by prefix-based equivalence classes, schedules classes among processors by the weights, distributes work among the processors and selectively duplicates databases in such a way that each processor can compute the frequent patterns independently. It uses depth-first search and a novel multiple level backtrack pruning strategy[11] and other powerful pruning strategies, along with vertical tid-list database format, counting frequency by simple tid-list intersection operation. These techniques eliminate the need for synchronization, drastically cutting down the I/O overhead. The analysis and experimental results demonstrate the superb efficiency of our approach in comparison with the previous work.

The rest of this paper is organized as follows: In section 2 we describe the maximal frequent pattern problem. The serial version MinMax is briefly described in Section 3. Section 4 describes our new algorithm Par-MinMax. We show the experimental results in section 5. The conclusions are in section 6.

2 Problem Statement

The problem of mining maximal frequent patterns is formally stated by definitions 1-4 and theorems 1-2. To describe our algorithm clearly, definition 5-9, propositions 1-2 and theorem 3 are given in this paper.

Let \( I = \{i_1, i_2, \ldots, i_m\} \) be a set of \( m \) distinct items. Let \( D \) denote a database of transactions where each transaction has a unique identifier (tid) and contains a set of items.

**Definition 1:** (pattern) A set \( X \subseteq I \) is called a pattern (an itemset). A pattern with \( k \) items is called a \( k \)-pattern.

**Definition 2:** (pattern’s frequency \( \sigma \)) The frequency of a pattern \( X \), denoted by \( \sigma (X) \), is defined as the number of transactions in which the pattern occurs as a subset, called the support of the pattern.

With the vertical database layout, the database is comprised by items with corresponding tid-lists. Let \( x \in I \), \( \text{tid-list}(x) = \{t_i | x \text{ appeared in } t_i \} \). \( \sigma (x) = |\text{tid-list}(x)| \). Let \( X \subseteq I \), \( X = \{x_1, x_2, \ldots, x_k\} \), \( \sigma (X) = |\bigcap_{x_i \in X} \text{tid-list}(x_i)| \).

**Definition 3:** (frequent pattern) Let \( \epsilon \) be the threshold minimum frequency value specified by user. If \( \sigma (X) \geq \epsilon \), \( X \) is called a frequent pattern. The frequent 1-pattern is called a frequent item. The set of all the frequent \( k \)-patterns in \( D \) is denoted by \( F_k \).

**Definition 4:** (maximal frequent pattern) If \( \sigma (X) \geq \epsilon \land \neg \exists Y (Y \supset X) \land \sigma (Y) \geq \epsilon \), we say \( X \) is a maximal frequent pattern.