CHAPTER 5

AN IMPROVED PARALLEL ALGORITHM FOR MINING FREQUENT PATTERNS

5.1 Introduction

A new parallel algorithm has been proposed to discover frequent data item sets with a user specified minimum support, and is different from other frequent pattern mining algorithm like Apriori algorithm [63]. The frequent item sets are determined by progressively enlarging the interval by 1 which is identified by the index.

By sharing the work between n processors, the frequent items are computed, and Initially the transactions are distributed to processors since processor $P_i$ has to compute $F_{ij}$ where $j=1, \ldots, n$. To compute the frequent data items in the data mining, there will be communication between the processors.

In this new parallel approach, the $P_i$ processor can have only the transaction containing the frequent item $i$, so that it promotes the computation speed and also reduces the scanning time. This proposed algorithm is efficient, fast, and better performance over the traditional algorithms.
5.2 Improved Parallel Algorithm

Consider m transaction and n individual frequent items. In the first step, the new candidates set will be subsets of the sets \{1, 2\}, \{2, 3\}, \{n − 1, n\}. In the second step the new candidates set will be subsets of the sets \{1, 2, 3\}, \{2, 3, 4\}, \{n − 2, n − 1, n\}. In the last step the new candidates set will be subsets of the set \{1, 2 . . . n\}.

This parallel approach simplifies the candidate generation function, and has bigger parallelism potential than Apriori algorithm. In Figure 5.1, an efficient parallel algorithm is presented and is used to discover all frequent data items available in the large database.

\[ F_{i,j} = \text{the set of all frequent itemsets from the interval [i, j] that are having minimum support} \]

\[ C_{i,j} = \text{the set of candidate itemsets from the interval [i, j].} \]

Step 1: A frequent item sets from \( F_{i,j} \) which does not simultaneously contain the items i and j belongs to \( F_{i,j-1} \) or \( F_{i+1,j} \).

Step 2: The new frequent item sets in \( F_{i,j} \) simultaneously contain the items i and j, that the candidates from the interval [i, j] are obtained in accordance with the following relation

\[ C_{i,j} = \{X ∪ Y | (X ∈ F_{i,j−1} ∧ i ∈ X) ∧ (Y ∈ F_{i+1,j} ∧ j ∈ Y) \} \]
for k = 1 to n − m do begin
for all i : i ∈ {1, . . . , m} do begin
    j ← i + k
    $F_{i,j} ← F_{i,j−1} \cup F_{i+1,j}$
    $C_{i,j} ← \{X \cup Y \mid (X \in F_{i,j−1} \wedge i \in X) \wedge (Y \in F_{i+1,j} \wedge j \in Y)\}$
    for all transactions $t \in D_i$ do begin
        $C_t ← \text{subset}(C_{i,j}, t)$ // candidates contained in $t$
        for all candidates $c \in C_t$ do begin
            $c\cdot\text{count} ← c\cdot\text{count} + 1$
        end
    end
    $F_{i,j} = F_{i,j} \cup \{c \in C_{i,j} \mid c\cdot\text{count} \geq \text{min\_supp} \}$
end
end
for all i : i ∈ {1, . . . , m} do begin
    send $F_{i,j}$ to $P_i$
end
for k = n − m + 1 to n − 1 do begin
for all i : i ∈ {1, . . . , n − k} par do begin
    j ← i + k
    $F_{i,j} ← F_{i,j−1} \cup F_{i+1,j}$
    $C_{i,j} ← \{X \cup Y \mid (X \in F_{i,j−1} \wedge i \in X) \wedge (Y \in F_{i+1,j} \wedge j \in Y)\}$
    for all transactions $t \in D_i$ do begin
        $C_t ← \text{subset}(C_{i,j}, t)$ // candidates contained in $t$
        for all candidates $c \in C_t$ do begin
            $c\cdot\text{count} ← c\cdot\text{count} + 1$
        end
    end
    $F_{i,j} = F_{i,j} \cup \{c \in C_{i,j} \mid c\cdot\text{count} \geq \text{min\_supp} \}$
end
end

Figure 5.1 Parallel Algorithm
5.3 Steps In Parallel Algorithm

Table 5.1 presents an example with 10 transactions, and 9 frequent individual items. The transactions are denoted by T1, T2, , T10 and the individual data items $i_1, i_2, \ldots, i_9$ are identified with their indices.

Table 5.1 Transaction Database for Parallel Algorithm

<table>
<thead>
<tr>
<th>TID</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>{1, 2}</td>
</tr>
<tr>
<td>T2</td>
<td>{1, 2, 3, 5, 9}</td>
</tr>
<tr>
<td>T3</td>
<td>{1, 4}</td>
</tr>
<tr>
<td>T4</td>
<td>{3, 4, 5, 6, 7, 8}</td>
</tr>
<tr>
<td>T5</td>
<td>{7, 8, 9}</td>
</tr>
<tr>
<td>T6</td>
<td>{2, 4, 6, 8}</td>
</tr>
<tr>
<td>T7</td>
<td>{1, 3, 5, 7, 9}</td>
</tr>
<tr>
<td>T8</td>
<td>{1, 4, 5, 6, 7}</td>
</tr>
<tr>
<td>T9</td>
<td>{2, 4, 5, 6, 9}</td>
</tr>
<tr>
<td>T10</td>
<td>{3, 4, 5, 6, 7}</td>
</tr>
</tbody>
</table>

The Minimum Support taken for this algorithm is 2. The new parallel algorithm used to compute the frequent itemsets by sharing the work between n processors. The sets $F_{i,j}$, $j = i, \ldots, n$ are computed by processor $P_i$, in successive steps. Processor $P_i$ uses $F_{i,j-1}$ from the previous step and $F_{i+1,j}$ received from the processor $P_{i+1}$. 
The set of transactions will be distributed to the processors, before the execution begins. The step1 of parallel algorithms are shown below.

Initially,

**Frequents items**: 1,2,3,4,5,6,7,8,9.

\[ F_{1,1} = \{1\}; F_{2,2} = \{2\}; F_{3,3} = \{3\}; F_{4,4} = \{4\}; F_{5,5} = \{5\}; F_{6,6} = \{6\}; F_{7,7} = \{7\}; F_{8,8} = \{8\}; F_{9,9} = \{9\}. \]

**Step 1**

Initial frequent itemsets in step 1:

\[ IF_{1,2} = \{1,2\}; IF_{2,3} = \{2,3\}; IF_{3,4} = \{3,4\}; IF_{4,5} = \{4,5\}; IF_{5,6} = \{5,6\}; IF_{6,7} = \{6,7\}; IF_{7,8} = \{7,8\}; IF_{8,9} = \{8,9\}. \]

Candidates:

\[ C_{1,2} = \{1,2\}; C_{2,3} = \{2,3\}; C_{3,4} = \{3,4\}; C_{4,5} = \{4,5\}; C_{5,6} = \{5,6\}; C_{6,7} = \{6,7\}; C_{7,8} = \{7,8\}; C_{8,9} = \{8,9\}. \]

New frequent itemsets:

\[ NF_{1,2} = \{1,2\}; NF_{2,3} = \emptyset; NF_{3,4} = \{3,4\}; NF_{4,5} = \{4,5\}; NF_{5,6} = \{5,6\}; NF_{6,7} = \{6,7\}; NF_{7,8} = \{7,8\}; NF_{8,9} = \emptyset. \]

Frequent itemsets at the end of the step 1:

\[ F_{1,2} = \{1,2\}; F_{2,3} = \{2,3\}; F_{3,4} = \{3,4\}; F_{4,5} = \{4,5\}; F_{5,6} = \{5,6\}; F_{6,7} = \{6,7\}; F_{7,8} = \{7,8\}; F_{8,9} = \{8,9\}. \]

Initially, for finding the frequent item sets of 1-itemsets, all the items that satisfies the minimum support count 2 will be included, and in this case, all the items have been included. In step1, initial frequent itemsets are to be generated by combining the current frequent itemsets with next frequent itemsets.
Then the candidate 2 itemsets are generated and are to be checked to see which one has satisfied the minimum support count, and that will be considered as new frequent itemsets. During the calculation of frequent itemsets at the end of step 1, both the initial and new frequent itemsets are combined together to form the frequent itemsets at the end of step 1.

**Step 2**

**Initial frequent itemsets in step 2:**
IF$_{1,3}$={1,2,3,\{1,2\}}; IF$_{2,4}$={2,3,4,\{3,4\}};
IF$_{3,5}$={3,4,5,\{3,4\},\{4,5\}}; IF$_{4,6}$={4,5,6,\{4,5\},\{5,6\}};
IF$_{5,7}$={5,6,7,\{5,6\},\{6,7\}}; IF$_{6,8}$={6,7,8,\{6,7\},\{7,8\}};
IF$_{7,9}$={7,8,9,\{7,8\}}.

**Candidates:**
C$_{1,3}$={\{1,3\},\{1,2,3\}}; C$_{2,4}$={\{2,4\},\{2,3,4\}};
C$_{3,5}$={\{3,5\},\{3,4,5\}}; C$_{4,6}$={\{4,6\},\{4,5,6\}};
C$_{5,7}$={\{5,7\},\{5,6,7\}}; C$_{6,8}$={\{6,8\},\{6,7,8\}};
C$_{7,9}$={\{7,9\},\{7,8,9\}}.

**New frequent itemsets:**
NF$_{1,3}$={\{1,3\}}; NF$_{2,4}$={\{2,4\}};
NF$_{3,5}$={\{3,5\},\{3,4,5\}}; NF$_{4,6}$={\{4,6\},\{4,5,6\}};
NF$_{5,7}$={\{5,7\},\{5,6,7\}}; NF$_{6,8}$={\{6,8\}};
NF$_{7,9}$={\{7,9\}}.

**Frequent itemsets at the end of the step 2:**
F$_{1,3}$={1,2,3,\{1,2\},\{1,3\}}; F$_{2,4}$={2,3,4,\{3,4\},\{2,4\}};
F$_{3,5}$={3,4,5,\{3,4\},\{4,5\},\{3,5\},\{3,4,5\}};
F$_{4,6}$={4,5,6,\{4,5\},\{5,6\},\{4,6\},\{4,5,6\}};
F$_{5,7}$={5,6,7,\{5,6\},\{6,7\},\{5,7\},\{5,6,7\}};
F$_{6,8}$={6,7,8,\{6,7\},\{7,8\},\{6,8\}}; F$_{7,9}$={7,8,9,\{7,8\},\{7,9\}}.

From the step 1, frequent item sets are successively combined together, and the item which satisfies the minimum support count will be included as initial frequent itemsets in step 2.
The candidate itemsets are generated, and are to be checked to see which one has satisfied the minimum support count, and that will be considered as new frequent itemsets. During the calculation of frequent itemsets at the end of step 2, both the initial and new frequent itemsets are combined together to form the frequent itemsets at the end of step 2.

**Step 3**

**Initial frequent itemsets in step 3:**
- \( IF_{1,4} = \{1,2,3,4,\{1,2\},\{1,3\},\{3,4\},\{2,4\}\} \);
- \( IF_{2,5} = \{2,3,4,5,\{3,4\},\{2,4\},\{3,4\},\{4,5\},\{3,5\},\{3,4,5\}\} \);
- \( IF_{3,6} = \{3,4,5,6,\{3,4\},\{4,5\},\{3,5\},\{3,4,5\},\{4,5\},\{5,6\},\{4,6\},\{4,5,6\}\} \);
- \( IF_{4,7} = \{4,5,6,7,\{4,5\},\{5,6\},\{4,6\},\{4,5,6\},\{5,6\},\{6,7\},\{5,7\},\{5,6,7\}\} \);
- \( IF_{5,8} = \{5,6,7,8,\{5,6\},\{6,7\},\{5,7\},\{5,6,7\},\{6,7\},\{7,8\},\{6,8\}\} \);
- \( IF_{6,9} = \{6,7,8,9,\{6,7\},\{7,8\},\{6,8\},\{7,8\},\{7,9\}\} \).

**Candidates:**
- \( C_{1,4} = \{1,4\},\{1,2,4\},\{1,3,4\},\{1,2,3,4\}\) ;
- \( C_{2,5} = \{2,5\},\{2,3,5\},\{2,4,5\},\{2,3,4,5\}\) ;
- \( C_{3,6} = \{3,6\},\{3,4,6\},\{3,5,6\},\{3,4,5,6\}\) ;
- \( C_{4,7} = \{4,7\},\{4,5,7\},\{4,6,7\},\{4,5,6,7\}\) ;
- \( C_{5,8} = \{5,8\},\{5,6,8\},\{5,7,8\},\{5,6,7,8\}\) ;
- \( C_{6,9} = \{6,9\},\{6,7,9\},\{6,8,9\},\{6,7,8,9\}\) .

**New frequent itemsets:**
- \( NF_{1,4} = \{1,4\} \); \( NF_{2,5} = \{2,5\} \);
- \( NF_{3,6} = \{3,6\},\{3,4,6\},\{3,5,6\},\{3,4,5,6\}\);
- \( NF_{4,7} = \{4,7\},\{4,5,7\},\{4,6,7\},\{4,5,6,7\}\); \( NF_{5,8} = \emptyset \);
- \( NF_{6,9} = \emptyset \).

**Frequent itemsets at the end of the step 3:**
- \( F_{1,4} = \{1,2,3,4,\{1,2\},\{1,3\},\{3,4\},\{2,4\},\{1,4\}\} \);
- \( F_{2,5} = \{2,3,4,5,\{3,4\},\{2,4\},\{3,4\},\{4,5\},\{3,5\},\{3,4,5\},\{2,5\}\} \);
- \( F_{3,6} = \{3,4,5,6,\{3,4\},\{4,5\},\{3,5\},\{3,4,5\},\{4,5\},\{5,6\},\{4,6\},\{4,5,6\},\{3,6\},\{3,4,6\},\{3,5,6\},\{3,4,5,6\}\} \);
- \( F_{4,7} = \{4,5,6,7,\{4,5\},\{5,6\},\{4,6\},\{4,5,6\},\{5,6\},\{6,7\},\{5,7\},\{5,6,7\},\{4,7\},\{4,5,7\},\{4,6,7\},\{4,5,6,7\}\} \);
- \( F_{5,8} = \{5,6,7,8,\{5,6\},\{6,7\},\{5,7\},\{5,6,7\},\{6,7\},\{7,8\},\{6,8\}\} \);
- \( F_{6,9} = \{6,7,8,9,\{6,7\},\{7,8\},\{6,8\},\{7,8\},\{7,9\}\} \).
Initially, from the step 2 find the initial frequent itemsets by combining 2 successive item sets that is current frequent itemset with next itemsets, and which satisfies the minimum support count will be included. The candidate itemsets are generated and check which one has satisfied the minimum support count and that will be considered as new frequent itemsets.

During the calculation of frequent item sets at the end of step 3, both the initial and new frequent itemsets are combined together and that will be the frequent itemsets at the end of step 3. This will be repeated till, step 8 have been finished.

**Step 8**

**Initial frequent itemsets in step 8:**
IF\(_{1,9}\)={1,2,3,4,5,6,7,8,9\(\{1,2\},\{1,3\},\{3,4\},\{2,4\},\{1,4\},\{1,3,5\},\{1,7\}\)

**Candidates:**
C\(_{1,9}\)={\{1,9\},\{1,2,9\},\{1,3,9\},\{1,4,9\},\{1,5,9\},\{1,6,9\},\{1,7,9\},\{1,8,9\},\{1,2,3,9\},\{1,3,4,9\},\{1,4,5,9\}......};

**New frequent itemsets:**
NF\(_{1,9}\)={\{1,9\},\{1,3,9\},\{1,5,9\}}.

**Frequent itemsets at the end of the step 3:**
F\(_{1,9}\) = \{1,2,3,4,5,6,7,8,9, \{1,2\}, \{1,3\}, \{3,4\}, \{2,4\}, \{1,4\}, \{1,3,5\}, \{1,7\},\{1,9\},\{1,3,9\},\{1,5,9\}\}.

Initially, from the step 7 find the initial frequent itemsets by combining 2 successive item sets that are current frequent itemset with next itemsets, and which satisfies the minimum support count will be included.
The candidate itemsets are generated and checked to find, which one has satisfied the minimum support count, and that will be considered as new frequent itemsets. During the calculation of frequent item sets at the end of step 8, both the initial and new frequent itemsets are combined together and that will be the frequent itemsets at the end of step 8.

The new parallel algorithm employed to compute the frequent itemsets are computed by sharing the work between n processors and the processor $P_i$ will compute the sets $F_{i,j}$, $j = i, \ldots, n$, in successive steps. For this processor $P_i$ uses $F_{i,j-1}$ from the previous step and $F_{i+1}$, received from the processor $P_{i+1}$ which is shown in figure 5.2.

![Figure 5.2. Processor Communication for Finding Frequent Itemsets](image-url)
The main advantage of our parallel algorithm is that the prior to the beginning of the analysis, set of transactions can be distributed to processors. This is possible because a processor $P_i$ needs to compute $F_{i,j}$, $j = i \ldots n$, only. Let $D_i$, be the set of the transactions which contain the individual frequent item $i$. Therefore, when the algorithm starts, $P_i$ needs to have access to $D_i$ and $F_{i,i}$ = frequent item $i$.

5.4 Implementation and Results

Simulation of parallel algorithm for mining frequent patterns is simulated using JAVA with nine processors. With various amount of data, the parallel algorithm is compared with Apriori algorithm.

Table 5.2 shows time taken by parallel algorithm for finding frequent patterns for different sets of data. The data sets are generated using data set generator for testing the algorithms, and three data sets are taken and the results are compared against Apriori and Parallel Algorithms.

The set of the transactions have been distributed to processors prior to the beginning of the analysis. Initially the 500kb of data has been taken, and compared, parallel algorithm requires 5 minutes and Apriori requires 11 minutes to find the frequent data items. In this way, 800kb and 1400kb of data are taken and compared.
Table 5.2 Performance Evaluation of Apriori and Parallel Algorithm.

<table>
<thead>
<tr>
<th>S.No</th>
<th>SIZE OF DATABASE (KB)</th>
<th>ALGORITHMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>APRIORI</td>
</tr>
<tr>
<td>1</td>
<td>500</td>
<td>11 min</td>
</tr>
<tr>
<td>2</td>
<td>800</td>
<td>18 min</td>
</tr>
<tr>
<td>3</td>
<td>1400</td>
<td>29 min</td>
</tr>
</tbody>
</table>

The set of the transactions have been distributed to processors prior to the beginning of the analysis. Initially the 500kb of data have been taken and compared, parallel algorithm requires 5 minutes and Apriori requires 11 minutes to find the frequent data items.

For a data size of 800 kb, Apriori algorithm generates the frequent itemsets in 18 minutes and parallel algorithm in 7 minutes. Finally for the data size of 1400 kb, Apriori generates the frequent itemsets in 29 minutes and parallel algorithm requires 11 minutes.
The time taken for calculating the frequent data items by parallel algorithm got reduced because of parallelism, and for large amount of data, the proposed algorithm works much better than the available algorithm. The main drawback of this algorithm is cost, because it requires N number of processors. Hence the amount to be spend for implementing this algorithm is more. Even though performance wise it is more efficient, it is not cost effective.