Fast Mining of Finding Frequent Patterns in Transactional Database using Incremental Approach

Pamli Basak  
Computer Engineering  
Department  
TCET, Kandivali (E)  
Mumbai

R.R. Sedamkar  
Professor, HOD-PG, Computer Engineering Department  
TCET, Kandivali (E)  
Mumbai

Rashmi Thakur  
A.P., Computer Engineering Department  
TCET, Kandivali (E)  
Mumbai

ABSTRACT

Datasets grow in size as they are increasingly being gathered by cheap and numerous information-sensing mobile devices, aerial, software logs, microphones, wireless sensor networks and cameras. This paper presents a structure for simply, easily and competently parallelizing data mining algorithms for those huge datasets together with the incremental mining. MapReduce concept is use to execute the parallel FP-Growth algorithm by running the windows services parallel. The proposed algorithm eliminates duplicated work and spurious items. Also, it shortens the response time to a query for the set of frequent items. The proposed algorithm is implemented by parallel running of many windows services and experimental results shows tremendous advantages. The proposed algorithm runs 66% faster than the traditional algorithm of data mining. Also, memory utilization reduces by 37%.

General Terms

Data mining, Association Rule Mining, Grid Computing

Keywords

Incremental Data Mining, Parallel FP-growth, MapReduce jobs, Incremental Parallel FP-growth.

1. INTRODUCTION

Data mining [1] is called knowledge discovery in databases (KDD). Both data mining and KDD are at best indistinctly defined. Their explanations mostly depend on the background and outlooks of the definers. The real meaning of data mining is the non-trivial process of recognizing suitable, potentially helpful and finally comprehensible patterns in data. Data mining is the use of statistical methods with computers to reveal helpful patterns in the databases. Huge data is a collection of large and complex datasets which is complicated to process by using conventional methods and offered technologies of data mining. To itinerary huge data using some systematic approach can even hardly conclude the work, it takes elongated time and the outcome might not be up to the mark. To resolve this problem, data mining is undertake with new chances and challenges.

Association rules mining [2], a kind of data mining algorithms, are if/then statements that help to discover associations between apparently not related data in a relational database or other information warehouse. In market basket analysis, association rules is "the customers who buy bread are most likely to buy butter" might be produced according to the processing outcomes.

Years of study, association rule mining algorithms sound effectual and recognized in majority of cases. Though, the conventional algorithms are not healthy work when it comes to huge data. In an actual situation, databases are incessantly updating on daily basis and threshold value also frequently changes with wants of mining. It is evidently inefficient to restart the entire mining process over again every time new data is inserted into the original database or initialize the mining parameters. This concerns leads to incremental mining conception. Additionally, algorithms for parallelization have become predictable to deal with the obscurities arising from massive scale data.

This paper talks about a parallelizing the FP-growth algorithm and parallelizing the incremental algorithm and the FP-growth algorithm mining techniques. Incremental parallel FP-growth algorithm solves the incremental problem brought by the dynamic threshold value and database at the same time, which evades frequent computation. MapReduce jobs are written on windows services and make them run simultaneously to achieve parallel mining. Grid computing [3] has been projected as an important computational model, illustrous from the conventional computing by its focus on huge resource sharing, revolutionary applications, and, in some cases, high-performance orientation. These days’ grid scan is used as efficient infrastructures for dispersed high-performance computing and data processing.

2. RELATED WORK

A basic necessity for mining association rules is to find out the frequent itemsets. Several algorithms are present for frequent itemset mining. Apriori and FP-Growth are the conservative method.

2.1 APRIORI

Agrawal et. al.[4] proposed an algorithm for frequent itemset mining and association rule learning over transactional databases called Apriori. It carry on by distinguishing the repeated individual items in the database and broadening them to larger itemsets provided that those itemsets appear passably often in the database. Its workings depend on Candidate Generation and Test Approach. There are mainly two steps have to perform in each pass: Candidate generation and Candidate counting and selection.

Apriori have two main disadvantages: Candidate generation, it generates huge candidate sets; 10\(^{10}\) frequent 1-itemset will generate 10\(^{10}\) candidate 2-itemsets and to discover a frequent pattern of size 100 and scanning the original database every time, any new data is inserted in the database.
2.2 FP-GROWTH

FP-growth [5] is specified to conquer the problem of Apriori, candidate generation. FP-growth is a program to discover frequent itemsets with the FP-growth algorithm, which writes to the transaction database as a prefix-tree that enhances with links that arrange the nodes into lists which points to the identical item. The search is carried out by analyzing the prefix-tree, running recursively on the outcome, and edging the original tree. The implementation also supports sifting for closed and maximal item sets with restricted itemset storage; even though the approach worn in the program fluctuates in whenever it used top-down prefix trees instead FP-trees. FP-growth reduce a large database into a reduce itemsets, Frequent-Pattern tree (FP-tree) arrangement with highly compact, but full for frequent pattern mining and evade expensive database scans. It develops a competent, FP-tree-based frequent pattern mining method with a divide-and-conquer tactic which crumbles mining tasks into smaller ones and shuns candidate generation.

The weakness of this algorithm consists in the Tid branch being too lengthy and extended, taking huge memory space as well as working out time for intersecting the long sets. Also, the algorithm scans the original database every time; any new data is inserted in the database.

2.3 Parallelization in Data Mining

Zhang et. al. proposed parallel FP-growth algorithm [6] on distributed machines. FFP panels computation in such a way that each machine performs an independent group of mining tasks. FP-tree building procedures and processing in this algorithm are to some extent similar to conservative FP-Growth, which run on a single computer node. This separation eradicates computational dependence between machines, and hence communication between them. Pradeepa et. al. presented Parallelized Apriori algorithm [7] to estimate an accurate and proficient organization technique, greatly spirited and scalable compared with other conservative and associative organization methods. Drawback of these algorithms are that it does not support incremental mining.

Osmar et. al. discuss an algorithm MLFPT [8] for parallel mining of frequent patterns, based on FP-growth mining, that uses two full I/O scans of the database, eradicating the need for generating the candidate itemsets, and distributing the work equally among processors to achieve near most favorable load balancing.

Figure 1 illustrates Parallel FP-growth data mining algorithm, which uses two MapReduce phases. Following are the steps of the parallel FP-growth algorithm.

- Step 1: Partitioning: Mapper partition the whole database into number of small divisions, likely to equal to number of Reducers. Also maintain the counts of 1-itemset
- Step 2: Parallel Counting of 1-itemset: Each Reducers will count the items of their transaction list and combine the count from the Mapper.
- Step 3: Parallel FP-growth: Now each Reducers will build its own FP-tree using FP-growth algorithm.
- Step 4: Aggregation: Finally the results from each Reducers are combined and frequent Itemsets are discover.

2.4 Incremental Data Mining Concept

Databases are updating endlessly in practical situation, where exactly expected algorithms like Apriori, FP-growth perform incompetently. If the previous algorithms could be examine to incrementally mine the frequent itemset from the newly included item database, the drawing out process would become more proficient and cost associated with the mining process would be lessen. The process of updating database continuously is Incremental Data Mining.

Incremental mining trimmed down the cost of mining process by using again the former mined results. Correct memory consumption and speed of overall mining process are the two main factors to monitor performance of incremental data mining algorithms. Incremental data mining makes the drawing out process more competent as per as time and space requirement concern and the overall cost of the process would be diminish. Figure 2 demonstrates the overview of incremental data mining.

Cheung et. al. described the FUP algorithm, which is a more common incremental method than FUP-FUP, is efficient not only on budding of a database but also on cutting the data. The thought of Apriori algorithm is use and offered in FUP algorithm [9] to revise association rules with incremental transactions. Although it still needs to examine the original DB several times and as the original DB is always very large so it is wasteful. T. Garib et. al. presented FIM algorithm [10], to perk up the effectiveness of FUP algorithm, where only one scan for the whole original DB is needed and hence shrink the generation of candidates.
In order to obtain enhanced competence of IUA, Chen et al. projected a new enhanced algorithm AIUA [11]. He demonstrates novel function, which connects the matching frequent itemsets and evade the iteration to generate many ineffectual candidates. Yuchen et al. describes FIM_AIUA [12] algorithm, that continue the idea of both algorithms FIM and AIUA algorithm to revise association rules with incremental transactions and with threshold value changes simultaneously by combining them. Incremental Updating Algorithm also suffering from the problem of numerous scan of original database and it also requires many similar steps to generate several futile candidates. As a result it is incompetent and time-consuming. This algorithm also enliven the efficacy and precise the pitfall of My_IUA algorithm. Hong et al. proposed fast updated FP-tree (FUPP-tree) structure [13], which helps to update the tree easily. It maintains the Header_Table which helps to fasten the mining process whenever new data is inserted in the database.

3. INCREMENTAL PARALLEL FP-GROWTH

Incremental Parallel data mining unites the features of both parallelism and incremental mining to improvise the competence. It is proved that association rule mining algorithms are well known and capable in wide-ranging cases after a long study. In tangible circumstances, database is updated periodically, continuously and minimum support frequently changes with wants of mining. Conversely, when large data comes into picture, associated algorithms are not full-grown and ineffective, also needs a further exploration. It is undoubtedly useless that the full mining process has to be revived from the beginning each and every time when new data is added into database or mining parameter is retune. Furthermore, to deal with the issues resulted from large-scale data, algorithm parallelization has become certain. Wei et al. proposed parallelized incremental FP-Growth mining strategy [14] successfully explains the incremental issue brought by the dynamic threshold value and database simultaneously, which shuns repetitive computation. This mining strategy is based on MapReduce jobs.

Proposed algorithm is also combines the advantages of incremental mining and parallel mining. Parallel mining is done by making windows services run parallel and thereby, create a grid services. Grid computing has been projected as an important computational model, illustrious from the conventional computing by its focus on huge-scale resource sharing, pioneering applications, and, in some cases, high-performance direction. Jointly with the grid move towards engineering and commerce applications, a parallel dangle in the way of the implementation of data grids has been indexed.

Figure 3 illustrates the working flow of the incremental parallel FP-growth mining strategies, proposed algorithm.

![Figure 3: Flow diagram of Incremental Parallel Data mining algorithm](image)

This procedure uses three MapReduce phases. The steps of incremental parallel data mining are shown as follows.

- **Step 1:** Partitioning: Mapper partition the entire database $D$ into number of small chunks, likely to equal to number of Reducers. Also maintain the counts of 1-itemset.
- **Step 2:** Parallel Counting of 1-itemset: Each Reducers will count the items of their transaction list and combine the count from the Mapper.
- **Step 3:** Parallel FP-growth: Now each Reducers will build its own FP-tree using FP-growth algorithm during the recursive progress, the frequent itemsets are pull out.
- **Step 4:** Aggregation: Finally the results from each Reducers are combined and frequent Itemsets are discover.
- **Step 5:** During this stage, incremental database $d$ is added into database and the threshold value is set to $s'$.
- **Step 6:** In this stage, new added datasets $d$ is taken into consideration. The datasets go through a
MapReduce pass like step 1, partitioning and parallel counting.

- Step 7: According to the new frequent list, database D and d are likely to be rescanned. Mapper allocates transactions related to new common items to resultant cluster; each Reducer updates their own local FP-tree using FP-growth algorithm; new frequent items are mined in updated FP-trees.
- Step 8: The last step is to merge all the frequent itemsets from stage 7 as the final result.

The final result will give the frequent itemsets of the respective dataset.

4. EXPERIMENTAL RESULTS

In this section, proposed algorithm Incremental Parallel FP-growth (IPFP), Parallel FP-growth (PFP) and one traditional association rule mining algorithm, FP-Growth were compared and examined through experiments. All the experiments were executed on 2.40 GHz Intel i5 processor with 4GB RAM. The program code is written in C# and executed on Dot Net framework.

Table 1 shows two datasets used for testing all mentioned algorithms of association rule mining.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size(MB)</th>
<th>Transaction</th>
<th>Items</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>T10I10D136K</td>
<td>2.93</td>
<td>136,000</td>
<td>870</td>
<td>8000</td>
</tr>
<tr>
<td>Department Retail</td>
<td>5.95</td>
<td>272,000</td>
<td>999</td>
<td>16000</td>
</tr>
</tbody>
</table>

The new threshold value degrees are shown on x-axis and total time required to execute algorithm is shown on y-axis. Figure 4 and figure 6 illustrates Experiment performed on dataset 1 “T10I10D136K”, Figure 5 and figure 7 illustrates Experiment performed on dataset 2 “Department Retail”. Experiments are performed on three algorithms: FP-growth, Parallel FP-growth and Incremental Parallel FP-growth.

From the outcomes we can make out that IPFP-growth takes the smallest amount of time, comparing with other three algorithms. When data size is small, deviation is not obvious. On the other hand, as the amount of data increases, IPFP-Growth shows tremendous advantage in total running time over other three algorithms, particularly when minimum support is low. Moreover, as minimum support increases, the amount of time required to execute the algorithm is also lower down. Results show in the table 2. Values are in milliseconds.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Dataset</th>
<th>Minimum_support</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>Fp-growth</td>
<td>6186</td>
<td>5175</td>
</tr>
<tr>
<td></td>
<td>7066</td>
<td>6523</td>
</tr>
<tr>
<td>PFP</td>
<td>4171</td>
<td>2581</td>
</tr>
<tr>
<td></td>
<td>5172</td>
<td>4434</td>
</tr>
<tr>
<td>IPFP</td>
<td>2367</td>
<td>1706</td>
</tr>
<tr>
<td></td>
<td>3618</td>
<td>2628</td>
</tr>
</tbody>
</table>

Figure 4: Experiment on T10I10D136K

Figure 5: Experiment on Department Retail for execution time

As per as memory utilization is concern proposed algorithm IPFP consumes significantly less as compared to other two algorithms. Figure 6 and figure 7 shows the result of two different datasets.

From the results one can formulate that IPFP-growth takes the least amount of memory, comparing with other three algorithms. Results show in the table 3. Values are in Megabytes.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Dataset</th>
<th>Minimum_support</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>Fp-growth</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>PFP</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td>IPFP</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>25</td>
</tr>
</tbody>
</table>
6. REFERENCES


[5] Jaiwei Han, Jian Pei and Yiwen Yin- “Mining Frequent Patterns without Candidate Generation,” in Int.Conf. ACM-SIGMID. pages 1-12, June 2000.


