Application of Parallelized Apriori in Grid Computing Environment

P. Asha¹ *, Dr. T. Jebarajan²

¹Research Scholar, Computer Science and Engineering Department, Sathyabama University, Chennai, Tamilnadu, India.
²Principal, Kings College of Engineering, Chennai, Tamilnadu, India.
drtjebarajan@gmail.com

Abstract

The goal of the strategy is to improve the performance of distributed algorithms and better their responsiveness. The association rule mining algorithms has high computational complexity due to the size of its search space and the high demands of data access. The work aims at mining the data in a grid computing environment, which computes by distributing the data to its clusters and mines it in parallel. Load balancing is yet another factor to be solved while moving onto grid and cluster. An effective mining and load balancing procedure has been proposed in this paper. Finally the best rules were filtered out of all rules generated by the ARM based on various interestingness measures.

Keywords: Sequential, Parallel, Data Mining, Grid Computing, Interestingness Measures.

1. Introduction

Rules that represent an association between the values of certain attributes and those of others are called association rules. The process of extracting such rules from a given dataset is called Association Rule Mining (ARM). If the mining is done sequentially, it consumes lots of time. Moreover since the grid is heterogeneous in nature and they are distributed in nature, it is a must we move onto parallel ARM. While we try to compute the ARM in parallel, improper load balancing among the clusters exists.

In this paper, the parallel and distributed ARM algorithms and its applicability in grid computing environments as well as the load balancing [10] so as to improve the performance were analyzed. The paper also explains the rule interestingness measures which will help us in retrieving out only effective, best and interesting rules. In this work, the data mining process was deployed in a grid environment where data and resources are distributed in nature; still data can be accessible efficiently, saving time and resources by the means of effective data mining algorithms.

A literature survey on the existing load balancing techniques in grid environment and its strengths and weaknesses has been studied (Table. 1).

Few Other review on related papers are given below:


Association rule mining is one of the major technique of data mining, involves finding of frequent item sets with minimum support and generating association rule among them with minimum confidence. The task of finding all frequent item sets for a large datasets requires a lot of computation which can be minimized by exploiting parallelism to the sequential algorithms. The paper explains the preliminaries of basic concepts about association rule mining, different sequential association rule mining algorithms on different hardware platforms and also it focuses the challenges in exploiting parallelism to these algorithms [15].
Table I. Literature Review

<table>
<thead>
<tr>
<th>S.No</th>
<th>Title</th>
<th>Author(s)</th>
<th>Year</th>
<th>Strength</th>
<th>Weakness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Association Rule Mining on Distributed Data [1].</td>
<td>Pallavi Dubey</td>
<td>2012</td>
<td>Low Communication cost, Response Time and fast access of data.</td>
<td>Load imbalance.</td>
</tr>
</tbody>
</table>


It explains about the extraction of significant patterns from transaction databases and generation of rules used in many decision support applications. Many organizations such as industrial, commercial, or even scientific sites may produce large amount of transactions and attributes. Mining effective rules from such large volumes of data requires much time and computing resources. This paper proposed a parallel FP-growth association rule mining algorithm for rapid extraction of frequent item sets from large dense databases. It approves that this algorithm can efficiently be parallelized in a cluster computing environment [12].


PC cluster is recently regarded as one of the most promising platforms for heavy data intensive applications, such as decision support query processing and data mining. During development of high performance parallel mining system on PC cluster, that heterogeneity is inevitable to take the advantage of rapid progress of PC hardware [14].


The development period of PC hardware is becoming extremely short, which results in heterogeneous system, where the clock cycle, the performance/capacity of disk drives etc are different among component PC's. Two strategies, called candidate migration and transaction migration has been proposed for effective load balancing [13].

2. System Architecture

The work aims at providing association rule mining under the grid environment and load balancing at run time. Hence we improve the system performance. There are two phases: 1) ARM and 2) Load Balancing.

2.1. Association Rule Mining

The first phase, Association Rule Mining is done by Apriori Algorithm. The apriori property (downward closure property) states that any subsets of a frequent item set are also frequent item sets. There are two steps:
- Find all item sets that have minimum support (frequent item sets, also called large item sets).
- Use frequent item sets to generate rules.

Use frequent item sets to generate rules.

Step 1: A frequent item set is an item set whose support is \( \geq \minsup \). It is an Iterative algorithm. (also called level-wise search) where it finds all 1-item frequent itemsets; then all 2-item frequent item sets, and so on. In each iteration \( k \), only consider itemsets that contain some \( k-1 \) frequent item set. The items are sorted in lexicographic order (which is a total order). The order is used throughout the algorithm in each item set [8]. The generation of candidates happens with 2 steps:

**Join step**: Generate all possible candidate item sets \( C_k \) of length \( k \).
**Prune step:** Remove those candidates in Ck that cannot be frequent.

Step 2: For each frequent item set X, and for each proper nonempty subset A of X, let B = X – A.

Then, A → B is an association rule if

\[
\text{Confidence (A → B) = } \frac{\text{support (A ∪ B)}}{\text{support (A)}}
\]

In order to evaluate the performance of the workload balancing strategy the sequential Apriori algorithm which is the fundamental algorithm is parallelized for frequent set counting algorithms with candidates’ generation. To reduce the number of accesses to the transactional database the depth-first Apriori algorithm is used. This version of Apriori needs only three passes over the transactional database, while classic Apriori needs k-passes (where k is the length of the maximal itemset).

Data parallelism is not sufficient to improve the performance of association rule mining algorithms. Subsets of extremely large data sets may also be very large. So, in order to extract the maximum of parallelism, a hybrid parallelization technique is applied (i.e. the combination of data and task parallelism).

A hybrid approach between candidate duplication and candidate partitioning is used. The candidate itemsets are duplicated all over the sites of the Grid, but they are partitioned between the nodes of each site. The reason for partitioning the candidate itemsets is that when the minimum support threshold is low they overflow the memory space and incur a lot of disk I/O. So, the candidate itemsets are partitioned into equivalence classes based on their common (k-2) length prefixes.

**Algorithm:**

- \( C_k \): Candidate itemset of size k
- \( L_k \): frequent itemset of size k
- \( L_1 = \{1 \text{ frequent items} \}; \)

- for \( (k = 1; L_k = \emptyset; k++) \) do begin
- \( C_{k+1} = \text{candidates generated from } L_k; \)
- For each transaction \( t \) in database do increment the count of all candidates in \( C_{k+1} \) that are contained in \( t \).
- \( L_{k+1} = \text{candidates in } C_{k+1} \text{ with min support} \)
- End
- return \( U L_k \)

The drawback associated with the algorithm is that it requires multiple scans for finding its support count. Moreover Databases or data warehouses may store a huge amount of data to be mined. Mining association rules in such databases may require substantial processing power. One solution can be parallelization. When we go for parallel Apriori, load imbalance among the cluster exists [7].

### 2.2. Grid Environment

Grid Computing is the integrated use of distributed computing, networks, storage and other computing resources in order to create a virtual computing environment for solving large scale problems.

So they can compute the data mining tasks in parallel. As the ARM algorithms like Apriori requires many compute passes over the database, the actual running time required to complete the algorithms becomes excessive. Hence parallelizing [11] association discovery data mining becomes essential. Here comes the grid based data mining.

The central grid machine submits the mining task to the Cluster Head Node (CHN) which is connected to it and the CHN in turn are connected to Cluster Compute Nodes (CCN). The clusters compute the task in parallel fashion and return the results to the grid node (Fig. 1). Hence the performance gets increased.

![System Architecture](Figure 1. System Architecture)
2.3. Load Balancing

Dynamic load balancing is necessary for the efficient use of highly distributed systems (like Grids) and when solving problems with unpredictable load estimates (like association rule mining). So a dynamic work load balancing strategy is developed. The proposed load balancing strategy depends on three issues. Database architecture (partitioned or not), Candidates set (duplicated or partitioned), Network communication parameter (bandwidth).

The strategy could be adopted by algorithms which depend on candidate item sets generation to solve the frequent set counting problem. A parallel Apriori Algorithm is used to generate the frequent itemset. It combines between static and dynamic load balancing and this by interfering before execution (i.e. static) and during execution (i.e. dynamic).

Before execution, in order to respond to the heterogeneity of the computing system a (Grid) the database is not just partitioned into equal partitions in a random manner. Rather than that, the transactional database is partitioned depending on the characteristics of different sites, where the size of each partition is determined according to the site processing capacity (i.e., different architecture, operating system, CPU speed, etc.). It is the responsibility of the coordinator of the Grid or site Coordinator to allocate to its site the appropriate database portion according to the site processing capacity parameters stored in its information system.

During execution, the load balancing strategy acts on three levels. Level one is the migration of work between nodes of the same cluster. If the skew in workload still persists the coordinator of the Cluster Head Node (CHN) moves to the next level, Level two depends on the migration of work between clusters within the same site, And finally if work migration of the previous two levels is not sufficient then the coordinator of the overloaded CHN asks from the coordinator of the Grid or Site Coordinator to move to the third level which searches for the possibility of migrating work between sites.

Steps:
1. Calculate the workload of every individual node in a cluster: wccn (workload of cluster compute node)
   \[ wccn = \text{processing time} \times \text{communication time} \]
2. Calculate the workload of a cluster: chn (cluster head node)
   \[ chn = \text{sum of wccn.} \]
3. Let Avg represents the average workload,
   \[ \text{Avg} = \frac{chn}{\text{No. of Nodes}}. \]
4. Then for every Node, check
   - If load value exceeds the average then term it as overloaded.
   - If load value is less than the average then term it as under loaded.
   - Otherwise assume that the load is balanced.
   
   Now, the work can be migrated from overloaded node to the under loaded node (Fig. 2).

---

**Figure 2. Workflow diagram of Load balancing**
After finding the association rules, the next job is to filter out the best rules alone since there are too many rules generated by means of Apriori, and every rule doesn’t seem to be interesting and many are redundant in nature.

2.4. Finding the Best N Rules

- The support of the rule is the number of cases that contain both the L (Left hand side rule) and the R (Right hand side rule).
- The coverage of the rule is the number of cases that contain the L.
- The strength is the support divided by the coverage. This represents the proportion of the cases that contain the L that also contain the R.
- Lift of a rule, L → R measures how many more times the items in L and R occur together in transactions than would be expected if the item sets L and R were statistically independent. Lift values greater than 1 implies interesting. They indicate that transactions containing L tend to contain R more often than transactions that do not contain L.
  \[
  \text{Lift} (L \rightarrow R) = \frac{\text{count}(L \cup R)}{\text{count}(L) \times \text{support}(R)}
  \]
- Leverage of a rule, L → R measures the difference between the support for LUR (items in L and R occurring together in the database) and the support that would be expected if L and R were independent. The value of leverage of a rule is always less that its support. The number of rules satisfying the support\(\geq\minsup\) and confidence\(\geq\minconf\) constraints can be reduced by setting a leverage \(\geq 0.0001\) corresponding to an improvement in support of one occurrence per 10000 transactions in a database [9].
  \[
  \text{Leverage} (L \rightarrow R) = \text{support}(L \cup R) - \text{support}(L) \times \text{support}(R)
  \]

3. Experimental Results and Performance Evaluation

Real dataset from Yahoo Financial Data Set Repository is chosen for this performance study. The dataset is available in biz.yahoo.com\(\text{mp}\).

Fig. 3 shows the site view. Site is a collection of clusters in which each cluster is formed with the group of similar nodes. Cluster’s coordinator distributes candidate itemsets between nodes according to their capacities. Candidates are distributed by their (k-1) common prefix. This coordinator performs the global reduction of supports to obtain global frequencies as well as workload balancing in its cluster.

First find the frequent itemset by joining and pruning with the Apriori algorithm. Finally association rules are made, after which we filter those rules. As rules framed from the final frequent itemsets were redundant, it is a must that we eliminate the replicas and retain the best rules only. Hence factors like confidence, lift and leverage are used to further filter the rules and obtain only the efficient and effective rules.

Fig. 4 shows the best rules that were obtained by considering the confidence level of the frequent itemset in which the confidence is greater than the minimum confidence level and lift value greater than 1 and leverage greater than or equal to 0.0001.

![Figure 3. Site and Cluster view](image)

![Figure 4. Best Rules](image)

```
View Load balance
N3 run job --> Load
N0 run job --> Load value is: 221
N2 run job --> Load value is: 142
N3 run job --> Load value is: 1856
N2 run job --> Load value is: 638
N0 run job --> Load value is: 1964
Time in ms: 2277
```

Figure 5. Load and Total Execution time before load balancing
Fig. 5 shows the load value and the total execution time before load balancing. The load value is determined by calculating the number of job requests for each node. It depicts that the load value varies for each processor and imbalance exists. The total execution time is high since the load of each processor varies.

![Cluster Coordinator Log]  

Figure 6. Load and Total Execution time after load balancing

Fig. 6 shows the load value and the execution time of each processor after load balance. The load of each processor is nearly equal, since the job in the processor where the load is high is assigned to the processor with low load. Hence the execution time is less when compared to the time before load balance.

Fig. 7 and Fig. 8 show the execution time taken by the processors before and after load balancing.

![Load Balance Graph]  

Figure 7. Execution time of Processors before load balancing

![Load Balance Graph]  

Figure 8. Execution time of Processors after load balancing
4. Conclusion

From the performance evaluation presented above, we conclude that the proposed work has shown reduced execution time that is, the responsiveness is better when compared to the previous works. Effective data mining and better load balancing as well as efficient rules were also filtered. This project work can be further extended by, migration of check point jobs which would be restarted from last checkpoint. If so, the performance can be improved further.

References