Association Rule Mining and Load Balancing Strategy in Grid Systems

Sarra Senhadji, Salim Khiat, and Hafida Belbachir
Computer Science Department, University of Sciences and Technology of Oran Mohamed Boudiaf, Algeria

Abstract: The parallel and distributed systems represent one of the important solutions proposed to ameliorate the performance of the sequential association rule mining algorithms. However, parallelization and distribution process is not trivial and still facing many problems of synchronization, communication, and workload balancing. Our study is limited to the workload balancing problem. In this paper, we propose a dynamic load balancing strategy of association rule mining algorithm under a grid environment. This strategy is built upon a hierarchical grid model with three levels: Super coordinator, coordinator, and processing nodes. The main objective of our strategy is to ameliorate the performances of the distributed association rule mining algorithm “APRIORI”.

Keywords: Association rule mining, load balancing, grid computer, APRIORI.

Received May 21, 2012; accepted January 28, 2013; published online April 4, 2013

1. Introduction

The fast growing of collect and storage data technologies has far exceeded our human ability for comprehension without powerful tools of data mining. Different techniques of data mining are widely used in several domains to analyze and extract useful knowledge.

Association rule technique is one of the most important data mining techniques. In the literature, we found many sequential association rule mining algorithms. Unfortunately, almost of this sequential association rule algorithms suffer from a high computational complexity which derives from the size and the distribution of the enormous data set.

Grid computing is recently regarded as one of the interesting platforms for data and computation intensive applications like data mining. That is why implementation association rule mining algorithms under such environment seems to be promising.

The parallel and distributed algorithms caused many problems of synchronization, communication and workload balancing. We limit our work to the workload balancing problem, which consists of distributing tasks and data to different sites in order to have nearly the same workload in everyone.

During the implementation of parallel or distributed algorithms, the resources can be used in a bad way. Many load balancing algorithms are proposed using two methods, static (before execution) and dynamic (during the execution).

The load balancing problem becomes more complex in a grid than in a traditional distributed system. The distributed systems assure the stability and the homogeneity of resources better then a grid environment which the apparition and the desperation of the resources can be fast and abrupt in the same time.

In this paper, we propose a dynamic load balancing algorithm applied to the association rule algorithm “APRIORI” under a grid environment. The rest of the paper is organized as follows: Section 2 describes the load balancing problem, section 3 develops some load balancing algorithms applied to some distributed association rule algorithms, and in section 4 we define our dynamic load balancing strategy. Experimental results obtained from implementing this strategy are shown in section 5. Finally, a conclusion is presented to summarize the main outcomes of this research.

2. Distributed Association Rule Mining Algorithms

The association rule is one of the interesting techniques of data mining that has attracted the attention of many researchers. This technique permits to find intelligible rules trough an amount set of data. This section explains the process of extraction of association rule mining.

Considering the transactional database \( D \), an association rule has the form \( X \Rightarrow Y \), where \( X \) and \( Y \) are two itemsets and \( X \cap Y = \emptyset \).

The support’s rule is the probability of the apparition of the two itemsets \( X \) and \( Y \) in the same transaction. The confidence of the rule is the conditional probability that a transaction contains \( Y \) given that it contains \( X \).
An item is the number of the apparition of a database’s attribute; this item is frequent if its support is higher than or equal to a pre-determined minimum support. A rule is strong if its confidence is greater than or equal to a user specified minimum confidence. The extraction of the association rule mining follows two steps:

1. The first step consists of finding all frequent itemsets that occur at least as frequently as the fixed minimum support and different approaches are proposed: All item sets generation: (APRIORI [2], DIC [12], APRIORI TID [2], etc.), maximum item set generation: (Max Miner [17], Max Clique [17], Max Eclat [17] etc.), and closed item set generation: (Close [12], Closet [13], Clomaint [9] etc.).

2. The second step consists of generating strong rules from these frequent itemsets.

It is true that association rule mining algorithm has a simple statement, but the first step is computationally intensive because of the phenomenal number of generated itemsets [6]. Many sequential algorithms for finding large itemsets have been proposed in the literature and most of them use “APRIORI” as fundamental algorithm [2].

Parallelism of sequential itemsets generating algorithms can play an important role in improving the execution time. This kind of parallel algorithms follows two main paradigms: Data repartition and tasks repartition. Many parallel algorithms for finding itemsets have been proposed in [15] and we can find the pioneers ones: Count Distribution (CD) [1], Data Distribution (DD) [1], Hybrid Distribution (HD) [1], etc.

3. Load Balancing Problem for Parallel Association Rule Mining Algorithms

In general, Work load balancing is the work assignment to processors to achieve goals such as minimizing communication cost and maximizing throughput. As load imbalances in dynamic systems occur performances can be improved by either transferring jobs from the currently heavily loaded hosts to the lightly loaded ones among the nodes [5, 8]. In this part we cite some of the existing algorithms of load balancing for parallel association rule mining.

The Intelligent Data Distribution (IDD) algorithm [10] is an amelioration of the DD algorithm which is a parallelization of the Apriori algorithm. This algorithm solve the communication model problems of the DD algorithm that use an all-to-all communication, by using an virtual circle topology of processors to ameliorate the performance and to reduce the communication’s costs, which is based on an asynchrony communication peer to peer between neighbors of the topology. Each processor disposes of two buffers: SBuf to send the data and RBuf to receive the data. Each processor initiate an synchronic operation of sending data to its right neighbor using SBuf, and an synchronic operation of receiving data from the left neighbor using RBuf. During these operations, each processor owns the transactions in the SBuf and calculates the assigned candidate’s support.

The type of the load balancing in this algorithm is static and is limited to the candidate’s partition. For this, we calculate for each item the number of candidates itemsets where the item is the prefix. After, in a system of p processors, the algorithm distributes equitably in p parts the prefix itemsets. Then, each processor generates locally and stores its itemsets. In this approach, when the number of the candidate’s itemsets beginning with the same item exceeds a threshold value, it will be very difficult to have equitable repartitions. To solve this problem the second candidate item is considered in the partitions.

The IDD algorithm can’t guarantee a good load balancing by assigning the same number of candidates to all the processors. In fact, the prefix items that are assigned to each processor and which have a higher support can receive more requests of calculation than the others. This generates an unbalance that can increase considerably when the number of processors is very higher comparing to the number of item sets.

The Load balancing Frequent Pattern Tree (LPFP-Tree) algorithm [11] is a load balancing algorithm applied to the Parallel FP-Tree (PFP-Tree) algorithm [15], which is an amelioration of the FP-Growth algorithm [7].

Load balancing is achieved dynamically in the LPFP-Tree algorithm. The strategy used in LPFP-Growth algorithm follows an equitable repartition of the FP-Tree structure over different nodes. For this, an FP-Tree branches migration strategy engenders the conditional databases migration, especially when a tree is very huge and very profound. For this, in the FP-Tree algorithm the execution time in each node depends on the number of items in the tree.

The load is calculated for each item using a formula based on the depth (Lj). Each processor sends its local load information to the coordinator node in order to detect eventual unbalancing cases. The suppression of the redundant branches of the structure by regrouping in one node reduces the memory space storage. Otherwise, the experimental results show that the value of threshold depth influences on the algorithm performances. In fact, when the threshold depth is lower, an important number of small conditional databases must be stored, this increases the work load; and when the threshold depth is higher, the load node is not balanced.

Our work is described in next section with the adopted grid model and the load balancing strategy used to ameliorate the performances of the distributed APRIORI algorithm.
4. Proposed Grid Model

In our study we model a grid as a set of n sites (clusters) managed by a super coordinator. Each site owns a set of worker nodes and a coordinator node, as shown in Figure 1.

![Figure 1. Proposed grid model.](image)

The super coordinator assures the following functions:

- Initial distribution of the global database to each site of the grid.
- Collection of the state load information from all the coordinators site.

The coordinator of each site assures the following operations:

- Collect the local state load information from the worker’s coordinator nodes.
- Calculate and send the global load to the super coordinator of the grid.
- Decide of the transaction’s migration and/or the candidate’s migration:
  - **Candidate’s Migration:** In intra-site (locally) transfer the candidates from the up loaded node to the under loaded node.
  - **Transaction’s Migration:** In inter-site (globally) when all the site nodes are overloaded then we decide to transfer a part of the transaction’s partition to another site respecting the following conditions:
    - The less loaded site.
    - The nearest site with a least cost of communication.

The worker nodes assure the following operations:

- Maintain their local load information.
- Send their local load information to their coordinator.
- Execute the load balancing decision plan sent by their coordinator.

4.1. Load Estimation

In our approach, we define three load states: under loaded, medium loaded and up loaded. In addition, we assign a numeric value to each state as follows:

- **Under Loaded 1:** The grid resource is inactive or down loaded.
- **Medium Loaded 2:** The grid resource is midway loaded.
- **Up Loaded 3:** The grid resource is overloaded.

According to the proposed grid model, we define the load state over three levels: Worker nodes, coordinator nodes and super coordinator.

- **Worker Nodes:** The node load is calculated following the length of the file (size n); this file contains the candidate’s itemset. For this, we define two thresholds (low and high) and we distinguish three cases:
  - Case 1: If file = not empty or n ≤ low then load = 1
  - Case 2: If file = not empty and low < n < high then load = 2
  - Case 3: If file = not empty and n ≥ high then load = 3

- **Coordinators:** The site load is calculated as the average of the state load information sent by all the workers nodes of the site i. We distinguish the following cases:
  - Case 1: If 1 = < average < = 1.5 then load = 1
  - Case 2: If 1.5 < average < 3 then load = 2
  - Case 3: If average = 3 then load = 3

- **Super Coordinator:** The grid load is estimated as the average of the state load information collected from each coordinator site. We distinguish there cases:
  - Case 1: If 1 = < average < = 1.5 then load = 1
  - Case 2: If 1.5 < average < 3 then load = 2
  - Case 3: If average = 3 then load = 3

4.2. Proposed Strategy

The associative rule mining algorithms are characterized by their dynamic nature. In fact, this kind of algorithms has a workload that is unpredictable during the computation, which is due to the high degree of correlation between itemsets generated at each iteration. Hence, an unbalance workload is detected. To correct this unbalance, we choose to develop a dynamic load balancing strategy based on the extraction frequent itemset algorithm APRIORI.

According to the proposed grid model, we distinguish two cases: intra-site load balancing (between the worker nodes of the same site) and inter-site load balancing (between the grid sites).

1. **Intra-Site Load Balancing:** Consists of transfer the candidates from an overloaded node to another site less loaded, respecting the following constraint:

   \[ \text{Load } i, k > \text{Load } j, k + \text{CCN } i, j, k \]  

   With:
   - Load \( i, k \): The estimated load of the source node \( i \) of the site \( k \).
Load \( j, k \): The estimated load of the target node \( j \) of the site \( k \).

CCN \( i, j, k \): The communication cost between the node \( i \) and the node \( j \) of the same site \( k \).

2. **Inter-Site Load Balancing:** When all the nodes of a site are overloaded, we decide to transfer a part of the transaction’s partition, that contribute in the calculation of the candidate’s support (the transactions that contain the candidate’s item) to another site less loaded and with a less communication cost.

\[
\text{Load } i, k > \text{Load } j, p + \text{CCS } k, p
\]  

(2)

With:

- **Load \( i, k \):** The estimated load of the source node \( i \) of the site \( k \).
- **Load \( j, p \):** The estimated load of the target node of the site \( p \).
- **CCS \( k, p \):** The communication cost between the site \( k \) and the site \( p \).

Otherwise, the cost of the intra-site transfer is less important than the cost of the intra-site transfer communication, that’s why we privilege the local load balancing to the global load balancing.

### 4.3. Load Balancing Strategy

Our strategy follows two global phases: Initialization phase and execution phase.

- **Initialization Phase (Before the Execution)**
  1. The global transaction’s data base is partitioned equitably by the super coordinator over all the grid sites, by dividing the number of the transactions by the number of the grid site.
  2. Determination and duplication of the candidates 1-itemset over all the grid sites.
  3. Each coordinator site partitioned the 1-itemsets candidates over their worker nodes, by dividing the total number of the candidates by the total number of the worker nodes. Each candidate is stored in the file of the worker node.
  4. Initialization of the load to the value 1 (under loaded) for all the grid nodes.

- **Execution Phase (During the Execution)**

  Repeat
  
  Each node:
  
  Calculates the \( k \)-itemsets candidate’s support from the portioned transactional database.
  
  Sends the result of the calculated \( k \)-itemset’s support to his coordinator.

  Each coordinator:
  
  Sends the candidates \( k \)-itemsets support to the super coordinator.

  The Super Coordinator
  
  Determines the frequent \( k \)-itemsets (support > min support).

  Generates then duplicates the \( k \)-itemsets candidates over all the sites.

  **Until all itemsets founded.**

**In parallel:**

The following operations are invoked during the generation of the candidates:

- Each node sends his local load information to his coordinator.
- Each coordinator updates then sends the load vector of his worker nodes to the super coordinator.
- The super coordinator is responsible of the detection of the eventual load unbalancing cases according to the following algorithm:

- **Calculation of the load average of each site.**
- **For each site verify the following conditions:**
  - If a site is up loaded (load=3) then
  - If exists another site less loaded with a minimal communication cost then transaction’s migration.
  - Else (the site is not overloaded) If a node is overloaded (load=3) then
  - If exists another node of the same site less loaded with a minimal communication cost then candidate’s migration.

For example, in Figure 2 we present a topology of 1 super coordinator SC, 3 sites with a coordinator <C1, C2, C3> and 8 processing nodes <N1, N2, ..., N8>.

Initially the global database is partitioned over the 3 sites. The 1-itemeset candidates are duplicated over the 3 sites and distributed to the processing nodes of each site. The load of nodes is under loaded (represented in the vector \( V \) with the value 1).

![Figure 2. Initialization phase.](image-url)
At the end, the calculated supports of the candidates are collected and the frequent ones are returned to the super coordinator, as shown in Figure 4.

Periodically, the $k + 1$ itemsets candidates are generated and replicated over the sites until all itemsets had been founded.

5. Experiments

We evaluate the performances of our strategy by using the grid simulator “Gridsim toolkit 4.2” [4] under the operating system “Windows XP SP 2”. In addition, for the generation of the frequent itemsets, we build a database that contains 6 items and 500000 transactions.

First, we implement a paralleled version of the algorithm “APRIORI” without load balancing. For this, we realize different experimentations, using several parameters (#items, #transactions and min support), shown in Table 1.

In the second time, we analyze the performance of our proposed approach to correct the unbalance of load.

Table 1. Parameter’s configuration.

<table>
<thead>
<tr>
<th>Global Transactional Data Base</th>
<th>MIN SUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Items #Transactions</td>
<td></td>
</tr>
<tr>
<td>6 10000</td>
<td>10, 20, 30, 40</td>
</tr>
<tr>
<td>6 25000</td>
<td>10, 20, 30, 40</td>
</tr>
<tr>
<td>6 50000</td>
<td>10, 20, 30, 40</td>
</tr>
</tbody>
</table>

5.1. Experiment 1

In this case, we consider a topology of: 1 super coordinator, 3 sites and 6 nodes, using the different parameter’s configuration shown in Table 1. The obtained experimentations are represented in Figures 5, 6 and 7.

The obtained results show that the execution time is nearly reduced with the use of our strategy of load balancing. In the next experimentation we increase the number of nodes to observe the gain in execution time.

5.2. Experiment 2

In this second case, we consider a topology of: 1 super coordinator, 4 sites and 8 nodes, using the same parameter’s configuration.

The obtained experimentations are represented in Figures 8, 9 and 10.
We observe that our strategy reduces the execution time.

6. Conclusions
In this paper, we presented the distributed data mining generalities, and then we detailed the technique of association rule mining. Then, we presented different strategies used in the load balancing. After, we defined our load balancing approach, following a hierarchical grid model with three levels “super coordinator, coordinator, worker nodes”.

Finally, in order to show the advantages of our strategy, we simulate our strategy by using “GridSim” with different experimentations by manipulating different parameters (the minimal support, the number of transactions, the number of sites, and the number of nodes). The results obtained showed the reduction of time execution for associative rule mining.

Our work offers many research perspectives, that’s why we present the most interesting ones:
1. Follow a policy that assures the fault tolerance of the nodes, especially the super coordinator node and the coordinator nodes.
2. Implementation of another extraction itemsets approach like the close frequent items.
3. Implementation of our strategy in a real grid.

References


**Sarra Senhadji** holds Master degree in computer science in 2009. Currently, she is a PhD student since 2009 at the University of Science and Technology of Oran, Algeria. Her research interest includes data mining, data replication, and data grid.

**Salim Khiat** holds Master degree in computer science in 2007, engineer degree in computer science in 1996. He has a membership of LSSD (Laboratory Signal, System and Data). His research interest includes databases, data mining, ontology, grid, and cloud computing.

**Hafida Belbachir** is working currently at the Department of Computing, USTO, Oran, Algeria. She hold PhD degree in computer science in 1990. Her research interest includes advanced databases, data mining, and data grid. She is co-director of Signal, Systems and Data Laboratory, and the head of the database system group.