A Scalable Strategy for Mining Association Rules under Grids

R. Tlili\textsuperscript{1}, K. ElBedoui\textsuperscript{2}, and Y. Slimani\textsuperscript{3}

\textsuperscript{1}Department of Computer Science, Faculty of Sciences of Tunis (FST), Campus Universitaire, Tunis, Tunisia
\textsuperscript{2}Department of Computer Science, Faculty of Sciences of Tunis (FST), Campus Universitaire, Tunis, Tunisia
\textsuperscript{3}Department of Computer Science, Higher Institute of Multimedia Arts of Manouba (ISAMM), Campus Universitaire, Manouba, Tunisia

Abstract—Sequential Association Rule Mining (ARM) algorithms are characterized by a high computational complexity due to two facts: (i) they have to mine a very large search space (ii) they have high demands of database access. Association rule mining technique have progressively been adapted to large-scale systems in order to benefit from the large-scale computing capabilities and the huge storage capacity provided by these systems. Performance issues (i.e., efficiency and scalability) are determinant factors for association rule mining algorithms [1].

In this paper we present an important part of our multilevel strategy that aims to improve the scalability of distributed ARM algorithms. Our main goal is to obtain a running time that grow linearly in proportion with the size of the database, given the available system resources (i.e., available computing nodes, their main memory and their disk space, etc.). The French research grid "Grid'5000" is used as our experimental test-bed.

Keywords: Distributed association rule mining, running time, Scalability, Grid'5000.

1. Introduction

Association rules mining is one of the most well studied data mining techniques [2]. It was first introduced by Agrawal for transaction data analysis [3]. Today, this technique is used in a wide variety of applications such as intrusion detection, heterogeneous genome data, mining remotely sensed images/data, product assortment decisions, telecommunication networks, market and risk management, etc [4].

Extracting useful knowledge from data sets measuring in Petabytes is a challenging research area for the data mining community. Sequential approaches suffer from a performance problem due to the fact that they have to mine large and also in the majority of real-world cases distributed databases [5], [6], [7]. Parallelism is introduced as an important solution that could improve the response time and the scalability of these approaches. However, because the parallelization process is not a trivial straightforward process, it introduces a plethora of new problems including the scalability problem [8], [9].

Although Grid computing systems share many aspects with parallel and distributed approaches, there are platform peculiarities and requirements implying extra efforts and new methodologies to deal with the heterogeneity of such systems. Running classical parallel and distributed algorithms under Grid systems will degrades their performance due to the load imbalance that appears between resources during execution time [9].

The research work of our paper is targeted to cope with the challenges brought by running association rule mining algorithms under grid computing environments. We embedded our multilevel scalability improving strategy into different parallel association rule mining algorithms. The rest of the paper is organized as follows: Section 2 introduces the multilevel strategy for improving the ARM algorithm scalability. Section 3 presents the processing level of our strategy. Sections 4, 5 and 6 gives the details and the different stages of the processing level. Experimental results obtained from implementing this strategy are showed in section 7. The paper concludes with section 8.

2. The multilevel strategy for improving the ARM algorithm scalability

The objective of the researcher in the domain of knowledge discovery is to design high performance ARM algorithms. Efficiency and scalability are the two determinant factors for the performance of ARM algorithms. For an algorithm to be scalable, its running time should grow linearly in proportion to the size of the database, given the available computing system resources [2]. So, scalability has two main dimensions: (i) The very large and incremental size of the databases that have to mined. An efficient ARM algorithm must be capable of extracting knowledge in a reasonable time even when the size of the database is continuously increasing. (ii) The ARM algorithm must scale very well when the number of computing resources provided by the grid increases. It must be capable of dealing with different synchronization barriers and results consolidations that must be done by the end of each iteration, without degrading the overall execution time. Both the dynamic behavior of ARM algorithms and grid characteristics have an impact on the design of our multilevel scalability improving strategy.
The scalable strategy is mainly based on three levels or axes of interference. Figure 1 depicts the sequence formed by these levels:

1) **Preparatory level**: Responsible of adapting the distributed ARM algorithm to the hierarchical grid model [10].

2) **Preventive level**: Since the workload imbalance has a direct impact on the scalability of the distributed algorithm [1], [11], the intervention on this axis aims to minimize the probability of workload imbalance during processing time. This is achieved by introducing a new data partitioning approach which takes into account the particularities of both ARM technique and grid environment [12].

3) **Corrective level**: This axis represents the processing phase of the multilevel scalability improving strategy. In this level the workload imbalance observed during the execution is corrected dynamically. This level is very critical and delicate, because if workload imbalance corrections are not delicately measured they could increase the overall execution time instead of decreasing it. These corrections are the key idea that help the distributed ARM algorithm to scale with the increasing size of the database and also the increasing number of computing nodes incorporated in execution. This level will be explained in details in what follows.

These three levels form together a complementary homogeneous sequence that aims to ensure a proper execution of the ARM algorithm under a grid computing environment, while maintaining workload balancing and providing rational exploitation of all available computing resources. The third level (i.e., the corrective level also called the processing level) is the longest phase in the process with a dynamic and evolving behavior. The processing level will be explained in details in what follows.

### 3. Processing level

The specific characteristics of ARM algorithm with those of the computing environment (Grid) must be taken into account. While association rule mining method is based on global supports, we are only disposed by partial supports at the end of each iteration. This is due to the fact of dataset distribution. Synchronization barriers are then necessary in order to obtain global supports of itemsets. The difference in processing capabilities of the components of the grid joined with the difference in the amount of work needed for each data partition lead to high time delays behind synchronization barriers. Our goal is to reduce the idle time periods of different computing resources by inducing workload balancing. This will allow the distributed algorithm to benefit as much as possible from the available computing power. The processing phase of our strategy will be presented in three parts:

1) **What to balance**: defining the concept of "workload" in ARM algorithms.
2) **Where to balance**: defining the levels of workload balancing (i.e., load balancing hierarchies).
3) **How to balance**: the modules of the workload balancing application.

### 4. What to balance?

Our goal is to control the workload of the ARM algorithm running under the grid and to balance it if necessary. So, it is necessary to start by defining the work (i.e., ARM tasks) that we want to control and balance and then finding a way to measure it. Each entity of our grid (node, cluster and site) has a specific workload that changes (increases or decreases) during the processing phase. As the node is the basic entity of a grid, we first define the workload at a given node.

#### 4.1 Node level

The workload of a node is the ratio between the computing capacity of the node (expressed in term of processing units also called intructions per second) and the amount of work to be done (expressed in the number of transactions to be processed and the number of generated candidate itemsets). The triplet of information (capacity calculation, number of transactions, number of candidate itemsets) allows us to deduce the necessary elements to define the workload of a node. The most important criterion for a node is not the amount of work already done but rather the amount of remaining work. Hence, we define the workload of a node, denoted by \( W/L_{ijk} \), as the amount of work that still have to be done, expressed in number of processing units needed to perform the remaining work. Starting from the speed of a node \( N_{dijkl} \), expressed in instructions per second, and knowing the number of processed and remaining transactions of this node at a given instant, and the time interval separating two periods of workload information...
calculation, we define first $PuTr_{ijk}$ which is the number of processing units required to process a transaction $Tr_i$.

$$PuTr_{ijk} = IS(Nd_{ijk})/NPtr_{ijk}$$

Thus, we can deduce the workload of a node as follows:

$$WL_{ijk} = PuTr_{ijk} * NRT_{ijk}$$

(2)

To ensure a permanent control over all entities of the grid, each node send periodically to the coordinator of the cluster to which it belongs, its workload $WL_{ijk}$. This communication is done under the form of a set of information, which we call "Node Workload State Vector" ($NWSV_{ijk}$).

4.2 Cluster level

For a cluster, two important information are calculated:

- The workload of the cluster: as each cluster $Cl_{ij}$ is composed of a fixed number of nodes. Then, its workload can be deducted from summing the workloads of its nodes received through the $NWSV_{ijk}$ of each $Nd_{ijk}$.
- The average workload of the cluster is defined by the following equation:

$$WL_{ij} = \frac{\sum_{k=1}^{N_{ij}} WL_{ijk}}{N_{ij}}$$

(3)

We define the computational workload of the cluster as the average workload of its nodes instead of the sum of these workloads because different clusters do not have the same number of nodes and hence the sum of the workloads can not be an eligible factor of comparison. So, the average workload allows us to deduce the workload state of the cluster regardless of the number of nodes constituting it.

At this level it is important for the coordinator $Coord(Cl_{ij})$ of the cluster $Cl_{ij}$ to have an idea about how much the workloads of the nodes under its control are spread out with respect to the computational workload $WL_{ij}$ of the cluster. For this, we define an interval based on the standard deviation.

The standard deviation of a cluster $Cl_{ij}$, denoted by $\sigma_{ij}$, shows how much variation or "dispersion" exists from the average workload. The standard deviation is the square root of the variance. The variance is a measure of how far the set of workloads are spread out and is calculated as the average of the squared differences from the mean.

Therefore, the workload of the cluster and the standard deviation calculated from the workloads of different nodes, allow the coordinator of the cluster to define two important thresholds as follows:

- The standard deviation associated to cluster $Cl_{ij}$ is defined by:

$$\sigma_{ij} = \sqrt{\frac{1}{N_{ij}} \sum_{k=1}^{N_{ij}} (WL_{ij} - WL_{ijk})^2}$$

(4)

- The $ThresMax$ determines the threshold above which a node is considered to be overloaded and needs a workload balancing phase.

$$ThresMax = WL_{ij} + \sigma_{ij}$$

(5)

- The $ThresMin$ allows to determine if a node is underloaded or not. If it is the case, it can receive an amount of work from an overloaded node.

$$ThresMin = WL_{ij} - \sigma_{ij}$$

(6)

Fig. 2: Workload states of computing nodes

These three values allow to the coordinator of a cluster $Cl_{ij}$ to divide its nodes into three classes (see Figure 2):

- Underloaded class formed by all nodes with a workload below $ThresMin$ (like the node $N3$ in Figure 2). The nodes of this class have the capacity of receiving additional work and therefore they represent the "Green Area" for the coordinator of the cluster.
- Balanced class: groups nodes with a workload between $ThresMin$ and $ThresMax$ (like nodes $N1$ and $N2$ in Figure 2). The workload of these nodes is close to the average workload of their cluster; therefore they are in a balance state.
- Overloaded class which is formed by all nodes having a workload above the $ThresMax$ (like node $N4$ in Figure 2). The workload of these nodes need to decrease their workloads through workload balancing operations. They represent the "Red Area" for the coordinator of the cluster. During a workload balancing operation, these nodes send part of their work to underloaded nodes.

4.3 Site level

The same logic applied to the cluster level is also used to define the load of a given site. Therefore, the workload of a site is equal to the average load of

The workload of site $S_i$ will be given by the following equation:
\[ WL_i = \sum_{j=1}^{M_i} WL_{ij}/M_i \quad (7) \]

Hence, the same principle used in classifying the nodes of a cluster into underloaded, overloaded and balanced will be applied to classify the clusters of a site as follows:

- The standard deviation associated to site \( S_i \) is defined by:
  \[ \sigma_i = \sqrt{1/M_i \cdot \sum_{j=1}^{M_i} (WL_i - WL_{ij})^2} \quad (8) \]

- Clusters with workloads greater than \( ThresMax \) are considered to be overloaded.
  \[ ThresMax = WL_i + \sigma_i \quad (9) \]

- Clusters with workloads less than \( ThresMin \) are considered to be underloaded.
  \[ ThresMin = WL_i - \sigma_i \quad (10) \]

- Clusters between \( ThresMin \) and \( ThresMax \) are in a balanced state.

4.4 Grid level

At this level, the workload of a grid \( G \) is considered to be the average workload of its sites and is calculated by the following equation:

\[ WL_G = \frac{\sum_{i=1}^{T} WL_i}{T} \quad (11) \]

Where, \( T \) is the total number of sites in the grid \( G \) and \( WL_G \) is the computational workload of \( G \).

The sites of the grid \( G \) could be classified to underloaded, overloaded and balanced as follows:

- The standard deviation associated to the grid \( G_i \) is defined by:
  \[ \sigma_G = \sqrt{1/T \cdot \sum_{i=1}^{T} (WL_G - WL_i)^2} \quad (12) \]

- Sites with workloads greater than \( ThresMax \) are considered to be overloaded.
  \[ ThresMax = WL_G + \sigma_G \quad (13) \]

- Sites with workloads less than \( ThresMin \) are considered to be underloaded.

5. Where to balance: defining the workload balancing hierarchy?

Based on the hierarchical modeling of the grid, defined in the preparatory level of our scalability improving strategy [10], is build a hierarchical and distributed load balancing plan. This plan is constituted of the following three levels:

1) **Intra-cluster workload balancing:** this first level concerns the migration of work between nodes of the same cluster. The coordinator of the cluster \( Coord(Clij) \) privileges primarily this local workload operation. It seeks through workload state vectors of its nodes to find the appropriate underloaded nodes that can alleviate the load of an overloaded node. If workload imbalance still persists, the coordinator of the cluster moves to the next level.

2) **Intra-site workload balancing:** this level depends on the migration of work between clusters within the same site. The site coordinator decides to initiate a workload balancing operation based on load information (workload state vectors) sent periodically by the coordinators of clusters. The site coordinator gives priority for balancing the workload by redistributing it locally between clusters which are under its control. This approach of locality aims to reduce the communication cost, by avoiding inter-sites communications which use the WAN network.

3) **Inter-sites workload balancing:** the workload balancing, at this third level, is triggered when some sites coordinators fail in their attempts to balance workload locally through their respective sites. The failure of local workload balancing may be due to the saturation of the site, or to insufficient charge offer induced by the lightly-loaded cluster with respect to the request formulated by overloaded nodes. In this case, the site coordinator tries to find another site which is able of accepting the current overload. This search is accomplished by negotiating the transfer of candidate itemsets, transactions or both from the overloaded site to the underloaded one.

A negotiation process is launched to find the appropriate nodes (either local or remote node) able to relieve the overloaded nodes. This process takes into account the communication factor in order to guaranty that the workload balancing process will improve the performance of the ARM algorithm running under the grid environment.
6. How to balance: application modules

The role of a node is limited to two basic functions:

- A support counting function which is ensured by the "DDMA" module.
- A monitoring function which consists of periodically generating the workload state vector of the node and sending it to the cluster coordinator. Other communication aspects between a computing node and its coordinator of cluster may be useful as receiving new candidate itemsets or a reduction in the number of candidate itemsets during execution. All these functionalities of monitoring and communication are provided by the module "Monitoring".

The functionalities of a Coordinator (cluster/site) are more complex than the ones at the node level:

- An initialization function, ensured by the "Initialization" module. This module covers the preprocessing phase of our workload balancing approach.
- A consolidation function where partial results provided by the "DDMA" module are exchanged. This function is done by the "DDMA" module.
- A monitoring function on the entities which are under the responsibility of the relevant coordinator: this function is provided by the module "Monitoring". In addition to that, this module ensures all internal and external communications of the coordinator with other grid entities, decision making when local workload balancing is needed or when receiving an external demand of inter-sites workload balancing.
- A workload balancing function which is performed by "Load Balancing" module. This module executes the workload balancing decisions made by the "Monitoring" module.

The previously defined structures have a well-defined logic of communication during execution time. The modules of each structure collaborate internally (within the same entity) and externally (with other entities of the grid) via communication links as shown in Figure 3.

All communications are based on a message exchange system that has been established for the purpose.

The module "Initialization" initializes the launching of execution by preparing the partitioning and the distribution of the transactional database. This module communicates with its corresponding "Initialization" module of the entity which is directly related to it: Site-Cluster and Cluster-Node.

The module "DDMA" executes the various tasks of the ARM algorithm like the generation of candidate itemsets, computing their frequencies, exchanging and consolidation of results between different coordinators. This module communicates with its corresponding "DDMA" module of other entities and also with the module "Workload Balancing" in the same entity to accept or delegate a new part of data to be processed in case of workload imbalance.

The module "Monitoring" controls the progress of execution at each grid entity. It ensures the exchange and the control of workload state vectors exchanged during execution. This module communicates with two modules: "DDMA" and "Workload Balancing" of the same entity.

The module "WorkLoad Balancing" intervenes when the module "Monitoring" detects a workload imbalance. This module involves different workload balancing levels provided by our strategy: intra-cluster, intra-site and inter-sites workload balancing. This module communicates with the module "DDMA" of its entity in case of transactions or candidates migration (i.e., addition/deletion of data or candidates to/from the entity). It also communicates with the module "Load Balancing" of other entities when remote workload balancing is needed to negotiate migration.

The module "Initializing" initializes the launching of execution by preparing the partitioning and the distribution of the transactional database. This module communicates with its corresponding "Initialization" module of the entity which is directly related to it: Site-Cluster and Cluster-Node.

The module "DDMA" executes the various tasks of the ARM algorithm like the generation of candidate itemsets, computing their frequencies, exchanging and consolidation of results between different coordinators. This module communicates with its corresponding "DDMA" module of other entities and also with the module "Workload Balancing" in the same entity to accept or delegate a new part of data to be processed in case of workload imbalance.

The module "Monitoring" controls the progress of execution at each grid entity. It ensures the exchange and the control of workload state vectors exchanged during execution. This module communicates with two modules: "DDMA" and "Workload Balancing" of the same entity.

The module "WorkLoad Balancing" intervenes when the module "Monitoring" detects a workload imbalance. This module involves different workload balancing levels provided by our strategy: intra-cluster, intra-site and inter-sites workload balancing. This module communicates with the module "DDMA" of its entity in case of transactions or candidates migration (i.e., addition/deletion of data or candidates to/from the entity). It also communicates with the module "Load Balancing" of other entities when remote workload balancing is needed to negotiate migration.

Figure 3, illustrates an example of exchanged messages between the entities of the grid. In this flow we distinguish two kind of messages:

- Data messages:
  - Partition: partitions of the transactional database.
  - Results: frequencies of itemsets.
  - Migration intra/inter-Site: candidate itemsets migrating between nodes of the same cluster and transactions migrating between clusters/sites.

- Control messages:
  - CSV: Cluster workload State Vector.
  - NSV: Node workload State Vector.
  - SOS_Intra_Site: Intra-site workload balancing request. This type of message is initiated by a site coordinator, to all entities under its responsibility, to search primarily for a possibility of local balancing. This is an internal message of the site.
  - SOS_Inter_Site: Inter-sites workload balancing request. This message is launched by a site coordinator for other site coordinators, to search for the possibility of balancing inter-sites. This message is triggered when the message "Response SOS Intra-
Site” returns a failure, which means that there is no possibility of local workload balancing. In the next section, we will evaluate the performance of the perviously detailed strategy using the Grid'5000 platform.

7. Performance evaluation

The performance evaluations presented in this section were conducted on Grid’5000 [13], a dedicated reconfigurable and controllable experimental platform featuring 13 clusters, each with 58 to 342 PCs, interconnected through Renater (the French Educational and Research wide area Network). It gathers roughly 5000 CPU cores featuring four architectures (Itanium, Xeon, G5 and Opteron) distributed into 13 clusters over 9 cities in France (Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia-Antipolis, and Toulouse) [13].

In order to do our tests, we implemented a parallel version of the DCI algorithm [7]. According to tests conducted during the FIMI workshop 1, DCI algorithm is found to be one of the fastest ARM algorithms. It can be used efficiently to find frequent itemsets even with very low support values. This algorithm adopts a hybrid approach to determine the supports of frequent itemsets, by exploiting a counting-based method during first iterations (i.e., it scans a horizontal transactional database layout), and an intersection-based method when the dataset can fit into main memory (i.e., during this phase, it scans a vertical transactional database layout). This algorithm also uses simple data structures with direct access for storing candidate itemsets. Then, we embedded our multilevel strategy within the parallel DCI algorithm. The datasets used in tests are synthetic, and are generated using the IBM-generator. Table 1 shows the datasets characteristics.

<table>
<thead>
<tr>
<th>Database</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># Items</td>
</tr>
<tr>
<td>DB5250M12235T</td>
<td>250 000</td>
</tr>
</tbody>
</table>

The following grid configurations are used:

- For the case of 2 and 4 nodes, there are: 1 coordinator, 1 site, 1 cluster.
- For the case of 8 nodes, there are: 2 coordinators, 1 site, 2 clusters, 10 cores.
- For the case of 16 nodes, there are: 4 coordinators, 1 site, 4 clusters, 20 cores.
- For the case of 32 nodes, there are: 7 coordinators, 2 sites, 5 clusters, 39 cores.
- For the case of 64 nodes, there are: 12 coordinators, 3 sites, 9 clusters, 76 cores.

The table displayed in figure 4 illustrates the execution time, the speedup and the efficiency of the DCI algorithm with and without the multilevel scalability improving strategy. For abbreviation, "NoLB-DCI” refers to the parallel version of DCI algorithm without workload balancing, while "LB-DCI” refers to the workload balanced DCI. From figure 5, we can notice that the execution time of NoLB-DCI with 64 nodes is equal to 3480 sec, while the execution time of LB-Apriori is equal to 2630 sec. This gives us a gain in the time of execution of about 24.43%. Also, as the number of computing resources increases from 2 to 64 the execution time of LB-DCI is remarkably decreasing. The optimal execution time of NoLB-DCI, for DB5250M12235T dataset, is obtained with 64 nodes. Beyond this number of nodes, the execution time starts increasing again. With LB-DCI, we still have an increasing improvement in execution time. This shows that LB-DCI scales very well with the increase in the number of computing nodes. As depicted in figure 6, the speedup of NoLB-DCI with 64 nodes is equal to 48.01, while the speedup of LB-DCI is equal to 63.53. This means that the LB-Apriori benefits more from available computing resources. The number of computing nodes incorporated in execution must be proportional to the amount of work needed by a specific database and with a specific support value. This means that, when an optimal value of execution time is reached experimentally, then there is no need to further increment the number of used nodes. The need of workload balancing increases as the number of nodes increases. From figures 5, 6 and 7 we can notice that our multilevel strategy offers great savings in processing time and improves the scalability of the parallel ARM algorithm.

![Fig. 6: Speedup of DCI algorithm with and without load balancing](image)
8. Conclusions

ARM algorithms are both data and computationally intensive. The expected benefits from embedding association rule mining into a grid environment are execution time acceleration and scalability. With workload imbalance, the distributed ARM algorithm can only execute at the speed of the most heavily loaded computing node. In this paper we proposed a multilevel strategy that corrects the skew in workload that occurs during the execution of the ARM algorithm under the goal of improving the scalability.

Experiments, using Grid’5000 platform, showed that our approach can successfully maintain the workload balance and this leads to an improvement in the scalability in terms of the amount of treated data. It also helps the distributed algorithm in benefiting as much as possible from the large-scale computing capabilities provided by a grid environment (i.e., the scalability in terms of computing resources).

References