A Novel Data Partitioning Approach for Association Rule Mining on Grids

Raja Tlili[1] and Yahya Slimani[2]
[1,2]*MOSIC, Computer Science Department, Faculty of Sciences of Tunis*
Tunis El Manar University, Tunis, Tunisia
[1]*raja_tlili@yahoo.fr*, [2]yahya.slimani@fst.rnu.tn

Abstract

Mining association rules refers to extracting useful knowledge from large databases. Algorithms of this technique are both data and computation-intensive, which make grid platforms very attractive for them. However, to exploit these platforms, new data partitioning features are required where the specificities of both association rule mining technique and grids must be taken into consideration.

In this paper, we propose a novel data partitioning approach for distributed association rule mining algorithms in the context of a grid computing environment. We conduct experiments using the French research grid "Grid'5000". Experimental results confirm that our data partitioning approach is very sufficient for balancing the load when homogeneous clusters are used. For heterogeneous clusters, the proposed data partitioning approach constitute the preprocessing phase of the process of dynamic load balancing of distributed association rule mining.

Keywords: *Grid platforms, distributed association rule mining, data partitioning, workload balancing.*

1 Introduction

The beginning of Grid Computing was in 1999 by the first exploration of David Gedye in using a large number of Internet-connected heterogeneous computers as a supercomputer for searching extraterrestrial intelligence. Grid computing, which is very different from parallel computing, offered an immense computational power and many great opportunities [12]. Recognized by the industry today, grid computing is gaining widespread adoption in various areas including data mining, customer relations, computational mechanics, biology and risk management in financial institutions. Nowadays, we are witnessing a trend of increasing presence of grid computing comparable to that of electricity [18]. Magoulès et al. [17] define a grid as a hardware and a software infrastructure that provides transparent, dependable, pervasive and consistent access to large-scale distributed resources owned and shared by multiple administrative organizations in order to deliver support for a wide range of applications with the desired qualities of service. These applications can perform high

throughput computing, on-demand computing, data intensive computing, or collaborative computing [12].

Data mining, or knowledge discovery in databases, aims at decreasing the gap between data and useful information. Association Rule Mining (ARM) is one of the most important data mining techniques [2, 3]. The most important challenge for this technique is quickly and correctly finding interesting correlation relationships between items in very large databases. As it is impossible to bring all the data to a centralized place due to technical and organizational reasons, the dataset to be mined is stored in a distributed heterogeneous data sources. Therefore ARM algorithms need high computing capabilities that could be provided by parallelism and distribution. Grid platforms can help in ameliorating the performances and reducing the execution time. They can be used to integrate the heterogeneous data sources into one virtual data set.

ARM algorithms are highly iterative and necessitates phases of synchronization between iterations to consolidate results. Computing nodes must reach a synchronization barrier almost at the same time, in order to benefit from parallelism. Consequently, the development of an efficient data partitioning approach is required to facilitate the mining of the distributed dataset. The proposed approach must take into consideration the specificities of both ARM technique and grid platforms. Furthermore, the heterogeneity of the grid components (i.e. nodes, clusters and sites) imposes different processing capabilities and therefore a data distribution which does not take into account this factor, may generate a remarkable skew between processing nodes in reaching a barrier of synchronization.

The work developed here is part of our contribution in workload balancing of distributed association rule mining using a grid paltform. We developed a hybrid load balancing strategy constituted of a static and a dynamic parts. The dynamic part of our strategy was published in [24], while in this paper we introduce the static part, where we propose a new data partitioning approach based on frequent itemsets supports. This part is performed before execution in order to minimize, as much as possible, the probability of load imbalance during execution. This is achieved by properly distributing, on the basis of tangible characteristics of the data to be processed, between different computing resources. The goal of our data partitioning approach is to have all processing nodes complete their job at the same time (in order to decrease a node idle time). This would help in eliminating the blocking statements during the treatment and thus considerably ameliorates the response time of the distributed ARM algorithm.

The rest of the paper is organized as follows: The state of the art is presented in section 2. The need of load balancing for grid-based association rule mining is discussed in section 3. In section 4, we propose our new data partitioning approach. The experimental evaluation and comparisons with related works are presented in section 5. Finally, the paper concludes with section 6.

2 State of the art

Association Rules Mining (ARM) technique was first introduced by Agrawal in the context of transactional databases [3]. It searches for interesting correlation relationships among a large set of data items. This technique was first used for market basket analysis. This process analyses customer buying habits by finding associations between different items that customers place in their "shopping baskets". Such information may be used to

plan marketing or advertising strategies, as well as catalog design. Each basket represents a different transaction in the transactional database, associated to this transaction the items bought by a customer [14].

Nowadays, ARM technique is widely applied to a diversity of domains, where ever there exist huge amounts of data that need to be analyzed under the goal of extracting useful correlations. This technique is used in the analysis of software problems of heterogeneous distributed systems, in the analysis of high dimension DNA or protein sequences, in engineering design and science exploration, etc [14, 15].

2.1 Basic concepts

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of m items. Let D, the task relevant data, be a set of database transactions where each transaction T is a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID. Let X be a set of items called ***itemset***. A transaction T is said to contain X if and only if $A \subseteq T$. An association rule is an implication of the form $X \Rightarrow B$, where $X \subseteq I$, $B \subseteq I$ and $AX \cap Y = \emptyset$. The rule $A \Rightarrow Y$ holds in the transaction set D with ***support s***, where s is the percentage of transactions in D that contain $X \cup Y$ (i.e. both X and Y). This is taken to be the probability, $P(X \cup Y)$. The rule $X \Rightarrow Y$ has ***confidence c*** in the transaction set D, if c is the percentage of transactions in D containing both X and Y at the same time. This is taken to be the conditional probability, $P(Y/X)$ [14].

2.2 Serial algorithms

Several sequential algorithms for solving the frequent set counting problem have been proposed in the literature. We can define two main methods for determining frequent itemsets supports: with candidate itemsets generation [3] and without candidate itemsets generation [9].

The Apriori algorithm [3] was the first effective algorithm proposed in the literature. This algorithm uses a generate-and-test approach which depends on generating candidate itemsets and testing if they are frequent. It uses an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets. During the initial pass over the database the support of all 1-itemsets is counted. Frequent 1-itemsets are used to generate all possible candidate 2-itemsets. Then the database is scanned again to obtain the number of occurrences of these candidates, and the frequent 2-itemsets are selected for the next iteration.

SEAR algorithm [19] is identical to Apriori, except that SEAR stores candidates in a prefix tree instead of a hash tree. In a prefix tree each edge is labeled by items, common prefixes are represented by tree branches and the unique suffixes are stored at the leaves.

DIC algorithm is a generalization of Apriori [6]. The database is divided into p equal-sized partitions so that each partition fits in memory. For partition 1, DIC gathers the supports of single items. Items found to be locally frequent (only in this partition) generate candidate $2 - itemsets$. Then, DIC reads partition 2 and obtains supports for all current candidate $1 - itemsets$ and candidate $2 - itemsets$. This process is repeated for the remaining partitions. DIC starts counting candidate $k - itemsets$ while processing partition k in the first database scan. After the last partition p has been processed, the processing wraps around to partition 1 again. A candidate itemset global support is known once

the processing wraps around the database and reaches the partition where it was first generated. DIC is effective in reducing the number of database scans if most partitions are homogeneous (have similar frequent itemset distributions). If data is not homogeneous, DIC might generate many false positives (itemsets that are locally frequent but not globally frequent) and scan the database more than Apriori does [7].

The DCI algorithm proposed by Orlando and others [20] is also based on candidate itemsets generation. It adopts a hybrid approach to compute itemsets supports, by exploiting a counting-based method (with a horizontal database layout) during its first iterations and an intersection-based technique (with a vertical database layout) when the pruned dataset can fit into the main memory.

The FP-growth algorithm [25] allows frequent itemsets discovery without candidate itemsets generation. First it builds from the transactional database a compact data structure called the FP-tree then extracts frequent itemsets directly from the FP-tree.

The analysis of large size datasets of nowadays scientific and commercial applications, residing on different geographical locations, necessitates the use of the high computational power of distributed and parallel systems [28]. Grid systems can play a significant role in providing an effective computational support for knowledge discovery applications.

2.3 Grid-based association rule mining

For grid-based ARM we have to deal with two important concepts: distribution and parallelism. The non-existence of a central memory and also the large volume of data to be analysed oblige us to distribute the dataset into portions and parallelism is needed to deal with these data portions.

Grid platforms provides interesting solutions to ARM algorithms, as these platforms can support the distribution of data by integrating the data sources into one virtual data set, and also help in ameliorating the response time by providing a high computational power.

Grid platforms provide protocols and services to share computing resources, data, and software managed by multi-institutional virtual organizations. Grid services are provided by middleware systems such as Globus, Legion, gLite, and Unicore. These middleware environments are meta-systems that run on the top of existing operating systems and networks [12]. However, services for data access and management provided by today's middleware are not sufficient to easily build Grid-based data mining applications. The development of Grid systems supporting data mining applications is needed to analyze and extract knowledge from huge amount of data stored within such environments [4].

A variety of systems, algorithms and projects that provide mechanisms for integration and deployment of classical ARM algorithms on grid were developed. The reader can refer to [4] for more details. In what follows, we will illustrate some of them in brief.

2.3.1 General purpose grid-based data mining systems

Several projects such as TeraGrid [9] and InfoGrid [13] focused on the creation of Grid infrastructure providing tools for management and manipulation of distributed data sources. Other projects GridMiner [5], WekaG [21], Weka4WS [23], DataMiningGrid [22] have focused on the creation of systems for distributed knowledge discovery over Grid infrastructures.

2.3.2 Algorithms ported to grid

Previously described grid-based data mining projects provide mechanisms for the integration and deployment of classical algorithms on grid platforms, but not new grid-specific algorithms. The focus of these projects was to run data mining tasks correctly but not efficiently and there were no performance or efficiency issues in the results available. In what follows we will mention some of the important attempts to port ARM algorithms to grids, where ARM specificities is taken into account.

- The DisDaMin project (Distributed Data Mining) [11] intends to tackle data mining tasks considering data mining specifics as well as grid computing specifics. For data mining problems, it is necessary to obtain an intelligent data partition, in order to compute more independent data fragments. The main problem is how to obtain this intelligent partition. First DisDaMin fragments the data using clustering methods then uses asynchronous collaborative techniques adapted to the specificities of Grid execution. The use of this fragmentation method makes it possible to carry out optimal local processing on each node, with a minimum amount of communications. Simulations were performed on the French national grid, named Grid'5000, showing the efficiency of this project.
- The Grid-based Distributed Max-Miner (GridDMM) [16] is an algorithm for mining maximal frequent itemsets from databases on a data-Grid system. It is designed to have low communication and synchronization overhead in order to deal with the grid environment. GridDMM implements a local mining phase and a global mining phase. During the local mining phase, each node scans the local database to discover the local maximal frequent itemsets. In the global mining phase they form a set of maximal candidate itemsets for a top-down search. The global candidate itemsets are stored on a prefix-tree. Counts are exchanged between neighbor nodes using the Grid-enabled Message Passing Interface (MPICH-G2). Experiments on GridDMM have been done using a cluster of workstations and the Globus Toolkit. They show that GridDMM algorithm has better performance than the CD algorithm [28].
- A method for integrating the Apriori algorithm in distributed databases with the Globus toolkit is proposed in [1].
- Yang et al. proposed a heuristic data distribution scheme for data mining applications on grid environments [26]. The grid is represented in a master/slave model. This model is represented by a star graph $G = (P0, P1, ..., Pn)$ where $P0$ is the master node and the other n nodes, $(P1, ..., Pn)$, are slave nodes. In addition, there is a virtual communication link Li connecting the master node and the slave node Pi. They proposed a performance based heuristic to solve the data partition problem for association rule mining applications.
- Yu et al. proposed a weighted distributed parallel Apriori algorithm [27] in which the transaction identifier of itemsets is stored in a table to compute their occurrence. This approach succeeded in reducing the communication cost overheads between processors. However, it suffers from a scalability problem due to the fact that there are only one master and several slaves (i.e. centralized approach).
- Fatta [10] proposed a distributed approach to the frequent subgraph mining problem to discover interesting patterns in molecular compounds. The general approach is to partition the problem into independent subtasks, in order to minimize communication

and synchronization among processors. In order to adopt a distributed approach in a large-scale computing environment, communication latency and node failures have to be tolerated. For this reason, we have adopted a partitioning strategy and discarded solutions based on a collaborative approach among processes.

- ISSDL-DM proposed by Zhou [29] uses a self-defined structure linked list to store item sequence, then to find the frequent item sets from large to small. The main idea of this structure design is to reduce finding time and to make insertion and deletion operations more efficient. Performances of this algorithm are tested only via simulations.

3 The need of load balancing for grid-based association rule mining

Association Rule Mining (ARM) algorithms exhibit an irregular computational structure where the workload is unpredictable and changes during execution. This causes a load imbalance when these algorithms run under parallel/distributed computing platforms. Furthermore, executing ARM algorithms under a grid computing environment will considerably increase the load imbalance due to the heterogeneity of the platform. This load imbalance is a major contributor to the inefficiency of parallel ARM algorithms, where:

- The parallel algorithm can only execute at the speed of the most heavily loaded processor.
- The increase in the number of computing nodes may slowdown the parallel algorithm (i.e., a scalability problem).

In order to prevent the load imbalance in parallel ARM algorithms we need:

1. An appropriate data partitioning approach:
    - ARM algorithms have an iterative nature. A phase of synchronization is needed by the end of each iteration in order to consolidate results (i.e. to calculate global supports). As a result, if the dataset is randomly partitioned between processing nodes, processors will not reach a synchronization barrier at the same time. This will increase processors idle time and will considerably increase the total execution time of the ARM algorithm.
    - Even by equally partitioning the dataset between processing nodes (i.e equal chunks), they may not reach the barrier of synchronization at the same time. This is due to the fact that, the amount of work needed for each database partition, depends on the degree of correlation between items in that partition. The degree of correlation is measured by the number of potential frequent itemsets (i.e. number of possible combinations).
2. A dynamic load balancing strategy, that adjust the load during execution. ARM algorithms have a dynamic nature because of their dependency on the degree of correlation between itemsets in the transactional database which cannot be predictable before execution. If ARM algorithms were to be executed in a homogeneous environment (SMM or classic DMM) then our proposed data partitioning approach would

be very sufficient for balancing the load. When ARM algorithms are executed under a grid computing environment, dynamic load balancing is needed, in addition to the partitioning approach, to deal with the heterogeneity of the environment.

In the coming section, we propose a new data partitioning approach based on the specificities of distributed ARM algorithms.

4 A novel data partitioning approach for distributed ARM algorithms

Data partitioning represents the preprocessing phase of the distributed execution of ARM algorithms. Our objective is to propose a new data partitioning approach which improves the conditions of the processing phase of the algorithm, under the goal of ameliorating the overall performance. This is done by:

- ***Eliminating*** the probability of load imbalance in the case where execution is done under a homogeneous environment (for example under a homogeneous cluster in the grid).
- ***Minimizing*** the probability of load imbalance, when the ARM algorithm runs under a heterogeneous environment.

In fact, both the amount and the nature of data play a determinant role in the generation of a potential overload within an entity (node, cluster and/or site) during processing. Also, the heterogeneity of the components of the grid imposes different processing capabilities and therefore a data distribution approach, which does not take into account this factor, is exposed to the risk of being penalized by the delays caused by a computing node idle time during the processing phase.

The preprocessing phase, as its name indicates, occurs before the actual start of data processing. Data is partitioned over a fixed number of sites, where an adequate partitioning, is the first step to successfully balance the load. It must minimize/eliminate the situations where one computing entity ends early while the other still has data to be processed.

Three variants have been studied for this phase:

4.1 Variant 1: Fair partitioning

This partitioning seems to be the most obvious and the most natural. It assigns equal sized portions to each site (i.e., dividing the global database by the number of partitions to generate). This approach is only concerned by the partition size without paying attention to the internal nature (i.e., amount of work needed) of this portion.

This approach allows sites to receive the same data amount to process, but does not guarantee an equal workload. Indeed two equal-sized partitions do not take the same processing time due to the differences in the degree of correlation between itemsets and their frequencies between different partitions. Therefore, even with homogeneous computational resources, having equal size data partitions do not guarantee the same workload.

As demonstrated by Figure 1, this partitioning approach can lead to unnecessary, waiting times, by the end of each iteration of the ARM algorithm. It constitutes one of the first sources of load imbalance that occurs during the execution of the ARM algorithm.

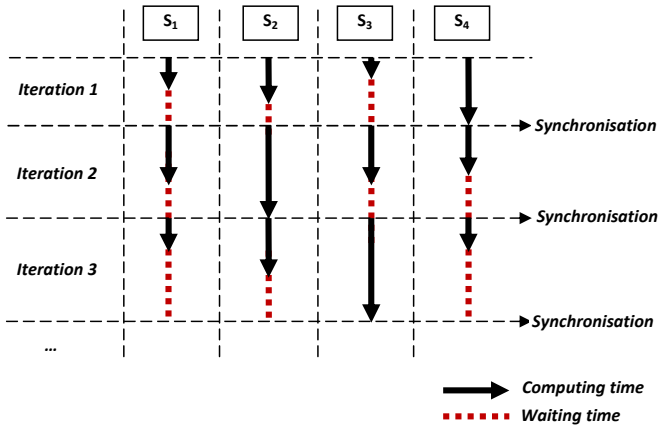

Figure 1. Waiting time caused by "fair partitioning"

4.2 Variant 2: Homogeneous-contents-based partitioning

The second variant is a direct optimization of the first approach, by trying to not only partition the dataset into equal-sized portions but also containing homogeneous transactions, in the sense that they produce the same workload during processing.

The principle is to consider the two determinant factors, namely:

- Equal size partitions.
- Homogeneous partitions contents. The homogeneity factor is in term of the amount of work to be done.

As explained above, the partitioning of the database into equal-sized partitions is not complicated. The hard question is how to have partitions with equal amount of work, knowing that the degree of correlation between itemsets within a partition could not be known before execution.

The rate of presence $Pr(i_l)$ helps us in tracing the curve of items presence and in generating a number of classes that we call "item class" $(Cls(i_l))$. These classes reflect the degree of importance of an item in the database D. The division into classes is done on a scale from 0% to 100% and is in the form of percentage ranges (example: $[0\%, 20\%]$; $[20\%, 40\%]$;...; $[80\%, 100\%]$). We illustrates in Figure 2, a simple example for the generation of three classes: $C_1 = "LowFreq"$, $C_2 = "MediumFreq"$ and $C_3 = "HighFreq"$, each class is based on a percentage range.

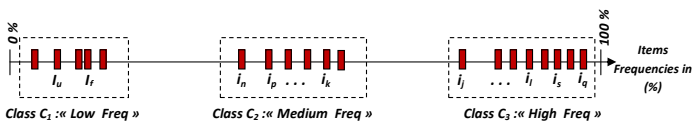

Figure 2. Items repartitioning and classes generation

From the classes of items we can generate classes for transactions. In effect, and as shown in Figure 3, by replacing each item i_l, within a transaction Tr_i, by its class $Cls(i_l)$

and then adding (i.e., summation operation) identical classes to count their total. We can obtain an image of the composition of the transaction and its tendency with respect to the generated classes.

From here, the class of the transaction will be the $Cls(i_l)$ (item class) the most present in it.

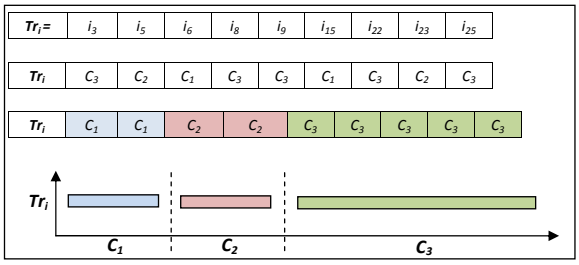

Figure 3. Tendency of a transaction Tr_i with respect to a class

By applying the same principle to the whole database D, classes generation process will be done according to the following steps:

1. From the database D, the frequency of items (i.e., $1 - itemsets$) is calculated and their presence rate in D is deducted.
2. Based on the previously calculated presence rate of items and their distribution, we can generate the classes of items and assign each item to its proper class (see Figure 4).

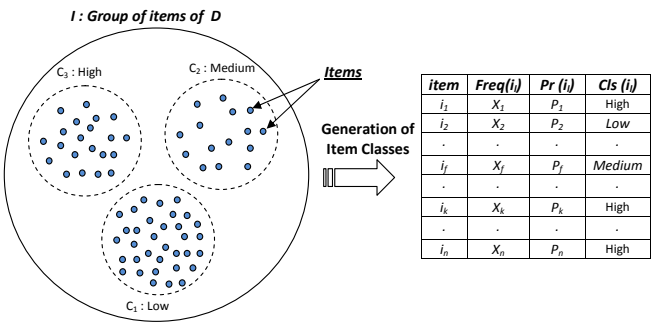

Figure 4. Generation of Frequencies and rates of presence of items

3. From the classes of items, and as already explained in Figure 3, we build the class of each transaction. The principle of generating the class of trancsaction Tr_i is applied to all transactions of D: by removing, from every Tr_i, all classes $TC\_Tr(Tr_i, X)$ with a size less then or equal to the maximum class size. By this we obtain the classes of different transactions of D as illustrated in Figure 5.

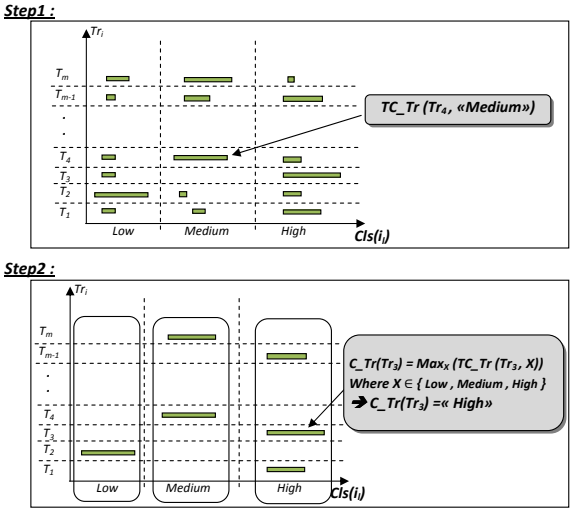

Figure 5. Transaction class generation

Once transactions have been assigned to their classes, we move to the process of partitioning which consists of distributing, in an equitable manner, each class of transactions over the number of database partitions to be created.

Partitions generation is done, as shown in Figure 6, by dividing each class by the number of database portions we want to create.

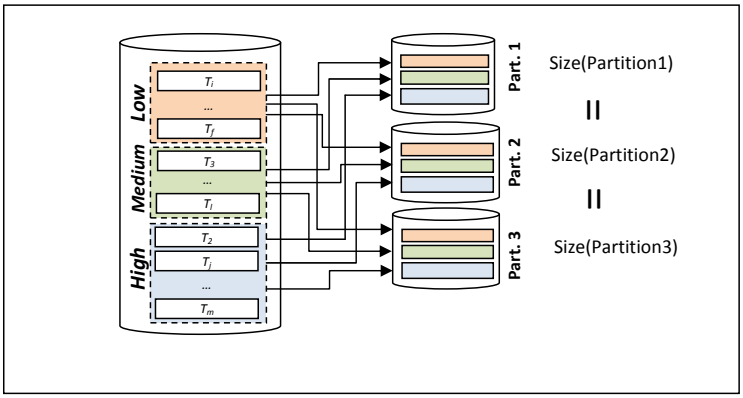

Figure 6. Partitions generation from transactions classes

We obtain at the end, database partitions with the same size and having also a homogeneous content (i.e., the same degree of correlation between items).

4.3 Variant 3: Contents-capacity-based partitioning

This third variant is proposed to improve the second one. This is done by considering the heterogeneity of the entities implied in processing.

Indeed, giving the same database size to each site, even with a fair amount of work, is sufficient in a dedicated homogeneous environment. For grid computing, the differences

in storage and processing capacities between computing nodes is a determinant factor in database distribution.

The capacity of a site is characterized by three criteria:

- $\omega c(S_i)$: processing capacity of site S_i.
- $\omega s(S_i)$: storage capacity of site S_i.
- $\omega t(S_i)$: communication cost inside the site S_i.

Table 1. Notations used

$D = \{Tr_1, Tr_2, ..., Tr_m\}$	A transactional database constituted of m transactions
$Tr = \{i_1, i_2, ..., i_n\}$	A transaction composed of n items
Nb_Site	Number of sites contributing in execution
$Avg\_\omega c$	Average processing capacity
$Avg\_\omega s$	Average storage capacity
$Avg\_\omega t$	Average communication cost
λ, β, δ	Multiplying factors: constants that defer from one grid to another. For Grid'5000, $\lambda = 2.5$, $\beta = 0.5$, $\delta = 1$
Q_i	Capacity of site S_i
$Part_i$	Partition i of database D
$Quota(Part_i)$	Quota of the database to be assigned to each site
$Nb\_Cls = \|C\|$	Number of generated classes
$TC(X) = \|X\|$	Number of transactions contained within the class X
$TC\_Tr(Tr_i, X)$	Size of the class X in the transaction Tr_i
$C\_Tr(Tr_i) = Max_x(TC\_Tr(Tr_i, X))$ where $X \epsilon C$	Transaction class: The maximum of classes size contained in the transaction Tr_i
Nb_Part	Number of D partitions to generate. Generally it is the number of sites contributing in the execution of the ARM algorithm.

Where $Avg\_\omega c$, $Avg\_\omega s$ and $Avg\_\omega t$ are calculated by the use of equations 1, 2 and 3 respectively.

$$Avg\_\omega c = \frac{\sum_{i=1}^{Nb\_Site} \omega c(S_i)}{Nb\_Site} \tag{1}$$

$$Avg\_\omega s = \frac{\sum_{i=1}^{Nb\_Site} \omega s(S_i)}{Nb\_Site} \tag{2}$$

$$Avg\_\omega t = \frac{\sum_{i=1}^{Nb\_Site} \omega t(S_i)}{Nb\_Site} \tag{3}$$

Therefore, the capacity of a site can be calculated by the following equation:

$$Q_i = (\lambda \times \frac{\omega c(S_i)}{Avg\_\omega c}) + (\beta \times \frac{\omega s(S_i)}{Avg\_\omega s}) + (\delta \times \frac{\omega t(S_i)}{Avg\_\omega t}) \tag{4}$$

Once the classes of the different transactions are generated, the distribution of each class is based on the quota allocated to each partition. This quota is calculated by the the following equation:

$$Quota(Part_i) = \frac{Q_i}{\sum_{i=1}^{Nb\_Site} Q_i} \tag{5}$$

Therefore, the partition $Part_i$ of the database D is calculated as follows:

$$Part_i = \bigcup_{i=1}^{Nb\_Cls} \{Tr_1, ..., Tr_p\} \tag{6}$$

$$where \quad p = TC(X) \times Quota(Part_i)$$

The algorithm of the third variant is displayed in what follows.

Algorithm 1 Contents-capacity-based partitioning Algorithm

Begin

- Calculate the frequencies of items in D
- Analyze frequencies:
  - Generate the number of classes of items to create
  - Affect items to classes
- Generate the classes of transactions:
  For All $Tr_i \epsilon D$
  - Replace i with $Class(i)$
  - Count $TC\_Tr(Tr_i, X)$
  - Generate $C\_Tr(Tr_i) = Max(TC\_Tr(Tr_i, X))$

  End For
- Count $TC(X)$
- **For** $i = 1$ **To** Nb_Part
  Create_Partition($Part_i$)
  End For
- **For** $i = 1$ **To** Nb_Part
  $Ouota(Part_i) = Count\_Quotas(S_i, \omega c(S_i), \omega s(S_i), \omega t(S_i))$
  End For
- **For All** $X \epsilon C$
  - $k = 1$
  - **For** $i = 1$ **To** Nb_Part
    - **For** $j = k$ **To** $(k + (TC(X)/Nb\_Part) - 1)$
      Add($Tr_j, Part_i$)
      End For

    End For
  - $k = (k + p)$

  End For

End

The following example will help to better understand the contents-capacity-based partitioning variant.

Example: Suppose that we have a transactional database D that we want to partition into four partitions (P_1, P_2, P_3, P_4) among four sites (S_1, S_2, S_3, S_4).
The database D contains 2000 transactions and 600 items.

1. Categories generation:
   - Calculating the frequencies of items.
   - Partitioning items into three categories (A, B and C) based on their frequencies

values. A for *High* frequencies, B for *Medium* frequencies and C for *Low* frequencies.

- The number of categories to generate is fixed based on the standard deviation of frequency values (in order to have a measure of how spread out values are).

2. Partitions generation:
   - The class of each transaction is generated on the basis of the categories of its items.
   - The generation of the class of each transaction allows the distribution of all transactions among the categories (or classes) A, B and C.
   - From the state vectors of different sites, we can calculate the capacity of each site and generate the adequate quota of the data to be assigned to it.
   - Instead of distributing classes A, B and C in a fair manner between the four sites, distribution must be based on the following weighting:
     - Partition P_1 of the site S_1 takes 30% of each class.
     - Partition P_2 of the site S_1 takes 20% of each class.
     - Partition P_3 of the site S_1 takes 35% of each class.
     - Partition P_4 of the site S_1 takes 15% of each class.

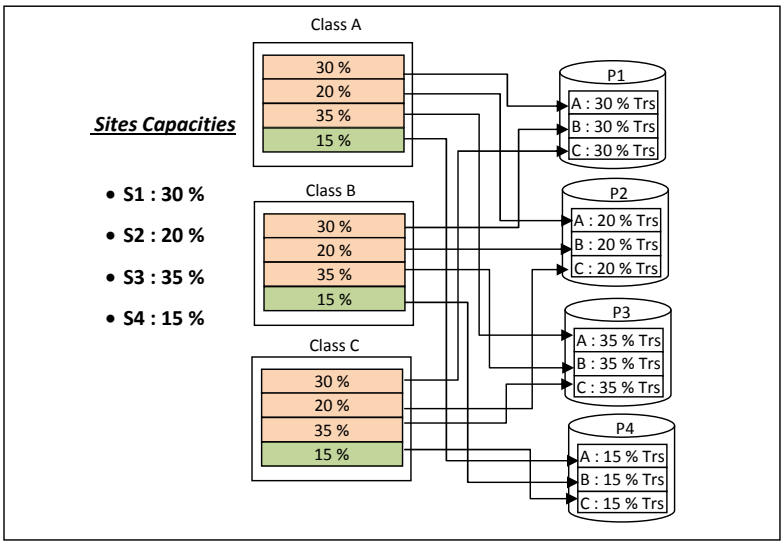

Figure 7. Partitions generation from classes of transactions

We obtain by the end, as mentioned in Figure 7, four partitions that take into account both the capacity of each site and the homogeneity of contents.

The contents-capacity-based partitioning variant the idle time of computing nodes during first iterations, and thus reduce considerably the number of load balancing interventions needed during the processing phase (i.e., at execution time).

5 Experimental evaluation

Tests are conducted using the French research grid, called Grid'5000 [8], 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).

The proposed data partitioning approach was the subject of specific experimentations in order to verify if this approach satisfies two criteria:

1. A first criterion related to the reduction of both communication and idle time induced by applying this partitioning approach approach. This will contribute to a rational exploitation of the available computational resources.
2. A second criterion related to the reduction of the number of calls to the load balancing process during the processing phase. Indeed, as already announced, the objective of the first phase (i.e., the preprocessing phase) is to improve the execution environment of the second phase (i.e., the processing phase) by reducing the load imbalances that occur during execution.

Since the grid execution environment is dynamic and the re-execution of the same algorithm on the same database several times do not necessarily give the same execution time (due to the dynamicity of ressources), experiments were made, for each database, several times and the average of obtained results was extracted.

Table 2 states the characteristics of the dataset used for tests. In what follows we illustrate the results obtained from running the three data partitioning variants. The **minimum support** value is equal to 0.75%

Table 2. Transactional database characteristics

Database	Characteristics			
	Items Number	Avg. Trans. Length	Transactions Number	Database Size
DB400T13M	6500	40	5400000	400 Mb

Experiments show that our data partitioning approach, proposed in variant 3, succeeded in balancing the load, without the need of run-time load balancing, when execution is done under a homogeneous cluster in the grid.

When several sites of the grid are used, our approach can help in eliminating the need of run-time load balancing during first iterations of the ARM algorithm.

5.1 Testing variant 1: Fair partitioning

It is important to mention that computing nodes used for following tests have different characteristics. The idle time of computing nodes is huge (see Figure 8). This is due to two reasons:

1. Equal data partitions, do not mean that the amount of work is equal in all partitions.
2. The heterogeneity of computing nodes.

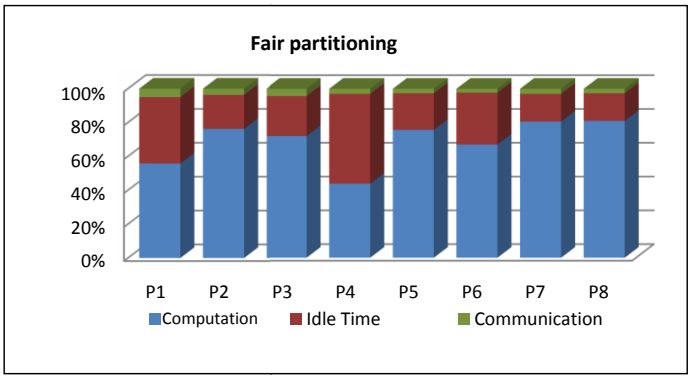

Figure 8. Evaluation of variant 1

Therefore, the first variant does not limit the probability of load imbalance. This is due to the fact that it does not take into account the difference in the correlation between itemsets in different partitions.

5.2 Testing variant 2: Homogeneous-contents-based partitioning

By the use of the second variant, an improvement was noted in the total amount of idle time (see Figure 9). Processors still have to wait behind barriers, and this is due to the differences in characteristics between computing nodes used for execution. This means that giving computing nodes the same amount of work, while they do not have the same computing power, is not sufficient.

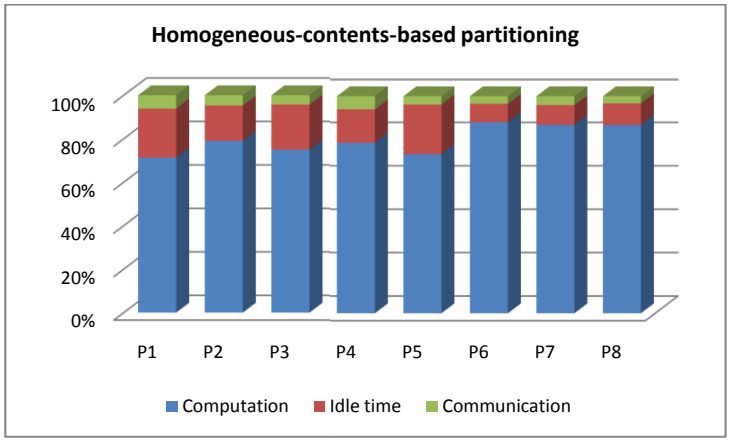

Figure 9. Evaluation of variant 2

5.3 Testing variant 3: Contents-capacity-based partitioning

A significant reduction of the idle time was noted when both the homogeneity of contents and the computing power of computing nodes are taken into consideration (see Figure 10).

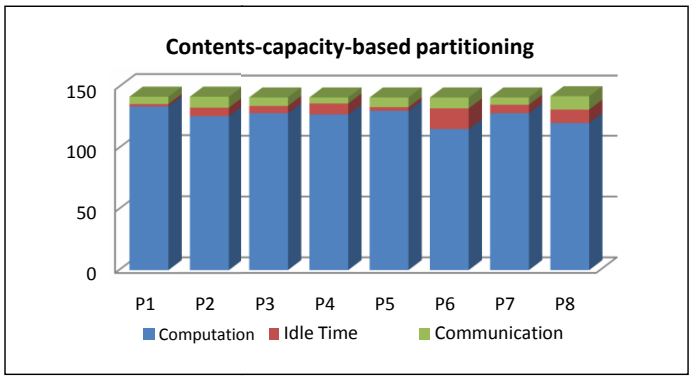

Figure 10. Evaluation of variant 3

Figure 11 shows that applying our data partitioning approach on the same database and using the same number of computing nodes, but with the variation of the number of sites, helps considerably in reducing the number of calls of the dynamic load balancing process. As immediate consequence, the response time is reduced. This improvement in performances is due to:

- The reduction of the total idle time.
- The reduction in the run-time load balancing overhead.

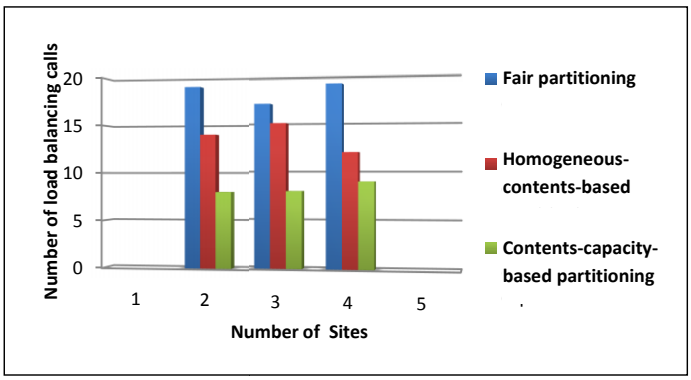

Figure 11. Comparing the three variants

5.4 Related works

Up to our knowledge, there exist only few works that provides data partitioning approaches which takes into account the specificities of both ARM algorithms and grid platforms.

The DisDaMin project (Distributed Data Mining) [11] fragments the data using a clustering method to generate totally independent data partitions and eliminates synchronization between the iterations of the ARM algorithm. While eliminating all phases of synchronization helps considerably in ameliorating the response time, we loose the exactitude of results. It is impossible to have totally independent classes in a transactional database. So, by this method we gain time but we sacrifice part of the result, where it is impossible

to generate all maximal frequent itemsets without consolidating results by the end of each iteration.

Yang et al. induced load balancing through a heuristic data partition technique that aims to reduce the total execution time of the program [26]. Yu et al. proposed a weighted distributed parallel Apriori algorithm [27] in which the transaction identifier of itemsets is stored in a table to compute their occurrence. The algorithm takes the factor of itemset counts into consideration in order to balance workloads among processors and reduce processor idle time. Both works [26, 27] do not support execution on large number of nodes. They are based on a centralized (master/slave) load balancing approach where there is one master responsible of data distribution and n computing slaves. This would cause a scalability problem as the number of computing nodes increases. Furthermore, they have only been studied on very small datasets. Contrariwise, our approach is scalable where it is totally distributed in order to respond to the high level of distribution in grid systems. In our tests, we used large datasets.

6 Conclusion

In this paper, we considered the problem of load imbalance caused from running ARM algorithms under a grid computing environment. We presented the preprocessing phase of our load balancing strategy. In this phase, the dataset is distributed between processing grid entities. We proposed a novel data partitioning approach that allows to predict the amount of work needed for each database partition. This approach can successfully maintain the load balance when the ARM algorithm is executed under a homogeneous cluster in the grid, and reduce considerably the number of load balancing interventions needed during the processing phase when execution is done under a heterogeneous grid platform.

As perspective of our work, we intend to experiment our data partitioning approach using real datasets.

ACKNOWLEDGMENTS

- We wish to thank Professor Margaret H. Dunham and Dr. Michael Hahsler for providing us with large datasets for tests.
- Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER, CPER Nord-Pas-de-Calais/FEDER Campus Intelligence Ambiante, several Universities as well as other funding bodies (see https://www.grid5000.fr).

References

[1] C. Aflori and M. Craus. Grid implementation of the apriori algorithm. In *Journal of Advanced Engineering Software*, volume 38, 2007.

[2] R. Agrawal and J.C. Shafer. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Eng.*, 8(6):962–969, dec 1996.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining associations rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 478–499, September 1994.

[4] M. BenHajHmida and A. Congiusta. Parallel, distributed, and grid-based data mining:algorithms, systems, and applications. *Handbook of Research on Computational Grid, IGI Global*, pages 90–119, May 2009.

[5] P. Brezany, J. Hofer, A. M. Tjoa, and A. Wohrer. Gridminer: An infrastructure for data mining on computational grids. In *Proceedings of the APAC Conference and Exhibition on Advanced Computing, Grid Applications and eResearch*, Queensland, Australia, October 2003.

[6] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 255–264, Arizona, USA, May 1997.

[7] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *In Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 255–264, USA, 1997.

[8] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Vicat-Blanc Primet, E. Jeannot, S. Lanteriand J. Leduc, N. Melab, G. Mornet, B. Quetier, and O. Richard. Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 99–106, Washington, USA, November 2005.

[9] C. Catlett et al. Teragrid: Analysis of organization, system architecture, and middle parallel, distributed, and grid-based data mining ware enabling new types of applications. In *Advances in Parallel Computing: High Performance Computing and Grids in Action*, volume 16, pages 225–249, 2008.

[10] G. D. Fatta and M. R. Berthold. Dynamic load balancing for the distributed mining of molecular structures. In *In IEEE Transactions on Parallel and Distributed Systems*, volume 17(8), pages 773–785, August 2006.

[11] V. Fiolet, R. Olejnik, G. Lefait, and B. Toursel. Optimal grid exploitation algorithms for data mining. In *Proceedings of the 5th IEEE International Symposium on Parallel and Distributed Computing*, pages 246–252, New York, United States, 2006.

[12] I. Foster and C. Kesselman. *The Grid2: Blue print for a New Computing Infrastructure.* Morgan Kaufmann, 2003.

[13] N. Giannadakis, A. Rowe, M. Ghanem, and Y. Guo. Infogrid: providing information integration for knowledge discovery. *Journal of Information Sciences*, 155:199–226, October 2003.

[14] J. Han and M. Kamber. *Data mining: concepts and techniques.* Maurgan Kaufman Publishers, 2000.

[15] S. Kotsiantis and D. Kanellopoulos. Association rules mining: A recent overview. In *GESTS International Transactions on Computer Science and Engineering*, volume 32(1), pages 71–82, 2006.

[16] C. Luo, A. L. Pereira, and S. M. Chung. Distributed mining of maximal frequent itemsets on a data grid system. In *Journal of Supercomputing*, volume 37, pages 71–90, 2006.

[17] F. Magoulès, T-M-H. Nguyen, and L. Yu. *Grid Ressource Management: Toward Virtual and Services Compliant Grid Computing.* CHAPMAN and HALL/CRC Press, 2009.

[18] F. Magoulès, J. Pan, K-A. Tan, and A. kumar. *Introduction to Grid Computing.* CHAPMAN and HALL/CRC Press, 2009.

[19] A. Mueller. Fast sequential and parallel algorithms for association rule mining: A comparison. In *Technical Report CS-TR-3515, University of Maryland*, 1995.

[20] S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. A scalable multi-strategy algorithm for counting frequent sets. In *In Proceedings of the 5th SLAM Workshop on High Performance Distributed Data Mining*, pages 398–416, Washington, USA, 2002.

[21] M.S. Perez, A. Sanchez, V. Robles, P. Herrero, and J. Pena. Adapting the weka data mining toolkit to a grid based environment. In *Lecture Notes in Computer Science*, volume 3528, pages 492–497, January 2005.

[22] V. Stankovski, M. Swain, V. Kravtsov, T. Niessen, D. Wegener, J. Kindermann, and W. Dubitzky. Grid-enabling data mining applications with datamininggrid: An architectural perspective. In *Journal Future Generation Computer Systems archive*, volume 24, April 2008.

[23] D. Talia, P. Trunfio, and O. Verta. Weka4ws: a wsrf-enabled weka toolkit for distributed data mining on grids. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 3721, pages 309–320, Porto, Portugal, October 2005.

[24] Raja Tlili and Yahya Slimani. A hierarchical dynamic load balancing strategy for distributed data mining. *International Journal of Advanced Science and Technology*, 2012.

[25] K. Wang, L. Tang, J. Han, and J. Liu. Top down fp-growth for association rule mining. In *In Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'02)*, pages 334–340, Taipei, Taiwan, May 2002.

[26] C. Yang, W. Shih, and S. Tseng. A heuristic data distribution scheme for data mining applications on grid environments. In *IEEE World Congress on Computational Intelligence (Fuzzy Systems'08)*, pages 2398–2404, June 2008.

[27] K. M. Yu and J. L. Zhou. A weighted load-balancing parallel apriori algorithm for association rule mining. In *In Proceedings of the IEEE International Conference on Granular Computing (GrC'08)*, pages 756–761, 2008.

[28] M.J. Zaki. Parallel and distributed association mining : A survey. *Concur-rency, Special Issue on Parallel Mechanisms for Data Mining*, 7(4):14–25, December 1999.

[29] L. Zhou, Z. Zhang, and M. Xu. Massive data mining based on item sequence set grid space. In *In Proceedings of the 2nd International Asia Conference on Informatics in Control, Automation and Robotics*, pages 208–211, March 2010.

Authors

Raja Tlili is an assistant at the University of Carthage (Tunisia). She received her B.Sc. degree from the United Arab Emirates University (UAE) and Master degree in computer science from the Faculty of Sciences (University of Tunis-Elmanar/Tunisia) in 1999 and 2003, respectively. Currently, she is pursuing a PhD in distributed data mining.

Yahya Slimani studied at the Computer Science Institute of Alger's (Algeria) from 1968 to 1973. He received the B.Sc.(Eng.), Dr Eng and Ph.D degrees from the Computer Science Institute of Alger's (Algeria), University of Lille (France) and University of Oran (Algeria), in 1973, 1986 and 1993, respectively. He is currently Full Professor at the Department of Computer Science of Faculty of Sciences of Tunis. His research activities concern Datamining, parallelism, distributed systems and Grid Computing. Dr. Yahya Slimani has published more than 100 papers from 1986 to 2009. He contributed to Parallel and Distributed Computing Handbook, Mc Graw-Hill, 1996. He is currently Scientific Expert for the European Union. He joined the Editorial Boards of the Information International Journal in 2000, the J.UCS Journal and others journals.