Parallel Data Mining Experimentation Using Flexible Configurations

José M. Peña[1]*, F. Javier Crespo[2], Ernestina Menasalvas[1], and Victor Robles[1]

[1] Universidad Politécnica de Madrid, Madrid, Spain
[2] Universidad Carlos III de Madrid, Madrid, Spain

Abstract. When data mining first appeared, several disciplines related to data analysis, like statistics or artificial intelligence were combined toward a new topic: extracting significant patterns from data. The original data sources were small datasets and, therefore, traditional machine learning techniques were the most common tools for this tasks. As the volume of data grows these traditional methods were reviewed and extended with the knowledge from experts working on the field of data management and databases. Today problems are even bigger than before and, once again, a new discipline allows the researchers to scale up to these data. This new discipline is distributed and parallel processing. In order to use parallel processing techniques, specific factors about the mining algorithms and the data should be considered. Nowadays, there are several new parallel algorithms, that in most of the cases are extensions of a traditional centralized algorithm. Many of these algorithms have common core parts and only differ on distribution schema, parallel coordination or load/task balancing methods. We call these groups *algorithm families*. On this paper we introduce a methodology to implement *algorithm families*. This methodology is founded on the MOIRAE distributed control architecture. In this work we will show how this architecture allows researchers to design parallel processing components that can change, dynamically, their behavior according to some control policies.

1 Introduction

Distributed data mining (DDM) deals with retrieval, analysis and further usage of data in a non-centralized scenario. The way this distribution is performed depends on two different factors: (i) Whether the data are not stored in a single site and the data sources are disperse and (ii) Whether data analysis process requires high performance computation techniques (like parallel processing). If any of these two factors is present traditional data mining has deep problems to achieve the expected results. Collective data mining [4] deals with distributed data source mining and parallel data mining tackles the second problem.
Modern data mining tools are complex systems in which extensiveness (new

* This research project is funded under the Universidad Politécnica de Madrid grant program

algorithms) and performance (mining time) are key factors. If new problems require these systems to be distributed, their complexity will also increase. The new parallel algorithms are also more complex than the original version they come from. Parameters like, network bandwidth, memory and processor usage or data access schemas are considered when these algorithms are designed. Our proposal presents a new method to design algorithms. Using this method, core algorithm components are developed once, as efficient and compact elements, and then they are combined and configured in many different ways. This method not only allows the researchers to combine algorithm components in different schemas, but also provides the mechanisms to change component behavior and performance dynamically.
On this contribution we propose MOIRAE architecture as a tool to implement flexible algorithms in a distributed computation environment.

2 Distributed Data Mining Systems and Algorithms

During the last two years, the first distributed and parallel data mining systems were developed. These systems can be divided into two different groups: (i) **Specific parallel/distributed algorithm implementations**, like, JAM [13], PADMA [5], BODHI [4] or Papyrus [2]. These systems implement only one algorithm or a very restricted set of algorithms/techniques. The system architecture has been designed to deal with this specific technique and algorithm. (ii) **General purpose systems**, for example Kensington [6], PaDDMAS [11] or DMTools [1]. Provost [10] analyzed two main parallel and distributed schemas (fine-grain and coarse-grain) and their application to distributed data mining problems. Krishnaswamy defined cost models for distributed data mining [7]. Joshi et al. [3] provided an overview of different parallel algorithms for both association and classification rules. Zaki also provides a very interesting analysis in [14].

2.1 Flexible Optimization Environment

Much effort has been addressed on the development of efficient distributed algorithms for data mining, nevertheless this is not the only way to outperform existing data mining solutions. Although the specific distributed implementation of a rule extraction algorithm plays an important role, distribution schema, task scheduling and resource management are also key factors.
We propose the capability to plug-in and update control strategies and decisions during the system run-time. As a consequence, we might have two or more sets of system configurations and behavior decisions. If the system state changes or if a different functional operation is expected the appropriate strategies is applied to the system. Based on this idea a new architecture is proposed as the foundation of the elementary parts of a system with these features. To develop this idea the first stage is to present how the M/P paradigm is used.
Our contribution presents how MOIRAE generic architecture [8,9] can be used to

implement *algorithm families*. These families are groups of algorithms based on the same core process. Many algorithm variants share the same basic operations and they only differ in terms of small features.

3 MOIRAE Architecture

MOIRAE architecture is a generic architecture that uses the Mechanism/Policy (M/P) paradigm to design flexible systems. System tasks are divided into two different levels:

- ❐ Operational level: Features provided by the system. This level contains all the actions (operations) performed by the system as well as the functions used to monitor its status and the environment.
- ❐ Control level: Decisions that rule the system. This second level defines the control issues applicable to the operational level functions. These rules describe when and how the operations are performed.

This M/P paradigm is quite common under design phases of complex applications like operating systems (OS). MOIRAE architecture is one step ahead. This architecture presents a run-time engine to manage control level decisions while the system is running. This technique allows the user not only to change control policies during execution time during the system design/implementation phases. The option to configure complex systems without any re-design has two main profits: (i) Research on new algorithms and techniques can easily test many different variations of the algorithm only defining new control policies, (ii) Complete systems may be configured for specific installations and user requirements customizing their control policies. As a drawback this control schema may include additional overhead because of the separation of responsibilities into control and operational elements. The effect of both benefits and drawbacks will be studied in the next sections.

3.1 MOIRAE Components

The design of the MOIRAE architecture is founded on the concept of *component*, as each of the software pieces of the distributed system. According to the division of operational and control tasks, each component is also divided in two planes:

- ❐ **Operational Plane**: That provides all the functions required to achieve the work performed by the component. The functions, their design/implementation and the exact elements included in the plane depend on the task performed by the component. The responsibilities of this plane heavily depend on the functions of the component.
- ❐ **Control Plane**: This plane interacts either with the control plane from other components or with the operational part of their own component. The control plane solves complex situations called *conflicts*, sending orders to the operational plane. Inside of this component there exists *Policy Engine*, that

manages the decisions taken by the control plane. When the control plane is summoned the *policy engine* is activated. Inside of this element, there is a *Policy Database* or p-DB. This database stores all the information applicable to conflict solving and all possible control actions. The *Policy Engine* queries p-DB to decide the control plans.

3.2 MOIRAE Architecture Model

The architecture model describes how multiple components are combined and interconnected to deal with the tasks performed by any specific system. The network of interconnected components is called *interaction graph*. This *graph* defines the components and their relationships. For any system, there are two different *graphs*: *operational graph* and *control graph*. The *operational graph* shows the relations used by the operational planes of the components. This *graph* depends on the task performed by the components and the services provided by the system. MOIRAE does not define any specification on this *graph*. *Control graphs* represent the control relationships among the components. This Model provides a generic schema of this *graph*. Control interactions are based on a hierarchical organization. This organization describes two types of components: (i) Control-oriented components and (ii) Operational-oriented components. The first group has few operational functions and the most important part of the component is its control part. Operational-oriented components provide important operational functions and have very basic control features. The hierarchical structure places operational-oriented components at the lower levels of the *graph*. These components performs simple control tasks. Global control decisions and complex configuration/optimization issues are managed in the upper levels by the control-oriented components. *Control* and *operational graphs* are completely independent. A component can have relations with different components on the control and on the operational plane.

3.3 MOIRAE Control Model

MOIRAE control model shows how control decisions are taken either locally or as a contribution of different control planes. When the control plane is activated, for example when a conflict is detected by the *event sensor*, the *policy engine* evaluates the alternatives to solve the problem. As a result, the control plane returns a sequence of actions to be performed to solve the conflict. For complex problems the control plane would be unable to achieve an appropriate solution by itself. Control Model specifies three different control actions that rule the cooperative solution of complex problems:

❒ **Control Propagation**: When a control plane is unable to solve a problem it submits the problem description (e.g.: the conflict) and any additional information to the control plane immediately superior in the hierarchy.
❒ **Control Delegation**: After receiving a control propagation from a lower element, the upper element may take three different alternatives:

① If it is also unable to solve the problem it propagates up the conflict as well.
② If it can solve the problem, it may reply to the original component with the sequence of actions necessary to solve the problem. This original component executes these actions.
③ In the last situation it is also possible that the component, instead of replying with the sequence of actions the component may provide the p-DB information necessary to solve the problem in the lower component. This information could be used also in any future situation. This alternative is called *Control Delegation.*

❒ **Control Revoke**: This action is the opposite to the *control delegation* one. Using this control action any upper component in the hierarchy may delete information from the p-DB of any lower element. This action may be executed anytime and not only as a response of a *control propagation.*

4 Association Rules: Distributed Generalized Calculation

Association rule extraction describes patterns present in OLTP databases. A variant of this problem is what is called generalized associations. These association patterns is related to a conceptual hierarchy. This hierarchy generalizes the possible values of a transaction from less to more generic concepts in a tree graph.
Shintani and Kitsuregawa [12] define different algorithms to extract generalized associations from a transaction database using a cluster of workstations. The basic steps in any association calculation are: (i) Candidate itemsets generation, (ii) Counting and (iii) Large itemsets selection. For real world data many possible items may appear in the transaction base. This makes it impossible to handle candidate itemsets in memory. Many distributed association strategies follow an approach based on itemset space partitioning. The authors describe five different association algorithms to deal with partitioned itemsets:

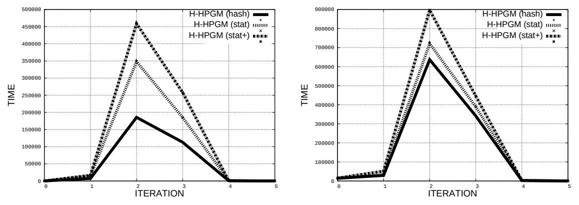
① **H–HPGM (hash)** (Hierarchical Hash Partitioned Generalized association rule Mining).
② **H–HPGM (stat)** (H–HPGM with stats).
③ **H–HPGM–TGD (stat+)** (H–HPGM with Tree Grain Duplication).
④ **H–HPGM–PGD (stat+)** (H–HPGM with Path Grain Duplication).
⑤ **H–HPGM–FGD (stat+)** (H–HPGM with Fine Grain Duplication).

Shintani and Kitsuregawa's paper [12] has a complete description of these algorithms and their comparative performance.

4.1 Algorithm Family Implementation

In our contribution we focus at how the five variants of the same algorithm (and future versions) could be implemented in the component schema mentioned above. The M/P paradigm could be used at many different points of this problem, but itemset partitioning is the key problem of these algorithms. The partitioning strategy is defined by the coordination node that gathers all the itemsets at the end of each iteration. This node computes the most frequent itemsets and then it distributes the next iteration candidates among the computational nodes. The nodes also have the necessary information to locate the appropriate node when a non-local itemset is counted.

- ❐ *Data Partitioning*: When candidate itemsets had to be distributed a control decision should be taken. The control plane defines a partitioning schema depending on which algorithm is selected.
- ❐ *Itemset Counting*: When the process component scans every transaction it updates local itemset counters if they are present in the transaction. If an itemset is not present a conflict event happened. This event is handled by the control plane to select which component should be notified for this itemset. The mechanism used to solve this conflict is previously unknown by the component and when the first conflict arises a control propagation–delegation is performed. At the beginning of each iteration this mapping information is deleted (control revoke) because a different partitioning schema may be selected.

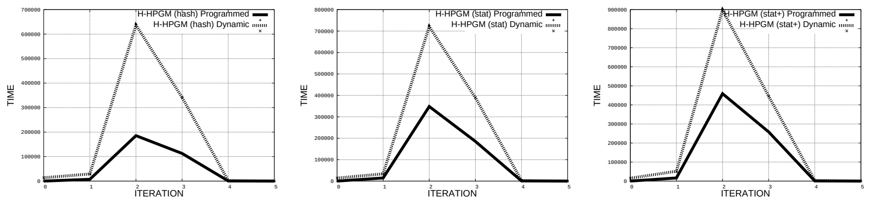


(a) Programmed (b) Dynamic Policies

Fig. 1. Candidate distribution time

4.2 Experimental Results

The plots 1.a and 1.b show candidate distribution time spent by the coordination node. This measure represents only decision time, not the time necessary for candidate sets re-transmission. In both cases, standard programmed solution and dynamic policy solution kept a similar ratio. If the different algorithm implementations (programmed vs. dynamic) are compared, different performance rations can be observed. **H–HPGM (hash)** implementation (figure 2.a) is 240% slower. In **H–HPGM (stat)** (figure 2.b) the performance difference is 107%. And in **H–HPGM (stat+)** (figure 2.c) is only a 95% slower. This candidate distribution time is less representative, in terms of algorithm overall performance,



(a) H-HPGM (hash) (b) H-HPGM (stat) (c) H-HPGM (stat+)

Fig. 2. Candidate distribution time: Programmed vs. Dynamic

than the time spent in each of the database scans. Once the candidate itemsets are distributed, each node processes its partition of the database. For each of the records read from the database the itemsets counters are updated. If the itemset is stored locally it is updated immediately, otherwise the changes are queued in a buffer. When this buffer is full it is broadcasted to all the nodes.

Table 1. Average control time: Programmed vs. Dynamic Reactive Policies

Phase	**Programmed**	**Dynamic**	**% Increment**
First itemset location failure	12.11ms	198.12ms	1536.0033%
Next location failures	12.17ms	12.98ms	6.6557%

In this case, the control decisions performed by the operational plane have been programmed as reactive control agents. These agents cannot handle complex control decisions but their performance is much better. Table 1 presents the average time of component operations[1], comparing standard programmed implementations and dynamic reactive policies.

5 Conclusions

We have presented an alternative for programming distributed data mining algorithms. This alternative fills the lack of the flexibility of a traditional programmed solution allowing the user to manipulate algorithm behaviour in a very configurable way. As a drawback, dynamic control represents an overhead in terms of performance form the standard programmed implementations, but this performance is minimal ($\sim$ 6%) in record-by-record management and it is only important between two iteration phases. If we take into account that each iteration takes several minutes to be completed this performance loss is insignificant if it is compared with the new flexibility opportunities offered.

[1] first itemset location failure requires control propagation–delegation management

References

1. Peter Christen, Ole M. Nielsen, and Markus Hegland. DMtools – open source software for database mining. *In PKDD'2001*, 2001.
2. Robert L. Grossman, Stuart M. Bailey, Harinath Sivakumar, and Andrei L. Turinsky. Papyrus: A system for data mining over local and wide-area clusters and super-clusters. In ACM, editor, *SC'99*. ACM Press and IEEE Computer Society Press, 1999.
3. Mahesh V. Joshi, Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar. *CRPC Parallel Computing Handbook, chapter Parallel Algorithms for Data Mining. Morgan Kaufmann, 2000.*
4. *H. Kargupta, B. Park, D. Hershbereger, and E. Johnson. Advanced in Distributed and Parallel Knowledge Discovery*, chapter Collective Data Mining: A new perspective towards distributed data mining. AAAI Press / MIT Press, 2000.
5. Hillol Kargupta, Ilker Hamzaoglu, and Brian Stafford. Scalable, distributed data mining – an agent architecture. page 211.
6. Kensingston, Enterprise Data Mining. Kensington: New generation enterprise data mining. White Paper, 1999. Parallel Computing Research Centre, Department of Computing Imperial College, (Contact Martin Khler).
7. S. Krishnaswamy, S. W. Loke, and A. Zaslavsky. Cost models for distributed data mining. Technical Report 2000/59, School of Computer Science and Software Engineering, Monash University, Australia 3168, February 2000.
8. José M. Peña. *Distributed Control Architecture for Data Mining Systems.* PhD thesis, DATSI, FI, Universidad Politécnica de Madrid, Spain, June 2001. Spanish title: "Arquitectura Distribuida de Control para Sistemas con Capacidades de Data Mining".
9. José M. Peña and Ernestina Menasalvas. Towards flexibility in a distributed data mining framework. In *Proceedings of ACM-SIGMOD/PODS 2001*, pages 58–61, 2001.
10. Foster Provost. *Advances in Distributed and Parallel Knowledge Discovery*, chapter Distributed Data Mining: Scaling Up and Beyond, pages 3–28. AAAI Press/MIT Press, 2000.
11. O.F. Rana, D.W. Walker, M. Li, S. Lynden, and M. Ward. PaDDMAS: Parallel and distributed data mining application suite. In *Proceedings of the Fourteenth International Parallel and Distributed Processing Symposium*, 2000.
12. T. Shintani and M. Kitsuregawa. Parallel algorithms for mining association rule mining on large scale PC cluster. In Mohammed J. Zaki and Ching-Tien Ho, editors, *Workshop on Large-Scale Parallel KDD Systems*, San Diego, CA, USA, August 1999. ACM. in conjunction with ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD99).
13. S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. Cost-based modeling for fraud and instrusion detection: Results from the JAM project. In *DARPA Information Survivability Conference and Exposition*, pages 130–144. IEEE Computer Press, 2000.
14. M. Zaki. *Large-Scale Parallel Data Mining*, volume 1759 of *Springer Lecture Note in Artificial Intelligence*, chapter Parallel and Distributed Data Mining: An Introduction. Springer Verlag, 1999.