© 2016 IJSRSET | Volume 2 | Issue 1 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology
DOI : https://doi.org/10.32628/IJSRSET162168

Distributed Data Mining: Implementing Data Mining Jobs on Grid Environments

Vishal Bhemwala, Bhavesh Patel, Dr. Ashok Patel

Department of Computer Science, Hem. North Gujarat University, Patan, Gujarat, India

ABSTRACT

Data mining technology is not only composed by efficient and effective algorithms, executed as standalone kernels. Rather, it is constituted by complex applications articulated in the non-trivial interaction among hardware and software components, running on large scale distributed environments. This last feature turns out to be both the cause and the effect of the inherently distributed nature of data, on one side, and, on the other side, of the spatiotemporal complexity that characterizes many DM applications. For a growing number of application fields, Distributed Data Mining (DDM) is therefore a critical technology. In this research paper, after reviewing the open problems in DDM, we describe the DM jobs on Grid environments. We will introduce the design of Knowledge Grid System.

Keywords: Data Mining, Knowledge Grid, Distributed Data Mining

I. INTRODUCTION

Due to the logistic organization of the entities that collects data – either private companies or public institutions – data are often distributed at the origin. Such data are typically too big to be gathered at a single site or, for privacy issues, can only be moved, if ever possible, within a limited set of alternative sites. In this situation the execution of DM tasks typically involves the decision of how much data is to be moved and where. Also, summaries or other forms of aggregate information can be moved to allow more efficient transfers.

In other cases, data are produced locally but due to their huge volume cannot be stored in a single site and are therefore moved immediately after production to other storage locations, typically distributed on geographical scale. Examples are Earth Observing Systems (EOS), i.e. satellites sending their observational data to different earth stations, high energy physics experiments that produce huge volumes of data for each event and send the data to remote laboratories for the analysis. In these cases, data can be replicated in more than one site and repositories can have a multi-tier hierarchical organization. Problems of replica selection and caching management are typical in such scenarios.

The need for parallel and distributed architecture is not only driven by the data, but also by the high complexity of DM computations. Often the approach used by the DM analyst is exploratory, i.e. several strategies and parameter values are tested in order to obtain satisfactory results. Also, in many applications data are produced in streams that have to be processed on-line and in reasonable times with respect to the production rate of the data and of the specific application domain. Using high performance parallel and distributed architectures is therefore imperative.

II. DISTRIBUTED DATA MINING SYSTEM

By analyzing three different approaches, we have provided some definitions of DDM Systems. They pose different problems and have different benefits. Existing

DDM systems can in fact be classified in one of these approaches.

Data-Driven: The simplest model for a DDM system only takes into account the distributed nature of data, but then relies on local and sequential DM technology. Since in this system the focus is solely posed in the location of data, we refer to this model as data-driven.

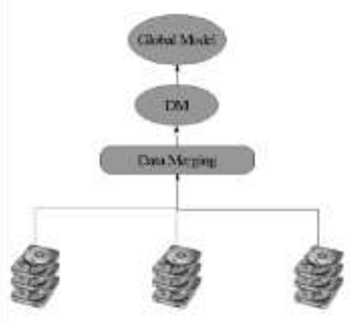

Figure 1 : Data-Driven Approach for Distributed Data Mining

In this model, data are located in different sites which do not need to have any computational capability. The only requirement is to be able to move the data to a central location in order to merge them and then apply sequential DM algorithms. The output of the DM analysis, i.e. the final knowledge models are then either delivered to the analyst' location or accessed locally where they have been computed.

The process of gathering data in general is not simply a merging step and depends on the original distribution. For example data can be partitioned horizontally – i.e. different records are placed in different sites – or vertically – i.e. different attributes of the same records are distributed across different sites. Also, the schema itself can be distributed, i.e. different tables can be placed at different sites. Therefore when gathering data it is necessary to adopt the proper merging strategy.

Model-driven: A different approach is the one we call model-driven. Here, each portion of data is processed locally to its original location, in order to obtain partial results referred to as local knowledge models. Then the local models are gathered and combined together to obtain a global model.

Also in this approach, for the local computations it is possible to reuse sequential DM algorithms, without any modification. The problem here is how to combine the partial results coming from the local models. Different techniques can be adopted, based on voting strategies or collective operations, for example. Multi-agent systems may apply meta-learning to combine partial results of distributed local classifiers.

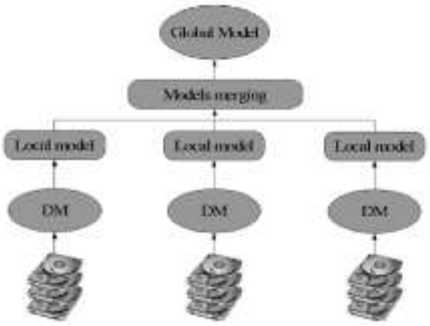

Figure 2 : Model-Driven Approach for Distributed Data Mining

The draw-back of the model-driven approach is than it is not always possible to obtain an exact final result, i.e. the global knowledge model obtained may be different from the one obtained by applying the data-driven approach (if possible) to the same data. Approximated results are not always a major concern, but it is important to be aware of that. Moreover, in this model hardware resource usage is not optimized. If the heavy computational part is always executed locally to data, when the same data is accessed concurrently, the benefits coming from the distributed environment might vanish due to the possible strong performance degradation.

Architecture-driven: In order to be able to control the performance of the DDM system, it is necessary to introduce a further layer between data and computation. As show in below Figure, before starting the distributed computation, we consider the possibility of moving data to different sites with respect to where they are originally located, if this turns out to be profitable in terms of performances. Moreover, we introduce a communication layer among the local DM computations, so that the global knowledge model is built during the local computation. This allows for arbitrary precision to be achieved, at the price of a higher communication overhead. Since in this approach for DDM the focus is on optimized resource usage, we refer to this approach as the architecture-driven.

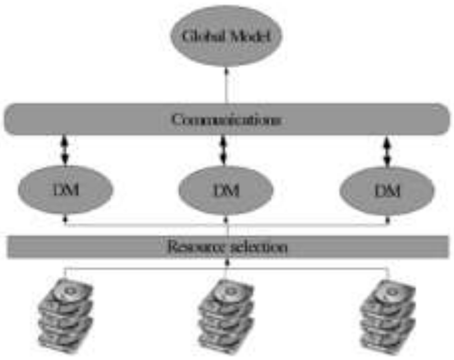

Figure 3 : Architecture-Driven Approach for Distributed Data Mining

The higher flexibility of this model and the potentially higher performance that it is possible to achieve, are payed in terms of the higher management effort that it is necessary to put in place. A suitable scheduling policy must be devised for the resource selection layer. Moreover, DM sequential algorithms are not reusable directly and must be modified or redesigned in order to take advantage of the communication channel among the different DM computations.

I. ISSUES IN DDM SYSTEM

Many architectural issues are involved in the definition of full DDM systems.

- Efficient communications are surely one of the main concerns.
- Try to optimize existing mechanisms for wide area data intensive applications.
- Efficient management of the resources available, namely scheduler components that have to determine the best hardware/software resources to execute the DDM.
- Worth mentioning is related to maintenance of the software components.

Rather third-parties can let the DDM system use their components, but remain the only responsible for updating or changing them when needed.

II. DATA AND KNOWLEDGE GRID

A significant contribution in supporting data intensive applications is currently pursued within the Data Grid effort, where a data management architecture based on storage systems and metadata management services is provided. The data considered here are produced by several scientific laboratories geographically distributed among several institutions and countries. Data Grid services are built on top of Globus, a middleware for Grid platforms, and simplify the task of managing computations that access distributed and large data sources.

The Data Grid frameworks share most of its requirements with the realization of a Grid based DDM system, where data involved may originate from a larger variety of sources. Even if the Data Grid project is not explicitly concerned with data mining issues, its basic services could be exploited and extended to implement higher level grid services dealing with the process of discovering knowledge from larger and distributed data repositories. Motivated by these considerations, in a specialized grid infrastructure named Knowledge Grid (K-Grid) has been proposed. This architecture was designed to be compatible with lower-level grid mechanisms and also with the Data Grid ones. The authors subdivide the K-Grid architecture into two layers: the core K-grid and the high level K-grid services. The former layer refers to services directly implemented on the top of generic grid services, the latter refers to services used to describe, develop and execute parallel and distributed knowledge discovery (PDKD) computations on the K-Grid. Moreover, the layer offers services to store and analyze the discovered knowledge.

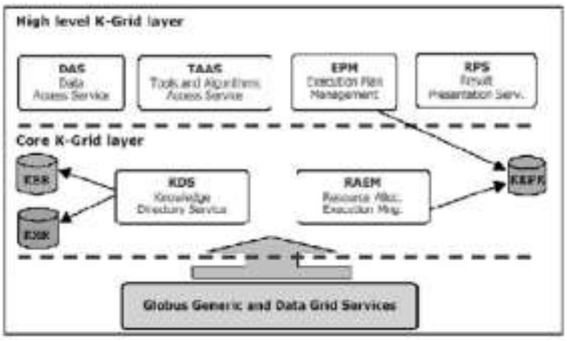

Figure 4 : General schema of the Knowledge Grid Architecture.

We concentrate our attention on the K-Grid core services, i.e. the Knowledge Directory Service (KDS) and the Resource Allocation and Execution Management (RAEM) services. The KDS extends the basic Globus Meta-computer Directory Service (MDS), and is responsible for maintaining a description of all the data and tools used in the K-Grid. The metadata managed by the KDS are represented through XML documents stored in the Knowledge Metadata Repository (KMR). Metadata regard the following kind of objects: data sources characteristics, data management tools, data mining tools, mined data, and data visualization tools. Metadata representation for output mined data models may also adopt the (PMML) standard.

The RAEM service provides a specialized broker of Grid resources for DDM computations: given a user request for performing a DM analysis, the broker takes allocation and scheduling decisions, and builds the execution plan, establishing the sequence of actions that have to be performed in order to prepare execution (e.g., resource allocation, data and code deployment), actually execute the task, and return the results to the user. The execution plan has to satisfy given requirements (such as performance, response time, and mining algorithm) and constraints (such as data locations, available computing power, storage size, memory, network bandwidth and latency). Once the execution plan is built, it is passed to the Grid Resource Management service for execution. Clearly, many different execution plans can be devised, and the RAEM service has to choose the one which maximizes or minimizes some metrics of interest (e.g. throughput, average service time).

III. DESIGN OF KNOWLEDGE GRID SYSTEM

We describe here the design of KGS. A model for the resources of the K-Grid, described in below figure, is composed by a set of hosts, onto which the DM tasks are executed, a network connecting the hosts and a centralized scheduler, KGS, where all requests arrive.

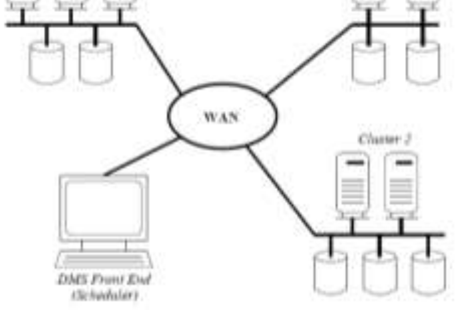

Figure 5 : Physical resources in K-Grid.

The first step is that of task composition. We do not actually deal with this phase and we only mention it here for completeness. As explained earlier, we consider that the basic building blocks of a DM task are algorithms and datasets.

DM components correspond to a particular algorithm to be executed on a given dataset, provided a certain set of input parameters for the algorithm. We can therefore describe each DM components _ with the triple:

$$\mathbf{A = (A, D, \{P\})}$$

where A is the data mining algorithm, D is the input dataset, and {P} is the set of algorithm parameters. For example if A corresponds to “Association Mining”, then {P} could be the minimum confidence for a discovered rule to be meaningful. It is important to notice that A does not refer to a specific implementation. We could therefore have more different implementations for the same algorithm, so that the scheduler should take into account a multiplicity of choices among different algorithms and different implementations. The best choice could be chosen considering the current system status, the programs availability and implementation compatibility with different architectures.

The original DM task on the left hand side, is composed by the application of a first clustering algorithm on a certain dataset, and then by the application of an algorithm for association mining on each cluster found. Finally all the results are gathered for visualization. We add a node to the top of the graph, which corresponds to the initial determination of the input dataset. Moreover, we detail the structure of the actual computation performed when we chose a specific implementation for each software component.

In this way, starting from a semantic DAG, we define a physical DAG, derived from the first one, with all the components mapped onto actual physical resources. This process is repeated for all the DAGs that arrive at the scheduler.

The global vision of the system is summarized in below Figure. The first step is the creation of the semantic DAGs from the basic components. This step is in general performed by several users at the same time. Therefore we have a burst of DAGs that must be mapped on the system. Semantic DAGs queue in scheduler and wait their turn. When a DAG is processed, the scheduler builds the physical and determines the best set of resources where the DAG can be mapped. This is done by taking into account current system status, i.e. network and machines load as induced by previous mappings, and also by verifying that all data dependencies are satisfied. Referring to the example above, the scheduler must first schedule the clustering algorithm and then the association mining.

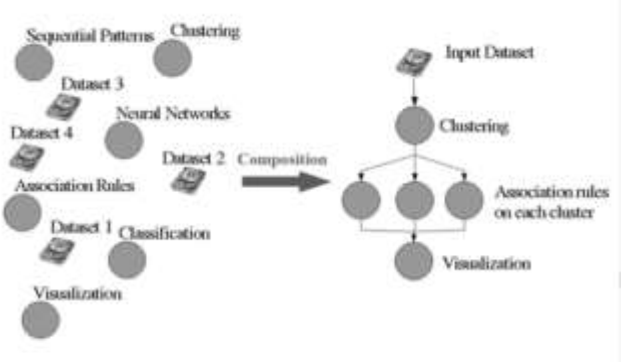

Figure 6 : Composition of a DM DAG in terms of basic building blocks: Datasets and algorithms.

Scheduling DAGs on a distributed platform is a non-trivial problem which has been faced by a number of algorithms in the past. Although it is crucial to take into account data dependencies among the different components of the DAGs present in the system, we first want to concentrate ourselves on the cost model for DM tasks and on the problem of bringing communication costs into the scheduling policy. For this reason, we introduce in the system an additional component that we call serialized, whose purpose is to decompose the tasks in the DAG into a series of independent tasks, and send them to the scheduler queue as soon as they become executable w.r.t. the DAG dependencies.

Such serialization process is not trivial at all and leaves many important problems opened, such as determine the best ordering among tasks in a DAG that preserver data dependencies and minimizes execution time.

IV.CONCLUSIONS

We designed a simulation framework to evaluate our MCT (Minimum Completion Time) on-line scheduler, which exploits sampling as a technique for performance prediction. We thus compared our MCT + sampling approach with a blind mapping strategy. Since the blind strategy is unaware of actual execution costs, it can only try to minimize data transfer costs, and thus always maps the tasks on the machines that hold the corresponding input datasets. Moreover, it cannot evaluate the profitability of parallel execution, so that sequential implementations are always preferred. Referring to the architectures for DDM systems proposed, here we are comparing the performance of an architecture-driven scheduler with those of a data-driven one (blind). The simple data-driven model turns out to be less effective in scheduling both communications and computations of DDM on the K-Grid.

We essentially checked the feasibility of our approach before actually implementing it within the K-Grid. Our goal was thus to evaluate mapping quality, in terms of makespan, of an optimal on-line MCT+sampling technique. We also assumed to also know in advance (through an oracle) the exact cost of the sampled tasks, instead of assuming an arbitrary small constant. In this way, since our MCT+sampling technique works in an optimal way, we can evaluate the maximal improvement of our technique over the blind scheduling one.

We analyzed the effectiveness of a centralized on-line mapper based on the MCT heuristics, which schedules DM tasks on a small organization of a K-Grid. The mapper does not consider node multitasking, is responsible for scheduling both dataset transfers and computations involved in the execution of a given task t_i, and also is informed about their completions. The MCT mapping heuristics adopted is very simple. Each time a task t_i is submitted, the mapper evaluates the expected ready time of each machine and communication links. The expected ready time is an estimate of the ready time, the earliest time a given resource is ready after the completion of the jobs previously assigned to it. On the basis of the expected ready times, our mapper evaluates all possible assignment of t_i, and chooses the one that reduces the completion time of the task. Note that such estimate is based on both estimated and actual execution times of all the tasks that have been assigned to the resource in the past. To update resource ready times, when data transfers or computations involved in the execution of t_i complete, a report is sent to the mapper.

V. REFERENCES

[1] M. Cannataro, C. Mastroianni, D. Talia, and Trunfio P. Evaluating and enhancing the use of the gridftp protocol for efficient data transfer on the grid. In Proc. of the 10th Euro PVM/MPI Users' Group Conference, 2003.

[2] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: towards an architecture for the distributed management and analysis of large scientific datasets. J. of Network and Comp. Appl., (23):187–200, 2001.

[3] I. Foster and C. Kasselman. The Grid: blueprint for a future infrastructure. Morgan Kaufman, 1999.

[4] Bart Goethals. Efficient Frequent Itemset Mining. PhD thesis, Limburg University, Belgium, 2003.

[5] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke. Gridftp protocol specification. Technical report, GGF GridFTP Working Group Document, 2002.

[6] R. L. Grossman and R. Hollebeek. Handbook of Massive Data Sets, chapter The National Scalable Cluster Project: Three Lessons about High Performance Data Mining and Data Intensive Computing. Kluwer Academic Publishers, 2002.

[7] H. Kargupta, W. S. K. Huang, and E. Johnson. Distributed clustering using collective principal components analysis. Knowledge and Information Systems Journal, 2001.

[8] H. Kargupta, B. Park, E. Johnson, E. Sanseverino, L. Silvestre, and D. Hershberger. Collective data mining from distributed vertically partitioned feature space. In Proc. of Workshop on distributed data mining, International Conference on Knowledge Discovery and Data Mining, 1998.

[9] M. Marzolla and P. Palmerini. Simulation of a grid scheduler for data mining. Esame per il corso di dottorato in informativa, Universita' Ca' Foscari, Venezia, 2002.

[10] C. L. Parkinson and R. Greenstonen, editors. EOS Data Products Handbook. NASA Goddard Space Flight Center, 2000.

[11] A. L. Prodromidis, P. K. Chan, and S. J. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In Advances in Distributed and Parallel Knowledge Discovery. AAAI/MIT Press, 2000.