A Scheduling Algorithm for Running Bag-of-Tasks Data Mining Applications on the Grid

Fabrício A.B. da Silva, Sílvia Carvalho, and Eduardo R. Hruschka

Universidade Católica de Santos (Unisantos)
R. Dr. Carvalho de Mendonça, 144 – 11070-100 – Santos (SP)
{fabricio,mygridgene,erh}@unisantos.edu.br

Abstract. Data mining applications are composed of computing-intensive processing tasks, which are natural candidates for execution on high performance, high throughput platforms such as PC clusters and computational grids. Besides, some data-mining algorithms can be implemented as Bag-of-Tasks (BoT) applications, which are composed of parallel, independent tasks. Due to its own nature, the adaptation of BoT applications for the grid is straightforward. In this sense, this work proposes a scheduling algorithm for running BoT data mining applications on grid platforms. The proposed algorithm is evaluated by means of several experiments, and the obtained results show that it improves both scalability and performance of such applications.

1 Introduction

Knowledge discovery in databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [1]. Data Mining (DM) is a step in this process that centers on the automated discovery of new facts and relationships in data. DM consists of three basic steps: data preparation, information discovery and analysis of the mining algorithm output. All these steps exploit huge amounts of data and are computationally expensive. In this sense, several techniques have been proposed to improve the performance of DM applications, such as parallel processing [3] and implementations based on cluster of workstations [4].

A computational grid, or simply grid for short, provides access to heterogeneous resources in a geographically distributed area, allowing the integration of these heterogeneous and distributed resources into a unified computer resource. Computational grids are usually built on top of specially designed middleware platforms, the so-called grid platforms. Grid platforms enable the sharing, selection and aggregation of a variety of resources including supercomputers, servers, workstations, storage systems, data sources and specialized devices that are geographically distributed and owned by different organizations [5].

Among the most suitable applications for running on a grid are the Bag-of-Tasks (BoT) applications, which are parallel applications whose tasks are independent of each other. Examples of BoT applications include Monte Carlo simulations, massive searches (such as key breaking), parameter sweeps, image manipulation, and data mining. In this work, we analyze the use of computational grids for data mining.

M. Danelutto, D. Laforenza, M. Vanneschi (Eds.): Euro-Par 2004, LNCS 3149, pp. 254–262, 2004.

BoT data mining applications can present scalability problems when executing on the grid. In this paper, we define scalability as the ability of a grid platform to reduce the total execution time of a BoT application as the number of machines added to the grid increases, given that the number of tasks of the BoT application is larger than the number of machines in the grid. This paper proposes a scheduling algorithm that improves scalability of BoT data mining applications on the grid, by grouping sets of independent tasks into larger ones, which are executed on grid machines. For the experimental results we used the MyGrid platform [6], which is specially designed to run BoT applications.

The remaining of this paper is organized as follows: Section 2 approaches the employed data mining algorithm (K-Means) in the context of BoT applications. This section also presents experimental results related with the scalability problem to be tackled by our scheduling algorithm. This algorithm is described and evaluated in Section 3. Finally, Section 4 presents the conclusions and points out some future work.

2 Running Data Mining Algorithms as BoTs Applications

Some recent works suggest that grids are natural platforms for developing high performing data mining services [7,8,9]. For instance, Orlando et al. [8] describe the application of two data mining algorithms in the Knowledge Grid [7]: DCP and K-means. The former algorithm enhances the popular Apriori Algorithm [10], which is an algorithm for frequent set counting, whereas the latter one is a clustering algorithm. In this work, we have also employed the K-Means algorithm (briefly described in Section 2.1) implemented as a BoT application on the MyGrid platform (depicted in Section 2.2). Section 2.3 reports experimental results that illustrate the need for a specific scheduling algorithm specially designed to run BoT applications.

2.1 K-Means Algorithm

Clustering algorithms involving the calculation of the mean (centroid) of each cluster are often referred to as K-means algorithms [2]. Basically, these algorithms partition a dataset of *n* objects into K clusters. The K value, which represents the number of clusters, is specified by the user. These algorithms try to minimize the error sum of distances (e.g. Euclidean) between objects of a cluster and its centroid. The implementation employed in our work can be summarized as follows:

1. Generate a random initial partition of objects into K nonempty clusters;
2. Compute the cluster centroids (mean vectors) of the current partition;
3. Assign each object to the cluster with the nearest centroid;
4. If the convergence criterion has been met, stop. Otherwise, go to step 2.

The convergence criterion can be defined either as the maximum number of iterations (*t)* of steps 2 and 3 or as the maximum absolute difference between centroids in two consecutive iterations. The K-Means Algorithm has two main drawbacks: (i) it may get stuck at local optimal partitions; (ii) the user has to provide the number of clusters (K). Besides, the K-Means can sometimes end up with less than K clusters if,

in a set of K centroids, at least one is worse than all the other ones. This situation can happen, for instance, if all objects are nearer to the other (K-1) centroids. However, this is not a hard limitation, because it can indicate that there are less than K clusters. Indeed, the choice of K is very hard and a usual practical approach is to try several different solutions (one for each K-value) and then choose the most suitable one, by plotting the value of the clustering criterion against the number of clusters. In a grid environment, this approach can be performed in a parallel way, in which each machine runs the algorithm for a specific K-value. In addition, it is also useful to run the K-Means for different initial partitions, and it can also be performed in parallel (this approach is employed in our work). Due to those characteristics, the K-Means algorithm is suitable for execution as a BoTs application on the MyGrid platform. In addition, K-Means is suitable for our experiments, because of its efficiency - $O(tKn)$.

2.2 The Platform MyGrid

The MyGrid platform [6] was conceived to support the execution of BoT applications, which constitute a class of parallel applications that can be partitioned in several independent tasks. Usually, these tasks have an infrequent need for communication.

The main benefits of MyGrid are twofold: minimal installation effort and ease of use. Most grid platforms (for example see [11]) can only be installed and configured by system administrators. Moreover, installation procedures are usually manually repeated in a considerable number of machines. MyGrid enables regular users to create their own grid to run applications on whatever resources they have access to, without the need for these users to get involved into grid details and administrative procedures for heterogeneous platforms.

Since MyGrid focuses on BoT applications, its working environment consists of a small set of services to enable its users to manipulate their files on the grid. Consequently, no previous software installation nor shared file system are needed on machines. A user is required to install and configure MyGrid only on one machine, which is called home machine. Interactions with other machines are supported by the Grid Machine Interface (GMI), i.e., a minimal set of services that must be available in a machine so it can be used as a machine for grid computing, the so-called grid machine. These services consist of: (1) remote executing on a grid machine; (2) termination of a running task; and (3) file transfers between the home and grid machines.

Mygrid implements two task scheduling algorithms, *Work Queue*[6] and *Work Queue with Replication* (WQR) [12]. WQR starts the execution of each task on idle grid machines just like the Workqueue algorithm. Once the queue is empty, Mygrid starts execution of replicated instances of unfinished tasks on idle machines. Once a task completes, all other replicas are terminated. If a machine is overloaded or fails, the WQR will eventually re-execute the corresponding task on another machine.

2.3 Running K-Means as a Bag-of-Tasks Application

Our experiments were performed in a dataset formed by nine overlapping clusters. These clusters contain 100 objects, randomly generated using bi-dimensional Gaussian distributions with standard deviations equal to 0.5 and centers [x,y] given by [1,1],[3,1],[5,1],[1,3],[3,3],[5,3],[1,5],[3,5],[5,5]. We have employed K=150 clusters

and t=500 iterations. These parameter values were chosen because they are suitable to characterize the scalability of the grid platform and to assess the performance of the proposed scheduling algorithm.

Initially, experiments to estimate the performance gains were executed on a grid of 25, 30 and 35 machines. Twenty machines are located at UniSantos and the others at the Federal University of Campina Grande, Brazil. A distance of about 3000 kilometers separates these two sites. The machines at UniSantos are 1.8 GHz Pentium IV machines with 128 MB of RAM. Machines at Campina Grande have 1.8GHz Pentium IV processors with 630 MB of RAM. The results shown in Figure 1 are the average of ten executions at different times of the day. A Pentium IV with 1GB of main memory was used as the *home machine*. This machine is dedicated to provide dataset distribution, task dispatching to grid machines and output file gathering. The scheduling strategy used was the standard Workqueue algorithm. Version 1.1 of MyGrid was used in experiments.

The execution time of DM applications can be significantly reduced on a Grid environment, but the scalability of a grid with more than 30 machines is poor, as illustrated in Figure 1. It is due to both the application behavior and the way MyGrid interacts with grid machines to manage input and output files. Thus, a more detailed view of the MyGrid working environment becomes necessary.

Each MyGrid task is divided in three subtasks (*init, remote* and *collect*), which are performed sequentially, in that order. Both *init* and *collect* subtasks are executed on the home machine, to send input files to grid machines and to collect the output files back to the home machine, respectively. The *remote* subtask runs on grid machines, and it is responsible to perform the computation itself. Depending on the size of input and output files and the number of grid machines, the execution of *init* and *collect subtasks* on the *home machine* may become a bottleneck. The overhead related to file transfers in the home machine depends on both the size of the input and output files and the number of machines added to the grid. In addition, the data transfer rate observed in the home machine can also be affected by the running time of *remote* subtasks. The shorter is the execution time of *remote* subtasks, the more frequent is the transfer of input and output files, which increases the data transfer rate in the home machine.

The main cause of the poor scalability depicted in Figure 1 is the bottleneck that appeared at the home machine. For each application task, a small number of processes are created at the home machine to manage the transferring of input and output files to and from grid machines. Such processes are eliminated as soon as their corre-

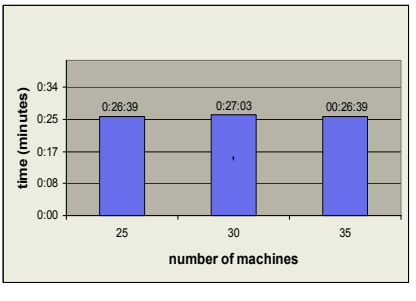

Fig. 1. Average execution times in a Grid environment (350 tasks).

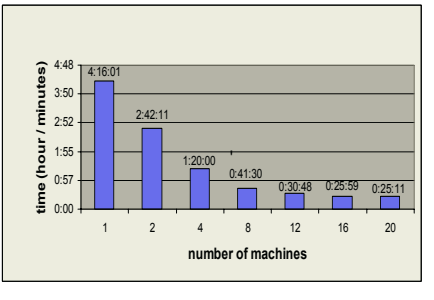

Fig. 2. Average execution times in a dedicated cluster environment.

sponding tasks have been completed. Thus, short-duration application tasks can increase the rate of creation/destruction of such processes to critical levels. Moreover, these processes execute concurrently, competing for memory, I/O subsystem, network bandwidth, CPU cycles, and other resources. Our tests have shown that the scalability of the application can be severely impacted when we execute a large number of tasks, due mainly to the bottleneck created at the home machine. This fact is evidenced by a new set of experiments generated on a dedicated cluster of Pentium IV machines. These results are shown in Figure 2.

For the results shown in Figure 2, we run the set of tasks sequentially on a single, stand alone machine with a Pentium IV (1.8 GHz) processor, and 1 GB of main memory. Then, the same set of tasks were run on 2, 4, 8, 12, 16 and 20 dedicated machines with similar hardware characteristics, interconnected by a 100 Mbps Ethernet LAN. The schedulling strategy was the standard Workqueue algorithm. By examining Figure 2, it is clear that scalability is poor for cluster platforms with more than 16 machines.

3 A Scheduling Algorithm for Data Mining Tasks

The bottleneck represented by the home machine can be characterized by examining the execution times of each independent task and isolating the times required for the corresponding *init*, *remote* and *collect* subtasks. For very short executions, like the ones here performed, the time waste transferring data sometimes is larger then the computational time. Beyond that, there is a reduction of performance when more machines are aggregated, because the home machine needs to manage more processes which compete for resources. In Mygrid 1.1, for each application task, a small number of processes are created to manage the transfer of input and output files to and from grid machines. Such processes are terminated as soon as their tasks have been accomplished. Thus, short application tasks can increase the rate of creation/destruction of such processes to critical levels. Moreover, these processes execute concurrently, competing for memory, I/O subsystem, network bandwidth, CPU cycles, and other resources in the home machine.

One way of reducing the impact of the bottleneck represented by the home machine on the scalability of the platform is to group sets of tasks in one larger task and then execute the “big” task in a grid machine. In the following we will refer to this big task as a job. The main idea is to group a set on tasks into one execution in a remote grid machine, when the number of machines of the grid grows up and the execution time of an individual task is small, i.e., having the same order of magnitude of the time need to transfer input and output files over the network.

In the following we propose scheduling algorithms for reducing the impact of the bottleneck at the home machine. As a consequence, scalability is improved. Two different platforms are considered: a dedicated cluster, in which all machines are located in the same administrative domain, and a grid environment composed of heterogeneous machines located at different sites.

For the results in the following sections we run a BoT K-means application composed of 350 independent tasks. These tasks simulate possible variations of initial partitions in the K-means algorithm (described in Section 2.1).

3.1 Dedicated Cluster

In a dedicated cluster we used the following algorithm to group tasks: (i) Tasks are grouped in a way that only one job (group of independent tasks) is created per machine. By considering just one job per machine we reduce the amount of work to be done by the home machine in managing jobs; (ii) Tasks are clustered into jobs in a way that the time needed for processing each job is about the same for every machine. In order to do so, static information about the configuration of each machine (such as processor speed and memory) should be available. Therefore, the amont of work to be executed by each processor is statically balanced.

We performed initial experiments using a cluster comprised of 20 machines at Unisantos in the same administrative domain. All machines have one Pentium IV (1.8 GHz) processor, and 128 MB of main memory. For those executions we used a home machine with 1 GB of main memory. Results for clusters composed of 8, 12, 16 and 20 machines are shown in Figure 3.

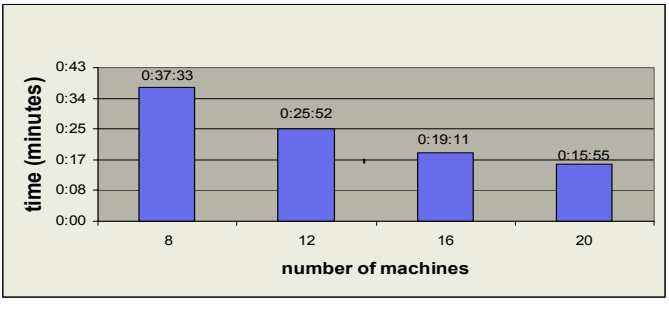

Fig. 3. Average execution times in a dedicated cluster - tasks grouped into jobs.

Since the platforms are composed of homogeneous machines, the number of independent tasks assigned to each machine is an integer in the interval $[\lfloor T/P \rfloor, \lceil T/P \rceil]$, where T is the total number of tasks (in this case 350) and P is number of machines. It is worth noting the improvement in scalability when tasks are grouped into jobs.

Since the files to be processed by each task are the same in this application, it is possible to send the files just once. As a consequence there is a reduction on both the amount of processing needed to manage remote tasks in home machine and network traffic. As a consequence there is significant reduction in the overall makespan, which can be verified by comparing the results in Figure 3 to the results in Figure 2.

3.2 Grid Platform

In grid platforms, faults are common, and this fact should be taken into account explicitly to define a scheduling algorithm. The degree of heterogeneity is also much larger in a grid, when comparing to a cluster. We used the algorithm proposed for the dedicated cluster as a starting point, and modified it as described below:

(i) Just like the dedicated cluster algorithm, tasks are grouped in a way that only one job is created per machine.
(ii) Cluster tasks into jobs in a way that the time needed for processing each job is about the same for every machine, *as if the machines are dedicated*, by using static information. In the case of a grid the static information should include the machine is local, i.e., if the machine is in the same administrative domain of the home machine. A larger amount of work should be assigned to local machines, as network delays are considerably smaller in this case.
(iii) For every **x** tasks completed, the grid machine should send the corresponding results back to the home machine. This mechanism is similar to a regular checkpointing. If the grid machine fails, the only tasks that have to be executed are those for which the results were not received. It is possible to obtain information about the current load of a grid machine by measuring the time needed for executing **x** tasks. Beyond that, the home machine can assert the failure of a grid machine by considering a dynamically adjustable timeout. If the home machine does not receive any results for a period of time equal or larger than the timeout, the grid machine is considered offline.
(iv) If a machine becomes idle, send a replica of the remaining tasks of the slowest machine to the idle machine. If the slowest machines' tasks have already been replicated, consider the second slowest machine, and so on (only one replica per task).

We also run experiments in a grid environment comprised of 25, 30 and 35 heterogeneous machines, 20 machines from Unisantos and the others from Federal University of Campina Grande. Initial results are shown in Figure 4. Given the static information available about the machines and networks, the load was distributed in a way that about 60 % to 80% of the tasks were initially assigned to local machines at Unisantos. The actual figure depends on the number of local machines that compose the grid. The other tasks were assigned to remote (Campina Grande) machines. Despite the short execution times and the relatively small number of machines, the results of Figure 4 show a very significant improvement when compared to results shown in Figure 1. Both the scalability and the performance are considerably improved. For instance, for the 35 machines plataform, the performance improvement in the overall makespan is about 56%.

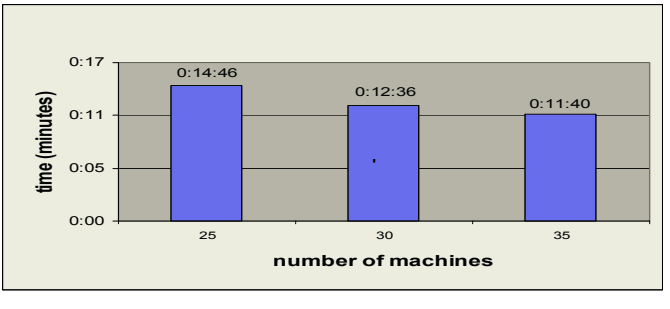

Fig. 4. Average execution times in a grid environment – tasks grouped into jobs.

4 Conclusion

This paper evaluated the scalability of a data mining method (K-Means Algorithm) implemented as a BoT application, using the middleware Mygrid. We proposed a scheduling strategy aimed to improve scalability, by grouping together sets of independent tasks to be executed in a grid machine. Our group is now investigating ways of improving even further the scalability of the platform by distributing the functions associated to the home machine among several machines.

Acknowledgements

We would like to acknowledge Walfredo Cirne and all the MyGrid/OurGrid team for the support provided during the execution of experiments. This work was developed in collaboration with HP Brazil R&D. Eduardo Raul Hruschka acknowledges CNPq (proc. 301.353/03-4) for its financial support.

References

1. Fayyad, U. M., Shapiro, G. P., Smyth, P. "From Data Mining to Knowledge Discovery : An Overview". In: Advances in Knowledge Discovery and Data Mining, Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., Editors, MIT Press, pp. 1-37, 1996.
2. Witten, I. H., Frank, E., Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, USA, 2000.
3. Freitas, A.A., Lavington, S.H., *Mining Very Large Databases with Parallel Processing*, Kluwer Academic Publishers, 1998.
4. Baraglia, R., Laforenza, D., Orlando, S., Palmerini, P., Perego, R., Implementation Issues in the Design of I/O Intensive Data Mining Applications on Clusters of Workstations, Proceedings of the 3rd Workshop on High Performance Data Mining, International Parallel and Distributed Processing Symposium 2000, Cancun, Mexico, May 2000.
5. Baker, M., Buyya, R., Laforenza, D. *Grids and Grid Technologies for Wide-area Distributed Computing*, Software, Pratice and Experience, v. 32, pp. 1437-1466, John Wiley and Sons, 2002.
6. Walfredo Cirne, Daniel Paranhos, Lauro Costa, Elizeu Santos-Neto, Francisco Brasileiro, Jacques Sauvé, Carla Oshtoff, Fabrício Silva, Cirano Silveira. *Running Bag-of_Tasks Applications on Ccmputational Grids: The MyGrid Approach*. Proceedings of the 2003 International Conference on Parallel Processing, October 2003.
7. Canataro, M., Talia, D. *The Knowledge Grid*, Communications of ACM, v.46, n.1, 2003.
8. Orlando, S., Palmerini, P., Perego, R., Silvestri, F., *Scheduling High Performance Data Mining Tasks on a Data Grid Environment*, Proceedings of Int. Conf. Euro-Par 2002, 27-30 August 2002, Paderborn, Germany, LNCS 2400 - Springer-Verlag - Pag. 375-384.
9. Hinke, H., Novotny, J., *Data Mining on NASA´s Information Power Grid*, HPDC 2000, Pittsburgh, Pennsylvania, USA, pp.292-293, IEEE Computer Society.
10. Agrawal, R., Mannila, H., Srikant, R., Tiovonen, H., Verkamo, A.I., *Fast Discovery of Association Rules* In: Advances in Knowledge Discovery and Data Mining, Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., Editors, MIT Press, pp. 307-328, 1996.

11. I. Foster, C. Kesselman. *Globus: A Metacomputing Infrastructure Toolkit.* Intl J. Supercomputer Applications, 11(2):115-128, 1997.
12. Daniel Paranhos, Walfredo Cirne, Francisco Brasileiro *Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids* Proceedings of International Conference on Parallel and Distributed Computting, 2003.