Future Generation Computer Systems 54 (2016) 68–78

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Distributed volunteer computing for solving ensemble learning problems

Eugenio Cesario *, Carlo Mastroianni, Domenico Talia

ICAR-CNR, via P. Bucci 41C, 87036 Rende (CS), Italy

HIGHLIGHTS

- We present MINING@HOME, a framework for distributed data mining.
- MINING@HOME combines the benefits of P2P protocols with those of the volunteer computer paradigm.
- The framework is used to discover classifiers by applying the "bagging" technique on real data.
- We present performance results showing the efficiency and scalability of our approach.
- Performance results are obtained through real experiments, simulation and analytical assessment.

ARTICLE INFO

Article history:
Received 30 July 2014
Received in revised form
15 May 2015
Accepted 22 July 2015
Available online 3 August 2015

Keywords:
Ensemble learning
Distributed data mining
Peer-to-peer
Volunteer computing

ABSTRACT

The volunteer computing paradigm, along with the tailored use of peer-to-peer communication, has recently proven capable of solving a wide area of data-intensive problems in a distributed scenario. The MINING@HOME framework is based on these paradigms and it has been implemented to run a wide range of distributed data mining applications. The efficiency and scalability of the architecture can be fully exploited when the overall task can be partitioned into distinct jobs that may be executed in parallel, and input data can be reused, which naturally leads to the use of data cachers. This paper explores the opportunities offered by MINING@HOME for coping with the discovery of classifiers through the use of the bagging approach: multiple learners are used to compute models from the same input data, so as to extract a final model with high statistical accuracy. Analysis focuses on the evaluation of experiments performed in a real distributed environment, enriched with simulation assessment – to evaluate very large environments – and with an analytical investigation based on the iso-efficiency methodology. An extensive set of experiments allowed to analyze a number of heterogeneous scenarios, with different problem sizes, which helps to improve the performance by appropriately tuning the number of workers and the number of interconnected domains.

1. Introduction

The global information society is a restless producer and exchanger of huge volumes of data and an increasing effort is needed for the extraction of valuable information, useful for business and scientific applications, from data. Fortunately, the notable advancements and the advent of new paradigms for distributed computing, such as Grids, P2P systems, and Cloud Computing, help us in many scenarios to cope with this data deluge. The efficiency of distributed approaches to data analysis has improved for several reasons: (i) the wide availability of Cloud infrastructures, which allow even a small company to offload applications to remote data centers or to integrate on-premises hosts with elastic resources utilized in a pay-per-use fashion; (ii) data links have larger bandwidths than before, enabling the assignment of tasks and the transmission of related input data in a distributed scenario; (iii) data caching techniques can help to reuse data needed by different tasks, (iv) Internet computing models such as the "public resource computing" or "volunteer computing" paradigm facilitate the use of spare CPU cycles of a large number of computers.

The algorithms, methodologies and architectural efforts aiming to extract knowledge in a distributed scenario are collectively known as "distributed data mining" [1,2]. Knowledge discovery is speeded up by concurrently executing a number of data mining tasks on different data subsets: specific attention must be given

* Corresponding author.
E-mail addresses: cesario@icar.cnr.it (E. Cesario), mastroianni@icar.cnr.it (C. Mastroianni), talia@dimes.unical.it (D. Talia).

http://dx.doi.org/10.1016/j.future.2015.07.010
0167-739X/

to the efficient combination of distributed analysis of data and centralized collection of results.

This paper deeply investigates the performance of MINING@HOME, a framework designed to solve distributed data mining problems through the distribution of data and parallelization of mining tasks. The approach used in the framework combines solutions developed in two different fields, volunteer computing and peer-to-peer data mining. Volunteer computing [3] has become a success story for many scientific applications, as a means for exploiting huge amount of low cost computational resources with a few manpower getting involved. So far this field has experienced little integration with the area of distributed and peer-to-peer data mining. The main reason for this is the centralized nature of popular volunteer computing platforms available today, such as *BOINC* [4] and *XtremWeb* [5,6], which requires all data to be served by a group of centrally maintained servers. However, the centralized approach can generate bottlenecks and single points of failure in the system. Moreover, a centralized solution is not naturally suited for applications in which input data files are initially stored in distributed locations.

These considerations inspired the design of the MINING@HOME architecture, which exploits the methodologies of volunteer computing and tailors them to properly match the characteristics and benefits of peer-to-peer protocols and algorithms. The MINING@HOME architecture is data-oriented, and it exploits distributed cache servers for the efficient dissemination and reutilization of data files. This kind of solution can improve the performance of public computing systems, in terms of efficiency, flexibility and robustness, and it can also enlarge the use of the public computing paradigm, since any user is allowed to define its own data mining application and specify the jobs that will be executed by remote volunteers. The approach differs from the centralized architectures, such as that one used in BOINC, in that MINING@HOME integrates P2P networking directly into the system, as job descriptions and input data are provided to a P2P network instead of being directly delivered to the hosts that execute the tasks.

In [7], MINING@HOME was assessed through an early simulator, and it proved able to extract *closed frequent itemsets* from a transactional database. After those early simulations, we worked to provide a full implementation in Java of the framework, and now it can efficiently support the execution of different data mining applications in distributed scenarios. To the best of our knowledge, MINING@HOME is the first implemented volunteer computing framework that run data mining tasks in a distributed environment. In [8] the basic implementation of the framework was sketched together with a preliminary evaluation on ensemble learning tasks. The ensemble learning approach combines multiple mining models together instead of using a single model in isolation [9]. In particular, the "bagging" strategy consists of sampling an input dataset multiple times, to introduce variability between the different models, and then extracting the combined model with a voting technique or a statistical analysis.

This paper extends the work presented in [8] in many ways: (i) the experimental testbed was extended from two to four computing domains, and the size of input data was extended from 2 million instances to 5 million instances of a reference dataset; (ii) experimental evaluations were complemented with a simulation tool that incorporates data coming from real experiments (for example, job durations) enabling the assessment of wider scenarios, with up to 128 workers distributed among 32 domains; (iii) mathematical analysis, based on iso-efficiency methodologies, was used to investigate how the performances, in particular in terms of execution time and speedup, are related to the number of workers, the number of domains and the dataset size. The results confirm the feasibility of the approach, the scalability and efficiency of the framework, and also show that it may be possible to optimize the performance by choosing the appropriate system and network configuration, for example, by tuning the number of workers and the number of domains on which the workers are distributed.

The reminder of the paper is organized as follows: Section 2 presents the architecture of MINING@HOME and the peer-to-peer protocol used for the assignment of tasks to workers. Section 3 discusses the ensemble learning strategy. Section 4 illustrates the scenario of the experiments, presents the results obtained in a real testbed and via simulation, and uses the iso-efficiency model to separately evaluate the contributions of useful and overhead computation. Finally, Section 5 discusses related work, specifically in the fields of distributed data mining and public resource computing, and Section 6 concludes the paper.

2. Architecture and implementation of Mining@home

As already mentioned, a simulator of the MINING@HOME framework was introduced in [7] for solving the problem of finding closed frequent itemsets in a transactional database, and simulation results were reported. After that, the MINING@HOME system was fully implemented and used for coping with a number of different data analysis scenarios involving the execution of different data mining tasks in a distributed environment. The architecture of the MINING@HOME framework distinguishes between nodes accomplishing the mining task and nodes supporting data dissemination. In the first group:

- the **data source** is the node that stores the dataset to be read and mined.
- the **job manager** is the node in charge of decomposing the overall data mining application in a set of independent tasks. This node produces a *job advert* document for each task, which describes its characteristics and specifies the portion of the data needed to complete the task. The job manager is also responsible for the collection of results.
- the **miners** are the nodes available for job execution. Assignment of jobs follows the "pull" approach, as required by the volunteer computing paradigm.

Data exchange and dissemination is done by exploiting the presence of a network of super-peers for the assignment and execution of jobs, and adopting caching strategies to improve the efficiency of data delivery. Specifically:

- **super peer** nodes constitute the backbone of the network. Miners connect directly to a super-peer, and super-peers are connected with one another through a high level P2P network.
- **data cachers** nodes operate as data agents for miners. In fact, data cachers retrieve input data from the data source or other data cachers, forward data to miners and store data locally to serve miners directly in the future.

The super-peer network allows the queries issued by miners to rapidly explore the network. The super-peer approach is chosen to let the system support several public computing applications concurrently, without requiring that miners know in advance the location of the job manager and/or of the data cachers and the data source. Super-peers are used as rendezvous points that match job queries issued by miners with job adverts generated by the job manager.

The algorithm is illustrated with reference to Fig. 1. Firstly, the job manager partitions the data mining application in a set of tasks that can be executed in parallel. For each task, a "job advert" specifies the characteristics of the task to be executed and the related input data. An available miner issues a "job query" message to retrieve one of these job adverts. If the miner knows the location of the job manager, it delivers the query directly to it. If the location

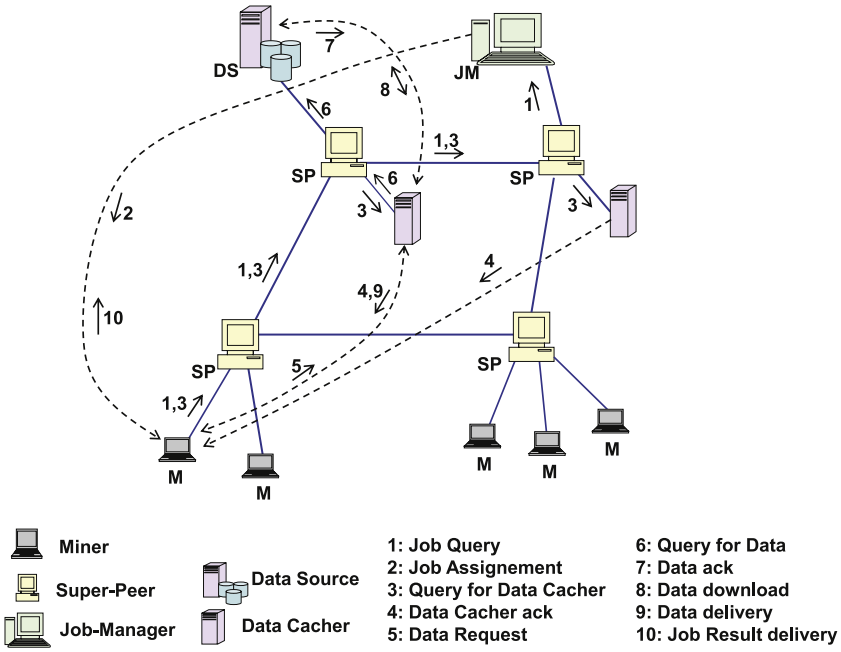

Fig. 1. Architecture of Mining@home.

of the job manager is unknown, the job query travels the network through the super-peer interconnections (messages labeled with number 1 in the figure). When a job advert is found that matches the job query, the related job is assigned to the miner (message 2 in the figure). The miner is also informed, through the job advert, about the data that it needs to execute the job. The required input data can be the entire dataset stored in the data source, or a subset of it.

The miner does not download data directly from the data source, but issues a query to discover a data cacher (message 3). This query can find several data cachers, each of which sends an ack to the miner (message 4). After a short time interval, the miner selects the most convenient data cacher according to a given strategy (message 5), and delegates to it the responsibility of retrieving the required data. The data cacher issues a "data request" (message 6) to discover the data source or another data cacher that has already downloaded the needed data. The data cacher receives a number of acks from available data cachers (message 7), downloads the data from one of those (message 8), stores the data, and forwards it to the miner (message 9). Now the miner executes the task and, at its completion, sends the results to the job manager (message 10).

The algorithm can be used in the general case in which the location of job manager, data source and data cachers is unknown to miners and other data cachers. For example, an applicative scenario, examined in [10], is represented by an environmental sensor network where big quantities of complex data (humidity, temperature, soil moisture, leaf wetness, solar radiation, etc.) are available and constitute a valuable source of information to be exploited for a better understanding of natural phenomena. Such kind of networks are highly dynamic, because some sensor nodes can be switched on/off (for energy or environmental reasons), others can kept some storage or computational functionalities, etc. In this context, the location of data sources, data cachers and miners is unknown a-priori and they can be dynamically connected/disconnected to the network over the time. The super-peer communication approach described here can be chosen to run several public computing applications concurrently, even in such a dynamic environment. In more static scenarios the algorithm can be simplified. Specifically, if the location of data source and data cachers are known, a job query (message 1) can be delivered directly to the job manager, instead of traveling the network, and messages 3–4 and 6–7 become unnecessary. Such simplifications are adopted for the experimental evaluation discussed in Section 4.

The Mining@Home prototype has been implemented in Java, JDK 1.7. As depicted in Fig. 1, the framework is built upon five types of nodes: job manager, data source, data cacher, super-peer and miner. Each node is multi-threaded, so that all tasks (send/receive messages, retrieve data, computation) are executed concurrently. Each miner exploits a *Mining Algorithm Library*, which contains the algorithms corresponding to the mining tasks.

3. Ensemble learning and bagging

Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. In contrast to ordinary machine learning approaches, which learn a single model from training data, ensemble methods build a set of models and combine them to obtain the final model. In a classification scenario, an ensemble method constructs a set of *base classifiers* from training data and performs classification by taking a vote on the predictions made by each classifier. As proved by mathematical analysis, ensemble classifiers tend to perform better (in terms of error rate) than any single classifier [11]. The basic idea is to build multiple classifiers from the original data and then aggregate their predictions when classifying unknown examples.

Bagging, also known as "bootstrap aggregating", is a popular ensemble learning technique [12]. Multiple training sets, or *bootstrap samples*, are sampled from the original dataset. The samples are used to train N different classifiers, and a test instance is labeled by the class that receives the highest number of votes by the classifiers. A logical view of the bagging method is shown in Fig. 2. Each bootstrap sample has the same size as the original dataset. Since sampling is done with replacement, some instances may appear several times in the same bootstrap sample, while others may not be present at all. On average, a bootstrap sample D_i contains approximatively 63% of the original training data. In fact, if the original dataset contains n instances, the probability that a specific instance is sampled at least once is: $1 - (1 - 1/n)^n \rightarrow 1 - 1/e \simeq 0.631$, where the approximation is valid for large values

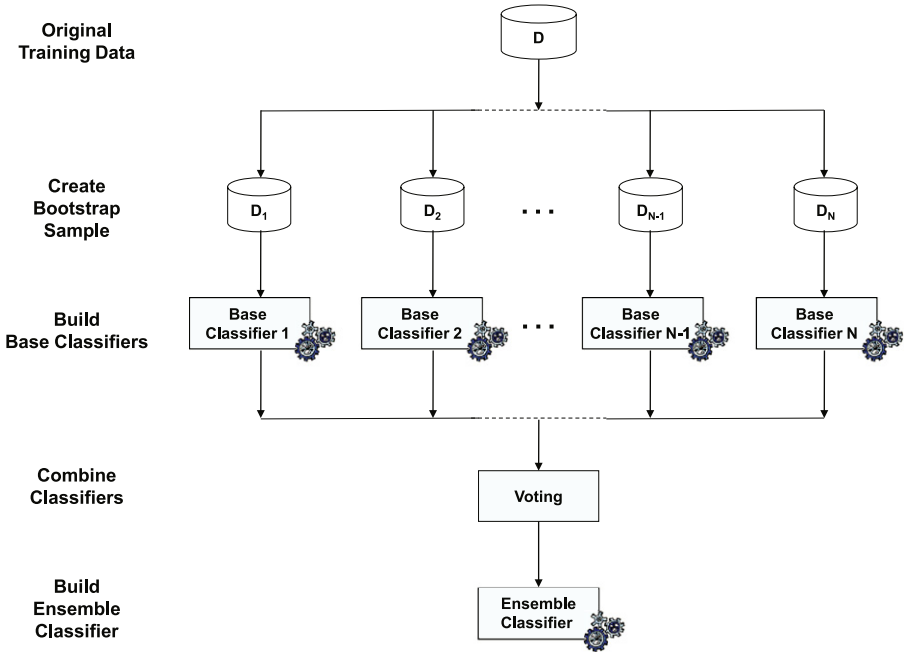

Fig. 2. A logical view of the bagging technique.

of n. This implies that two different samples share, on average, about $0.631 \cdot 0.631 \approx 40\%$ of the n instances.

An application implementing the bagging technique can naturally exploit the MINING@HOME system. The described scenario matches the two main conditions that must hold in order to profitably exploit the features of MINING@HOME:

1. *Base classifiers are independent*. Each base classifier can be mined independently from each other. Thus, it is possible to have a list of mining tasks to be executed, each one described by a distinct job descriptor. This fits the MINING@HOME architecture: each available miner takes the task of building one base classifier from a bootstrap sample of data, and at the end of execution transmits the discovered classification model to the job manager. Then, the miner may give its availability for a new job.

2. *Data can be re-used*. As mentioned before, in general different jobs need overlapping portions of input data, which is the rationale for the presence of distributed cache servers. After being assigned a job, the miner asks the input data to the closest data cacher, which may have already downloaded some of this data to serve previous requests. The data cacher retrieves only the missing data from the data source, and then sends the complete bootstrap sample to the miner. Of course, this leads to save network traffic and to a quicker response from the data cacher.

4. Experimental evaluation

The performance of the MINING@HOME framework has been evaluated on a classification problem tackled with the bagging technique. We deployed the framework in a real network composed of four computing domains connected through a Wide Area Network, as depicted in Fig. 3. Eight miners per domain are available, each hosted by a different node. Moreover, at each domain a further node is used to host both the super-peer and the data cacher. The node hosting the data source belongs to one of the domains, but an artificial delay has been introduced to simulate the scenario in which the data source is external to the domains.

Each node runs an Intel Pentium 4 processor with CPU frequency 1.36 GHz and 2 GB RAM. The inter-domain transfer rate between the data source and each of the domains (more precisely, the super-peers) is about 4 MB/s, while the intra-domain transfer rate ranges between 8.2 MB/s and 8.7 MB/s. The experiments were performed in a scenario where the job manager builds 128 base classifiers, by exploiting the bagging technique, on a transactional dataset D. The application proceeds as follows. The job manager builds a *job list*, containing the descriptions of the jobs that must be assigned to available miners. Each job is a request of building a J48 base classifier from a specific bootstrap sample. When all the jobs are executed, the job manager collects the extracted base classifiers and combines them to produce the final ensemble classifier.

The input dataset D is a subset of the *kddcup99* dataset.[1] The dataset, used for the KDD'99 Competition, contains a wide amount of data produced during seven weeks of monitoring in a military network environment subject to simulated intrusions. More in detail, it is composed of 5 million transactions, for a total size of 709 MB. Originally, it was constructed from a simulation performed by the Defense Advanced Research Projects Agency (DARPA) and was released for a classifier learning contest. The dataset is very compute-demanding due to both its size and a great inner variability among features. For such a reason, it is considered a valuable benchmark data to test the scalability of high performance pattern recognition tasks, and it has been exploited to validate the efficiency and effectiveness of several machine learning algorithms as well as distributed architectures (as in [13–16]).

Our evaluation followed two parallel avenues: we obtained experimental data for scenarios with 2–4 domains and 2 to 32 miners. Fig. 4 reports the turnaround time (from the time the application is started to the completion time of the last job) vs. the number of miners, ranging from 1 to 32, for different values of the dataset size $|D|$. The miners are equally distributed among the number of domains, except when only one or two miners are used. The results confirm the notable advantage deriving from the use of multiple miners. For example, with $|D|$ corresponding to 5 million transactions, the turnaround time decreases from 84 h (3.5 days), by using only one miner, to about 3 h when 32 miners are used. Fig. 5 reports the speedup measured in the same experiments. The scalability of the framework is confirmed both by the very

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

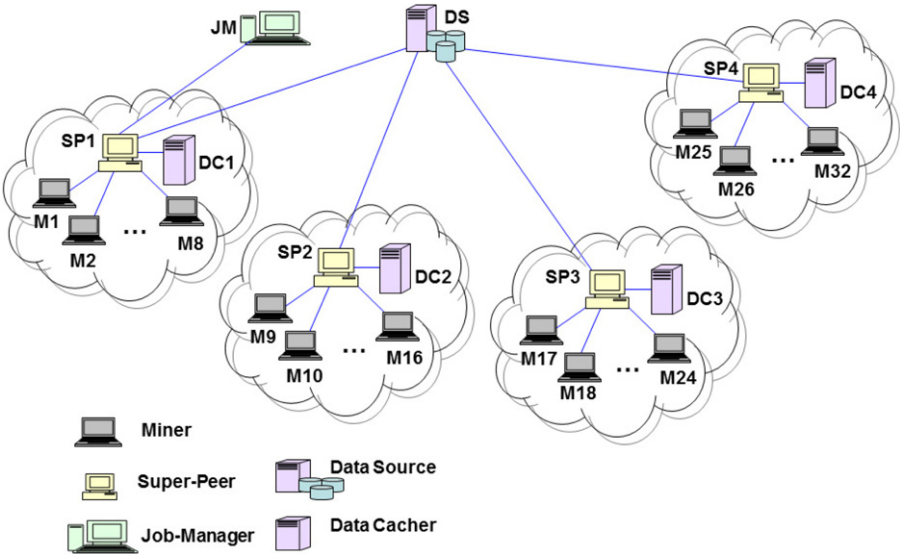

Fig. 3. Network architecture for the MINING@HOME experiments.

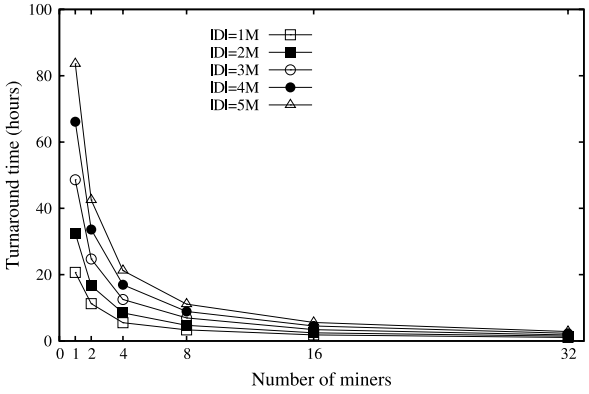

Fig. 4. Turnaround time vs. the number of miners evenly distributed among four domains, for different data sizes. Experimental results.

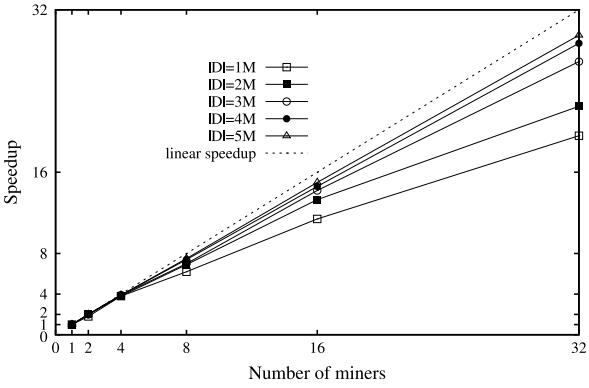

Fig. 5. Speedup vs. the number of miners evenly distributed among four domains, for different data sizes. Experimental results.

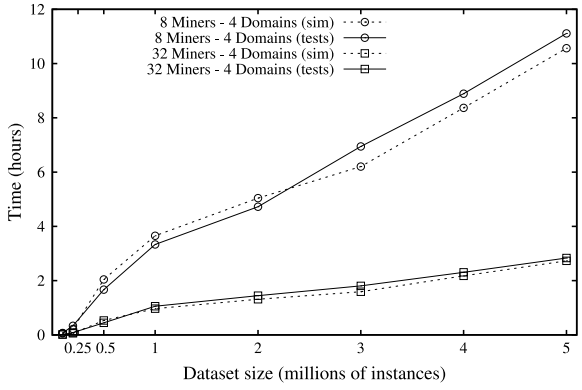

Fig. 6. Turnaround time vs. the size of the dataset, using 8, 32 miners evenly distributed among four domains. Experimental and simulation results are compared .

good values of speedup – optimum/linear speedup is reported as a reference – and by the fact that the speedup increases with the size of the problem, here represented by the dataset size.

For wider scenarios, we performed simulation tests using an ad hoc event-based simulator written in Java. An event queue is used to exchange messages and data among Java objects associated with the system components, i.e., super-peers, miners, data source and data cachers. To validate the simulator we first assessed simulation results by comparing them with experimental ones. Moreover, the simulator uses some data obtained in real experiments, specifically, the sequential execution time of jobs for any dataset size, the intra-domain and inter-domain transfer rates and the time needed to extract the final model. This data are used as parameters of the simulator to obtain the behavior of the system on a larger scale. Fig. 6 reports the turnaround time vs. the dataset size $|D|$, with 8 and 32 miners evenly distributed among the four domains. The figure shows that the simulator results are very close to those obtained from using the fully implemented system. This result validates the simulator and allowed us to use it to analyze larger scenarios, with the number of domains N_D ranging from 2 to 32, and the number of miners N_M ranging from 2 to 128. Accordingly, the results reported in the following were obtained through simulation unless otherwise stated.

Results shown in Figs. 7 and 8 prove the ability of the framework to scale with the system size. The experiments were performed varying the number of available miners and for different numbers of domains, from 1 to 32, while the dataset size $|D|$ is set to 4 millions instances. In multiple domain scenarios the miners are equally partitioned among the domains. The values of the turnaround time are reported in Fig. 7, using a log scale due to the wide range of obtained values. The turnaround time needed to process 4 million instances decreases from about 66 h (about 4000 min) with one miner to about 40 min with 128 miners. This

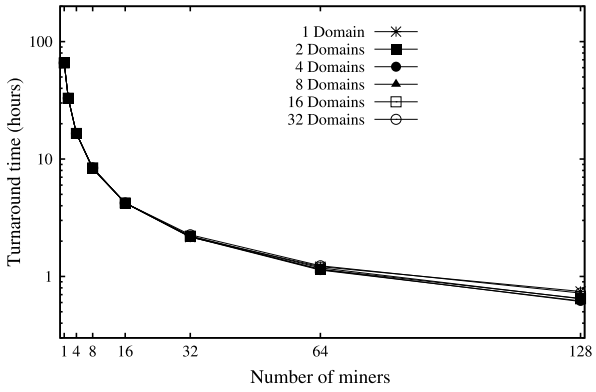

Fig. 7. Turnaround time vs. the number of available miners, for different numbers of domains (dataset size = 4 millions instances).

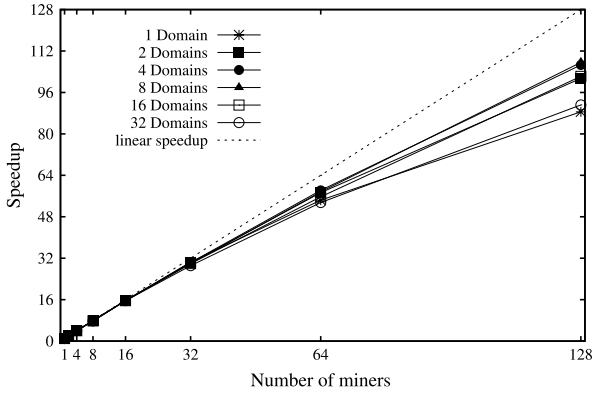

Fig. 8. Speedup vs. the number of available miners, for different numbers of domains (dataset size = 4 millions instances).

shows that using a large number of miners is highly efficient when the problem is big. This is an index of good scalability properties, since scalable systems can be defined as those for which the number of workers that optimizes the performance increases with the problem size [17].

Fig. 8 reports the values of speedup—defined as the ratio of the turnaround time obtained with a single node to the turnaround time computed with n nodes. Up to 32 nodes, the trend is very close to the optimal one (i.e., linear, shown for reference). When the number of nodes is larger (64 or 128), the slope of the curve begins to decrease since the overhead time, mostly related to communications among the miners, becomes relevant with respect to the processing time. When using 64 or 128 miners, we also notice a non-negligible impact of the number of domains on which the miners are distributed. However, speedup obtained on 128 nodes is about 100, corresponding to an efficiency value equal to 0.8. The influence of the number of domains is discussed with more details in the following section.

4.1. Efficiency evaluation

To gain a deeper understanding of the system performance, we used the "iso-efficiency" [17] methodology commonly adopted for the analysis of parallel and distributed algorithms. Let T_s be the *sequential execution time*, i.e., the time needed by a single miner to execute all the mining jobs sequentially, and T_o the *total overhead time* (mostly due to data transfers) experienced when the jobs are distributed among multiple miners. The total time spent by all the processors to complete the application can be expressed as:

$$nT_p = T_s + T_o \tag{1}$$

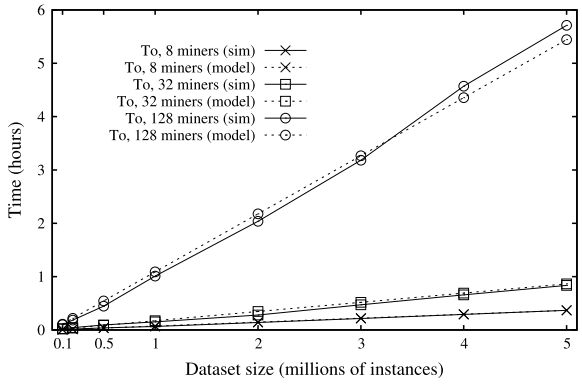

Fig. 9. Overhead time T_o vs. the size of the dataset, using 8, 32 and 128 miners, evenly distributed among four domains. Simulation results are compared to the theoretical prediction.

in which n is the number of miners and T_p is the parallel execution time when n miners are used in parallel. The speedup S – defined as the ratio between T_s and T_p – is then $S = \frac{T_s}{(T_s+T_o)/n} = \frac{nT_s}{T_s+T_o}$, and the efficiency E – defined as the ratio between the speedup and the number of miners – can be expressed as $E = \frac{1}{1+T_o/T_s}$. Therefore, the efficiency of the parallel computation is a function of the ratio between T_o and T_s: the lower this ratio, the higher the efficiency.

Let us start examining T_o. The total overhead time comprises the time needed to transfer data from the data source to data cachers and from these to miners. Both types of download are composed of a start up time (needed to open the connection and start the data transfer) and a time that is proportional to the amount of transferred data. Start up times are negligible with respect to the actual transfer time, therefore an approximation for T_o, in a scenario with N_D domains, is:

$$T_o = \frac{|D|}{R_{DS}} \cdot N_D + \sum_{i=1}^{N_D} \left(\frac{f \cdot |D|}{R_i} \cdot N_i \right) \tag{2}$$

where $|D|$ is the input dataset size, R_{DS} is the average rate at which data is downloaded from the data source to a data cacher, R_i is the download rate from a data cacher to the local miners within the i-th domain, and N_i is the number of jobs assigned to the i-th domain. The expression of the first term derives from the necessity of delivering the whole dataset, in successive data transfers, to all the data cachers. For any of the N_i jobs that are executed under domain i, a fraction f (with $f \simeq 0.63$) of the dataset is downloaded by the miner from the local data cacher, which explains the second term.

In the following, the impact of the dataset size, the number of domains and the number of miners on the values of T_o are discussed. The overhead time increases linearly with the dataset size $|D|$. This is clearly visible in Fig. 9, which reports the values of T_o versus the dataset size, obtained with 8, 32 and 128 miners. The figure compares the simulation values to those obtained with expression (2) and shows that the gap between analytical and simulation results is small for any value of $|D|$.

The number of domains N_D influences the two terms of expression (2) in different ways. The first term is explicitly proportional to N_D. Conversely, the second term is inversely proportional to N_D in an implicit fashion: when a given number of miners are distributed on a larger number of domains, each data cacher must serve a lower number of local miners. As a consequence, the intra-domain transfer rate R_i is higher – and the value of the second term is lower – when the number of domains increases. Finally, let us examine the impact of the number of miners on T_o. When more miners request data from the local data cacher at the same time, the intra-domain transfer rate R_i tends to decrease, because the uplink bandwidth of

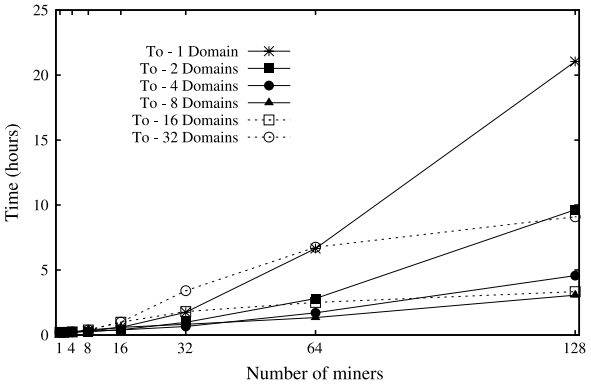

Fig. 10. Overhead time T_o vs. the number of available miners, for different numbers of domains (dataset size = 4 millions instances).

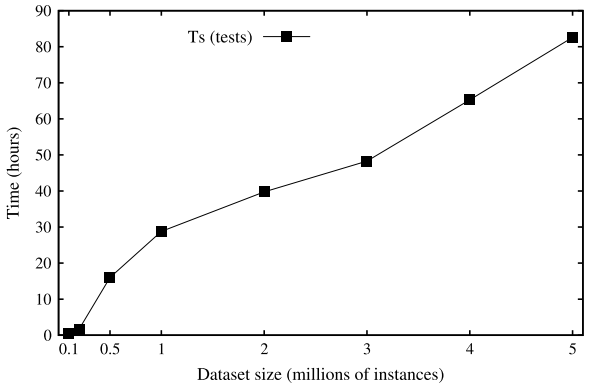

Fig. 11. Sequential time T_s vs. the size of the dataset.

the data cacher is shared among multiple data connections. Since the frequency of concurrent downloads increases when more miners are active in the same domain, the overall value of T_o increases with the number of miners.

Fig. 10 shows how the overhead time depends on the number of miners and on the number of domains, when 4 millions instances are processed. The considerations made above are confirmed. In particular: (i) the overhead time increases with the number of miners, but the increase rate is lower when the miners are distributed on a larger number of domains; (ii) due to the opposite relationships of the two terms of expression (2) with respect to $|D|$, it is possible to optimize the number of domains: for example, with 128 miners, the overhead time is reduced when distributing the miners to a number of domains comprised between 4 and 16.

As opposed to T_o, the sequential time T_s does not depend neither on the number of domains or on the number of miners, as it is simply the sum of job computation times. Of course, T_s does depend on $|D|$, because the length of each job increases with the size of input data. Fig. 11 reports the value of the sequential time vs. the dataset size, computed experimentally and averaged over 20 runs.

To help analyze the efficiency, the values of T_s and T_o, measured in a scenario with four domains have been plot together in Fig. 12, using a log scale due to the wide range of reported values. The three curves for T_o are obtained in the cases that 8, 32 and 128 miners are used to execute the 128 jobs. As the figure scale is logarithmic, the ratio T_o/T_s – which determines the value of efficiency – can be evaluated as the distance, on the logarithmic plot, between the curves corresponding to T_o and T_s. In Fig. 12 it is interesting to notice that the ratio T_o/T_s notably decreases in the first part the curve (i.e., the distance between the corresponding curves increases), which is a sign that the efficiency of the architecture

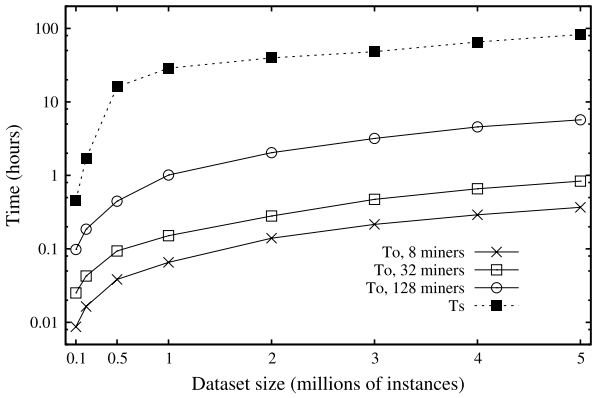

Fig. 12. Sequential time T_s and overhead time T_o vs. the size of the dataset, using 8, 32 and 128 miners, evenly distributed among four domains.

improves with the problem size. However, when the number of instances is higher than two millions instances, the gap between T_s and T_o, as well as the efficiency, becomes more stable.

Figs. 13 and 14 report the relative weighs of T_s and T_o when using, respectively, 32 and 128 miners. Each figure shows the results obtained with different values of the dataset size $|D|$ (from 0.1 to 5 millions instances) and different numbers of domains (from 1 to 32). The main conclusions that can be derived from these plots are:

- the weight of the overhead time is always relatively low – therefore, the efficiency is high – except a few circumstances (e.g., when processing a dataset size of 100,000 instances and 128 miners are concentrated in only one domain). This is essential to justify the adoption of a distributed solution: if the overhead time were predominant, the benefit derived from the parallelization of work would not compensate the extra time needed to transfer data to remote data cachers and miners.
- the efficiency is lower when using more miners, due to the impact of concurrent downloads of multiple miners from the same data cachers. In fact, the relative weight of T_o is higher in Fig. 14 with respect to Fig. 13. However, using more miners is still very useful because it allows the value of turnaround time to be significantly reduced, as shown in Figs. 7 and 8;
- the efficiency depends on the number of domains on which the miners are distributed. Using from 4 to 8 domains is a good choice in all the considered scenarios.

The last consideration is better highlighted in Figs. 15 and 16, which report the turnaround time and the efficiency versus the number of domains, when using 128 miners. Specifically, Fig. 15 shows that the optimal distribution of miners among the domains is particularly relevant when the problem is bigger. In Fig. 16 it is appreciated that the efficiency is always higher than 0.68 and reaches values up to 0.85 when correctly combining the number of miners and the number of domains.

5. Related work

The approach we used in the implementation of the distributed data mining framework we are discussing inherits methodologies from two specific fields, peer-to-peer (P2P) computing and public resource computing. The following describes some state-of-the art about current contributions of these two fields to data mining analysis.

P2P data mining is emerging as a new distributed computing paradigm for many novel applications, where storage of data and mining analysis are performed by exploiting a large number of peers with little or no centralized coordination. Even though it

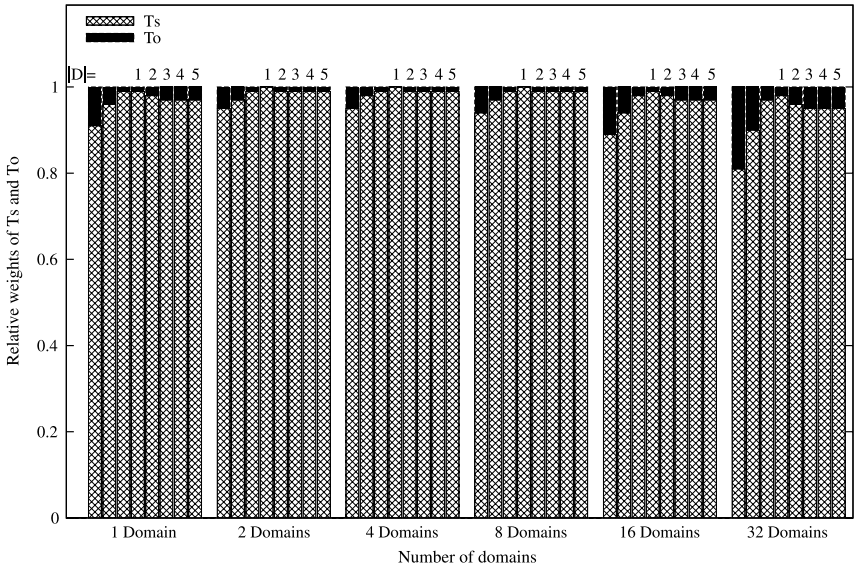

Fig. 13. Relative weights of T_s and T_o vs. the number of domains, using 32 miners, and for different values of $|D|$, expressed in millions instances. The last 5 values are reported on top of the bars. The first 3 are 0.1, 0.2 and 0.5.

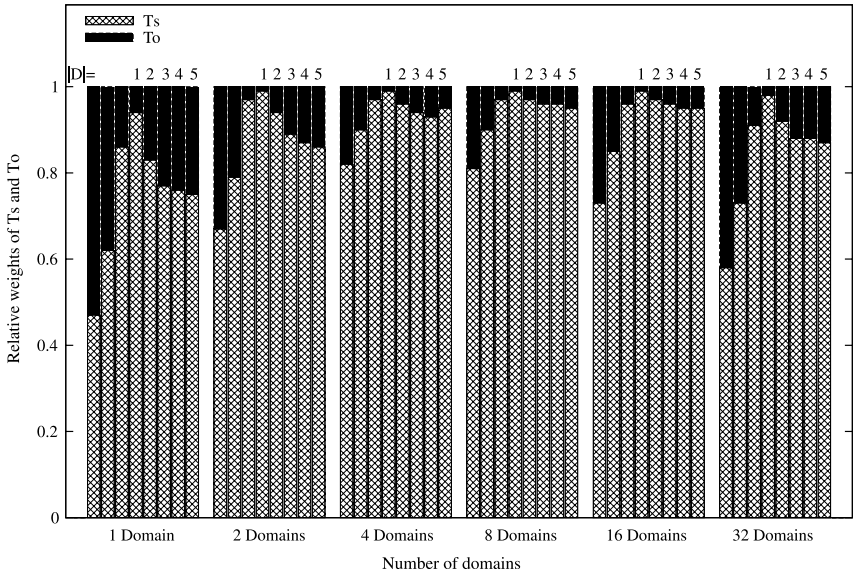

Fig. 14. Relative weights of T_s and T_o vs. the number of domains, using 128 miners, and for different values of $|D|$, expressed in millions instances. The last 5 values are reported on top of the bars. The first 3 are 0.1, 0.2 and 0.5.

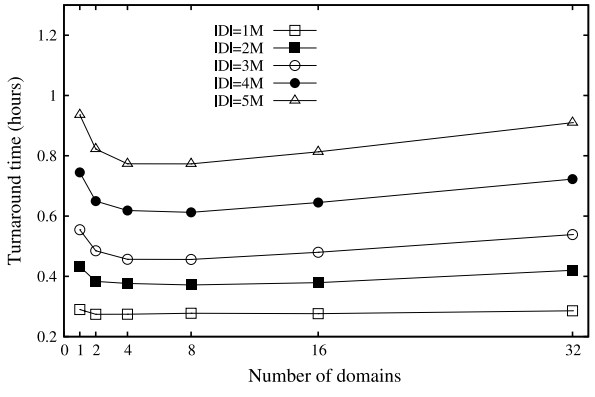

Fig. 15. Turnaround time vs. the number of domains, for different dataset sizes, using 128 miners.

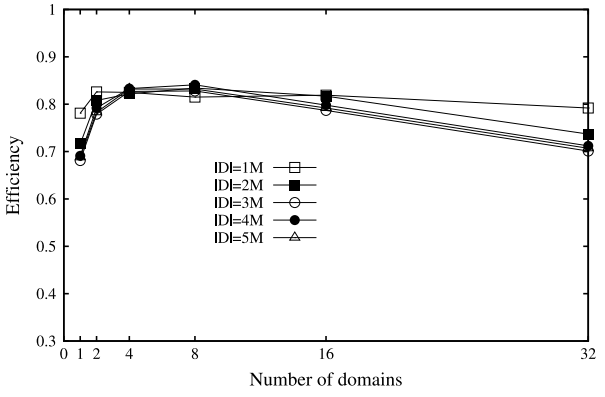

Fig. 16. Efficiency vs. the number of domains, for different dataset sizes, using 128 miners.

is a recent research area, some interesting contributions have been published in the last years. An important challenge is that standard centralized algorithms can be communication-expensive and impractical because of synchronization requirements. In [1], a scalable and robust distributed algorithm for decision tree induction in large P2P environments is presented. In order to achieve good scalability in a distributed environment, the proposed technique works in a completely asynchronous manner and offers low communication overhead. In [18] an approach for mining association rules is proposed for a scenario where databases are partitioned in a P2P computer network. The algorithm combines sequential association rule mining, executed locally at each node, with a majority voting protocol that discover the association rules valid for the combined database. In particular, each node in the system can discover a subset of rules by using information gathered from its neighborhood. This locality property ensures good scalability in very large networks.

In [19], the problem of keeping a mining model up-to-date in P2P networks is analyzed. This is an important issue in systems, like sensor networks, which generate and process huge amounts of data every day, and therefore are highly time varying. As an alternative to the simple but inefficient solution of periodically recomputing the model, the authors propose a reactive approach. Specifically, a highly efficient local algorithm detects when the L2 norm of the data exceeds a given threshold and uses it as a feedback loop for monitoring complex predicates on the data. As soon as the L2 algorithm detects that the current model no longer represents the data, the model is rebuilt. Experiments with the k-means algorithm are claimed to achieve good accuracy with costs that, on stationary periods, are negligible.

The issue of analyzing multivariate regression in large P2P environments is tackled in [20]. In particular, the authors propose an efficient distributed algorithm that monitors the quality of the current regression model, and rebuilds the model if it appears to be outdated. The algorithm, despite its distributed nature, exploits only local information, which guarantees low monitoring cost, as pointed out by experimental results. The algorithm has been designed for distributed inferencing, data compaction, data modeling and classification tasks for many distributed domains, like bioinformatics, astronomy, social networking, sensor networks and web mining.

In [21] the authors propose a sampling-based technique for approximate answering of ad-hoc aggregation queries in P2P databases. In particular, they deal with the problem of computing a high-quality random sample of a P2P database, an issue complicated by three main factors: (i) data is distributed and unfairly partitioned among a large number of peers; (ii) within each peer the data is often highly correlated, and may not be representative of the whole distributed database and, (iii) even collecting a random sample from the peers is a difficult task. To counter these problems, an adaptive two-phase sampling approach, based on random walks over the P2P network and on block-level sampling techniques, has been proposed and extensively experimented in different scenarios. In [22], the authors describe a technique for the execution of top-k retrieval tasks in P2P information infrastructures. To this purpose, an algorithm based on the collection of query statistics is proposed. In particular, in order to minimize communication cost, the algorithm exploits local indexing to optimize the necessary query routing and process intermediate results in inner network nodes. Experimental evaluation shows that the technique scales well to large numbers of peers, and efficiently adapts to dynamic additions/deletions of peers.

The authors of [23] cope with the problem of learning from multiple information sources, both labeled and unlabeled, which cannot be integrated into a single information source. They introduce two propagation methods to label a set of training objects in unlabeled sources, in accordance with the class label information extracted from labeled sources and with the internal structure information of unlabeled sources. Then they use the ensemble learning model, as in our case, to predict the labels of test objects. In [24] a sentiment analysis on Twitter is performed using an ensemble learning approach. In particular, first the ensemble classifiers are trained using supervised and semi-supervised learning; second, sentiment lexicons and bag-of-words are combined for the comparison and, finally, lexicons and bag-of-words are used in supervised and semi-supervised data contexts.

So far, the research ares of Distributed Data Mining and volunteer computing have experienced little integration. Volunteer computing is a form of network based distributed computing, which allows public participants to share their idle computing resources, and helps run computationally expensive projects. The volunteer computing [3] paradigm has been exploited in several scientific applications (i.e., Seti@home, Folding@home, Einstein@home), but its adoption for mining applications is more challenging. The two most popular volunteer computing platforms available today, *BOINC* [4] and *XtremWeb* [5,6], are especially well suited for CPU-intensive applications but are somewhat inappropriate for data-intensive tasks, for two main reasons. First, the centralized nature of such systems requires all data to be served by a group of centrally maintained servers. Consequently, any server in charge of job assignment and data distribution is a clear bottleneck and a single point of failure for the system. For example, the BOINC task server, described and analyzed in [25], partitions the work into multiple tasks, dispatches them to clients, and processes the returned results. Second, the client/server data distribution scheme does not offer valuable solutions for applications in which input data files can be initially stored in distributed locations or may be reused by different workers. A recent survey paper on volunteer computing [26] confirms that data management is among the main open problems of volunteer computing and solicits future research work to design and implement frameworks that can efficiently integrate data coming from heterogeneous sources and handle partial data results.

Some approaches to overcome such limitations have been recently proposed. In [27] the authors analyze, through a simulation framework, a volunteer computing approach that exploits decentralized P2P data sharing practices. The application scenario discussed in [27] concerns the analysis of gravitational waveforms for the discovery of user specified patterns that may correspond to orbiting neutron stars. In [7], an approach inspired by the volunteer paradigm was devised to cope with the problem of identifying closed frequent itemsets in a transactional dataset. The functionality and performance of the framework were evaluated through a simulation study, tailored to that specific application domain. After that the framework, named MINING@HOME, was fully implemented and made capable of coping with a number of different data analysis scenarios involving the execution of data mining tasks in a distributed environment. To the best of our knowledge, MINING@HOME is the first fully implemented public resource computing framework that executes data mining tasks over distributed data. In particular, its data-oriented architecture as well as its protocols and algorithms are general and can be easily adapted to a wide set of distributed data mining problems.

The authors of [28] analyze the possible future integrations of the volunteer computing and crowdsourcing paradigms. Indeed, volunteer computing can be considered as a form of crowdsourcing, in which solving various classes of problems, for example decision making problems, requires contributions to a large group of people, and each contributor adds a small portion to the overall result.

More in general, several distributed data mining algorithms and systems have been proposed. In [1], a scalable and robust

distributed algorithm for decision tree induction in distributed environments is presented. In order to achieve good scalability in a distributed environment, the proposed technique works in a completely asynchronous manner and offers low communication overhead. A distributed meta-learning technique is proposed in [29], where knowledge probing is used to extract descriptive knowledge from a black box model, such as a neural network. In particular, probing data is generated using various methods such as uniform voting, trained predictor, likelihood combination, etc. Differently from the classical meta-learning, the final classifier is learned from the probing data.

6. Conclusion

This paper discussed how MINING@HOME, a tool that exploits the public resource paradigm to implement large-scale data mining applications in a peer-to-peer infrastructure, can be used to run complex data analysis applications based on ensemble learning, a machine learning paradigm where multiple models are trained to solve the same problem and achieve better statistical accuracy. MINING@HOME has been executed on a real deployment in which a classifier model is extracted from transactional datasets. Performance results have been obtained in a distributed scenario where up to 32 miners have been run on four interconnected domains. Simulation experiments have been performed to extend the evaluation to much larger scenarios and networks. Furthermore, a model based on the iso-efficiency methodology has been used to investigate speedup and efficiency metrics and give a mathematical foundation to experimental results. The results confirmed the applicability of the framework as well as its efficiency and scalability. Moreover, the flexibility of MINING@HOME offers the user the possibility of optimizing the performance by choosing the appropriate system and network configuration, for example, by tuning the number of workers and the number of domains on which the workers are distributed.

Acknowledgments

This research work has been partially funded by the MIUR projects DICET-INMOTO (PON04a2_D) and DOMUS (PON03PE_00050_2).

References

[1] Kanishka Bhaduri, Ran Wolff, Chris Giannella, Hillol Kargupta, Distributed decision-tree induction in peer-to-peer systems, Stat. Anal. Data Min. 1 (2) (2008) 85–103.
[2] Eugenio Cesario, Domenico Talia, Distributed data mining patterns and services: An architecture and experiments, Concurr. Comput.: Pract. Exper. 24 (15) (2012) 1751–1774.
[3] David P. Anderson, Public computing: Reconnecting people to science, in: Proceedings of Conference on Shared Knowledge and the Web, Madrid, Spain, November 2003, pp. 17–19.
[4] David P. Anderson, BOINC: A system for public-resource computing and storage, in: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, GRID'04, 2004, pp. 4–10.
[5] Franck Cappello, Samir Djilali, Gilles Fedak, Thomas Herault, Frederic Magniette, Vincent Neri, Oleg Lodygensky, Computing on large-scale distributed systems: Xtrem web architecture, programming models, security, tests and convergence with grid, Future Gener. Comput. Syst. 21 (3) (2005) 417–437.
[6] Gilles Fedak, Cecile Germain, Vincent Neri, Franck Cappello, XtremWeb: A generic global computing system, in: Proceedings of the IEEE Int. Symp. on Cluster Computing and the Grid, Brisbane, Australia, May 2001.
[7] Claudio Lucchese, Carlo Mastroianni, Salvatore Orlando, Domenico Talia, Mining@home: Towards a public resource computing framework for distributed data mining, Concurr. Comput.: Pract. Exper. 22 (5) (2010) 658–682.
[8] Eugenio Cesario, Carlo Mastroianni, Domenico Talia, Using Mining@home for distributed ensemble learning, in: Proc. of the 5th International Conference on Data Management in Cloud, Grid and P2P Systems, Globe 2012, Wien, Austria, September 2012.
[9] Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[10] Eugenio Cesario, Domenico Talia, Mining frequent items and itemsets from distributed data streams for emergency detection and management, in: Proceedings of the PROACTIVE Workshop on Models and Architectures for Emergency Management, Milan, Italy, pp. 34–35.
[11] P.N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Pearson International Edition, 2006.
[12] Leo Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[13] Jianjun Xie, Viktoria Rojkova, Siddharth Pal, Stephen Coggeshall, A combination of boosting and bagging for KDD cup 2009—fast scoring on a large database, in: Proceedings of KDD-Cup 2009 Competition, Paris, France, June 28, 2009, 2009, pp. 35–43.
[14] Iago Porto-Díaz, David Martínez-Rego, Amparo Alonso-Betanzos, Oscar Fontenla-Romero, Combining feature selection and local modelling in the kdd cup 99 dataset, in: Cesare Alippi, Marios Polycarpou, Christos Panayiotou, Georgios Ellinas (Eds.), Artificial Neural Networks, ICANN 2009, in: Lecture Notes in Computer Science, vol. 5768, Springer, Berlin, Heidelberg, 2009, pp. 824–833.
[15] Mohammad Khubeb Siddiqui, Shams Naahid, Analysis of kdd cup 99 dataset using clustering based data mining, Int. J. Database Theory Appl. 6 (5) (2013) 23–34.
[16] Eugenio Cesario, Domenico Talia, Distributed data mining patterns and services: an architecture and experiments, Concurr. Comput.: Pract. Exper. 24 (15) (2012) 1751–1774.
[17] Ananth Y. Grama, Anshul Gupta, Vipin Kumar, Isoefficiency: Measuring the scalability of parallel algorithms and architectures, IEEE Concurr. 1 (1993) 12–21.
[18] Ran Wolff, Assaf Schuster, Association rule mining in peer-to-peer systems, IEEE Trans. Syst. Man Cybern. B 34 (6) (2004).
[19] K. Bhaduri, R. Wolff, H. Kargupta, Local l2-thresholding based data mining in peer-to-peer systems, in: SIAM'06: Proceedings of the 2006 SIAM Conference Data Mining, 2006, pp. 430–441.
[20] Kanishka Bhaduri, Hillol Kargupta, An efficient local algorithm for distributed multivariate regression in peer-to-peer networks, in: Proceedings of the SIAM International Conference on Data Mining, SDM 2008, 2008, pp. 153–164.
[21] Benjamin Arai, Gautam Das, Dimitrios Gunopulos, Vana Kalogeraki, Efficient approximate query processing in peer-to-peer networks, IEEE Trans. Knowl. Data Eng. 19 (7) (2007) 919–933.
[22] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, Uwe Thaden, Progressive distributed top-k retrieval in peer-to-peer networks, in: Proceedings of the 21st International Conference on Data Engineering, ICDE'05, April 2005.
[23] Yaojin Lin, Xuegang Hu, Xindong Wu, Ensemble learning from multiple information sources via label propagation and consensus, Appl. Intell. 41 (1) (2014).
[24] Tawunrat Chalothom, Jeremy Ellman, Simple approaches of sentiment analysis via ensemble learning, in: Kuinam J. Kim (Ed.), Information Science and Applications, in: Lecture Notes in Electrical Engineering, vol. 339, Springer, Berlin, Heidelberg, 2015, pp. 631–639.
[25] David P. Anderson, Eric Korpela, Rom Walton, High-performance task distribution for volunteer computing, in: Proc. of the First International Conference on e-Science and Grid Computing, Melbourne, Australia, December 2005, pp. 196–203.
[26] Muhammad Nouman Durrani, Jawwad A. Shamsi, Volunteer computing: requirements, challenges, and solutions, J. Netw. Comput. Appl. 39 (2014) 369–380.
[27] Carlo Mastroianni, Pasquale Cozza, Domenico Talia, Ian Kelley, Ian Taylor, A scalable super-peer approach for public scientific computation, Future Gener. Comput. Syst. 25 (3) (2009) 213–223.
[28] Jerzi Balicki, Piotr Brudlo, Piotr Szpryngier, Crowdsourcing and volunteer computing as distributed approach for problem solving, in: Proc. of the 13th International Conference on Software Engineering, Parallel and Distributed Systems, SEPADS'14, Gdansk, Poland, May 2014, pp. 115–121.
[29] Yike Guo, Janjao Sutiwaraphun, Probing knowledge in distributed data mining, in: Proceedings of the Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining, PAKDD'99, 1999, pp. 443–452.

Eugenio Cesario received the Laurea degree in Computer Science Engineering from the University of Calabria, Italy in 2002. From 2003 to 2006, he was a research fellow at the Institute of High-Performance Computing and Networks, National Research Council (ICAR-CNR), Italy, working on data mining systems and applications. In 2006, he received the Ph.D. degree in Systems and Computer Engineering from the University of Calabria. He has been a researcher at ICAR-CNR since 2007. He coauthored more than 40 papers published in international journals, including IEEE Transactions on Knowledge and Data Engineering, Concurrency and Computation, and conference proceedings. He served as a chair, organizer, or program committee member of several international conferences.

His research interests include Distributed Data Mining, Cloud and Grid services architectures, Knowledge Discovery applications.

Carlo Mastroianni received the Laurea degree and the Ph.D. degree in computer engineering from the University of Calabria, Italy, in 1995 and 1999, respectively. He has been a researcher at the Institute of High Performance Computing and Networking of the Italian National Research Council (ICAR-CNR) in Cosenza, Italy, since 2002. Previously, he worked at the Computer Department of the Prime Minister Office in Rome. He coauthored more than 100 papers published in international journals, including IEEE/ACM Transactions on Networking, IEEE Transactions on Evolutionary Computation and ACM Transactions on Autonomous and Adaptive Systems, and conference proceedings. He edited special issues for the journals Future Generation Computer Systems, Journal of Network and Computer Applications, Multiagent and Grid Systems. His research interests include Cloud and Grid computing, P2P, bio-inspired algorithms, and multiagent systems. He is a cofounder of the Eco4Cloud company (www.eco4cloud.com).

Domenico Talia is a full professor of computer engineering at the University of Calabria. He is a partner of two startups, Exeura and DtoKLab. His research interests include parallel and distributed data mining, cloud computing, grid services, knowledge discovery applications, mobile computing, peer-to-peer systems, and parallel computing paradigms. Talia published ten books and more than 300 papers in archival journals such as CACM, Computer, IEEE TKDE, IEEE TSE, IEEE TSMC-B, IEEE Micro, ACM Computing Surveys, FGCS, Parallel Computing, Internet Computing, and in international conference proceedings. He is a member of the editorial boards of IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, the Future Generation Computer Systems journal, the International Journal on Web and Grid Services, the Scalable Computing: Practice and Experience journal, MultiAgent and Grid Systems: An International Journal, International Journal of Web and Grid Services, and the Web Intelligence and Agent Systems International journal. Talia has been an evaluator for several international institutions such as the European Commission, Aeres in France, Austrian Science Fund, Croucher Foundation, and the Russian Federation Government. He served as a chair, organizer, or program committee member of several international conferences and gave many invited talks and seminars in conferences and schools. Talia is a member of the ACM and the IEEE Computer Society.