A Cloud Framework for Parameter Sweeping Data Mining Applications

Fabrizio Marozzo
DEIS, University of Calabria
Rende (CS), Italy
Email: fmarozzo@deis.unical.it

Domenico Talia
ICAR-CNR
DEIS, University of Calabria
Rende (CS), Italy
Email: talia@deis.unical.it

Paolo Trunfio
DEIS, University of Calabria
Rende (CS), Italy
Email: trunfio@deis.unical.it

***Abstract*—Data mining techniques are used in many application areas to extract useful knowledge from large datasets. Very often, parameter sweeping is used in data mining applications to explore the effects produced on the data analysis result by different values of the algorithm parameters. Parameter sweeping applications can be highly computing demanding, since the number of single tasks to be executed increases with the number of swept parameters and the range of their values. Cloud technologies can be effectively exploited to provide end-users with the computing and storage resources, and the execution mechanisms needed to efficiently run this class of applications. In this paper, we present a *Data Mining Cloud App* framework that supports the execution of parameter sweeping data mining applications on a Cloud. The framework has been implemented using the Windows Azure platform, and evaluated through a set of parameter sweeping clustering and classification applications. The experimental results demonstrate the effectiveness of the proposed framework, as well as the scalability that can be achieved through the parallel execution of parameter sweeping applications on a pool of virtual servers.**

***Keywords*-Data mining; Cloud computing; Parameter sweeping**

I. Introduction

The IT market has been moving from the demand and supply of products to a service-oriented model in which all resources - processors, memories, data and applications - are provided as services to customers through Internet. Such convergence between Internet technologies and services, combined with the use of virtualization techniques, has led to the development of *Cloud computing*. Thus, Cloud computing can be defined as a high-performance distributed computing paradigm in which all the resources, dynamically scalable and often virtualized, are provided as a service through Internet.

A key aspect of such paradigm is that end-users do not need to have neither knowledge nor control over the infrastructure that supports their applications. In fact, Cloud infrastructures are based on large sets of computing resources, located somewhere "in the Cloud," which are assigned to applications on-demand. Cloud resources are provided in highly scalable way, i.e., they are allocated dynamically to applications depending of the current level of requests. Although similar in overall aims to a Grid system, Clouds are different because hide the complexity of the underlying infrastructure, providing services ready to use where end-users pay only for the resources effectively used (pay-per-use).

Cloud computing vendors classify their services into three categories: *Infrastructure as a Service* (IaaS), also known as Cloud infrastructure services, provides the computing resources like CPUs, memory, and storage, for running virtualized systems over the Cloud (e.g., Amazon EC2, RackSpace Cloud); *Platform as a Service* (PaaS), also known as Cloud platform services, in which Cloud providers offer platform services such as databases, application servers, or environments for building, testing and running custom applications (e.g., Google Apps Engine, Microsoft Azure, Force.com); *Software as a Service* (SaaS), where each software or application executed is provided through Internet to customers as ready-to-use services (e.g., Google Calendar, Microsoft Hotmail, Yahoo Maps).

We adopted the SaaS approach to design a *Data Mining Cloud App* framework that supports the execution of data mining applications on a Cloud. The framework supports both *single applications* and *parameter sweeping applications*. In the first case, a single data mining task such as classification, clustering, or association rules discovery is performed on a given data source. In parameter sweeping applications, a single dataset is analyzed in parallel using multiple instances of the same data mining algorithm with different parameters.

Often data analysis applications need to run a data mining task several times by using different parameter values before getting significant results. This is a time consuming process that, if it is run sequentially, can require very long execution times. For this reason, parameter sweeping applications are widely used in data mining applications to explore in parallel the effects of using different values of algorithm parameters on the results of data analysis. Since the number of tasks to be executed increases with the number of swept parameters and the range of their values, parameter sweeping applications require a large amount of computing resources. To overcome this problem, the Data Mining Cloud App exploits Cloud technologies to provide end-users the computing resources and execution mechanisms needed to

efficiently run this class of applications.

The framework has been implemented using Windows Azure[1], a PaaS by Microsoft, and has been evaluated through a set of parameter sweeping clustering and classification applications executed on a Microsoft Cloud data center. The experimental results presented in this paper demonstrate the effectiveness of the Data Mining Cloud App framework, as well as the scalability that can be achieved through the parallel execution of parameter sweeping applications on a pool of virtual servers.

The remainder of this paper is organized as follows. Section II provides a short background on Windows Azure. Section III describes architecture, execution mechanisms and user interface of the Data Mining Cloud App framework. Section IV presents a performance evaluation. Section V discusses related work. Finally, Section VI concludes the paper and outlines future work.

II. Windows Azure

Azure is an environment and a set of Cloud services that can be used to develop Cloud-oriented applications, or to enhance existing applications with Cloud-based capabilities. The platform provides on-demand compute and storage resources exploiting the computational and storage power of the Microsoft data centers. Azure is designed for supporting high availability and dynamic scaling services that match user needs with a pay-per-use pricing model. The Azure platform can be used to perform the storage of large datasets, execute large volumes of batch computations, and develop SaaS applications targeted towards end-users. Windows Azure includes three basic components/services:

- *Compute* is the computational environment to execute Cloud applications. Each application is structured into roles: *Web role*, for Web-based applications; *Worker role*, for batch applications; *VM role*, for virtual-machine images.
- *Storage* provides scalable storage to manage: binary and text data (*Blobs*), non-relational tables (*Tables*), queues for asynchronous communication between components (*Queues*), and NTFS volume (*Drives*).
- *Fabric controller* whose aim is to build a network of interconnected nodes from the physical machines of a single data center. The Compute and Storage services are built on top of this component.

The Windows Azure platform provides standard interfaces that allow developers to interact with its services. Moreover, developers can use IDEs like Microsoft Visual Studio and Eclipse to easily design and publish Azure applications.

III. Data Mining Cloud App

In this section we describe architecture, execution mechanisms and user interface of the Data Mining Cloud App framework.

[1]http://www.microsoft.com/windowsazure

A. Architecture and Execution Mechanisms

Figure 1 shows the architecture of the Data Mining Cloud App framework, as it is implemented on Windows Azure. The framework includes the following components:

- A set of *Blobs* used to store data to be mined (*input datasets*) and the results of data mining tasks (*data mining models*).
- A *Task queue* that contains the data mining tasks to be executed.
- A *Task Status Table* that keeps information about the status of all tasks.
- A pool of k *Workers*, where k is the number of virtual servers available[2], which are in charge of executing the data mining tasks submitted by users.
- A *Website* that allows users to submit, monitor the execution, and access the results of the data mining tasks (interface and usage of this component will be described in Section III-B).

The first three components of the framework are implemented by exploiting the Azure Storage services, while the Azure Compute services are used to implement the Workers and the Website. In particular, each Worker is a Worker Role instance, while the Website is hosted on a Web Role instance.

The following steps are performed to execute a data mining application through the Data Mining Cloud App (see Figure 1):

1) The user accesses the Website and submits his/her data mining application, by specifying: location of the input dataset, name of the data mining algorithm, and values of its parameters.
2) The Website inserts a set of tasks into the Task Queue on the basis of the data mining application submitted by the user. If the user submitted a single data mining application, one data mining task is inserted into the Task Queue. If the user submitted a parameter sweeping application, one task for each combination of the input parameters values is created[3].
3) Each idle Worker picks a task from the Task Queue, and starts its execution on a virtual server.
4) Each Worker gets the input dataset from the location specified by the user. To this end, a file transfer is performed from the Blob where the dataset is located, to the local storage of the virtual server the Worker is running on.
5) After task completion, each Worker puts the result on a Blob.

[2]We assume here that each virtual server is equipped with a single CPU core; if there are m cores per virtual server, we can have up to $m \times k$ Workers.

[3]In general, the number of tasks is given by $\prod_{i=1}^{n} v_i$, where n is the number of input parameters and v_i is the number of values assumed by the i^{th} parameter

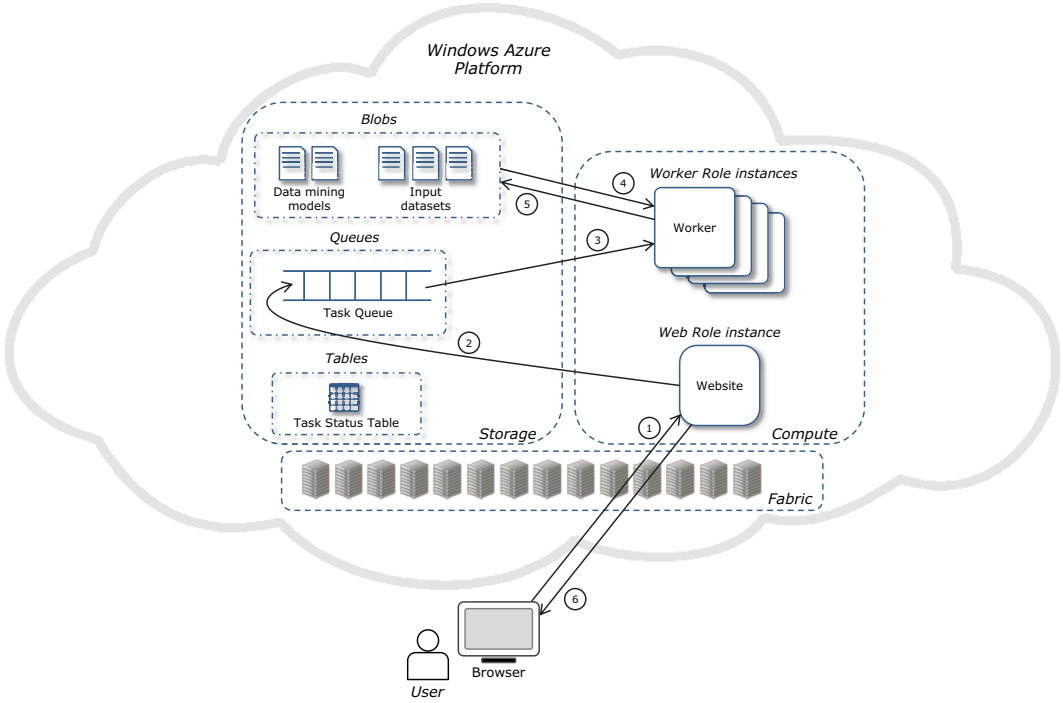


Figure 1. Architecture of the Data Mining Cloud App framework.

6) The Website notifies the user as soon as his/her task(s) have completed, and allows him/her to visualize the results.

Note that the Task Status Table is dynamically updated whenever the status of a task changes. The Website periodically reads and presents the content of the Task Status Table, thus allowing users to monitor the status of their tasks.

More in detail, the actions performed by each Worker (steps 3 to 5) are described in Algorithm 1. As shown by the algorithm, input data are temporarily staged on a server for local processing. To reduce the impact of data transfer on the overall execution time, it is important that input data are physically close to the virtual servers where the workers run on. In our framework, this is achieved by exploiting the Azure's *Affinity Group* feature, which allows storage and servers to be located near to each other in the same data center for optimal performance.

Currently, the Data Mining Cloud App includes a wide range of data mining algorithms from Weka [1], and supports the *arff* format for the input datasets. Since the Weka algorithms are written in Java, each Worker includes a Java Virtual Machine to run the corresponding data mining tasks.

```
while true do
    if TaskQueue.isNotEmpty() then
        task ← TaskQueue.getTask();
        TaskStatusTable.update(task, 'running');
        localInput = <local input location>;
        localOutput = <local output location>;
        transfer(task.inputBlobURI, localInput);
        taskStatus ← execute(task.algorithm, task.paramaters,
        localInput, localOutput);
        if taskStatus = 'done' then
            transfer(localOutput, task.outputBlobURI);
            TaskStatusTable.update(task, 'done');
        end
        else
            TaskStatusTable.update(task, 'failed');
        end
        TaskQueue.remove(task);
        delete(localInput);
        delete(localOutput);
    end
end
```

Algorithm 1: Cyclic operations performed by each Worker.

B. User Interface

As mentioned earlier, the Website allows a user to submit, monitor the execution, and access the results of the data mining tasks. The Website includes three main sections: i) *Task Submission* that allows users to submit data mining tasks; ii) *Task Status* that is used to monitor the status of submitted tasks and to access results; iii) *Data Management* that allows users to manage input data and results.

In the following, we focus on task submission and management, by describing how the Data Mining Cloud App Website is used to submit a parameter sweeping data mining application. Figure 2 shows some screenshots of the Website taken during the execution of such application.

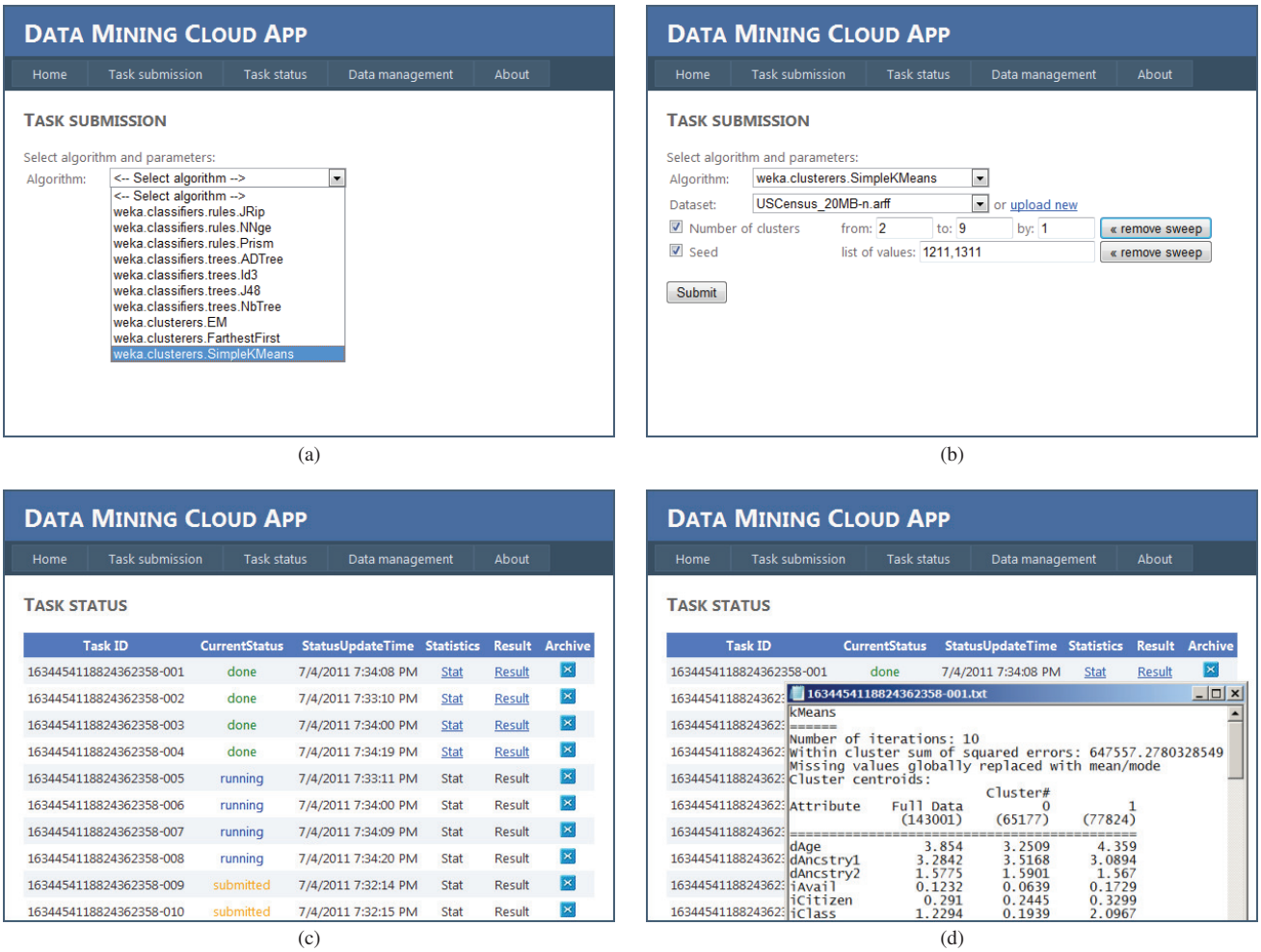


Figure 2. Four screenshots of the Data Mining Cloud App in action: (a) selection of the data mining algorithm; (b) choice of the algorithm parameters; (c) task status monitoring; (d) visualization of a task result.

After logging into the Website, a user goes the Task Submission section to select the algorithm to be used (see Figure 2(a)). A list of the available algorithms is shown to the user, who selects the one of interest. In the example, the *K-Means* clustering algorithm [2] from the Weka library is selected.

As soon as the algorithm has been selected, the Website shows to the user a form with the relevant parameters that he/she can specify for the algorithm (see Figure 2(b)). For K-Means, besides the input dataset, the relevant parameters are the *number of clusters* and the *seed*. The user can choose whether to sweep or not a certain parameter. In the example, the user chose to sweep both the number of clusters and the seed. For the former, a range of values is specified, while for the latter, a list of values is provided.

After submission, the system generates a number of independent tasks that are executed on the Cloud as discussed earlier. The user can monitor the status of each single task through the Task Status section of the Website (see Figure 2(c)). For each task, the current status (submitted, running, done or failed) and status update time are shown. Moreover, for each task that has completed its execution, the system enables two links: the first one (*Stat*) gives access to a file containing some statistics about the amount of resources consumed by the task; the second one (*Result*) visualizes the task result.

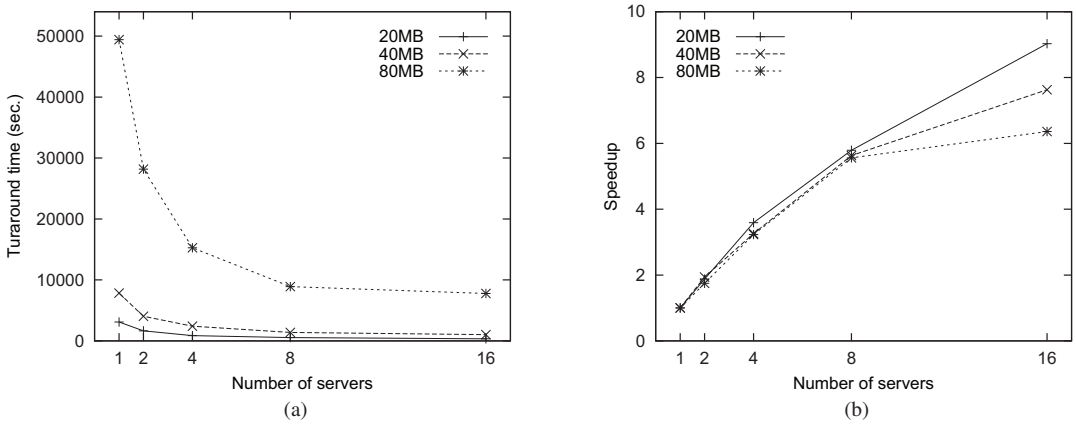
Results visualization for the first completed task is shown in Figure 2(d). The output is presented as it is generated by the K-Means data mining algorithm.

IV. Performance Evaluation

We evaluated the performance of the Data Mining Cloud App through the execution of a set of parameter sweeping data mining applications on a pool of virtual servers hosted by a Microsoft Cloud data center. We present, in particular, performance results obtained by executing *clustering* and *classification* parameter sweeping data mining applications on a set of publicly available datasets.

Table I
TURNAROUND TIMES AND COSTS FOR THE PARAMETER SWEEPING CLUSTERING APPLICATION.

N. of servers	20MB dataset		40MB dataset		80MB dataset	
	Turnaround time	Cost	Turnaround time	Cost	Turnaround time	Cost
1	0:51:37	$0.04	2:10:37	$0.11	13:43:45	$0.69
2	0:27:25	$0.04	1:07:23	$0.11	7:49:33	$0.72
4	0:14:22	$0.04	0:40:02	$0.11	4:14:39	$0.65
8	0:08:55	$0.04	0:23:11	$0.11	2:28:14	$0.65
16	0:05:43	$0.04	0:17:07	$0.11	2:09:32	$0.69



(a) (b)

Figure 3. Turnaround times and speedup values for the parameter sweeping clustering application by varying number of virtual servers and dataset size.

The experiments have been performed using 1 virtual server for the Web role instance (which hosts the Website), and up to 16 virtual servers for the Worker Role instances. Each virtual server was equipped with a single-core 1 GHz CPU, 0.75 GB of memory, and 20 GB of disk space, with a cost of $0.05 per hour. Each test has been executed by varying both the size of the input dataset and the number of virtual servers used to run the application. As performance indicators, we used the *turnaround time*, the achieved *speedup*, and the total *cost* paid.

A. Clustering Application

The clustering application discussed here is the same introduced in Section III-B to describe the user interface of the system.

As input data source we used the *US Census 1990*'s dataset[4] from the UCI KDD archive [3], which contains part of US 1990's census information. Each tuple in the dataset contains information about a US citizen, consisting of 68 categorical attributes (e.g., sex, age). The original dataset is composed of about 2,458,000 instances, stored in a file of 345 MB. In order to evaluate the system with increasing workloads, we extracted three datasets with size of 20 MB, 40 MB and 80 MB, with 143,000, 286,000 and 572,000 tuples, respectively.

[4]http://kdd.ics.uci.edu/databases/census1990/USCensus1990.html

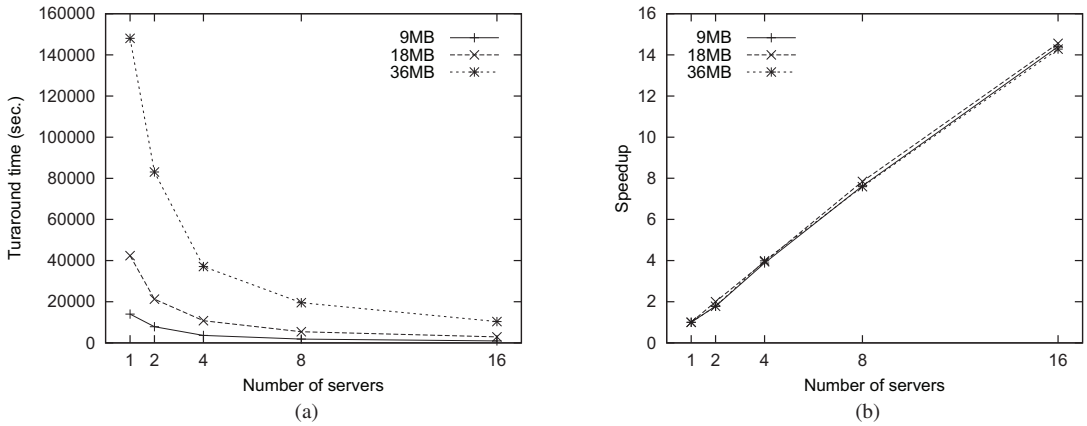
For each of the three datasets, we submitted the execution of the K-Means clustering algorithm with the following swept parameters: *number of clusters* (N) from 2 to 9; *seed* (S) equal to 1211 or 1311. We have 8 different values for N, which combined with the 2 values of S generate 16 configurations. Therefore, for each dataset size, the Data Mining Cloud App executed 16 independent tasks.

Table I presents the turnaround times and costs of the clustering application when 1, 2, 4, 8 and 16 virtual servers are used. The turnaround time includes the file transfer overhead (copying the input dataset from a Blob to the local storage of a virtual server). In our experiments this overhead was very low (only a few seconds) even with the largest of the three datasets, due to the use of the Affinity Group feature provided by Azure, as already discussed in Section III-A.

For the smallest dataset (20 MB), the turnaround time decreases from about 52 minutes obtained with a single server, to about 6 minutes using 16 servers. For the medium-size dataset (40 MB), the turnaround time passes from 2.2 hours to 17 minutes, while for the largest dataset (80 MB), the turnaround time ranges from 13.7 hours to 2.2 hours. As shown in the table, for a given dataset size, the total cost paid does not change with the number of virtual servers used, because the total execution time does not vary significantly by changing configuration.

Table II
TURNAROUND TIMES AND COSTS FOR THE PARAMETER SWEEPING CLASSIFICATION APPLICATION.

N. of servers	9MB dataset		18MB dataset		36MB dataset	
	Turnaround time	Cost	Turnaround time	Cost	Turnaround time	Cost
1	3:53:15	$0.19	11:46:14	$0.59	41:07:17	$2.06
2	2:11:05	$0.19	5:53:56	$0.59	23:04:56	$2.06
4	1:00:05	$0.19	2:59:16	$0.59	10:18:40	$1.99
8	0:30:34	$0.19	1:30:01	$0.58	5:25:04	$2.03
16	0:16:12	$0.20	0:48:30	$0.60	2:52:44	$2.10



(a) (b)

Figure 4. Turnaround times and speedup values for the parameter sweeping classification application by varying number of virtual servers and dataset size.

Fig. 3 shows turnaround times and speedup values obtained by varying number of virtual servers and dataset size. For the 20 MB dataset, the speedup passes from 1.9 using 2 servers to 9.0 using 16 servers. For the 40 MB dataset, the speedup ranges from 1.9 to 7.6. Finally, with the 80 MB dataset, we obtained a speedup ranging from 1.8 to 6.4.

The speedup achieved does not increase linearly with the number of servers used since the 16 clustering tasks are very heterogeneous in terms of execution times. In fact, the turnaround time is bound to the execution time of the slowest task instances. In our case, the slowest task instances are those in charge of grouping data into a larger number of clusters (e.g., $N = 9$), which are much slower (up to six times) than those with small values of N.

B. Classification Application

In this second set of experiments, we use the Data Mining Cloud App to run a parameter sweeping classification.

The dataset *covertype*[5], has been used as data source. This dataset contains information about forest cover type for a large number of sites in the United States. Each dataset instance, corresponding to a site observation, is described by 54 attributes that give information about the main features of a site (e.g., elevation, aspect, slope, etc.). The 55th attribute contains the cover type, represented as an integer in the range 1 to 7. The original dataset is made of 581,012 instances and is stored in a file having a size of 72MB. From this dataset we extracted three datasets with 72500, 145000 and 290000 instances and a file size of 9 MB, 18 MB and 36 MB respectively. As the classification algorithm we used *J48*, the Weka implementation of the C4.5 algorithm [4].

For each dataset size, we submitted to the Data Mining Cloud App the execution of the J48 algorithm, by sweeping its *confidence value* parameter from 0.05 to 0.50 with a step of 0.03, which produces 16 different tasks. Table II shows the turnaround times of the clustering application when 1, 2, 4, 8 and 16 virtual servers are used.

For the 9 MB dataset the turnaround time decreases from 3.9 hours obtained with a single server, to about 16 minutes using 16 servers. For the 18 MB dataset the turnaround time passes from 11.8 hours to 49 minutes. With the 36 MB dataset, the turnaround time ranges from about 41 hours to 2.9 hours. As already noted for the clustering application, also in this case the total cost paid does not significantly vary with the number of virtual servers used.

Fig. 4 shows turnaround times and speedup values by varying number of virtual servers and dataset size. For the 9 MB dataset, the speedup passes from 1.8 using 2 servers to 14.4 using 16 servers. For the 18 MB dataset, the speedup

[5]http://kdd.ics.uci.edu/databases/covertype/covertype.html

ranges from 2.0 to 14.6. Finally, with the 36 MB dataset, the speedup ranged from 1.8 to 14.3.

Note that, differently from the clustering experiments discussed earlier, in this case the speedup does increase linearly with the number of servers used, since the 16 classification tasks are homogeneous in terms of execution times.

C. Remarks

The experimental results presented above demonstrate the effectiveness of the Data Mining Cloud App in supporting parameter sweeping data mining applications on a Cloud system. For the clustering experiments, we obtained a fairly good speedup (up to 9.0 using 16 virtual servers), which however did not increase linearly due to the heterogeneity of tasks. On the other hand, when the parameter sweeping application generates homogeneous tasks, as in the classification experiments, we achieved almost a linear speedup (up to 14.6 using 16 virtual servers). This resulted in reducing the execution time of the mining application that analyzed 36 MB of data from 41 hours (about 2 days) to less than 3 hours with evident benefits for the user.

Besides performance considerations, we point out that the main goal of the Data Mining Cloud App is providing an easy-to-use SaaS interface to reliable data mining algorithms, thus enabling end-users to focus on their data mining applications without worrying about low level computing and storage details, since they are transparently managed by the underlying Cloud infrastructure.

V. Related Work

Significant research efforts have been invested in leveraging distributed computing infrastructures to implement high-performance data mining systems. Most of such systems exploit Grid middleware to provide high-level data mining services that combine hardware and software resources from many dispersed sites [5]. Some popular Grid-based data mining systems are *DataMiningGrid* [6], *Discovery Net* [7], *GridMiner* [8], *Knowledge Grid* [9], and *Weka4WS* [10].

More recently, some Cloud-based systems for data mining and scientific data analysis have been proposed, including *Sector/Sphere* [11], *All-Pairs* [12], and *Dryad* [13].

Sector/Sphere [11] is a framework designed to run data analysis applications on large distributed datasets. It consists of two complementary components: a storage component (*Sector*) and a compute component (*Sphere*). Sector provides a long term archival storage to access and index large distributed datasets. It is designed to support different types of network protocols, and to safely archive data through replication mechanisms. Sphere enables the parallel execution of user-defined functions on data stored in Sector. It splits the input files into data segments that are processed in parallel by servers called Sphere processing elements. A data segment can be a single entry record, a collection of data records or a file. An evaluation of Sector/Sphere for executing data analysis applications on a wide area is reported in [14].

All-Pairs [12] is a programming model and a framework to implement data intensive scientific applications on a cluster or a Cloud. The framework can be used for applications that have to compare the elements of two datasets on the basis of a user-defined comparison function. The user defines the problem to be solved through a specification, called *abstraction*; the framework includes an *engine* that chooses how to implement the specification using the available resources. The engine partitions and transfer data to a set of disks in the cluster, and then dispatches batch jobs to execute locally on each data split.

Dryad [13] is a Microsoft framework to run data-parallel applications on a cluster or a data center. A Dryad application combines computational *vertices* with communication *channels* to form a dataflow graph. The application runs by executing the vertices of the graph on a set of available computers and communicating as appropriate through files, TCP pipes, and shared-memory. The users define sequential programs for each vertex, and the framework schedules and executes them in parallel on multiple CPU cores or computers. Dryad has been used in combination with LINQ (a language for adding data querying capabilities to .NET languages) to implement a set of data analysis applications, including a data clustering application based on the K-Means algorithm [15].

Besides the systems discussed above, some other data intensive applications have been implemented in Cloud environments using the MapReduce programmining model [16]. MapReduce is inspired by the map and reduce primitives present in Lisp and other functional languages. A user defines a MapReduce application in terms of a map function that processes a (key, value) pair to generate a list of intermediate (key, value) pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Most MapReduce implementations, like Hadoop[6], are based on a master-slave architecture. A job is submitted by a user node to a master node that selects idle workers and assigns a map or reduce task to each one. When all the tasks have been completed, the master node returns the result to the user node. The MapReduce paradigm is appropriate to implement data mining tasks in parallel. An example is Disco [17], a framework built on top of Hadoop for data pre-processing and co-clustering. Other relevant examples are the use of MapReduce for K-Means clustering [18], and to run a semi-supervised classification on large scale graphs [19].

Differently from most of the frameworks described above, our system has been designed to provide a high-level SaaS interface to support data mining application execution on

[6]http://hadoop.apache.org

a Cloud infrastructure. Even if the Data Mining Cloud App currently supports simple and parameter-sweeping data mining applications, it can be extended to support also workflow-based data mining applications. We are currently working toward this goal.

VI. Conclusions

Cloud computing infrastructures can be effectively used to run data intensive applications. To help users in this task, simple but high-level environments must be provided. This paper presented the *Data Mining Cloud App* framework that has been designed to support the efficient execution of parameter sweeping data mining applications in a Cloud. The framework has been implemented using the Windows Azure platform and evaluated through a set of parameter sweeping clustering and classification applications.

The user interface is very simple and hides the complexity of the Cloud infrastructure used to run applications. The experimental results discussed in the paper demonstrates the effectiveness of the proposed framework, as well as the scalability that can be achieved through the parallel execution of parameter sweeping applications on a pool of virtual servers.

Other than supporting users in designing and running parameter sweeping data mining applications we intend to exploit Cloud computing platforms for running service-oriented knowledge discovery processes designed as a combination of several data analysis steps to be run in parallel on Cloud computing elements. To achieve this goal, we are currently extending the framework for supporting also *workflow-based KDD applications*, in which complex data analysis applications are specified as graphs that link together data sources, data mining algorithms, and visualization tools.

References

[1] H. Witten, E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann Publishers, 2000.

[2] J. B. MacQueen. "Some Methods for classification and Analysis of Multivariate Observations". 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, UK, 1967.

[3] S. Hettich, S. D. Bay. The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science, 1999.

[4] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[5] D. Talia, P. Trunfio. How Distributed Data Mining Tasks can Thrive as Knowledge Services. *Communications of the ACM*, 53(7), 132-137, 2010.

[6] V. Stankovski, M. T. Swain, V. Kravtsov, T. Niessen, D. Wegener, J. Kindermann, W. Dubitzky. Grid-enabling data mining applications with DataMiningGrid: An architectural perspective. *Future Generation Computer Systems*, 24(4), 259-279, 2008.

[7] S. AlSairafi, F. S. Emmanouil, M. Ghanem, N. Giannadakis, Y. Guo, D. Kalaitzopoulos, M. Osmond, A. Rowe, J. Syed, P. Wendel. The Design of Discovery Net: Towards Open Grid Services for Knowledge Discovery. *Int. Journal of High Performance Computing Applications*, 17(3), 297-315, 2003.

[8] P. Brezany, J. Hofer, A. M. Tjoa, A. Woehrer. "GridMiner: An Infrastructure for Data Mining on Computational Grids". Proc. APAC Conference and Exhibition on Advanced Computing, Grid Applications and eResearch (APAC'03), Gold Coast, Australia, 2003.

[9] A. Congiusta, D. Talia, P. Trunfio. Distributed data mining services leveraging WSRF. *Future Generation Computer Systems*, 23(1), 34-41, 2007.

[10] D. Talia, P. Trunfio, O. Verta. The Weka4WS framework for distributed data mining in service-oriented Grids. *Concurrency and Computation: Practice and Experience*, 20(16), 1933-1951, 2008.

[11] Y. Gu, R. Grossman. Sector and Sphere: The Design and Implementation of a High Performance Data Cloud. *Philosophical Transactions, Series A: Mathematical, physical, and engineering sciences*, 367(1897), 2429-2445, 2009.

[12] C. Moretti , J. Bulosan , D. Thain , P. J. Flynn. "All-Pairs: An Abstraction for Data-Intensive Cloud Computing". IEEE Int. Symposium on Parallel and Distributed Processing (IPDPS'08), Miami, USA, 2008.

[13] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly. "Dryad: distributed data-parallel programs from sequential building blocks". 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys'07), Lisbon, Portugal, 2007.

[14] R. Grossman, Y. Gu. "Sector and Sphere: Data mining using high performance data clouds: experimental studies using sector and sphere". 14th ACM SIGKDD Int. Conference on Knowledge discovery and data mining (KDD'08), New York, USA, 2008.

[15] J. Ekanayake, T. Gunarathne, G. Fox, A. S. Balkir, C. Poulain, N. Araujo, R. Barga. "DryadLINQ for Scientific Analyses". 5th IEEE International Conference on e-Science (e-Science'09), Oxford, UK, 2009.

[16] J. Dean, S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1), 107-113, 2008.

[17] S. Papadimitriou, J. Sun. "DisCo: Distributed Co-clustering with Map-Reduce: A Case Study towards Petabyte-Scale End-to-End Mining". 8th IEEE International Conference on Data Mining (ICDM'08), Pisa, Italy, 2008.

[18] J. Ekanayake, S. Pallickara, G. Fox. "Mapreduce for data intensive scientic analyses". 4th IEEE International Conference on e-Science (e-Science'08), Indianapolis, USA, 2008.

[19] D. Rao, D. Yarowsky. "Ranking and semi-supervised classification on large scale graphs using map-reduce". Workshop on Graph-based Methods for Natural Language Processing (TextGraphs'09), Stroudsburg, USA, 2009.