A Cloud Framework for Big Data Analytics Workflows on Azure

Fabrizio MAROZZO[a], Domenico TALIA[a,b] and Paolo TRUNFIO[a]
[a] *DIMES, University of Calabria, Rende (CS), Italy*
[b] *ICAR-CNR, Rende (CS), Italy*

Abstract. Since digital data repositories are more and more massive and distributed, we need smart data analysis techniques and scalable architectures to extract useful information from them in reduced time. Cloud computing infrastructures offer an effective support for addressing both the computational and data storage needs of big data mining applications. In fact, complex data mining tasks involve data- and compute-intensive algorithms that require large and efficient storage facilities together with high performance processors to get results in acceptable times. In this chapter we present a Data Mining Cloud Framework designed for developing and executing distributed data analytics applications as workflows of services. In this environment we use data sets, analysis tools, data mining algorithms and knowledge models that are implemented as single services that can be combined through a visual programming interface in distributed workflows to be executed on Clouds. The first implementation of the Data Mining Cloud Framework on Azure is presented and the main features of the graphical programming interface are described.

Keywords. Distributed data mining, Cloud computing, Service-oriented computing.

Introduction

Digital data repositories are more and more massive, complex, and ubiquitous. Therefore, we need smart data analysis techniques and scalable architectures to extract useful information from them efficiently. Research activities in this area must push Clouds moving from a computation and data management platform to a pervasive, scalable data analytics infrastructure. This trend needs new models and technologies for enabling Cloud computing to support the implementation of smart data analysis techniques that should become scalable and dynamic in resource allocation on Clouds.

Cloud computing infrastructures offer an effective support for addressing both the computational and data storage needs of big data mining applications. In fact, complex data mining tasks involve data- and compute-intensive algorithms that require large and efficient storage facilities together with high performance processors to get results in acceptable times [1, 2]. However, efforts must be carried out to implement Cloud-based big data analytics services, tools and applications.

In this chapter we present a Data Mining Cloud Framework designed for developing and executing distributed data analytics applications as workflows of services. In this environment we use data sets, analysis tools, data mining algorithms and knowledge models implemented as single services. These services can be combined through a visual programming interface to program distributed workflows to be executed on Clouds. The first implementation of the Data Mining Cloud Framework on Azure is presented and the main features of the programming interface are described.

The chapter is organized as follows. Next section introduces big data analytics concepts and sketches implementation issues on Clouds. Section 3 presents the architecture of the Data Mining Cloud Framework and Section 4 describes its programming interface through a sample application. Section 5 concludes the chapter.

1. Big Data Analytics

The term *big data* today is used to specify a massive collection of digital data that is so large and complex to make difficult its processing by using traditional data management tools and techniques [1]. Although there is hype on this subject and big data is an over-used term, the topic is very important in computer science and in daily human activities since the amount of data today available is very huge and it generates the need to be effectively used in science, business, and social activities. Big data actually does not specifically refers to the large size of data, but it also includes the degree of complexity and variety of data, the value that can be obtained from smart analysis techniques, and the velocity of data collection and processing. Big data nowadays come from mobile devices, social networks, bioinformatics repositories, and financial database, from the Web and from the sky. We need to cope with data deluge by using innovative data analytics techniques to transform big data in big value.

In science and business, people are analyzing data to extract information and knowledge useful for making new discoveries or for improving the business by smart decision. This can be done by exploiting big data analytics techniques and tools. Big data analytics is the advanced use of mining techniques on big data sets [3]. Putting big data and knowledge discovery techniques together with scalable computing systems will produce new insights in shorter time. This combination will provide a significant added value in many application domains. In fact, as the size, complexity and number of the digital data sources generated in science and business increase, transforming this data into information and knowledge requires addressing several challenges. Cloud computing can provide the data storage and access facilities together with high-performance computing power needed to support the exploration and extraction of knowledge from data sets.

Unfortunately, very few Cloud-based analytic platforms are available today, even if several users could have benefit from them. However, we expect that data analytics Clouds will become common platforms for big data analytics within a few years. Some solutions are based on Hadoop, others are proprietary solutions as those provided by IBM, EMC and Kognitio. As Clouds will become common scalable platforms for big data analytics, programming tools, suites and data mining strategies will be ported on such platforms for developing big data discovery solutions.

Both the PaaS (Platform as a Service) and Saas (Software as a Service) models can be adopted for implementing big data analytics solution on clouds. PaaS will be useful

to support data analytics programming suites and environments where data mining developers can design scalable data analytics services and applications. On the other hand, the SaaS model will offer complete big data analytics applications to end users that can execute analysis on large and/or complex data sets by exploiting scalability of Clouds in data storage and processing power.

The Cloud framework described in this chapter can be used as a high-level PaaS data analytics programming environment and also to provide a set of SaaS suites for big data analytics that, built on the PaaS layer, can be used by end users whose goal is to make complex analysis without worrying about the Cloud platform details and the way in which the analytics suite has been programmed.

2. Data Mining Cloud Framework

The software environment that we developed for supporting data analytics on Clouds is called Data Mining Cloud Framework. This framework enables the execution of distributed data analysis applications on top of Cloud computing and storage services by providing a visual programming interface for knowledge discovery workflows design and execution. The framework has been implemented using Windows Azure and has been used to run distributed data mining applications on a Microsoft Cloud data center.

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

- a set of binary and text data containers (Azure 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, in charge of executing the data mining tasks submitted by the users;
- a *website* that allows users to submit, monitor the execution, and access the results of data mining tasks.

The following steps are performed to develop and execute a knowledge discovery application through the system (see Figure 1):

1. A user accesses the Website and develops her/his application as a workflow of services through a Web-based interface. A service catalog provides the user with the available data and software services that can be used to design the application as a workflow. After composing the application, she/he can submit the workflow for execution on Azure.
2. After the application submission, a set of tasks that compose the workflow are created and inserted into the Task Queue.
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 application. To this end, a file transfer is performed from the container 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 data storage element.
6. The Website notifies the user as soon as her/his task(s) have completed, and allows her/him to access the results.

The set of tasks created on the second step depends on the type of application submitted by the user. In the case of a simple application expressed by a single task, just one data mining task is inserted into the Task Queue. In the general case, the user submits a multi-task workflow-based application. In this case the set of tasks created depends on how many data mining tools are invoked within the workflow. The Task Status Table is dynamically updated whenever the status of a task changes. The Website periodically reads and shows the content of such table, thus allowing users to monitor the status of their tasks.

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. For example, in the Azure implementation of our framework, this has been carried out 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.

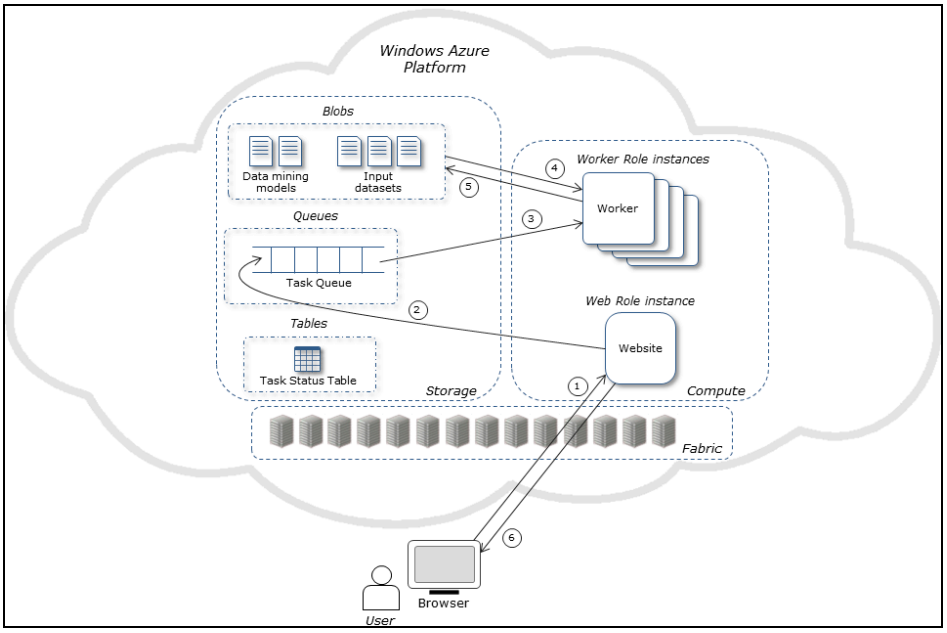

Figure 1. Architecture of the Data Mining Cloud Framework.

3. Visual Programming Interface

The user interface of the framework is composed of two main parts: one pane for

composing and running both single-task and parameter-sweeping applications, described in a previous work [4], and another pane for programming and execution of workflow-based knowledge discovery applications. Since in this paper we focus on the workflow programming features of the framework, we discuss the second pane. In particular, by using a data mining application composed of several sequential and parallel steps, in this section we present the main features of the visual programming interface of the Cloud-based framework.

As mentioned before, we implemented the visual programming interface and its services to support the composition and execution of workflow-based knowledge discovery applications on Cloud platforms. Workflows allow the implementation of research and scientific processes by providing a paradigm that can encompass all the steps of discovery based on the execution of complex algorithms and the access and analysis of scientific data. In data-driven discovery processes, knowledge discovery workflows can produce results that can confirm real experiments or provide insights that cannot be achieved in laboratories. In particular, when data sources are massive and/or decentralized, service-oriented workflow allow programmers to express analysis programs in a direct and user-friendly way.

Following the approach proposed in [2] and [5], we model a knowledge discovery workflow as a graph whose nodes represent resources (datasets, data mining tools, data mining models), implemented as Cloud services, and whose edges represent dependencies between resources. For supporting workflow composition, we implemented the Website by using native HTML 5 features. It allows users to design service-oriented knowledge discovery workflows with a simple drag-and-drop approach.

3.1. A distributed data mining application

Here we use a concurrent data mining application composed of several sequential and parallel steps as an example for presenting the main features of the visual programming interface of the Data Mining Cloud Framework.

The example application analyzes a dataset by using *n* instances of the J48 classification algorithm (an Java implementation of C4.5 [3] provided by the Weka toolkit [6]), which work on *n* partitions of the training set and generate *n* knowledge models. By using the *n* generated models and the test set, *n* classifiers produce in parallel *n* classified datasets (*n* classifications). In the final step of the workflow, a voter generates the final dataset by assigning a class to each data item, choosing the class predicted by the majority of the models.

Figure 2 shows a snapshot of the visual interface where the first step of the workflow is designed. In particular, we can see the splitting of the original dataset in training and test set operated by a partitioning tool. A set of configuration parameters is associated with each workflow node. The parameters of a given node can be specified through a pop-up panel that appears when that node is selected. For example, the right part of Figure 2 shows the configuration panel for the partitioning tool. In this case, only one parameter can be specified, namely the percentage of the input dataset that must be used to produce the training set (70% in the given example).

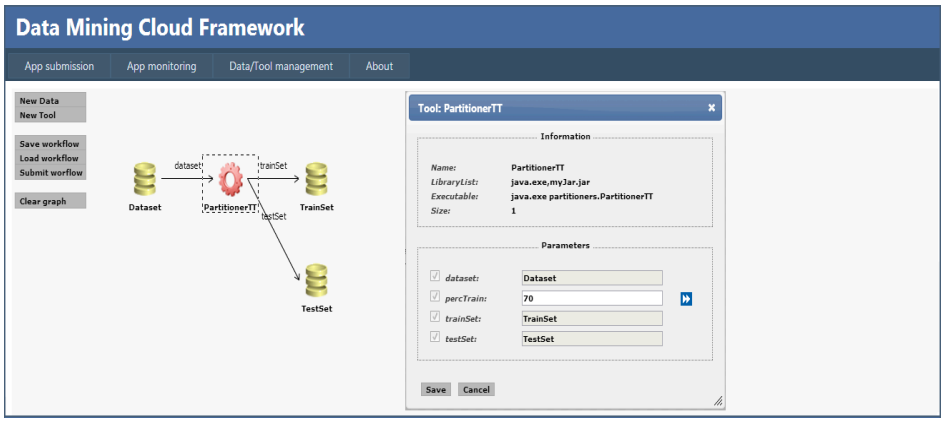

Figure 2. First step: The input dataset is partitioned into training and test set.

As a second step, the training set is partitioned into 10 parts by using another partitioning tool (see Figure 3). The 10 training sets resulting from the partitioning are represented in the workflow as a single *data array node*, labeled as *TrainSetPart[10]*. A data array is an ordered collection of input/output data sources, e.g., a collection of dataset to be analyzed or a collection of models that a classifier can use to classify an unlabeled dataset. Using this simple but powerful array notation, the *i*-th element of the data partition is identified by the dataset name and the index *i* in square brackets (e.g., *Dpartion[i]*) and a similar notation is visualized in the graphical interface, as can be seen in the rightmost workflow node in Figure 3.

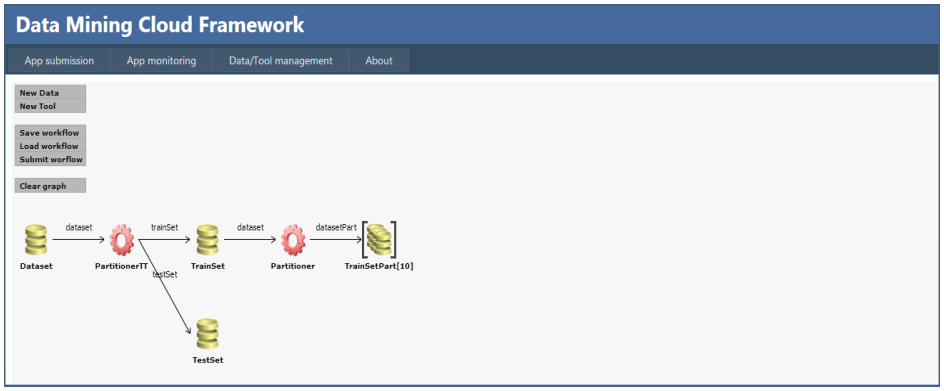

Figure 3. Second step: The train set is partitioned into 10 parts.

Figure 4 shows the third step of the workflow, in which the 10 training sets are analyzed in parallel by 10 instances of the J48 classification algorithm, to produce the same number of classification models. As in the previous example, the array notation is used here also to express an array of similar software tool instances that will be run in parallel on a set of Cloud virtual machines. In particular, a *tool array node*, labeled as *J48[10]*, is used to represent the 10 instances of the J48 classification algorithm, while another data array node, labeled as *Model[10]*, represents the models generated by the

classification algorithms. In practice, this portion of the workflow specifies that *J48[i]* takes as input data partition *TrainSetPart[i]* to produce in parallel *Model[i]*, for $1 \leq i \leq 10$.

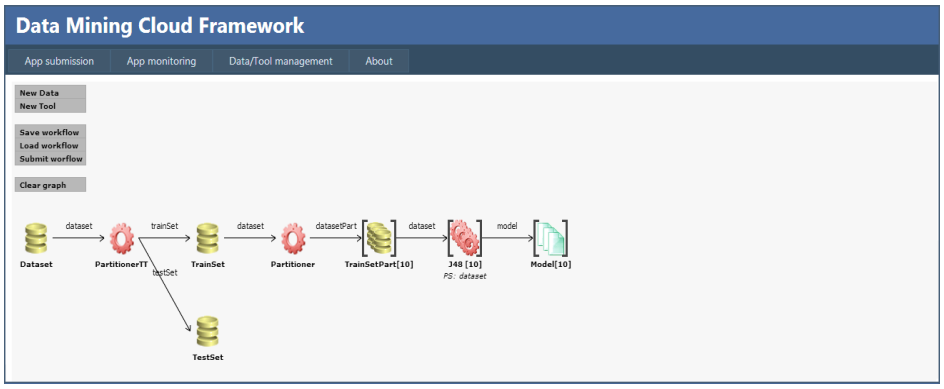

Figure 4. Third step: Each train set part is analyzed by an instance of the J48 classification tool; as a result, 10 classification models are produced in parallel.

The fourth step classifies the test set created before by using the ten models generated on the previous step (see Figure 5). The classification is performed by ten classifiers that run in parallel to produce 10 classified test sets. More in detail, *Classifier[i]* takes in input the *TestSet* and *Model[i]* to produce *ClassTestSet[i]*, for $1 \leq i \leq 10$.

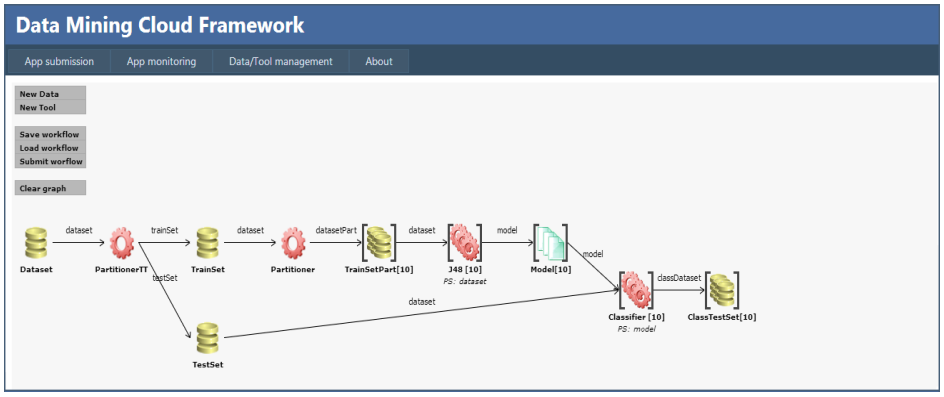

Figure 5. Fourth step: The test set is classified by 10 classifiers, which perform the tasks concrrently by using the 10 classification models generated on the previous step.

During the last step of the workflow (see Figure 6), the 10 classified test sets are passed to a voter tool that produces the final dataset, labeled as *FinalClassTestSet*.

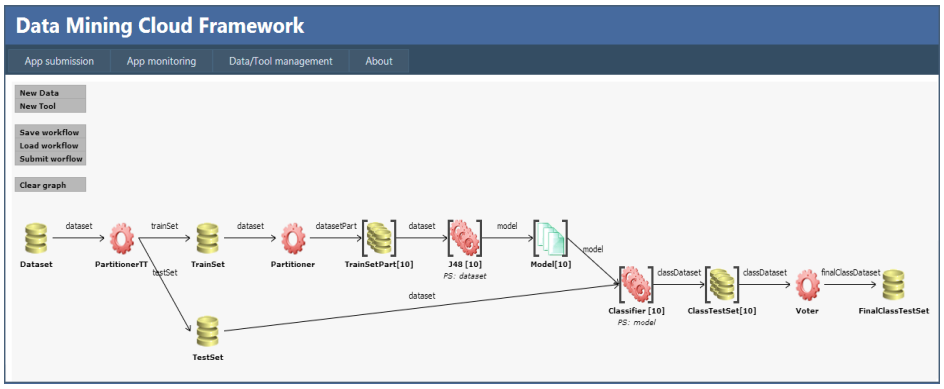

Figure 6. Fifth step: The 10 instances of the classified test set are compared instance-by-instance by a voter to produce the final classified test set.

When the workflow is complete, that is every node has been specified and connected, it can be submitted for execution. Figure 7 shows a snapshot of the workflow taken during its execution. During the execution, the status of each node (tool or data) is shown through a small symbol on the bottom right part of the icon. For example, the figure shows that *PartitionerTT* has completed the execution, *Partioner* is running, while the other tools have been submitted, but not yet executed since they depend on the completion of preceding tasks in the workflow.

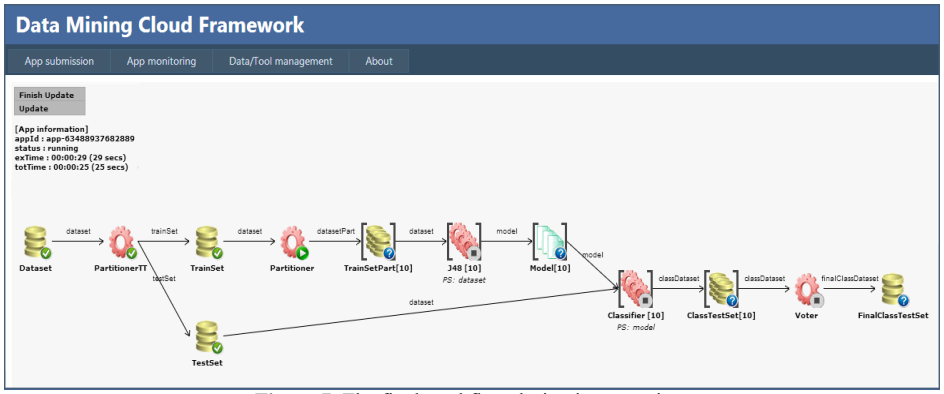

Figure 7. The final workflow during its execution.

Finally, Figure 8 shows the workflow visualization after completion of its execution. Some statistics about the overall application are shown on the upper left part of the window. In this example, it is shown that, by using 10 virtual machines, the workflow completed 264 seconds after its submission, with a total execution time (i.e., the sum of the execution times of all the tasks) of 1604 seconds.

The total execution time corresponds to the sequential execution time of all the workflow tasks. Therefore, although this data analysis example includes sequential tasks and parallel ones (and so the scalability cannot be linear), we can have a simple measure of how the completion time can be reduced by using at most 10 Cloud virtual machines. By running concurrently the parallel tasks included in the workflow, the data

analysis was completed in 1/6 of the sequential time. Even if the example is not so huge, this result shows the effectiveness of the proposed approach based on Cloud resource exploitation for running data analysis applications.

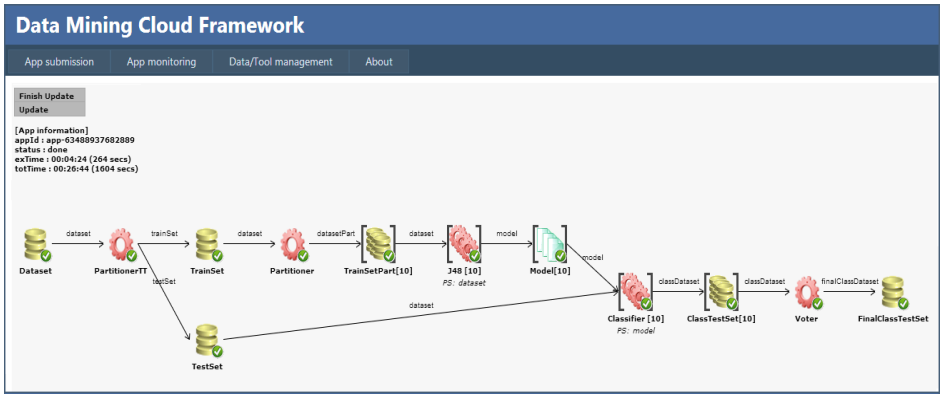

Figure 8. The final workflow after completion of its execution.

4. Conclusions

New parallel and distributed computing infrastructures are vital to run smart scalable analysis techniques to solve more challenging problems in science, particularly when large data sets are involved or real-time evaluation is needed. Cloud computing systems today are used for implementing dynamic data centers and for high-performance computing on massive number of processors. They can also be effectively used as scalable infrastructures for running big data analysis that often involve large data sets and complex algorithms. Thus, Clouds can be platforms for the implementation and delivery of scalable service-oriented knowledge discovery applications.

Based on this approach, we designed the Cloud Data Mining Framework, a service-oriented software environment for large-scale data analysis on Cloud infrastructures. Here we described, through an application example, the main features of the visual programming interface of the framework, focusing in particular on the workflow programming interface. In this environment we use data sets, analysis tools, data mining algorithms and knowledge models that are implemented as single services that can be combined in distributed workflows to be executed on Clouds.

We also evaluated the execution time of the data analysis application using our framework on ten virtual servers hosted by a Microsoft Cloud data center. The experimental results, presented in this chapter, showed the effectiveness of the framework, as well as the potential scalability that can be achieved through the parallel execution of the workflow tasks on a pool of virtual servers.

Currently, we are working on the workflow composition interface with the aim of extending the supported design patterns (e.g., conditional branches and iterations) and to experimentally evaluate its functionality and performance on Windows Azure during the design and execution of complex knowledge discovery workflows on large data on

the Cloud.

References

[1] MIKE2.0, Big Data Definition, mike2.openmethodology.org/wiki/Big_Data_Definition, Oct. 2012.

[2] D. Talia, P. Trunfio, *Service-Oriented Distributed Knowledge Discovery*, Chapman & Hall/CRC, USA, 2012.

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

[4] F. Marozzo, D. Talia, P. Trunfio. A Cloud Framework for Parameter Sweeping Data Mining Applications. Proc. of the 3rd International Conference on Cloud Computing Technology and Science (CloudCom 2011), Athens, Greece, pp. 367-374, 2011.

[5] E. Cesario, M. Lackovic, D. Talia, P. Trunfio. A Visual Environment for Designing and Running Data MiningWorkows in the Knowledge Grid. In: D. Holmes, L. Jain (Eds.), Data Mining: Foundations and Intelligent Paradigms, pp. 57-75, Springer, 2012.

[6] H. Witten, E. Frank, Data Mining: *Practical Machine Learning Tools with Java Implementations*, Morgan Kaufmann, USA, 2000.