Using Clouds for Scalable Knowledge Discovery Applications

Fabrizio Marozzo[1], Domenico Talia[1,2], Paolo Trunfio[1]

[1] DIMES, University of Calabria
[2] ICAR-CNR
Via P. Bucci, Cubo 41c
87036 Rende(CS), Italy
{fmarozzo,talia,trunfio}@deis.unical.it

Abstract. Cloud platforms provide scalable processing and data storage and access services that can be exploited for implementing high-performance knowledge discovery systems and applications. This paper discusses the use of Clouds for the development of scalable distributed knowledge discovery applications. Service-oriented knowledge discovery concepts are introduced, and a framework for supporting high-performance data mining applications on Clouds is presented. The system architecture, its implementation, and current work aimed at supporting the design and execution of knowledge discovery applications modeled as workflows are described.

1 Introduction

Cloud platforms, HPC systems, and large-scale distributed computing systems can be used to solve big and complex problems in several scientific and business domains. In particular, the huge amount of data available today in digital repositories requires smart data analysis techniques and scalable algorithms, techniques, and systems to help people to deal with it.

Cloud computing is a paradigm and a technology that provide location-independent storing, processing and use of data on remote computers accessed over the Internet. Users can exploit almost unlimited computing power on demand, thus they do not need to buy hardware and software to fulfill their needs [1].

Users can employ Cloud systems to store any kind of information and to use software as a service (e.g., office automation tools, music players, games). Organizations, in science, business and public services, can use Cloud services to run data centers and implement different kinds of IT activities. People and enterprises can leverage Clouds to scale up their activities without investing in large physical infrastructures.

Data mining techniques are widely used in several application areas for analyzing and extracting useful knowledge from large datasets. In most cases, the core of data analysis applications is represented by compute-intensive data mining algorithms, thus leading to very long execution times when a single computer is used to analyze a large dataset. Cloud systems can be effectively used to handle data mining processes since they provide scalable processing and storage services, together with software platforms for developing knowledge discovery systems on top of such services.

For instance, scalable computing systems give a fundamental support to life science discoveries in several aspects:

- Distributed/parallel data management and processing,
- Scalable data integration and interoperability,
- Distributed/parallel data mining,
- Collaborative data exploration and visualization.

Considering this scenario, we worked to design a framework for supporting the scalable execution of knowledge discovery applications on top of Cloud platforms. The framework has been designed to be implemented on different Cloud systems; however, an implementation of this framework has been carried out using Windows Azure[3] and has been evaluated through a set of data analysis applications executed on a Microsoft Cloud data center.

The framework has been designed to support three classes of knowledge discovery applications: *single-task applications*, in which a single data mining task such as classification, clustering, or association rules discovery is performed on a given dataset; *parameter-sweeping applications*, in which a dataset is analyzed by multiple instances of the same data mining algorithm with different parameters; *workflow-based applications*, in which knowledge discovery applications are specified as graphs linking together data sources, data mining tools, and data mining models.

The remainder of this paper is structured as follows. Section 2 discusses the system architecture, execution mechanisms, and an early version of the user interface designed to support single-task and parameter-sweeping applications. Section 3 describes current work aimed at providing a Web-based interface and associated execution mechanisms for designing and running knowledge discovery applications as workflows. Finally, Section 4 concludes the paper.

2 System architecture and implementation

The architecture of the designed framework, shown in Figure 1, includes the following components:

- A set of binary and text data containers 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 resulting from the data mining applications submitted by the users.

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

– A *Website* that allows users to submit, monitor the execution, and access the results of their knowledge discovery applications.

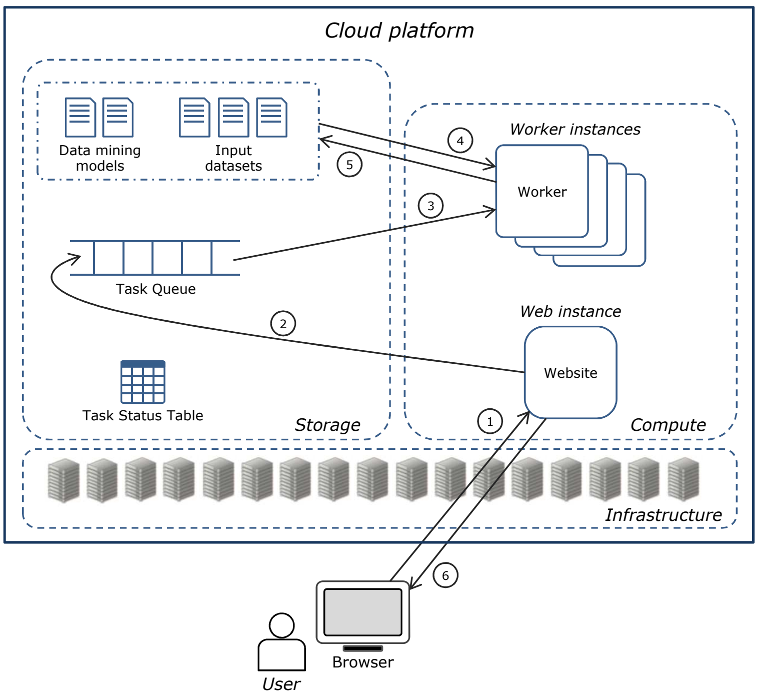


Fig. 1. System architecture and application execution steps.

2.1 Applications execution

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 develop her/his application (either single-task, parameter-sweeping, or workflow-based) through a Web-based interface. After completing the application, she/he can submit it for execution on Azure.
2. After the application submission, a set of tasks are created and inserted into the Task Queue on the basis of the application submitted by the user.
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 single-task application, just one data mining task is inserted into the Task Queue. If the user submits a parameter-sweeping application, one task for each combination of the input parameters values is executed. In the case of a workflow-based application, 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 implementaion of our framework, this had been done 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.

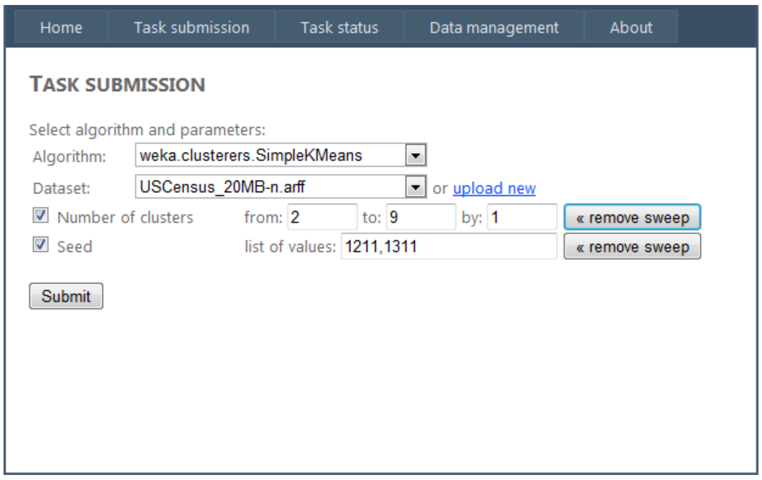
Currently, the framework includes a wide range of data mining algorithms from Weka [2], and supports the *arff* format for the input datasets. Since the Weka algorithms are written in Java, each Worker includes a Java Virtual Machine to run the corresponding data mining tasks.

2.2 User interface

The user interface of the system is composed of two main parts: one pane for composing and running both single-task and parameter-sweeping applications and another pane for composition and execution of workflow-based knowledge discovery applications.

The first part of the user interface (the Website), presented in [3], was implemented to support single-task and parameter-sweeping applications. It includes three main sections: i) *Task submission* that allows users to submit their applications; ii) *Task status* that is used to monitor the status of tasks and to visualize results; iii) *Data management* that allows users to manage input data and past results.

Figure 2 shows a screenshot of the Task submission section, taken during the execution of a parameter-sweeping application. An application can be configured by selecting the algorithm to be executed, the dataset to be analyzed, and the relevant parameters for the algorithm. The system submits to the Cloud a number of independent tasks that are executed concurrently on a set of virtual servers.



TASK SUBMISSION

Select algorithm and parameters:

Algorithm:	weka.clusterers.SimpleKMeans						
Dataset:	USCensus_20MB-n.arff	or upload new					
☑ Number of clusters	from:	2	to:	9	by:	1	« remove sweep
☑ Seed	list of values:	1211,1311					« remove sweep

Fig. 2. Screenshot of the Task submission section.

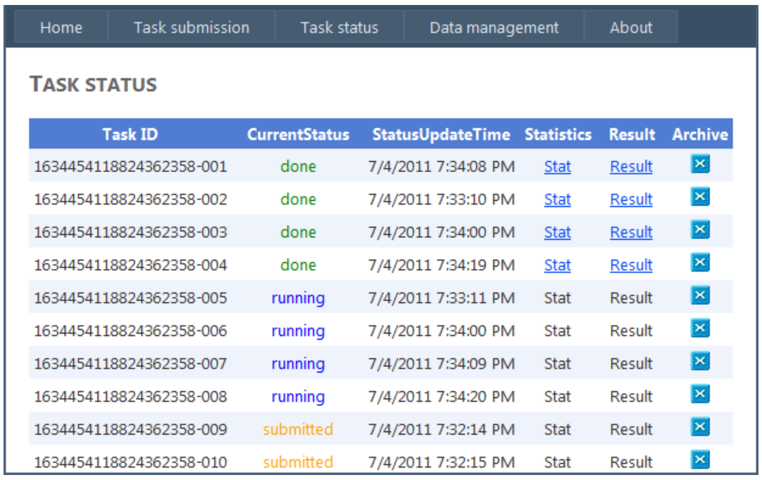
The user can monitor the status of each single task through the Task status section, as shown in Figure 3. For each task, the current status (submitted, running, done or failed) and status update time are shown. Moreover, for each task that has completed its execution, two links are enabled: the first 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.

3 Cloud knowledge discovery workflows

We prototyped the programming interface and its services to support the composition and execution of workflow-based knowledge discovery applications in our Cloud framework. Workflows support research and scientific processes by providing a paradigm that may 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.

Following the approach proposed in [5] and [6], 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.

To support the workflow composition, we implemented the Website part that, by using native HTML 5 features, allows users to design service-oriented knowledge discovery workflows with a simple drag-and-drop approach. Figure 4 shows a screenshot of the Website taken during the composition of a knowledge discovery workflow.



TASK STATUS

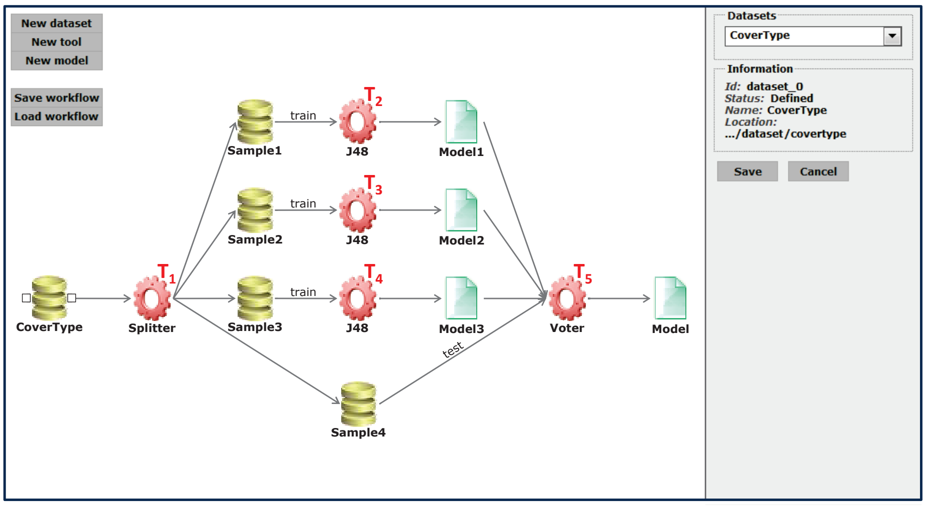
Task ID	CurrentStatus	StatusUpdateTime	Statistics	Result	Archive
1634454118824362358-001	done	7/4/2011 7:34:08 PM	Stat	Result	☒
1634454118824362358-002	done	7/4/2011 7:33:10 PM	Stat	Result	☒
1634454118824362358-003	done	7/4/2011 7:34:00 PM	Stat	Result	☒
1634454118824362358-004	done	7/4/2011 7:34:19 PM	Stat	Result	☒
1634454118824362358-005	running	7/4/2011 7:33:11 PM	Stat	Result	☒
1634454118824362358-006	running	7/4/2011 7:34:00 PM	Stat	Result	☒
1634454118824362358-007	running	7/4/2011 7:34:09 PM	Stat	Result	☒
1634454118824362358-008	running	7/4/2011 7:34:20 PM	Stat	Result	☒
1634454118824362358-009	submitted	7/4/2011 7:32:14 PM	Stat	Result	☒
1634454118824362358-010	submitted	7/4/2011 7:32:15 PM	Stat	Result	☒

Fig. 3. Screenshot of the Task status section.

On the top-left of the window, three icons allow a user to insert a new dataset, tool, or model node into the workflow. Once placed into the workflow, a node can be linked to others to establish the desired dependencies. Relevant information for each node (e.g., the location for a dataset, algorithm name and associated parameters for a tool) can be specified through a configuration panel available on the right of the window.

As an example, the workflow shown in Figure 4 implements a knowledge discovery process known as *bagging*. The workflow begins, on the left, with the input dataset (CoverType) connected to a Splitter tool that extracts four samples from it. The first three samples are used as training sets by three instances of the J48 classification tool (an open source implementation of C4.5 [4]), which generate three independent classification models from them. Then, a Voter tool receives the three models and the fourth sample as test set, and produces the final classification model through a voting procedure.

The five tools invoked in the workflow (the Spitter, three J48 instances, and the Voter) are translated into an equal number of tasks, indicated as $T_1 ... T_5$ in Figure 4. Differently from parameter-sweeping applications whose tasks are independent each other and therefore can be executed in parallel, the execution order of workflow tasks depends on the dependencies specified by the workflow edges. To ensure the correct execution order, each workflow task is associated with a list of tasks that must be completed before starting its execution. For instance, Figure 5 shows the content of the Task Queue after having submitted the workflow shown in Figure 4. For each task, the list of tasks to be completed before its execution is reported. According with the tasks dependencies specified by the workflow, after completion of T_1, the execution of T_2, T_3 and T_4 can proceed concurrently. Moreover, T_5 can be executed only after completion of T_2, T_3 and T_4.



Datasets

CoverType

Information

Id: dataset_0
Status: Defined
Name: CoverType
Location:
.../dataset/covertype

Fig. 4. Workflow composition interface.

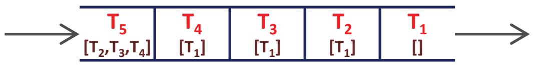


Fig. 5. Content of the Task Queue after submission of the workflow in Figure 4.

4 Conclusions

We need new distributed infrastructures and smart scalable analysis techniques to solve more challenging problems in science. Cloud computing systems can be effectively used as scalable infrastructures for service-oriented knowledge discovery applications. Based on this vision, we designed a Cloud-based framework for large-scale data analysis.

We evaluated the performance of the system through the execution of a set of long-running parameter-sweeping knowledge discovery applications on a pool of virtual servers hosted by a Microsoft Cloud data center. The experimental results, presented in [3], demonstrated the effectiveness of the framework, as well as the scalability that can be achieved through the execution of parameter-sweeping applications on a pool of virtual servers. For example, the classification of a large dataset (290,000 records) on a single virtual server required more than 41 hours, whereas it was completed in less than 3 hours on 16 virtual servers. This corresponds to an execution speedup equal to 14.

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. The European Commission. *Unleashing the Potential of Cloud Computing in Europe*. Brussels, 2012.
2. H. Witten, E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann Publishers, 2000.
3. 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.
4. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.
5. E. Cesario, M. Lackovic, D. Talia, P. Trunfio. A Visual Environment for Designing and Running Data Mining Workflows in the Knowledge Grid. In: D. Holmes, L. Jain (Eds.), *Data Mining: Foundations and Intelligent Paradigms*, pp. 57-75, Springer, 2012.
6. D. Talia, P. Trunfio. *Service-Oriented Distributed Knowledge Discovery*. Chapman and Hall/CRC Press, 2012.