Nubytics: Scalable Cloud Services for Data Analysis and Prediction

Eugenio Cesario*‡, Andrea Raffaele Iannazzo*, Fabrizio Marozzo*§, Fabrizio Morello*, Domenico Talia*§, Paolo Trunfio*§

*DtoK Lab Srl, Italy. Email: [iannazzo, morello]@dtoklab.com

§DIMES, University of Calabria, Italy. Email: [fmarozzo, talia, trunfio]@dimes.unical.it

‡ICAR-CNR, Italy. Email: cesario@icar.cnr.it

***Abstract*—The analysis of very large data sources requires scalable systems to reduce execution time and make the Big Data paradigm viable. Cloud infrastructures can be effectively used as scalable computing and storage platforms for implementing high-performance data analysis services. Nubytics is a Software-as-a-Service (SaaS) system that exploits Cloud facilities to provide efficient services for analyzing large datasets. The system allows users to import their data to the Cloud, extract knowledge models using high performance data mining services, and exploit the inferred knowledge to predict new data and behaviours. In particular, Nubytics provides data classification and regression services that can be used in a variety of scientific and business applications. Scalability is ensured by a parallel computing approach that fully exploits the resources available on a Cloud. The paper describes the main services provided by Nubytics and presents an experimental performance analysis to show its scalability.**

I. Introduction

Data analysis techniques are used in many application areas to extract useful knowledge from large datasets. The analysis of very large data sources requires scalable systems to reduce execution time and make the Big Data paradigm viable. Cloud technologies can be effectively exploited to provide end users with the computing and storage resources, and the execution mechanisms needed to efficiently run complex data analysis tasks. Indeed, several Cloud-based data analysis systems have been proposed in the latest years. A key advantage of Cloud computing 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 that are assigned to applications on-demand. Cloud resources are provided in highly scalable way, i.e., they are allocated dynamically to applications depending on the current level of requests [1].

We worked in the area of Cloud-based data analysis services by designing and developing Nubytics, a system that exploits Cloud facilities to provide scalable services for analyzing large datasets. The system allows users to import their data to the Cloud, extract knowledge models using high performance data mining services, and use the inferred knowledge to predict new data. Nubytics provides data classification and regression services that can be used in a variety of scientific and business applications. Scalability is ensured by a parallel computing approach that fully exploits the resources available on the Cloud. In addition, Nubytics is provided in accordance with the Software-as-a-Service (SaaS) model. This means that no installation is required on the user's machine: the Nubytics interface is offered by a web browser, so it can be run from most devices, including desktop PCs, laptops, and tablets. This is a key feature for users who need ubiquitous and seamless access to scalable data analysis services, without needing to cope with the installation and system management issues of traditional analytics tools.

The remainder of the paper is structured as follows. Section II discusses related work. Section III introduces the architecture of Nubytics. Section IV describes the main services provided by the system. Section V shows the system functionalities through a use case. Section VI presents an experimental performance analysis demonstrating the scalability of the system. Finally, Section VII concludes the paper.

II. Related work

Several software tools and frameworks have been developed and used for implementing data analysis applications and workflows on Cloud systems.

Hadoop [2] is an implementation of the MapReduce programming model [3] for developing parallel data analysis applications. It can be adopted for developing applications using many programming languages (e.g., Java, Ruby, Python, C++). Hadoop relieves developers from having to deal with distributed computing issues such as load balancing, fault tolerance and data locality.

Spark [4] is a framework for in-memory data analysis designed to efficiently perform both batch processing applications (similar to Hadoop) and dynamic applications like streaming, interactive queries, and graph analysis. Spark is compatible with Hadoop data and can run in Hadoop clusters. In contrast to Hadoop in which intermediate data are always stored in distributed file systems, Spark stores data in memory to obtain better performance.

Microsoft Azure Machine Learning (Azure ML) [5] is a SaaS that provides a Web-based machine learning IDE for creation and automation of machine learning workflows. Azure ML includes a rich catalog of processing tools that can be easily included in a workflow to prepare/transform data or to mine

data through supervised learning (regression and classification) or unsupervised learning (clustering) algorithms.

Galaxy[6] is a web-based bioinformatics workflow management system. It runs on Linux/Unix based servers, and therefore several organizations execute Galaxy on private or public Cloud IaaS. In order to improve Galaxy's capabilities with respect to interfacing with large scale computational systems and running workflows in a parallel manner, Galaxy has recently been integrated with Swift/T [7], a large-scale parallel programming framework.

Taverna [8] is a workflow management system mostly used in the life sciences community. It can orchestrate Web Services and these may be running in the Cloud. Recently, the Tavaxy system [9] has been developed for integrating Taverna and Galaxy. In particular, Tavaxy allows users to create and execute workflows employing an extensible set of patterns that enables the integration of existing workflows from Taverna and Galaxy.

E-Science Central (e-SC) [10] allows scientists to store, analyze and share data in the Cloud. Its in-browser workflow editor allows users to design applications by connecting services, either uploaded by themselves or shared by other users of the system. One of the most common use cases for e-SC is to provide a data analysis back end to a standalone desktop or Web application.

ClowdFlows [11] is a Cloud-based platform for the composition, execution, and sharing of interactive data mining workflows. Its service-oriented architecture allows using third-party services (e.g., Web services wrapping open-source or custom data mining algorithms). The server side consists of methods for the workflow editor to define workflows, and a relational database of workflows and data.

Data Mining Cloud Framework (DMCF) [12] allows users to program data analysis applications using two languages: VL4Cloud, a visual formalism for programming data analysis workflows graphically; JS4Cloud, a scripting language for programming workflows based on JavaScript. The DMCF's runtime fully exploits the Cloud infrastructure enabling workflow execution on multiple virtual machines.

BigML [13] is a SaaS for discovering predictive models from data sources and using data classification and regression algorithms. The distinctive feature of BigML is that predictive models are presented to users as interactive decision trees. The decision trees can be dynamically visualized and explored within the BigML interface, downloaded for local usage and/or integration with applications, services, and other data analysis tools.

Nubytics differs from general purpose data analysis frameworks like Azure ML, Hadoop and Sparks, or data-oriented workflow management systems like ClowdFlows and DMCF, as it provides specialized services for data classification and prediction. These services are provided by a Web interface that allows data analysts to focus on the data analysis process without worrying on low level programming details. This approach is similar to that adopted by BigML. However, Nubytics also focuses on scalability, by implementing an ad hoc parallel computing approach that fully exploits the distributed resources of a Cloud computing platform.

III. Architecture

The Nubytics architecture includes storage and compute components (see Figure 1). The storage components are:

- *Data Folder* that contains data sources and the results of data analysis, and *Tool Folder* that contains algorithms for data analysis and prediction.
- *Data Table*, *Tool Table* and *Task Table* that contain metadata information associated with data, tools, and tasks.
- *Task Queue* that contains the tasks to be executed.

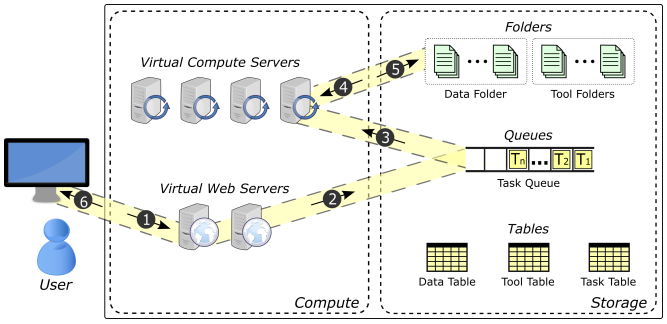


Fig. 1. System architecture and execution steps.

The compute components are:

- *Virtual Compute Servers* that execute the data analysis tasks.
- *Virtual Web Servers* that host the system front end, i.e., the Nubytics web interface.

The architecture manages submission and execution of data analysis tasks by the steps shown in Figure 1:

1) Using the services provided by the front end, the user can configure and submit the desired data analysis task (e.g., training a classification model from a dataset).
2) Exploiting a data parallel approach, the system models the task as a set of parallel sub-tasks that are inserted into the Task Queue for processing.
3) Each idle Virtual Compute Server picks a sub-task from the Task Queue and concurrently starts its execution.
4) Each Virtual Compute Server gets the part of data assigned to it from the Data Folder where the original dataset is stored.
5) After sub-task completion, each Virtual Compute Server puts the result on the Data Folder.
6) The front end notifies the user as soon as the task has completed, and allows her/him to access the results.

IV. Services

The Nubytics front end is divided into three sections - *Datasets*, *Tasks* and *Models* - corresponding to the three groups of services provided by the system: dataset management, task management and model management.

A. Dataset management

The datasets of a user are stored in a Cloud storage space associated to the user's account. The *Datasets* section provides several data management services, including: importing (uploading) a dataset from the user's terminal; exporting (downloading) a dataset to the user's terminal; listing and searching the available datasets; modifying the metadata of a dataset; creating a copy, deleting, or restoring a dataset.

As an example, Figure 2 shows a screenshot of the Datasets section showing some statistics on a dataset imported by a user. For each dataset field, the system shows name, type, number of instances present and missing, and a histogram representing the distribution of the instance values for that field.

B. Task Management

The *Tasks* section provide services for configuring, submitting and managing data analysis tasks. Two classes of tasks can be submitted: training tasks and prediction tasks.

A training task takes as input a dataset and produces a classification or regression model from it. The goal of classification is to derive a model that classifies the instances of a dataset into one or more classes. Using a classification model, we can predict the membership of a new data instance to a given class from a set of predefined classes. The goal of regression is to build a model that associates a numerical value to the instances of a dataset. Therefore, a regression model can be used to forecast a quantitative value starting from the field values of a new data instance.

The configuration of a training task is made by selecting the input dataset, the class field (which is categorical in case of classification and numerical in case of regression), and the predictive fields that must be considered for the analysis (they can be all - or a subset of - the original dataset fields). A parallel computing approach is used to speedup the execution of training tasks. This is done using a data parallel approach that divides the original task in sub-tasks, assigns a sub-task to a different virtual compute server on the Cloud, and joins the partial results computed by multiple servers into a single model. The scalability that is achieved with this approach is discussed in Section VI.

A prediction task takes two input elements: a model generated by a training task, and a new dataset whose instances must be classified or regressed. As a result, the new dataset will include a new field containing the predicted class label (in case of classification) or numerical value (in case of regression) of each tuple. Also in this case, parallelism is exploited by performing the prediction task in parallel on multiple Cloud servers.

Multiple tasks can be submitted to the system, and the user can monitor the status of each one through a task management interface, as shown by the screenshot in Figure 3. For each task, the interface shows the task type (prediction or training), some information about execution (start, end, and elapsed time), and the current status. Additional details on a task can be seen by selecting the corresponding row. For instance, the figure shows Input Dataset and Output Model of the second task, which is a Training task.

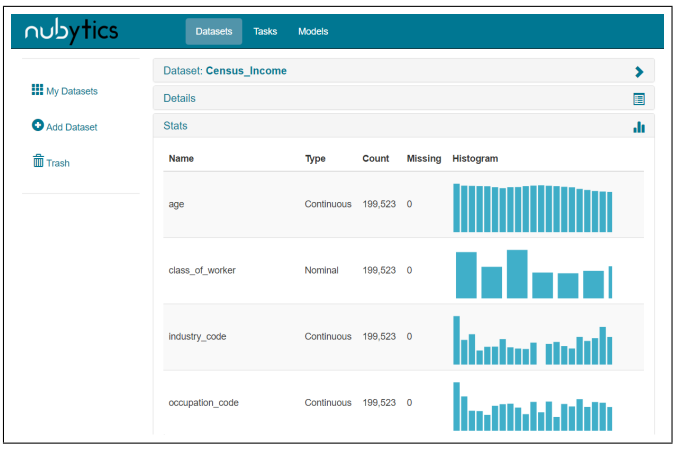


Fig. 2. Screenshot of the Datasets section.

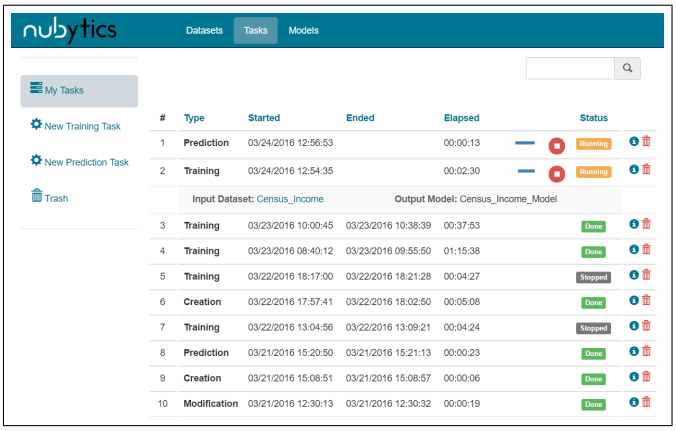


Fig. 3. Screenshot of the Tasks section.

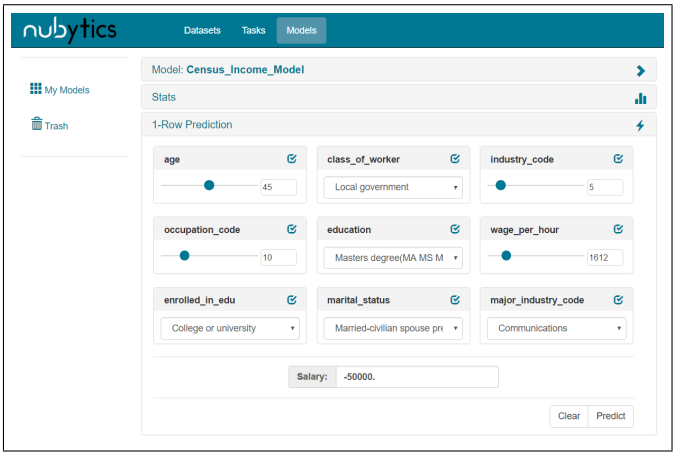


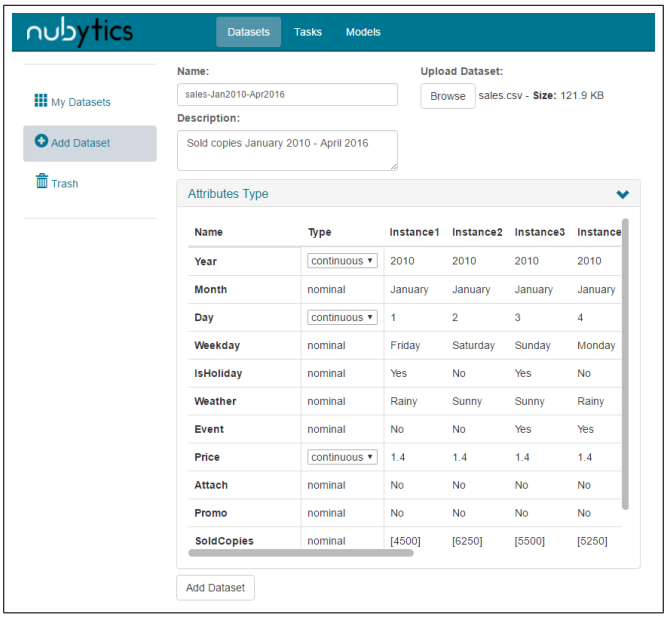
Fig. 4. Screenshot of the Models section.

C. Model Management

The *Models* section provides services for managing (e.g., searching, listing, etc.) the models previously generated by

	A	B	C	D	E	F	G	H	I	J	K
1	Year	Month	Day	Weekday	IsHoliday	Weather	Event	Price	Attach	Promo	SoldCopies
2	2010	January	1	Friday	Yes	Rainy	No	1.4	No	No	[4500]
3	2010	January	2	Saturday	No	Sunny	No	1.4	No	No	[6250]
4	2010	January	3	Sunday	Yes	Sunny	Yes	1.4	No	No	[5500]
5	2010	January	4	Monday	No	Rainy	Yes	1.4	No	No	[5250]
6	2010	January	5	Tuesday	No	Sunny	No	1.4	No	No	[5000]
7	2010	January	6	Wednesday	Yes	Sunny	No	1.4	No	Yes	[3250]
8	2010	January	7	Thursday	No	Sunny	No	1.4	No	No	[5000]
9	2010	January	8	Friday	No	Sunny	No	1.4	No	No	[5000]
10	2010	January	9	Saturday	No	Rainy	Yes	1.4	No	No	[6500]
11	2010	January	10	Sunday	Yes	Sunny	Yes	1.4	No	No	[5250]
12	2010	January	11	Monday	No	Sunny	No	1.7	Yes	No	[5000]
13	2010	January	12	Tuesday	No	Rainy	No	1.4	No	No	[4500]
14	2010	January	13	Wednesday	No	Rainy	No	1.4	No	No	[4500]
15	2010	January	14	Thursday	No	Sunny	No	1.4	No	No	[4750]
16	2010	January	15	Friday	No	Sunny	No	1.4	No	No	[4750]
17	2010	January	16	Saturday	No	Sunny	No	1.4	No	No	[6000]
18	2010	January	17	Sunday	Yes	Sunny	No	1.4	No	No	[4500]
19	2010	January	18	Monday	No	Sunny	No	1.4	No	No	[4750]
20	2010	January	19	Tuesday	No	Rainy	No	1.4	No	Yes	[4500]
21	2010	January	20	Wednesday	No	Rainy	No	1.4	No	No	[4500]
22	2010	January	21	Thursday	No	Sunny	No	1.4	No	No	[5000]
23	2010	January	22	Friday	No	Sunny	No	1.4	No	No	[5000]
24	2010	January	23	Saturday	No	Sunny	Yes	1.4	No	No	[7000]
25	2010	January	24	Sunday	Yes	Sunny	No	1.4	No	No	[4500]
26	2010	January	25	Monday	No	Sunny	No	1.7	No	No	[5000]
27	...	...	...	...	...	...	...	...	...	...	...

(*a*)



(*b*)

Fig. 5. Use case: *a*) Excerpt of the csv file; *b*) Dataset imported to the system.

training tasks. In addition to management functionalities, the section provides the *1-Row Prediction* service, which allows users to predict the class or the numerical value of a single data instance. This is a fast tool that can be convenient in case a user needs to make prediction on single tuples, rather than on entire datasets, in which case a standard prediction task should be submitted to the system. Figure 4 shows a screenshot of the 1-Row Prediction tool. After choosing a model, the tool shows to the user a panel containing the fields in the model. Then, the user chooses a value for each field and the system predicts the corresponding class.

V. Use case

As a real-life use case, let us assume that the distributor of a newspaper registered in a csv file the number of copies sold per day over recent years in a given city. A small excerpt of the csv file is shown in Figure 5*a*. As shown in the figure, some information were registered every day: *Weekday*; *IsHoliday* (yes/no); *Weather* (Rainy/Sunny); *Event* (yes/no); *Price*; *Attach* (yes/no); *Promo* (yes/no), *SoldCopies*.

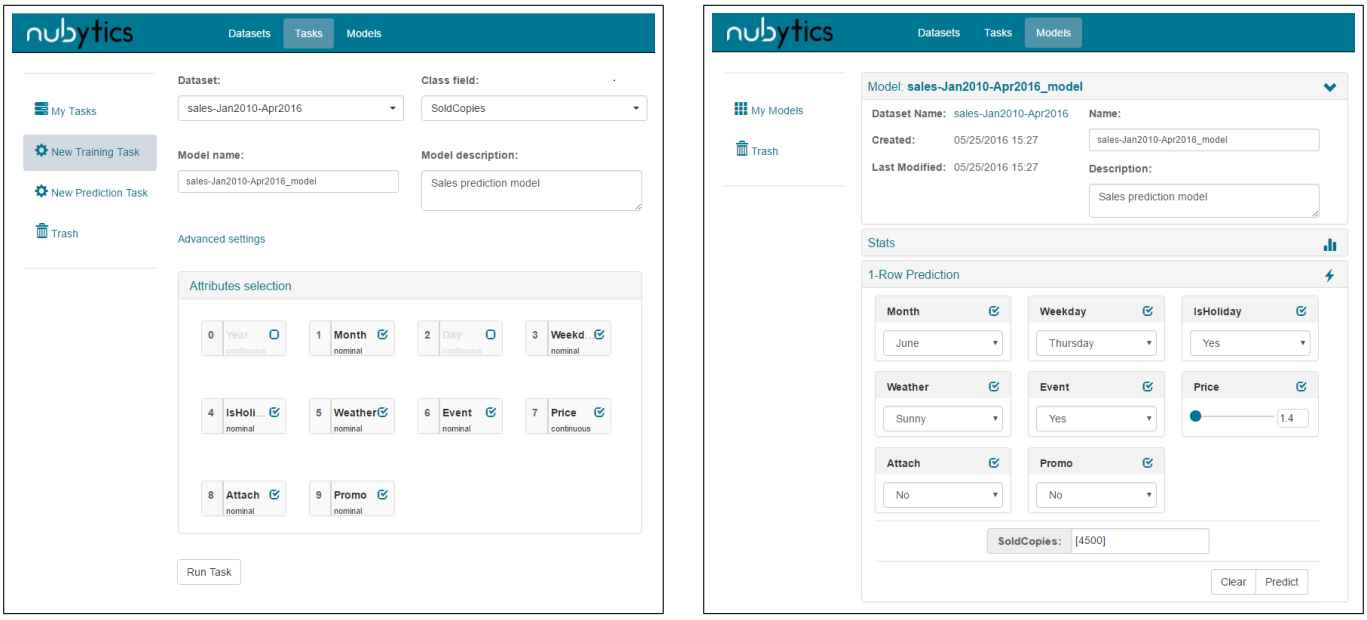
The goal of the distributor is using the data stored in the csv file to create a model for predicting how many copies will be sold in a future day, based on some infos about the day. Nubytics can be used to reach this goal in three steps:

1) The csv file is imported to Nubytics. Figure 5*b* shows a screenshot of the system after having imported the csv file. The user can also get some statistics about the dataset, i.e., histograms representing the distribution of the instance values for every field, using the functionality already shown in Figure 2.
2) A training task is submitted to create a predictive model, as shown in Figure 6*a*. The user specifies: the dataset used to train the model; the class field, i.e., which attribute must be predicted (in this case, *SoldCopies*); which attributes must be considered by the model (in this case, the user selected all the attributes except *Year* and *Day*). After submission, the user can monitor the task status through the management interface shown in Figure 3.
3) Once the predictive model is ready, it can be used to predict the number of sold copies in a future day. The *1-Row Prediction* tool can be conveniently used for the purpose. As shown in Figure 6*b*, the tool presents a panel containing the fields in the model, i.e., all the fields in the dataset except *Year* and *Day* that were excluded in the previous step. By inserting a value for each field, the system predicts the sold copies based on the model.

VI. Evaluation

In this section we present some experimental performance results obtained performing a training task for data classification using Nubytics, to show the scalability of the system in processing high volumes of data. The Cloud environment used for the experimental evaluation was composed of 128 virtual servers, each one equipped with a single-core Intel Xeon E5-2660 2.2GHz CPU, 3.5GB of memory, and 50GB of disk space. The measured bandwidth between two virtual servers is about 480 Mbps [14].

The input dataset used for the experiments has been generated from the *Census-Income Database* [15], which contains weighted census data extracted from the 1994 and 1995 Current Population Surveys conducted by the U.S. Census Bureau. Each instance corresponds to a US citizen and is described by 42 attributes, giving information about demographic and employment issues (e.g., *age*, *sex*, *education*, *class of worker*, *major occupation code*, etc.). The class attribute is the *income*



(a) (b)

Fig. 6. Use case: a) Submission of the training task; b) Use of the predictive model.

level for the person represented by the record, whose values are discretized in the two categories "-50000" and "50000+", i.e. representing an income lower or higher than $50,000$. The original dataset contains about 200,000 instances. By replicating the instances in the original dataset, we generated an input dataset containing 4.4 million instances with a total size of about 2,097 MB. The goal of the classification task on this dataset is to train a knowledge model, that can be used to predict the income level for a person described by the attributes.

To evaluate the efficiency and scalability of the Nubytics system, we report here some measures collected during the classification of the aforementioned dataset. More in detail, we evaluated the results through the following performance metrics:

- *Turnaround time*: the elapsed time from task submission until the final result is produce, varying the number of processing nodes (virtual servers) used to execute the task;
- *Speedup*: the ratio of the turnaround time on 1 processing node to the turnaround time on n nodes, which measures how much performance gain is achieved over a sequential implementation;
- *Efficiency*: the ratio between speedup and the number of processing nodes, which measures the percentage of time for which nodes are usefully exploited for computation (and not for communication tasks or even idling).

First, we measured the turnaround times of the classification task, using from 1 to 128 servers. Figure 7 shows such results, that can also be seen as a comparison between a sequential and parallel executions. In particular, the figure shows how the time required to execute the entire task strongly decreases with the increase of computing resources. For instance, the turnaround time decreases from about 5 hours required to obtain the data classification on a single node, to 1 hour and 15 minutes using 4 servers, to about 10 minutes using 32 servers, and to about 3 minutes using 128 servers. These values show a good performance of the system.

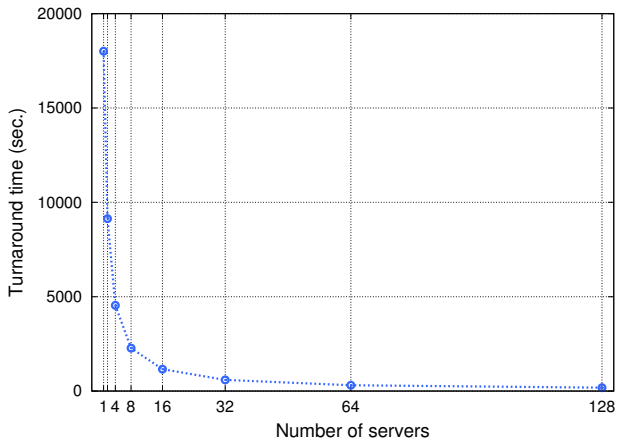


Fig. 7. Turnaround time vs number of servers.

Figure 8 plots the execution speedup values, for different number of nodes. As shown in the figure, the speedup is almost linear up to 32 nodes (the optimum/linear speedup is reported as a reference in the plot) and maintains a notable trend even for higher number of nodes. For instance, it reaches the values 15.51 on 16 servers and 30.48 for 32 servers. When the number of nodes is higher (64 or 128), the slope of the curve begins to decrease because the overhead time, mostly related to communications among the nodes, becomes not

negligible with respect to the processing time. However, the system achieves speedup values equal to 58.48 and 101.19 on 64 and 128 nodes, respectively. These results confirm the high scalability of the system.

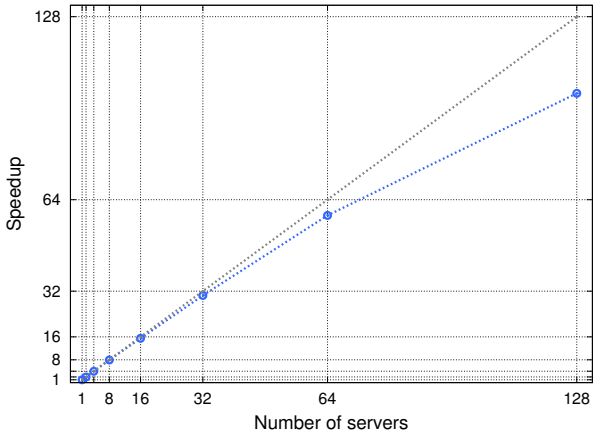


Fig. 8. Speedup vs number of servers.

Finally, Figure 9 reports the efficiency versus the number of nodes. It is worth noting that it assumes values higher than 0.95 up to 32 nodes, that is, more than 95% of the computing power of each used server is exploited. The efficiency decreases to 0.91 on 64 servers and 0.79 on 128 servers. Overall, this is a notable result and is a another sign of good scalability properties.

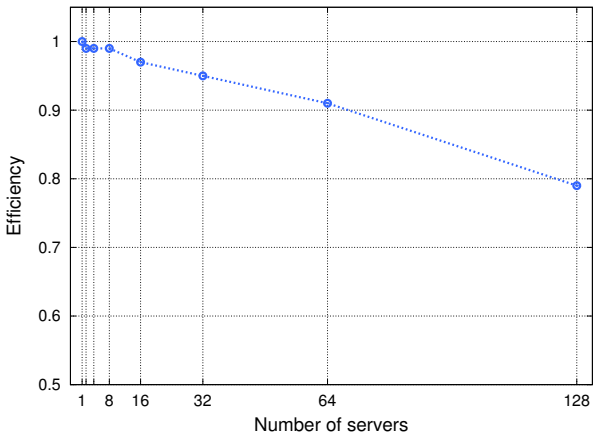


Fig. 9. Efficiency vs number of servers.

VII. Conclusion

Data analysis techniques are increasingly being used to extract useful knowledge from large datasets in many application areas, from science to business. The analysis of very large data sources requires scalable systems to reduce execution time and make the Big Data paradigm viable. Cloud technologies can be effectively exploited to provide end users with the computing and storage resources, and the execution mechanisms needed to efficiently run complex data analysis tasks.

Nubytics is a system that exploits Cloud facilities to provide scalable services for analyzing large datasets. It allows users to import their data to the Cloud, discover knowledge models using high performance data mining services, and use the inferred knowledge to predict new data. The system provides data classification and regression functionalities that can be used in a variety of application scenarios. Nubytics is provided in accordance with the SaaS model, which is a key advantage for users who need ubiquitous and seamless access to scalable data analysis services.

To ensure scalability on large datasets, Nubytics implements a parallel computing approach that fully exploits the resources available on the Cloud. The experimental results demonstrate the effectiveness of the approach in processing high volumes of data. The experiments presented in the paper show that the speedup is almost linear up to 32 virtual servers and maintains very goods trends even for higher number of nodes. This allows Nubytics to significantly reduce the processing time on large datasets, which is fundamental to deliver results in a timely way to business users.

Nubytics is available at https://www.nubytics.com.

References

[1] D. Talia, P. Trunfio, and F. Marozzo, *Data Analysis in the Cloud*. Elsevier, October 2015.

[2] "Apache Hadoop," http://hadoop.apache.org, accessed: June 2016.

[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[4] "Apache Spark," http://spark.apache.org, accessed: June 2016.

[5] "Azure ML," http://azureml.net, accessed: June 2016.

[6] J. Goecks, A. Nekrutenko, J. Taylor, and T. G. Team, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome Biology*, vol. 11, no. 8, p. R86, 2010.

[7] K. Maheshwari, A. Rodriguez, D. Kelly, R. Madduri, J. Wozniak, M. Wilde, and I. Foster, "Enabling multi-task computation on galaxy-based gateways using swift," in *IEEE Int. Conference on Cluster Computing*, 2013, pp. 1–3.

[8] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher *et al.*, "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud," *Nucleic Acids Research*, vol. 41, pp. 557–561, 2013.

[9] M. Abouelhoda, S. Issa, and M. Ghanem, "Tavaxy: Integrating taverna and galaxy workflows with cloud computing support," *BMC Bioinformatics*, vol. 13, no. 1, 2012.

[10] H. Hiden, S. Woodman, P. Watson, and J. Cala, "Developing cloud applications using the e-Science Central platform," *Philosophical Transactions of the Royal Society A: Mathematical,Physical and Engineering Sciences*, vol. 371, no. 1983, 2013.

[11] J. Kranjc, V. Podpečan, and N. Lavrač, "ClowdFlows: A Cloud Based Scientific Workflow Platform," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science. Springer, 2012, vol. 7524, pp. 816–819.

[12] F. Marozzo, D. Talia, and P. Trunfio, "Js4cloud: Script-based workflow programming for scalable data analysis on cloud platforms," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 5214–5237, December 2015.

[13] "BigML," https://bigml.com, accessed: June 2016.

[14] F. Rodrigo Duro, F. Marozzo, J. Garcia Blas, D. Talia, and P. Trunfio, "Exploiting in-memory storage for improving workflow executions in cloud platforms," *The Journal of Supercomputing*, 2016.

[15] "Census income dataset," https://archive.ics.uci.edu/ml/machine-learning-databases/census-income-mld/census-income.data.html, accessed: June 2016.