A New Cloud Computing Architecture for the Classification of Remote Sensing Data

Victor Andres Ayma Quirita, *Student Member, IEEE*, Gilson Alexandre Ostwald Pedro da Costa, *Member, IEEE*, Patrick Nigri Happ, *Member, IEEE*, Raul Queiroz Feitosa, *Member, IEEE*, Rodrigo da Silva Ferreira, Dário Augusto Borges Oliveira, and Antonio Plaza, *Fellow, IEEE*

***Abstract*—This paper proposes a new distributed architecture for supervised classification of large volumes of earth observation data on a cloud computing environment. The architecture supports distributed execution, network communication, and fault tolerance in a transparent way to the user. The architecture is composed of three abstraction layers, which support the definition and implementation of applications by researchers from different scientific investigation fields. The implementation of architecture is also discussed. A software prototype (available online), which runs machine learning routines implemented on the cloud using the Waikato Environment for Knowledge Analysis (WEKA), a popular free software licensed under the GNU General Public License, is used for validation. Performance issues are addressed through an experimental analysis in which two supervised classifiers available in WEKA were used: random forest and support vector machines. This paper further describes how to include other classification methods in the available software prototype**

***Index Terms*—Distributed computing, image classification, remote sensing.**

I. Introduction

THE amount of data available from remote sensing systems is increasing at an extremely fast pace due to the recent advances in modern Earth Observation technologies [1]–[3]. Hundreds of remote sensing satellites are now in orbit, acquiring large amounts of information about the Earth's surface every day. Improvements related to spatial resolution, revisit frequency, and number of spectral bands are the main drives of this increasing data availability. For instance, Sentinel-1, from the European Space Agency, alone generates about 1.5 GB per day [4], and the NASA EOSDIS project produces about 16 TB of data per day [5]. This scenario leads to new challenges, related to the capability of handling such huge volumes of data [6], [7], with respect to computational techniques and resources.

In this sense, remote sensing data handling may be considered as a big data problem, due to the high volume (TB/day), the variety (radar, optical images, etc.), and the generation velocity of the data to be processed [8], [9]. Fortunately, dealing with big data is currently a common problem faced by different fields at industry and research centers. In this context, cloud computing is a trend [10] since it delivers a powerful infrastructure to perform large-scale computing, which is usually available in a "pay-as-you-go" model, and which alleviates users from the need to purchase and maintain complex computing hardware.

Regarding machine learning, there are many available techniques for both supervised and unsupervised classification processes, which are capable of analyzing small- to medium-size datasets [11]. However, few works focusing on handling big datasets can be found in the literature. An overview of some of the machine-learning approaches for handling very large datasets can be found in [12], those approaches are, however, more concerned with the training than with the classification procedure itself.

In this study, we describe a new cloud computing architecture, which is designed to enable supervised classification processes on large volumes of remote sensing data. The architecture supports distributed execution, network communication, and fault tolerance capabilities in a way that is completely transparent to the user. The architecture contains three layers, which provide different abstraction levels and can be implemented independently. Each layer targets a different user type, according to their particular expertise.

The proposed architecture, named *InterCloud Data Mining Architecture*, is ready for cloud computing environments, allowing users to elastically allocate processing power and storage space so to effectively handle very large datasets in the order of petabytes. Thus, it enables the efficient use of available computational resources by scaling them up, at low costs, depending on the size of the problem.

In addition to the architecture description, another important contribution of this study is that it describes how to extend it by

Manuscript received February 16, 2016; revised July 4, 2016; accepted August 16, 2016. Date of publication September 13, 2016; date of current version January 23, 2017. This work was supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior and by the European Union Seventh Framework Programme for Research and Technological Development as a part of the Tools for Open Multi-Risk Assessment Using Earth Observation Data Project. *(Corresponding author: Victor Andres Ayma Quirita).*

V. A. A. Quirita, P. N. Happ, and R. Q. Feitosa are with the Electrical Engineering Department, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro 22451-000, Brazil (e-mail: vaaymaq@ele.puc-rio.br; patrick@ele.puc-rio.br; raul@ele.puc-rio.br).

G. A. O. P. da Costa is with the Informatics and Computer Science Department, State University of Rio de Janeiro, Rio de Janeiro 20550-900, Brazil (e-mail: gilson.costa@ime.uerj.br).

R. S. Ferreira is with the IBM Research Brazil Lab, Rio de Janeiro 22290-900, Brazil (e-mail: rosife@br.ibm.com).

D. A. B. Oliveira is with the General Electric Global Research Center, Rio de Janeiro 21941-600, Brazil (e-mail: darioaugusto@gmail.com).

A. Plaza is with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, Cáceres E-10071, Spain (e-mail: aplaza@unex.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTARS.2016.2603120

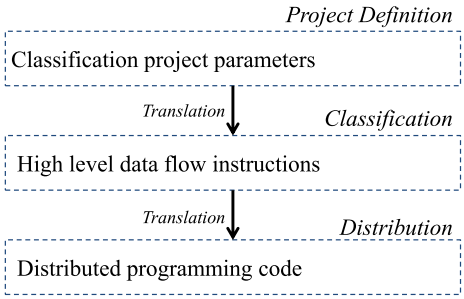

Fig. 1. InterCloud data mining architecture.

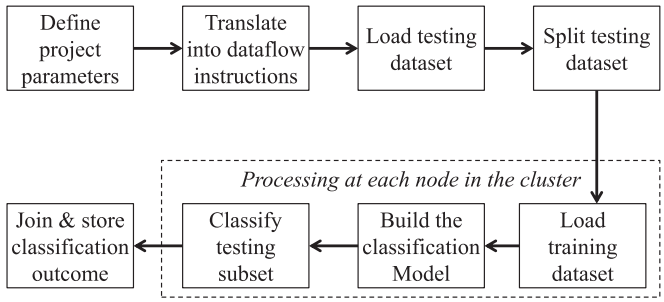

Fig. 2. InterCloud data mining general outline.

integrating different classification algorithms. We also validate experimentally the architecture through a software prototype (available at http://www.lvc.ele.puc-rio.br/wp/?p = 1861) that contains some of the classification algorithms present in the Waikato Environment for Knowledge Analysis (WEKA), a popular free software licensed under the GNU General Public License [13]. The results are analyzed in terms of speedups, relative to the processing of different dataset sizes.

The remainder of this paper is organized as follows: Section II describes the proposed architecture. Section III describes an implementation of the architecture, as well as the adopted frameworks. The guidelines to extend the implementation of the architecture with other classification algorithms are presented in Section IV. Two study cases are presented as experimental validation in Section V and, finally, conclusions are summarized in Section VI.

II. InterCloud Data Mining Architecture

InterCloud Data Mining can be regarded as an independent module of *InterCloud*—a distributed platform for automatic interpretation of large remote sensing datasets [14].

InterCloud Data Mining architecture was designed to support the interaction between machine learning algorithms and large datasets, through the distribution of data and processing (classification) tasks among machines connected through a network.

As shown in Fig. 1, the architecture contains three abstraction layers: the project definition layer; the classification layer; and the distribution layer.

The *project definition layer* should support end user interaction. By end user we mean a user that does not need to have programming skills. The information to be provided by the user through the project definition layer comprises what we call the *classification project*, which contains all the information required for the execution of the classification application, namely the classification algorithm to be used; the parameter values of such algorithm; the number of instance (processing) nodes to be allocated in the cloud computing environment; the location of the training and classification (testing) datasets; and any other cloud specific settings.

Through the next layer, *the classification layer*, users with conventional programming skills (as opposed to users with distributed programming skills), should be able to embed new classification algorithms into an implementation of the architecture, so that a user interacting with the *project definition layer* would be able to later select those algorithms. The classification layer is structured on a high-level programming language that hides the complexity of dealing directly with the distributed programing model.

The interface between the project definition layer and the classification layer is defined as a *translation process*. This process is responsible for translating the classification project into processing and data flow instructions, coded in the *classification layer*'s programming language.

The *distribution layer* is responsible for the distributed execution of the classification applications, and should be managed by users that are experts in distributed programming models.

The interface between the *classification layer* and the *distribution layer* is defined as another translation process. This process is responsible for translating the (high level) instructions defined in the *classification layer* into distributed programming code, necessary to run classification applications in a distributed fashion.

Therefore, each abstraction layer contains a different representation of the classification application, in a different level of abstraction.

In order to process the distributed classification, it is necessary first to set the lower layers: *distribution* and *classification*, which are described in the next section. Then, it is possible to rely solely on the *project definition layer* to define and start the execution of the classification processing chain.

According to Fig. 2, the distributed classification processing chain is started after the classification project definition. First, the project definitions are translated into data flow instructions, according to the classification layer definitions. Then, the classification dataset is split into smaller disjoined subsets; a link to the training dataset is configured and the classification algorithm is loaded in each processing node. In sequence, the classification algorithm is executed in a distributed fashion.

The *classification model* can be built using a training set sampled from the complete training dataset in some way. Note that if the training process carried out in each node uses the same set of training samples, e.g., the whole training dataset, the classification models in all nodes should be similar, if not identical. Next, each node classifies one subset of the classification dataset independently. Finally, all parts of the classified dataset are joined and stored into a cloud repository as a single classification outcome.

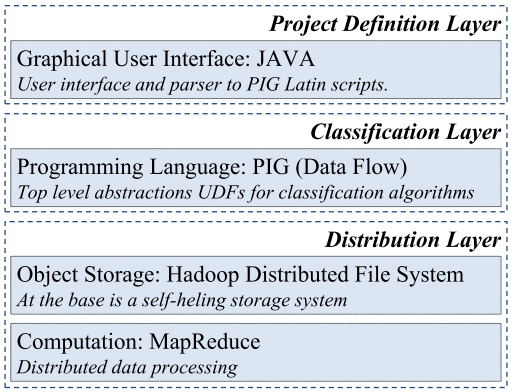

Fig. 3. InterCloud data mining implementation choices.

III. InterCloud Data Mining Implementation

The realization of the *InterCloud Data Mining* architecture is accomplished by the implementation of its different layers and translation processes using concrete programming frameworks. There are many alternatives, and in this section, we describe a particular implementation, as shown in Fig. 3, used for experimental validation.

The distribution layer should be based on a framework that supports processing and data distribution. In this case, we choose MapReduce [15], one of the most popular programming models for processing very large datasets [16], and its open-source implementation in Apache Hadoop. Hadoop is a widely used framework for distributed processing of large datasets across clusters of computers. Hadoop is based on two main components: the distributed file system (HDFS) and the MapReduce programming model. HDFS is designed and optimized for high processing performance and works best with large files (gigabytes or larger) [17]. Hadoop MapReduce is a programming model based on a simple data processing paradigm composed of three main phases: map, shuffle, and reduce [18].

Considering the classification layer, an intermediary framework that interfaces with the distribution layer must be defined. This framework should also provide an easy way to instantiate custom (or user defined) functions. Therefore, we choose the Pig framework for the implementation of the classification layer.

The Pig framework provides a language for expressing data flows, called Pig Latin, and an engine that compiles Pig Latin scripts into MapReduce jobs. Pig Latin makes it easy for programmers to interact with MapReduce, as it is a high level and extensible programming language [19]. User-defined functions (UDFs) provide this extension capability, enabling the integration of external libraries and scripts created by third party developers, and, thus, offering an easy and efficient way to include new functionalities into the Pig framework.

The project definition layer was implemented as a graphical user interface (GUI) coded in the Java programming language. Through the GUI, the user can set all the necessary definitions for the execution of the classification application. A parser, also written in Java, is responsible for the translation of the classification project into Pig Latin scripts. These scripts are in turn translated into MapReduce instructions, through the Pig framework.

Lastly, each classification algorithm is structured as a Pig UDF, coded in Java, so that they can be called from Pig Latin scripts. As mentioned before, this implementation of the proposed architecture can be extended with the addition of (external) classification algorithms. The next section explains how to do that.

IV. Guidelines for Integrating Classification Algorithms

Integrating classification algorithms depends on the particular implementation of the architecture. As in our case, we used the Hadoop and Pig frameworks, two main steps are required: embedding the classification algorithm in a Pig UDF and creating a Pig Latin script that calls the UDF.

The structure of a classification UDF is presented in Algorithm 1. It requires as inputs: 1) the absolute path to the training dataset (previously allocated on the cloud); 2) the options (parameter values) of the classification algorithm to be used; and 3) the testing dataset, which is a data bag containing all the tuples (feature vectors) to be classified.

Algorithm 1:–Structure for designing the Classification UDF.

1: Get the absolute path (URL) to the training dataset.
2: Establish the URL connection for stream reading.
3: Buffer the input training data in local memory.
5: Select the training samples.
6: Get the options for the classification algorithm.
7: Create the classification model.
8: Classify the incoming data from the test dataset.
9: Return the classified data.

Since training data are stored in an auxiliary repository on the cloud, its corresponding URL should be provided in order to establish the connection for streaming the data to the local memory of each node. Once the connection to the training dataset is established, the training samples that will be later used for training the classifier in each cluster node should be selected. At this point, the developer may choose to select all training patterns in the training dataset, or a particular subset of it. In the implementation of the architecture presented in this study, the first alternative was chosen, so that the same classification model is created in each node. The options for the classification algorithm are then read and set.

The next step is the creation of the classification model using the training samples and the classification algorithm parameter values. In sequence, the classification model is used for classifying each incoming tuple from the subset of the testing dataset. Note that each subset of the testing dataset is automatically generated by the distributed framework, through a MapReduce instruction, and that the testing subsets are disjoint. Finally, the classification output is joined and stored in the cloud repository by the distributed framework. Fig. 4 shows the general processing chain of the classification process.

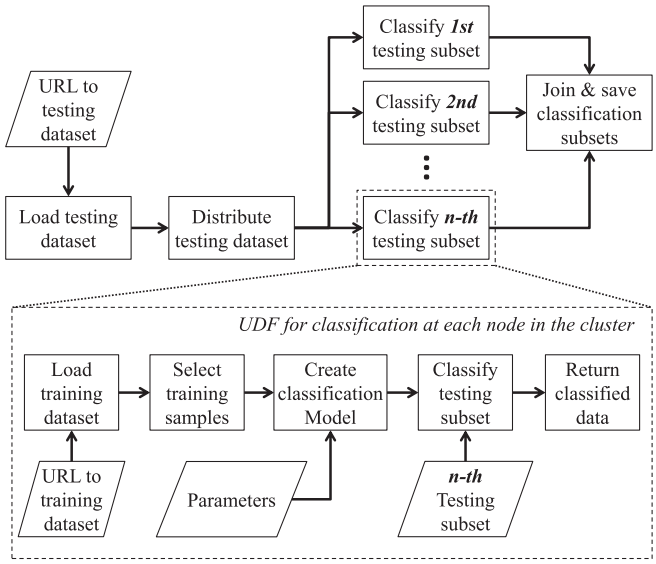

Fig. 4. General processing chain of the classification process.

A Pig Latin script is responsible for controlling the entire classification process, by defining the training and testing datasets, selecting the classification algorithm, and storing the classification outcome in a given repository, as shown in Algorithm 2. The script contains instructions for registering the classification UDFs and any other required libraries. Then, the selected classifier should be defined with its corresponding parameters and with the absolute path to the training dataset. In sequence, the testing dataset should be loaded and the classification UDF called to classify each tuple, in a distributed way. Finally, in a reduction step, the outcomes of the distributed classification processes are joined into a single output which is stored in a given repository on the cloud.

Algorithm 2:–Pig Latin script for definition of the Classification Process.

1: REGISTER the path to *UDF-Classifier* files.
2: REGISTER the path to *Libraries* files.
3: DEFINE the classifier to be used
 - Define the path to training dataset.
 - Define the classifier parameters.
4: LOAD the test dataset.
5: FOREACH tuple in the test dataset GENERATE a classification output by calling the Classification UDF.
6: REDUCE the classification outputs.
7: STORE the classification output.

V. Experimental Design and Results

To evaluate the architecture and its implementation, described in previous sections, we carried out two series of experiments, with two different hyperspectral images. This section reports the experimental analysis carried out on both tests.

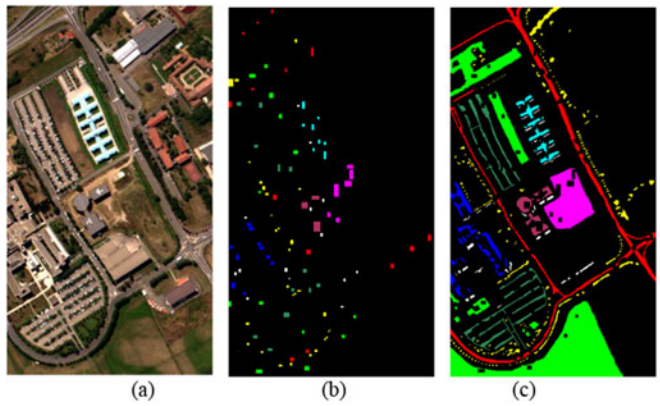

Fig. 5. Pavia hyperspectral image: (a) False color composition. (b) Training dataset. (c) Testing dataset.

A. Cloud Environment

The experiments were conducted on Amazon Web Services, which is a combination of commercial services for cluster processing. Amazon Simple Storage Service was used to store the input and output data, as well as the programs and libraries required. Amazon Elastic MapReduce was used to manage the Hadoop framework in order to distribute and process the data across the cluster that was dynamically built using Amazon Elastic Compute Cloud instances.

For each experiment, clusters with increasing number of nodes were used, starting with 2 (baseline), 5, 10, 20, and 50 nodes each time. The machine nodes were m3.xlarge type. They contain an Intel Xeon E-5-2670 v2 processor operating at 2.5 GHz with 4 physical cores (8 logical cores), 15 GB of RAM, and two disks of 40 GB. The Hadoop and Pig versions used were, respectively, 2.4 and 0.12.

It is important to notice that one node is always considered master and it is reserved for the Hadoop JobTracker, which is responsible for scheduling and managing the tasks, and is not available for performing the classification.

B. Dataset

Two hyperspectral images were used in the experiments: Pavia and Indian Pines. The Pavia image [20] (see Fig. 5(a)) was collected by the ROSIS optical sensor over the University of Pavia, Italy. The image contains 610 $\times$ 340 pixels with 1.3-m spatial resolution and 103 spectral bands. The target classes comprise 21 urban, soil, and vegetation classes. Fig. 5(b) and (c) shows, respectively, the extents of the reference training and testing sets of pixels. Only the first nine principal components, computed for each testing reference pixel were used for classification, yielding a 20-Mb data file. From this original testing dataset, synthetic datasets were built by replicating it 100, 200, and 500 times, yielding data files with around 2, 4, and 10 Gb, respectively.

The Indian Pines hyperspectral image [21], shown in Fig. 6(a), is an airborne visible infrared imaging spectrometer (AVIRIS) image over an area with forest and mixed agriculture, in northwestern Indiana, USA. The image contains 610 $\times$ 2678

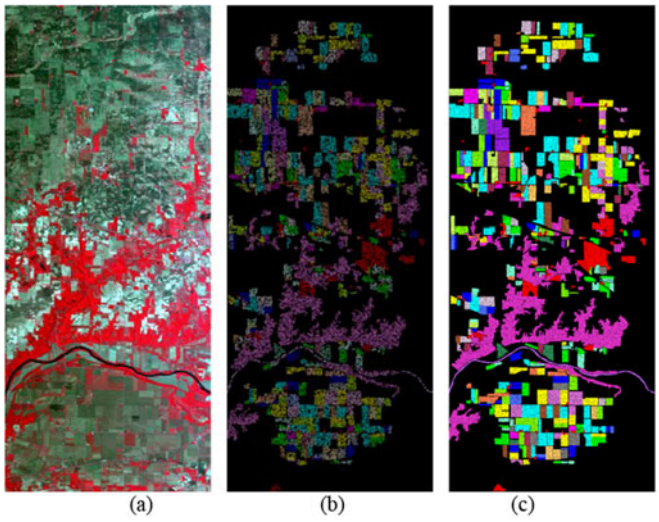

Fig. 6. Indian Pines hyperspectral image: (a) False color composition. (b) Training dataset. (c) Testing dataset.

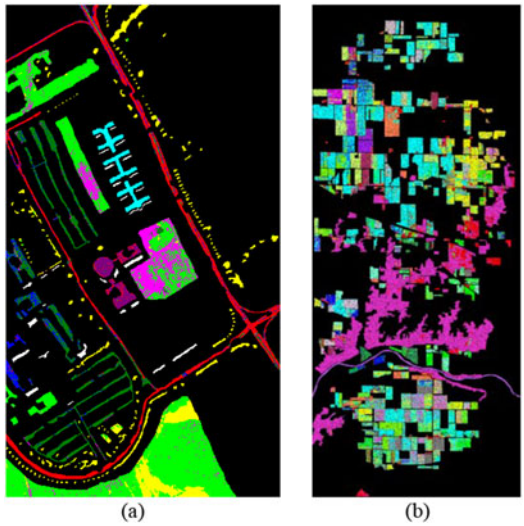

Fig. 7. (a) SVM classification outcome for Pavia hyperspectral dataset. (b) Random forest classification outcome for Indian Pines dataset.

pixels with 20-m spatial resolution and 202 spectral bands. The classes comprise 21 land cover classes. The training and testing reference pixels' extents are shown in Fig. 6(b) and (c). The 34 first principal components were used for classification, yielding a 400-Mb data file. As in the previous case, synthetic testing datasets were built by replicating the original one 5, 10, 30, and 50 times, producing files of around 2, 4, 12, and 20 Gb, respectively.

It should be noted that in both cases, replication was only done in the spatial dimension, so that the resulting testing datasets represent image mosaics with the same number of bands (9 and 34, respectively). Furthermore, refering to Section II, the subsets of the testing datasets represent disjoined cuts of the corresponding images.

C. Classification Methods

The current implementation of *InterCloud Data Mining* contains four supervised classification algorithms from the Weka environment: Näive Bayes, Decision Trees, Random Forest, and Support Vector Machines (SVM) [13]. In our tests, we used the last two algorithms: Random Forest for Indian Pines dataset and SVM for Pavia dataset. The parameters corresponding to classification algorithms were configured according to [13], as follows: 1) the Random Forest has 100 trees and a fixed random seed; 2) for SVM, a multiclass pairwise classification with polynomial function kernel was set, with complexity parameter $C = 1.0$ and exponent value $\gamma = 1.0$, over a fivefold cross-validation procedure.

D. Results

As mentioned, all the experiments were performed using a particular *InterCloud Data Mining* implementation, which is freely available at http://www.lvc.ele.puc-rio.br/wp/?p = 1861 (Ayma *et al.* [22]). The reported results represent the average of ten executions of each combination: dataset—classification algorithm—number of cluster nodes.Fig. 7 shows the classification outcome for each hyperspectral image. The classification (overall) accuracies were: 78,26% for the Pavia dataset and 64,41% for the Indian Pines dataset. As expected, the same classification accuracies were obtained with any corresponding replicated versions of the datasets.

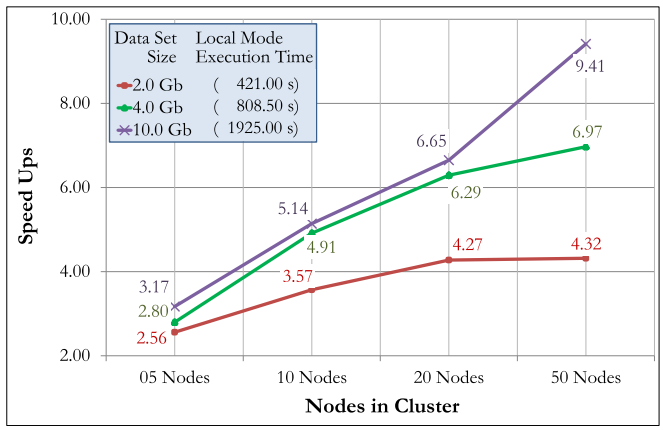

Fig. 8. Speedups for the SVM classification process on the Pavia dataset.

–Fig. 8 shows the speedups achieved on the Pavia dataset. For the first dataset size (2 Gb), the speedups were 2.56, 3.57, 4.27, and 4.32, for 5, 10, 20, and 50 nodes, respectively. The graph also shows that as the dataset got bigger, each cluster configuration produced better speedups. These facts are related to the exploitation of the distributed environment. Smaller data volumes imply lower scalability potential, whereas bigger data volumes allow for higher speedups, since data are distributed over more nodes.

It is elucidative to look at the results obtained for different data volumes and a fixed number of nodes. The speedups were almost constant when five nodes were used. However, as more nodes were added to the cluster, the speedups improved considerably as larger volumes of data were processed. For example, with

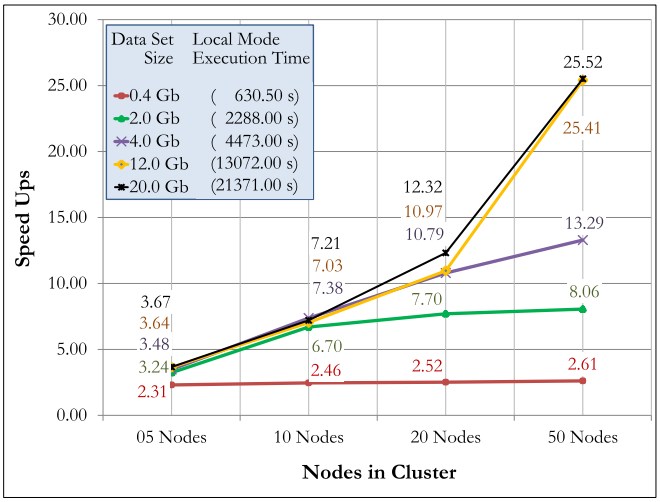

Fig. 9. Speedups for the random forest classification process on the Indian Pines dataset.

50 nodes, the speedup grew from 4.32 to 6.97 and then to 9.41 as the size of the classification dataset (replicated versions) increased. Again, this can be explained by the underuse of the cluster resources: 2 Gb is a considerable amount of data for a five nodes cluster, but is a small volume for a cluster containing 50 nodes.

Similar conclusions can be drawn from Fig. 9, which shows the speedups reached on the AVIRIS Indian Pines dataset. These experiments confirmed that smaller dataset sizes result in lower speedup values and, as the dataset size is increased, the speedup also increases. Observing the curve for the 0.4-Gb dataset, the speedup was almost constant varying from 2.31 (five nodes) to 2.61 (50 nodes). When the size increased to 4 Gb, for instance, the speedup behavior was almost linear going from 3.48 (five nodes) to 13.29 (50 nodes). Once again, when the dataset size was further enlarged, the speedup presented an exponential growing trend. In the case of the 20-Gb dataset size, speedup values ranged from 3.67 (five nodes) to 25.52 (50 nodes).

A similar behavior, with respect to the Pavia dataset experiments, can be observed by looking at a fixed cluster size. For a five-node cluster, the increment of data size did not result in substantial increment of the speedup. Using ten nodes, the difference became significant only for 0.4 Gb. This trend was also noted for 20 and 50 nodes. With 20 nodes, a large difference for the two lower data sizes was observed. With 50 nodes, a significant difference was observed for almost all data sizes, except for the two larger ones. This can be explained by considering the balance between data size and number of nodes, i.e., for each number of nodes, there is a limit related to distributed capacity that restricts the speedup increment. For instance, the 50 nodes configuration seems to have reached a saturation limit for the 12-Gb dataset.

Table I shows the average processing times for (from top to bottom) reading the training data, training the classifier, classifying the test data, and finally, the time for aggregating and storing the classification outcome on the cloud. As expected, in all cases, the time for reading the training data was very low relative to the overall processing time. The table also shows that the time spent in the classification step decreases quickly as more nodes are added, whereas the times involved in the other steps do no vary substantially.

TABLE I
AVERAGE PROCESSING TIMES FOR EACH STEP OF THE CLASSIFICATION PROCESS ON THE CLOUD ENVIRONMENT (IN S)

Dataset	02 Nodes	05 Nodes	10 Nodes	20 Nodes	50 Nodes
Pavia 2.0 Gb	2.5	2.8	2.2	2.1	2.1
	2.4	3.6	1.9	1.2	0.9
	344.6	81.9	41.3	28.1	22.7
	36.8	40.9	43.1	40.8	43.3
Pavia 4.0 Gb	2.6	2.8	2.9	2.2	2.0
	2.3	3.6	3.4	1.6	1.1
	700.4	174.7	62.6	33.3	24.9
	69.0	75.5	62.1	65.2	64.5
Pavia 10.0 Gb	2.4	2.9	2.9	3.0	2.2
	2.3	3.5	3.8	3.8	1.7
	1749.3	433.0	201.3	109.8	40.7
	136.4	134.5	134.6	139.9	132.9
Indian Pines 0.4 Gb	3.2	2.7	3.6	2.6	2.9
	271.7	171.6	159.2	160.0	158.6
	307.5	57.9	52.7	45.8	41.5
	16.5	19.1	18.5	20.7	16.9
Indian Pines 2.0 Gb	3.3	3.9	3.2	2.8	2.7
	247.6	390.6	192.4	168.0	159.0
	1960.9	238.2	76.4	58.2	53.7
	44.5	42.9	43.9	45.3	46.1
Indian Pines 4.0 Gb	3.4	4.0	3.6	2.9	2.7
	245.9	379.1	355.7	191.0	164.0
	4116.4	796.1	132.1	74.7	68.1
	76.0	74.2	82.0	73.9	79.0
Indian Pines 12.0 Gb	3.4	3.8	3.9	4.0	3.2
	243.9	368.6	383.2	382.5	199.0
	12512.8	2990.7	1234.9	574.9	82.8
	282.0	192.2	204.8	195.9	202.9
Indian Pines 20.0 Gb	3.4	3.9	4.0	4.0	3.6
	243.3	368.2	386.3	405.9	326.2
	20781.9	5120.6	2239.7	980.7	168.9
	300.1	301.9	303.3	311.2	306.7

At each cell, from top to bottom, elapsed times for: a. Read training data at each node. b. Build the classification model. c. Classify the testing subsets. d. Join and store the classification outcome.

VI. CONCLUSION

In this paper, we presented a cloud computing-based architecture for classification of very large remote sensing datasets, which we named *InterCloud Data Mining* architecture. An implementation of the proposed architecture was created for validation. Such implementation exploits the benefits of working on clusters with the Hadoop framework, providing a robust and flexible platform that allows working with big datasets on distributed infrastructures. Additionally, we give guidelines on how to extend the implementation of the architecture through the addition of classification algorithms.

The experimental analysis, carried out with the *InterCloud Data Mining* implementation described in this paper, demonstrated the scalability of the proposed architecture and its potential to handle big datasets. As the dataset increases, adding more nodes provides for more efficient processing.

It was also observed that the speedups increase with the amount of data being processed and the number of nodes available. As the size of the dataset increases, clusters containing more nodes deliver higher speedups. This is because, for larger datasets, the distributed resources can be better exploited, resulting in higher parallelization. Additionally, the results show that the implementation of the architecture presented here is a stable platform, in which the processing time depends essentially on the size of the dataset and the number of nodes in the cluster.

The results also demonstrated that increasing the number of nodes in the cluster does not necessarily provide a corresponding reduction of execution times. Thus, to optimize computational performance, there must be a balance between the amount of data to be processed and the number of nodes to be used. Furthermore, the optimum cluster configuration depends not only on the operations to be executed but also on the amount of input data to be processed.

Considering that the classification model is built on each cluster node, using the same training dataset and the same parameter values for the classification algorithm, the classification outcome should be similar, if not identical, regardless of the number of cluster nodes used. However, identical results as the ones obtained by the corresponding sequential version of the classification application (performed on a single processing unit) are only guaranteed when the classification algorithm chosen has no stochastic behavior or if such behavior is somehow controlled. The Random Forest classification procedure used in this study, for instance, has a parameter to control the random number generator, which was configured in order to produce consistent identical classification results in our experiments.

Although the architecture is not limited to problems for which training dataset sizes are considered small when compared to the size of the dataset to be classified, those are the problems that can best benefit from the particular implementation of the architecture presented in this study.

REFERENCES

[1] M. Ullah *et al.*, "Real-time big data analytical architecture for remote sensing application," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 8, no. 10, pp. 4610–4621, Oct. 2015.

[2] M. Datcu,"HD-03: Big data from earth observation: Analytics, mining and semantics," in *IEEE Int. Geosci. Remote Sens. Symp.*, Milan, Italy, 2015. [Online]. Available: http://www.igarss2015.org/Tutorial_HD3.asp

[3] L. Zhang, Q. Du, and M. Datcu,"Special section guest editorial: Management and analytics of remotely sensed dig bata," *J. Appl. Remote Sens.*, vol. 9, no. 1, pp. 1–2, Jul. 2015.

[4] O. Grabak, "Sentinel-1 mission status," in *15th Meet. Int. Ice Charting Working Group*, 2014. [Online]. Available: ftp://sidads.colorado.edu/pub/projects/noaa/iicwg/IICWG-2014/Grabak_Sentinel-1_Mission_Status.pdf

[5] NASA EARTHDATA. (2015, Feb.). EOSDIS Annual Metrics Reports. [Online]. Available: https://earthdata.nasa.gov/about/system-performance/eosdis-annual-metrics-reports

[6] J.-G. Lee and M. Kang, "Geospatial big data: Challenges and opportunities," *Big Data Res.*, vol. 2, pp. 74–81, Jun. 2015.

[7] D. Kishor, "Big data: The new challenges in data mining," *Int. J. Innov. Res. Comput. Sci. Technol.*, vol. 1, no. 2, pp. 39–42, Sep. 2013.

[8] Y. Ma *et al.*, "Remote sensing big data computing: Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 51, pp. 47–60, Oct. 2015.

[9] S. Schade, "Big data breaking barriers—First steps on a long trail," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. XL-7/W3, pp. 691–697, May 2015.

[10] A. Fernández *et al.*, "Big data with cloud computing: An insight on the computing environment, mapreduce, and programming frameworks," *Wiley Interdisciplinary Rev., Data Mining Knowl. Disc.*, vol. 4, no. 5, pp. 380–409, Sep./Oct. 2014.

[11] C. Aggarwal, *Data Classification: Algorithms and Applications*. New York, NY, USA: Chapman & Hall/CRC, 2015.

[12] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[13] Machine Learning Group at the University Waikato. (2014). Weka 3: Data Mining Software in Java. [Online]. Available: http://www.cs.waikato.ac.nz/∼ml/weka/index.html

[14] R. Ferreira *et al.*, "InterIMAGE 2: The architecture of an open source, high performance framework for automatic, knowledge-based image interpretation," in *Proc. Int. Geogr. Object-Based Image Anal. Conf.*, Thessaloniki, Greece, pp. 275–280, 2014.

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

[16] M. D. Assunção, R. N. Calheiros, S. Bianchic, M. A. S. Netto, and R. Buyya, "Big Data computing and clouds: Trends and future directions," *J. Parallel Distrib. Comput.*, vols. 79/80, pp. 3–15, May 2015.

[17] A. Holmes, *Hadoop in Practice*. Shelter Island, NY, USA: Manning, 2012.

[18] T. White, *Hadoop: The Definitive Guide*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media/Yahoo Press, 2010.

[19] A. Gates, *Programming Pig*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2012.

[20] Hypercomp Research Group. (2014). New Digital Repository for Remotely Sensed Hyperspectral Imagery with Unmixing Based Retrieval Functionality. [Online]. Available: http://www.hypercomp.es/repository/

[21] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 band AVIRIS hyperspectral image data set: June 12, 1992 Indian Pine test site 3," Purdue University Research Repository, 2015, doi:10.4231/R7RX991C.

[22] V. A. Ayma *et al.*, "Classification algorithms for big data analysis, a MapReduce approach," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. XL-3W2, pp. 17–21, Mar. 2015.

Victor Andres Ayma Quirita (S'15) received the B.S. degree in electronical engineering, with specialization in telecommunication systems, from San Antonio Abad del Cusco National University, Cusco, Peru, in 2008, and the M.S. degree in electrical engineering, with emphasis in signal processing and control, from Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, in 2013, where he is currently working toward the Ph.D. degree in electrical engineering.

From March 2014 to September 2014, he was a Visiting Ph.D. Student in the Department of Technology of Computers and Communications, University of Extremadura, Caceres, Spain. His research interests include image analysis, computer vision, optimization techniques, remote sensing, machine learning, data mining, and big data analysis.

Gilson Alexandre Ostwald Pedro da Costa (M'14) received the B.Sc. degree in computer engineering from the Pontifical Catholic University of Rio de Janeiro (PUCRio), Rio de Janeiro, Brazil, in 1991, the M.Sc. degree in computer engineering, with emphasis on geomatics, from the Rio de Janeiro State University (UERJ), Rio de Janeiro, in 2003, and concluded his Ph.D. research in 2009, which was partially realized at the Leibniz Hannover University, Hannover, Germany, receiving the Ph.D. degree in electrical engineering from PUC-Rio.

He took part in various scientific development projects, collaborating with research groups in UERJ, PUC-Rio, UnB, INPE, Embrapa, INRIA, CNES, Leibniz Hannover University, University of Salzburg, University of Pavia, and University of Extremadura. He is currently an Adjunct Professor in the Department of Informatics and Computer Science, UERJ. His research interests include computer vision and automatic interpretation of remote-sensing data.

Dr. Costa is currently the Vice-Chair of the Brazilian Chapter of the IEEE Geoscience and Remote Sensing Society.

Patrick Nigri Happ (S'15–M'16) received the B.S. degree in electrical engineering, with specialization in computer and systems, from Rio de Janeiro State University, Rio de Janeiro, Brazil, in 2009, and the M.Sc. and Ph.D. degrees in electrical engineering, with emphasis on signal processing and control, from Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, in 2011 and 2015, respectively.

In 2014, he was a visiting Ph.D. Student at the University of Pavia, Italy. He is currently a Postdoctoral Fellow at PUC-Rio. His research interests include digital image analysis, computer vision, remote sensing, machine learning, pattern recognition, high-performance computing and cloud computing.

Raul Queiroz Feitosa (M'14) received the B.Sc. degree in electronic engineering and the M.Sc. degree in engineering from the Technological Aeronautics Institute, São José dos Campos, Brazil, in 1979 and 1983, respectively, and the Dr. Ing. degree in computer architecture from the University of Erlangen-Nürnberg, Erlangen, Germany, in 1988.

Since 1988, he has been in the Electrical Engineering Department, Pontifical Catholic University of Rio de Janeiro, Brazil, and in the Systems Engineering Department, Rio de Janeiro State University, Rio de Janeiro, where he is currently an Associate Professor. His research interest includes image analysis and its applications in remote sensing, biometrics, and medicine.

Prof. Feitosa is currently the Chair of the Brazilian Chapter of the IEEE Geoscience and Remote Sensing Society, and the vice-president of Commission I of ISPRS.

Rodrigo da Silva Ferreira received the B.S. degree in electrical engineering, with specialization in computer and systems, from Rio de Janeiro State University, Rio de Janeiro, Brazil, in 2009, and the M.S. degree in electrical engineering, with emphasis on signal processing and control, and the Ph.D. degree in electrical engineering from Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, in 2011 and 2016, respectively.

From 2013 to 2014, he was a Visiting Ph.D. Student in the Information and Industrial Engineering Department, University of Pavia, Pavia, Italy. He is currently a Natural Resources Analytics Researcher with IBM Research Brazil, Rio de Janeiro. His research interests include distributed systems, image analysis, computer vision, remote sensing, machine learning, pattern recognition, data mining, and big data.

Dário Augusto Borges Oliveira received the B.S. degree in electrical engineering, with specialization in computer and systems, from Rio de Janeiro State University, Rio de Janeiro, Brazil, in 2007, and the M.S. and Ph.D. degrees in electrical engineering, with emphasis on signal processing and control, from Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, in 2009 and 2013, respectively.

In 2009, he visited the Instituto Superior Técnico, Lisbon, Portugal. From 2011 to 2012, he was a Visiting Ph.D. Student with Leibiniz Universitat, Hannover, Germany. From 2014 to 2015, he was a Postdoctoral Fellow with the Institute of Mathematics and Statistics, University of Sao Paulo, Brazil. He is currently a Computer Vision Researcher with the General Electric Global Research Center, Rio de Janeiro. His research interests include computer vision, pattern recognition, and image processing applied to different areas, such as medicine, remote sensing, mining, biometry, and historical data.

Antonio Plaza (M'05–SM'05–F'15) received the B.S., M.Sc., and Ph.D. degrees in computer engineering from the University of Extremadura, Caceres, Spain, in 1997, 1999, and 2002, respectively.

He is currently an Associate Professor (with accreditation for Full Professor) in the Department of Technology of Computers and Communications, University of Extremadura, where he is also the Head of the Hyperspectral Computing Laboratory. He has been the advisor of 12 Ph.D. dissertations and more than 30 M.Sc. dissertations. He was the Coordinator of the Hyperspectral Imaging Network, a European Project with a total funding of 2.8 million Euro. He has authored more than 500 publications, including 162 journal papers (more than 100 in IEEE journals), 22 book chapters, and more than 240 peer-reviewed conference proceeding papers (94 in IEEE conferences). He has reviewed more than 500 manuscripts for more than 50 different journals. He has edited a book on High-Performance Computing in Remote Sensing (New York, NY, USA: CRC Press/Taylor and Francis, 2007) (the first book on this topic in the published literature). His main research interests include remotely sensed hyperspectral image analysis and efficient implementations of large-scale scientific problems on high-performance computing architectures.

Dr. Plaza served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society in 2011–2012 and has been the President of the Society's Spanish Chapter since November 2012. He has also served as a Proposal Evaluator for the European Commission (Marie Curie Actions, Engineering Panel), the European Space Agency, the Belgium Science Policy, the Israel Science Foundation, and the Spanish Ministry of Science and Innovation. He has participated in the Tenure Track Selection Committee of different universities in Italy, Spain, and Australia. He is an Associate Editor of the IEEE ACCESS and was a Member of the Editorial Board of the IEEE GEOSCIENCE AND REMOTE SENSING NEWSLETTER from 2011 to 2012 and the IEEE GEOSCIENCE AND REMOTE SENSING MAGAZINE in 2013. He was also a Member of the steering committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS). He has been a Guest Editor of eight special issues on hyperspectral remote sensing for different journals. He is currently an Associate Editor of the *Journal of Real-Time Image Processing*. He is also currently the Editor-in-Chief of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING. He received the Best Ph.D. Dissertation Award at the University of Extremadura in 2002. He received the Best Paper Award at the *IEEE Symposium on Signal Processing and Information Technology* in 2008. He also received the recognition of Best Reviewers of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS in 2009 and the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING in 2010, a journal for which he served as an Associate Editor in 2007–2012. He was a Coauthor of the Best Student Paper at the *IEEE International Conference on Space Technology* in 2011. He received the Best Paper Award from JSTARS in 2013 and of the most highly cited paper in 2005–2010 in the Journal of *Parallel and Distributed Computing*.