Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 1050293, 11 pages
http://dx.doi.org/10.1155/2016/1050293

Research Article

Cloud for Distributed Data Analysis Based on the Actor Model

Ivan Kholod, Ilya Petukhov, and Andrey Shorov

Faculty of Computer Science and Technology, Saint Petersburg Electrotechnical University (LETI), Professora Popova Street 5, Saint Petersburg 197376, Russia

Correspondence should be addressed to Andrey Shorov; ashxz@mail.ru

Received 29 April 2016; Accepted 29 June 2016

Academic Editor: Fabrizio Messina

This paper describes the construction of a Cloud for Distributed Data Analysis (CDDA) based on the actor model. The design uses an approach to map the data mining algorithms on decomposed functional blocks, which are assigned to actors. Using actors allows users to move the computation closely towards the stored data. The process does not require loading data sets into the cloud and allows users to analyze confidential information locally. The results of experiments show that the efficiency of the proposed approach outperforms established solutions.

1. Introduction

Presently the terms "cloud computing," "Internet of Things," and "Big Data" have become quite popular. They refer to technologies for collecting, storing, and handling large volumes of data, with a variety of types and a high rate of generation (Big Data). Modern data warehouses provide storage for large amounts of different data. However, all these are worthless if it is not possible to apply analysis and obtain new knowledge from the data.

The technologies machine learning, data mining, and knowledge discovery are used for the aforementioned used tasks. They use complex mathematical methods and algorithms that need powerful computing resources to analyze vast amounts of data. Cloud and cluster technologies provide scalable resources for those tasks.

During the last year, solutions in this area developed from research to product level. The leaders in cloud services are Amazon, Microsoft, IBM, and Google. They all recently launched their public Clouds for Data Analysis (CDA). They provide services for different user requirements and solve different tasks. However, they suffer from a restricted set of analysis tasks and do not extend. Also all of them are paid services and allow only to analyze data that have been uploaded into the cloud. Therefore these services can hardly be used to analyze confidential information.

We suggest to build a Cloud for Distributed Data Analysis (CDDA) that uses the approach *decomposition of data mining algorithms into sets of functional blocks* [1, 2]. It allows us to extend cloud services with new algorithms and modifications of existing algorithms. The mapping of functional blocks on distributed environments (in particular on actor environments) allows us to distribute calculations among different clouds. Moving the data mining algorithm closely to the data removes the restriction and keeps the data in the cloud.

In contrast to existing approaches, CDDA has the following key characteristics:

(i) implementation of both SaaS and PaaS cloud computing service models;

(ii) extension list of data mining algorithms in the cloud by adding new or modifying their functional blocks;

(iii) processing of data sets stored outside the cloud;

(iv) ability to analyze confidential information;

(v) execution of distributed data analysis among several clouds.

The paper is organized as follows. Section 2 is a review of similar cloud-based systems. Section 3 contains the description of a general approach that allows the decomposition of an algorithm into blocks on actor-model systems. Section 4 describes the CDDA architecture. Section 5 discusses experiments and compares the performance of the developed prototype with similar solutions. Finally, Section 6 presents the main conclusions and future work directions.

2. Related Work

Nowadays, a series of cloud computing service platforms have been developed to provide data analysis services for the public sector.

The Chinese Mobile Institute was one of the first who began working in this field. In 2007 they started their research in cloud computing. Later, in 2009, a platform for cloud computing, BigCloud, has been officially announced: *Data Mining Big Cloud-Parallel Data Mining* (BC-PDM) [3]. It is a collection of tools for the parallel execution of algorithms.

BC-PDM is a SaaS platform based on Apache Hadoop. Users can upload data to a repository (hosted in the cloud) from different sources and apply a variety of applications for data management, data analysis, and business applications. Those include the analysis of the performance of parallel operating applications: ETL processing, social network analysis, analysis of texts (text mining), data analysis (data mining), and statistical analysis.

It includes about ten algorithms and cannot be extended by the user. This cloud is only used for research in the Chinese Mobile Institute. Therefore it is not available for public use.

Azure Machine Learning (Azure ML) [4] is a SaaS cloud-based predictive analytics service from Microsoft Inc. It has been launched in February 2015. Azure ML provides paid services, which allow users to execute the full cycle of data mining: data collection, preprocessing, defining features, choosing and applying an algorithm, evaluating a model, and publication. The service is for experienced users with knowledge in machine learning algorithms.

The analysis process is designed as a workflow. Each step of the workflow is a module, designed to execute a single subtask of the analysis (data reader, data transformation, an algorithm, or other). A module can be executed on a single cluster node. Thus, a number of modules can be executed in parallel if the workflow allows it.

Azure ML can import data from local files, online sources, and other cloud projects (experiments). The reader module allows us to load data from external sources on the Internet or other file storages.

The user can only apply the proprietary machine learning algorithms of Azure ML: classification (Boosted Decision Trees, Random Forests, Logistic Regression, SVM, Averaged Perceptron, and neural networks), regression (Linear Regression, Boosted Decision Trees, and neural networks), anomaly detection (SVM, PCA), and clustering (K-Means). Additional algorithms are available for purchase at the Machine Learning Marketplace.

In April 2015 Amazon has launched their *Amazon Machine Learning* service that allows users to train predictive models in the cloud [5]. This service provides all required stages of data analysis: data preparation, construction of a machine learning model, its settings, and eventually the prediction. The user can build and fine-tune predictive models using large amounts of data.

Special knowledge in the field of machine learning is not required. Amazon Machine Learning solves only classification and regression tasks. It supports three distinct types of tools: binary classification, multiclass classification, and regression. New algorithms and machine learning tasks cannot be implemented into the service.

It allows users to analyze data stored in others Amazon services (Amazon Simple Storage Service, Amazon Redshift, or Amazon Relational Database Service). To scale computations, the service uses Apache Hadoop.

Google made its *Cloud Machine Learning* platform [6], which is used by Google Photos, Translate, and Inbox, available to developers in March 2016. It is a managed platform that empowers users to build machine learning models. The platform provides pretrained models and helps to generate customized models. It allows users to apply neural network based machine learning methods, which are used by other Google services including Photos (image search), the Google app (voice search), Translate, and Inbox (Smart Reply).

The Google services will provide application programming interfaces (APIs) for automatic image recognition, speech recognition, language translation, and more. The cloud provides four basic machine learning services:

(i) Vision API enables developers to recognize the content of an image by encapsulating powerful machine learning models.

(ii) Speech API enables developers to convert audio to text by applying powerful neural network models.

(iii) Translate API provides a simple programmatic interface to translate arbitrary strings into any supported language.

(iv) Prediction API can help to analyze users' data to add applicable features (customers sentiment analysis, message routing decisions, churn analysis, and others) to a user's application.

All of these services are provided by REST API for client applications. Users can only analyze data stored in Google storage. Also user cannot add new machine learning algorithms.

In the beginning of 2016, IBM presented their analytic service—*Watson Analytics* [7]. It is a smart data discovery service, available on the IBM cloud. The service solves high-level analytic tasks. Users are able to deploy queries in natural language. Watson Analytics' three main areas are Explore, Predict, and Assemble:

(i) Explore allows users to create queries to data. Users can use existing templates or enter text based queries.

(ii) Predict allows predicting one or more variables based on other variables. Therefore classification and regressions methods are used.

(iii) Assemble allows users to create analytic reports, presentations, and data visualization.

Watson Analytics handles data sets that have been uploaded into the cloud in .csv or MS Excel (.xls) formats. Users can only use methods, provided by Watson Analytics.

There are also some data mining algorithm libraries for distributed environments. They can be used to create a CDA. The most famous are the following.

Data Mining Cloud Framework (DMCF) [8] is designed for developing and running distributed data analytics applications as a collection of services. The first implementation of the framework has been carried out on the Windows Azure cloud platform and has been evaluated through a set of data analysis applications executed on a Microsoft Cloud data center. The framework treats the data sets, data mining algorithms, and mining models as services, which can be combined through a visual interface to produce distributed workflows executed on a cloud platform. DMCF supports JavaScript for Clouds (JS4Cloud) as an additional and more flexible programming interface.

Apache Spark Machine Learning Library (MLlib) [9] is a scalable machine learning library for the Apache Spark platform. It consists of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, and lower-level optimization primitives and higher-level pipeline APIs. It has an own implementation of MapReduce, which uses memory for data storage (versus Apache Hadoop that uses disk storage). It allows us to increase the efficiency of the algorithm's performance. The user can extend the set of machine learning algorithms by own implementations. However, users must decompose their algorithms according to MapReduce and other Spark's specific functions. It strongly limits the abilities for parallelization of data mining algorithms.

Apache Mahout [10] is also a data mining library concerning the MapReduce paradigm. It can be executed on Apache Hadoop or Spark based platforms. It contains only a few data mining algorithms for distributed execution: collaborative filtering (User-Based, Item-Based, and Matrix Factorization with ALS), classification (Naive Bayes/Complementary Naive Bayes, Random Forests), clustering (K-Means, Fuzzy K-Means, Streaming K-Means, and Spectral Clustering), and dimensionality reduction (Stochastic SVD, PCA, and QR Decomposition). Users can extend the library by adding new data mining algorithms. The core libraries are highly optimized and also show good performance for nondistributed execution.

Weka4WS [11] is an extension of the famous open-source data mining library Weka (The Waikato Environment for Knowledge Analysis) [12]. The extension implements a framework to support the execution in WSRF [13] enabled grids. Weka4WS allows the distribution and execution of all its data mining algorithms on remote grid nodes. To enable remote invocation, the data mining algorithms provided by the Weka library are extended to a Web Service, which can be easily deployed on the available Grid nodes. Weka4WS can only handle a data set contained in a single storage node. This data set is then transferred to computing nodes to be mined. Unfortunately, this library is not supported now. The latest version is from July 2008.

Table 1 summarizes some features of the above-mentioned systems. They also show the following disadvantages:

(i) the fact that data sets must be stored inside a cloud, which does not allow users to analyze confidential information;

(ii) a restricted number of analysis tasks in public clouds and no full analysis tool chain in the frameworks;

(iii) using basically MapReduce paradigm (in particular the Apache Hadoop) for distributed analysis.

The MapReduce paradigm is adapted only for data processing functions, which are list homomorphisms [14]. Therefore, not all data mining algorithms can be decomposed into map and reduce functions.

We suggest the architecture of a cloud, based on the actor model, that allows users to execute data mining algorithms on a hybrid cloud (public and private cloud). The proposed cloud uses an approach to map an algorithm, decomposed into functional blocks, on a set of actors. Using actors allows users to move handling data sets towards the stored data. The process does not request uploading data sets into the cloud and allows users to analyze confidential information locally.

3. Mapping Data Mining Algorithms in Distributed Environments

3.1. Common Approach. According to [15], a data mining algorithm can be written as a sequence of functional blocks (based on functional language principles). Classical functions in functional languages are pure functions. A data mining algorithm can be written as a function (with two input arguments: data set d and mining model m) that can be decomposed into a number of nested functions:

$$\begin{aligned}\operatorname{dma}(d,m) &= f_n \circ f_{n-1} \circ \cdots \circ f_i \circ \cdots \circ f_1 (d,m) \\ &= f_n (d, f_{n-1} (d, \ldots, f_i (d, \ldots, f_1 (d,m), \ldots), \ldots)),\end{aligned} \tag{1}$$

where f_i is pure function of the type FB:: $D \rightarrow M \rightarrow M$, in which

(i) D is the input data set that is analyzed by function f_i and

(ii) M is the mining model that is built by function f_i.

We called this function functional block.

For example, we decomposed the K-Means algorithm into a set of functional blocks, ready for distributed execution. Therefore it can be represented as a chain of functional expressions:

$$K\text{-Means} = \text{findClusters} \circ \text{initClusters}, \tag{2}$$

in which

(i) *initClusters* creates a set of centroids in a random way;

(ii) *findClusters* finds centroids of clusters.

The *findClusters* block is the cycle which calls the next functional block while the cluster's centroids are changed:

(i) *distributeVectors* computes the distances between vectors (from data set D) and centroids of the clusters to distribute the vectors between the clusters;

(ii) *updateCentroids* updates centroids of clusters with new sets of vectors.

According to the Church-Rosser theorem [16] the reduction (execution) of such functional expressions (algorithm) can be done concurrently.

Table 1: The solutions for building data analysis cloud services.

Capabilities	BC-PDM	Azure ML	Amazon machine learning	Google Cloud machine learning	Watson Analytics	DMCF	Apache Spark MLlib	Apache Mahout	Weka 4 WS
Cloud service model	SaaS	SaaS	SaaS	SaaS	SaaS	SaaS	—	—	—
User interface	Web	Web	Web	API	Web	Web	—	—	Desktop
User's level	Developer	Knowledge ML algorithms	Analytic	Developer	Analytic/manager	Developer	Developer	Developer	Developer
API Interface	No	REST	REST	REST	REST	JS4Cloud	Yes	Yes	Yes
Scalable computing	Yes	For single modules	Yes	Yes	Yes	Yes	Yes	Yes	No
Data source location	Inside cloud	Inside cloud	Inside cloud	Inside cloud	Inside cloud	Inside cloud	Outside	Outside	Any
Distributed computing platform	Apache Hadoop	—	Apache Hadoop	Apache Hadoop	—	SOA	Apache Spark	Apache Hadoop	WSRF
Full analysis cycle	Yes	Yes	Yes	Yes	No	Yes	No	No	No
Included data mining algorithms	Classification, clustering, and association	Classification, anomaly detection, regression, and clustering	Classification, regression	Classification, regression	Classification, regression	Classification, regression, and clustering	Classification, regression, Clustering, dimensionality reduction, and feature extraction	Collaborative filtering, classification, clustering, and dimensionality reduction	Classification, association, regression, and clustering
Adding new algorithms	No	From Machine Learning Marketplace	No	No	No	Yes	Yes	Yes	Yes
Using	No	Paid	Paid	Paid	Paid	No	Open Source	Open Source	Open Source

3.2. The Function for Parallel Execution of a Data Mining Algorithm. One of the main advantages of building algorithms from functional blocks is the possibility of parallel execution. For this task we need to transform the sequential expression (1) into a form, in which the functional blocks will be invoked as arguments. For this the high-order *map* function can be used. It allows us to apply some functions to elements of lists. The function can be executed parallel for different elements. The *map* function returns the list of results. To reduce the list into a single result the high-order *fold* function can be used. Thus, the function for the transformation of sequential expressions into a parallel form can be presented as follows:

$$\begin{gathered}\text{parallel}\colon \text{FB} \longrightarrow D \longrightarrow M \longrightarrow ([M] \longrightarrow M) \longrightarrow (\text{FB} \longrightarrow [\text{FB}]) \longrightarrow (D \longrightarrow [D]) \longrightarrow (M \longrightarrow [M]) \longrightarrow (\text{FB} \longrightarrow D \longrightarrow M \longrightarrow H) \longrightarrow (H \longrightarrow M) \longrightarrow M, \\ \text{parallel}\left(f, d, m, \mathit{join}, \text{distr}F, \text{distr}D, \text{distr}M, \mathit{start}, \mathit{get}\right) = \text{fold}\left(\mathit{join}, m, \mathit{get}, \left(\mathit{map}\left(\mathit{start}, \text{distr}F\left(f\right), \text{distr}D\left(d, m\right), \text{distr}M\left(d, m\right)\right)\right)\right),\end{gathered} \tag{3}$$

in which

(i) f (1st argument of the parallel function) is a functional block that is executed concurrently;

(ii) distr$F$, distr$D$, and distr$M$ functions divide the the functional block f, the input data set d, and the mining model m into the lists:

$$\begin{gathered}\text{distr}F\colon (D \longrightarrow M \longrightarrow M) \longrightarrow [(D \longrightarrow M \longrightarrow M)] \\ \text{distr}D\colon D \longrightarrow [D] \\ \text{distr}M\colon M \longrightarrow [M];\end{gathered} \tag{4}$$

(iii) *start* function applies each functional block f from the list $[F]$ to the elements of the lists $[D]$ and $[M]$ and returns a handler h for the parallel execution of the parallel functional block f:

$$\mathit{start}\colon \text{FB} \longrightarrow D \longrightarrow M \longrightarrow H; \tag{5}$$

(iv) *get* function reads the mining model from the handler h:

$$\mathit{get}\colon H \longrightarrow M; \tag{6}$$

(v) *join* function joins the mining models from the list $[M]$ and returns the merged mining model M:

$$\mathit{join}\colon [M] \longrightarrow M; \tag{7}$$

(vi) *map* function applies function *start* to elements of the lists $[F]$, $[D]$, and $[M]$:

$$\begin{gathered}\mathit{map}\colon (\text{FB} \longrightarrow D \longrightarrow M \longrightarrow H) \longrightarrow [\text{FB}] \longrightarrow [D] \longrightarrow [M] \longrightarrow [H] \\ \mathit{map}\left(\mathit{start}, \{f_0, f_1, \dots, f_k\}, \{d_0, d_1, \dots, d_k\}, \{m_0, m_1, \dots, m_k\}\right) = \text{list}\left(\left(\mathit{start}\left(f_0, d_0, m_0\right)\right), \dots, \left(\mathit{start}\left(f_k, d_k, m_k\right)\right)\right);\end{gathered} \tag{8}$$

(vii) *fold* function reduces the list of mining models to a single result:

$$\begin{gathered}\mathit{fold}\colon ([M] \longrightarrow M) \longrightarrow M \longrightarrow (H \longrightarrow M) \longrightarrow [H] \longrightarrow M \\ \mathit{fold}\left(\mathit{join}, m, \mathit{get}, \{h_0, h_1, \dots, h_k\}\right) \\ = \mathit{join}\left(m, \mathit{get}\left(h_0\right), \mathit{get}\left(h_1\right), \dots, \mathit{get}\left(h_k\right)\right).\end{gathered} \tag{9}$$

3.3. Mapping the Parallel Function on Actor Model. A list of handlers parsing between the *map* and *fold* functions is part of some distributed execution environments. In general, an execution environment can be represented as a set of handlers:

$$E = \left\{h_0, h_1, \dots, h_j, \dots, h_n\right\}, \tag{10}$$

in which

(i) h_0 is handler to execute sequential functional blocks of an algorithm;

(ii) h_1–h_n are handlers to execute parallel functional blocks in the distributed environment.

Each handler must implement the *start* and *get* functions to use them in the *parallel* function. The implementation of the *start* and *get* functions is specified by the execution environment which includes these handlers. Examples of these handlers are threads, actors, web services, and others.

The execution environment, which is implemented on the basis of the actor model [17, 18], uses actors as handlers. Actors interact with each other through the exchange of messages.

In order to send and receive a message, actors implement two functions:

(i) *send (msg, a)*: to send the message *msg* to the actor *a*;

(ii) *receive(a)*: to receive the message *msg* from the actor *a*.

One of the popular implementations of the actor model is the AKKA system, which uses the following actors to route messages:

(i) x is the inbox, which receives messages from actors, stores them in the memory, and sends them to other actors;

(ii) r is the router, which obtains messages from the inbox and distributes them among actors, depending on their type and load.

Thus, the actors environment can be written as

$$E = \left\{r, x, a_0, a_1, a_2, \ldots, a_j, \ldots, a_n\right\}, \tag{11}$$

in which

(i) x is the inbox, which stores messages;

(ii) r is the router, which distributes messages among actors;

(iii) a_0 is the actor, which carries out the main algorithm sequence;

(iv) a_1–a_n are the actors, which carry out the parallel function of the algorithm.

The functions of the handlers *start* and *get* can be presented in the following way:

$$\begin{gathered} start\left(f, d, m\right) = send\left(\left\langle f, d, m\right\rangle, x\right); \\ get\left(x\right) = receive\left(x\right). \end{gathered} \tag{12}$$

Thus, actors can execute functional blocks and therefore carry out a distributed implementation of the data mining algorithm.

The described approach was implemented as the data mining algorithm library DXelopes [19]. The library has adapters for the integration in different distributed environments. We used the actor model environment to create the prototype of a cloud for distributed data mining.

4. Architecture of the Cloud for Distributed Data Analysis

4.1. Levels of the Cloud for Distributed Data Analysis. The architecture of CDDA can be divided into several levels (Figure 1):

(i) Hardware;

(ii) virtual distributed environment;

(iii) analysis services.

The *hardware level* includes the pool of computers, storage, and networking resources, available in the cloud. We distinguish the following nodes:

(i) control nodes that run services to manage computation nodes and virtual networks between them;

(ii) computation nodes that run the hypervisor portion of computation, operating tenant virtual machines or instances;

(iii) storage nodes that contain the disks that provide space for tenant virtual machine instances.

The *virtual distributed environment level* provides controls for the virtual machines, the network connections between them, the disk spaces, and the user authorization. Therefore, we use a stack of cloud technologies: hypervisors, network managers, file storages, and so forth. The OpenStack software [20] integrates these technologies and allows us to control all the resources.

This level keeps previously prepared images of the virtual machines (VMs). Each VM has preinstalled software to execute the functional blocks of data mining algorithms: OS, AKKA, and DXelopes libraries (Figure 2). The VMs are united in a virtual network. Thus it forms the environment for distributed execution of a data mining algorithms. OpenStack balances VMs loads and provides optimal execution of the data mining algorithm in the cloud.

The *analysis services level* includes modules to work with the CDDA. They are installed on the control nodes and the VMs. The following units are deployed on the control nodes: Web interface, API CDDA, user control module, project control module, and analysis control module.

Each VM includes the following modules: analysis control module, data mining algorithms library, that is, DXelopes, ETL tools, and distributed system.

The user interface is implemented as a Web interface and users can do the full cycle of the analysis: data preprocessing, setting and execution of data mining algorithms, selecting and setting of the execution environment, estimation of the created mining model, and visualization and application of the created mining model.

The CDDA provides an API interface. It is REST API according to JSR 73 JDM API [21]. It allows developers to integrate other third-party systems with the CDDA.

Thus the users can manage work with the CDDA through the Web and the API interfaces with the help of the following modules:

(i) user control module provides the user authorization, verification of user permissions, and user registration;

(ii) project control module provides the user's projects management: creating, editing, and removing;

(iii) analysis control module provides the execution of full analysis cycle.

The data mining library DXelopes is the engine of the CDDA. It can be extended by

(i) data mining algorithms;

(ii) adapters for different ETL tools;

(iii) adapters for different execution environments.

The DXelopes library allows us to add algorithms as well as to construct new algorithms from existing functional blocks or the restructuring of the existing algorithms. The adapters for the ETL tools allow us to integrate the library within different systems, which implement data extraction, data transformation, and data loading. Thus, the integration of the DXelopes library and the ETL tools enables analysis of Big Data in the CDDA.

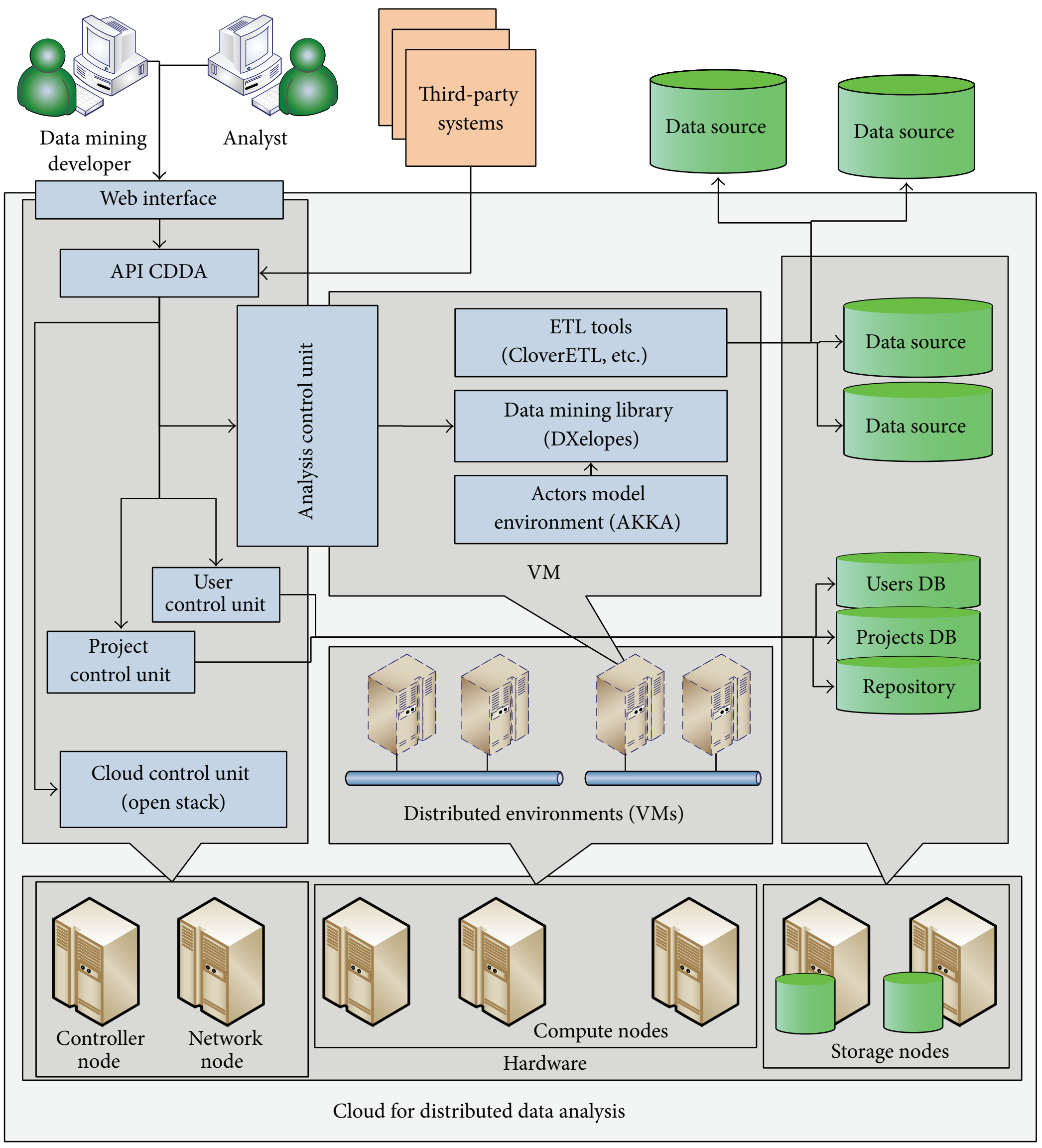

Figure 1: The architecture of Cloud for Distributed Data Analysis based on the actor model.

4.2. Using the Actor Model for the Cloud for Distributed Data Analysis. The adapters for the execution environment allow us to integrate the library within different distributed systems. So far the library has been integrated with the actor model environment (the actor model environment is presented by AKKA framework [22]). Thus, all data mining algorithms implemented in the library can be executed in these environments.

For each environment the CDDA contains sets of VMs with installed software (Figure 2):

(i) analysis control module;

(ii) DXelopes library;

(iii) configured actor system from the AKKA framework.

Such an approach to CDDA architecture allows us to transfer VMs to other clouds, hence distributing the analysis among them. This property is important when working with "private" clouds; the data of the latter must not be transferred to public clouds (e.g., CDDA). In order to fulfill this requirement using an image of a VM, which is stored in the CDDA repository, a VM must be designed with an installed actors environment and executed in a private cloud (Figure 3). A part of the data mining algorithm will be executed in this VM. This part will directly process data. The results of data processing will be sent to the CDDA as a knowledge model, which will actually not contain the data.

For example, concerning the K-Means algorithm, the actor with the *distributeVectors* functional block can handle all the vectors on the VM in a private cloud and then the actor with the *updateCentroids* functional block will recalculate the centroids of the clusters in the CDDA (Figure 3). The *distributeVectors* functional block receives the clusters centroids, determinates a vector belonging to the cluster, and sends

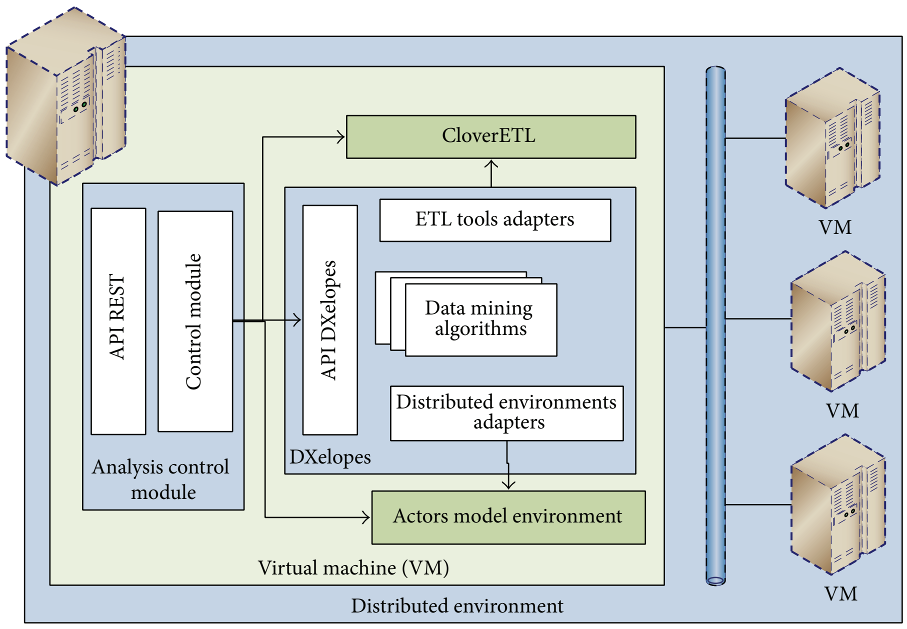

FIGURE 2: The configuration of the VM for data mining.

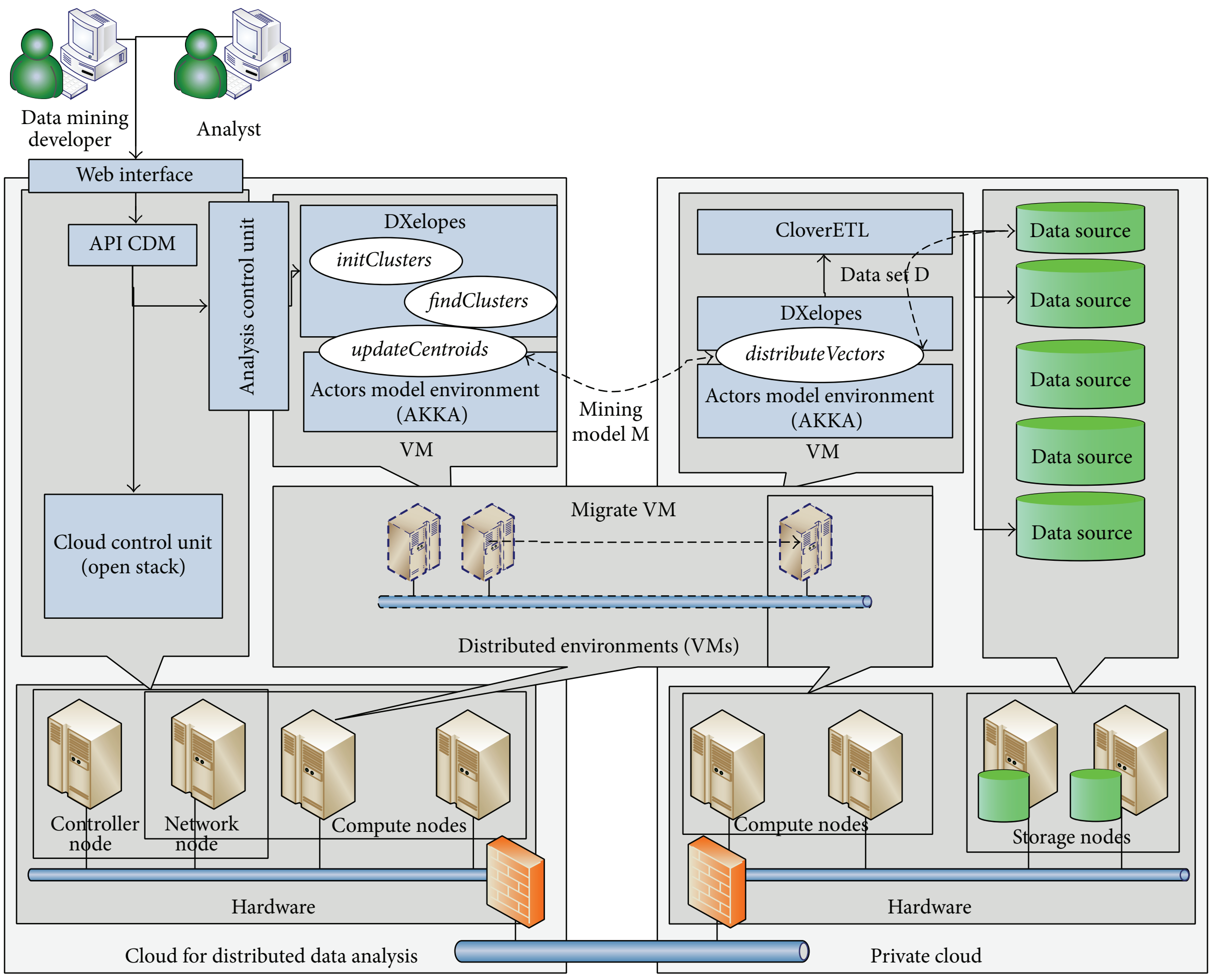

FIGURE 3: The configuration of the VM for data mining.

Table 2: Cloud computing infrastructure.

Characteristics	IBM FlexSystem X240	IBM FlexSystem P260	Huawei FusionServer RH2288 V3
CPU	Intel Xeon 2.9 GHz (2 CPU on 6 cores)	Power 7 3.3 GHz (2 CPU on 4 cores)	Intel® Xeon® E5-2600 (2 CPU on 4 cores)
RAM	128 GB	128 GB	128 GB
OS	3 win 2012/Hyper-V systems, 5 rhel/kvm systems	2 AIX/PowerVM systems	3 win 2012/ Hyper-V systems, 5 rhel/kvm systems
Performance	200 GFlops	400 GFlops	200 GFlops

Table 3: Experimental data sets.

Input data set	Number of rows	Number of attributes	Size of file (Kb)
Iris two-class data (ITCD)	100	4	2
Telescope data (TD)	19020	10	1 499
Breast cancer info (BCI)	102294	5	4 832
Movie ratings (MR)	227472	4	6 055
Flight on-time performance (FOTP) (raw)	504397	5	39 555
Flight delays data (FDD)	2719418	5	136 380

the accumulated distances of each vector for each cluster to the cloud. The *updateCentroids* functional block receives the accumulated distances, updates the centroids, and sends them to the VM in the private cloud. Thus the information is not transferred from the private cloud in the public the CDDA.

5. Experiments

A series of the experiments were done to verify the effectiveness of the described the CDDA implementation. We compared the performance of CDDA with Azure ML and the Spark MLlib's performance. The CDDA and Spark MLlib had been executed on high-performance servers, supporting hardware virtualization and providing high-performance cloud computing systems. The computing cluster infrastructure for the experiments is shown in Table 2.

We checked availability of distributed analysis for private and public clouds. Therefore we deployed VMs with actor environments into the second cloud based on Huawei FusionServer.

The data sets from Azure ML (you can download these data sets from https://studio.azureml.net/Home/Anonymous as Guest) were used for the experiments. The parameters of the data sets are presented in Table 3.

For these data sets we solved clustering task with the K-Means algorithm that is implemented in Azure ML, Spark MLlib, and CDDA. To compare acceleration for centralized systems we performed experiments for 1 and for 4 handlers

The experiments for Azure ML and Spark MLlib are executed with centralized data sets. Hence we loaded all data sets into the clouds. The data sets loading time and the time for data analysis were measured separately.

The experiments with CDDA were executed with local and with distributed data sets. In the second case all data sets were stored in the second cloud. We transferred the VM with the actor system (AKKA) into this cloud (see Figure 3). As a result the data sets were processed locally without upload into the CDDA.

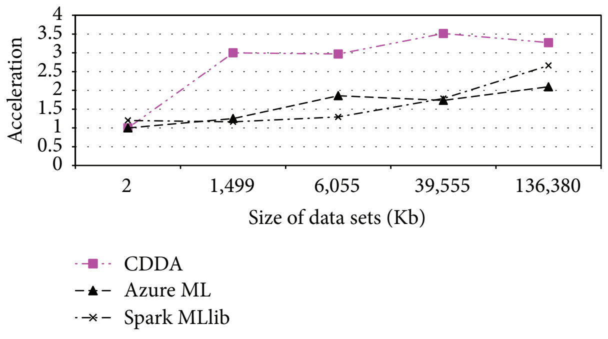

Figure 4: The acceleration of the parallel execution.

The experimental results are provided in Table 4.

The execution time of the algorithm for centralized analysis in all systems is almost the same. The CDDA is a little bit faster than Apache Spark MLlib and Azure ML. The acceleration (as time for 1 handler/time for 4 handlers) and efficiency of parallel algorithm execution (see Figures 4 and 5) in the CDDA are better than in the Azure ML and the Apache Spark because the DXelopes library allows to distribute the algorithm's blocks between the handlers more flexibly.

The Apache Spark is restricted by MapReduce paradigm and can parallely execute only the map and reduce functions. The Azure ML can only execute whole algorithms on single nodes.

The centralized analysis of local data sets is executed faster than the analysis of distributed data sets. However, if we summarize the time of analysis and the data loading time, the accumulated time of centralized data analysis is bigger than that in distributed data analysis (Figure 6).

This effect is achieved by reduction of data transferred within a network. In case of distributed data analysis,

Table 4: Experimental results (s).

Cloud	Action	ITCD	TD	MR	FOTP	FDD
—	Data set loading time	1	1	2	5	15
Azure ML	Local data centralized analysis (1 handler)	4	5	13	26	132
	Local data centralized analysis (4 handlers)	4	4	11	17	63
	Data set loading and centralized analysis	**5**	**7**	**20**	**43**	**83**
Spark MLlib	Local data centralized analysis (1 handler)	3	4	6	11	157
	Local data centralized analysis (4 handlers)	2	3	5	6	59
	Data set loading and centralized analysis	**3**	**6**	**10**	**16**	**76**
CDDA	Local data centralized analysis (1 handler)	0.4	1	9	12	121
	Local data centralized analysis (4 handlers)	0.5	1	3	4	37
	Data set loading and centralized analysis	**1.5**	**2.8**	**8**	**15**	**74**
CDDA	Distributed data analysis (4 handlers)	**1**	**2**	**4**	**6**	**41**

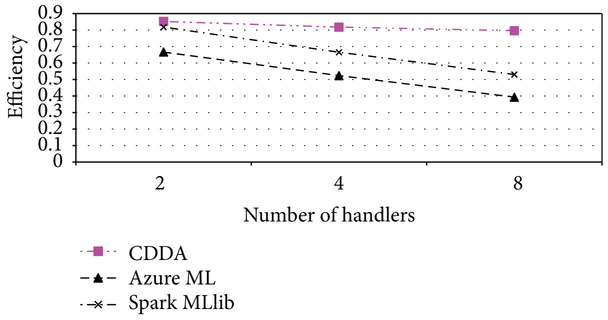

Figure 5: The efficiency of the parallel execution.

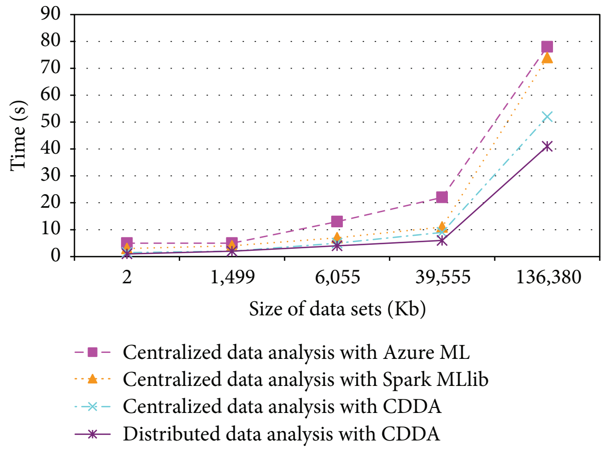

Figure 6: Comparison of the experimental results.

a mining model is transferred between the clouds instead of the data sets.

By increasing the amount of data being analyzed the difference between distributed and centralized analyses is increasing. This is due to the difference between the transfer time of the data and time to transfer the model. When a substantial increase in the size of data occurs the size of the model increases slightly. Therefore, the time to transfer the model also increases slightly.

6. Conclusion

The representation of a data mining algorithm as functional expression makes it possible to divide the algorithm into blocks. Such a splitting helps to map it to an actor environment. We implemented this approach as the DXelopes library. It contains different algorithms and allows us to add new algorithms. The library has adapters to support the integration within actor-model system—AKKA. It allowed us to create a prototype of the cloud for distributed data analysis.

The cloud uses clusters of VMs. Each VM contains the DXelopes library and an AKKA system. A user can deploy VMs in other clouds and execute distributed data analysis.

Thus the created CDDA has the following key characteristics, which distinguish it from other similar solutions:

(i) implementation of both SaaS and PaaS cloud computing service models;

(ii) extension list of data mining algorithms in the cloud by adding new functional blocks or modifying their functional blocks;

(iii) processing of data sets stored outside the cloud;

(iv) ability to analyze confidential information;

(v) execution of distributed data analysis among several clouds.

The last property allows us

(i) to increase data security and to do so without the storage of data in a public cloud;

(ii) to reduce network traffic due to the prevention of data transfer between the clouds;

(iii) to enhance the efficiency due to the "localization" of calculations.

Available solutions do not have the above-mentioned features and limit the advantages of cloud technologies and data analysis technologies integration significantly.

In the future we plan to extend the supported distributed platforms and ETL tools and develop a release version of the CDDA.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work has been performed in Saint Petersburg Electrotechnical University (LETI) within the scope of the contract Board of Education of Russia and Science of the Russian Federation under the Contract no. 02.G25.31.0058 from 12.02.2013. The paper has been prepared within the scope of the state project "Organization of Scientific Research" of the main part of the state plan of the Board of Education of Russia, the project part of the state plan of the Board of Education of Russia (Task 2.136.2014/K) as well as supported by grant of RFBR 16-07-00625, supported by the Russian President's fellowship.

References

[1] I. Kholod, M. Kupriyanov, and A. Shorov, "Decomposition of data mining algorithms into unified functional blocks," *Mathematical Problems in Engineering*, vol. 2016, Article ID 8197349, 11 pages, 2016.

[2] I. Kholod and I. Petukhov, "Creation of data mining algorithms as functional expression for parallel and distributed execution," in *Parallel Computing Technologies*, V. Malyshkin, Ed., vol. 9251 of *Lecture Notes in Computer Science*, pp. 62–67, Springer, New York, NY, USA, 2015.

[3] L. Yu, J. Zheng, W. C. Shen et al., "BC-PDM: data mining, social network analysis and text mining system based on cloud computing," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*, pp. 1496–1499, ACM, Beijing, China, August 2012.

[4] C. J. Gronlund, "Introduction to machine learning on Microsoft Azure," https://azure.microsoft.com/en-gb/documentation/articles/machine-learning-what-is-machine-learning/.

[5] J. Barr, "Amazon Machine Learning-Make Data-Driven Decisions at Scale," Amazon Machine Learning, 2016, https://aws.amazon.com/ru/blogs/aws/amazon-machine-learning-make-data-driven-decisions-at-scale/.

[6] Google Cloud Machine Learning at Scale, https://cloud.google.com/products/machine-learning/.

[7] A. Lally, J. M. Prager, M. C. McCord et al., "Question analysis: how Watson reads a clue," *IBM Journal of Research and Development*, vol. 56, no. 3-4, pp. 2:1–2:14, 2012.

[8] F. Marozzo, D. Talia, and P. Trunfio, "A workflow-oriented language for scalable data analytics," in *Proceedings of the 1st International Workshop on Sustainable Ultrascale Computing Systems (NESUS '14)*, Porto, Portugal, August 2014.

[9] X. Meng, J. Bradley, B. Yavuz et al., "MLlib: machine learning in apache spark," *Journal of Machine Learning Research*, vol. 17, pp. 1–7, 2016.

[10] G. Ingersoll, *Introducing Apache Mahout. Scalable, Commercial-friendly Machine Learning for Building Intelligent Applications*, IBM, 2009.

[11] D. Talia, P. Trunfio, and O. Verta, "The Weka4WS framework for distributed data mining in service-oriented Grids," *Concurrency Computation: Practice and Experience*, vol. 20, no. 16, pp. 1933–1951, 2008.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[13] K. Czajkowski, D. Ferguson, I. Foster et al., "From open grid services infrastructure to ws-resource framework: refactoring & evolution," 2004.

[14] S. Gorlatch, "Extracting and implementing list homomorphisms in parallel program development," *Science of Computer Programming*, vol. 33, no. 1, pp. 1–27, 1999.

[15] I. Kholod and I. Petukhov, "Creation of data mining algorithms as functional expression for parallel and distributed execution," in *Parallel Computing Technologies*, pp. 62–67, Springer, 2015.

[16] A. Church and J. B. Rosser, "Some properties of conversion," *Transactions of the American Mathematical Society*, vol. 39, no. 3, pp. 472–482, 1936.

[17] C. Hewitt, P. Bishop, and R. Steiger, "A universal modular actor formalism for artificial intelligence," in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, pp. 235–245, Morgan Kaufmann Publishers, Stanford, Calif, USA, August 1973.

[18] W. D. Clinger, *Foundations of Actor Semantics*, 1981.

[19] I. Kholod, "Framework for multi threads execution of data mining algorithms," in *Proceedings of the 2015 IEEE North West Russia Section Young Researchers in Electrical and Electronic Engineering Conference (ElConRusNW '15)*, pp. 82–88, IEEE, St. Petersburg, Russia, February 2015.

[20] K. Jackson, C. Bunch, and E. Sigler, *OpenStack Cloud Computing Cookbook*, Packt Publishing, 2015.

[21] JSR-000073 Data Mining API. (Maintenance Release), https://jcp.org/aboutJava/communityprocess/mrel/jsr073/index.html.

[22] D. Wyatt, *Akka Concurrency*, Artima Incorporation, 2013.