2014 IEEE Symposium on Computational Intelligence in Big Data (CIBD): proceedings (2014). IEEE, 1-8.
DOI: http://dx.doi.org/10.1109/CIBD.2014.7011537

A Scalable Machine Learning Online Service for Big Data Real-Time Analysis

Alejandro Baldominos*, Esperanza Albacete*, Yago Saez* and Pedro Isasi*
*Computer Science Department
Universidad Carlos III de Madrid, Madrid, Spain
Email: {alejandro.baldominos, esperanza.albacete, yago.saez, pedro.isasi}@uc3m.es

***Abstract*—This work describes a proposal for developing and testing a scalable machine learning architecture able to provide real-time predictions or analytics as a service over domain-independent big data, working on top of the Hadoop ecosystem and providing real-time analytics as a service through a RESTful API. Systems implementing this architecture could provide companies with on-demand tools facilitating the tasks of storing, analyzing, understanding and reacting to their data, either in batch or stream fashion; and could turn into a valuable asset for improving the business performance and be a key market differentiator in this fast pace environment. In order to validate the proposed architecture, two systems are developed, each one providing classical machine-learning services in different domains: the first one involves a recommender system for web advertising, while the second consists in a prediction system which learns from gamers' behavior and tries to predict future events such as purchases or churning. An evaluation is carried out on these systems, and results show how both services are able to provide fast responses even when a number of concurrent requests are made, and in the particular case of the second system, results clearly prove that computed predictions significantly outperform those obtained if random guess was used.**

I. Introduction

Each day, the amount of data and the number of changing data sources continue to grow. As companies are collecting vast amounts of data from the Internet, their own web sites, social media channels, customer information, call center reports or financial transactions; the need for analytical tools able to leverage knowledge behind all these data is imperative. Even more important than this is the fact that this growth is going to increase exponentially in the future, as there are other emerging areas which are about to come such as Smart Cities and Internet of Things, where the number of potential devices capable of generating high volumes of data is going to be multiplied as a result of what is known as M2M (machine-to-machine interaction). All these incoming changes will require an adequate scaled infrastructure which allows storing, processing and responding to an increasing number of batch and stream requests.

This vast amount of information, if conveniently processed, can reveal relevant insights about each business. For instance, analysis of the former data may serve to predict the upcoming *friendships* or interests of a social network user, suggest related products in which a customer may be interested to purchase or adapt the content and structure of an online course to better fit the students' needs. At this point it is where machine learning techniques can help to analyze big data sources and extract the important trends, links, rules... or in other words: knowledge. This field has been studied since the first appearance of the Knowledge Discovery in Databases (KDD) concept, but depending on the data sources and the domains, different approaches and techniques were used, such as association, clustering, classification, prediction, sequential patterns identification, decision trees or, what is more usual, a hybrid approach resulting from a combination of these approaches. However, when a big data framework involves real-time analytics, a specific software architecture is needed. Typically, a distinction is made when considering how this data is analyzed with regard to time constraints:

- **Batch processing**, where a set (typically a very big one) of data is processed to retrieve some statistics of other information. This processing is not required to happen in real-time, as the expected result is not needed within a strong time constraint. This processing is the most adequate for those machine learning techniques or algorithms that require to run periodic training and update processes.
- **Stream processing**, where new data must be processed in real-time, in many times considering the historical data as well, in order to generate a value. Most often, it involves the use of previously trained models, in order to avoid too much processing and ultimately reduce response times.

Typical examples of batch processing would involve computing trends or extracting patterns from customers activity during a period, that can be categorized per product, website, geographical distribution or user profiles in a social network. For this type of operations, frameworks such as the MapReduce [1] paradigm are suitable, as they enable distributed processing of the data over a cluster of inexpensive nodes. However, additional value can be obtained when stream processing comes into play, as it unveals new possibilities such as providing real-time recommendations to a customer. For instance, these systems can dynamically interact with customers offering specific products and empowering their engagement by means of an accurate prediction of when they are about to leave the site. This prediction enables generating the appropriate events to modify the customer behaviour.

II. State of the Art

As stated in the previous section, this paper proposes an architecture serving as the framework for developing systems providing batch and stream data processing as a service.

Therefore, this work tackles the last two stages of the big data value chain [2], i.e., storage and analytics of big data.

Starting with storage, a core decision to be taken for the development of this architecture is the underlying filesystem. To this respect, the fact that huge amounts of data are to be stored by the systems suggests the use of a distributed filesystem such as HDFS [3], built after the Google File System [4] as an opensource implementation forming part of Hadoop's core. The advantage of this filesystem is its ability to scale linearly as new nodes are added to the cluster and to manage replication automatically, which significantly simplifies the development tasks. Moreover, in the cases where semi-structured data is to be stored, HBase, an opensource implementation of Google Bigtable [5], turns to be a good option that automatically stores records on top of HDFS.

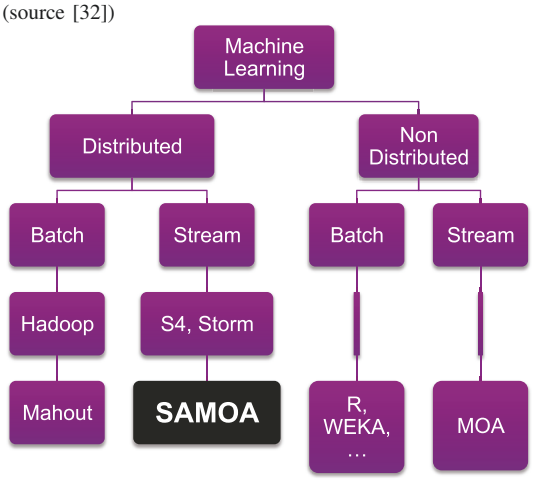
Moving to analytics, and for the matter of batch processing, the MapReduce framework invented by Google [1] and further built as an opensource development for Hadoop turns out to be a suitable approach, as it naturally handles data stored in HDFS and is able to process them in a parallel manner. Additionally, many programming tools have been developed which provide an abstraction over MapReduce, such as Hive [6] or Pig [7][8], thus simplifying the development. Finally, more recent approaches have appeared which try to address some of the issues of the original MapReduce, including reducing time costs. Some of these new tools include MapReduce++ [9], GraphLab [10] or Spark [11].

In the recent years, many works have been published which tackled the problem of performing machine learning over big datasets. Some of these works include performing analytics over Twitter [12], computing k-means clustering over big data in the cloud [13], providing recommendations [14][15], studying the behavior of tourists [16], performing sentiment analysis [17], minimizing product escapes in aerospace test environments [18], improving a predictive model in a healthcare domain [19], detecting astrophysical objects [20], discovering communities in social networks [21] and many more. Also, recent works have provided detailed studies of technologies for batch processing techniques over big data [22], as well as current applications and systems for this purpose [23] and proposed APIs for distributed machine learning [24].

Besides the technologies supporting batch-processing, this work also deals with stream analytics. This kind of real-time analytics require valuable responses of data in movement typically measured in milliseconds. Complex Event Processing (CEP) allows data analytics, such as detecting patterns while the data is still in motion, i.e. while events and transactions are still happening. In addition, in some environments a millisecond response-time advantage makes the difference. CEP techniques are not new, however, the significant attention that big data technologies are experiencing these days is going to definitely push them up.

The history of Stream Processing Engines (SPEs) date back to a decade ago. One of the earliest SPEs are Aurora [25] and STREAM [26], which proposed database management systems suitable to fit the requirements of monitoring applications where continuous input is received from different sources. A more recent generation of SPEs includes Borealis [27], an evolution of Aurora incorporating distributed computing; SPC [28], which is also distributed and supports non-relational operators; and SPADE [29], which provides a distributed system along with a language and a toolkit with predefined stream operators. The state-of-the-art generation of SPEs includes S4 [30], Storm and MOA [31]. More recently, SAMOA [32], which emerges as a combination of S4 and MOA, is a platform maintained by Yahoo! for analyzing and performing machine learning over big data streams.

Fig. 1. Machine learning taxonomy according to *Yahoo!*, where a classification of big-data machine learning technologies is made depending on whether they support batch or streaming processing, in a distributed or non-distributed fashion (source [32])



An image of how the different approaches and technologies mentioned in this section correlate among them can be seen in the machine learning taxonomy shown in figure 1.

The interest of real-time analytics in the market is growing [33], as it can be seen by looking at the acquisitions of Apama and StreamBase, CEP techniques that were purchased by the end of 2013 by Software AG and Tibco respectively. Software AG, a global leader in business processes, integration and big data, conducted a session at the IDC Big Data and Business Analytics Conference held in November 2013, during which they explored the value that organizations across industries can benefit from by accessing, analyzing and responding to business events at the right time, concluding that it will be a key market differentiator in this fast pace environment.

Increasingly remarkable interest in the area of stream processing can be seen by just staring at the key strategies adopted by the big Internet players such as Google, Microsoft or Amazon, who just released their commercial products on recent years. For instance, Google released BigQuery[1] [34], a cloud-based tool for quickly analyzing very large datasets; and has recently launched the Google Prediction API[2], to provide real-time predictions. According to Google, both these services working together allow to upload 100,000 rows of real-time data per second and analyze it in near real time, being this high streaming capacity the main advantage with most of their competitors. Another example of this emerging area is Amazon's product called Kinesis[3], a service launched

[1] https://cloud.google.com/products/bigquery/
[2] https://cloud.google.com/products/prediction-api/
[3] http://aws.amazon.com/kinesis/

in 2013 that scales elastically for real time processing of big data streaming, such as analyzing website clickstream usability engagement, which works over other Amazon Cloud products such as Redshift and Elastic MapReduce. Finally, at the time of writing this paper (July, 2014), Microsoft is announcing its Machine Learning services working over Microsoft Azure[4].

Besides the options provided by these big competitors, there are other commercial options such as IBM InfoSphere Streams[5] along with their propietary language SPL [35]; Adello[6] (formerly known as HStreaming), a platform for mobile advertising; SQLstream[7] or Splunk[8] to mention a few. Also, there are a significant number of alternative open-source projects, such as those provided by Twitter, like Storm[9], a stream-processing software for Hadoop, and Summingbird[10], a streaming MapReduce library working over Storm; or H_2O from 0xdata[11]. Other products involve Apache Spark Streaming[12] [36] and Kafka[13]. To give a rough idea about the importance and newness of this field, most of the projects described previously are not even in their first release (e.g. Storm current release is v0.9.02, Spark 1.0.0 was released this May 30th, 2014 and Kafka current last release is 0.8.1.1).

Moreover, a significant number applications performing real-time analytics over big data have been published over the last couple of years. To mention a few of them, there are works aimed at learning classifier chains for data mining [37], improving routing and navigation services [38], recommending musical preferences [39], detecting trends on Twitter and Bitly [40], filtering spam [41], processing image and video streams [42], analyzing streams of aviation data [43], monitoring underwater acoustics [44], discovering communities in Twitter during natural disasters [45], detecting issues in electric power systems [46], searching in logs [47] or summarizing microblogging data [48], among others. The latter four of the forementioned works use HBase and Hadoop, proving it to be a convenient storage system for performing streaming analysis; and some of the previous works also provide their products as a service, so that the real time analytics system can be accessed through a REST API.

Many other recently published works propose their own systems and architectures to provide real-time analytics over big data, either based on Storm [49], HBase [50] or specifically aimed at unstructured data [51]. Finally, other less specific works focus on reviewing the state-of-the-art of real-time platforms for distributed analytics [52] and looking at other research challenges and issues to develop a platform for analyzing big data on real time [53].

III. Our Proposal

As stated before, this work proposes the design of an architecture for developing machine learning tools which are bundled as a service, thus enabling cheap on-demand batch and stream analysis of big data. The architecture of this system is shown in figure 2. The exploitation of the system should be easy, however, as an added-value service, a team of data scientists and consultants would offer to analyze each customer's case in order to allow obtaining the most value from the data and to properly customize the system.

As it is shown in the figure, the architecture is composed of two major modules: the batch machine learning module and the stream machine learning module, which are themselves responsible for carrying out all the data processing tasks. In addition to these, the architecture provides a storage module, a dashboard, and a RESTful API. All these components are described in this section.

A. Batch Machine Learning Module

The batch machine learning module is suitable for those tasks which are not time-critical. When dealing with big data, some tasks can be expected to take seconds or even minutes to complete. In particular, three processing tasks are well suited to this module:

- Performing query analytics, where some information is extracted from the raw data. These analytics include, but are not limited to computing data distributions, operating over subsets of the data, mapping data to a different domain and generating data summaries to be passed to the dashboard for its visualization.
- Performing data clustering, i.e., in the case where the data involves several records and a distance function among them can be defined, these records can be grouped in several clusters, so that all records within a cluster are closer among them than with other records in an external cluster.
- Building machine learning models, in order to enable fast processing later by the stream machine learning module. Again, some examples of these models could be nave Bayes classifiers, bayesian networks, neural networks, decision trees, markov models, etc.

While this statement does not always hold, it could be said that in most cases, batch processing tasks will be implemented by means of MapReduce routines, which enable parallel processing of big data over a cluster of inexpensive nodes. However, progressive evolution to Spark is being done, as it outperforms Hadoop in many different cases [11].

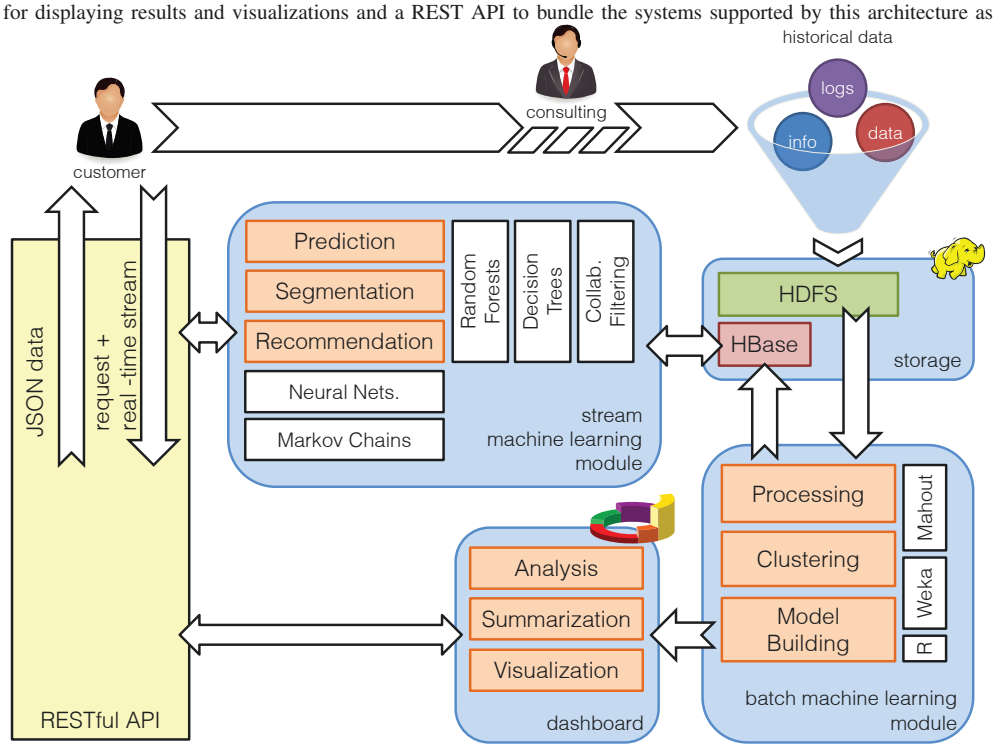
B. Stream Processing Module

The stream processing module is designed to attend the requirements of time-critical applications, so that it can execute certain tasks and provide a result within few milliseconds. In most cases, this module will take advantage of machine learning models previously built by the batch processor, thus applying a variety of state-of-the-art machine learning techniques along with the real-time stream of incoming data to generate an output in real time. In particular, this module adapts well to three tasks which require very fast responses:

- Carrying out prediction, i.e., making an assumption over a future event based on a set of historical events.

[4] http://azure.microsoft.com/en-us/campaigns/machine-learning/
[5] www.ibm.com/software/products/infosphere-streams
[6] https://www.adello.com/
[7] http://www.sqlstream.com/
[8] http://www.splunk.com/
[9] https://storm.incubator.apache.org/
[10] https://github.com/twitter/summingbird
[11] http://0xdata.com/h2o-2/
[12] https://spark.apache.org/streaming/
[13] http://kafka.apache.org/

Fig. 2. Architecture supporting machine learning over big data, including a storage module built over a distributed filesystem, batch- and stream-processing modules, a dashboard for displaying results and visualizations and a REST API to bundle the systems supported by this architecture as a service



- Performing classification or segmentation, so that given an instance of a certain user or record of any type and a set of classes or types, the system can assign that instance to one or more of those.
- Providing recommendations, such as suggesting a product to a user assuming that the user will be potentially interested in that product.

C. Storage Module

The Hadoop Distributed File System (HDFS) settles the basis for this module. HDFS itself is suitable for storing raw data (such as logs, historical data, text files or any other kind of unstructured data such as documents or pictures), as it provides reliable distributed storage over a cluster of inexpensive computers, relieving developers from having to manage data distribuion and replication among each node.

However, when structured or semi-structured information is to be stored, Apache HBase provides a convenient abstraction as it is a non-relational database built on top of HDFS. Also, the fact that it indexes row keys enables running faster queries on it, while HDFS itself is better suited to performing reads over entire big files. Without loss of generality, it could be said that raw data will mostly be used for batch processing, and also for retrieving semi-structured information or building models which will be stored in HBase, a process for which the batch machine learning module is responsible.

D. Dashboard

The dashboard is a module which provides an intuitive and graphical way to visualize some analytics performed over the data by the batch processing module, and can also provide interesting results which can be summarized or displayed to the customer, thus revealing potentially relevant insights about the data. Moreover, this visualization will also provide feedback to the user, who could decide whether the input data is appropiate, and take corrective measures if required.

E. RESTful API

The RESTful API allows offering real-time analysis tools as a service, so that specific analytics could be performed on demand. Moreover, the customer could establish a pipe so that all the data generated is automatically streamed to the API, taking the most advantage from the real-time features.

IV. Use Cases

A prototype system based on the proposed architecture has been already built, in order to be used as a proof of concept of these emerging technologies. This system has been used to successfully attend two different use cases based on real customer scenarios, serving respectively to the purposes of displaying ads in a website and predicting users' behavior in a social ecosystem. In this section, these use cases are described, providing some hints on how the proposed architecture can support a system to unleash value from raw data and eventually increase the conversion rate in different domains.

A. Web Recommender System

The first use case involves the development of a system to display real-time advertisements to web visitors, so that these ads are potentially interesting to users. The customer can choose whether the system will provide recommendations using content-based filtering or collaborative filtering. The

Fig. 3. Architecture of the web recommender system. Raw logs are stored in HDFS and a clustering batch process extracts categories of both webs and users in a non-time-critical fashion. Then, the stream processor is able to provide recommendations in real-time as they are requested

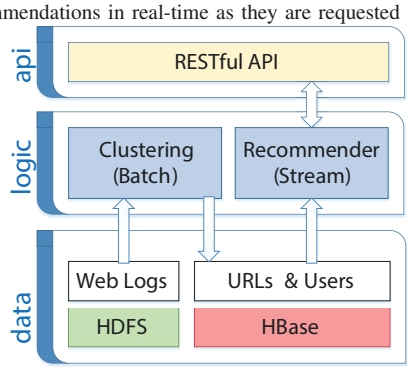
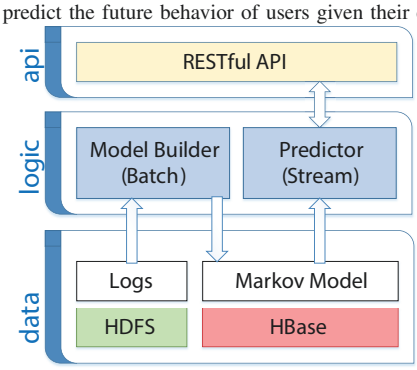


Fig. 4. Architecture of the user behavior prediction system. Event logs are stored in HDFS and a job periodically processes them in order to update a probabilistic model stored in HBase. Then, the stream processor is able to use the model to predict the future behavior of users given their current behavior



architecture for the web recommender system is shown in figure 3. It is worth noting that new HTTP requests come with a very high frequency, and ads should be displayed as soon as the page is loading, so real-time turns out to be an important requirement for this system.

The raw logs, recorded from historical HTTP requests are stored *as-they-are* in HDFS, along with some metadata of the visited webpage, such as keywords or the description. A batch process, which is not time-critical, is executed periodically in order to extract web categories and user categories from these, i.e., to group webpages which tends to be topic-related and to group users visiting similar websites. This task is performed by the batch machine learning module, and in particular, a k-means implementation included in Mahout was in charge of computing both these web and users categories.

A real-time processor was developed within the stream machine learning module, in order to take advantage of the result of the clustering so that, when a new HTTP request is received, both the website and the user can be rapidly categorized (if they were not before) and a useful recommendation can be provided. The recommendation would consist of one or more URLs in which the user may be interested, so that an advertisement can be displayed in the website.

The API for this service allows to log new HTTP requests, which results in updating the web and users categories. Also, it provides recommendations using collaborative or content-based filtering from a set of specified websites.

B. User Behavior Prediction System

The second use case is based on a social game which was interested in predicting the future behavior of his users given a historical of past events. To do so, a variable-order Markov model [54] is learnt, as it is able to capture how a variable number of past events are able to influence future events from a probabilistic approach, and it has been successfully applied before to other domains such as predicting drivers' behavior [55] or filtering spam [56]. The architecture for this system is shown in figure 4. Again, this application requires from real-time processing of data, as gamers may perform several events per second, and thus a fast response is critical.

For this use case, historical records containing the events carried out by each user was provided, which would first be stored in HDFS. These logs are periodically processed by a batch process, which converts the event logs into event sequences, and uses these for training (or updating) a probabilistic model based on variable-order Markov and storing it in HBase.

As the user executes new events, the service is called and the stream machine learning module is in charge to predict how the user will behave in the future, so that the site owner can take specific actions in order to condition the user behavior. For instance, if the user is supposed to be leaving the site (i.e., to turn into a churner) in the short future, an special offer could be suggested in order to try to retain the user. In order to try to infer the shortcoming behavior of a user, the stream processor will use the previously computed Markov model along with the last events he or she performed.

The API for this service allows to record new events sequences which update the Markov probability matrix used for predicting the gamer behaviors (either by providing the next most likely event or all possible next events along with their probability to occur). In addition, it allows to recommend an action than influences user behavior to a specified one.

Similar works can be found in the literature which proposes systems to learn behavioral models for videogames [57] and predicting churn [58], while these are not really focused on the big amounts of recorded data.

V. Evaluation

After the machine learning architecture was applied for the development of the use cases described in the previous section, an evaluation was designed in order to validate it. Successful results would be attained if the web service provided real-time responses whenever it is required and, in the case of prediction, results were accurate (in the case of recommendation, this accuracy has not been tested so far). The purpose of this evaluation is to provide a hint of how the architecture performs when attending real-time requests. Still, a more robust evaluation is planned using benchmarks when available, and examining how performance evolves when the system is scaled horizontally (i.e., new nodes are attached to it).

TABLE I. AVERAGE AND MEDIAN TIMES (IN MS.) AS WELL AS THE STANDARD DEVIATION FOR THE RECOMMENDER SERVICE, BOTH WITH SEQUENTIAL AND WITH CONCURRENT (10 AND 30) REQUESTS

	Average	Median	Std. Dev.
Sequential	447.77	435	47.84
Concurrent (10)	538.80	540	40.76
Concurrent (30)	802.93	792	89.58

A. Web Recommender System

A preliminary evaluation of the web recommender system has been performed using a dataset with 200,000 URLs (clustered in 100 different categories), 10,000 users (clustered in 20 different profiles) and 2,000 ad campaigns, each of them having at least one but potentially several different ads. The running environment involves a single-node cluster with 8 processing cores and 16GB of RAM virtualized over VMware ESXi 5.0. Hortonworks HDP 2.1 was chosen as the preferred distribution, which provides Hadoop 2.4 and HBase 0.98, while the web service was deployed over JBoss AS 7.

Table I shows the average and median times measured in milliseconds, as well as the standard deviation when the recommender service is called 5,000 times with different input parameters. Additionally, the times for each call are recorded also when several requests are performed concurrently in batches of 10 and 30 calls respectively. As it can be seen, average times are always under 1 second even in those cases where 30 concurrent requests are happening, thus providing acceptable real-time responses. Also, there are no significant differences in the execution time between different calls, as it is shown by the small standard deviation and the proximity between the median and the average.

B. User Behavior Prediction System

A preliminary evaluation of the prediction system has been carried out in order to check whether it successes when trying to predict user behavior in the domain of online social games, and also whether the prediction service is able to scale to attend multiple concurrent requests.

In order to validate the prediction system, data obtained from a social online game has been used to train and test the prediction model. The whole dataset was divided in a training set and a test set of 70%-30% of the total size respectively.

In particular, the dataset used recorded events performed by gamers, which can be modeled as a sequence. For this reason a variable-order Markov model was used as it is suitable to learn from a sequence of events of different lengths. For this dataset, the behavior of 10,000 users were recorded during 20 days. This time, running environment involves a 4-nodes cluster with 8 processing cores and 16GB of RAM each virtualized over VMware ESXi 5.0. As in the previous use case, the installed distribution was Hortonworks HDP 2.1 and the web service was deployed over JBoss AS7.

Table II shows the results in terms of precision and recall of the different events (names are hidden and a numeric identifier is shown instead), as well as the relative frequency of each of these events in the dataset. Moreover, figure 5 plots the recall of each event versus its relative frequency. As it can be seen, there is a significant advantage over simple "random guess", which translates into the fact that the probabilistic model is able to successfully learn behavioral patterns performed by the users from the events sequences themselves.

TABLE II. PRECISION AND RECALL FOR EACH EVENT IN THE ONLINE SOCIAL GAME TEST DATASET, ALONG WITH THEIR RELATIVE FREQUENCY OF APPEARANCE

	Frequency	Precision	Recall
#1	2.89%	25.45%	8.19%
#2	1.25%	66.08%	33.98%
#3	38.87%	82.09%	83.20%
#4	7.52%	6.84%	26.08%
#5	0.5%	96.37%	93.95%
#6	0.05%	100%	2.86%
#7	7.16%	88.11%	81.19%
#8	0.82%	61.86%	18.58%
#9	1.82%	71.27%	77.78%
#10	33.66%	47.03%	67.92%
#11	3.12%	87.42%	49.07%
#12	0.34%	98.93%	99.78%
#13	1.98%	55.95%	48.62%

Fig. 5. Recall vs. relative frequency for each event of the online social game domain. Results show that the prediction system is able to learn from past events to predict behavior, as in most cases there is a significant improvement over the accuracy provided if random guess was used.

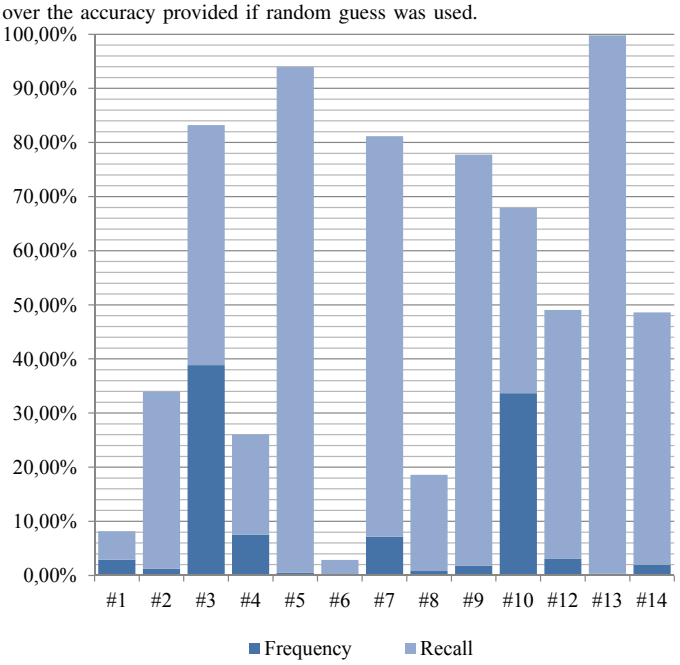


Table III shows the average and median times measured in milliseconds, as well as the standard deviation when the prediction service is called 3,000 times, each one requesting a behavior prediction for a different user. Again, the times for each call are recorded also when several requests are performed concurrently in batches of 10 and 30 calls. As it can be seen, average times are always far under 500 milliseconds even when many requests are executed in parallel, which turns out to be a very fast response.

TABLE III. AVERAGE AND MEDIAN TIMES (IN MS.) AS WELL AS THE STANDARD DEVIATION FOR THE PREDICTION SERVICE, BOTH WITH SEQUENTIAL AND WITH CONCURRENT (10 AND 30) REQUESTS

	Average	Median	Std. Dev.
Sequential	20.29	18	10.55
Concurrent (10)	98.43	79	283.21
Concurrent (30)	235.04	228	77.98

VI. CONCLUSIONS

This work has introduced and described a novel architecture for performing machine learning over big data streams. The architecture provides reliable persistent storage of data over the Hadoop Distributed File System and HBase, which provides technical feasibility to scale the system in an easy manner in case it is required. The core of the architecture is comprised of the batch- and stream-processing modules, which provide machine learning tools and algorithms, so that developers can easily take advantage of them to carry out tasks such as prediction, clustering, recommendation, classification, etc. A dashboard will be responsible for summarizing the results of the batch analysis and displaying it to the user. Finally, the architecture provides a RESTful API which allows to bundle machine learning systems supported by this architecture as a service, so that the user can perform requests while stream big data is flowing. This paper does not propose new machine learning tools or algorithms, although this could be an interesting future research line. However, it provides an architecture that facilitates the task of analyzing and extracting value from big data to customers by means of a service that abstracts most of the complexities underlying Hadoop-based tools for machine learning.

Two use cases have been proposed and two systems have been developed to attend these use cases. The first of them involves a recommender system for web advertising, which must provide real-time recommendations of ads to be displayed to users as soon as the website loads. The second use case describes a scenario where the behavior of social gamers is studied, in order to learn from their history of events to eventully provide predictions of their future behavior, information which could later be used to condition users' actions to increase income or avoid churning.

Once both systems were developed, a preliminary evaluation was designed with the purpose of validating whether response times were convenient for real-time requests and, in the case of prediction, whether the model was able to accurately predict future events over a test set. Results are encouraging, as response times are always under one second even when up to 30 concurrent requests take place, and the accuracy of the behavior prediction system clearly shows its ability to learn from historical data. Still, studying the performance of the recommender system with real users and comparing how the system is able to scale out horizontally is left as future work resulting in the design and execution of a more complete evaluation, as well as a side-by-side comparison of the game prediction system using public datasets.

Finally, additional work involves re-implementing batch machine learning algorithms over Spark, as it has shown to outperform MapReduce in many cases. This migration is partially solved as Mahout announced that versions starting from 1.0 will reject new MapReduce algorithm implementations, which in turn will run over Apache Spark. Also, it is planned to study how deploying a Storm topology contributes to the performance of the stream-processing module.

ACKNOWLEDGMENT

This research work is part of *Memento Data Analysis* project, co-funded by the Spanish Ministry of Industry, Energy and Tourism with identifier TSI-020601-2012-99.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proc. 6th Symp. Operating Syst. Des. and Impl. (OSDI'04)*, 2004, pp. 137–150.

[2] H. Hu, Y. Wen, T. Chua, and X. Li, "Toward Scalable Systems for Big Data Analytics: a Technology Tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014.

[3] K. Shvachko, "The Hadoop Distributed File System," in *Proc. 2010 IEEE 26th Symp. Mass Storage Syst. and Technol. (MSST'10)*, 2010, pp. 1–10.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in *Proc. 19th ACM Symp. Operating Syst. Principles (SOSP'03)*, 2003, pp. 29–43.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," in *Proc. 7ty Symp. Operating Syst. Des. and Impl. (OSDI'06)*, 2006, pp. 205–218.

[6] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Lui, P. Wyckoff, and R. Murthy, "Hive - A Warehousing Solution Over a Map-Reduce Framework," *Proc. VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[7] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a High-Level Dataflow System on Top of Map-Reduce: the Pig Experience," *Proc. VLDB Endowment*, vol. 2, no. 2, pp. 1414–1425, 2009.

[8] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig Latin: a Not-so-Foreign Language for Data Processing," in *Proc. 2008 ACM SIGMOD Int. Conf. Manag. of Data (SIGMOD'08)*, 2008, pp. 1099–1110.

[9] G. Zhang, C. Li, Y. Zhang, C. Xing, and J. Yang, "MapReduce++: Efficient Processing of MapReduce Jobs in the Cloud," *J. of Comput. Inf. Syst.*, vol. 8, no. 14, pp. 5757–5764, 2012.

[10] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed GraphLab: a Framework for Machine Learning and Data Mining in the Cloud," *Proc. VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.

[11] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets," in *Proc. 2nd USENIX Conf. Hot Topics in Cloud Comput. (HotCloud'10)*, 2010.

[12] J. Lin and A. Kolcz, "Large-Scale Machine Learning at Twitter," in *Proc. 2012 ACM SIGMOD Int. Conf. Manag. of Data (SIGMOD'12)*, 2012, pp. 793–804.

[13] R. Steves, "K-means Clustering in the Cloud – A Mahout Test," in *Proc. 2011 IEEE Workshops of Int. Conf. Advanced Inf. Netw and Appl. (WAINA'11)*, 2012, pp. 514–519.

[14] L. Ma, E. Haihong, and K. Xu, "The Design and Implementation of Distributed Mobile Points of Interest (POI) based on Mahout," in *Proc. 6th Int. Conf. Perv. Comp. and Apps. (ICPCA'11)*, 2011, pp. 99–104.

[15] C.-R. Lee, T. Hua, and Y.-F. Chang, "Enhancing Accuracy and Performance of Collaborative Filtering Algorithm by Stochastic SVD and its MapReduce Implementation," in *Proc. 2013 IEEE 27th Int. Conf. Parallel and Distrib. Process. Symp. Workshops & PhD Forum (IPDPSW'13)*, 2013, pp. 1869–1878.

[16] R. Irudeen and S. Samaraweera, "Big Data Solutions for Sri Lankan Development: A Case Study from Travel and Tourism," in *Proc. 2013 Int. Conf. Advances in ICT for Emerging Regions (ICTer'13)*, 2013, pp. 207–216.

[17] B. Liu, E. Blasch, Y. Chen, D. Shen, and G. Chen, "Scalable Sentiment Classification for Big Data Analysis using Naïve Bayes Classifier," in *Proc. 2013 IEEE Int. Conf. Big Data*, 2013, pp. 99–104.

[18] T. Armes and M. Refern, "Using Big Data and Predictive Machine Learning in Aerospace Test Environments," in *Proc. 2013 IEEE AUTOTESTCON*, 2013, pp. 301–305.

[19] L. Li, S. Bagheri, H. Goot, A. Hasan, and G. Hazard, "Risk Adjustment of Patient Expenditures: A Big Data Analytics Approach," in *Proc. 2013 IEEE Int. Conf. Big Data*, 2013, pp. 12–14.

[20] P. Huijse, P. Estevez, P. Protopapas, J. Principe, and P. Zegers, "Computational Intelligence Challenges and Appl. on Large-Scale Astronomical Time Series Databases," *IEEE Comput. Intell. Magazine*, vol. 9, no. 3, pp. 27–39, 2014.

[21] J. Shi, W. Xue, W. Wang, and Y. Zhang, "Scalable Community Detection in Massive Social Networks using MapReduce," *IBM J. of Research and Develop.*, vol. 57, no. 3, 2013.

[22] K. Hammond and A. Varde, "Cloud Based Predictive Analytics: Text Classification, Recommender Systems and Decision Support," in *Proc. 2013 IEEE 13th Int. Conf. Data Mining Workshops (ICDMW'13)*, 2013, pp. 607–612.

[23] T. Condie, P. Mineiro, N. Polyzotis, and M. Weimer, "Machine Learning on Big Data," in *Proc. 2013 IEEE 29th Int. Conf. Data Eng. (ICDE'13)*, 2013, pp. 1242–1244.

[24] E. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. Gonzalez, M. Franklin, M. Jordan, and T. Kraska, "MLI: An API for Distributed Machine Learning," in *Proc. 2013 IEEE 13th Int. Conf. Data Mining (ICDM'13)*, 2013, pp. 1187–1192.

[25] D. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: A New Model and Architecture for Data Stream Management," *The Int. J. on Very Large Data Bases*, vol. 12, no. 2, pp. 120–139, 2003.

[26] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, K. Ito, R. Motwani, U. Srivastava, and J. Widom, "STREAM: The Stanford Data Stream Management System," Stanford InfoLab, Tech. Rep., 2004.

[27] D. J. Abadi, Y. Ahmad, M. Balazinska, M. Cherniack, J. hyon Hwang, W. Lindner, A. S. Maskey, E. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik, "The Design of the Borealis Stream Processing Engine," in *Proc. 2nd Biennial Conf. Innovative Data Syst. Research (CIDR'05)*, 2005, pp. 277–289.

[28] L. Amini, H. Andrade, R. Bhagwan, F. Eskesen, R. King, Y. Park, and C. Venkatramani, "SPC: A Distributed, Scalable Platform for Data Mining," in *Proc. 4th Int. Workshop on Data Mining Standards, Services and Platforms (DMSSP'06)*, 2006, pp. 27–37.

[29] B. Gedik, P. S. Yu, H. Andrade, M. Doo, and K. lung Wu, "SPADE: The System S Declarative Stream Processing Engine," in *Proc. 2008 ACM SIGMOD Int. Conf. Manag. of Data (SIGMOD'08)*, 2008, pp. 1123–1134.

[30] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed Stream Computing Platform," in *Proc. 2010 IEEE 10th Int. Conf. Data Mining Workshops (ICDMW'10)*, 2010, pp. 170–177.

[31] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *J. of Mach. Learn.*, vol. 11, pp. 1601–1604, 2010.

[32] G. D. F. Morales, "SAMOA: A Platform for Mining Big Data Streams," in *Proc. 22nd Int. Conf. World Wide Web Companion (WWW'13)*, 2013, pp. 777–778.

[33] G. Rao, "Big Data and Real Time Analytics," in *Proc. 2011 Int. Conf. Recent Trends in Inf. Technol. (ICRTIT'11)*, 2011, p. 2.

[34] K. Sato, "An Inside Look at Google BigQuery," Google, Inc., Tech. Rep., 2012.

[35] M. Hirzel, H. Andrade, B. Gedik, G. Jaques-Silva, R. Khandekar, V. Kumar, M. Mendell, H. Nasgaard, S. Schenider, and R. S. and. K.L. Wu, "IBM Streams Processing Language: Analyzing Big Data in Motion," *IBM J. of Research and Develop.*, vol. 57, no. 3, 2013.

[36] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica, "Discretized Streams: an Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters," in *Proc. 4th USENIX Conf. Hot Topics in Cloud Comput. (HotCloud'12)*, 2012.

[37] J. Xu, C. Tekin, and M. van der Schaar, "Learning Optimal Classifier Chains for Real-Time Big Data Mining," in *Proc. 51st Annu. Allerton Conf. Comm., Control and Comput. (Allerton'13)*, 2013, pp. 512–519.

[38] M. Bakillah, A. Mobasheri, S. Liang, and A. Zipf, "Towards an Efficient Routing Web Processing Service through Capturing Real-Time Road Conditions from Big Data," in *Proc. 5th Comp. Sci. and Electron. Eng. Conf. (CEEC'13)*, 2013, pp. 152–155.

[39] E. Olmezogullari, I. Ari, O. Celebi, and S. Ergut, "Data Stream Mining to Address Big Data Problems," in *Proc. 21st Signal Process. and Commun. Appl. Conf. (SIU'13)*, 2013.

[40] T. Chardonnens, P. Cudre-Mauroux, M. Grund, and B. Perroud, "Big Data Analytics on High Velocity Streams: a Case Study," in *Proc. 2013 IEEE Int. Conf. on Big Data*, 2013, pp. 784–787.

[41] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and Evaluation of a Real-Time URL Spam Filtering Service," in *Proc. 2011 IEEE Symp. Security and Privacy (SP'11)*, 2011, pp. 447–462.

[42] S. Najmabadi, M. K. an Z. Wang, Y. Baroud, and S. Simon, "Stream Processing of Scientific Big Data on Heterogeneous Platforms – Image Analytics on Big Data in Motion," in *Proc. 2013 IEEE 16th Int. Conf. Comput. Sci. and Eng. (CSE'13)*, 2013, pp. 965–970.

[43] S. Ayhan, J. Pesce, P. Comitz, D. Sweet, S. Bliesner, and G. Gerberick, "Predictive Analytics with Aviation Big Data," in *Proc. 2013 Integrated Commun., Navigation and Surveillance Conf. (ICNS'13)*, 2013.

[44] J. Hayes, H. Kolar, A. Akhriev, M. Barry, M. Purcell, and E. McKeown, "Real-Time Analysis and Management of Big Time-Series Data," *IBM J. of Research and Develop.*, vol. 57, no. 3, 2013.

[45] Y. Huand, H. Dong, Y. Yesha, and S. Zhou, "A Scalable System for Community Discovery in Twitter During Hurricane Sandy," in *Proc. 2014 IEEE/ACM 14th Int. Symp. Cluster, Cloud and Grid Comput. (CCGrid'14)*, 2014, pp. 893–899.

[46] F. Zhao, G. Wang, C. Deng, and Y. Zhao, "A Real-Time Intelligent Abnormity Diagnosis Platform in Electric Power System," in *Proc. 16th Int. Conf. on Advanced Commun. Tech. (ICACT'14)*, 2014, pp. 83–87.

[47] B. Jun, "Feasibility Analysis of Big Log Data Real Time Search based on HBase and ElasticSearch," in *Proc. 9th Int. Conf. Natural Comput. (ICNC'13)*, 2013, pp. 1166–1170.

[48] S. Lee, S. Shakya, R. Sunderraman, and S. Belkasim, "Real Time Micro-Blog Summarization based on Hadoop/HBase," in *Proc. 2013 IEEE/WIC/ACM Int. Joint Conf. Web Intell. (WI'13) and Intell. Agent Technol. (IAT'13)*, 2013, pp. 46–49.

[49] W. Yang, X. Liu, L. Zhang, and L. Yang, "Big Data Real-Time Processing based on Storm," in *Proc. 2013 IEEE 12th Int. Conf. Trust, Security and Privacy in Comp. and Comm. (TrustCom'13)*, 2013, pp. 1784–1787.

[50] F. Li, M. Ozsu, G. Cheng, and B. C. Ooi, "R-Store: a Scalable Distributed System for Supporting Real-Time Analytics," in *Proc. 2014 IEEE 30th Int. Conf. Data Eng. (ICDE'14)*, 2014, pp. 40–51.

[51] J. Kim, N. Kim, B. Lee, J. Park, K. Seo, and H. Park, "RUBA: Real-Time Unstructured Big Data Analysis Framework," in *Proc. 2013 Int. Conf. ICT Convergence (ICTC'13)*, 2013, pp. 518–522.

[52] A. Osman, M. El-Refaey, and A. Elnaggar, "Towards Real-Time Analytics in the Cloud," in *Proc. 2013 IEEE 9th World Congr. Services (SERVICES'13)*, 2013.

[53] R. Ranjan, "Streaming Big Data Processing in Datacenter Clouds," *IEEE Cloud Comput.*, vol. 1, no. 1, pp. 78–83, 2014.

[54] R. Begleiter, R. El-Yaniv, and G. Yona, "On Prediction using Variable Order Markov Models," *J. of Artificial Intell. Research*, vol. 22, no. 1, pp. 385–421, 2004.

[55] X. Guangtao, L. Zhongwei, Z. Hongzi, and L. Yunhuai, "Traffic-Known Urban Vehicular Route Prediction based on Partial Mobility Patterns," in *Proc. 2009 15th Int. Conf. Parallel and Distrib. Syst. (ICPADS'09)*, 2009.

[56] A. Bratko, G. V. Cormack, B. Filipič, T. R. Lynam, and B. Zupan, "Spam Filtering using Statistical Data Compression Models," *J. of Mach. Learn. Research*, vol. 7, pp. 2673–2698, 2006.

[57] J.-L. Hsieh and C. T. Sun, "Building a Player Strategy Model by Analyzing Replays of Real-Time Strategy Games," in *Proc. 2008 Int. Joint Conf. Neural Netw. (IJCNN'08)*, 2008, pp. 3106–3111.

[58] C. Bauckhage, K. Kersting, R. Sifa, C. Thurau, A. Drachen, and A. Canossa, "How Players Lose Interest in Playing a Game: An Empirical Study Based on Distributions of Total Playing Times," in *Proc. 2012 Int. Conf. Comput. Intell. and Games (CIG'12)*, 2012, pp. 139–146.