Real-Time Analysis of Students' Activities on an E-Learning Platform based on Apache Spark

Abdelmajid Chaffai
RITM LAB, CED ENSEM
Hassan II University
Casablanca, Morocco

Larbi Hassouni
RITM LAB, CED ENSEM
Hassan II University
Casablanca, Morocco

Houda Anoun
RITM LAB CED ENSEM
Hassan II University
Casablanca, Morocco

***Abstract*—Real time analytics is the capacity to extract valuables insights from data that comes continuously from activities on the web or network sensors. It is largely used in web based business to drive decisions based on user's experiences, such dynamic pricing and personalized advertising. Many universities have adopted web based learning in their learning process. They use data-mining techniques to better understand students' behavior, and most of the tools developed are based on historical and stored data, and do not allow real time reactivity. Online activities of learners generate at high speed a huge amount of data in form of users' interactions which have all characteristics to be considered as Big data. Deal with volume and velocity of these data in order to inform and enable decisions-makers to act at right time lead us to use new methods to capture E-Learning data, and process it in real time.**

This paper focuses on the design and implementation of modern and hybrid real time data pipeline architecture using Apache Flume to collect data, Apache Spark as an unified engine computation for performing analytics on students' activities data and Apache Hive as a data warehouse for storing the processed data and for use by various reporting tools. To conceive this platform we conduct an experiment on Moodle database source.

Keywords*—Real time analytics; e-learning; big data; Hadoop; spark; Moodle; change data capture; streaming; data visualization clustering

I. INTRODUCTION

E-Learning is a revolutionary and very promising field that brings about a radical change in the field of learning. Web based technologies are used to create virtual classrooms with attractive materials and resources, and provides a wide range of solutions that support the learning process and services that are accessible anytime from anywhere.

Interactions of students with an E-Learning platform often come in three forms:

- Learner-learner
- Learner-instructor
- Learner-content

Learning Analytics (LA) is a recent field of research and development of tools and technologies that help to analyze and understand the interactions of learners with educational resources. In the first international Conference on Learning Analytics and Knowledge (LAK 2011), it was defined as "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs" [1]. A related domain is Educational data mining (EDM) which is a data-driven field defined in the community site [2] as "Educational Data Mining is an emerging discipline, concerned with developing methods for exploring the unique and increasingly large-scale data that come from educational settings, and using those methods to better understand students, and the settings which they learn in." LA aims to improve E-Learning [3] based on the analysis of the learners' behavior during their interactions with the course.

As more the learning in Higher education are occurring on the web, online activities generate, at high speed, huge amounts of information in the form of users' traces. Deal with volume and velocity of data in order to extract valuable information that can support real-time decision making, lead us to design a modern and flexible architecture that can manage and scale to the continuous stream data.

In this work, we propose a solution for the near-real time analysis of students' activities on a web based learning platform, the most widely used in Moroccan higher education institutions which is Moodle. For this, we have built a system of complete Data Analytics Pipeline which is composed of three main layers. The first layer ensures the data capture from Moodle database. The second layer performs real time processing. The third layer provides a flexible data persistence which can be used by different reporting tools.

All operations are executed in a distributed environment on inexpensive hardware. We use open source technologies such Apache Spark as the main computing engine and Hive to conceive the data warehouse on top of Hadoop cluster.

Analyzing data stream that come continuously from the Moodle platform can greatly help us to track students' progress in courses and detect the students at risk. It can also allow us to monitor the daily health of the E-Learning platform by using fresh reports which can be useful to deduct smart ideas in order to redefine the decisions strategies at right time by adjusting and improving the courses content that respond to students' needs.

The rest of the paper is structured as follows. Section 2 defines big data analytics. Section 3 discusses related work. Section 4 presents the tools used in our work. Section 5 presents the structure of the proposed system and the data processing methodology adopted. Then, it presents and

discusses the experiment results. Finally, Section 6 concludes the paper and describes the future research directions.

II. Big Data Analytics

Big data [4] is a huge amount of data that is generated from various sources. It may be structured when data come from flat files or relational databases, and unstructured or semi structured when data come from the web activities or equipment sensors. Acquiring data very fast does not create the value to the business [5], it needs additional efforts to be meaningful. Big data analytics is the process to apply statistical analysis, data mining, predictive analytics, and text mining on large amount of data using a distributed platform. It depends on speed at which data arrives, and can be divided in two categories [6]:

1) Batch Processing: Computation and analysis are applied on data that comes in big batches, then fixed and stored in distributed file system. This type of processing is largely used to learn from historical data by using clustering and classification techniques to create machine learning models which can be applied on new data.

2) Streaming Processing: Computation and analysis are applied in real time on recent data that come in continuous records. There are two distinct approaches to analyze live data. The first is to process each record individually, and the second is to split the input data in discretized units called mini-batch according to the interval batch. Stream processing solution must be connected with the source in real time in order to continuously ensure the capturing data.

III. Related Work

P.K. Udupi et al. [7] proposed a smart learning system model, they describe the possibilities to integrate the E-Learning paradigm with the big data analytics concept and smart utilization. The proposed system contains three layers of different technology framework. The first layer is an E-Learning framework which contains the information and data of user performance evaluation. The second layer is a big data framework which performs a set of different tasks like data extraction, data process and analysis. The third layer is a smart technology framework which enables support of technology need for capturing, predicting, analyzing, decision making and initiates necessary actions as control parameters.

B. Logica et al. [8] lead a study where they discuss the benefits of the use of big data technologies, in order to resolve the problem of managing the massive increase in the produced data volume in educational setting and extracting value from these data to enhance the learning process. They proposed a model for big learning data on cloud architecture based on Hadoop cluster, which can be integrated with the existing Learning Management System (LMS) that the universities usually already own. The different levels of the proposed architecture are designed for collecting any type of data, processing them using Hadoop cluster, performing classification on data stored, and exploring unstructured data using the graphical Gephi tool.

Sunita B Aher et al. [9] proposed a framework for recommendation of courses in E-Learning system Moodle. They use the enrolled data related to a specific set of courses collected from Moodle database. They use different machine learning algorithms: classification, association rules, and clustering to produce a final model for recommendation. All steps of building the dataflow and model are performed on Weka.

Yassine Tabaa et al. [10] described a learning analytics system for MOOCs based on Hadoop cluster deployed on a private cloud. The main core component of this system is the analytics engine which relies on Map and Reduce model programming, for performing many different analytics jobs, on data that comes from relational database, by using a data integrator based on Apache Sqoop for bulk transfer data from sql sources to HDFS. The analytics platform can help the decision-makers to early identify the students at risk.

San et al. [11] conducted a study in the field of smart grid research. They proposed a complete automation system, where large pool of sensors is embedded in the existing power grids system for controlling and monitoring it by utilizing modern information technologies. Data used in the experiment is in form of times series data available from Texas Synchrophasor Network. The proposed solution uses Apache Kafka to ingest data in real time into the processing layer based on Apache Spark, responsible to perform analytics in fast way. Computation is done in parallel across the cluster of machines.

IV. Tools

A. Apache Hadoop

Apache Hadoop [12], [13] is an open source framework written in java. It is used to build a cluster for both distributed storage and computation on inexpensive hardware. Hadoop is a master slave architecture which hides technical complexity with high level abstraction in terms of network I/O operations management, fault tolerance and easy horizontal scalability. The main subsystems of Hadoop distributed system are (see Fig. 1):

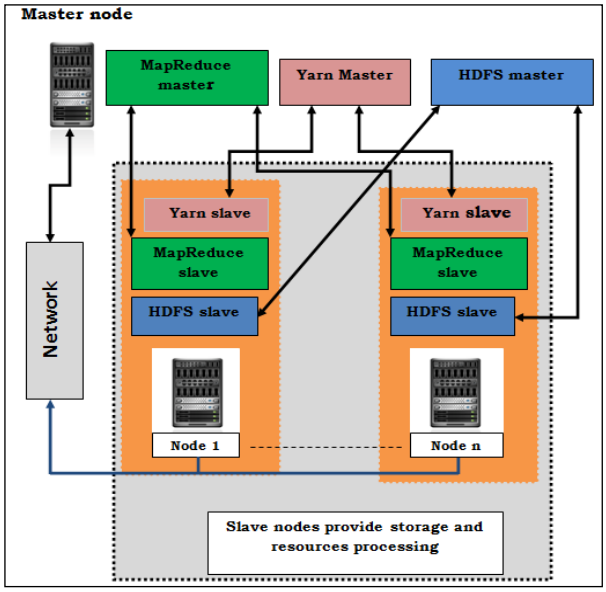


Fig. 1. Hadoop architecture.

- *HDFS*: It is a distributed file system, inspired from the Google file system GFS [14].When data is stored in HDFS, it is divided into a set of blocks over different nodes of the cluster. The default size of a block is 128 MB.
- *Yarn (Yet Another Resource Negociator)*: It is a distributed resource manager introduced in Hadoop version 2.
- *MapReduce*: It is a distributed and batch-based computing model developed after Google paper on MapReduce [15]. It allows parallelizing the job in small functions map and reducing, and moving the tasks to data locality across a cluster. MapReduce in Hadoop version 2 runs as Yarn application.

B. Apache Hive

Apache Hive [16], [17] is a data warehouse and an analysis system initially developed at Facebook [18]. It allows query and manage large datasets stored in Hadoop distributed cluster using a language called Hive Query Language (HQL) similar to SQL. Hive converts the queries in one or more MapReduce jobs that are executed on Hadoop cluster and returns the results to the user. Hive stores all metadata in a relational database, and uses by default Derby which is an embedded Java relational database. We used MySQL because Derby cannot be used in a multi-user environment.

C. Apache Spark

Apache Spark [19], [20] is an open source distributed Framework built in Scala, developed at the University of California Berkeley. It is a Java Virtual Machine designed for fast data processing in the main memory of nodes in the cluster. It can interact with HDFS and Hive and can run as YARN application. The strength of spark resides in its programming model based on high level abstraction of representing a data structure in cluster memory called Resilient Distributed Dataset. RDD [21] is the main component of Spark core. It is resilient because it is capable to rebuild data in case of failures in cluster. RDD is an immutable distributed collection of objects partitioned across different nodes of cluster, and can be created in different ways from external sources or in local and from transformations or actions on existent RDDs. Spark contains several components built on its core like Spark SQL, Spark Streaming, MLlib (Machine Learning library), and GraphX (graph processing), thus it offers to programmers an unified programming [22] platform. This is the main motivation for choosing it to build our system. It allows data sharing between jobs instead of storing intermediate results in the disk compared to MapReduce; and is well suited for iterative operations. Any application submitted to Spark cluster which is master/workers architecture, activates five elements in the following order (see Fig. 2):

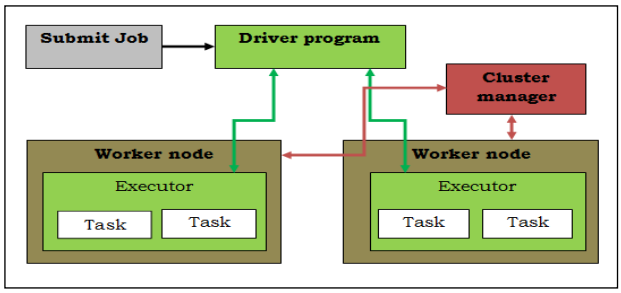


Fig. 2. Spark components architecture.

1) Driver program: Any program submitted to spark starts with an instantiation of SparkContext object, which is the main entry point to use spark library. SparkContext object use an instance of SparkConf which allows setting parameters and the required resources to run the application.

2) Workers: Worker is a slave node, which provides resources such computing (CPU), storage, and memory.

3) Cluster manager: Spark uses a cluster manager to allocate cluster resources for executing a job, and manage the resources across the cluster of worker nodes.

4) Executor: Each application has its own executors. Executor is a Java virtual machine process which is created on a worker for executing tasks.

5) Task: This is the smallest unit work of executor that will be sent to one executor which is launched to compute a RDD partition.

Spark streaming library [23], [24] allows consuming live data; it divides the stream in mini batch into time periods equal to batch interval. After every batch it produces a DStream (see Fig. 3), which is a sequence of Resilient Distributed Dataset (RDD). From there, live data can be processed by spark library in the same way like a batch processing.

D. Apache Flume

Apache Flume [25] is a distributed service designed to ingest the streaming data into Hadoop storage system. The data loading process is triggered by an event using an event driven pipeline architecture based on the principle of the data flow. The Event flows from Source to Channel to Sink (see Fig. 4), orchestrated by a JVM process called "flume agent" responsible to manage the following components:

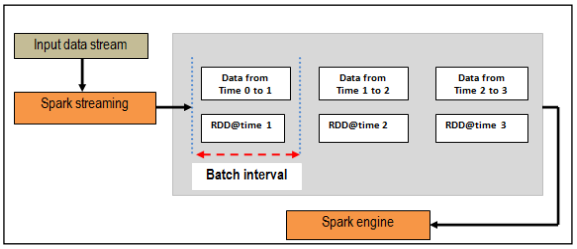


Fig. 3. Discretized stream abstraction.

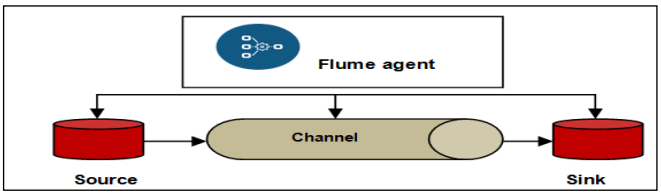


Fig. 4. Dataflow architecture in Apache Flume.

1) Sources: It allows connecting to the data sources and collecting the events. There are different types of sources, below we cite the main used in real world applications:

- SpoolingDirectorySource: Retrieves the contents of log files that arrive in a directory.
- ExecSource: Executes a bash command. The most used one is tail command which, when it is executed, retrieves the last line from a log file.
- SyslogSource: Redirects the logs data from a syslog server to Flume.
- AvroSource: Allows setting the flume agent to listen on a TCP port and pump logs in Avro format.

2) Channel: It ensures the storage of collected data and the fault tolerance in case of failure of the flume agent. Channel keeps the event until a sink consumes it. Flume provides three types of storage:

- FileChannel: Persists data on a file system.
- MemoryChannel: Stores data in memory for better performance.
- JDBCChannel: Uses a JDBC as persistence solution.

3) Sink: Removes and consumes the event from the channel, and moves it to the external destination. Below, we cite two types of sinks:

- AvroSink: Redirects data in Avro format to a distant TCP port.
- HDFSSink: Delivers and writes the events to a local file system.

V. Experiment

A. Description

We have conducted this experiment since October 2016. We started by setting up our system around the Moodle E-Learning platform which is used in HASSAN II University. In collaboration with a team of teachers, we have submitted and published the following courses:

- Object Oriented Programming with java.
- Programming in C ++.
- Programming in Android.
- Software Analysis with UML.

We have authorized access to the courses only to three groups of students. We gave them the following services:

- Consult the course content and download as PDF.
- View the videos.
- Click on web links to references in relation to content.
- Take tests.

Our goal is to build a real time data pipeline system around the existent data source; this system ensures the following tasks:

- Data Integration in two modes offline and online.
- In- memory Data processing.
- Storage of data aggregation and result in a distributed data warehouse in real time. This data warehouse is flexible in order to interact and respond to queries performed by different client applications such the reporting and analysis tools.

B. Data

Moodle stores its data in the relational database natively Mysql. We used Workbench to visualize the schema of the database which contains about 250 tables. The tables we are interested in are those which contain data profile about students and courses such as mdl_course, mdl_user, and those which contain information about interactions with the platform and more specifically with courses such as mdl_log_store_standard, mdl_lastaccess, mdl_quizz_attempts.

C. Data Integration Methodology

To capture data changes on the Moodle database in order to integrate them in our system processing layer, different approaches are possible depending on the data changes' nature (INSERT or UPDATE) carried out during students' activities on the Moodle platform.

The tables required in our context are divided into two categories:

1) Tables whose content does not change during the web activity, such as mdl_course, mdl_user, mdl_groups, and mdl_role_assignements. These tables are used as the reference where we can retrieve the profile information about users and courses, such as username, course name, etc.

We perform the batch replication of these tables to Hive data warehouse by using Apache Sqoop [26] which is an efficient tool designed to transfer bulk data between a relational databases and HDFS. Sqoop allows to extract the content of table using SQL queries, import the updates made in a database and export the result to Hive data warehouse. Several solutions in big data management and analytics use Sqoop as main part of data ingestion.

Code Example :
-Create database in Hive called moodle-experiment from hive terminal:
Hive> create database moodle_experiment
-Bulk transfer data from mdl_user to our data warehouse

moodle_experiment

$ sqoop import --connect jdbc:mysql://URL/moodle_db
--username root -P
--table mdl_user
--hive-import
--hive-table moodle-experiment.users -m 1

2) Tables whose content changes during the web activity, such as mdl_ logstore_standard_log , mdl_user_lastaccess.

The mdl_log_store_standard table is the table where Moodle inserts rows in an incremental way. The other tables undergo changes in their columns like mdl_user_lastaccess, to capture data from these tables; we created Mysql triggers on these tables to capture the transactions occurring on them, and incrementally populate new tables we created in Moodle database.

Intercept recent data from different tables via multiple Flume agents generate a lot of streams. To organize the data traffic in subjects and manageable categories, we need a middleware or a central hub capable to interact with Spark and enable real-time data processing. For this, we use Apache Kafka [27] as a pivot point in our system to receive records from Flume and push them into Apache Spark.

Kafka is a distributed persistent subscribe messaging system initially developed at LinkedIn. Kafka stores streams of events in categories called topics. A topic is a logical collection that will receive data from Flume in our context. Kafka uses Zookeeper [28] to manage its components and check the operations status.

We created manually different topics in Kafka cluster using the script Kafka-topics.sh which is a part of Kafka bin files.

Example:

- Creation of a topic named log_action_1

kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1--partitions 1 --topic log_action_1

We use Apache Flume to intercept the latest lines in the tables, in order to interact with both Moodle database and Kafka cluster, by adding to the flume library the following jar files: Flume-ng-sql [29], mysql connector, kafka_2.11-0.10.0.0.

The created topic log_action_1 receives fresh records from Flume via a customized flume-agent configuration file where we set the parameters of source, channel, sink and topic (=log_action_1).

Example:

#flume-agent configuration file
#channel & source
agent.channels = ch1
agent.sinks = kafkaSink

agent.sources = sql-source
agent.channels.ch1.type = memory
agent.channels.ch1.capacity = 1000000
agent.sources.sql-source.channels = ch1

agent.sources.sql-source.type = org.keedio.flume.source.SQLSource

 # database
agent.sources.sql-source.connection.url =jdbc:mysql://URL/moodle_experiment
agent.sources.sql-source.user = root
agent.sources.sql-source.password = password
agent.sources.sql-source.table = moodl_experiment.mdl_logstore_standard_log
#seclect colums to intercept
agent.sources.sql-source.columns.to.select = courseid,userid,action
agent.sources.sql-source.incremental.column.name = id
agent.sources.sql-source.incremental.value = 0
agent.sources.sql-source.run.query.delay=10000
agent.sources.sql-source.status.file.path = /var/lib/flume
agent.sources.sql-source.status.file.name = sql-source.status
agent.sinks.kafkaSink.type=org.apache.flume.sink.kafka.KafkaSink
#topic
agent.sinks.kafkaSink.brokerList=master:9093
agent.sinks.kafkaSink.topic= log_action_1
agent.sinks.kafkaSink.channel=ch1
agent.sinks.kafkaSink.batchSize=10

D. Environment Experiment

We deployed a small local cluster for Hadoop and Spark on 11 nodes running Ubuntu 14.04 LTS and interconnected via one switch of 1Gb/s. The Hadoop cluster is built using Hadoop version 2.7.3. The Spark cluster is built using Spark version 2.0.0. One machine is designed as Master for both Spark and Hadoop, the others nodes are both the Hadoop slaves and Spark workers. The configuration is the same for all nodes:

- Intel(R) Core(TM) i5-3470 CPU 3.20GHz (4CPUs).
- 1Gb/s network connection.
- 300GB hard disk.
- 8GB Memory.

We built the different layers using Java version 8, Scala version 2.11.8, Flume version 1.7.0, Kafka version 2.11-0.10, Hive version1.7.4, Sqoop version 1.4.6.

As Fig. 5 shows, our real time data pipeline architecture is composed of three main layers:

- *Data capture and integration layer*: Responsible for capturing data change from Moodle and ingesting data in the processing layer using Flume, Sqoop, and Kafka.
- *Data processing layer*: Consumes and processes live data and stores the result continuously in the persistence layer.
- *Persistence layer*: Hosts the data warehouse and responds to queries from different client applications like reporting tool.

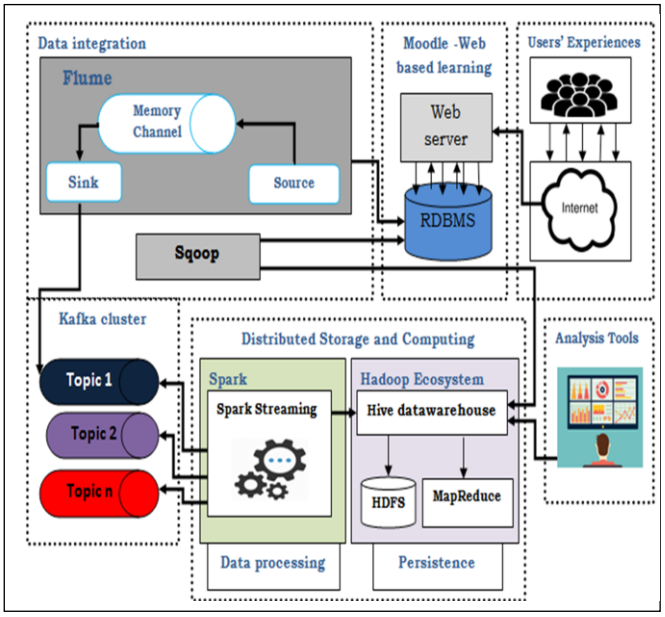


Fig. 5. Real time data pipeline architecture.

E. Event Processing

An event is the latest row added in Mysql table, intercepted by the corresponding flume agent and respectively stored in Kafka topic. We write different streaming programs in Scala in order to ensure the following tasks:

1) Capture the incoming data from Kafka topic and create the DStream via the customized receivers.

2) Extract the value from the raw RDDs in the DStream, apply transformations such cleaning, parsing in objects and finally generate the new RDDs.

3) Convert the new RDDs to data frame then create a temporary view table to store the new events in a structured format which can be queried.

4) Use Spark SQL to extract statistics from the temporary view tables and tables already stored in data warehouse like profile data.

5) Persists the result continuously in Hive data warehouse.

KafkaUtils API is used to create the input stream in order to consume data from Kafka topic by using the createDirectStream method. Each event which comes from Flume is a line text that contains the headers data and the data in interest. From DStream we extract the value and clean message with different map operations.

After cleaning the message, we obtain a new RDD in a string comma separated values, then we create a RDD of row objects by inferring schema corresponding to a data type using Scala case class that encapsulates data as objects.

To process each RDD in real time we use foreachRDD method. The following sample code explains briefly the main steps:

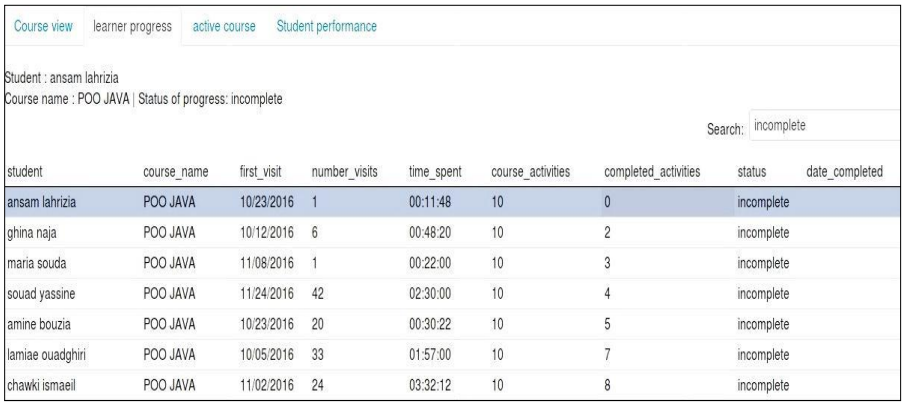
//parameters required to subscribe to a given Topic: log_action1

```
 "key.deserializer" ->classOf[StringDeserializer],
 "value.deserializer" ->classOf[StringDeserializer],
 "group.id" -> "moodle-consumer-group",
 "auto.offset.reset" -> "earliest",
 "enable.auto.commit"-> "true",
"auto.commit.interval.ms"-> "1000",
"session.timeout.ms"-> "30000"
)
val kafkaTopics = " log_action1"
val topicsSet = kafkaTopics.split(",").toSet
//receive events from a Topic in plain text format
val stream = KafkaUtils.createDirectStream[String,
String](ssc,PreferConsistent,Subscribe[String,
String](topicsSet, kafkaParams))
//Extract the value from a stream and process each RDD
with foreachRDD
   val lines = stream.map(_.value)
    lines.foreachRDD { rdd =>
if (!rdd.isEmpty) {
val sqc = new SQLContext(sc)
import sqc.implicits._

// Clean Convert RDD[String] to RDD[case class] to
DataFrame
val linesDataFrame =
lines.map(_.replace("","")).map(_.split(",")).
map(p =>logCaseExemple(p(0).toDouble, p(1).toDouble,
p(2))) .toDF()

// Creates a temporary view table using the DataFrame
    linesDataFrame.createOrReplaceTempView("view")

//Insert continuous streams into hive table
   sqc.sql("insert into table logm_hive_table select * from
view")

// select the parsed messages from table using SQL and
print it
    val linesDFquery = sqc.sql("SELECT courseid,
count(distinct userid) from view where courseid> 1 group by
courseid ") } }

linesDFquery.show()

  // Start the computation on data stream
ssc.start()
ssc.awaitTermination()
```

F. Data Visualization

Hive provides a service called hiveserver2 [30] based on Thrift RPC [31], which allows any client like Java, C++, php, and Javascript to interact with its data warehouse.

We build a web application connected to our data warehouse in order to retrieve live result and visualize a dashboard containing a set of indicators as student progress in courses, count course views, active courses, and student performance. Fig. 6 illustrates the visualization in near real time of the data extracted from the table named progress stored in data warehouse.



student	course_name	first_visit	number_visits	time_spent	course_activities	completed_activities	status	date_completed
ansam lahrizia	POO JAVA	10/23/2016	1	00:11:48	10	0	incomplete	
ghina naja	POO JAVA	10/12/2016	6	00:48:20	10	2	incomplete	
maria souda	POO JAVA	11/08/2016	1	00:22:00	10	3	incomplete	
souad yassine	POO JAVA	11/24/2016	42	02:30:00	10	4	incomplete	
amine bouzia	POO JAVA	10/23/2016	20	00:30:22	10	5	incomplete	
lamiae ouadghiri	POO JAVA	10/05/2016	33	01:57:00	10	7	incomplete	
chawki ismaeil	POO JAVA	11/02/2016	24	03:32:12	10	8	incomplete	

Fig. 6. Learners progress.

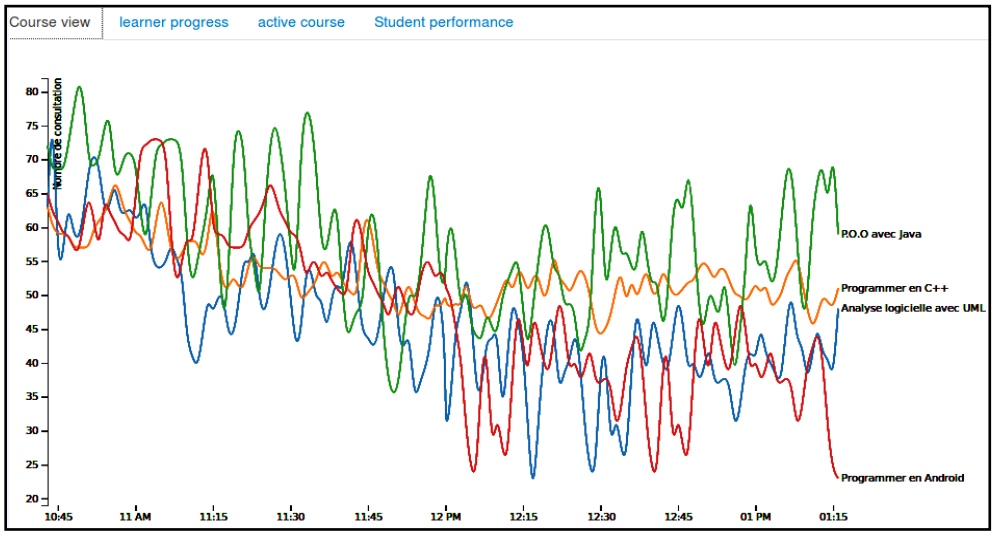


Fig. 7. Real time *course* view.

It summarizes information about students' progress in each course, such student name, course name, number of visits, the first visit, number of access, total of completed activities, status of progress in activities (completed or incomplete).

The dashboard offers the possibility to apply filters on the result. Fig. 7 illustrates the real time count view in all courses. The data is extracted from the table in data warehouse named count_vcourse.

G. Analysis of Students' Behavior using Clustering

Clustering is an unsupervised machine learning technique, used in data exploratory, knowledge discovery and is also the starting point of building a recommender system. Clustering algorithm attempts to find natural groups of similar items in data, and put these data points in the same cluster. Two standard methods are used in clustering [32] hierarchical clustering and partitioning clustering.

K-means is the best known partitioning algorithm and can be described as follows:

1) Choose random k points as initial cluster centers called centroids.

2) Assign each data point to their nearest centroid according to the Euclidean distance function.

3) Update the centroids for the clusters by calculating the mean value of the points assigned to the cluster.

4) Repeat phases 2 and 3 until the centroids do not change or the maximum number of iterations is reached.

A good K-means clustering model will split the objects in clusters by minimizing the total within-cluster variation or total within-cluster sum of square (known as WCSS) defined by the following formula:

$$WCSS(K) = \sum_{k=1}^{K} \sum_{x_1 \in C_k} (x_i - \mu_k)^2$$

Where, xi is a data point in cluster C_k and μ_k is the mean value of the points assigned to the cluster C_k.

The dataset used in this section is extracted from data warehouse and contains 179 observations. Each observation is described by 9 attributes (see Table 1) related to the students' actions in the most active course which is Object Oriented Programming with Java.

Analysis is performed in Rstudio by using a SparkR library [33] which enables large scale data analysis in Spark engine from the R environment.

We did not include the attributes student_id, student_name in data preparation so the resulting dataset consists of 7 attributes. We have normalized all numerical data with z-score standardization method, in order to avoid the dominance of some features since they vary in range.

The appropriate cluster number is found as follow:

1) Execute k-means clustering algorithm for different numbers of k from 1 to 15 by using an implementation of k-means algorithm which is included in Spark MLlib (Spark Machine Learning library).

2) Compute the total within-cluster sum of square (WCSS) for each number of cluster and plot the curve of WCSS corresponding to values of k.

3) According to the Elbow method, the curve looks like an arm (see Fig. 8), the location of the "elbow" represents the optimal number of clusters.

As the goal of this analysis is to study the clusters of the students with similar browsing behavior we give in Fig. 9 the coordinates of cluster centroids. Because values are standardized, positive values represent the values that are above the overall mean for all students in dataset, and negative values represent the values that are below the mean.

In Fig. 10 below, the values represent variable means for each cluster in the original metric.

TABLE I. ATTRIBUTES DESCRIPTION OF THE DATASET

Attribute	Description
student_id	Student identifier
studentname	Student name
totalAcess	Number of times the student has visited the course
total_sectionsvisite	Number of views on sections
total_videovisited	Total of viewed videos
total_linksvisited	Total of visited links
time_spentcourse	Time spent in sections in minutes
time_spent_test	Time spent in tests in minutes
avgscore	Average score obtained in all tests (from 0 to 10)

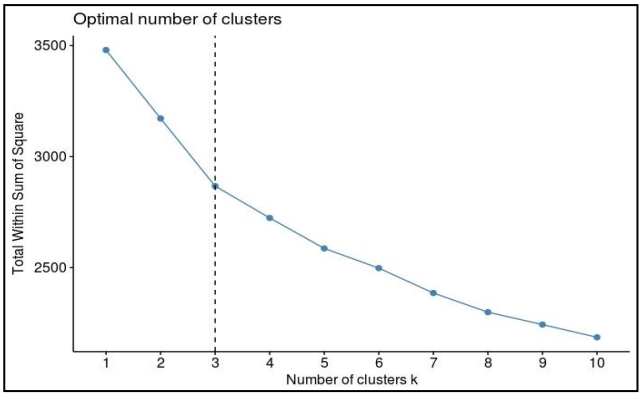


Fig. 8. Find optimal number of clusters with Elbow method.

Examining Fig. 9 and 10, we note that:

- The average of all actions of students in cluster 1 is below the global mean except the score.
- The average of the number of visits to the course, the number of views on sections and the time spent in course by the students in cluster 1 fall between those of the other clusters.
- The students of cluster 1 have consulted less videos and links, and have spent less time doing tests compared to students in other clusters.
- The cluster 2 represents the students at risk.
- The cluster 3 is the group of the average students.

```
totalAcess total_sectionsvisite total_videovisited total_linksvisited
1 -0.3392931        -0.04409049       -0.26585464        -0.4084128
2 -0.5894367         0.47547916        0.21567242         0.6305977
3  0.8436063        -0.39347589        0.04358658        -0.2057686
  time_spentcourse     time_spent_test   avgscore
1       -0.1090387         -0.20270743  1.0606550
2       -0.3204454          0.09550445 -0.5281638
3        0.3905054          0.09605830 -0.4767004
```

Fig. 9. Coordinates of the cluster centroids

```
  cluster totalAcess totalsectionsvisite total_videovisited total_linksvisited
1       1   23.23214           63.33929           25.84821           30.84821
2       2   19.95575           79.93805           31.68142           40.30973
3       3   38.72581           52.17742           29.59677           32.69355
  time_spentcourse     time_spent_test avgscore
1         180.6518           91.95536 8.750000
2         167.6549           96.45133 4.991150
3         211.3629           96.45968 5.112903
```

Fig. 10. Variable means in original metric

We can deduce that the cluster 1 is the group of students who have adopted a moderate behavior in all actions and achieved good results.

It is very early to confirm or deny that the strong presence in a platform guarantees a good result; therefore we can't generalize this result. To do this, we need more additional information, so we have to study how to feed our data warehouse with other data that are related to the students' profile, such as the academic past, personal data and other data interactions with the platform which are not available in the database of the E-Learning platform.

VI. CONCLUSION

In this paper we addressed the challenge to implement an event-driven system around a web based learning platform. This system is in the form of a real time data pipeline capable to capture data change from RDBMS database source and extract valuable information. We adopt a big data concept to design, on inexpensive hardware, a flexible and distributed architecture composed of three layers: data capture, data processing and data persistence. We combine Apache Flume and Sqoop to collect fixed and live data. Apache Kafka is

responsible for organizing the data traffic. To process data in real time we use Spark Streaming library. Apache Hive is used to build our data warehouse hosted in a distributed storage system.

During this work which is based on a real experience we have identified new directions to extend the proposed work. The first is to study and investigate new methods to combine social networks data, past academic and personal data with actual data in data warehouse, to get more information about students. The second is to develop an adaptive learning system based on machine learning models like predictive and recommender system in order to apply these models to assist students during their interactions with the E-Learning platform.

REFERENCES

[1] G. Siemens and D. Gasevic (2012). Guest Editorial –"Learning and Knowledge Analytics". Educational Technology & Society, 15 (3), 1–2.

[2] EDM: http://www.educationaldatamining.org/.

[3] G. Siemens and R.S.J.d. Baker, "Learning Analytics and Educational Data Mining: Towards Communication and Collaboration", LAK '12 Proceedings of the 2nd International Conference on Learning Analytics and Knowledge Pages 252-254.

[4] S. Khan, X. Liu, K. A. Shakil, M. Alam, A survey on scholarly data: From big data perspective, Information Processing & Management, Volume 53, Issue 4, 2017, Pages 923-944, ISSN 0306-4573, http://dx.doi.org/10.1016/j.ipm.2017.03.006.

[5] M. Ali-ud-din Khan, M. F. Uddin, N. Gupta, "Seven V's of Big Data Understanding Big Data to extract Value", Proceedings of 2014 Zone 1 Conference of the American Society for Engineering Education (ASEE Zone 1).

[6] Tom White, "Hadoop - The Definitive Guide: Storage an d Analysis at Internet Scale (2. ed.)", Publisher: O'Reilly, ISBN: 978-1-449-38973-4.

[7] P.K. Udupi, P. Malali and H. Noronha, "Big data integration for transition from e-learning to smart learning framework", 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC), Muscat, 2016, pp. 1-4. doi: 10.1109/ICBDSC.2016.7460379.

[8] B. Logica and R. Magdalena, "Using Big Data in the Academic Environment", Procedia Economics and Finance, Volume 33, 2015, Pages 277-286, ISSN 2212-5671, http://dx.doi.org/10.1016/S2212-5671(15)017128.(http://www.sciencedirect.com/science/article/pii/S2212567115017128).

[9] S.B. Aher and Lobo L.M.R.J., "Course Recommender System in E-learning", International Journal of Computer Science and Communication, Vol. 3, No. 1, January-June 2012, pp. 159-164.

[10] Y. Tabaa and A. Medouri, "LASyM: A Learning Analytics System for MOOCs", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 4, No. 5, 2013.

[11] Shyam R., Bharathi Ganesh HB, Sachin Kumar S, Prabaharan Poornachandran and Soman K P, "Apache Spark a Big Data Analytics Platform for Smart Grid", Procedia Technology 21 (2015): 171-178. DOI: 10.1016/j.protcy.2015.10.085.

[12] Han Hu, Yonggang Wen, Tat-Seng Chua and Xuelong Li, "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial", IEEE Access, Vol. 2, 2014, pp.652-687, DOI: 10.1109/ACCESS.2014.2332453.

[13] Apache Hadoop: http://hadoop.apache.org/.

[14] S. Ghemawat, H. Gobioff and Shun-Tak Leung, "The Google File System", ACM SIGOPS operating systems review, vol. 37, no. 5,p. 29-43. ACM, 2003.

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", In Proceedings of the 6th USENIX OSDI, pages 137–150,2004.

[16] Apache Hive: https://hive.apache.org/.

[17] G.P. Haryono and Y. Zhou, "Profiling apache HIVE query from run time logs", Big Data and Smart Computing (BigComp), 2016,IEEE DOI: 10.1109/BIGCOMP.2016.7425802.

[18] A. Thusoo, J.S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu and R. Murth, "Hive - a petabyte scale data warehouse using Hadoop", In Proceedings of International Conference on Data Engineering (ICDE), 2010, pp. 996- 1005.

[19] Apache Spark: http://spark.apache.org/ .

[20] A. Shkapsky , M. Yang, M. Interlandi, H. Chiu, T. Condie and C. Zaniolo, "Big Data Analytics with Datalog Queries on Spark", SIGMOD'16, June 26-July 01, 2016, San Francisco, CA, USA,2016 ACM, DOI: http://dx.doi.org/10.1145/2882903.2915229.

[21] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing", in Proc. 9th USENIX Conf. Netw. Syst. Des. Implement., 2012, p. 2.

[22] M. Zaharia, R.S. Xin, P. Wendell, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, I. Stoica, "Apache Spark: a unified engine for big data processing", Magazine Communications of the ACM, Vol. 59 No. 11, Pages 56-65.

[23] Spark streaming: http://spark.apache.org/streaming/.

[24] W. Wingerath, F. Gessert, S. Friedrich, and Norbert Ritter, "Real-time stream processing for Big Data", it – Information Technology 2016; 58(4): 186–194,DOI 10.1515/itit-2016-0002.

[25] Deepak Vohra, "Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools", Publisher: Apress, ISBN: 978-1-4842-2199-0, pp.287-300.

[26] Apache Sqoop: http://sqoop.apache.org/.

[27] Apache Kafka: https://kafka.apache.org/.

[28] Apache Zookeeper: https://zookeeper.apache.org/.

[29] Flume-ng-sql: https://mvnrepository.com/artifact/org.keedio.flume.flume-ng-sources/flume-ng-sql-source/1.3.7.

[30] Hiveserver2: https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients.

[31] Apache Thrift: http://thrift.apache.org/docs/concepts.

[32] Tuffery Stéphane, "Data Mining et statistique décisionnelle L'intelligence des données- 4 ème édition", Publisher: Technip, ISBN: 9782710810179, pp. 265-317.

[33] S. Venkataraman, Z. Yang, D. Liu, E. Liang, H. Falaki, X. Meng, R. Xin, A. Ghodsi, M. Franklin, I. Stoica, M. Zaharia, "SparkR: Scaling R Programs with Spark", SIGMOD'16, June 26–July 1, 2016, San Francisco, CA, USA. ACM, 2016.