RESEARCH

CrossMark

Large-scale e-learning recommender system based on Spark and Hadoop

Karim Dahdouh[1*] , Ahmed Dakkak[1], Lahcen Oughdir[1] and Abdelali Ibriz[2]

*Correspondence:
karim.dahdoh@gmail.com
[1] Engineering Sciences
Laboratory, FPT, Sidi
Mohamed Ben Abdellah
University, Taza, Morocco
Full list of author information
is available at the end of the
article

Abstract

The present work is a part of the ESTenLigne project which is the result of several years of experience for developing e-learning in Sidi Mohamed Ben Abdellah University through the implementation of open, online and adaptive learning environment. However, this platform faces many challenges, such as the increasing amount of data, the diversity of pedagogical resources and a large number of learners that makes harder to find what the learners are really looking for. Furthermore, most of the students in this platform are new graduates who have just come to integrate higher education and who need a system to help them to take the relevant courses that take into account the requirements and needs of each learner. In this article, we develop a distributed courses recommender system for the e-learning platform. It aims to discover relationships between student's activities using association rules method in order to help the student to choose the most appropriate learning materials. We also focus on the analysis of past historical data of the courses enrollments or log data. The article discusses particularly the frequent itemsets concept to determine the interesting rules in the transaction database. Then, we use the extracted rules to find the catalog of more suitable courses according to the learner's behaviors and preferences. Next, we deploy our recommender system using big data technologies and techniques. Especially, we implement parallel FP-growth algorithm provided by Spark Framework and Hadoop ecosystem. The experimental results show the effectiveness and scalability of the proposed system. Finally, we evaluate the performance of Spark MLlib library compared to traditional machine learning tools including Weka and R.

Keywords: Big data, Spark, Hadoop, E-learning, Online learning, Course recommender system, MLlib methods, Association rules, Parallel FP-growth algorithm

Introduction

The computing environment for human learning is changing rapidly, due to the emergence of new information and communication technology such as big data [1] and cloud computing [2]. Furthermore, the learning methods are changing every day. Therefore, e-learning systems need to develop more techniques and tools to meet the increased needs of millions of learners around the world.

This article exposes a smart recommender system applied in an online learning environment in order to be able to provide personalized courses and guide students to select more suitable courses. For example, emailing or sending notifications through the user interface of distance learning platform, to students who follow courses in a specific field

and recommend the suitable educational resources that are likely to be interesting for them. Also, learners can be guided to enroll in the latest courses in their interest areas based on historical data of all users over a large dataset of courses enrollments. In this article, we are interested in improving learning platforms through a recommender system. Our system uses association rules for extracting more interesting relationships between learners' behaviors. Indeed, it aims to find similarities between courses enrollments in the transaction database. Thus, discovering association rules enables us to target students who learn two or more courses together, i.e. finding a list of frequent courses enrollments to determine those that are more likely chosen by the learners. So, based on the discovered patterns, we can guide students to take specific courses. The pedagogical team can also improve the quality of non-frequent courses or create new ones.

The rest of the article is organized as follows: In "Background" section, we present a state of the art of recommender systems for e-learning environments. In "Method" section, we introduce the basic concepts of the association rules technique. Then we give a detailed description of the FP-growth algorithm. It also presents briefly the set of technologies employed in this work including spark and Hadoop. In "Conclusion" section, we implement the course recommender system and illustrate the experiments results of the historical data analysis. Then, we evaluate the performance of our solution compared to other machine learning tools including Weka and R. Finally, the article shows some monitoring tools for tracking the execution of the spark jobs.

Background

Related work

Many research works have conducted in the field of distance learning in higher education using big data techniques and including data mining and machine learning methods. There are a lot of applications of these techniques. Particularly, recommendation engine which is used in several areas such as basket analysis (Amazon), social networks (LinkedIn, Twitter), government, education, etc. In this section we provide a brief overview of the previous researches regarding the development recommender based on machine learning models.

Mihai et al. [3] proposed the prototype of a recommender system based on association rules for the distributed learning management system. The article uses distributed data mining algorithms and data obtained from Learning Management Systems (LMS) database in order to identify strong correlations between sets of courses followed by students. It also gives a brief description of the architecture and methodology of the course recommender system without providing an implementation of the proposed architecture.

Jooa et al. [4] focused on implementing a recommender system using association rules and collaborative filtering. The proposed system uses the distance data from the Global Positioning System (GPS) to recommend products that customers are likely to purchase based on their preferences.

Perušić Hrženjak et al. [5] applied association rules technique in learning management system of the Rijeka University. They use students from MudRi e-learning database, which is based on the Moodle open source software. Then, they apply the FP-growth

algorithm for finding connections between various actions. They find that students have better success in the course when they are using videos course. Also, they identify which lessons seem to have a greater connection to the final grades.

In another related work, Panigrahi et al. [6] proposed a hybrid solution to implement recommender a system using collaborative filtering and clustering techniques like K-means. It is based on in-memory computation of Apache Spark as big data platform allowed speed up the running time to make recommendation. Next, this work aims to alleviate the cold start problem of traditional Collaborative Filtering by correlating the users to products through features.

Li et al. [7] proposed an algorithm to parallelize the frequent itemset discovery bi finding hidden pattern to support query recommendation of large dataset. This algorithm enables to reduce computation cost by spreading calculation between cluster nodes in such a way that each node executes an independent set of mining tasks. They achieved best performance through distributing the processing using MapReduce infrastructure over cluster of computers.

Zhou et al. [8] implemented an alternating least squares (ALS) algorithm by utilizing the collaborative filtering approach for the Netflix Prize. ALS is a simple parallel algorithm which aims to tackle the scalability issue with very large datasets. It used for building a large-scale movies recommender system for predicting user ratings. The results achieved show a performance improvement of 5.91%.

Jiahui et al. [9] developed a large-scale recommender system in order to offer personalized news based on past click behavior of users. Their work combines both content-based recommendation and collaborative filtering methods to suggest most relevant news articles. The experiment results showed that the hybrid approach enhance the quality of news recommendations by attracting more readers to visit Google website news.

Many recommender systems for social networks have been developed using the user interactions and behaviors. An approach is suggested in [10] to analyze the users' interest of twitter platform. They combined sentiment analysis and classification of tweets by analyzing the topics discussed by the users. The implementation of their work gives encouraging results.

To incorporate the parallel processing and advanced analysis of machine learning techniques in education field, especially online learning environment, we propose a smart courses recommender system using association rules method and the latest big data technologies such as Spark and Hadoop ecosystem. Moreover, the system presented in this article is deployed in a distributed computing environment. It runs on a cluster of nodes which reduces the time spent on extracting the recommendations results. This approach is efficient especially when the size of the data size is very large.

Methods

Association rules [11] is an unsupervised learning method that is widely used in many fields including recommendation engines, retail analysis of the transaction, and clickstream analysis across web pages [12]. It aims to discover hidden patterns in large amounts of data, in the form of interesting rules.

The term Association rules is often referred to as Market Basket Analysis application. Because the first time used was in 1993 by Agrawal et al. [13] in order to find useful relationships between items through a large database of customer transactions. Each transaction consists of items purchased by a customer. In order to discover all significant connections between items bought by a customer over a period of time not necessarily consist of items bought together at the same time. In general, the commendation systems consist of three principle steps; first, collect data from large transaction database; second, find similarities between users behavior's, according to more frequent item set, and finally, recommend more suitable items for users.

Considering $C = \{c_1, c_2, c_3, \ldots, c_n\}$ a set of all items or courses enrollments stored in database and $L = \{l_1, l_2, l_3, \ldots, l_n\}$ a set of learner profiles. Each learner l_i enrolls into k courses, where k is a subset of courses chosen from set of items C. In association rules, we define a rule as an implication of $X \Rightarrow Y$, where $X, Y \subseteq C$ (X and Y are sets of courses) and $X \cap Y = \emptyset$, which means when course X is followed by the learner l_i, course Y is likely followed as well with a high probability. The set of attributes X is called antecedent or left-hand-side (LHS) of the rule; the set of attributes Y is called consequent or right-hand-side (RHS) of the rule [13].

Generally, The association rules technique produce a large number of rules $X \Rightarrow Y$, but to select interesting rules from the set of all generated rules, there are two important measures to determine the quality of an association rule, the most known are minimum thresholds of support and confidence. The support is the percentage (%) of transactions in the dataset that contain the itemset X while confidence is defined as the percentage (%) of transactions that contain X, which also contain Y. The formal definition of the confidence is: $\text{conf}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}$. Therefore, a strength association rule $X \Rightarrow Y$ should satisfy: $\text{supp}(X \cup Y) \geq \sigma$ and $\text{conf}(X \Rightarrow Y) \geq \delta$, where σ and δ are the minimum support and minimum confidence, respectively.

In the context of our research, we apply association rules technique in the online learning. Accordingly, a transaction in our case is represented by the student's profile. Similarly, items are replaced by courses followed by a given student during the learning process. So, we can define the support and confidence respectively as follows:

$$\text{supp}(X \Rightarrow Y) = \frac{\text{number of learners following X and Y courses}}{\text{total number of learner enrollments in database}} \tag{1}$$

$$\text{conf}(X \Rightarrow Y) = \frac{\text{number of learners following X and Y courses}}{\text{number of learners profiles following X courses}} \tag{2}$$
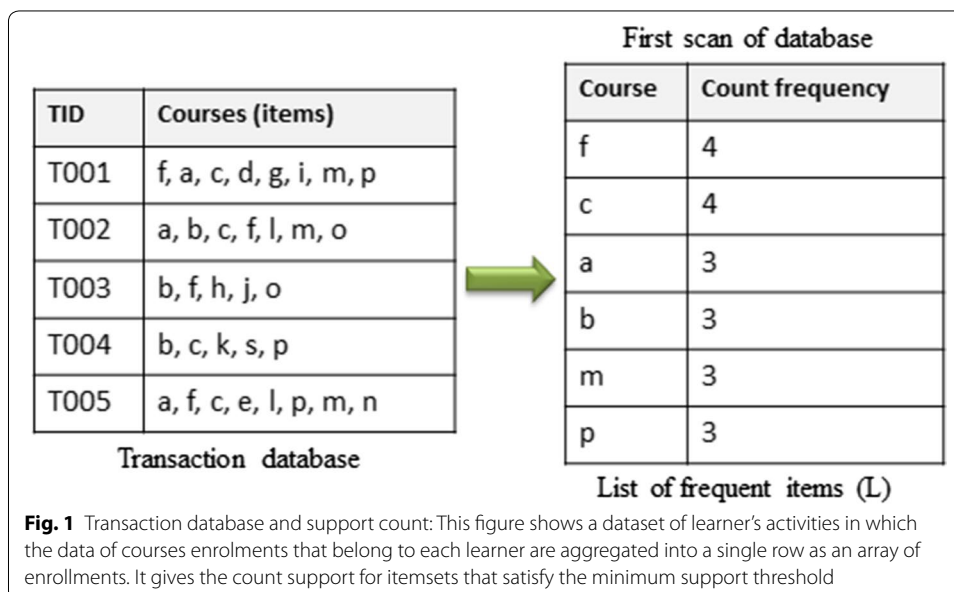
Support is the first criteria which is defined as the percentage of transactions that contain X, It means, support is an indication of how frequently the itemset appears in the database. On the other hand, confidence is defined as the percentage (%) of transactions (students profiles) that follow X, which also follow Y.
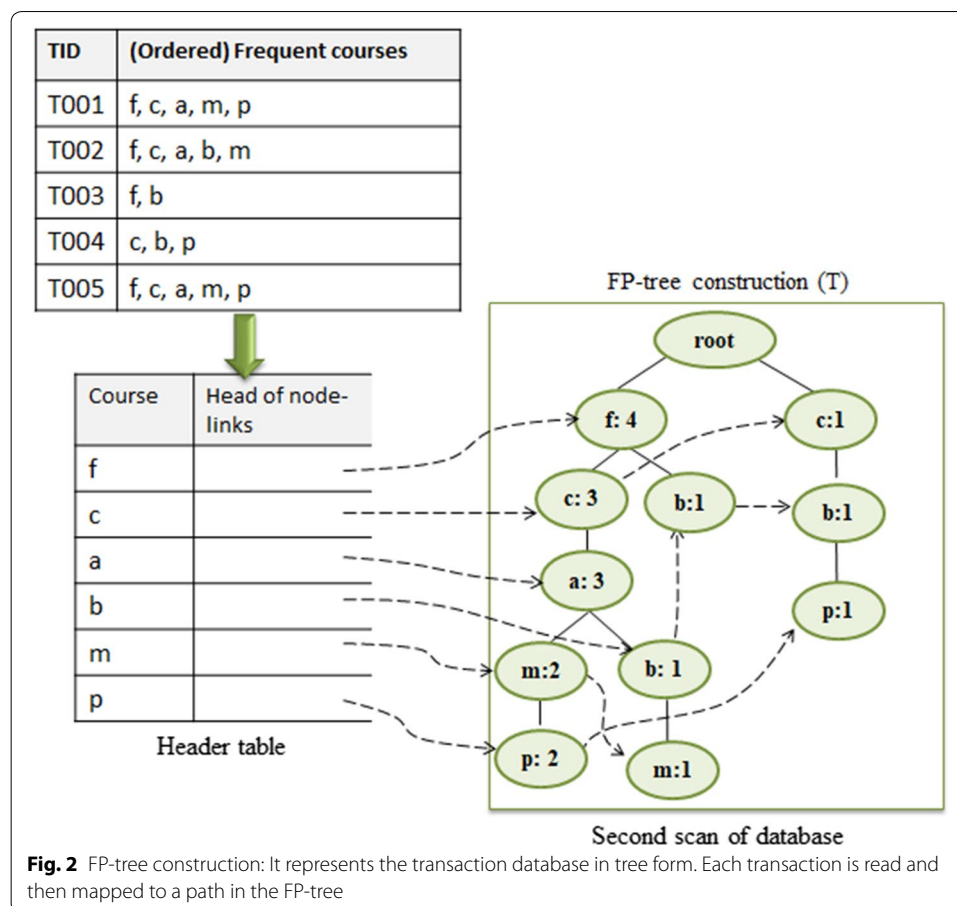
Traditional FP-growth algorithm

There are several algorithms for implementing association rules method to determine the most interesting relationships between variables or items in a large database, through finding frequent itemsets. FP-growth (frequent pattern growth) [14] is an efficient and

scalable algorithm for extract items that more likely appear together in a large trans-action database. We find other algorithms such as Apriori [15], MAFIA (Maximal Frequent Itemset Algorithm) [16] and Eclat [17]. But FP-growth is the fastest one because it allows frequent itemset discovery without candidate item set generation which is more expensive in both memory and time. Moreover, candidate generation and test require multiple database scans. In fact, by using FP-growth the number of database scan is reduced to two. The first scan counts the support of each item; the infrequent items are deleted while the frequent items are sorted in decreasing support counts as a list of frequent items (L) as shown in Fig. 1. And in the second scan, FP-growth constructs FP-tree. Those operations form the first step of the algorithm. On the other hand, the second step aims to extract frequent itemsets from the constructed FP-tree.

Let's consider the following transaction database which contains 5 transactions and 16 items or courses. Suppose that minimum support is 3. In the beginning, FP-growth algorithm scans the transaction dataset for the first time to count the support of each item and find frequent items as list L, in which items sorted according to the support descending order. Then, the frequent items of all transactions are reorder based on list (L) order. Next, FP-growth scans the database for the second time and constructs FP-tree. After creating the root ("null") of FP-tree (T). It reads the first transaction then build the first branch {(f:1), (c:1), (a:1), (m:1), (p:1)} of the tree (the number after item indicates the support). The second transaction T002 shares a common prefix (f, c, a) with the path of the first transaction (T001). So FP-growth algorithm increments by 1 the count of each node belong to the prefix, i.e. the accounts of first path become {(f:2), (c:2), (a:2), (m:1), (p:1)}. And a new node (b:1) is inserted in the tree and linked as a child of node (a:2); also another new node (m:1) is created that has as a parent the node (b:1). This operation is recursively repeated for each transaction in the dataset until scanning and mapping all the transactions to a path in the FP-tree, as is illustrated in Fig. 2. In general, the insertion of a new transaction in FP-tree performed as follows. If T (FP-tree)

TID	Courses (items)
T001	f, a, c, d, g, i, m, p
T002	a, b, c, f, l, m, o
T003	b, f, h, j, o
T004	b, c, k, s, p
T005	a, f, c, e, l, p, m, n

Transaction database

First scan of database

Course	Count frequency
f	4
c	4
a	3
b	3
m	3
p	3

List of frequent items (L)

Fig. 1 Transaction database and support count: This figure shows a dataset of learner's activities in which the data of courses enrolments that belong to each learner are aggregated into a single row as an array of enrollments. It gives the count support for itemsets that satisfy the minimum support threshold

Fig. 2 FP-tree construction: It represents the transaction database in tree form. Each transaction is read and then mapped to a path in the FP-tree

has a child N such that item name of N has the same as the one of the item name of the scanning transaction, then increment N's count by 1; else create a new node M, and initialize its count to 1, its parent link be linked to T, and its node-link be linked to the nodes with the same item name [14].

The second step of the FP-growth algorithm consists in mining frequent patterns using FP-tree. In brief, this step including the construction of conditional pattern base for each item in the header table, the construction of the conditional FP-tree from each conditional pattern base, and recursively mine conditional FP-trees and grow frequent patterns. The mechanism of mining frequent pattern is described in details in the paper of Han et al. [14].

Parallel FP-growth

There are several implementations of the FP-growth algorithm, the best one is that which implements a parallel version of this algorithm called parallel FP-growth (PFP). PFP is based on a novel computation distribution scheme. Indeed, it distributes the job across a cluster of nodes using the MapReduce model. So, it is more scalable and fast than its traditional implementation based on single-machine.

The library Spark MLlib (spark.ml) [18] provides an in-memory implementation of PFP algorithm that facilitates the use of Association rules techniques in a distributed

computing environment. The PFP provided by Spark available in 4 programming language, including Scala, Java, Phyton. In addition to default settings of conventional FP-growth (minimum support and minimum confidence thresholds), spark.ml package also takes the numPartitions parameter which specifies the number of partitions used to split the job. The advantage of this algorithm is that it gives better performance in terms of execution time and scalability.

Apache Hadoop ecosystem

Apache Hadoop [19] is an open source platform that supports big data storage and processing. It is a solution inspired by Google's article "MapReduce: Simplified Data Processing on Large Clusters" [20] as a model of distributed processing on large clusters. It is developed as a result of the limitations of traditional approaches to meeting the challenges of analyzing large volumes of data produced by companies. It is designed to be deployed on commodity hardware. Also, it supports three different modes including standalone, pseudo-distributed and fully-distributed mode. Actually, Hadoop ecosystem contains several related-projects to deal with many problems ranging from anlysis (Spark), querying (Pig), loading (Apache Flume and Sqoop), and distributed real-time computation (Storm). It consists of two main components including Hadoop Distributed File System (HDFS), which manage data, and MapReduce used for processing large amounts of data, collected from various sources, in a distributed way across a cluster of machines [1]. Hadoop technology is already used by different companies and cloud providers such as Yahoo, Facebook, Amazon Web Sevices, and Microsoft Azure.

HDFS

The Hadoop distributed file system (HDFS) [21] is a file system for data management across large clusters with a master/worker architecture. Its development is inspired by the GFS (Google File System) file system. HDFS is highly fault-tolerant and is designed to be deployed on basic hardware. HDFS provides high-speed access to data and it is suitable for applications that have large datasets. It creates an abstraction of disk resources to allow the management of the distributed physical storage of several nodes as if there is a single hard disk. In HDFS architecture, the data are managed across the cluster in different Datanodes in the form of files structured in blocks. The locations of these blocs and namespace of files and directories are kept in Namenode.
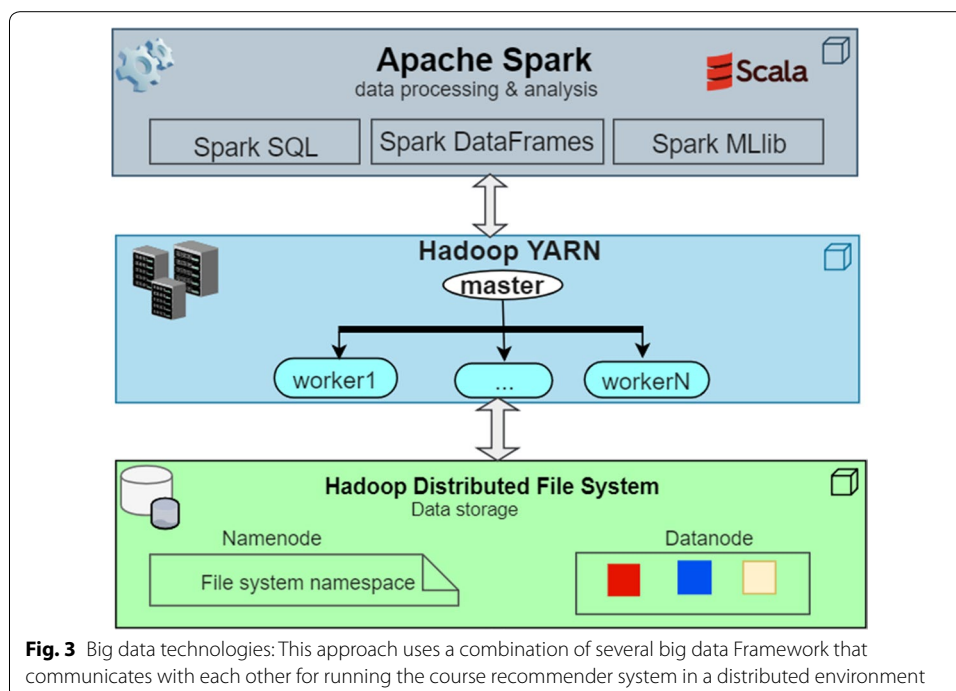
Yarn

Yarn [22], yet another resource negotiator, is the successor of MapReduce (version 2). It is a framework dedicated to Job scheduling and cluster resource management. Yarn's main idea is to separate resource management from the computation model. Such separation allows its architecture to support a number of distributed processing frameworks such as Spark [23], Giraph [24], Storm [25]. It is consist of a master machine named ResourceManager (RM) and a set of workers named NodeManagers (NM), which forms a generic system for managing applications in a distributed manner. The Resource Manager is a scheduler responsible for tracking, sharing available resources between applications and optimizing cluster utilization. On the other hand, the NodeManager

is responsible for executing tasks and monitoring the use of resources by each worker node.

Spark

Apache Spark [23] is a unified analysis framework for distributed big data processing. It was originally developed by UC Berkeley University in 2009. The power of Spark is in its ability to perform calculations in-memory which allows it to be faster 100 times than MapReduce. Spark can be deployed in standalone mode, Hadoop Yarn, Mesos or Kubernetes. It supports several data management systems such as HDFS, HBase, Hive, Cassandra, Amazon S3 and several DBMS (Oracle, Mysql). Spark APIs provides a rich collection of libraries to support a variety of advanced analysis use cases. Among its components, there are Spark SQL (structured data processing), Streaming (real-time processing), MLlib, GraphX, SparkR. In addition, it supports several programming languages, including Scala, Java, Phyton, R.

In brief, Spark job consists of a set of transformations. These transformations build up a Directed Acyclic Graph (DAG) of operations. During the execution of a work submitted by a client, spark runs a graph of instruction as a single job by breaking it down into stages. Each stage contains a set of parallel tasks to be executed across the cluster. We can summarize the big data technologies used to build our recommender system in the following diagram (Fig. 3). Actually, there are 3 layers: First, Storage and replication layer represented by HDFS. In the second level, we find Yarn as cluster resource manager of nodes. And, the top layer is Spark, responsible for data processing and analyzing. This work use Spark SQL (JDBC) and MLlib (parallel FP-growth) libraries of Spark Framework.

Fig. 3 Big data technologies: This approach uses a combination of several big data Framework that communicates with each other for running the course recommender system in a distributed environment

Dahdouh *et al. J Big Data*   (2019) 6:2

Page 9 of 23

Experiments

ESTenLigne project

The present work is a part of the ESTenLigne [26] project, which is the result of several years of experience for the development of e-learning in the Sidi Mohamed Ben Abdellah University of Fez. It was started since 2012 by the EST network of Morocco, which aims the development of distance education based on new information and communication technologies through the implementation of open, adapted and free online learning platform, and taking into account the dimensions of exchange, sharing and mutualization of pedagogical resources [27, 28]. Several works have been done as part of this project including the training of experts across e-learning in the context of the Coselearn I project, and teacher training through Franco-Moroccan EST [29] and IUT [30] cooperation [28, 31]. Furthermore, there are some researches that have been done around this project such as the analysis of the use of educational resources where the objective was to analyze the use of pedagogical resources in some courses namely the algorithmic course [32]. Also, a case study for collaboration analysis of online course based on activity theory [33].

In fact, the students have a lot of difficulties and are lost in the diversity of educational resources, particularly the large number of available courses. This requires the adaptation of the teaching to meet the needs of students. To solve these problems, we develop a course recommender system to promote learning to learners through creating a smart solution. It is able to generate the most appropriate courses automatically based on historical data of learner's activities.

Experiment dataset

For the experiment purposes, we have used an operational database of ESTenLigne platform that is built on LMS Moodle [34]. Indeed, it is an open source learning management system. It uses a relational database which has around 250 tables. We focus only on student's enrollments into courses. Especially, we focus on four tables that represent the information we need to implement our recommender system. The class diagram depicted in Fig. 4 shows the structure, the attributes, and the relationships between the four classes used in the experiments.

The class diagram, as seen in Fig. 4, describes four important tables for collecting data about historical data of learner's enrollments. First, there is mdl_user that gives information about the student's profile. Second, all user enrollments have recorded into mdl_user_enrollments table. Third, mdl_enrol table contains data of courses enrollments. For the same course, there can be different enrollment start and end dates. Some students may require a course for one period of time but other users may want to enroll in the same course for a different period of time. Fourth, mdl_course table stores all the details of the courses that are uploaded to the learning management system. It stores the names of the course, category, full name, short name, summary, time created, time modified etc. [35]. This database contains 1218 learners study in different fields (computer science, electrical, chemistry, mechanics, economics, energy, etc.) at High School of Technology of Fez. Furthermore, it also provides 153 courses proposed by teachers from various areas of education.
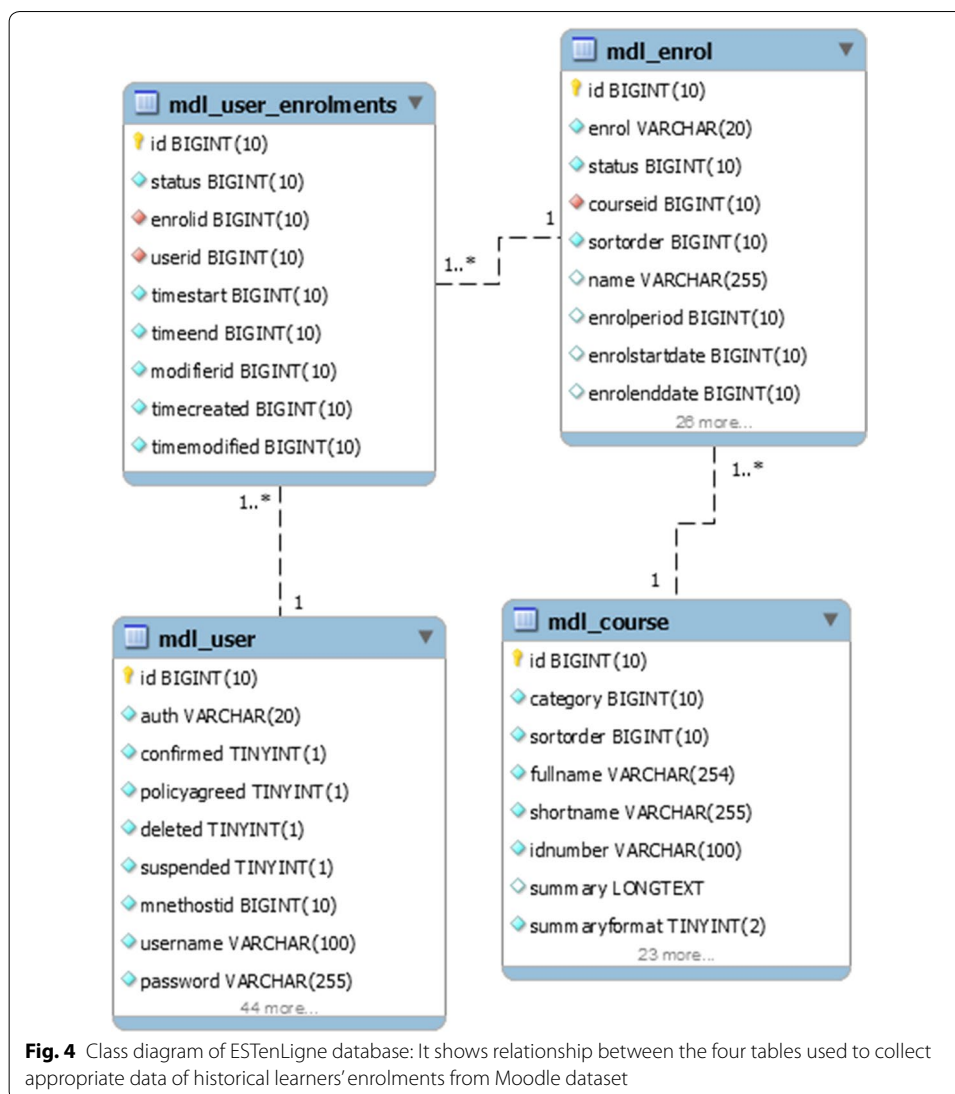
Fig. 4 Class diagram of ESTenLigne database: It shows relationship between the four tables used to collect appropriate data of historical learners' enrolments from Moodle dataset

System architecture

In general, our approach consists in finding hidden patterns from historical learners' activities. So, the input of the system is courses enrolments of the students. Actually, the commendation process consists of the following steps:
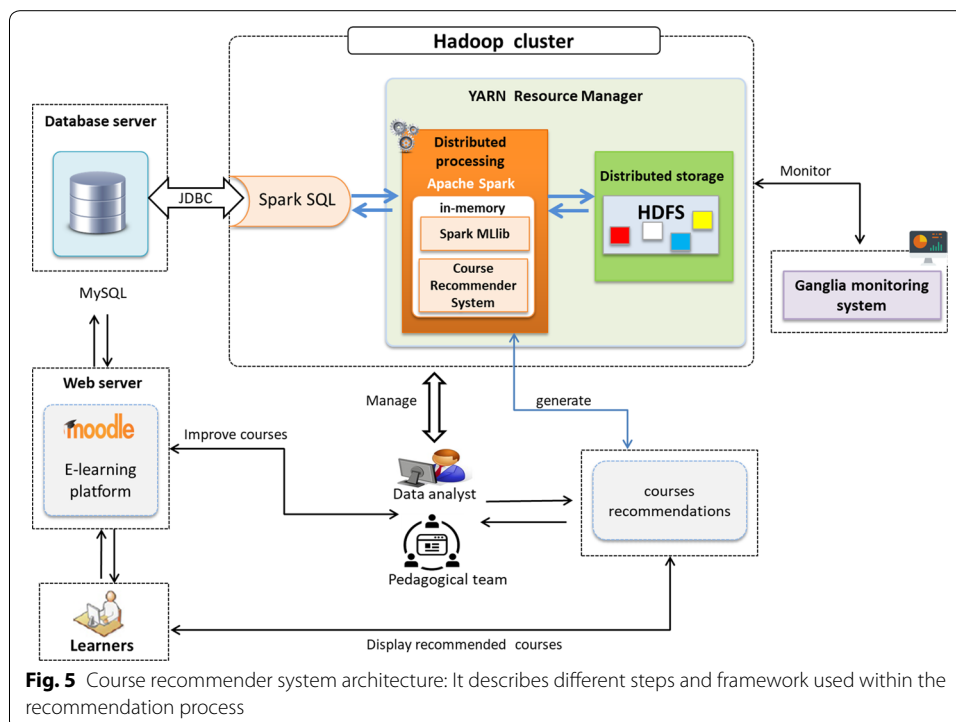
In the beginning, we have to load the data produced by the ESTenLigne platform. Then the data loaded by Spark SQL library are processed in a distributed way using the Spark framework which is executed on Hadoop cluster and managed by Yarn resource manager. In fact, we process data using the FP-growth algorithm by employing Spark MLlib which provides a large-scale implementation of association rules techniques. Next, Spark as much as a framework for distributed computing will connect with Hadoop HDFS for storing the data on clusters of machines. Finally, the output is the catalog of recommended courses that match the learner interests. Then, we can display the result of the recommendation engine to the user in order to guide and
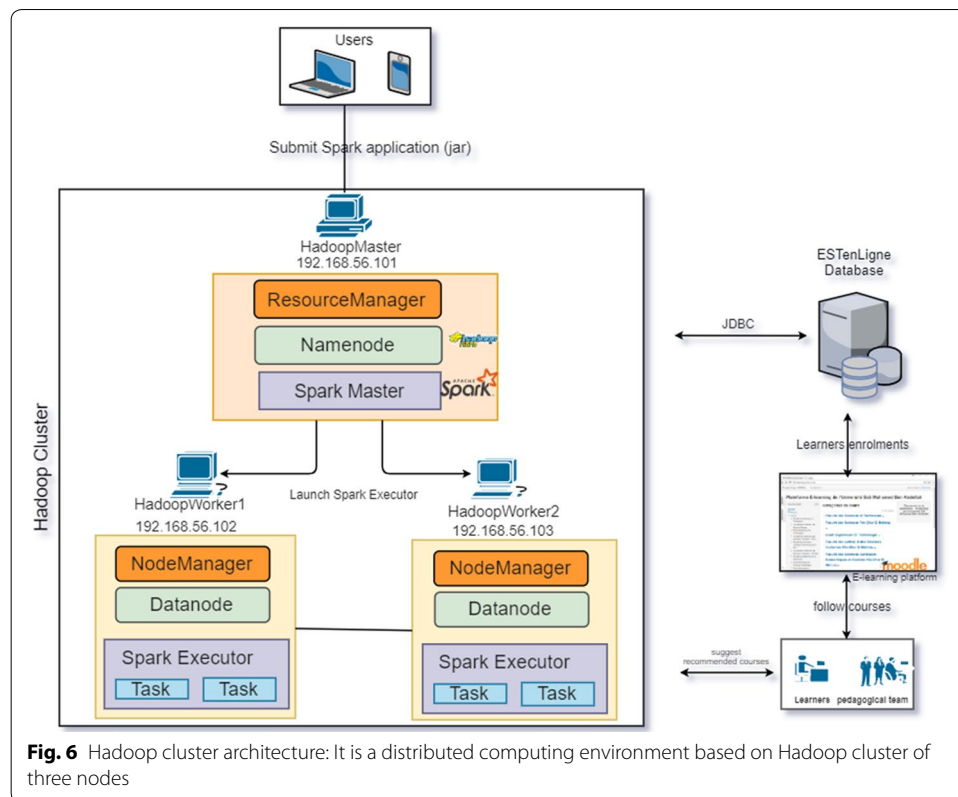
suggest them relevant pedagogical resources, after that the learner can browse these courses and start learning new and suitable courses, as shown Fig. 5.

On the technological side, our architecture uses a combination of several technologies of big data ecosystem that communicates with each other to load, aggregate, process, and store historical data of learner's courses activities. Actually, it is an architecture that implements 3 mainly frameworks, include:

1. Apache Spark: as a unified engine for the distributed computing of data collected from e-learning (ESTenLigne) dataset. To do this, it uses JDBC interfaces to connect to e-learning database. Additionally, Spark provides a particular library dedicated to Machine Learning techniques, called MLlib. Indeed, this library gives an implementation in Scala language of parallel FP-growth algorithm which is used in this article.
2. Hadoop Yarn: cluster resources manager enabling job scheduling between the nodes of a cluster.
3. Hadoop HDFS: The default distributed file system of Hadoop software.

The architecture shown in Fig. 6 describes the mechanism of functioning of the proposed system which operates on a cluster of machines manager by Yarn framework. In general, Spark is used as a framework for the distributed computing to develop the courses recommender application. It consists of a number of elements that work together to perform the job, submitted by a client (jar). In fact, Spark application runs as a set of independents processes on a cluster of machines, coordinated by the SparkContext object.

Fig. 5 Course recommender system architecture: It describes different steps and framework used within the recommendation process

Fig. 6 Hadoop cluster architecture: It is a distributed computing environment based on Hadoop cluster of three nodes

Furthermore, Apache Spark applications consist of two essential components which are driver process, Spark Master and executor processes. The driver process maintains the relevant information about SparkSession during the lifetime of the Spark application; it runs also the user main() function, declares transformations and actions, and submit the job the Spark Master. The second element in this architecture is Spark Master who creates the Tasks and distributes it to the worker's nodes to be actually executed (parallel operations). Spark Master also coordinates between the job Stages across the executors on the cluster. On the other hand, executors processes are responsible for carrying out the work assigned to them by the master and returning the state and the results of data processing to the master. When the driver process needs resources to execute a job, it requests master to allocate the required resources from the workers. Next, each worker node creates executors to perform the tasks. Then driver process runs the job on these executors.

The distributed computing environment of the proposed system consists of a cluster of multi-nodes. Typically, one machine in the cluster is defined as the master node. The master has two components Namenode for distributed storage (HDFS daemon) and ResourceManager as computing management services (Yarn daemon). The rest of nodes in the cluster are designated as Workers or Slaves. Each worker composed of Datanode and NodeManager.

There is another component used to build this architecture, it is Hadoop Distributed File System that aims to manage the data during the life cycle of our system. Indeed, First, Namenode represents the master node that keeps the namespace system of files

Table 1 Cluster nodes configuration

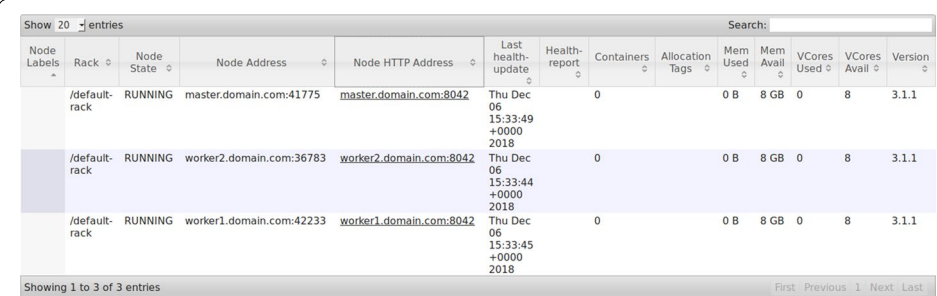| Machine	Network	Cores	Memory (GB)	Disk (GB)
Master	192.168.56.101 (master.domain.com)	8 Core i5 (7th Gen)	8	32
Woorker1	192.168.56.102 (worker1.domain.com)	8 Core i5 (7th Gen)	8	32
Worker	192.168.56.103 (worker2.domain.com)	8 Core i5 (7th Gen)	8	32

Fig. 7 Apache Yarn Web UI for cluster resource manager: It shows the configuration of a cluster that contains three nodes. One node defined as a master and the rest are the workers

and directories of Hadoop cluster the naming system and control access to data managed by workers. While, Datanode read and write the data in physical files in the form of blocks; it is responsible for block creation, deletion; replication, etc.

Experiment setup

For experimental tests, we have prepared a cluster composed of tree virtualized nodes. Virtual Box is used as hypervisor which is a powerful virtualization solution. It is an Open source Software that offers a rich set of tools and allows access to virtual machine resources like virtual disk, memory, Ethernet and so on.

With the available computing resources, we created 3 virtual machines by installing Ubuntu 18.04.1 LTS operating system across all nodes of the cluster. These nodes are connected with each other using a virtual private local network. The capacity and configuration of all the virtual machine are described in Table 1 and Fig. 7 via the Web User Interface of Hadoop Yarn resource manager.

After the clusters' virtual network configuration is finished, we have unpacked and installed Hadoop 3.1.1 and Spark 2.3.1 in the cluster node. Then, we have transferred the installation folder of both Frameworks into the slaves' nodes using SSH protocol, especially, Secure Copy (SCP) function in order to have the same copy of Apache Hadoop and Spark. Also in every node has been installed the java version 1.8.0_181, and we set up passwordless ssh between the nodes such that Hadoop master can connect, start, stop and execute jobs in different workers.

Spark application development

In Spark, an application is the combination of two things: a Spark cluster to runs jobs, and code source that implements higher-level APIs namely SparkSQL, DataFrames, and Datasets. In the development of the course recommender system, Spark cluster

is manager by Hadoop Yarn. On the other hand, the program code is written in Scala which represents a native programming language of Spark Framework. Moreover, the application is managed using Apache Maven as building and packaging tool; and it created and compiled via Eclipse 4.7.0 (Oxygen) IDE that provides Scala 2.12.6 plug-in.

To implement the parallel FP-growth algorithm, the developed Spark application needs two Spark APIs, which are: Spark SQL (org.apache.spark.sql) to connect to the data source, and MLlib (org.apache.spark.ml) for applying advanced analysis techniques and machine learning algorithms (Additional file 1). In order to prepare the required data, before launching the execution of the analytical model, we collect data from ESTenLigne database. To do that, we develop a SQL query to extract the list of courses followed by all learners using database structure in Fig. 4. Indeed, we focus essentially on four tables includes mdl_user, mdl_user_enrolments, mdl_enrol and mdl_course. To do this task efficiently, we use Spark SQL library that allows JDBC driver to easily connect and execute SQL statements. Then, we need to record courses by student id (user_id), so the individual courses followed by a given learner are aggregated into a single record as an array of courses) which represent the input data of PFP-growth algorithm. Furthermore, it is important to split the ESTenLigne dataset into training (70%) and test (30%) datasets to evaluate the performance of the Association Rules model. The training data is used for learning and fit the model to identify frequent items, whereas the test data aims to examine the input items against all the association rules and summarize the consequents as predictions.

Results and discussion

To distribute and execute the code on a cluster, Spark application must be packaged in a JAR file. Indeed, we create an assembly package containing the code and its dependencies. And in order to run jar package, we use spark-submit from the command line that allows scheduling the job across a cluster of three nodes. In fact, spark-submit is responsible for sending and launching the execution of spark application code on a Yarn resource manager. When running spark course recommender application, we create 12 executors distributed across the three nodes of the Hadoop cluster, and the memory size of each executor is 2G. The script for submitting Spark application into a Hadoop cluster is shown below:

```
bin/spark-submit –master yarn –deploy-mode cluster
–class org.karim.spark.sparkmaven.FpGrowth
–executor-memory 2G –num-executors 12 ′PFP
— growth-jar-with-dependencies.jar′
```

To run PFP-growth we should specify the minimum support and confidence threshold, respectively, to find more strong relationships between courses enrollments. The number of interesting association rules changes according to the value of support and confidence and the database size. We use the minimum support threshold of 5% and we set 60% as a minimum confidence threshold. In fact, PFP-growth generates three types of results that satisfy the specified minimum support and confidence thresholds. First, it calculates the list frequent courses in e-learning database, as shown in Table 2 that displays the 10 top frequent itemsets.

Table 2 Frequent itemsets

Items	Count
[43]	167
[11]	128
[9]	125
[42]	123
[14]	93
[15]	88
[45]	87
[46]	84
[9, 43]	80
[46, 45]	80

Table 3 Top 10 generated association rules

Rule	Antecedent	Consequent	Confidence
[1]	[46, 11]	[45]	1.000
[2]	[45, 11]	[46]	0.975
[3]	[46]	[45]	0.952
[4]	[45]	[46]	0.919
[5]	[6, 7]	[18]	0.868
[6]	[7]	[18]	0.818
[7]	[6, 18]	[7]	0.785
[8]	[7, 18]	[6]	0.733
[9]	[6]	[18]	0.711
[10]	[7]	[6]	0.690

Second, it gives a set of interesting relationships between courses enrolments rules. The top 10 useful rules ordered by the confidence measure are illustrated in Table 3.

According to the obtained results of confidence; as we can see from the experimental results in Table 3, the association between courses {11 and 46} and {45} has the highest confidence; also the association between courses {7} and {6} are the lowest. According to the most interesting rules extracted from transaction database and the calculated values of the confidence, it is clear which courses are more likely followed by learners and we can determine the suitable course to recommend for each learner. For example, the rule 1 {11, 46} $\Rightarrow$ {45} has the highest confidence, so our system recommend course 45 to students who enroll in courses {11 and 46}. According to Table 3, the efficiency of rule 1 is 100% because there are 56 students who enroll in courses {11 and 46} where 56 among them enroll also in course {45}. For course rule 2, there are 57 students in historical data of learners enrollments who take both of courses {11 and 45}, 56 among them follow course {46} in subsequent courses, so the efficiency of rule 2 is 98%. So the system recommends the course {46} to students who enrolled in courses {11 and 45}. With regard to rule 3, there are 123 learners enroll in course {46}, because a high proportion (118 learners) of them enroll also in course {45} in subsequent courses, the efficiency of rule 3 is 95%, so the system recommends the course {45} to students who enrolled in course

{46}, and so on. For the top 10 rules, we notice that the values of confidence are between 0.69 (69%) and 1.00 (100%), which proves that we have obtained good results. Thus, we can conclude that the proposed course recommender system provides the most appropriate pedagogical resources to learners.

Once the recommender model is established on training data (70%), it can be used to predict the outcome of course recommendations for learners using test dataset (30%). Then, the recommender system suggests a catalog of courses for each learner profile. Table 4 shows the top 10 predictions among 48 predictions in total.

The result of RS prediction, as shown in Table 4, illustrates for each learner (id): list of course in which he or she is participating and catalog of courses predictions. For example, the proposed course recommender system suggests courses 18 and 7 to the learner (id$=$541) who are enrolled in courses 46, 6, 11 and 45 courses. Also, it recommends course 14 to the learner (id$=$720) who follows 45, 46 and 15 courses, and so on.

In order to measure quality of predictions produced by the courses recommender system about the most relevant set of courses for each learner, we need to evaluate the relevance of the recommended courses using several metrics. To do this, we have applied an offline evaluation methodology because it is very safe a more suitable for our use case. This type of evaluation doesn't require any interaction with real users, so it hasn't any risk on disturbing users. The offline evaluation estimates the prediction error generated by using an existing dataset of learners' enrolments.

Essentially, we have employed two important measures include Precision and Mean Average Precision. They are respectively defined by the following equations:

$$P(k) = \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{K} \sum_{J=0}^{\min(|D|,k)-1} relDi\big(Ri(j)\big) \tag{3}$$

$$MAP = \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{|Di|} \sum_{J=0}^{Q-1} \frac{relDi\big(Ri(j)\big)}{j+1} \tag{4}$$

where r: recommended document, D: set of ground truth relevant document, it can be defined as: $D_i = \{d_0, d_1, \ldots, d_{N-1}\}$, R: list of Q recommended documents, it can be

Table 4 Courses recommender system predictions

Learner ID	Items	Prediction
541	[46, 6, 11, 45]	[7, 18]
720	[45, 46, 15]	[14]
19	[6]	[7, 18]
277	[18, 14, 6, 9, 3, …]	[43]
287	[17, 43, 42, 9, 2 …]	[6]
155	[15, 17, 42, 9]	[14, 43]
1157	[6]	[7, 18]
184	[40, 6, 43, 42]	[7, 18]
274	[7, 6, 15, 4, 3, …]	[43]
766	[15, 45, 46]	[14]

defined as: $R_i = \{r_0, r_1, \ldots, d_{Q-1}\}$, rel_D: a function returns a relevance score for the recommended document, it is defined as:

$$rel_D(r) = \begin{cases} 1, & if \ r \in D \\ 0, & otherwise \end{cases}$$

Unfortunately Spark-ML doesn't provide an implementation of these measures for frequent pattern mining model. Waiting for the Spark community to provide one of the metrics to evaluate the FP-growth parallel prediction results, we will develop a trail implementation of these evaluation measures adapted to our use case by using Spark RDD data structure. Our future work aims to provide more detailed information about this implementation.

FP-growth vs Apriori performance

In order to evaluate the performance of our course recommendation engine, we measure the result of the FP-growth algorithm compared to Apriori. Table 5 shows the difference in execution time between FP-growth and Apriori algorithm using to supports thresholds ranging from 0.5 to 3%.

The results of the experiments show that the performance of FP-growth is much better than the Apriori algorithm. Actually, Apriori needs more time to generate frequent itemset compared with FP-growth. This is because as the minimum support threshold decrease, the number and the length of frequent itemsets increase exponentially. As illustrated in the line chart (Fig. 8), for the support threshold of 2.5%, Apriori passes 0.06 s to find interesting rules while FP-growth needs only 0.04 s. Because Apriori

Table 5 FP-growth vs Apriori run-time

Support	FP-growth	Apriori
0.005	0.07	0.09
0.01	0.06	0.08
0.015	0.05	0.07
0.02	0.05	0.07
0.025	0.04	0.06
0.03	0.04	0.06

Fig. 8 Performance of FP-growth vs Apriori based on the support threshold: the experiments results shows that parallel FP-growth algorithm is more faster than Apriori
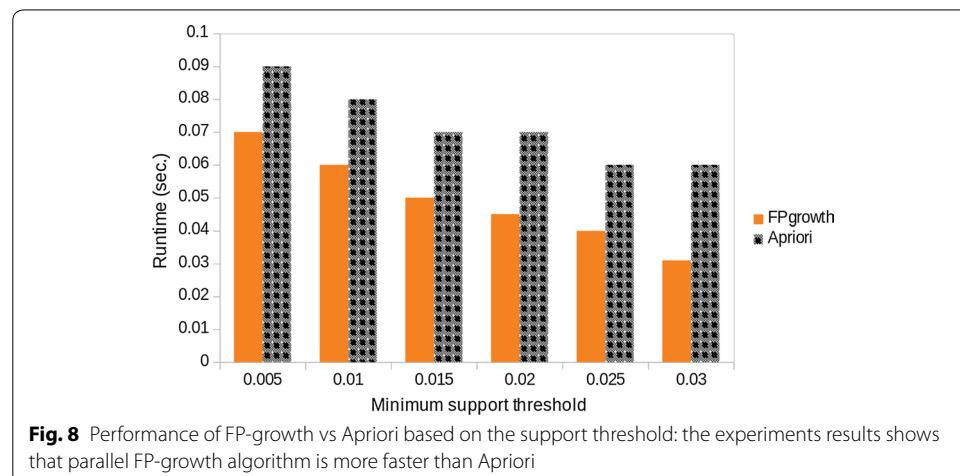
requires many database scans, in other words transactions are scanned at each candidate itemset generation; whereas FP-growth reduces the number of scans to two. First, FP growth scans the database to determine the frequent itemsets in descending order based on their support, and infrequent itemsets are discarded; the second scan is done for the construction of FP-tree [8]. According to the obtained results, it clear that our recommender system using FP-growth run faster rather than when we choose to work with the Apriori algorithm.

Apache Spark vs Weka and R performance

We establish a comparative analysis between execution time on Spark, R and Weka tool to show the scalability of our solution. For that, we increase the number of transactions in order to get a sufficient database size. Then, we measure the speed of the FP-growth algorithm using Scala and MLlib library compared to the same algorithm in Weka. Also, we measure the performance of our system using RStudio software. In fact, we have compared the running time of FP-growth in the cluster (spark) against single-machine (Weka). The results are plotted in Fig. 9 which illustrates the line chart of execution times of different environments.

As we can see from the line chart, running association rules model with Apache Spark is faster that Weka and R tools. It takes only 34 s to process 3000 transactions when Weka takes 86 s and RStudio 65 s. The parallel FP-growth algorithm of Spark Mllib achieves good scalability due to the distributed computing on cluster nodes. SparkMLlib processes data in less time because it splits the work into several tasks which are executed on workers nodes. From the above analysis, we may conclude that spark represents the best tool to implements the developed course recommender system.

Monitoring tools

Several monitoring tools are used to help us to see the progress of running the submitted jobs and understand how Spark application runs on a Hadoop cluster. Spark UI [36] is a monitoring tool that tracks the execution process of Spark application on Hadoop cluster. It shows important information about application execution, including jobs, stages,
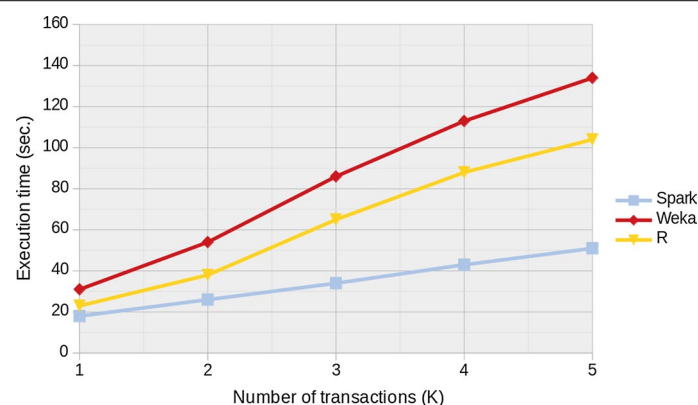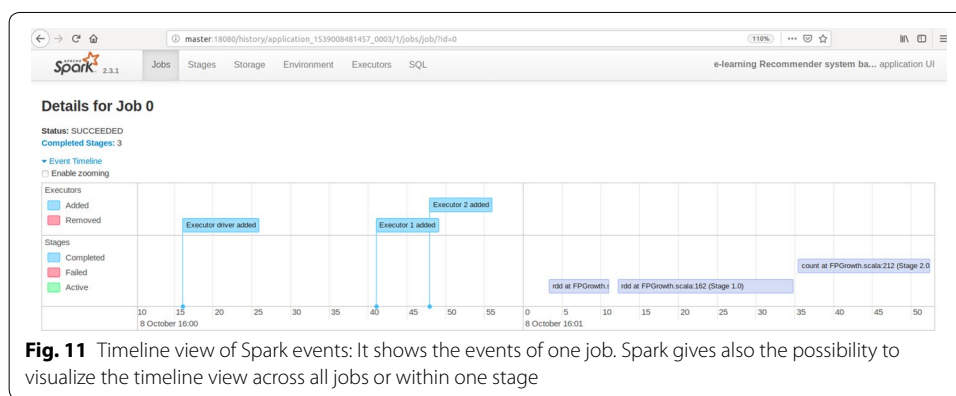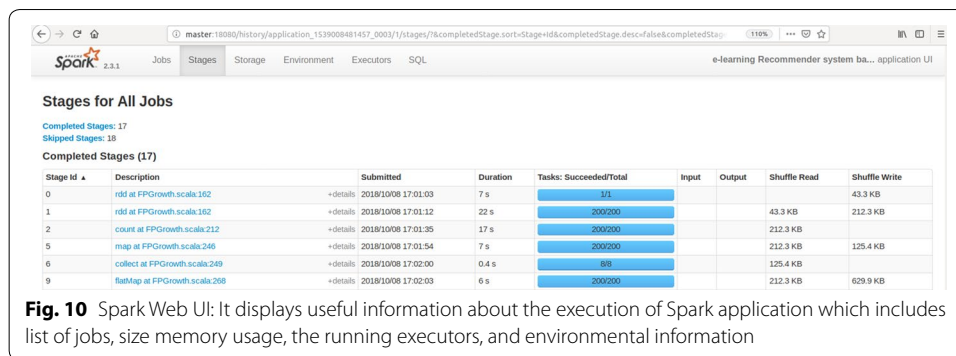

Fig. 9 Comparative study between Spark, Weka and R: As we can see from the histogram, Spark MLlib algorithm allows best performance compared to Weka and R tools
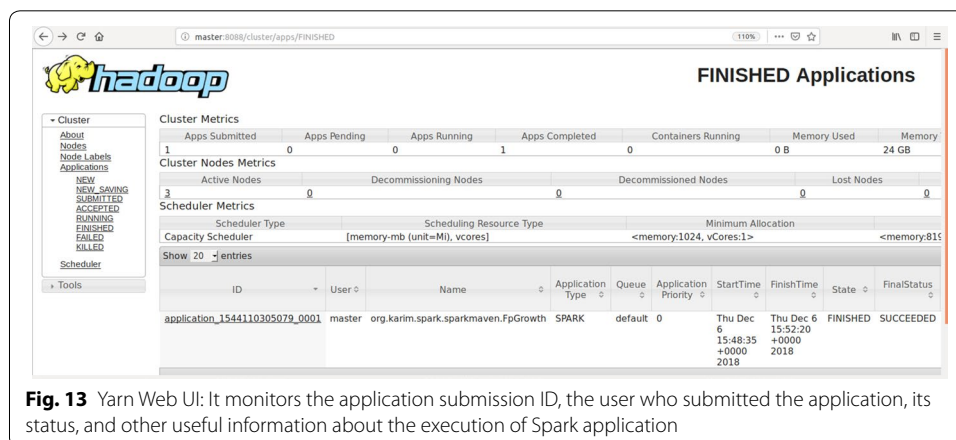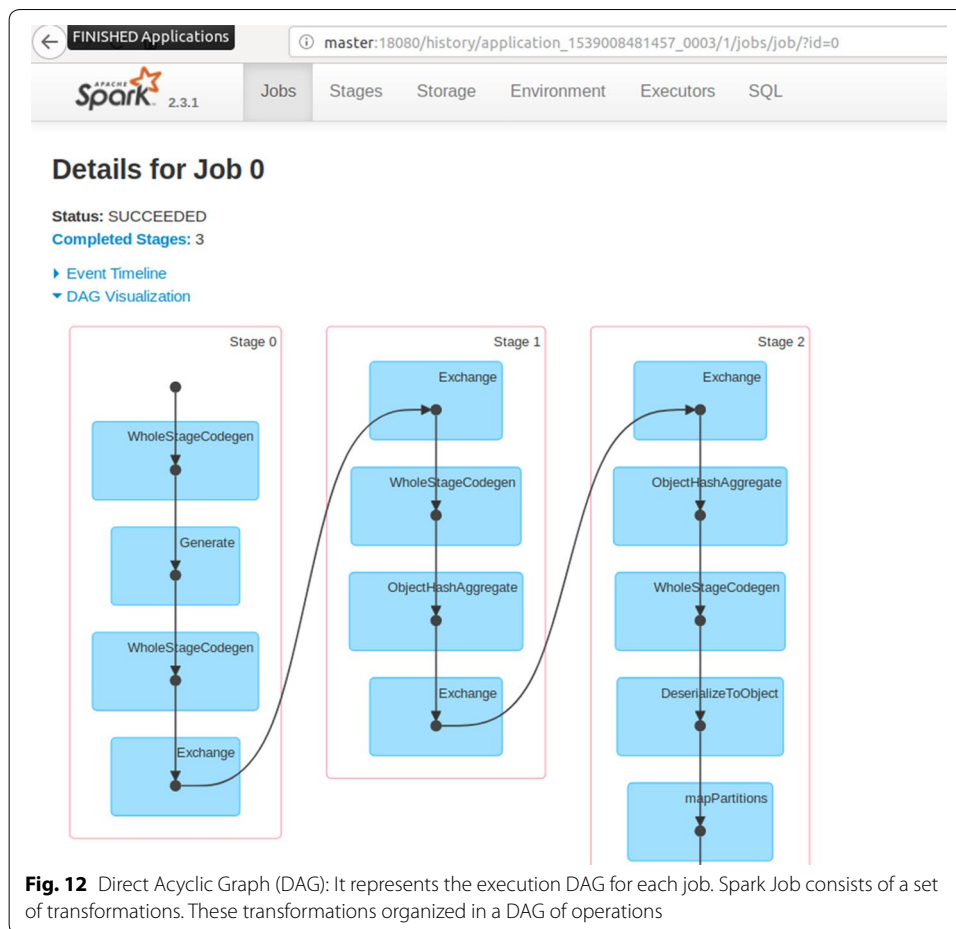
Fig. 10 Spark Web UI: It displays useful information about the execution of Spark application which includes list of jobs, size memory usage, the running executors, and environmental information

Fig. 11 Timeline view of Spark events: It shows the events of one job. Spark gives also the possibility to visualize the timeline view across all jobs or within one stage

and tasks. Also, it displays environment variables and allows DAG visualization. We can view the behavior of a job via a web browser, available at http//:master:18080.

As we can see from Fig. 10, a stage can have one of the three stages: active, pending and completed. Additionally, we have the possibility to displays Spark events in a timeline (Fig. 11) such that the relative ordering, which is available on three levels: across all jobs, within one job, and within one stage. Also, Spark UI visualization allows displaying the execution DAG for each job. In Spark, a job is set of RDD organized in DAG (direct acyclic graph) that looks like the following (Fig. 12). Furthermore, Hadoop provides Web UI for HDFS and another for Yarn Resource manager. The job's progress can be monitored via Yarn Web UI [37] (Fig. 13) of fully distributed mode which is available at http//:master:8088. We can observe the state of each job if finished with success or not. Yarn Web UI, we give us the possibility to kill a job during its runtime. In addition, if there are failed jobs will be listed in the failed section.

Ganglia monitoring tool

Ganglia [38] is a distributed and scalable open source tool for monitoring large computing environments such as clusters and grids. It is a powerful solution for big data infrastructure for enabling to measure the performance and resources consumption covering lot of metrics including CPU, memory, storage of each node across the cluster. It allows also monitoring the network by knowing the use of bandwidth. Furthermore, it is

Fig. 12 Direct Acyclic Graph (DAG): It represents the execution DAG for each job. Spark Job consists of a set of transformations. These transformations organized in a DAG of operations

Fig. 13 Yarn Web UI: It monitors the application submission ID, the user who submitted the application, its status, and other useful information about the execution of Spark application

graphically summarizes efficiently and clearly the overall resources utilization overview over the past hour, day, week, month and year. Figure 14 shows the main page of ganglia interface. It gives a clear overview of load, cpu, memory, network metrics of three monitored nodes of our cluster.
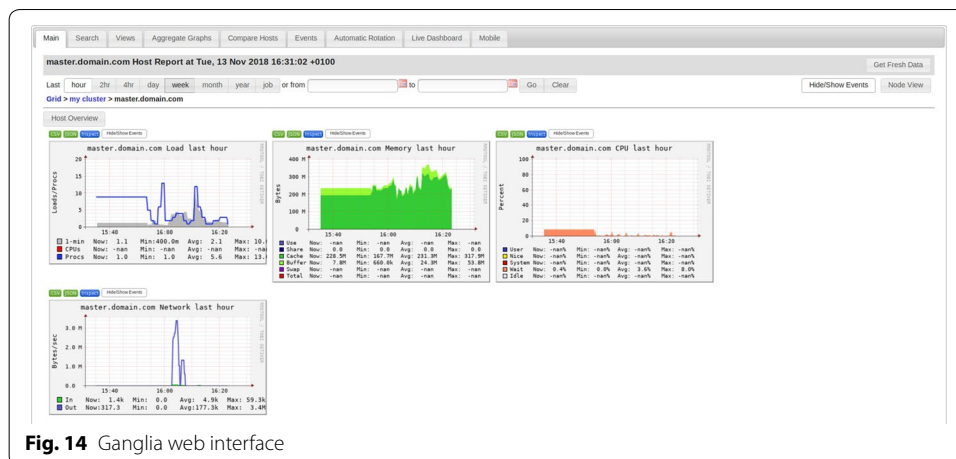
Fig. 14 Ganglia web interface
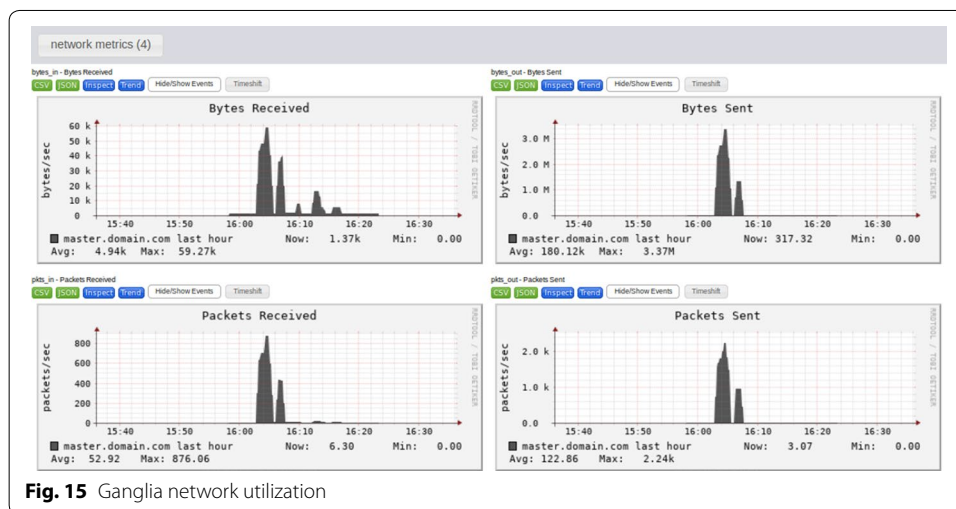
Fig. 15 Ganglia network utilization

Ganglia can also show amount of data transferred across a network as illustrated in Fig. 15. It shows host bandwidth including bytes or packets sent and received. In fact, it provides statistics about the number of bytes received and sent of the master machine. Also, Ganglia gives important information about packets size.

Conclusion

Recent development in new information and communication technologies, especially big data paradigm provides powerful frameworks and techniques which can be used to deal with online learning problems. This paper aims to create a distributed course recommender system for helping students' to take more appropriate pedagogical resources. It recommends a catalog of more suitable courses to the students in order to improve the level of learner satisfaction and performance through suggesting them with personalized and adapted recommendations and educational resources. The core concept behind the recommendation engine is analyzing past learners' behavior and activities within the e-learning platform using association rules method. To facilitate parallel processing of data and reduce the computation cost we employed Spark Framework and Hadoop ecosystem.

The implementation of the recommendation engine is built using the parallel FP-growth algorithm of MLlib machine learning library. Next, the deployment and execution of the proposed system are done across a distributed computing environment formed of a cluster of three nodes managed by Hadoop Yarn resources management. This system is based on a decentralized approach for data processing and analysis which gives the best performance compared to traditional machine learning tools which are running on a single machine. To validate our model, a real dataset of the e-learning platform of High School of Technology of fez is employed for collecting historical learner's activities. The results of the experiment of our system and the comparative study with Weka and RStudio software shows that in-memory computation of Spark is the fastest and scalable solution. In addition, our recommendation system has the ability to manipulate a huge volume of data and support a significant scalability through intensive and massively parallel processing. Furthermore, the presented system can be applied and easily integrated into other learning environments such as MOOC (Massive Open Online Courses).

The presented work showed certain benefits of integrating Spark and Hadoop Framework for improving online learning environments. However, it is still a lot of big data technologies and machine learning models to be implemented. In our future work will focus on incorporating Spark Streaming in the e-learning field. Actually, through real-time analysis of spark, it is possible to quickly extract value from live data streams. Accordingly, online learning professionals can assess rapidly the effectiveness of learning strategies in order to offer more personalized educational resources.

Additional file

> **Additional file 1.** Spark application of the e-learning recommender system.

Abbreviations
EST: High School of Technology; ESTenLigne: e-learning platform of EST of Fez; FP-Growth: frequent pattern growth; PFP: parallel FP-growth; supp: support measure; conf: confidence measure; MAFIA: Maximal Frequent Itemset; MLlib: machine learning library; numPartitions: number of partitions; GraphX: graph-parallel computation; DAG: Directed Acyclic Graph; YARN: yet another resource negotiator; HDFS: Hadoop distributed file system; IDE: integrated development environment.

Authors' contributions
KD developed and deployed the e-learning recommender system presented in this article, prepared and analyzed the data, interpreted the results, and authored the manuscript. Comments and guidance were received from AD. The data collection and acquisition was conducted by LO and AI. All mentioned authorscontribute to the elaboration of the article. All authors read and approved the final manuscript.

Author details
[1] Engineering Sciences Laboratory, FPT, Sidi Mohamed Ben Abdellah University, Taza, Morocco. [2] High School of Technology, Sidi Mohamed Ben Abdellah University, Fez, Morocco.

Publisher's Note

References

1. Dahdouh K, Dakkak A, Oughdir L, Messaoudi F. Big data for online learning systems. Educ Inf Technol. 2018;23:2783–800.
2. Dahdouh K, Dakak A, Oughdir L. Integration of the cloud environment in e-learning systems. Trans Mach Learn Artif Intell. 2017;5:4.
3. Mihai G. Recommendation system based on association rules for distributed e-learning management systems. ACTA Universitatis Cibiniensis. 2015;67:99–104.
4. Jooa J, Bangb S, Parka G. Implementation of a recommendation system using association rules and collaborative filtering. Procedia Comput Sci. 2016;91:944–52.
5. Hrženjak MP, Bakarić MB. Mining association rules in learning management systems; 2015.
6. Panigrahi S, Lenka RK, Stitipragyan A. A hybrid distributed collaborative filtering recommender engine using apache Spark. Procedia Comput Sci. 2016;83:1000–6.
7. Li H, Wang Y, Zhang D, Zhang M, Chang EY. Pfp: parallel fp-growth for query recommendation. In: Proceedings of the 2008 ACM conference on recommender systems—RecSys'08. Lausanne: ACM Press; 2008. p. 107.
8. Zhou Y, Wilkinson D, Schreiber R, Pan R. Large-scale parallel collaborative filtering for the Netflix prize. In: Fleischer R, Xu J, editors. Algorithmic aspects in information and management. Berlin: Springer; 2008. p. 337–48.
9. Liu J, Dolan P, Pedersen ER. Personalized news recommendation based on click behavior. In: Proceedings of the 15th international conference on intelligent user interfaces—IUI'10. Hong Kong: ACM Press; 2010.
10. Mangal N, Niyogi R, Milani A. Analysis of users' interest based on Tweets. In: Gervasi O, Murgante B, Misra S, Rocha AMAC, Torre CM, Taniar D, Apduhan BO, Stankova E, Wang S, editors. Computational science and its applications—ICCSA 2016. Cham: Springer International Publishing; 2016. p. 12–23.
11. Frawley WJ. Knowledge discovery in databases: an overview. AI Magaz. 1992;14:57.
12. Larose DT, Larose CD. Data mining and predictive analytics. Hoboken: Wiley; 2015.
13. Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. New York: ACM; 1993.
14. Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation. In: International conference on management of data, proceedings of the ACM SIGMOD; 2000. p. 1–12.
15. Agrawal R, Srikant R. Fast algorithms for mining association rules in datamining. In: Proceedings of the 20th VLDB conference Santiago, Chile, vol. 2. 1994. p. 13–24.
16. Burdick D, Calimlim M, Flannick J, Gehrke J, Yiu T. MAFIA: a maximal frequent itemset algorithm. IEEE Trans Knowl Data Eng. 2005;17:1490–504.
17. Zaki MJ. Scalable algorithms for association mining. IEEE Trans Knowl Data Eng. 2000;12:372–90.
18. MLlib: Main Guide—Spark 2.3.2 Documentation. https://spark.apache.org/docs/latest/ml-guide.html. Accessed 14 Oct 2018.
19. Apache Hadoop. http://hadoop.apache.org/. Accessed 14 Oct 2018.
20. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM. 2008;51:107.
21. The Hadoop Distributed File System (HDFS). https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. Accessed 14 Oct 2018.
22. Apache Hadoop YARN. https://hadoop.apache.org/docs/r3.1.1/hadoop-yarn/hadoop-yarn-site/YARN.html. Accessed 14 Oct 2018.
23. Apache Spark™. Unified Analytics Engine for Big Data. https://spark.apache.org/. Accessed 14 Oct 2018.
24. Giraph. Welcome To Apache Giraph! http://giraph.apache.org/. Accessed 14 Oct 2018.
25. Apache Storm. http://storm.apache.org/. Accessed 14 Oct 2018.
26. ESTenLigne platform of High School of Technology of Fez. http://elearn.est-usmba.ac.ma/. Accessed 14 Oct 2018.
27. Ibriz A. Une Démarche Innovante de Conduite de Projet Elearning: C.D.I.O. 2015.
28. Ibriz A, Safouane A. L'Innovation Pédagogique dans les EST du Maroc : Le Model et la Conduite d'un cas réussi à travers le Projet ESTenLigne. 2014.
29. EST de Fes. http://www.est-usmba.ac.ma/. Accessed 14 Oct 2018.
30. Instituts Universitaires de Technologie (IUT). http://www.iut.fr/. Accessed 14 Oct 2018.
31. Oughdir L, Ibriz A, Harti M. Modélisation de l'apprenant dans le cadre d'un environnement d'apprentissage en ligne. 2011.
32. Benslimane M, Ouazzani K, Tmimi M, Berrada M. Proposal of an approach of online course design and implementation: a case study of an algorithmic course. Int J Comput Technol Appl. 2016;7:7.
33. Ibriz A, Benslimane M, Ouazzani K. Didactics in online learning technical courses: a case study based on activity theory. Int J Comput Sci Inf Technol. 2016;7:849–54.
34. Moodle. Open-source learning platform|Moodle.org. https://moodle.org/. Accessed 14 Oct 2018.
35. Garg A. SCORM based learning management system for online training. 2012.
36. Monitoring and Instrumentation. Spark 2.3.1 Documentation. https://spark.apache.org/docs/latest/monitoring.html. Accessed 14 Oct 2018.
37. Yarn UI: Apache Hadoop 2.9.2. Hadoop: YARN-UI V2. https://hadoop.apache.org/docs/r2.9.2/hadoop-yarn/hadoop-yarn-site/YarnUI2.html. Accessed 14 Oct 2018.
38. Ganglia Monitoring System. http://ganglia.sourceforge.net/. Accessed 14 Oct 2018.