Open access • Journal Article • DOI:10.1007/S10586-013-0256-9

Distributed-based massive processing of activity logs for efficient user modeling in a Virtual Campus — Source link

Santi Caballé, Fatos Xhafa

Institutions: Open University of Catalonia, Polytechnic University of Catalonia

Published on: 01 Dec 2013 - Cluster Computing (Springer US)

Topics: Server log, Web log analysis software, User modeling, Big data and PlanetLab

Distributed-based massive processing of activity logs for efficient user modeling in a Virtual Campus

Santi Caballé · Fatos Xhafa

Abstract This paper reports on a multi-fold approach for the building of user models based on the identification of navigation patterns in a virtual campus, allowing for adapting the campus' usability to the actual learners' needs, thus resulting in a great stimulation of the learning experience. However, user modeling in this context implies a constant processing and analysis of user interaction data during long-term learning activities, which produces huge amounts of valuable data stored typically in server log files. Due to the large or very large size of log files generated daily, the massive processing is a foremost step in extracting useful information. To this end, this work studies, first, the viability of processing large log data files of a real Virtual Campus using different distributed infrastructures. More precisely, we study the time performance of massive processing of daily log files implemented following the master-slave paradigm and evaluated using Cluster Computing and PlanetLab platforms. The study reveals the complexity and challenges of massive processing in the big data era, such as the need to carefully tune the log file processing in terms of chunk log data size to be processed at slave nodes as well as the bottleneck in processing in truly geographically distributed infrastructures due to the overhead caused by the communication time among the master and slave nodes. Then, an application of the massive processing approach resulting in log data processed and stored in a well-structured format is presented. We show how to extract knowledge from the log data analysis by using the WEKA framework for data mining purposes showing its usefulness to effectively build user models in terms of identifying interesting navigation patters of on-line learners. The study is motivated and conducted in the context of the actual data logs of the Virtual Campus of the Open University of Catalonia.

Keywords Massive processing · Log files · Cluster computing · PlanetLab · Web mining usage · WEKA framework · Navigation patterns · Virtual Campus

1 Introduction

User modeling [3] is a mature research field mostly involved in the information technology context. It is mainly utilized in software systems for inferring the users' goals, skills, knowledge, needs and preferences, thus achieving more adequate adaptation and personalization on the basis of the user activity pattern built. This inference process relies in turn on being able to track the users' actions when interacting with the application such as the users' choice of buttons and menu items [10].

In this paper, we focus on and are interested in web-based learning applications that support virtual campuses. These applications, due to the high degree of user interaction, take great advantage of the tracking-based techniques of user modeling, such as providing broader and better support for the users of Web-based educational systems [10]. An important consideration in this context is the building of learner models based on the identification of navigation patterns, which allows for adapting the system's usability to the actual learners' needs resulting in a great stimulation of the learning experience. However, the information generated

S. Caballé (✉)
Open University of Catalonia, Rambla Poblenou, 156,
08018 Barcelona, Spain
e-mail: scaballe@uoc.edu

F. Xhafa
Technical University of Catalonia, c/ Jordi Girona, 1-3,
08034 Barcelona, Spain
e-mail: fatos@lsi.upc.edu

in web-based learning applications can be of a great variety of type and formats [22]. Moreover, these applications are characterized by a high degree of user-user and user-system interaction which stresses the amount of interaction data generated.

The above features can be extracted to a large extent from information kept in log data files of on-line web-based learning applications. As a matter of fact, log file data processing and analysis of the information captured from the actions performed by learners is a core function for the modeling of the learner's behavior during the learning process and of the learning process itself as well [5]. It is clear that, the larger the taxonomy of events captured in log data files, the richer is the information and knowledge extracted due its processing. It is thus of interest to persist in log files all possible information due to the events produced during user-to-system and user-to-user interaction. Keeping all possible information, however, comes to the price of (a) dealing with all sorts of data log formats and (b) large amounts of data, which are not human readable nor ready to be analyzed by statistical or data mining techniques [14, 21].

Therefore, massive processing is a must to efficiently process large amount of log data files. In fact, one can as well use log data files as a basis for real time decision taking system, and thus, high performance data processing using large scale distributed systems are needful. In this paper we present the implementation and evaluation of some massive processing techniques under different distributed infrastructures such as Cluster Computing [9] and PlanetLab [18]. The aim is to identify the limits of the size of the log files for which these infrastructures are efficient solutions to such processing.

The rest of the paper is organized as follows. In Sect. 2 we describe the aims and background of our approach by first presenting the Virtual Campus and its log data file system as the real context of our study and then showing the problem of processing huge amounts of log data coming from the user interaction in the Virtual Campus. In Sect. 3 we describe the research methodology to validate our approach and the evaluation results achieved from massive processing of the log data files using different distributed infrastructures. Finally, in Sect. 4 we present a data mining application that leverages our massive log data processing approach in order to extract useful knowledge in form of navigation patterns. We end the paper in Sect. 5 with some concluding remarks and outlining future directions of research.

2 Aims and background

In this section we first describe the real context of learning of our study and then present a description of the log data files of our virtual campus. Finally, we present the problem of massive processing of log files.

2.1 The Virtual Campus

The context of our study is the academic and administrative activity at the Open University of Catalonia (UOC) [19], which is sustained by a large web-based Virtual Campus counting at the time of this writing on about 65,000 students, lectures and tutors from everywhere who participate in some of the 70 official degrees and other PhD and post-graduate programs resulting in more than 3,000 on-line classrooms.

The on-line Web-based campus of the UOC is completely virtualized. It is made up of individual and community virtual areas such as mailbox, agenda, classrooms, library, secretary's office, and so on. See left column of Table 1 for a complete list of the virtual areas of the UOC. Students and other users (lecturers, tutors, administrative staff, etc.) continuously browse these areas where they request for services to satisfy their particular needs and interests. For instance, students make intensive use of the mailbox area so as to communicate with other students and lecturers as part of their learning process.

2.2 Log data files of the Virtual Campus

All users' requests in the Virtual Campus are chiefly processed by a collection of Apache web servers [1] as well as database servers and other secondary applications, all of which are providing service to the whole community, thus satisfying a large number of users. For load balance purposes, all HTTP traffic is smartly distributed among the different Apache web servers available. Each web server stores in a log file each user request received and the information generated from processing it. Once a day (namely, at 01:00 a.m.), all web servers in a daily rotation merge their logs producing a single very large log file containing the whole user interaction with the campus performed in the last 24 hours.

For the purpose of registering the campus activity, log files entries were set up with the purpose of capturing the following information in several fields: (i) Who performed a request (i.e. user's IP address along with a session key that uniquely identifies a user session); (ii) When the request was processed (i.e. timestamp); (iii) What type of service was requested (a URL string format description of the server application providing the service requested along with the input values); and (iv) Where (i.e. an absolute URL containing the full path to the server application providing the service requested).

The log files of the UOC's Virtual Campus are made up of millions of lines, each of which representing an operation performed by a particular user (student, teacher, staff member,...) in the Virtual Campus. Among other information, these lines register the IP address from which the user accessed the campus, and the user name of the student, ensuring greater privacy. As an example, following is a line

Table 1 Log Data Files Samples (in every row: Total number of log lines (top) and % of number of log lines (down)). Row with *Indefinite log lines* refer to those log lines without relevant information

Area/Log	LOG1	LOG2	LOG3	LOG4	LOG5	Avg (%)
TOTAL	2.7E+07	1.6E+07	2.6E+07	2.7E+07	2.4E+07	100 %
	100 %	100 %	100 %	100 %	100 %	
Login	2943646	1855002	2900904	3318269	2927306	11.64 %
	11.02 %	11.28 %	11.35 %	12.40 %	12.15 %	
My UOC	5691130	3635355	5561056	5602895	5160005	21.50 %
	21.31 %	22.11 %	21.75 %	20.94 %	21.41 %	
Services	64325	35994	58397	53584	38946	0.21 %
	0.24 %	0.22 %	0.23 %	0.20 %	0.16 %	
Community	184781	117051	179219	184895	164851	0.70 %
	0.69 %	0.71 %	0.70 %	0.69 %	0.68 %	
Rooms	3749263	2380543	3465224	3846491	3341856	14.06 %
	14.04 %	14.48 %	13.55 %	14.37 %	13.87 %	
Secretaria	347024	185218	335855	299620	259359	1.19 %
	1.30 %	1.13 %	1.31 %	1.12 %	1.08 %	
Tutoring	445202	264995	424024	420972	388788	1.62 %
	1.67 %	1.61 %	1.66 %	1.57 %	1.61 %	
Library	2272	16341	20474	21039	18101	0.07 %
	0.01 %	0.10 %	0.08 %	0.08 %	0.08 %	
Research	1173	283	931	892	1164	0.00 %
	0.00 %	0.00 %	0.00 %	0.00 %	0.01 %	
Intrauoc	23317	7550	24525	23058	17661	0.08 %
	0.09 %	0.05 %	0.10 %	0.09 %	0.07 %	
News	167480	88739	153551	138785	118298	0.45 %
	0.63 %	0.54 %	0.06 %	0.52 %	0.49 %	
Bolonia Space	48	18	48	56	14	0.00 %
	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	
Inbox (New)	421305	196709	407673	34511	314802	1.39 %
	1.58 %	1.20 %	1.59 %	1.29 %	1.31 %	
Inbox (New)	2982332	1763556	279633	2976493	2649881	10.99 %
	11.17 %	10.72 %	10.94 %	11.12 %	11.00 %	
Agenda	635494	387673	611917	651428	575066	2.39 %
	2.38 %	2.36 %	2.39 %	2.43 %	2.39 %	
My Profile	22203	16818	21836	22627	20599	0.07 %
	0.08 %	0.10 %	0.09 %	0.09 %	0.02 %	
Groupwork	3631	1932	3769	3745	3460	0.01 %
	0.01 %	0.01 %	0.02 %	0.01 %	0.01 %	
Service attention	720	229	662	675	567	0.00 %
	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	
Favorites	177	112	184	212	205	0.00 %
	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	
Search	54534	42542	51514	67740	56991	0.23 %
	0.20 %	0.26 %	0.20 %	0.25 %	0.24 %	
Accessibility	872	509	2833	3025	1721	0.01 %
	0.00 %	0.00 %	0.01 %	0.01 %	0.01 %	
Preferences	20955	12545	2833	36392	25327	0.22 %
	0.79 %	0.08 %	0.01 %	0.13 %	0.11 %	
Logoff	7785	4491	7212	6527	6173	0.03 %
	0.03 %	0.03 %	0.03 %	0.02 %	0.03 %	
Indefinite log lines	8935395	5431082	8536732	8735702	8012468	33.13 %
	33.36 %	33.025 %	33.385 %	32.644 %	33.242 %	

that is part of a real log data file of the Virtual Campus (note the IP address has been anonymized for privacy):

```
[13/Mar/2012:00:15:42 +0100]
xxx.xxx.xxx.xxx "POST /tren/trenacc
HTTP/1.1" 200
"https://cv.uoc.edu/tren/trenacc"
"Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/535.11 (KHTML, like Gecko)
Chrome/17.0.963.78 Safari/535.11" 14943 157
```

It should be noted that the example is a simplified line, usually the URLs listed in the log files are accompanied by numerous parameters.

The size of these log files keeps growing, due, on the one hand, to the increase in the number of users of the virtual campus, and on the other hand, to the variety of event information kept in the log data files. Currently a typical daily log file size may be up to 15–20 GB (about more than 50 % increase in log data files size of 5 years before [7]). This great amount of information is first pre-processed using filtering techniques in order to remove a lot of futile, non relevant information (e.g. information coming from automatic control processes, the uploading of graphical and format elements, etc.). However, after this pre-processing, about 1.5–2.0 GB a day of potentially useful information corresponding to about 5,000,000 of log entries in average still remains.

Log file entries are structured following a type of format known as Common Log Format (CLF) [8], which is produced by most of web servers including Apache and is fairly configurable. At this point, we highlight certain problems arisen by dealing with these log files. Each explicit user request generates at least an entry in the log file and after being processed by a web server, other log entries are generated from the response of this user request; certain nontrivial requests (e.g. user login) involve in turn requesting others and hence they may implicitly trigger new log entries; the What and Where fields contain very similar information regarding the URL strings that describe the service requested and the parameters with the input values; certain information is found in a very primitive form and is represented as long text strings (e.g. user session key is 128-character string long). Therefore, there is a high degree of redundancy, tedious and ill-formatted information as well as incomplete as at some cases certain user actions do not generate any log entry (e.g. user may leave the campus by either closing or readdressing the browser) and thus these actions have to be inferred. As a consequence, treating this information is very costly in time and space needing a great processing effort.

2.3 The need for massive processing of log data

From our experience at the UOC, the description and prediction of our students' behavior and navigation patterns when interacting with the campus is a first issue. Indeed, a well-designed system's usability is a key point to stimulate and satisfy the students' learning experience. In addition, the monitoring and evaluation of real, long-term, complex, problem-solving situations is a must in this context. Our goal is to understand and adapt the learning process and objects to the actual students' learning needs as well as to validate the campus' usability by the actual usage of the campus.

In order to achieve these goals, the analysis of the campus activity and specifically the users' traces captured while browsing the campus is essential in this context. The collection of this information in log files and the later analysis and interpretations of this information provide the means to model the user's behavior and activity patterns. For instance, from the log data it is possible to capture the different areas browsed by a student during his/her user session along with the timestamp when accessing to these areas. This allows us to know what the most popular areas are, how long in average students remain in each area, user session time in average and in different daily periods, navigation patterns combining both the most and the least visited areas, and so on. More advanced processing would address web sessions clustering to be used for web site restructure, decision taking and recommender system, user activity prediction, server load prediction, etc.

However, in Web-based learning applications in general, extracting navigation and behavior patterns from the analysis of user interactions is a difficult task due to both the amount and the complexity of information generated. This makes its later treatment very tedious and time-consuming.

Therefore, in order to construct a reliable, effective, useful learner models, dealing with log files in eLearning applications must involve three separate, necessary steps: collection of information, analysis and presentation [4]. During the first step, a tight structuring and classification of the original log data is needed. This information is then analyzed and interpreted in order to extract the desired knowledge. The final step is to provide users with the obtained knowledge. In this paper we consider and deal with the two first steps of this process.

During the first stage of this process, the most important issue while monitoring learning activity is the efficient collection and storage of the large amount of information generated. Given that such informational data may need a long time to be processed, Web-based learning systems have to be designed in a way that filter and pre-process the resulting information effectively. The aim is, on the one hand, to correctly collect and store the learning activity and, on the other hand, to increase the efficiency during the later data processing and analysis stages. In the context of our university, the whole user interaction generates a huge amount of information in a day which is filtered and collected in large daily log files. Furthermore, this large information is found in an ill-structured highly redundant form needing a great amount of computational power to constantly process log data.

To sum up, the main issues with log files that concern in the Virtual Campus are:

– Log file size: Daily log files contain hundreds of thousands to millions of lines and therefore require considerable computing power to be read and processed.
– Log file structure: Although the log files follow a defined structure, this structure does not allow to directly extract navigation patterns, and thus a preprocessing of these log files is required. The resulting output of the preprocessing are files ready to be used by mining analysis tools.

2.4 Related work

Most of the existing approaches in the literature consider a sequential approach mainly due to three reasons: (i) processing for a specific purpose (i.e. limiting the quantity of information needed for that purpose); (ii) processing the information afterwards (i.e. not in real time) and (iii) processing of small data samples, usually for research and testing purposes (i.e. not for real learning needs). Yet, the lack of sufficient computational resources is the main obstacle to process large amounts of data in real time and hence in real situations this processing tends to be done off-line in order to avoid harming the performance of the logging application, but as it takes place after the completion of the learning activity has less impact on it [6, 23].

Recently, distributed infrastructures, such as Cluster Computing [9], is increasingly being used to reduce the overall, censored time in processing data by taking advantage of its large computing support. The concept of distributed infrastructures has emerged as a way of capturing the vision of a networked computing system that provides broad access not only to massive information resources, but to massive computational resources as well. Thus, in this paper, we show how a distributed infrastructure approach can be used to match the time processing requirements.

Several studies have been conducted at the UOC [5–7, 16] to show that a distributed infrastructure can increase the efficiency of processing a large amount of information from group activity log files [22]. Some of these studies have involved the interaction data collected from the log files of both the BSCW system [2] used at the UOC to support Problem-Based Learning practices in small groups and the own virtual campus of the UOC. The experimental results allowed us to show first the gain provided by the distributed approach in terms of relative processing time and, second, the benefits of using the inherent scalable nature of the approach while the input log files are growing up in both number and large size.

In this paper we report the different experiments carried out at the UOC that first show the feasibility of the Cluster Computing and PlanetLab infrastructures in terms of the complexity and challenges of massive processing in the big data era and then provide guidelines of how to achieve an effective embedding of the extracted knowledge into e-learning practices.

3 Research methodology and computational results

In order to deal with the above mentioned problems and inconveniences, this section presents a research methodology to validate our processing approach mentioned in previous sections. To this end, we first propose an algorithm for parallelizing the processing of log files. Then, we present several proofs of concepts to evaluate the computational results by using several distributed infrastructures. Finally we will interpret and discuss on the experimental results.

3.1 Parallelizing the processing of log files

We have developed a simple application in Java, called *UOCLogsProcessing* that processes log files of the UOC. However, as the processing is done sequentially, it takes too long to complete the work and it has to be done after the completion of the learning activity, which makes the construction of effective real-time user models not possible [16].

The parallel implementations in the distributed infrastructures that we propose in this section follows the master-worker (MW) [22] paradigm . In a nutshell, a log file of the UOC Virtual Campus is split off into a certain number of parts, which can be exactly equal to the number of peer nodes (slaves) that will participate in the processing or can be larger. In this later case some peer nodes could receive more than one part for processing. By splitting the original file into more parts than peer node candidates for processing, we can achieve different degrees of granularity of the parallel processing. Achieving different degrees of granularity is very desirable in distributed environments given the high heterogeneity of computing resources. Note that we have a perfect split of the problem in many independent parts. In the end, the master node just needs to append to a unique file the arriving of partial solutions (partial result files after processing).

The main steps of the MW parallel algorithm to process a log file in our distributed infrastructures are as follows:

1. [*Pre-processing phase*]: *UOCLogsProcessing* counts the total number of lines of the log file, `totalNbLines`, and knowing the total number of parts to split the file off, `nbParts`, each peer node will receive and process a `totalNbLines/nbParts` of lines from the file.
2. [*Master Loop*]: Repeat
   (a) Read `totalNbLines/nbParts` lines from the original file and create a file with them.

(b) Create a request and submit the partial file to the distributed infrastructure.

(c) [*Parallel processing*]:

(i) The request is assigned to a peer node of the distributed infrastructure.

(ii) The peer node, upon receiving the petition, reads according to the petition's description, the part of the file it has to read via HTTP. The peer runs *UOCLogProcessing* functionality for processing the lines of the file, one at a time, and stores the results of the processing in a buffer.

(iii) The peer node, once the processing of the petition is done, sends back to the master node the content of the buffer.

Until the original log file has been completely scanned.

3. [*Master's final phase*]: Receive messages (partial files) from slave nodes and append in the correct order the newly received resulting file to the final file containing the information extracted from the original log file.

3.2 Computational results

We present some computational results for the massive processing of log files up to 15 GB size. The implementations follow the master-slave paradigm [16], a classical parallel model for massive processing. It should be noticed here that the nature of log files recording entries of different independent events makes it possible their splitting into a number of smaller files for processing.

The study of this processing is divided into two parts: in the first part, we deal with a fixed number of nodes and vary the size of logs, and in the other part, we vary the number of nodes. The aim is to see the scalability of the massive processing in terms of the size of the log file and also in terms of the processing nodes of the distributed infrastructure.

The results are obtained using two different platforms: (i) a cluster computing environment and (ii) the PlanetLab platform infrastructure. Results are also given for processing in a local node to have a reference of the sequential processing. In processing the log file data, we consider two parameters, namely, processing time and size of the log files. Chunks of files ranging from 50 MB to 15 GB are used.

3.2.1 Local processing

A PC with standard configuration is used for local sequential processing of the log files. We present in Tables 2 and 3 the processing times as a function of log file size, and the graphical representation is shown in Fig. 1.

Table 2 Local processing of Log files (from 50 MB to 500 MB)

Size	50 MB	100 MB	250 MB	500 MB
Time	13 s	21 s	46 s	93 s

Table 3 Local processing of Log files (from 1 GB to 15 GB)

Size	1 GB	2 GB	4 GB	8 GB	15 GB
Time	197 s	413 s	817 s	1741 s	3076 s

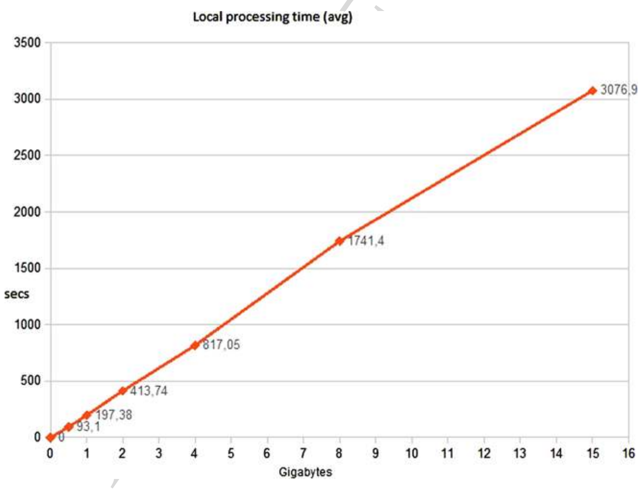


Fig. 1 Local processing time of Log files

3.2.2 Cluster processing

For the cluster processing we use a cluster infrastructure[1] as shown in Fig. 2. The master node in the cluster (called UOCLogs node) has the following hardware characteristics and configuration:

```
CPU: QEMU Virtual
CPU version 0.12.3 64bits 3192.498Mhz
RAM: 2005MB
Hard drive: 100GB
```

The other slave nodes (called Eagles) have hardware characteristics similar to the master node:

```
CPU: QEMU Virtual
CPU version 0.12.3 3192,364 Mhz
RAM: 2005MB
Hard drive: 100GB
```

Now, obviously, the processing time is different due to the distributed processing of the chunks of the log files at different nodes of the cluster. More precisely, we have three phases of processing with the corresponding timing. First, the log file is divided into as many chunks as nodes (slave

[1] eagle.lsi.upc.edu.

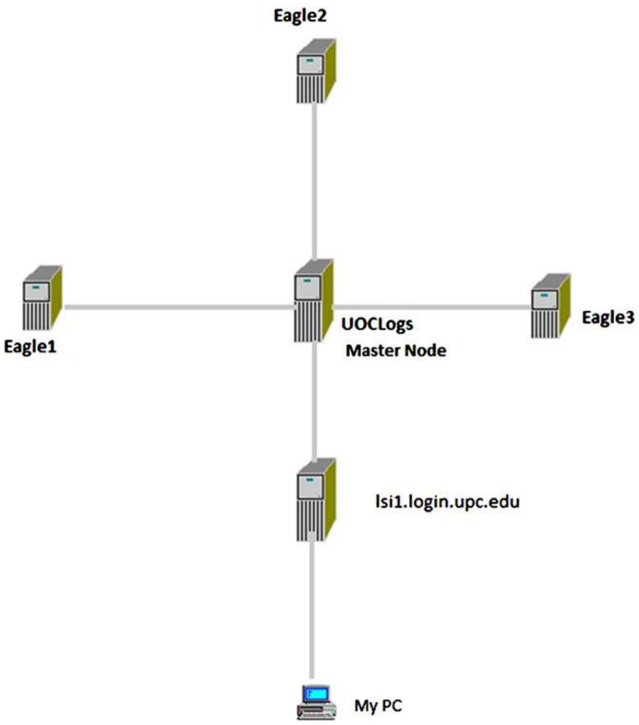


Fig. 2 Distributed processing infrastructure

Table 4 Cluster processing of Log files (with 2 nodes)

Size	50 MB	100 MB	250 MB	500 MB
Time	5 s	6 s	12 s	18 s

Table 5 Cluster processing of Log files (with 2 nodes)

Size	1 GB	2 GB	4 GB	8 GB	15 GB
Time	49 s	111 s	335 s	902 s	1437 s

Table 6 Cluster processing of Log files (with 4 nodes)

Size	50 MB	100 MB	250 MB	500 MB
Time	4 s	7 s	10 s	15 s

nodes) available (excluding the master node). Next, the different chunks are distributed to different nodes and finally, the results of the processing, which are files, are joined into one single file. The processing time (averaged) is then the sum of the processing times of the three phases.

We present in Tables 4 and 5 the processing times for various sizes of log files using two nodes of the cluster and the graphical representation in Fig. 3.

We present in Tables 6 and 7 the processing times for various sizes of log files using four nodes of the cluster and the graphical representation in Fig. 4.

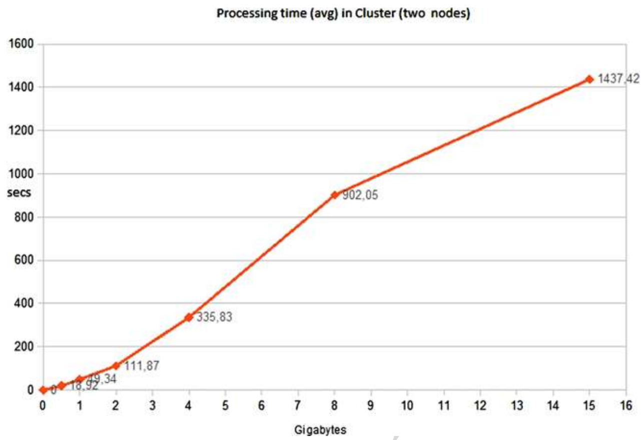


Fig. 3 Cluster processing time of Log files

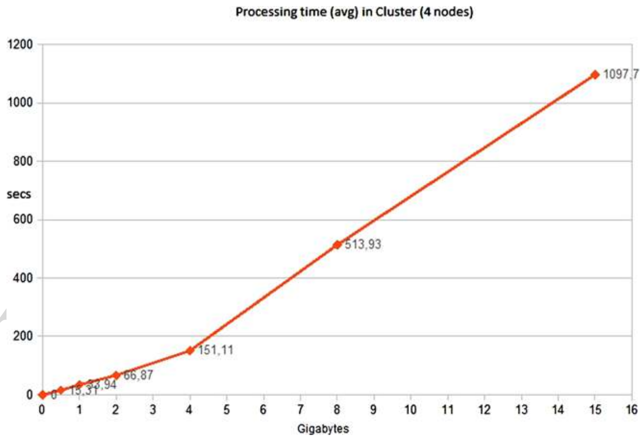


Fig. 4 Cluster processing time of Log files

Table 7 Cluster processing of Log files (with 4 nodes)

Size	1 GB	2 GB	4 GB	8 GB	15 GB
Time	33 s	66 s	151 s	513 s	1097 s

3.2.3 PlanetLab processing

PlanetLab is a distributed computing infrastructure currently with 1161 nodes distributed in 548 different sites among universities, research centers or homes. The nodes are located outside a firewall and have to be visible from anywhere in more than one DNS. Currently, the characteristics of the node must be the following [18]:

- 4 GB RAM
- At least 500 GB hard disk
- At least 1 MB/sec connection to the Internet
- cores@2.4Ghz 4× Intel (e.g., 2× dual core or quad core)
- External or built-in CPU, remote-access power-reset capability, accessible from PLE, such as IntelAMT, HPiLO, DellRAC, IPMIv2, etc.

Table 8 PlanetLab nodes

PlanetLab Node	Location
dplanet2.uoc.edu	(Spain)
planetlab2.iitr.ernet.in	(India)
planetlab3.upc.es	(Spain)
planetlab01.cs.tcd.ie	(Ireland)
planetlab3.di.unito.it	(Italy)
planetlab2.fct.ualg.pt	(Portugal)
planetlab1.upc.es	(Spain)
onelab1.info.ucl.ac.be	(Belgium)
planetlab4.cs.uiuc.edu	(USA)
ple1.cesnet.cz	(Czech Republic)
onelab3.warsaw.rd.tp.pl	(Poland)
planetlab2.willab.fi	(Finland)
planetlabpc2.upf.edu	(Spain)
planetlab2.upc.es	(Spain)
dplanet1.uoc.edu	(Spain)
dplanet2.uoc.edu	(Spain)
planetlab2.iis.sinica.edu.tw	(Taiwan)
planetlab-02.ece.uprm.edu	(Puerto Rico)
planetlab1.pop-mg.rnp.br	(Brasil)
planetlab2.rd.tut.fi	(Finland)
planetlabpc1.upf.edu	(Spain)
planetlab2.cis.upenn.edu	(USA)
pl1.cis.uab.edu	(USA)
planetlab2.urv.cat	(Spain)
planetlab-01.ece.uprm.edu	(Puerto Rico)
planetlab4.n.info.eng.osaka-cu.ac.jp	(Japan)
planetlab1.cs.ucla.edu	(USA)
server4.planetlab.iit-tech.net	(USA)
pl1.pku.edu.cn	(China)
pl2.pku.edu.cn	(China)
pl2.cis.uab.edu	(USA)
planetlab2.eurecom.fr	(France)
planet2.elte.hu	(Hungary)

Table 9 PlanetLab processing of Log files (16 nodes)

Size	50 MB	100 MB	250 MB	500 MB
Time	2 s	3 s	5 s	10 s

For the purpose of this study we use up to 32 nodes of the PlanetLab. We continue to use as the master node that of UOCLogs to divide and join the logs but without participating itself in the processing.

The list of PlanetLab nodes used are listed in Table 8, which as can be observed are geographically distributed across Europe, USA. South America and Asia.

Table 10 PlanetLab processing of Log files (16 nodes)

Size	1 GB	2 GB	4 GB	8 GB	15 GB
Time	23 s	45 s	101 s	270 s	528 s

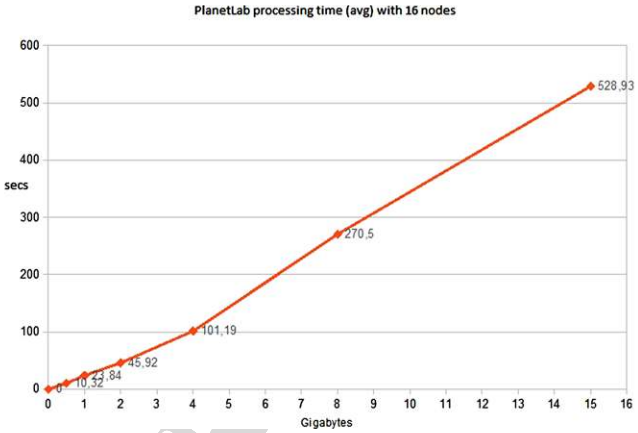


Fig. 5 PlanetLab processing time of Log files with 16 nodes

Table 11 PlanetLab processing of Log files (32 nodes)

Size	50 MB	100 MB	250 MB	500 MB
Time	1 s	2 s	3 s	7 s

Table 12 PlanetLab processing of Log files (32 nodes)

Size	1 GB	2 GB	4 GB	8 GB	15 GB
Time	21 s	42 s	81 s	242 s	458 s

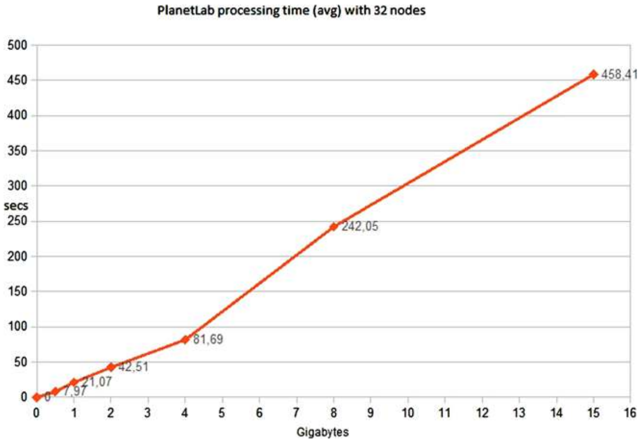


Fig. 6 PlanetLab processing time of Log files with 32 nodes

We present in Tables 9 and 10 the processing times using 16 nodes of PlanetLab platform (the 1st 16 nodes of Table 8) and the graphical representation in Fig. 5. In addition, Tables 11 and 12 present the processing times using 32 nodes and the corresponding graphical representation is shown in Fig. 6.

Table 13 Comparative processing of Log files with one node in three platforms (Local-Cluster-PlanetLab)

	Local	Cluster	PlanetLab
500 MB	93 s	30 s	61 s
1 GB	197 s	73 s	121 s
2 GB	413 s	167 s	268 s
4 GB	817 s	335 s	557 s
8 GB	1741 s	693 s	–
15 GB	3077 s	1299 s	–

Table 14 Comparative processing of Log files with multiple nodes in PlanetLab platform

	500 MB	1 GB	2 GB	4 GB	8 GB	15 GB
1 node	61 s	121 s	269 s	557 s	–	–
2 nodes	36 s	76 s	153 s	430 s	1129 s	–
4 nodes	18 s	42 s	93 s	191 s	616 s	1325 s
8 nodes	13 s	28 s	63 s	126 s	324 s	821 s
16 nodes	10 s	23 s	47 s	101 s	271 s	529 s
24 nodes	8 s	22 s	46 s	92 s	251 s	481 s
32 nodes	8 s	21 s	43 s	82 s	242 s	458 s

3.3 Evaluation

In Table 13 we can see a comparative results of the battery of tests for each of the three platforms described above with one node (i.e., Local, Cluster, PlanetLab). Note that no results are shown in PlantetLab for log files bigger than 4 GB due to the space restriction of this platform. Also note that the PlanetLab times may be influenced by the node workload carried out at the time of the experiment. Finally, the time difference between the local and cluster processing is to do with the different processing capacity between a home PC and a server in the cluster. In all cases, the results for one node shows a linear increase over time.

From the experimental study shown in the previous section for more than one node (see Figs. 3–6), we can see that the results do not grow linear anymore. In Table 14 and the corresponding Fig. 7 we can see an example of processing log files with multiple nodes in PlanterLab platform. We can see also that by using a distributed infrastructure of up to 16 nodes, a considerable speed up (more than 40 %) is achieved in processing large log file data (up to 15 GB). For infrastructures larger than 16 nodes truly geographically distributed, such as PlanetLab, the speed up reduces with the increase in the number of nodes due to the significant communication time (in receiving the chunk files and sending back the results to master node).

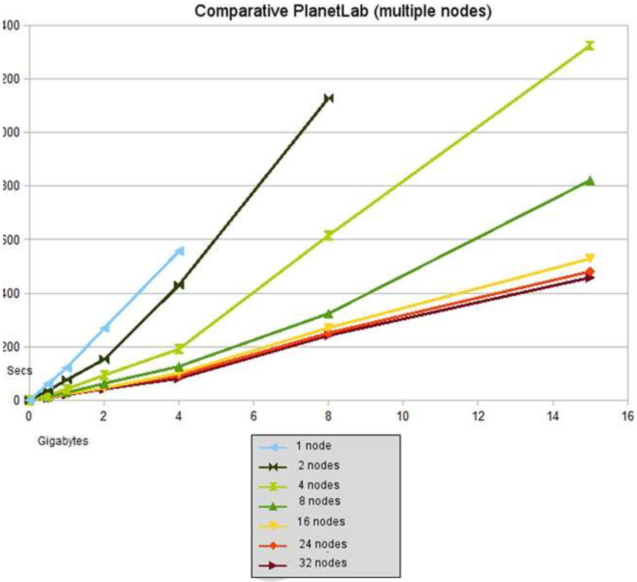


Fig. 7 Comparative of processing time of Log files in PlanetLab with multiple nodes

Table 15 Comparative of processing time of Log files in the Cluster infrastructure

	1 node	2 nodes	3 nodes	4 nodes
500 MB	30 s	19 s	18 s	15 s
1 GB	73 s	49 s	49 s	33 s
2 GB	167 s	167 s	80 s	66 s
4 GB	336 s	335 s	208 s	151 s
8 GB	693 s	902 s	655 s	514 s
15 GB	1299 s	1437 s	1213 s	1098 s

The above results are confirmed by the results in the Cluster platform, where we can see that not considerable gain of time is achieved with a maximum of 4 nodes (see Table 15 and the corresponding Fig. 8). Even with 2 nodes the processing time is higher than with 1 node for large log files due to the overhead caused by first the split function that divides the original log files and then merging the processing results. Note that by comparing PlanetLab and Cluster platforms, the latter achieves better results in similar conditions (i.e., same number of nodes and log size) due to both the greater power provided by machines in a cluster infrastructure with fast communication and also full availability for the experiments being PlanetLab platform constantly busy.

Once the original log files have been processed by our *UOCLogsProcessing* algorithm and the results have been merged in well-structured files, we observed a enormous reduction of the overall data up to 95 %. This allows a feasible storing of the processing results in a database for further analysis and knowledge extraction. Indeed, the difference between handling an original 15 GB log file and a resulting 800–900 MB files is very considerable. Next section reports a complete experience of using the results of log file processing for knowledge extraction in our virtual campus.

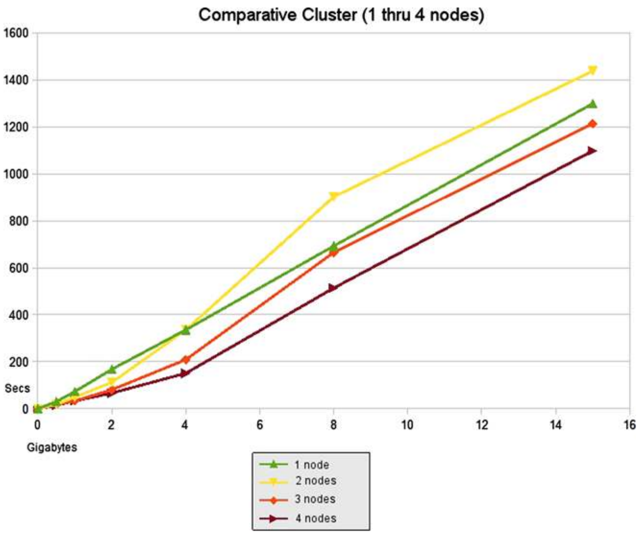


Fig. 8 Comparative of processing time of Log files in Cluster infrastructure

4 An application: extracting navigation patterns in the Virtual Campus

In this section we present an application of the results of log data processing obtained in the previous section for extracting navigation patterns in our Virtual Campus. In the general information and knowledge management process in eLearning systems presented in Sect. 2.3, consisting of three stages (data processing, extraction of knowledge and knowledge presentation) [4], the study presented in this section corresponds with the second stage, namely the extraction of knowledge from the analysis of the log data processed and stored in a structured way, such as a database and a well-structured text file.

We first describe the navigation patterns of our Virtual Campus we are interested in. Then, we introduce some data mining methods suitable for our purposes as well as the use of WEKA framework to implement the selected data mining methods. Finally, we show the computational results and the eventual association rules obtained forming the interested navigations patterns.

4.1 Navigation patterns

As precisely described in Sect. 2.2, a user access to the UOC's Virtual Campus, on the one hand, provide access to a range of resources (typically, files or web pages). On the other hand, accessing to the Virtual Campus means the start of a work session by the user logged in. This work session is identified by an alphanumeric string. From the point of view of log files of the Virtual Campus, users are identified by the IP address they have accessed the Campus and/or a resource, reflected in a line in the log file corresponding to the day/time on which the access has been made. Moreover, resources are identified by their URL in the Virtual Campus, which incorporates the session identifier of the user who accessed the resource.

For the purposes of this work we concentrate on the analysis of two navigation patterns:

1. *Commonly accessed resources:* This pattern will identify those resources most frequently requested by users of the UOC's Virtual Campus. Given the couple `<IP address, session ID>`, we define a navigation sequence as the set of resources visited by a particular IP address during a work session, i.e. a navigation sequence shall consist of the URLs that a user has accessed from logging on the Virtual Campus until the end of the session activity. Thus, if the same user (identified by the IP address) initiates a new session, the set of resources accessed would be considered part of another sequence.
2. *Common navigation rules:* This pattern will identify the most common behaviors of users of the UOC's Virtual Campus. For example, given the set of resources [R1, R2, R3, R4] for which users' hits have been recorded in a log file of the Virtual Campus, we could determine those users who have accessed R1 have accessed also to R2, and users who have accessed R3 have accessed to R4 as well.

4.2 Data mining methods

In order to determine the above mentioned navigation patterns, we apply the following data mining methods.

4.2.1 K-Means

K-Means (or centroid method) is a method of aggregation/clustering, which from a data set computes a list of groups (clusters) of objects having similar characteristics. Specifically, the K-means method is based on obtaining a number of K groups K, set at the beginning of the process [17].

According to the method, the process starts by setting a starting point of the space (called a seed) as a potential center of the group to be formed. This seed can be either one of the objects that are part of the initial data set, or a combination of artificially created values representing the characteristics of various objects. Thus, the main steps taken by the K-means method are as follows:

1. Select the initial seeds.
2. Calculate the centers.
3. Assign objects to the nearest center group.
4. Recalculate the centers.
5. Repeat until there is no variation in the groups.

For the present study, the groups to be computed are sequences of users browsing the Virtual Campus of the UOC, so that we can determine the resources that were accessed more frequently.

4.2.2 *The Apriori algorithm*

Apriori is a classical algorithm for learning association rules [13]. The algorithm is based on prior knowledge (*a priori*) of the common data sets. An item is considered frequent if its frequency is greater than the value of trust (confidence). It should be noted that this algorithm does not support numerical values so that depending on the characteristics of the data set to be processed, it may be required to perform a pre-processing of the same, usually to make a discretization.

Association rules An association rule is an expression of the form $X \Rightarrow Y$ where X and Y are item sets. This expression should be interpreted as: when there is a data set that contains the item X, it is also often to contain the element Y.

Confidence and support Given the set of elements X and Y, and binary database r, the confidence, represented as $conf(X \Rightarrow Y, r)$ is the conditional probability that a randomly chosen line within r matches X, also matches with Y. In other words, confidence refers to cases that a rule predicts correctly.

On the other hand, the support (or frequency) refers to cases that covers a rule. It is usually represented as $Supp(X, r)$ where X is a set of elements and r a binary database. Generally, the association rules of interest are those with a high support value.

For the present study, the application of the Apriori algorithm is to discover association rules on the access to resources made by users of the UOC's Virtual Campus, recorded in log files.

4.2.3 *FPGrowth algorithm*

FPGrowth is an algorithm that behaves more optimally and faster than Apriori with large volumes of data, since it only performs two iterations [13]. This algorithm is based on storing information in compressed form in a tree structure called FP-tree and this is where improvement is achieved on Apriori to reduce the number of iterations required to process information.

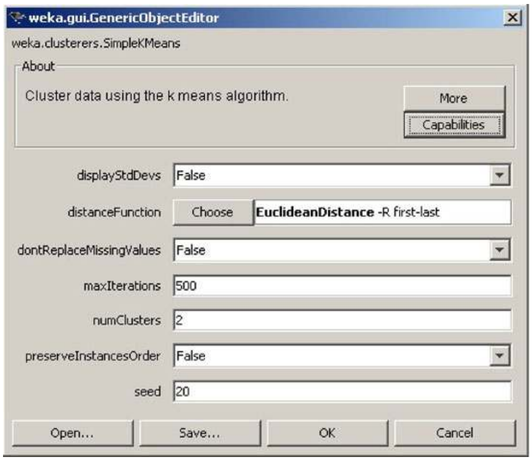
4.3 WEKA application

We have selected WEKA application for the data mining of the log data files. WEKA implements the three data mining methods mentioned above (K-means, Apriori and FP-Growth) [21].

4.3.1 *Features*

WEKA [20] is a Java application that has a collection of visualization tools and algorithms for preprocessing data analysis and predictive modelling, coupled with a graphical user interface for easy access to its functionality.

The main features of this tool are the following:


weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

displayStdDevs: False

distanceFunction: EuclideanDistance -R first-last

dontReplaceMissingValues: False

maxIterations: 500

numClusters: 2

preserveInstancesOrder: False

seed: 20

Fig. 9 Graphical User Interface of WEKA (K-means method)

- This is free software, licensed under GNU.
- It is easily portable to other platforms by being completely developed in Java.
- Supports various data mining tasks, namely: data preprocessing, clustering, classification, regression, visualization, and selection.
- Makes it possible to read data files in plain text (CSV, ARFF, ...) or directly from relational databases through the JDBC API.
- It is extensible, so new algorithms can be incorporated.

4.3.2 *Implementation of K-means in WEKA*

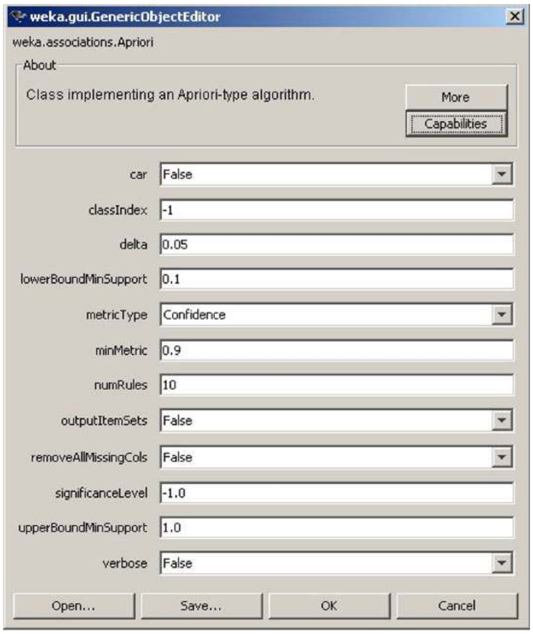
WEKA implements the method K-means through a known algorithm, which in turn is implemented in the class SimpleKMeans `weka.clusterers.SimpleKMeans`. The implementation of this algorithm is parameterized (see Fig. 9) by a set of options from which the most relevant are the `numClusters` and `Seed`.

4.3.3 *Implementation of Apriori in WEKA*

WEKA implements the Apriori association algorithm through `weka.associations.Apriori` class. The implementation of this algorithm is configurable through a series of options that can be seen in snapshot of Fig. 10.

4.4 Computational results

We present here some computational results for the real log data files of the UOC's Virtual Campus. The starting point is the pre-processing tasks described in Sect. 3, which provide the final files in a well-structured format for suitable processing in the WEKA framework. As mentioned in Sect. 3, this preprocessing consisted in several steps (from reading the original log file, removing unnecessary lines, identifying sessions and session Ids, removing URL options, identifying IP, session and resource, conversion to CSV format of



weka.gui.GenericObjectEditor

weka.associations.Apriori

About

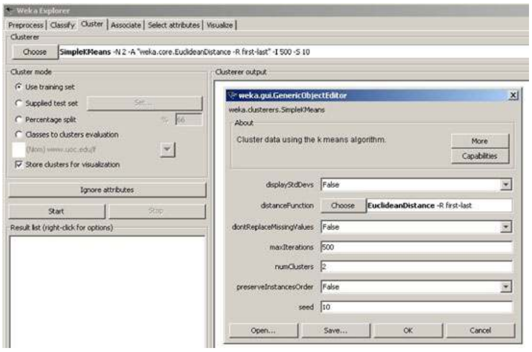
Class implementing an Apriori-type algorithm.

More | Capabilities

car	False
classIndex	-1
delta	0.05
lowerBoundMinSupport	0.1
metricType	Confidence
minMetric	0.9
numRules	10
outputItemSets	False
removeAllMissingCols	False
significanceLevel	-1.0
upperBoundMinSupport	1.0
verbose	False

Open... | Save... | OK | Cancel

Fig. 10 Graphical User Interface of WEKA (Apriori method)



Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Clusterer

Choose SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

Cluster mode

● Use training set

○ Supplied test set Set...

○ Percentage split % 66

○ Classes to clusters evaluation

(Nom) www.uoc.edu/f

☑ Store clusters for visualization

Ignore attributes

Start | Stop

Result list (right-click for options)

Clusterer output

weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm.

More | Capabilities

displayStdDevs	False
distanceFunction	Choose EuclideanDistance -R first-last
dontReplaceMissingValues	False
maxIterations	500
numClusters	2
preserveInstancesOrder	False
seed	10

Open... | Save... | OK | Cancel

Fig. 11 Parameters of 1st run of K-means in WEKA

Table 16 Results of SimpleKMeans using default parameters

Number of lines	1.384
Lines in group 0	168 (12 %)
Lines in group 1	1.216 (88 %)

Table 17 Results of SimpleKMeans (numClusters = 10)

Number of Lines	1.384
Lines in group 0	40 (2.89 %)
Lines in group 1	259 (18.71 %)
Lines in group 2	133 (9.61 %)
Lines in group 3	36 (2.60 %)
Lines in group 4	429 (31.00 %)
Lines in group 5	1 (0.07 %)
Lines in group 6	117 (8.45 %)
Lines in group 7	87 (6.29 %)
Lines in group 8	227 (16.40 %)
Lines in group 9	55 (3.87 %)

final file for mining). The final file ready for mining has the following format:

```
IP,resource1,resource2,resource3
111.111.111.111,T,T,F
222.222.222.222,T,F,F
333.333.333.333,T,F,T
444.444.444.444,F,F,T
555.555.555.555,T,T,T
111.111.111.111,T,F,F
```

As can be seen, in the final file, it is distinguished between the sequence of actions performed by the IP address 111,111,111,111 during an initial session and the sequence of actions performed by the same IP address in a different session. ‘T’ means the resource has been accessed and ‘F’ means there has been no access.

The final file is then processed in WEKA and results are analyzed and interpreted.

4.4.1 *Results obtained with WEKA K-means*

First Run Firstly, we executed the aggregation (clustering) SimpleKMeans algorithm, using the default parameters set by Weka (see snapshot of Fig. 11).

Running the algorithm with default parameters set by WEKA produced two groups (clusters) with the following distribution given in Table 16.

As a matter of fact, the 1st run of WEKA SimpleKMeans using default parameters produced poor results (in terms of identified sessions and resources). Due to the large number of data, the execution of the SimpleKMeans setting produced only two groups (clusters), which could be considered biased and inconclusive results (this is very noticeable through the tables of results of the first iteration), therefore, we performed a second execution of the algorithm by defining 10 groups (WEKA numClusters parameter value 10). The results obtained after the re-execution of the algorithm are as shown in Table 17.

Given the resources that have been marked as accessed in any of the groups and the total set of groups generated by the algorithm, the results obtained were as shown in Table 18. In the table, ‘T’ indicates that the resource has been accessed and ‘F’ (represented by an empty cell) that there has been no access to the resource.

Analysis of results From the results in Tables 17 and 18 the following can be noted:

1. The largest group identified is the group 4, which has brought together a total of 429 lines.

Table 18 Results of accessing the resources per groups (‘T’ represents a resource accessed by a group)

Resource/Group	0	1	2	3	4	5	6	7	8	9
/webapps/classroom/081_common/jsp/eventFS.jsp			T				T			
/WebMail/listMails.do				T						
/UOC2000/b/cgi-bin/ma_filter						T				
/cgi-bin/uocapp						T				
/UOC/a/jsstuff_mail.html									T	
/WebMail/resources/html/bodyHeight.html	T			T						
/webapps/classroom/081_common/jsp/iniciAula.jsp			T							
/UOC2000/b/extern_0.html						T				
/avis.html		T				T	T			T
/UOC2000/b/cgi-bin/ma_folders						T				
/UOC/a/ext_menu.html										T
/WebMail/readMailSecure.do	T			T						
/cgi-bin/avis						T	T			
/UOC2000/b/ext_menu.html						T				
/UOC/a/extern_0.html										T
/WebMail/resources/html/logobar.html				T						
/cgi-bin/ma_folders						T		T		
/cgi-bin/ma_buttons						T		T		
/cgi-bin/ma_mssgs						T		T		
/rb/inici/navigation/redir									T	
/UOC/js/banner.dat									T	
/webapps/classroom/081_common/jsp/event.jsp			T							
/UOC2000/b/cgi-bin/ma_buttons						T				
/cgi-bin/ma_filter						T				
/webapps/widgetsUOC/widgetsNovetatsExternesWithProviderServlet		T				T				T
/WebMail/sendMail.do	T			T						
/WebMail/readMail.do				T						
/webapps/widgetsUOC/widgetsRssServlet		T				T				
/UOC2000/b/cgi-bin/ma_mssgs						T				
/webapps/widgetsUOC/widgetsIcalServlet		T				T				
/WebMail/contacts.do	T			T						
/webapps/classroom/081_common/jsp/entrada.jsp			T				T			
/UOC/a/menu.htm									T	
/webapps/classroom/download.do			T							
Total access/group	4	4	5	7	0	16	4	3	4	4

2. The second and third largest groups are the groups 1 and 9 with 259 lines and 227 lines, respectively.
3. There is a group consisting of a single line.
4. Group 4, which has the largest number of lines has grouped values ‘F’ only.
5. Group 5, which only has one line, is the one for which the most resources have taken value ‘T’.
6. The resource that has taken more values ‘T’ in all the groups is ‘/avis.html’ (4 values), then ‘/webapps /widgetsUOC/widgetsNovetatsExternesWithProviderServlet’ (3 values). Both resources are found in rows 9th and 25th of Table 18.

Analyzing the content of the final file, the resources that have been considered by the algorithm have been accessed at least 42 times. Looking at only the accesses, arguably the most commonly accessed resources are represented in group 5 but the issue with that group is that it has only one line. Nor does it appear that the most common navigational sequence is represented by group 4, since the result is ‘distorted’ by the large number of values ‘F’ present in the final file. The most common resources are those who have been accessed

Table 19 Total number of accesses to resources

Resource	Total number of access
/avis.html	454
/webapps/classroom/081_common/jsp/entrada.jsp	406
/webapps/widgetsUOC/widgetsRssServlet	334
/webapps/widgetsUOC/widgetsIcalServlet	318
/rb/inici/navigation/redir	313
/webapps/classroom/081_common/jsp/eventFS.jsp	308
/webapps/widgetsUOC/widgetsNovetatsExternesWithProviderServlet	277
/UOC/a/jsstuff_mail.html	245
/cgi-bin/avis	245
/UOC/a/menu.htm	242

more often given the set of IP addresses and sessions of the final file (remind that one line of the final file is determined by the pair IP address-session). The most accessed resources are shown in Table 19.

Beyond the global statistics of Table 19, the data mining techniques employed here can give valuable insights on the navigation patters and accesses to the resources.

- Resources accessed by each student (identified by its IP address and session work) are quite heterogeneous, i.e. there are many differences between actions of students during a work session. On the other hand, we notice that students do not access a large number of resources during a single work session. This has resulted in the appearance of a large number of 'F' values in the final file. The fact that the data mining method has generated a group with many 'T' values but only grouping a line (group 5) states that it is unusual for a student to access too many resources during a work session on the Virtual Campus.

  The fact that a group has been created with all its values to 'F' (group 4), can also conclude that it is common that students do not perform many accesses to resources in each working session (only a few resources worth taking 'T' in each row of the final file) but also that overall students access to diverse resources and hence as many as 411 columns do appear in the final file.
- The analysis of groups 1 and 9 (the second and third that grouped most lines), allows us to conclude that, regardless of the total number of accesses, the students of the Virtual Campus tend to access resources whose URL is `/avis.html` and `/webapps/widgetsUOC/widgetsNovetatsExternes WithProviderServlet`.

  This is supported by the fact that these two resources are the ones that have taken more 'T' values in all the groups. This is so because the URLs that include the term `'Avis'` refer to some kind of notification issued in the Virtual Campus, just as the term `'widget'` could refer to the modules of the home page of the Virtual Campus, that `'classroom'` could refer to the classrooms to which students have access and `'mail'` to email.

4.4.2 *Results obtained with WEKA Apriori*

We started by running the Apriori algorithm using default parameters set by WEKA. The problem encountered was that the WEKA Apriori is not able to process large amounts of data, confirming again the scalability issue of the Apriori algorithm [13].

Because of this, it was necessary to find an alternative algorithm that would allow obtaining association rules using Weka. Based on the suggestion in [13] we considered the FPGrowth algorithm.

Results obtained with WEKA FPGrowth We run the FGGrowth for the first time using the default parameters and then decided to make a second run where the default parameters were modified as follows:

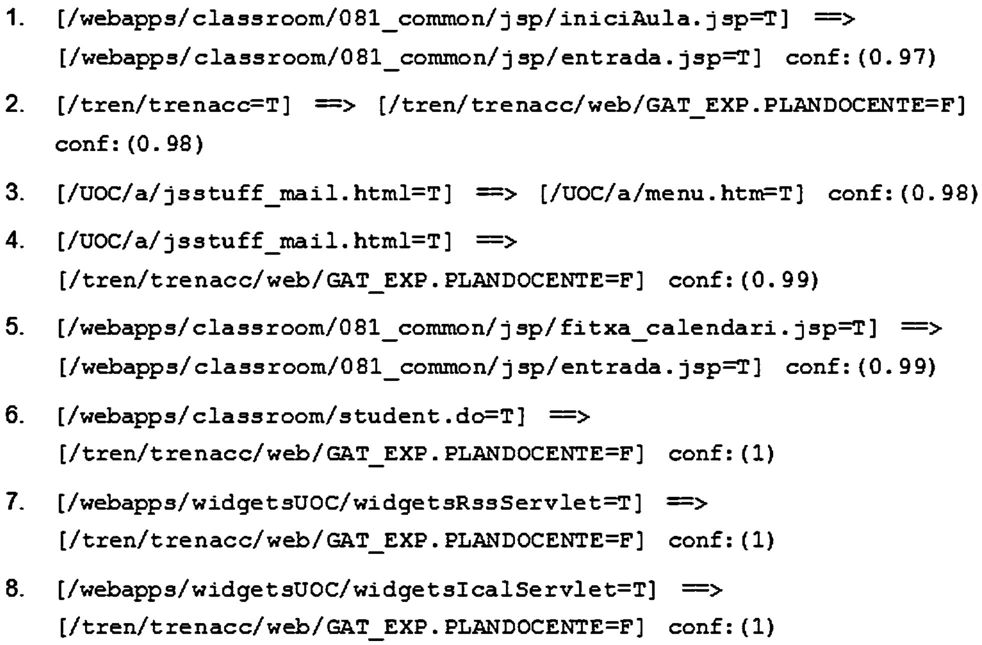
`findAllRulesForSupportLevel=True` and `maxNumberOfItems=2`.

While the parameter `findAllRulesForSupportLevel` was changed in order to obtain all the possible rules for the level of support set, the parameter `maxNumberOfItems` was changed to achieve the opposite effect, restricting the number of rules to those formed by a maximum of two elements.

Some of the most interesting rules found are shown in Fig. 12.

Analysis of the results Analyzing each of the obtained association rules, the following are en excerpt of navigation patterns of students' sessions extracted from the Virtual Campus:

1. The students that have accessed to a classroom have also accessed to classrooms spaces.
2. The students that have accessed to mailbox have not accessed to teaching plan.
3. The students that have accessed to teaching activities, have navigated (passed) through classrooms.



```
1. [/webapps/classroom/081_common/jsp/iniciAula.jsp=T] ==>
   [/webapps/classroom/081_common/jsp/entrada.jsp=T] conf:(0.97)
2. [/tren/trenacc=T] ==> [/tren/trenacc/web/GAT_EXP.PLANDOCENTE=F]
   conf:(0.98)
3. [/UOC/a/jsstuff_mail.html=T] ==> [/UOC/a/menu.htm=T] conf:(0.98)
4. [/UOC/a/jsstuff_mail.html=T] ==>
   [/tren/trenacc/web/GAT_EXP.PLANDOCENTE=F] conf:(0.99)
5. [/webapps/classroom/081_common/jsp/fitxa_calendari.jsp=T] ==>
   [/webapps/classroom/081_common/jsp/entrada.jsp=T] conf:(0.99)
6. [/webapps/classroom/student.do=T] ==>
   [/tren/trenacc/web/GAT_EXP.PLANDOCENTE=F] conf:(1)
7. [/webapps/widgetsUOC/widgetsRssServlet=T] ==>
   [/tren/trenacc/web/GAT_EXP.PLANDOCENTE=F] conf:(1)
8. [/webapps/widgetsUOC/widgetsIcalServlet=T] ==>
   [/tren/trenacc/web/GAT_EXP.PLANDOCENTE=F] conf:(1)
```

Fig. 12 Some rules found by FPGrowth of WEKA

4. The students that have accessed to classroom participant list, have not accessed to teaching plan.
5. The students that have accessed to campus RSS, have not accessed to teaching plan.
6. The students that have accessed to teaching calendar, have not accessed to teaching plan.

5 Conclusions and future work

In this paper we have presented an innovative approach for the building of learner models based on the identification of navigation patterns, allowing for adapting the system's usability to the actual learners' needs resulting in a great stimulation of the learning experience. This approach is based on a massive pre-processing of log files of a Virtual Campus, which generate huge amounts of information of great variety of type and formats. The massive processing is implemented following the master-slave paradigm and evaluated using different distributed infrastructures (Cluster and PlanetLab platforms). The study showed the need to carefully tune the application in terms of chunk data size to be processed at slave nodes as well as the bottleneck in processing in truly geographically distributed infrastructures due to the overhead caused by the communication time among the master and slave nodes.

We show then how to extract knowledge from the analysis of the resulting log data from the Virtual Campus pre-processed and stored in a well-structured format ready for data mining purposes aiming to identify the navigation patterns of online users. To this end, we used WEKA framework for mining the data and more specifically, we used the K-means, Apriori and FPGrowth algorithms implemented in WEKA. The proposed approach showed its usefulness in effectively identifying navigation patters of online users of the Virtual Campus.

Ongoing work is adding the Hadoop platform to our study of massive log processing with multiple distributed infrastructures. Apache Hadoop [11] is an open-source framework that allows for scalable, reliable, distributed processing of large data sets across clusters of computers using simple programming models. It implements a computational paradigm named MapReduce [12], where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster following the master-slave paradigm. Rather than rely on hardware to deliver high-availability, Hadoop is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. Our aim is to compare the log processing results obtained from the above distributed platforms with the results obtained by using Hadoop.

In our future work, we would like to compare the results of the data mining methods employed in this work with that of a bi-clustering algorithm initially studied in [23]. This approach is also useful to extract relevant knowledge about user activity for other purposes beyond navigation patterns, such as identifying activities performed by students as well as to study time parameters related to such activities.

References

1. Apache HTTP Server Project: http://httpd.apache.org/

2. Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Kerr, D., Sikkel, S., Trevor, J., Woetzel, G.: Basic support for cooperative work on the world wide web. Int. J. Hum.-Comput. Stud. **46**(6), 827–846 (1997)
3. Bushey, R., Mauney, J.M., Deelman, T.: The development of behavior-based user models for a computer system. In: Proc. of the 7th Intl. Conf. on User Modeling (UM 99), pp. 109–118. Springer, Berlin (1999)
4. Caballé, S., Daradoumis, T., Xhafa, F., Conesa, J.: Enhancing knowledge management in online collaborative learning. Int. J. Softw. Eng. Knowl. Eng. **20**(4), 485–497 (2010)
5. Caballé, S., Xhafa, F., Fernández, R., Daradoumis, Th.: Efficient enabling of real time user modeling in on-line campus. In: Proc. of the User Modeling 2007, pp. 365–369. Springer, Berlin (2007)
6. Caballé, S., Paniagua, C., Xhafa, F., Daradoumis, Th.: A grid-aware implementation for providing effective feedback to on-line learning groups. In: Proceedings of the Second International Workshop on Grid Computing and Its Application to Data Analysis (GADA 2005), pp. 274–283. Springer, Berlin (2005)
7. Carbó, J.M., Mor, E., Minguillón, J.: User navigational behavior in e-learning virtual environments. In: The IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), pp. 243–249 (2005)
8. Apache Server Log Files. http://httpd.apache.org/docs/1.3/logs.html
9. Foster, I., Kesselman, C.: The Grid: Blueprint for a Future Computing Infrastructure, pp. 15–52. Morgan Kaufmann, San Francisco (1998)
10. Gaudioso, E., Boticario, J.G.: Towards web-based adaptive learning communities. In: Proceedings of Artificial Intelligence in Education. IOS Press, Sydney, Australia (2003)
11. Apache Hadoop. http://hadoop.apache.org/
12. Apache Opensource Hadoop Map/Reduce framework. http://wiki.apache.org/hadoop/ProjectDescription
13. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min. Knowl. Discov. **8**(1), 53–87 (2004)
14. Ciesielski, V., Lalani, A.: Data mining of web access logs from an academic web site. In: Abraham, A., Koppen, M., Franke, K. (eds.) Proceedings of the Third International Conference on Hybrid Intelligent Systems (HIS'03): Design and Application of Hybrid Intelligent Systems, December, pp. 1034–1043. IOS Press, Amsterdam (2003)
15. Bindu Madhuri, Ch., Anand Chandulal, J., Ramya, K., Phanidra, M.: Analysis of Users' Web Navigation Behavior using GRPA with Variable Length Markov Chains. International Journal of Data Mining & Knowledge Management Process (IJDKP) **1**(2) (2011)
16. Paniagua, C., Xhafa, F., Caballé, S., Daradoumis, T.: A grid prototype implementation for real time processing of group activity log data in collaborative applications. In: Proceedings of the 2005 PDPTA'05, Las Vegas, USA (2005)
17. Park, S., Suresh, N.C., Jeong, B.K.: Sequence-based clustering for web usage mining: a new experimental framework and ANN-enhanced K-means algorithm. Data Knowl. Eng. **65**(3), 512–543 (2008)
18. PlanetLab. http://www.planet-lab.org/
19. Open University of Catalonia. http://www.uoc.edu
20. Weka 3: Data mining software in Java. http://www.cs.waikato.ac.nz/ml/weka/
21. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann, Burlington (2011)
22. Xhafa, F., Caballé, S., Daradoumis, Th., Zhou, N.: A grid-based approach for processing group activity log files. In: Proceedings of the First International Workshop on Grid Computing and Its Application to Data Analysis (GADA 2004), pp. 175–186 (2004)
23. Xhafa, F., Caballé, S., Barolli, L., Molina, A., Miho, R.: Using bi-clustering algorithm for analyzing online users activity in a virtual campus. In: Proceedings of the INCoS 2010 International Conference on Intelligent Networking and Collaborative Systems, pp. 214–221 (2010)