Using a Grid Platform for Enabling Real Time User Modeling in On-line Campus

Santi Caballé[1], Fatos Xhafa[2], Thanasis Daradoumis[1], Joan Esteve[2], Leonard Barolli[3], Arjan Durresi[4]

[1]Open University of Catalonia, Department of Computer Science, Multimedia, and Telecommunication
Rbla. Poblenou, 156, 08018 Barcelona, Spain
{scaballe,adaradoumis}@uoc.edu
[2]Dept. of Languages and Informatics Systems, Polytechnic University of Catalonia
Jordi Girona 1-3, 08034 Barcelona, Spain
{fatos,jestever}@lsi.upc.edu
[3]Department of Information and Communication Engineering
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka 811-0295, Japan
barolli@fit.ac.jp
[4]Department of Computer Science
Louisiana State University
298 Coates Hall, Baton Rouge, LA 70803, USA
durresi@csc.lsu.edu

Abstract

User modelling in on-line distance learning is an important research field focusing on two important aspects: describing and predicting students' actions and intentions as well as adapting the learning process to students' features, habits, preferences, and so on. The aim is to greatly stimulate and improve the learning experience. Indeed, on the one hand, students' intentions may change during the realization of learning activities and thus their actions evolve accordingly as the learning process moves forward. On the other hand, adaptive systems can effectively plan and design appropriate learning tasks according to students' features, habits and interests with the aim of facilitating the achievement of the learning goal. In this context, user modelling implies a continuous processing and analysis of user interaction data during long-term learning activities, which produces large and considerably complex information. As a consequence, processing this information is costly and could require computational capacity beyond that of a single computer. In this paper, we show how a Grid approach can considerably decrease the processing time of log data of on-line distance educational web-based systems. Our prototype is based on the master-worker paradigm and is implemented using a peer-to-peer platform running on the Planetlab nodes. The results of our study show the feasibility of using Grid middleware to speed and scale up the processing of log data and thus achieve an efficient and dynamic user modeling in on-line distance learning.

1. Introduction

User modeling [1, 2] 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 and 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 [3].

Therefore, on the one hand, the information captured from tracking is used by a user modeling algorithm in order to predict future users' actions, intentions and so on. On the other hand, based on the knowledge acquired from the user model, an adaptive system can adjust and personalize the system to individual user characteristics, preferences and needs. Indeed, adaptive systems [2, 3] monitor the user model and automatically adjust the interface or content provided by the system to accommodate such user differences as well as changes in user skills, knowledge and preferences. Thus, for instance, constantly maintaining the user model allows developers to receive continuous and useful feedback about the system's usability and adapt the user interface design to the actual users' needs whilst they evolve as time goes by.

The ultimate aim of using user modeling and adaptive methods and techniques is to stimulate and improve the users' experience when interacting with the system [1].

Proceedings of the First International Conference
on Complex, Intelligent and Software Intensive Systems (CISIS'07)
0-7695-2823-6/07 $20.00 © 2007 IEEE

User modeling provides a set of well-established techniques. Some of these techniques are fairly intrusive, such as requiring users to explicitly provide information through questionnaires. Seamless alternatives include the tracking of user's actions by analyzing the interactions occurring in these applications. In this paper, we focus on Web-based applications that support on-line distance learning. 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 [2]. Indeed, the data 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. In addition, the building of learner models may help identify navigation patterns and adapt the system's usability to the actual learners' needs and thus stimulating the learning experience [3].

As a consequence of the complex processes involved in learning, we need to capture all and each type of possible data gathered in log files. However, the information generated in Web-based learning applications can be of a great variety of type and formats [5]. Moreover, these applications are characterized by a high degree of user-user and user-system interaction which stresses the amount of interaction data generated. Therefore, there is a strong need for powerful solutions that record the large volume of interaction data and can be used to perform an efficient interaction analysis and knowledge extraction.

In the literature, questions related to efficiently process the information obtained from learning activity have been, to the best of our knowledge, hardly investigated. Most of the existing approaches in the literature consider a sequential approach for the processing of log data and try to overcome the performance problem by: (i) processing for specific purpose (i.e. limiting the quantity of information needed for that purpose); (ii) processing of small data samples, usually for research and testing purposes. In addition, they do not address the issue of processing time requirements that might result from the huge amount of data that are to be processed, which is a common issue in web-based learning environments. Moreover, the need to make the analyzed information available in real time entails that we may come across with processing requirements beyond those of a single computer.

Grid [4] technology is increasingly being used to reduce the overall, censored time in processing data by offloading these computationally costly tasks from the computing elements running them onto the Grid. The concept of a computational Grid has emerged as a way of capturing the vision of a network computing system that provides broad access to massive computational resources.

Based on this vision, a preliminary study was conducted [5] to show that a Grid approach based on the Master-Worker (MW) paradigm [6] might increase the efficiency of processing a large amount of information from user activity log files. This allowed us to develop a real Grid-aware prototype that shows (i) how easily we are able to offload onto the grid the online processing of log data from the application, (ii) how a simple MW scheme suffices to achieve considerable speed-up, (iii) the gain provided by the Grid approach in terms of relative processing time and, (iv) the benefits of using the inherent parallel and scalable nature of Grid while the input log files are growing up in both number and large size. In order to show the feasibility of our approach, we use the log data from the internal campus of the Open University of Catalonia though our approach is generic and can be applied for reducing the processing time of log data from any web-based application in general.

The rest of the paper is organized as follows. In Section 2, the importance and problems of modeling the students' behavior in Web-based environments is shown through the case of the Open University of Catalonia. Section 3 presents a sequential application for processing offline the campus log data generated by real online learning activity. Section 4 introduces how the problem of processing this log data can be parallelized using the MW paradigm on a peer-to-peer platform called Juxta-CAT. Section 5 explains in detail our approach to parallelize and offload onto the Grid the sequential application. Finally, in Section 6, we present some computational results achieved and we end in Section 7 with some conclusions and outline ongoing and future work.

2. Modeling students' behavior in Web-based distance learning settings: the case of the Open University of Catalonia

Our real web-based learning context is the Open University of Catalonia (UOC) [7] which offers distance education through the Internet in different languages. As of this writing, about 40,000 students, lectures and tutors from everywhere participate in some of the 23 official degrees and other PhD and post-graduate programs resulting in more than 600 official courses. The campus is completely virtualized. It is made up of individual and community areas (e.g. personal electronic mailbox, virtual classrooms, digital

Proceedings of the First International Conference
on Complex, Intelligent and Software Intensive Systems (CISIS'07)
0-7695-2823-6/07 $20.00 © 2007 IEEE

library, on-line bars, virtual administration offices, etc.) through which users are continuously browsing in order to fully satisfy their learning, teaching, administrative and social needs.

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 our 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.

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, this information has to enter a process to be effectively collected, processed and analyzed. 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. As a matter of fact, the computational cost is the main obstacle to process this data in real time [5, 8, 9] and hence in real situations this processing tends to be done offline 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.

Next section presents a sequential approach of processing log data information from Web-based distance learning applications. Main problems and inconveniences will be arisen and discussed through a detailed analysis of the processing of log data coming from the campus activity of our university.

3. Processing log files from an on-line distance learning campus

The on-line Web-based campus of the UOC is made up of individual and community virtual areas such as mailbox, agenda, classrooms, library, secretary's office, and so on. 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 strong use of email service so as to communicate with other students and lecturers as part of their learning process.

All users' requests are chiefly processed by a collection of Apache [10] web servers as well as database servers and other secondary applications, all of which are providing service to the whole community and thus satisfying a large number of users. For load balance purposes, all HTTP traffic is smartly distributed among the different Apache web servers available and 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.

A typical daily log file size may be up to 10 GB. 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 1.8 GB of potentially useful information corresponding to 3,500,000 of log entries in average still remains [11].

Log file entries are structured following a type of format known as Common Log Format (CLF) [12] which is produced by most of web servers including

Apache and is fairly configurable. For the purpose of registering the campus activity, log files entries were set up with the purpose of capturing the following information: who performed a request (i.e. user's IP address along with a session key that uniquely identifies a user session); when the request was processed (i.e. timestamp); 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 where (i.e. an absolute URL containing the full path to the server application providing the service requested).

At this point, we point out some 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 non-trivial 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.

In order to deal with these inconvenients, we have developed a simple application in Java, called *UOCLogsProcessing* that processes log files of the UOC. In particular, this application runs offline on the same machine as the logging application server. As an input, it uses the daily log files as a result of merging obtained after merging those log files generated by the web servers so as to: (i) identify the log entries boundaries and extract the fields that make up each entry, (ii) capture the specific information contained in the fields about users, time, sessions, areas, etc., (iii) infer the missing information, (iv) map the information obtained to typed data structures, and (v) store these data structures in a persistent support.

4. Juxta-CAT: a JXTA-based Grid platform

In this section, we briefly introduce the main aspects of the grid platform, called Juxta-CAT [13], [14], which we have used for the processing of log files.

The Juxta-CAT platform has been developed using the JXTA [15] protocols and offers a shared Grid where client peers can submit their tasks in the form of java programs stored on signed jar files and are remotely solved on the nodes of the platform. The architecture of Juxta-CAT platform is made up of two types of peers: *common client peers* and *broker peers*. The former can create and submit their requests using a GUI-based application while the later are the administrators of the Grid, which are in charge of efficiently assigning client requests to the Grid nodes and notify the results to the owner's requests. To assure an efficient use of resources, brokers use an allocation algorithm, which can be viewed as a price-based economic model, to determine the best candidate node to process each new received request. The implementation and design of peers, groups, job and presence discovery, pipe-based messaging, etc. are developed using the latest updated JXTA libraries (currently release 2.3.7) and JDK 1.5 version.

The Juxta-CAT platform has been deployed in a large-scale, distributed and heterogeneous P2P network using nodes from PlanetLab [1] platform. Juxta-CAT Project and its official web site have been hosted in Java.NET community [14].

Juxta-Cat architecture. As mentioned above, Juxta-Cat is made up of common client peers and broker peers.

Client peers are the end users of the Juxta-CAT and are obtained by downloading and installing the application from the official page of Juxta-CAT. Once the machine is "converted" into a client peer, the user will connect to the peer-to-peer network and can submit execution requests to their peer group nodes. Also, client peers will be able to process received requests sent to them by other nodes through the brokering and notify them the result of the requests, once they are completed.

Broker peers are in charge of receiving and allocating the requests sent by clients of the peer group. Whenever a broker receives a request, it explores the state of the rest of nodes currently connected to the network, examining their working and connection statistics. Then, it uses this historical/statistical data to select, according to a price-based economic model, the best candidate peer for processing that request.

[1] http://www.planet-lab.org/. Current distribution of 714 nodes over 337 sites, as of November 13th, 2006. Polytechnic University of Catalonia has joined PlanetLab with several proper nodes.

5. Processing of log files in the Juxta-CAT platform

We explain now how is done the processing of log files in the Juxta-CAT platform. The implementation follows the well-known MW paradigm. We note first that the sequential java class *UOCLogsProcessing* encapsulates also functionalities to provide the division of the log file into as many equal parts as grid nodes will be used for processing them; these parts will be later on submitted for processing to the Juxta-CAT. The main steps that would follow the user (the master node) to process a log file in the Juxta-CAT are as follows:

1. [**Preparation phase**]: Provide the necessary information (to the Master) for the preparation of the requests to submit to the Juxta-CAT:
   a. Indicate the path to the log file and its name and the number of nodes participating in the processing. *UOCLogsProcessing* counts the total number of lines of the log file, totalNbLines, and knowing the number of grid nodes to be used, nbNodes, each node will read and process a totalNbLines/nbNodes of lines from the file.
   b. Indicate an FTP server, a user name and a password as well as a public address where the parts of the file will be uploaded. The implementation of FTP for Java, known as PureFTP, is included in the Jakarta Apache *commons-net-1.4.1.jar* library [16].
2. [**Master Loop**]: Repeat
   a. Read totalNbLines/nbNodes lines
   b. Upload the file to the indicated public address via FTP
   c. Create a request and submit it to Juxta-CAT

   Until the original log file has been completely scanned.
3. [**Juxta-cat processing**]:
   a. Each time a request is received by brokers of Juxta-CAT, it is assigned to a peer node of the platform.
   b. The peer node, upon receiving the request, reads according to the request'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.
   c. The peer node, once the processing of the request is done, sends back to the master node the content of the buffer.
4. [**Master's final phase**]: Receive messages from peers and append the new received resulting file to the final file containing the information extracted from the original log file.

The program *UOCLogsProcessing* is compiled in a unique java jar package, which includes the library developed by Jakarta Apache needed for the FTP transfer. The code has been optimized using Java Proguard 3.5 so that the final jar file size is 28.7 KB. We show in Figure 1 and 2 the submission of a request to Juxta-CAT and the state information once it is processed. Note that the user has to just provide the in formation needed in Step 1 (see above); the rest is automatically done by Juxta-CAT.

6. Experimental results

In this section we present the experimental results obtained for a test battery in order to measure the efficiency obtained by the grid processing. This battery test uses both large amounts of log information (i.e. daily log files) and well-stratified short samples consisting of representative daily periods with different activity degrees (e.g. from 7 p.m. to 1 a.m. as the most active lecturing period and from 1 a.m. to 7 a.m. as the period with least activity in the campus). On the other hand, other tests involved a few log files with selected file size forming a sample of each representative stratum. This allowed us to obtain reliable statistical results using an input data size easy to use.

Table 1. PlanetLab nodes.

Host	Description
planet1.manchester.ac.uk	University of Manchester
lsirextpc01.epfl.ch	École Fédérale de Lausanne
planetlab1.polito.it	Politecnico di Torino
planetlab1.info.ucl.ac.be	University of Louivain
planetlab2.upc.es	Universitat Politècnica de Catalunya
planetlab1.sics.se	Swedish Institute of Computer Sci.
planetlab1.ifi.uio.no	University of Oslo
planetlab3.upc.es	Universitat Politècnica de Catalunya
planetlab1.ls.fi.upm.es	Universidad Politécnica de Madrid
planetlab1.hiit.fi	Technology Institute of Helsinki
planetlab-1.cs.ucy.ac.cy	University of Cyprus
planetlab1.ru.is	University of Reykjavik
planetlab2.sics.se	Swedish Institute of Computer Sci.
planetlab1.mini.pw.edu.pl	Telekomunikacja Polska Warsaw
planetlab1.cs.uit.no	University of Tromso
planetlab-02.ipv6.lip6.fr	Laboratoire d'Informatique de Paris

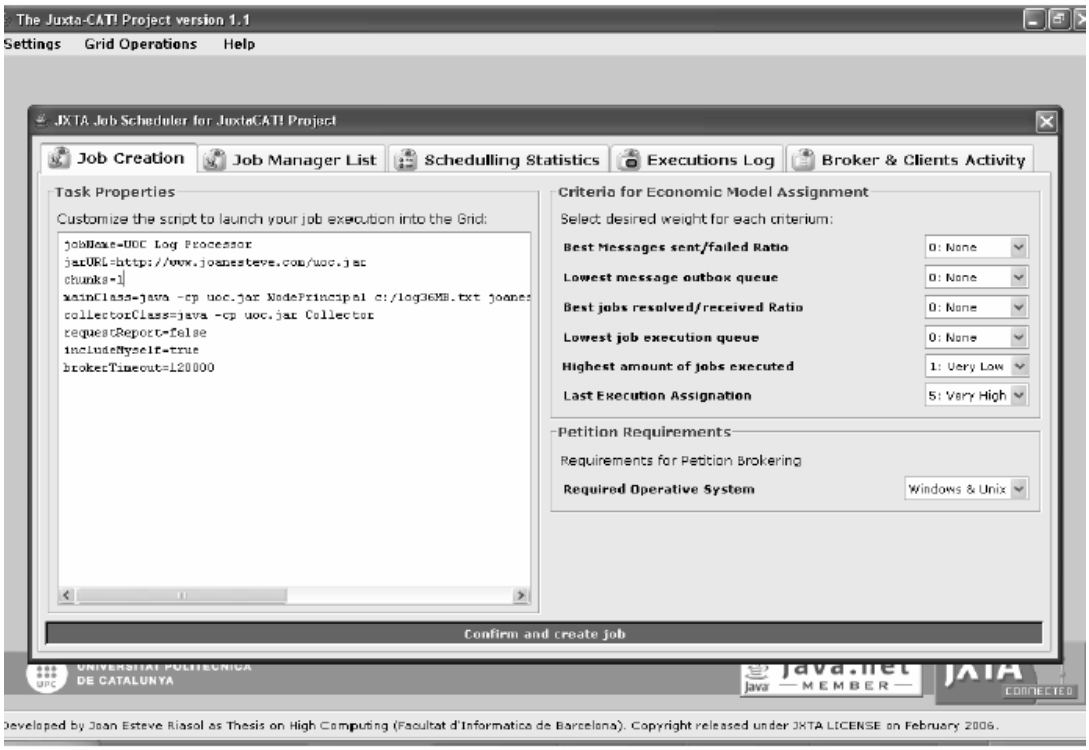

Fig. 1. GUI view of submitting the processing of a log file to the Juxta-CAT.

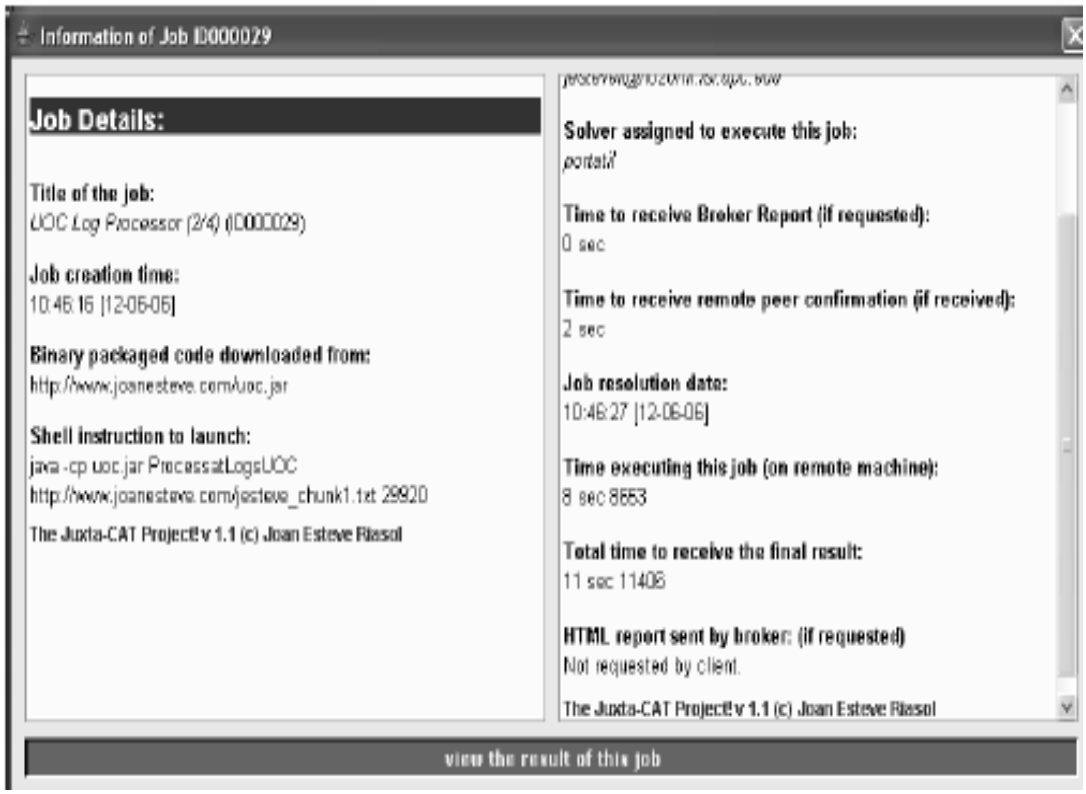

Fig. 2. State information of a request once it is processed.

Proceedings of the First International Conference
on Complex, Intelligent and Software Intensive Systems (CISIS'07)
0-7695-2823-6/07 $20.00 © 2007 IEEE

The battery test was processed by the *UOCLogsProcessing* application executed on single-processor machines involving usual configurations. The battery test was executed several times with different workload in order to have more reliable results in statistical terms involving file size, number of log entries processed and execution time along with other basic statistics. On the other hand, the same battery test was processed by Juxta-CAT with different number of nodes, specifically, 2, 4, 8, and 16 nodes, using PlanetLab nodes (see Table 1).

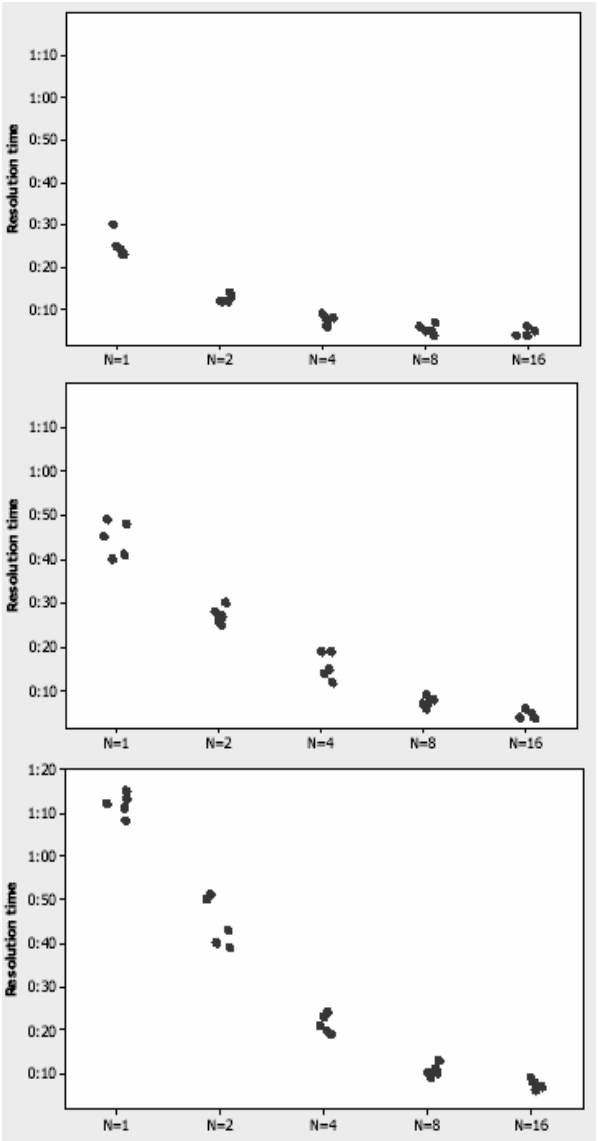

Fig. 3. Three execution times results for log files with sizes of 12MB, 24MB and 36MB, respectively; x-axis indicates the number of processors and y-axis the processing time (mm:ss).

Parallel speed-up is used to measure the performance gain from a parallelized execution of the application over its serial execution, defined as follows:

$$S(s,p) = T_S(s) / T_P(s,p),$$

where s is the size of the log file, $T_S(s)$ is the total running time of the sequential execution for a log file of size s and $T_P(s,p)$ is the total running time of the parallel execution for a log file of size s, using p processors.

Parallel efficiency measures the degree of utilization of the computing resources involved in the parallel computation and is defined as the speed up divided by the number of computing resources (i.e. processors):

$$E(s,p) = S(s) / p.$$

From the execution times shown in Figure 3 and the formulas previously introduced, we show in Table 2 the gain in terms of parallel speed-up and efficiency we achieved.

Table 2. Parallel speed-up and efficiency.

Log file size	Speed-up	Efficiency
12 MB	6.1	38.2 %
24 MB	7.4	46,2 %
36 MB	9.1	56.8 %

7. Conclusions and further work

Web-based applications that support on-line distance education have been gaining a lot of attention due to the capability of offering training, long-life learning and education in general widely and easily available. In this context, it is essential to capture and understand the learner's behavior so as to predict future intentions, provide appropriate support and adapt the learning process and environment to learners' needs, preferences, knowledge, skills, and so on. The aim is to greatly stimulate the learning experience. To this end, we have shown how to model the learner's behavior and activity pattern by using user modeling tracking-based techniques.

However, the information generated from tracking the learners when interacting with the virtual learning environment is usually very large, tedious, redundant and ill-formatted and as a result processing this information is time-consuming. In order to overcome this problem, in this paper we have proposed a Grid-aware implementation that considerably reduces the processing time of log data and allow to build and

constantly maintain user models. For the purposes of both showing the problem of dealing with log data and testing our grid prototype we have described and used the log data coming from the virtual campus of the Open University of Catalonia.

The experimental results show a considerable gain in speedup and efficiency while parallelizing the processing of log files. This makes us confident of the feasibility and usefulness of using our Grid approach.

Further work will include the implementation of a more thorough mining process of the log files, which due to the nature of the log files of our virtual campus will require more processing time as compared to the *UOCLogsProcessessing* used in this work.

Acknowledgments

This work has been partially supported by the Spanish MCYT project TSI2005-08225-C07-05.

8. REFERENCES

1. Bushey, R., Mauney, JM., and Deelman, T., The Development of Behaviour-Based User Models for a Computer System. In Judy Kay (ed.), User Modeling: Proceedings of the Seventh International Conference, UM99. Springer Wien New York, pp. 109-118, 1999.
2. Brusilovsky, P. and Peylo, C., Adaptive and intelligent Web-based educational systems. In P. Brusilovsky and C. Peylo (eds.), International Journal of Artificial Intelligence in Education 13 (2-4), Special Issue on Adaptive and Intelligent Web-based Educational Systems, pp. 159-172, 2003.
3. Gaudioso, E., Boticario, J.G., Towards web-based adaptive learning communities. Proceedings of Artificial Intelligence in Education 2003, Sydney, Australia. IOS Press, 2003.
4. Foster, I. and Kesselman, C., The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann, San Francisco, CA, pp. 15-52, 1998.
5. Xhafa, F., Caballé, S., Daradoumis, Th. and Zhou, N., A Grid-Based Approach for Processing Group Activity Log Files. In: proc. of the GADA'04, Cyprus, 2004.
6. Master-Worker: http://www.cs.wisc.edu/condor/mw/, Web page as of October 2006.
7. Open University of Catalonia: http://www.uoc.edu, Web page as of October 2006.
8. Paniagua, C., Xhafa, F., Caballé, S. and Daradoumis, T., A Grid Prototype Implementation for Real Time Processing of Group Activity Log Data in Collaborative Applications. In: Proc. of the 2005 PDPTA'05. Las Vegas. USA, 2005
9. Caballé, S., Paniagua, C., Xhafa, F., and Daradoumis, Th., A Grid-aware Implementation for Providing Effective Feedback to On-line Learning Groups. In: proc. of the GADA'05, Cyprus, 2005.
10. Apache HTTP Server Project: http://httpd.apache.org/, Web page as of June 2006.
11. Carbó, JM., Mor, E., Minguillón, J., User Navigational Behavior in e-Learning Virtual Environments. The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), pp. 243-249, 2005.
12. Common Log Format: http://httpd.apache.org/docs/1.3/logs.html#common.
13. Esteve J., Xhafa F., Juxta-CAT: A JXTA-based Platform for Distributed Computing. The ACM International Conference on Principles and Practice of Programming in Java (PPPJ 2006), 2006.
14. Juxta-Cat: https://juxtacat.dev.java.net/, Web page as of October 2006.
15. JXTA: http://www.jxta.org/, Web page as of October 2006.
16. Jakarta Project: http://jakarta.apache.org/, Web page as of October 2006.