Optimizing the Access Records of Students in the Moodle Virtual Learning Environment Database

Eliakim Zacarias¹, Luciano R. de Almeida¹, Juliano B. Prettz¹, João Paulo C. L. da Costa¹², Edison P. de Freitas³, Edna D. Canedo¹, and Rafael T. de Sousa Júnior¹

¹Department of Electrical Engineering, University of Brasília, Brazil
²Institute for Information Technology, Ilmenau University of Technology, Ilmenau, Germany
³Fraunhofer Institute for Integrated Circuits IIS, Erlangen, Germany

Abstract: Moodle is a freeware solution for Learning Management System (LMS) applied to Virtual Learning Environments (VLS). Despite the several features to support online courses, some desirable information is not easily provided, for instance, the amount of time the user spent in the system or in a specific course. A custom implementation can measure the time spent on the user’s searches into the internal Moodle log and the distance between the records through the session time. However, for a large number of recorded logs, searching a single record passing through the other records requires a large time processing. As a consequence, the generation of custom reports is also costly. This work proposes a real-time plugin for Moodle to count the time spent by a student logged into the system retrieving the information from a pre-computed central table. The objective of the solution is to improve the counting precision and allow a faster retrieval of data. Our proposed plugin is validated using real data from the National School of Public Administration (Enap) in Brazil.

© 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Generation of Reports, Moodle Plugin, Time Counting Precision, Measuring Online Time.

1. INTRODUCTION

Although Moodle [1] is a popular platform for distance learning environments, there are still essential limitations such as the lack of tracking tools for the students. However, these limitations can be overcome by adding third-party solutions such as plug-ins or even a whole separate system [2].

The student tracking data is crucial to generate reports monitoring the student's behavior in the offered courses. Moodle has available the user activity log, which can be used to control and to capture relevant tracking information.

The National School of Public Administration, in Portuguese Escola Nacional de Administração Pública (Enap), provides distance learning courses for the federal employees in Brazil using Moodle. In order to support the students to improve their performance, Enap used an ad hoc solution that seeks all access records of selected students in a certain course. Search loops were repeated for all students passing through the activity log data by filtering course from the course start date until the end date. Despite the feasibility in small databases, such approach is unfeasible if the activity log increases. Therefore, the total query time was proportional to the amount of selected students.

This paper aims to provide a solution to the problem by standardizing data and separating them from the activity logs so the query can be made by consulting a single record.

This paper is divided into four sections besides this introduction. Section 2 presents and discusses the state of the art related to the subject. Section 3 presents the proposed approach to address the problem, while Section 4 presents the acquired results. Section 5 concludes the paper and also provides directions for future work.

2. RELATED WORKS

Moodle [1] is one of the leading platforms for free distance learning. It is an open source system that includes the participation of collaborative communities for their development. Its structure is formed by modules or extensions, smaller systems that provide specific or specialized features. These modules are also called plug-ins [2].

NAGI and SUESAWALUK [3] present an analysis based on reports of Moodle to measure the level of interactivity in the virtual environment of teaching courses at the University of the Assumption. The aim of [3] is the use of data captured behavior and iteration of students in the Moodle environment. These pieces of information are collected in real-time, including information that is not registered in the database, like views and clicks. One of the conclusions in [3] is that a communication platform to improve the interaction between students and instructors can be created and a ranking of the courses with more iterations and collaborations of students can be also made.
In BOVO, 2013 [4] an analysis of the log of the virtual system Moodle learning is made. With data mining techniques, machine learning and artificial intelligence indicates which groups are the most relevant to form support tools for managers and teachers. Such data mining techniques can also predict the behavior of the students and their performance. It can also allow the teachers and mentors trace paths and counseling so that students get better results.

In [5], managers and teachers needed the complete information on the behavior of students, in order to analyze and propose ways to improve student achievement. To meet this demand the Merlin was developed, composed of three layers. The first layer contains the data of the education system. The second layer consists of tools that analyze and form a database of system logs, cookies, logins on the system and polls of answers, all with the function to track and record the actions of students during activities in the virtual system. The third layer is the presentation of reports based on pooled data from data mining tools.

3. THE PROPOSED METHOD

The proposal of this paper seeks to avoid dependence on activity logs and centralize the storage of already processed data. Thus, the search of the required information is held in a separate table, which links the students and their total active time.

Since the solution does not require a history of accesses, but only the final total time the user remained active until completion of the course, the system keeps only the total sum of minutes. Thus, the user time registration is required, along with another record connecting the user and the course, storing the sum of this.

Considering the impossibility to increase the server's hardware capabilities for this case, and also the impossibility to wait for the full-time report generation that could exceed 10 minutes, the solution aims to replace the intensive processing load during the report generation through periodic requests. At the end of each minute, an asynchronous request is sent informing that the user remains accessing the system. Such a request is allowed through technology for Asynchronous Javascript and XML (AJAX), implemented in browsers as the XMLHttpRequest class, in the first level of development [6].

The AJAX technology is possible and accessible to browsers that support the Moodle system since version 2.3. Since Moodle itself requires JavaScript for its better functioning, the solution does not interfere with the usability and requirements, taking advantage of resources already offered.

To implement the solution, two parts were needed. A Javascript is added to a standard content to all pages, usually to the theme customization files. The code creates an event that happens every minute, which makes a call to a specific page.

As shown in Figure 1, the Javascript code includes the header or footer of pages of Moodle, to print and run on all system pages. Inclusion is usually performed in the theme customization files. Then, an event is initiated and repeated at a predefined interval as the minimum time that should be considered for user access, also in view of each iteration, a request to the server is made. The code makes the request to a specific page.

![Diagram of the browser part of the plugin.](image1)

Fig. 1. Browser process: Diagram of the browser part of the plugin.

Figure 2 shows the process triggered in the server for each request so that the server is responsible for storage of the student's access time. Every time the server is requested, the records belonging to the user and the full system access are rescued. Next the minimum required time since the last update is checked, and if it is higher, the past period is added.

![Diagram of the server part of the plugin.](image2)

Fig. 2. Server process: Diagram of the server part of the plugin.
Note that this minimum acceptable period between a request and another is not exactly a minute. A shorter period should be defined. Thus, when an update delays, the next call is not below this period. Otherwise, all requests below the period should be ignored.

Limiting the time between calls is necessary to avoid that when a user opens more than once the page, in tabs, for example, the total amount of access time is not multiplied by the number of open pages. Moreover, this condition also prevents an application for false claims, to explain more minutes than the actual access time.

If the server cannot retrieve the user time information due to the lack of prior registration, instead of updating the registry incrementing the old value, a new record is inserted.

When the user closes all instances of Moodle system in the browser, the JavaScript is also interrupted, causing disruption of recurring requests and stop increasing the time.

The solution allows a greater precision than searching activity logs, for the considered period of time can be variable and depends not on the session length. It is also more accurate because of the format proposed to consider the times when the user has the page open without interacting with the system by clicking links, while performing tasks such as reading and even watching an embedded video content.

Thus, by this method, it is possible to obtain a ready to store information relating to the time the student accessed the system without the need for additional processing or requiring prolonged consultations relate many records.

4. RESULTS

The validation of the proposed approach was possible thanks to ENAP, by using a clone of the organization’s production database with real data access. For performance testing purposes, it was chosen the course with more user activity logs, offered and concluded in the second half of 2015. Due to the number of records, courses with similar size were prevented from generating any reports that would require knowing the time students remained with the open system. For validation, it was obtained and compared the query time in the database through both solutions.

Table 1 shows a comparison of the database query time to retrieve all the information from all students who have accessed a particular course from the log.

<table>
<thead>
<tr>
<th>Table 1. Query time comparison selecting all users at once</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current solution</td>
</tr>
<tr>
<td>Time (milliseconds)</td>
</tr>
<tr>
<td>Row count</td>
</tr>
</tbody>
</table>

The current implementation calculates the time between records for the total time the user was active in the system. Thus, this requires that all data belonging to an ongoing activity to be retrieved from the database for analysis. The number of lines (rows) obtained concerns to the number of activities within the course, including the activities of all users.

The proposed solution, however, adds the user access time while it is still connected to the system, and stores the sum in a single record linked to the user and the course. Thus, to meet the total user time on a course, the system checks the records belonging to him. Table 1 is obtained 7059 records, which is the number of users who accessed the tested course.

Note that the current query implementation and returns all the data associated with user activity. It includes but is not limited to, source access, IP address that caused the access, the exact time of the activity access, the accessed activity. However, it became clear that the implementation did not have any interest in these data, but only the time and the user identifier numbers were used.

In view of this, it was decided to optimize the query specifying which fields the database should return and run the tests again.

Table 2 shows the difference in time between queries. By selecting which fields are retrieved, ignoring unnecessary fields for processing, one obtains a reduction of up to 59 % in total query time, about 7 seconds to the current implementation. However, the same variation does not occur with the proposed solution, given the smaller amount of records needed.

<table>
<thead>
<tr>
<th>Table 2. Query time comparison selecting specific columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current solution</td>
</tr>
<tr>
<td>Time selecting all fields (milliseconds)</td>
</tr>
<tr>
<td>Time selecting specific fields (milliseconds)</td>
</tr>
<tr>
<td>Row count</td>
</tr>
</tbody>
</table>

Another problem that arises to recover all data in a single request relates to the use of the virtual memory of PHP. Since the transformation of the records taken from the result of a query, which is converted into PHP objects, this also causes higher memory consumption by obtained record. Thus, when a query returns many records, given the processing of data, the virtual memory limit available for the process is reached stopping the request [7].

To solve this problem, in the current implementation it was chosen to select records from a single user to query and repeat the query for each user of the course.

Query performed on Table 3 shows a comparison between this method and the current implementation of the proposed solution for a single user. Based on this fact, it was possible...
to realize that retrieving data from a single user is feasible in both solutions. However when interacting through each user of a course is necessary, the process may become much slower. The course tested exceeds 7 thousand students, so that, if multiplied by the number of students query time, the entire process takes over 3500 seconds, about 1 hour. The proposed solution, however, is able to perform the procedure for 7 thousand students in approximately 5 minutes.

<table>
<thead>
<tr>
<th>Table 3. Query time comparison per user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current solution</td>
</tr>
<tr>
<td>Time (milliseconds)</td>
</tr>
<tr>
<td>Row count</td>
</tr>
</tbody>
</table>

However, comparing Table 3 and Table 2, it is noted that by using the proposed solution, in a search for each student is not feasible. Moreover, it is not necessary for the proposed solution, since it is possible to recover all data in a single request without causing the total consumption of virtual memory, once the data amount is much smaller.

The proposed solution solves the problem of knowing the time the student spent accessing the system at any given course, as needed at ENAP. The solution proposes a new implementation that allows the generation of information in a viable way.

The solution without additional changes cannot provide more accurate data on the activities that the user accessed or even consult for periods of time within the course. However, it can be adapted to obtain such specific information as needed, or for very specific cases, the user activity log is still available and can be consulted.

5. CONCLUSIONS AND FUTURE WORK

Many current systems collect their data directly from the Moodle activity logs. Such approach allows normalizing the information and making faster searches and improvements of accuracy by storing other desirable information. Since Moodle is a fairly permissive tool for changes and implementation of new features, it is possible to add processes without direct changes in the core.

This work reports the development of a Moodle plugin to count the time the students spend in the system. The goal was to develop a tool that could efficiently perform this task, improving the counting precision and allowing a faster retrieval of data. The acquired results showed that the proposed solution was able to reduce the processing load and at the same time prevent the overhead do the virtual memory management. These results are evidence that the proposed solution fulfills the demanded efficiency requirements, drastically diminishing the queries processing time.

As future work it is possible to change the solution so that it can record information that relates the student to the course and the time spent in performing a specific activity, such as reading a text, watching a video, answering a questionnaire or joining a forum.

Also, it is possible to optimize the solution to differentiate or just record the time that the user spent interacting with the system, by observing the visibility of the page, mouse movements, keyboard input, or even video player events.

ACKNOWLEDGEMENTS

The authors wish to thank the Brazilian Public Administration School ENAP (Grant TED 083/2016) for their support to this work.

REFERENCES


