Network Traffic Monitoring for Improving Reliability and Time Response

DUMITRU DAN BURDESCU, MARIAN CRISTIAN MIHĂESCU
Software Engineering Department
University of Craiova
Bvd. Decebal, Nr. 107, 200440, Craiova, Dolj
ROMANIA
http://software.ucv.ro/~burdescu_dumitru/
http://software.ucv.ro/~mihaescu_cristian/

Abstract: - The paper addresses the problem of network traffic monitoring and improving reliability and response times for an e-Learning platform. The activities that are performed within Tesys [1] e-Learning platform require sometimes heavy data traffic between platform itself and users. That is why, within the platform there was implemented a data traffic monitoring mechanism at byte level and at activity level. The levels of data traffic and performed activities are monitored and constitute the raw data within the analysis process. The analysis process uses state of the art machine learning algorithms and dynamic data structures in order to produce knowledge. This expertise is further used within the platform to cache delivered data, using the principle of temporal locality of data.
Experiments showed the factors that influence the response times are: granularity of traffic monitoring, employed learning algorithm and the data structure used for caching. The results are promising in the way that after prototype implementation of analysis process in the form of an Expertise Module, the response time decreased. The main advantage of Expertise Module is that it virtually introduces insignificant delays since all analysis is performed off-line..

Key-Words: - data traffic monitoring, machine learning, e-Learning

1 Introduction

There was designed and developed an e-Learning platform called Tesys [1]. This platform has implemented facilities for following type of users: system administrators, secretaries, professors and students. Some activities implemented for students, like downloading course materials or taking tests or exams are sometimes very heavy regarding the computational load of the server and the data traffic transfer to and from the user.
This paper presents the structure and functionality of an Expertise Module (EM) that runs along the Tesys e-Learning platform. The main purpose the EM is to decrease the response time of the platform at user's requests. The functionality of the EM module is presented in Figure 1.

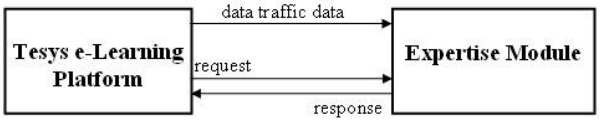

Figure 1. General functionality of Expertise Module

As presented in Figure 1 the input of EM is represented by data traffic data. The data are obtained by a custom implemented logging mechanism embedded within the platform's business logic. The platform is represented by the setup put in place in order to perform all necessary activities within the e-Learning process. The setup consists of course materials, test and exam quizzes that are set up by course managers and the overall setup performed by secretaries.
The data traffic is saved into structured format and after that it is fed to the Expertise Module. Once the EM is initialized, it can provide data back to Tesys e-Learning platform in the form of a response to specific requests. Within EM, there are two problems that are addressed.
One refers to employed machine learning algorithms. This problem is considered from two points of view. One is from the point of view of the general architecture of EM and the other is from the point of view of analysis process itself. The second problem refers to the architecture of EM as a service. Once the EM has been instantiated it will work as a service for Tesys e-Learning platform.
The activity of a student is seen as a sequence of sessions. A session starts when the student logs in and finishes when the student logs out. A session is

represented by a sequence of actions. The next figure presents the activity diagram from platform

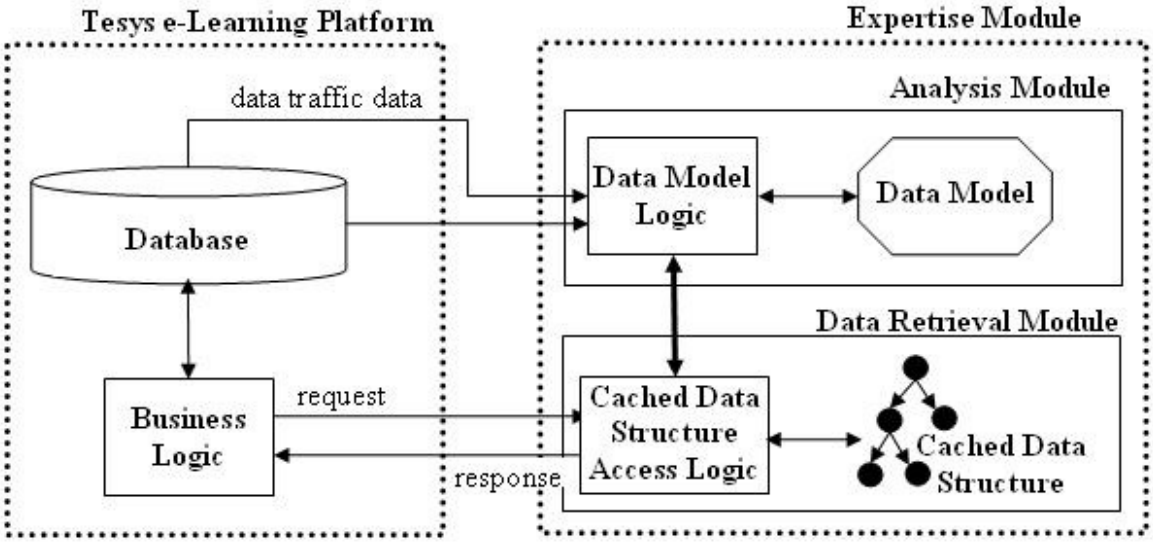

Figure 2. Detailed functionality of Expertise Module and integration with Tesys e-Learning platform

point of view. Within the platform each student has an associated activity diagram.

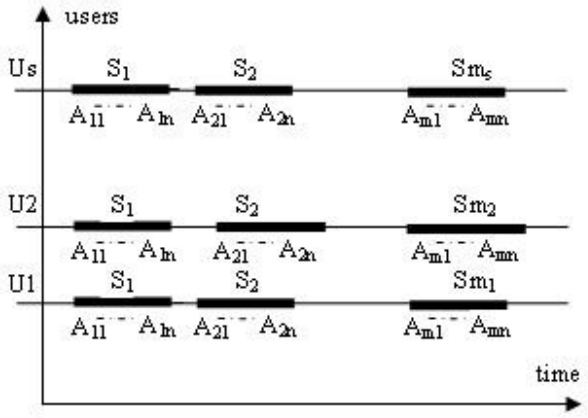

Figure 3. Activity diagram for students

In the diagram it may be seen the activity of all s users (U1,, U2, …, Us). The activity of each user is composed of a number of sessions. User Us in the diagram has ms associated sessions. At finest level, a session is composed of a number of actions, session Sms has mn associated actions. In a session, the first action is to login and the last one is logout. After one hour of inactivity the user is automatically logged out such that user sessions can be precisely determined. The notion of "user session" was defined as being a temporally compact sequences of Web accesses by a user. A new distance measure between two Web sessions that captures the organization of a Web site was also defined. The goal of Web mining is to characterize these sessions. In this light, Web mining can be viewed as a special

case of the more general problem of knowledge discovery in databases.

Still, in performed experiments there were taken into consideration heavy traffic sessions. In this light, a session becomes more or less a time interval in which a user performs many requests as a part of a certain e-Learning activity. For example, taking a test is a heavy traffic session. Within a session there may be many heavy traffic sessions. For such sessions there was monitored the quantity of transferred data, the duration of the session and other information regarding that session like the user who performed the actions and the accessed resources. This data represent the raw data that is used as input in the analysis process.

Under these circumstances, a more detailed functionality diagram of Expertise Module is presented in Figure 2. The EM is basically an application that functions as a service for Tesys e-Learning platform. Its main goal is to decrease the response time of the platform by caching data in an intelligent way. The intelligent feature of caching is given by the mechanisms implemented within the EM: Analysis Module and Data retrieval Module. These modules have specific architectures according to implemented functionality.

The Analysis Module has as input the data traffic data and the data from the database. All these data is processed by the Data Model Logic such that a data model is created. Once the data model has enough accuracy, necessary data is passed to Data Retrieval Module. This module is the one that manages the

cached data. The business logic is represented by Cached Data Structure Access Logic and the data themselves are stored in the Cached Data Structure. Besides managing the Cached Data Structure, the logic implemented in Data Retrieval Module is also responsible for interfacing with the business logic of the platform. The communication is accomplished in a client-server architecture. When the business logic of Tesys needs specific data it firstly sends a request to Expertise Module and more exactly to business logic from the Data Retrieval Module. If requested data is not found than classical way of obtaining it is used.

The Analysis Module implements the classical steps of target modeling presented in figure 4. Within this module the most important step is the processing of the model. In this implementation the machine learning algorithm used for processing the model is the Decision Tree Induction algorithm.

The Data Retrieval Module implements a dynamical data structure used for managing in main memory the data that is available for deployment through the Tesys e-Learning Platform. For implementing this structure there are used AVL trees [5,9]. A AVL tree is a self-balancing binary search tree. In an AVL tree the heights of the two child subtrees of any node differ by at most one, therefore it is also called height-balanced. Lookup, insertion, and deletion all take O(log n) time in both the average and worst cases. Additions and deletions may require the tree to be rebalanced by one or more tree rotations.

2 Tesys e-Learning Platform

The main goal of the application is to give students the possibility to download course materials, take tests or sustain final examinations and communicate with all involved parties. To accomplish this, four different roles were defined for the platform: sysadmin, secretary, professor and student.

The main task of sysadmin users is to manage secretaries. A sysadmin user may add or delete secretaries, or change their password. He may also view the actions performed by all other users of the platform. All actions performed by users are logged. In this way the sysadmin may check the activity that takes place on the application. The logging facility has some benefits. An audit may be performed for the application with the logs as witness. Security breaches may also be discovered.

2.1 Main Functionalities and Architecture

Secretary users manage sections, professors, disciplines and students. On any of these a secretary may perform actions like add, delete or update. These actions will finally set up the application such that professors and students may use it. As conclusion, the secretary manages a list of sections, a list of professors and a list of students. Each discipline is assigned to a section and has as attributes a name, a short name, the year of study and semester when it is studied and the list of professors that teach the discipline which may be maximum three. A student may be enrolled to one or more sections.

The main task of a professor is to manage the assigned disciplines while s discipline is made up of chapters. The professor sets up chapters by specifying the name and the course document. Only students enrolled in a section in which a discipline is studied may download the course document and take tests or examinations. Besides setting up the course document for each chapter, the professor manages test and exam questions.

Tesys application offers students the possibility to download course materials, take tests and exams and communicate with other involved parties like professors and secretaries.

Students may download only course materials for the disciplines that belong to sections where they are enrolled. They can take tests and exams with constraints that were set up by the secretary through the year structure facility.

Students have access to personal data and can modify it as needed. A feedback form is also available. It is composed of questions that check aspects regarding the usability, efficiency and productivity of the application with respect to the student's needs.

The e-learning platform consists of a framework on which a web application may be developed. On server side we choose only open source software that may run on almost all platforms. To achieve this goal Java related technologies are employed. In figure 2 we present the most general view of the software architecture from MVC point of view.

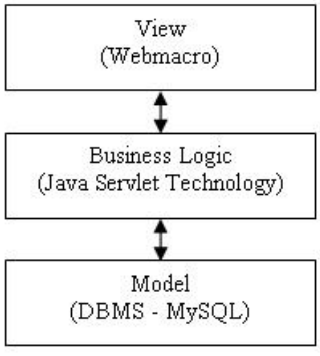

Figure 4. MVC architecture of the Tesys e-Learning platform

This architecture of the platform allows development of the e-learning application using MVC architecture. This three-tier model makes the software development process a little more complicated but the advantages of having a web application that produces web pages in a dynamic manner is a worthy accomplishment. The model is represented by DBMS (Data Base Management System) that in our case is represented by MySQL.

2.2 Data Traffic Logging Mechanisms

In this part we shall focus on describing the monitoring capabilities of the platform when running. The platform implements two ways of monitoring activity. Since business logic is implemented in Java, the log4j utility package was chosen to be used in order to log specific events. The next lines present how the utility was set up.

log4j.appender.R.File=D:/Tomcat/idd.log
log4j.appender.R.MaxFileSize=1000KB
log4j.appender.R.MaxBackupIndex=5

These lines state that all the logging process will be done in idd.log file and will have a maximum file size of 1000KB in maximum five files.

This utility package is also used in debugging process and the logs may be very useful in finding security breaches like unsuccessful attempts to log in or run actions that are not allowed.

The main disadvantage of this technique is the semi structured way in which information is stored. This makes the information retrieval and analysis to be not so easy.

The second method of monitoring user activity within the platform is through a database table called activity. In this table a record is added each time a user performs an action. In the next table it is presented the structure of activity table.

Field	Description
id	**primary key**
userid	**identifies the user who performed the action**
date	**stores the date when the action was performed**
action	**stores a tag that identifies the action**
details	**stores details about performed action**
level	**specifies the importance of the action**

Table 1. Structure of activity table

The details field stores specific information regarding the action that was executed. For example, if a secretary modifies the profile of a student in the details field there will be stored information about what fields were updated.

The level field specifies the importance of the executed action. There are defined three level of importance: 0,1 and 2 where level 0 specifies the critical actions.

So far, in activity table there are close to 40,000 recorded actions in almost four month of running the platform. At the end of the cycle there are expected almost 100,000 recorded actions.

3 The Expertise Module

3.1 The Analysis Module

The Analysis Module implements the classical steps of classical modeling presented in figure 5 [2]. Defining the goal represents the first step. Our goal is to create a model of analysis for data obtained from Tesys e-Learning platform that is to be finally used for minimizing response time. Setting up the goals is accomplished by formally defining the criteria that is to be evaluated and optimized. Selection and preparation of data are the next steps. Here, we have to determine the necessary data that will enter the modeling process. The preparation gets that data and puts it into a form ready for processing of the model. Since the processing is done using machine-learning algorithms implemented in Weka workbench [4], the output of preparation step is in the form of an arff file. The Data Model Logic module is responsible for querying the platform's database in order to create the input data file called activity.arff that is used for building the data model. This process is automated and is driven by a property file in which there is specified what data will lay in activity.arff file.

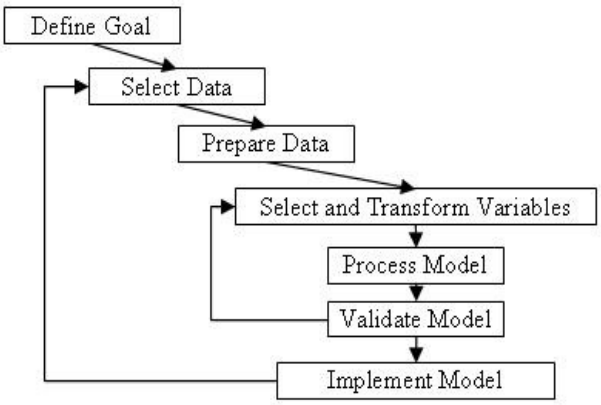

Figure 5. Steps for target modeling

Choosing between two learning algorithms given a single dataset is not a trivial task. Firstly, we make sure the data is relevant. We test the "goodness" of data trying to build a decision tree like C4.5 [6] from data. A decision tree is a flow-like-chart tree

structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test and leaf nodes represent classes [7].
The basic algorithm for decision tree induction is a greedy algorithm that constructs the decision tree in a top-down recursive divide-and-conquer manner [7].
The computational cost of building the tree is O(mn log n)[8]. It is assumed that for n instances the depth of the tree is in order of log n, which means the tree is not degenerated into few long branches.
The information gain measure is used to select the test attribute at each node in the tree. We refer to such a measure an attribute selection measure or a measure of goodness of split. The algorithm computes the information gain of each attribute. The attribute with the highest information gain is chosen as the test attribute for the given set [7].
For a learner in Tesys e-Learning platform we may have a very large number of attributes. Still, in our procedure we used only three: the number of loggings, the number of taken tests and the number of sent messages. Here is how the arff file looks like:

```
@relation activity
@attribute noOfLogins {<10,<50,<70,<100,>100}
@attribute noOfTests {<10,<20,<30,<50,>50}
@attribute noOfSentMessages {<10,<20,<30,<50,>50}
@attribute dataTraffic {<10,<20,>20}
@data
<50,<10,<10,<10
>100,<20,<20,<20
```

The first line in data section says that the learner logged in less than fifty times, took less than ten tests, sent less than ten messages to professors and had a data traffic less than 10MB.
Finally, the cross-validation evaluation technique measures the correctly and incorrectly classified instances. We consider that if there are more than 80% of instances correctly classified than we have enough good data. The obtained model is further used for analyzing learner's goals and obtain recommendations. The aim of the QM is to "guide" the learner on the correct path in the decision tree such that he reaches the desired class.

3.2 The Data Retrieval Module

The Data Retrieval Module is the one responsible for managing the data that is to be retrieved to Tesys platform at request. In this prototype implementation there were implemented the basic operations on the AVL tree structure: insertion, look-up and deletion.
The reason for choosing the AVL tree data structure is because it has the advantage of being simpler to implement than other self-balancing binary search trees, such as red-black trees or multiway trees like B-trees or T-Trees while their average-case performance is just as efficient.
Since the implemented operations are the classic ones the node structure has an very important role. Table 2 presents the structure of a node from the AVL tree structure.

Field	Description
activityId	**key-identifies the performed activity**
metaInfo	**meta information regarding the performed activity**
userId	**identifies the user who performed the actions within the session**
date	**stores the date when the actions were performed**
data	**stores the data**

Table 2. Structure of node from the AVL tree

The *activityId* field is the key of the structure. Whenever a user starts performing a specific activity than this activity is looked-up in the Cached Data Structure. The *metaInfo* field holds information specific to performed actions. For example, in the case of a DOWNLOAD action in the *metaInfo* field there will be stored information regarding the time duration of the session, the average data traffic speed in bytes/second and data about the assets involved (file names, sizes, locations, etc.) The *userId* field is a foreign key that identifies the user who performed the actions within the heavy traffic session. The date field records when the actions were performed and the data field holds the important data that is to be retrieved to Tesys e-Learning platform at request.
The managing is performed by Cached Data Structure Access Logic. This business logic interfaces with Data Model Logic for insertion and with the logic of Tesys for look-up and deletion. Whenever a call for look-up data within the Cached Data Structure this is accomplished firstly according to *activityId* field. Once the correct activity has been identified than the *metaInfo* field is inspected. When the needed data matches regarding the looked asset the data is then retrieved to Tesys e-Learning platform for deployment.

4 Experimental Results

The study started by setting up the Tesys e-Learning platform. This means that all the learners, course managers and secretary accounts have been created

and the platform was populated with data: course materials, test and exam questions.
This platform is currently in use and has three sections and at each section, four disciplines. Twelve professors are defined and more than 650 learners. At all disciplines, there are edited almost 2500 questions. In the first month of usage, almost 500 tests were taken. In the near future, the expected number of learners may be close to 1000.
Recording learner's activity under these circumstances provides great information regarding user traffic. After six month of usage, there are more than 40,000-recorded actions.
Once the platform was up and running the processing the Analysis Module started receiving data regarding user traffic. The Analysis Module started building models. At beginning, when the data was scarce, the accuracy level was quite low, around 30-40% of instances being correctly classified by cross-validation. After three month of running the algorithm, the obtained decision tree had 17 leaves (which represent in fact classes) and 25 nodes. The time to build the model was 0.13 seconds. The stratified cross-validation evaluation technique revealed that 575 (88.6 %) instances were correctly classified and 75 (11.4%) were incorrectly classified. The confusion matrix showed exactly the distribution of incorrectly classified instances among classes. The results prove that obtained model is accurate enough for starting inserting data into the Cached Data Structure from Data retrieval Module. The behavior of learners has a very important role in obtaining challenger learner's models that at some point may replace the current one. This is a continuous process of improving the model.
After another three month of running the Cached Data Structure reached a number of almost 800 nodes and a size of almost 250 MB. From this point we started to study how response time is influenced by the employed architecture. There was build a mechanism that performed two requests virtually at the same time: one for normal retrieval of data from the hard drive and one that looked up for the data within the cached data structure. The results are good, especial at subsequent similar requests, due to time locality of data. Table 3 presents the obtained results for three situations.

Activity	meta info	Normal access time retrieval	Cached access time retrieval	Difference
DOWNLOAD	**materieId = 7 courseId=5 chapterId=3 size=5MB**	**0.9 s**	**0.78 s**	**-13.3%**
DOWNLOAD	**materieId = 10 courseId=1 chapterId=2 size=3MB**	**0.8 s**	**0.67 s**	**-16,25%**
DOWNLOAD	**materieId = 6 courseId=3 chapterId=5 size=0.9MB**	**0.6 s**	**0.65 s**	**+8.3%**

Table 3. Experimental measurements – comparison between normal access and cached access

The experimental results showed in general a decrease in access time. Still, there are some situations then the response time is greater when the response is delivered through Expertise Module. This is the situation when the overhead introduced by auxiliary logic of Data Retrieval Module is greater that the normal retrieval. The worst-case time situation showed an overhead of 35% over the normal access while best-case time situation showed an improvement of 30%. The average-case performance shoed a general decrease in access time retrieval of 13.5 percent.

4 Conclusions

This paper presents an Expertise Module that runs along an e-Learning platform and whose goal is to decrease the access time of users to assets. This module was divided into two sub modules: Analysis Module and Data Retrieval Module. The first one creates a learner's model in the form of a decision tree based on attributes regarding activities performed and data traffic transferred by users.
The Expertise Module produces data for Tesys e-Learning platform through Data Retrieval Module which implements a AVL tree data structure. The data is inserted through the Analysis Module which manages a learner's model based on their activity and levels of data traffic.
A decision tree learner is used for estimating whether or not the data may be used to obtain significant results. The outcome of decision tree validation is the percentage of correctly classified instances. We say that a value of over 80% in correct classified instances is a promise that we might finally obtain useful knowledge. This business logic in implemented within Analysis Module. A AVL tree structure is used as caching data structure. This is mainly performed to obtain data in timely manner.
The Expertise Module has been tested on data obtained from the e-Learning platform on which 650 learners were enrolled and had activity for six month. The results are satisfactory and prove that

the Expertise Module can be successfully used in an e-Learning process for decreasing the time response. The Expertise Module is planned for running on the same e-Learning platform (same disciplines and same test and exam questions) but on different set of learners. This may lead to further and continuous improvement of the system.
The Expertise Module may also run near other evaluation environments in order to decrease the time response. Within each module there may be used different machine learning techniques or data structures such that different setups may be tested.

References:

[1] Burdescu, D.D., Mihăescu, M.C., 2006. Tesys: e-Learning Application Built on a Web Platform. Proceedings of International Joint Conference on e-Business and Telecommunications. Setubal, Portugal, pp. 315-318.
[2] Rud, O.P., 2001. Data Mining Cookbook – Modeling Data for Marketing, Risk, and Customer Relationship Management, Wiley Computer Publishing, San Francisco, USA.
[3] www.cs.waikato.ac.nz/ml/weka
[4] Quinlan, J.R., 1986. Induction of decision trees. Machine Learning, 1, 81-106, 1986.
[5] G. Adelson-Velskii, E.M. Landis, 1962. An algorithm for the organization of information. Doklady Akademii Nauk SSSR, 146:263–266, (Russian). English translation by Myron J. Ricci in Soviet Math. Doklady, 3:1259–1263, 1962.
[6] Quinlan, J. R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, USA.
[7] Han, J., Kamber, M., 2001. Data Mining – Concepts and Techniques. Morgan Kaufmann Publishers, Wiley Computer Publishing, San Francisco, USA.
[8] Witten, I.H., Eibe F., 2000. Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers, Wiley Computer Publishing, San Francisco, USA.
[9] Donald Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Pages 458–475 of section 6.2.3: Balanced Trees. Note that Knuth calls AVL trees simply "balanced trees".