Towards optimizing server performance in an educational MMORPG for teaching computer programming
Christos Malliarakis, Maya Satratzemi, and Stelios Xinogalos

Citation: AIP Conference Proceedings 1558, 345 (2013); doi: 10.1063/1.4825494
View online: http://dx.doi.org/10.1063/1.4825494
View Table of Contents: http://scitation.aip.org/content/aip/proceeding/aipcp/1558?ver=pdfcov
Published by the AIP Publishing
Towards Optimizing Server Performance in an Educational MMORPG for Teaching Computer Programming

Christos Malliarakis\textsuperscript{a}, Maya Satratzemi\textsuperscript{a} and Stelios Xinogalos\textsuperscript{a}

\textsuperscript{a}University of Macedonia, 156 Egnatia Street, Thessaloniki, Greece
\{malliarakis, maya, stelios\}@uom.gr

Abstract. Web-based games have become significantly popular during the last few years. This is due to the gradual increase of internet speed, which has led to the ongoing multiplayer games development and more importantly the emergence of the Massive Multiplayer Online Role Playing Games (MMORPG) field. In parallel, similar technologies called educational games have started to be developed in order to be put into practice in various educational contexts, resulting in the field of Game Based Learning. However, these technologies require significant amounts of resources, such as bandwidth, RAM and CPU capacity etc. These amounts may be even larger in an educational MMORPG game that supports computer programming education, due to the usual inclusion of a compiler and the constant client/server data transmissions that occur during program coding, possibly leading to technical issues that could cause malfunctions during learning. Thus, the determination of the elements that affect the overall games resources’ load is essential so that server administrators can configure them and ensure educational games’ proper operation during computer programming education. In this paper, we propose a new methodology with which we can achieve monitoring and optimization of the load balancing, so that the essential resources for the creation and proper execution of an educational MMORPG for computer programming can be foreseen and bestowed without overloading the system.

Keywords: optimizing, server performance, computer programming education, MMORPG

PACS: 68U01

INTRODUCTION

Web-based games have become significantly popular during the last few years. This is due to the gradual increase of Internet speed, which has led to the ongoing games development and more importantly the emergence of the Massive Multiplayer Online Role Playing Games (MMORPG) field [7]. In parallel, similar technologies called educational games have started to be developed and put into practice in various educational contexts, resulting in the field of Game Based Learning. These games increase students’ motivation during learning because they are already familiarized with their usage and are entertained by executing game-based actions while also learning [5].

Hence, since motivation is the most significant added value of educational games, their constant and proper operation during learning is important so that learners’ attention will not be deviated due to potential system delay or failure issues. These technologies require significant amounts of resources in order to function properly, such as plentiful bandwidth, RAM and CPU capacity, a powerful host server etc. Focusing on MMORPGs, the most crucial factor that determines their successful operation is the server’s performance. This is jeopardized even more when MMORPGs are developed or configured in order to support computer programming education and thus include a compiler. Supplementary technologies currently used for teaching computer programming consist of educational programming environments (e.g. BlueJ, JGrasp) and microworlds (e.g. ObjectKarel, Jeroo). However, the resources needed for educational MMORPGs are much more demanding. More specifically, the resources transmitted from the clients to the server and vice versa are amplified because players are constantly required to write lines of code in order to pass assigned tasks and the server is required to constantly process and compile programs being sent by multiple clients. Moreover, the server has to provide the appropriate feedback to each player and change the game correspondingly depending on each player’s programming actions. During this iterated process, an even larger amount of resources is bound and the danger for possible performance malfunction increases. This, however, could significantly compromise the learning process. For example, if the operation of the game faces periodic technical problems, the corresponding motivation and students’ interest will face the same decrease. Thus, the optimization of the server’s performance is an even more necessary process for educational games used in computer programming.

The next section briefly describes the related work and its limitations, followed by a section that presents the system performance analysis and the corresponding creation of a mathematical model that estimates how active players are hosted within an MMORPG game in a server so that the server administrators can take the necessary
measures that will ensure optimized server’s performance. Moreover, a process to ensure server’s performance optimization is shown in the following section, while the paper concludes with conclusions drawn.

RELATED WORK

Several studies have been carried out regarding the server system’s performance in online multiplayer games. For example, Borella and Farber have created their own model on game network traffic [1,4]. Additionally, Chang et al. studied the players’ behaviour according to network traffic as well as the factors that affect it [3]. The most commonly mentioned factors include bandwidth delay and consumption [2,6,8]. However, these studies focus mostly on network traffic of strategy or shooting games and less on MMORPGs that require more resources. All changes in a player’s status (e.g. score, new level, new task achieved) are required to be recorded and presented as a feedback not only to the player in question but also to all other players that could see him/her in their view scope [9]. This way, network traffic in MMORPGs involves change recordings and multiple message documentations and transmissions to players of the game.

SYSTEM ANALYSIS – OUR MODEL

The Architecture on which most MMORPGs are based comprises of one or more servers and several clients inter-connected through internet access. The server is usually protected by a firewall or a proxy server mainly for stability and safety reasons. The server administrator is responsible for ensuring the stable performance of the system and the optimization of the data stored in the server’s database. This data is gathered by the game’s system, documented in the database connected with the game and it includes information about each player, such as the player’s status, the world’s condition, the possible actions that can be executed as well as any instances created by the chat function usually embedded within MMORPGs.

More specifically, the player’s status includes all elements that relate to the specific player. Such elements can be the player’s statistics which are constantly renewed along with his/her corresponding evolution during the game (e.g. current difficulty level of programming concepts, health points, strength etc.), the player’s position in the virtual world and all the game objects under the player’s possession. The world’s status comprises of all the characteristics of the virtual world, such as the world map, the virtual players (e.g. intelligent agents) that are programmed by the game creators as well as actions that take place by each player. The actions are all the possible moves and operations that a player can realize or that are occurring at a specific moment (e.g. a spell is being cast; a player’s character is walking etc.). These actions involve both the ones carried out by the actual players and agents. Finally, each MMORPG contains groups of players, thus most games include a chat tool that supports the group members’ communication through the broadcasting of messages amongst them.

According to the standard MMORPG infrastructure which is a typical multi-tier Client/Server architecture, apart from the internet connection, a defining factor in the system’s performance is the smooth execution of the server, since it transmits and receives a large amount of information regarding each player at any given time that the game is operational. However, as the number of active players increases, the chat function that transmits simple text messages continues to have a relative small complexity, so it should not be taken into consideration during the search for the system’s complexity simplification. Moreover, the internet connection speed will also not be considered as a factor, since it could vary depending on the client and its changes cannot be predicted.

It is important that the system hosting a MMORPG is studied for the thorough identification of all factors that influence its performance and thus could potentially lead to system overload. This way, we can determine which elements to take into consideration during load balancing efforts. To this end, this section will propose a mathematical model for the appropriate estimation of these factors and the subsequent configuration of the system should their values pose a risk to the game’s successful execution by learners.

For example, if we assume that we have one player that is playing in a MMORPG, then the server’s performance would depend on the player's status, the world's status and the possible actions that are caused or that can be caused. All these elements are recorded and documented in a database installed within a database server inside the main server; so the more the data produced during a game’s operation the heavier the capacity transferred from the individual clients to the main server and stored into the database server. Hence, the rate in which the above elements’ values change can significantly affect the server’s overall performance and potentially obstruct the game’s proper execution and subsequently the learning process’s successful realization. To this end, we are proposing a model that will estimate the computational cost and the server administrators can take the appropriate actions for possible server upgrading based on these estimations (i.e. dynamic prediction of resources commitment).
This model includes the usage of an algorithm that measures and grades the players’ behaviour, i.e. the value changes of the varying factors that we are examining for each player as they are being stored in the system’s database. The indicators can take values between the scale [0.1, 10] and are considered to be versus time, which means that we are recording the activity occurring across time, as transmitted from clients to the server. This activity’s value is stored as a vector in the database as shown below:

\[
S_{cs}(t) = \left( P^{cs}(t) \middle| W^{cs}(t) \middle| A^{cs}(t) \right) = \sum_{i=1}^{3} \log s_i^{ts}(t) \tag{1}
\]

CS is the indicator of the score that suggests the client-server message transmission, while \( t \) is the time parameter, since the values that these variables take, change through time. The \( S_{cs} \) (Score) is a vector that corresponds to each player and indicates how active the player is in the game’s progression. This score is dependant on the player’s status’ change rate as opposed to a) the other players’ status (variable \( P^{cs} \)), b) the virtual world (variable \( W^{cs} \)) and c) the possible actions that can be executed (variable \( A^{cs} \)). With \( P^{cs}(t) \middle| W^{cs}(t) \middle| A^{cs}(t) \), we indicate that in every time instance the variables’ values change and each new value is documented in the database over time. We further map the \( P^{cs}, W^{cs} \) and \( A^{cs} \) variables to the indices \( x=\{1, 2, 3\} \) respectively in order to alleviate some of the complex representation of the relation and, thus ratings are finally represented as \( s_{cs} \). For example, in a game, when a player moves towards the opponent, the changes that occur are the following: a) his relative position towards the opponent as he is recorded in his screen, b) his absolute position for the virtual world and finally, c) the possible actions that can be executed and the results that can be presented, e.g. when he is near enough he can damage the opponent, an action that could not have happened if he had been far away. Similarly, in a multiplayer educational game oriented to learning programming if a player tries to make moves or other actions by writing lines of code, the changes that occur are relative to the other players’ situation and their code should be configured accordingly.

Moreover, we can estimate that the metric of the activity’s score for each player at a \( t \) time by the \( c \) client to the \( s \) server is the sum of the \( P^{cs}, W^{cs} \) and \( A^{cs} \) metrics. In order to avoid extreme values, we logarithm them, so that they end up being in a \([-1, 1]\) range, where the negative values mean that the player is less active, while the more the values tend towards 1, the more active the player is. This process is mathematically portrayed by the following relation:

\[
S_{cs}(t) = \sum_{i=1}^{3} \log s_i^{ts}(t) \tag{2}
\]

Player’s final score is configured by his interaction with the other players as well as by the agents. The difference lies in that we can shape the system players’ behaviour so that we ensure optimized load balancing, whereas we cannot intervene in the agents’ behaviour, as it is automatically determined by the game system.

Finally, we define two indicators, namely the \( u_1 \) indicator that symbolizes the experience of the player’s interaction with the game’s agents and the \( u_2 \) indicator that symbolizes the experience of the player’s interaction with the other players. So for a player \( a \) the final relation is configured as follows:

\[
T_{scs}(t) = \frac{u_1}{u_1+u_2} \sum_{\forall t, i} s_i^{ts}(t) + \frac{u_2}{u_1+u_2} \sum_{\forall t, j \neq a} s_j^{ts}(t) \tag{3}
\]

The above relation estimates the Total Score (\( TS_{scs} \)) for a player, which depends on values of the elements stored in our game database. In the above relation of the Total Score (\( TS_{scs} \)), the first sum refers to the player’s (\( a \)) interaction with the system, while the second part of the relation refers to the player’s (\( a \)) interaction with the rest of the players (\( j \) where \( j \neq a \) for each previous time instance \( t \)). With the addition of the \( \frac{u_2}{u_1+u_2} \) in the relation we measure the multitude of the user’s interaction with the system while with the \( \frac{u_1}{u_1+u_2} \), we measure the multitude of the user’s interaction with the rest of the players. It is also important to mention that since these variables change through time, the last values they take (i.e. the most recent) should be considered more in the final value of the player’s total score. To represent this, as we show in the relation, we multiply the rating calculated at a specific time instance \( t \), with itself. For example, an instance of the database table that holds this information is depicted in the following Table.

**TABLE 1. MMORPG database instance of a player’s recorded activity**

<table>
<thead>
<tr>
<th>PlayerID</th>
<th>P</th>
<th>W</th>
<th>A</th>
<th>t</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.2</td>
<td>0.1</td>
<td>0.6</td>
<td>1</td>
<td>S</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0.1</td>
<td>0.4</td>
<td>2</td>
<td>S</td>
</tr>
<tr>
<td>2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
<td>3</td>
<td>O</td>
</tr>
</tbody>
</table>
Les us assume that we are trying to measure the total score for the player with an ID 2. This player has executed activities during three different time instances as shown in the t column. Moreover, the u column shows that two of these activities include interaction with the system, while the last one represents interaction with other players of the game. Using the proposed model, the value of the total score will be estimated taking into consideration the fact that the player’s interaction with the system’s agents (the first two rows that have the value u=S) has increased in the 2nd time instance, which shows he is becoming more active and thus could need more resources in the future.

OPTIMIZING PERFORMANCE ANALYSIS

This section demonstrates a way to reduce the big computational cost at run-time indicated by the double sums in the last relation of the previous section. The run-time computational complexity of relation (3) is O(n*t), where n is the number of agents acting in the system and t is the total amount of transactions. Both of these numbers increase rapidly in a MMORPG game. More specifically, the ∑x[u=2]t[s=0]=t sum does not need to be estimated at run time, but incrementally, as each rating vector is gathered and stored. We re-symbolize this sum with the σj*0(t) variable. Additionally, the ∑x[u=2]t variable should also be estimated incrementally and it will be replaced with the σj*T(t) variable. This way, relation (3) is transformed into the following:

\[ T_{CS}(t) = \frac{u_x}{u_x + u_0} \sum_{i=1}^{3} \left[ \sigma_{j*0}^T(t) \right] \cdot \frac{u_0}{u_x + u_0} \sum_{i=1}^{3} \left[ \sigma_{j*0}^T(t) \right] \]  \(4\)

This relation’s complexity is O(n), where n is the total number of agents and players inside the system. The sum in the three dimensions does not count because its limit is stable and already known that it is 3. Thus, we are left with only a sum inside the second addendum. The initial relation (3) has a run-time complexity of O(n*t), i.e. the number of agents and players (n) over the number of transactions (t), which is much larger than the complexity of relation (4).

CONCLUSIONS

The current paper analyses the factors that affect the performance of a server that hosts an MMORPG aiming to address the need for server performance optimization on MMORPGs that support computer programming education. The unbalanced usage of resources available in the server could lead to stagnation when trying to create a new or properly operate an existing educational game that will not face technical issues that can be avoided. To this end, we elaborated on the indicators that influence the server’s performance, representing the variable that includes the total estimation as the player’s total score. This process led to the proposal of a new model that measures and records a player’s status change rate, so that the identified values can be immediately inter-connected with the server’s resources. This way, the server administrator can achieve optimized server performance by configuring the factors that show increased values in the model at run-time or incrementally and thus contribute to an overloaded server.

REFERENCES

7. J. Shanahan, Students Create Game-based Online Learning Environment that Teaches Java Programing, Publisher ACMSE, 2009.