A Model for Massively Multiplayer Role-playing Games System Performance

Jiyi WU, Xiangguo GONG, Jiwen ZHENG
Distance Open Lab, Zhejiang University of Radio and Television, Hangzhou 310012, P.R. China
Dr_PMP@yahoo.com.cn

Abstract

Massively multiplayer role-playing games（MMORPGs）, are among the most popular types of online game. After the introduction of a typical multitiered client/server architecture for MMORPGs, a method for modeling MMORPG system performance is presented. Then experimental and analytical results for a zoned MMORPG and a seamless MMORPG are proposed. The performance model presented here can be used for automated IT resource allocation at runtime and is thus useful in the context of utility computing and on demand systems.

1. Introduction

As the trend of the day, online games are becoming increasingly popular. By 2008, industry revenue will grow to over $2 billion, International Data Corporation (IDC) estimated. Online games enable multiple players to simultaneously interact in a "game world" to which they connect over a network. Most online games follow a C/S model; research is ongoing concerning the feasibility of other types of architecture, such as peer-to-peer and grid architectures. MMORPGs are among the most popular types of online game, it is estimated that MMORPGs hold a 95.5 percent share of the MMOG market. Some examples of MMORPGs are EverQuest, Lineage, and World of Warcraft. Many studies have been performed to understand the online game traffic model and its impact on the Internet.

2. Typical multitiered C/S architecture for MMORPGs

Depending on whether the server process boundaries are explicitly observable inside the game, there are two types of architecture for MMORPGs: the zoned architecture and the seamless architecture.

In the original format of zoned architecture, each zone runs its own process on its own server and manages state in its own memory space. Later, this design was improved to allow a unique process to manage all zones on a single game server but keep each zone independent by mapping between zones and physical game servers with a static process and using configuration files. Such a solution has some limitations. Separating players into separate shards limits their ability to interact. Because players are split first by zone and then by shard, players on different zones of the same shard can only engage in limited interaction, such as text chatting, while players in different shards have no chance to meet each other. This solution also causes abnormal interruption of game playing when the player switches to a "full" zone, and at the moment when he or she disconnects from the previous zone, the player cannot connect to the new zone. In this case, the player may lose his or her status in the game unless it is written to external storage.

To address the limitations of the "zoned plus shard" solution, the seamless architecture was developed. The major difference is that in a seamless architecture game servers need to collaborate with each other to process the game events that have impact across process boundaries and update the status of influenced avatars properly. The major advantages of a seamless game world lie in the larger contiguous game world that is enabled. This leads to a more immersive environment for players and increases the flexibility of game design. Load balancing at runtime increases the scalability of the whole system. At runtime, game processing load can be moved from either failed servers or crashed game processes to other servers. The major disadvantage is that this architecture adds complexity to many aspects of game design and implementation. For example, players' interaction across servers has to be implemented asynchronously (e.g., using message passing or shared memory). Middleware is being developed to solve this problem and simplify such implementations.

Proceedings of the 16th International Conference on
Artificial Reality and Telexistence--Workshops (ICAT'06)
0-7695-2754-X/06 $20.00 © 2006 IEEE

3. A model for MMORPG system performance

3.1. Network traffic in MMORPGs

Traffic related to game logic and to ancillary functions is included in game traffic. Each of these traffic types is comprised of an incoming part and an outgoing part. Normally, updates of a player's status are sent not only to the player but to all other players whose "area of interest" (AOI) includes that player. The AOI of a player represents the scope of that player's perceptions in the game world, according to the game design. Most MMORPGs allow players to chat by using text messages. Accordingly, our network traffic model consists of three parts: the output traffic model, the input traffic model, and chat messages.

⑴Output traffic model

Unlike normal Web applications, the MMORPGs, server-side processing is based on "rounds," that is, the players take turns in controlling the game world. Each round may last several hundred milliseconds. The incoming requests from clients are first put in an incoming queue. In each processing round, the game server iteratively picks up requests in sequence from the incoming queue, processes them, and puts the outgoing messages (updates) in another queue, the outgoing queue. Based on this sequence of events, the output message traffic model can be described by:

$$N_{Out}(t) = \mu_n \times n(t) \qquad (1)$$

Where n(t) is the number of concurrent players at time t, and μ_n is the message size coefficient.

⑵Input traffic model

The dominant part of all incoming traffic is the requests from connected clients to perform some action in the game, such as moving, fighting, or chatting. Chatting is discussed in the next subsection. Another part of the incoming traffic is composed of synchronous packets for purposes of connection maintenance, which are either "heartbeat" messages sent by the client when the player does not take any action for a specific period, or acknowledgement packets responding to a game server's query. Because connection maintenance is necessary only for inactive players who comprise a small part of all players, we can roughly estimate that input traffic is proportional to the number of players and the heartbeat rate. Nonetheless, the actions of players are often successive and bursty and exhibit temporal locality. A more accurate model for input traffic requires detailed study of the behavior of game players; our simplified treatment is open to debate, and we will discuss it further in the next section.

Our input traffic model can be described as:

$$N_{In}(t)=n_{Action}\times n(t)+h_n \times n(t) \qquad (2)$$

n_{Action} is a coefficient based on action messages, which are related to the player's action style and distribution, and h_n is the average heartbeat rate for n players.

⑶Chat messages

Chatting by using text messages is the most popular collaboration mechanism for players in MMORPGs. New types of collaboration mechanism are emerging, such as voice chat. Chat messages could be treated as a kind of action message by the game server or could be dispatched by a dedicated chat server. In either case, chat messages fall into one of three categories:

①*Peer-to-peer messages*—A player sends messages to another player. The traffic caused by such messages can be described by Equation 3, where ∂ is the message size coefficient.

$$N_{P2P}(t) = \partial \times n(t) \qquad (3)$$

②*Broadcast messages*—A player broadcasts messages to all the other players. It is obvious that the traffic caused by a single broadcast message is proportional to the number of concurrent players: one incoming message and n(t) − 1 outgoing messages. Hence the entire traffic caused by broadcast chatting is proportional to the square of the number of concurrent players, where β is the message size coefficient.

$$N_{Broadcast}(t) = \beta \times n(t)^2 \qquad (4)$$

③*Multicast chat messages*—A player sends messages to a group of players. Because the size of the group is relatively small, the model can be simplified to Equation 3, resulting in

$N_{Chat} = N_{P2P} + N_{Broadcast}$

Putting all of these factors together, we have:

$$N(t)=N_{Out}(t)+N_{In}(t)+N_{Chat}(t) \qquad (5)$$

As mentioned in the last section, both incoming traffic and chat traffic depend on the behavior of players. For example, when players fight each other or nonplayer characters in a battlefield, the coefficient in Equation 2 is fairly high. However, according to the design philosophy of MMORPGs, a good game should be a balanced one, that is, one in which the different kinds of action available keep some sort of balance. In our case, we noticed that large-scale battlefields are the territories of senior players who are more powerful, whereas junior players, who make up the largest portion of the population, are busy self-training individually or playing in small groups to improve their skills. Hence, we can roughly assume that each individual player's behavior is independent of that of the other players in this study. Furthermore, as we discussed in the introduction, the traffic for a game shard does not show an apparent periodicity property due to the diversity of global-event update frequency. Thus, from the overall game world and statistic perspective, the user-behavior-related coefficients

Proceedings of the 16th International Conference on
Artificial Reality and Telexistence--Workshops (ICAT'06)
0-7695-2754-X/06 $20.00 © 2006 IEEE

η_{Action} and ∂ should be constant. This allows the traffic model in MMORPG to be simplified to:

$$\begin{aligned} N(t) &= N_{Out}(t) + N_{In}(t) + N_{Chat}(t) \\ &= (\mu_n + \eta_{Action} + h_n + \partial) \times n(t) + \beta \times n(t)^2 \\ &= \varphi_\lambda \times n(t) + \beta \times n(t)^2 \end{aligned} \quad (6)$$

where φ_λ and β are the coefficients which should be constant at the game shard level. Equation 6 could be further simplified if the traffic caused by broadcast chat is small and thus negligible to:

$$\begin{aligned} N(t) &= N_{Out}(t) + N_{In}(t) + N_{Chat}(t) \\ &= \varphi_\lambda \times n(t) + \beta \times n(t)^2 \\ &\approx \varphi_\lambda \times n(t) \end{aligned} \quad (7)$$

Thus, network traffic can be modeled by the number of concurrent players.

3.2. Server performance in MMORPGs

It is well known that in traditional Web applications, server performance can be modeled by the arrival rate. For most Internet applications, this model can be expressed as a linear function:

$$U(t) = \lambda(t) \cdot u_\lambda + b \quad (8)$$

where U(t) is the resource utilization rate, $\lambda(t)$ is the arrival rate at time t, usually defined as the number of requests from clients, b stands for the server resources used by the functions deployed at the server side that are not related to any requests, and u_λ is the resource utilization rate for one request. In our study, b and u_λ represent the CPU usage rate of the game servers.

Although the pattern of MMORPG network traffic is quite different from that of Web applications, Equation 8 can be used to determine the resource utilization rate at the server side for the following reasons. First, the design of game servers follows the producer-consumer pattern in which incoming requests from clients are first put into the incoming queue, and in each round of processing, the game server iteratively picks up requests in sequence from the incoming queue and processes them. Whereas in a given processing round the number of incoming requests could differ from the number of processed requests (e.g., in the statistically unlikely event of queue overflow), under normal conditions the game server should process all requests. Second, as shown in Equation 2, the incoming requests are proportional to the number of players, and thus the arrival rate of incoming messages can be represented by the number of concurrent players.

Another factor that would undermine the linear relationship between server performance and number of concurrent players is the load-balancing algorithm. The number of concurrent players is defined as the number of players in an entire game world, and these players are distributed to each proxy server and game server by the load balancer. If we tokenize the number of players on each game server as $\lambda^i_g(t)$ and the number of players on each proxy server as $\lambda^i_p(t)$ we get the total number of players at time t:

$$\lambda(t) = \sum_{i=1}^{N} \lambda^i_g(t) = \sum_{i=1}^{N} \lambda^i_p(t), \quad (9)$$

where $\lambda(t)$ is the total number of players at time t and N is the total number of game servers. Because the number of proxy servers is the same as the number of game servers, the total number of proxy servers is also N. In the game system that we analyzed, N = 4. Therefore, a point that needs to be considered is: How does the load balancer distribute the players to each server? If the load balancer distributes the players to each server randomly, it cannot be guaranteed that the data will exhibit a linear relationship, even if each of them follows Equation 10, where $\lambda^i_g(t)$ is the number of players on game server i at time t, $U(t)_i$ is the resource utilization rate of server i, b_i indicates the server resources used by the functions deployed at the server side that are not related to any requests, and $u_{\lambda i}$ is the resource utilization rate used by one request of server i:

$$\lambda^i(t) = U(t)_i / u_{\lambda i} - b_i / u_{\lambda i} \quad (10)$$

We define a load balancer to be proportion-consistent if the algorithm used by it dispatches the traffic to each resource proportionally and the proportion does not change over time. Using this definition, we can state that if each server's performance has a linear relationship to the number of players on this server and the load balancer is proportion-consistent, then every server's performance also has a linear relationship with the number of players in the entire game world. Because the proof of this is straightforward, it is omitted here.

4. Experimental and analytical results

4.1. Server performance in a zoned MMORPG

In order to evaluate the server performance model, the Windows** Performance Monitor was used to record the CPU utilization of the related game processes every 5 seconds. These results were then summed to obtain the entire utilization on each server. Two metrics, server performance and number of players, were analyzed and the server performance was averaged during the interval.

$$\lambda(t) = U(t)/u_\lambda - b/u_\lambda. \quad (11)$$

In this equation, U(t) the server performance, u_λ represents the server's CPU utilization cost per player, and b represents the resources consumed by non-game-related processes or daemons. As in Equation 8, b and u_λ are the coordinates of the CPU usage rate of game servers. Once U(t) is obtained, the number of

concurrent players λ(t) can be calculated by performing integration.

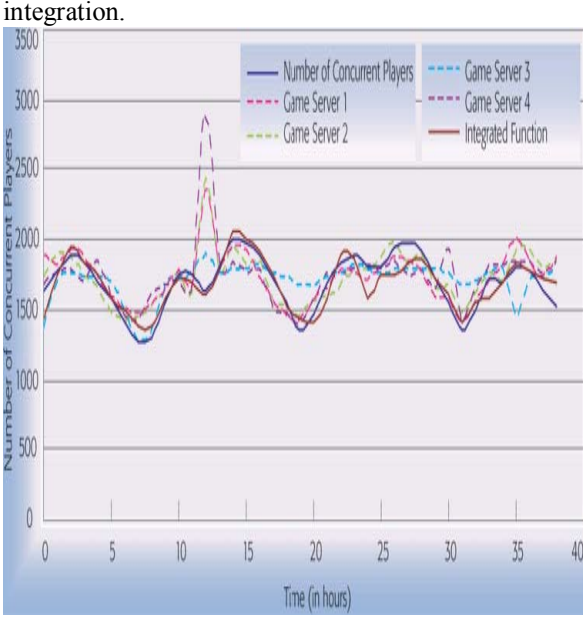

Figure 1. Linear relationships between server performance and number of concurrent players

Figure 1 plots the robust regression result of the servers. It can be seen that although the correlation coefficient of game server 4 is 0.3491, its server performance still fits Equation 8 very well. If some outlying points, less than 5 percent of the original data set, are removed, game server 4 also exhibits a high linear relationship with the concurrent player number with a correlation value of 0.6731.

Because each game server is associated with some zones of the game world, the number of players on each server is equal to the number of players on those zones in the server. Therefore, whether the load-balancing algorithm for the game server is proportion-consistent is decided by the geographical distribution of players. This is why the correlation coefficients of game servers are poorer than those of proxy servers.

4.2. Server performance in a seamless MMORPG

In order to evaluate the server performance model, the CPU utilization of the related game processes was logged every 1 second; these results were then summed to obtain the entire utilization, as we did for the zoned MMORPG.

According to Equation 12, the linear model was evaluated by calculating the correlation coefficient between the number of players and CPU utilization. Then the regression algorithm was used to find the parameters of Equation 11.

$$Cov[n(t), N(t)] = \frac{\sum\{[n(t) - E(n(t))] \cdot [N(t) - E(N(t))]\}}{\sqrt{D[n(t)]\ D[N(t)]}} \quad (12)$$

Unlike the previous experiment on zoned MMORPGs, we can obtain each server's number of players and CPU in this case and display them separately. Because the simulation robot keeps adding avatars into the game world, in the following figures the number of players is always increasing during the test period. From these figures, a strong linear relationship between the number of concurrent players and the CPU utilization can be clearly seen.

5. Conclusion

This study has proposed a performance model for MMORPGs. As the MMORPG is quickly evolving in terms of adopting features of other game genres, the game system will definitely become more and more complicated, as will the behavior of players. Much work should be done on the study of the changes taking place in both the design pattern of MMORPGs and user behavior and the development of a more accurate model for the purposes of prediction.

References

[1] M .Ye and L.Cheng, *iMMOG Design Report*, IBM China Research Lab,2004.

[2] W.C. Feng, F. Chang, "A Traffic Characterization of Popular On-line Games," *IEEE/ACM Transactions on Networking*, 2005, pp. 151-156.

[3] M. S. Borella, J. Farber, W. Feng, and K. Chen, "Source Models of Network Game Traffic", *Computer Communications*, 2000, pp. 403-410.

[4] A. Shaikh, S. Sahu, M. Rosu, M. Shea, and D. Saha, "Implementation of a Service Platform for Online Games", *Proceedings of ACM SIGCOMM 2004 Workshops on NetGames '04*, ACM Press, New York ,2004, pp. 106–110.

[5] B. S. Woodcock, "An Analysis of MMOG Subscription Growth–Version 18.0," MMOGCHART.COM online publication, 2005, http://www.mmogchart.com/.

[6]M.Ye, L.Cheng, "System-performance modeling for massively multiplayer online role-playing games", *IBM Systems Journal*, 2006, 45(1).