An Adaptive Load Balancing Management for Distributed Virtual Environment Systems

Yuanxing Yao[1], Tae-Hyung Kim[1],

[1] Department of Computer Science and Engineering, Hanyang University, 1271 Sa 3-Dong, Ansan, Kyunggio-Do, S. Korea
rainfly00@gmail.com, taehyung@hanyang.ac.kr

Abstract. Distributed Virtual Environment Systems are widely used in massively multiplayer online games. With an efficient distributed architecture and load balancing algorithm, they can support tens of thousands of players interacting with each other. In this paper, we propose an adaptive load balancing algorithm to solve the problem of the center-crowdedness in the DVE systems, while reducing the intra-server communication and client migration cost at the same time.

Keywords: distributed virtual environment, MMOGs, load balancing, graph-based partitioning algorithm.

1 Introduction

With the high-speed growth of players in massively multiplayer online games (MMOGs), the importance of an efficient load balancing management cannot be overemphasized in the distributed virtual environment (DVE) systems [1, 2]. In order to support the game worlds with huge numbers of avatars, we need an efficient load balancing and partitioning algorithm which tries to mitigate a heavy requirement on server resources and communicating overloads. Recently, many works focus on how to reduce the intra-server communication cost caused by adjacent cells which are distributed to different servers in partitioning step as an efficient way to minimize load balancing cost. They generally use connected graph-based algorithm to prevent the high correlation cells are assigned to distinct servers [2, 3, 5, 6]. In this paper, we propose an adaptive load balancing approach that takes into account not only the intra-server cost, but the overhead of client migration for load balancing. We also use a self-adaptive partitioning algorithm to distinguish the situation without border nodes which many graph-based partitioning algorithms cannot be reached.

2 Adaptive Load Balancing Management

In this section, we present an adaptive load balancing approach for DVE systems. Some important definitions are presented to explain our algorithm. An adaptive partitioning is to handle various situations such as no boarder node.

2.1 Area of Interest

In order for users to interact with each other, each client must maintain the local copies of other avatars' states which need to be updated periodically by region servers. If a region server has to broadcast the state information to one and each of all users, the communication bandwidth will be wasteful. That's why the concept of the area of interest is employed: clients are able to receive and send the needed information update only from the adjacent clients with which are being actively interacted. In fact, the area of interest is a circular area whose center is decided by the coordinates of the player's location in the game world and the radius of circle usually is considered as a vision distance because it is the limit to which the player can see.

In this paper, the size of a square microcell is determined by player's vision distance. As a result, a player's area of interest is defined as an 8-direction square group. Fig. 1. (left) shows player P's area of interest according to our definition. Players 1-5 are inside of the player P's 8-direction square vision, so they are player P's interacting client, which means all of them are supposed to frequently exchange update data with each other.

2.2 Border Buffer Region

With the area of interest shown above, when an avatar is moving in the border area between two servers, it should receive and/or send state data to the other avatars who are in the adjacent sever border area. It will also cause the intra-server cost. In order to avoid this overhead and provide a seamless gaming experience, we use these border area microcells to build up a *buffer region*. As shown in Fig. 1. (right), when an avatar enter a buffer region, the area of interest will cover microcell which is in the adjacent server buffer region. Therefore not only the server where the avatar resides holds the state update data, but also the adjacent server will copy the avatar's data as a shadow object including establishing a connection with avatar's client.

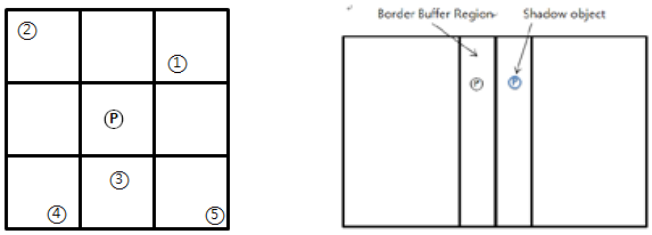

Fig. 1. Area of Interest for the Player P (left), Border Buffer Region and Shadow Object (right).

2.3 Partitioning Algorithm

Take into account that servers in the same clusters are generally connected by internal network and intra-server communication cost is not important, we also suggest our partitioning approach begin from the border cells if overloaded server has like most graph-based partitioning algorithm. But to go one step further, we also need to

consider that there is no border cell or there is no free adjacent server which leads to moving border nodes become no sense.

3 Conclusion

To observe the performance behavior of our algorithm, we establish a simulated virtual world. These virtual worlds are implemented with Microsoft Visual Studio 2005 in multi-server environment. Our multi-server environment consists of a simple 2-machine cluster of Intel core i5 3.20GHz CPU Windows 7 connected by a 100Mbps Ethernet Network card. One server is used as a main server to maintain the virtual world. When the virtual environment is overloaded, the simulator is also responsible for shedding the weight to another server, which is used as a secondary server to maintain the virtual world with light weight. When the main server is overloaded, it is responsible for receiving the shed weight from main server.

In this simulation, we use a small virtual world with a dimension of 6*6 cells. The load threshold of main server is set randomly. The overloaded number of clients in this virtual world is floating between 20 and 40. We also compare our algorithm to the ProGReGA algorithm by Carlos and Claudio [4] and the Layering Partitioning (LP) by Lui and Chan [5]. Our simulation results show that our mechanism works better than ProGReGA and LP using rather smaller system cost. Based on our experimental results, we claim that the load balancing algorithm evaluation model is reasonable enough and our adaptive partitioning algorithm is more efficient among the representative connected graph-based algorithms.

References

1. Aggrawl, G., Motwani, R., and Zhu, A.: The Load Balancing Problem. Proceedings of ACM Symposium on Parallel Algorithms and Architectures, pp. 258-265 (2003)
2. Lety, E., Turletti, T., and Baccelli, F.: Cell-based Multicast Grouping in Large-Scale Virtual Environments. Technical Report No. 3729, INRIA, France (1999)
3. Vleeschauwer, B., Bossche, V., Verdickt, T., Turck, F., Dhoedt, B., and Demeester, P.: Dynamic Microcell Assignment for Massively Multiplayer Online Gaming, Proceedings of the 4th ACM SIGCOMM Workshop on Network and System Support for Games, pp. 1-7 (2005)
4. Bezerra, C. and Geyer, C.: A Load Balancing Scheme for Massively Multiplayer Online Games, Multimedia Tools Application, Vol. 45, pp 263-289 (2009)
5. Lui, J. and Chan, M.: An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems, IEEE Transactions on Parallel and Distributed Systems, Vol. 13(3), pp. 193-211 (2002)
6. Chen, J., Wu, B., Delap, M., Knutsson, B., Ku, H., and Amza, C.: Locality aware Dynamic Load Management for Massively Multiplayer Games, Proceedings of the 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 289-300 (2005)