A Dynamic Load Balancing for Massive Multiplayer Online Game Server

Jungyoul Lim, Jaeyong Chung, Jinryong Kim and Kwanghyun Shim

Digital Content Research Division
Electronics and Telecommunications Research Institute
Daejeon, Korea
{astroid, jaydream,jessekim, shimkh }@etri.re.kr

Abstract. On-line games are becoming more popular lately as the Internet becomes popular, game platforms become diverse and a ubiquitous game environment is supported. Therefore, distributed game server technology is required to support large numbers of concurrent game users simultaneously. Especially, while game users are playing games, many unpredictable problems can arise, such as a certain server handles more server loads than recommended because many game users crowded into a specific region of a game world. These kinds of situations can lead to whole game server instability. In this paper, global dynamic load balancing model and distributed MMOG(Massive Multiplayer Online Game) server architecture are proposed to apply our load balancing algorithm. Many different experiments were carried out to test for efficiency. Also an example of applying real MMOG application to our research work is shown.

1 Introduction

On-line games are becoming more popular lately as the Internet becomes popular, game platforms become diverse and a ubiquitous game environment is supported. Furthermore, as multi-platform game technology, which allows one game to be played in a different platform simultaneously, advances, development of distributed game server technology is required to support large numbers of concurrent game users safely. MMOG(Massive Multiplayer Online Game), in which large number of users play a game in the same game world, is a big genre of heavy server loads because of game event handling, NPC(Non Player Character) control and persistency in game world management. Generally, MMOG distributed game server system is used to handle large numbers of game users, and to provide the necessary consistency to provide game users the same feeling of the same game world, the biggest computational power and network bandwidth is used [1].

In reality, in on-line games, when a certain server load becomes very big to the other servers by an unpredictable case, caused by many gamers crowded in a specific region of a game world, the whole server can become unstable. This can cause response time delay, game server instability etc. and results in an undesirable

situation to maintain consistency. In this case, many different ways of load balancing can solve these problems [1][9][10]. A global load balancing technique which minimizes the load discrepancy by monitoring the whole load and a local load balancing technique which distributes the load to the nearest server using threshold values are researched for a dynamic load balancing technique[2][4][11].

In this paper, to support very large numbers of simultaneous game users, we propose a hierarchical structured distributed server architecture, and global a dynamic load balancing model. The rest of this paper is organized as follows: Section 2 explains MMOG game server system building method for dynamic load balancing; Section 3 defines a dynamic load balancing model which can be applied to a distributed server. Section 4 shows the formation of suggested distributed server architecture and a dynamic load balancing model test is carried out. This shows an example of a real application applied to this system. Finally, section 5 concludes with our results and discusses shortcomings.

2 Distributed MMOG Server system

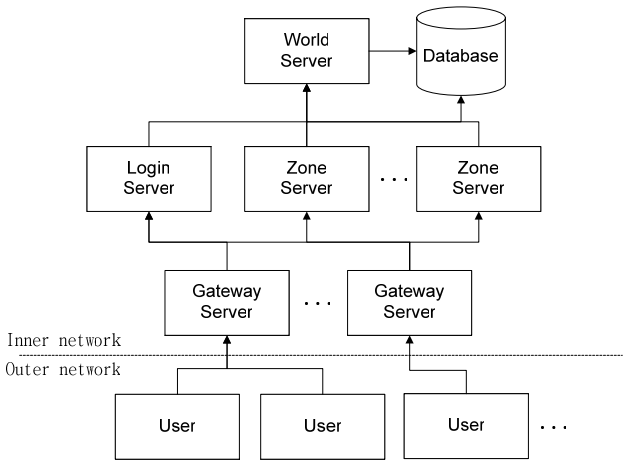


Fig. 1. MMOG structure that has three network layers

2.1 Server Architecture

Because in Distributed MMOG Server multiple game servers support one game, it is important to distribute the game load effectively to maximize efficiency. Generally after splitting each game server into regional groups, game users belonging to a split

area and NPC (Non Player Character) are handled. MMOG Server load is proportionate to the user numbers belonging to the split area.
As illustrated in Fig.1, to form Distributed MMOG Server the following are necessary: World Server distributes and manages a game world; Zone Server handles game processes in a given game space; Login Server authenticates users who participate in the same game. Also, a Gateway Server is required, which handles internal server changes such as server's regional processing area change flexibly by controlling the network communication between User and server.

World Server divides game spaces by the number of Zone Servers. It uses a re-dividing method of dividing the divided areas again after dividing into a maximum number of 4 game spaces. It is managed to quad tree type. Fig.2 shows the splitting order of game spaces and allocated processing areas according to connecting orders of Zone Server.

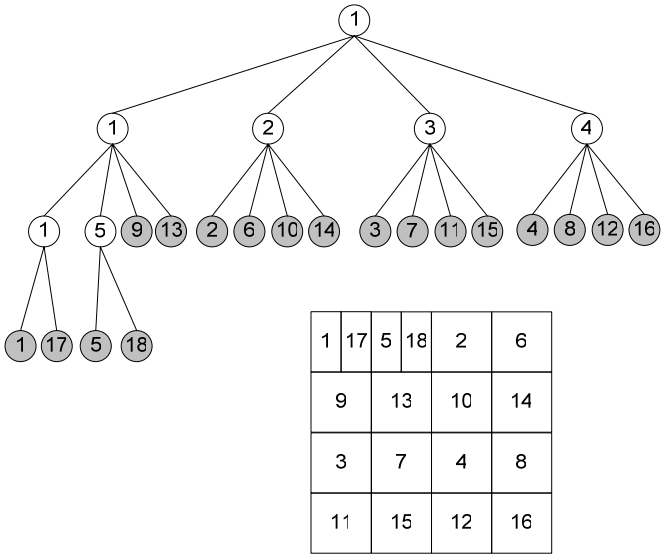


Fig. 2. Game space divided by the connecting order of Zone Server and management of quad tree

2.2 Division and management of game space

Users in MMOG participate in a game after choosing an avatar that represents itself, and while playing a game it sends its information or receives other characters' information through communication to the server. Because the MMOG Server restricts each avatar to exchange information to avatars within an AOI (Area Of Interest) it reduces severe network load and the total amount of network transmission [10][12][13]. For distributed processing of game space form a game world to regular

defined size cell. If the cell size is set up to the same size of AOI radius of an avatar, the AOI of an avatar is defined to be a collection of 9 cells that consists of a cell that it belongs to and cells that are adjacent to it as shown in figure 3.

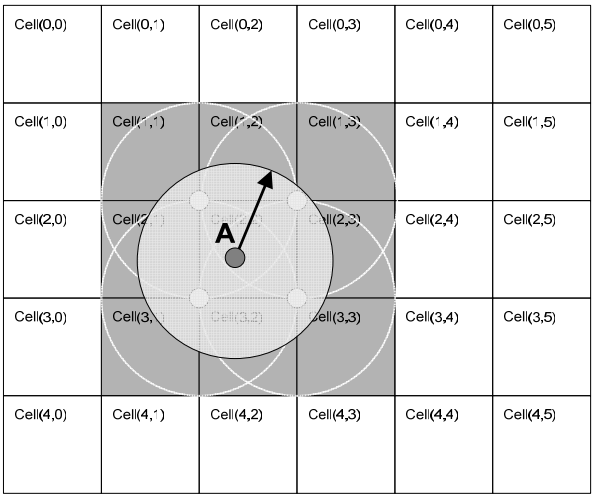


Fig. 3. The interest area of avatar A is a game space divided into 30 cells (the circle formed at the center of A is the Visual field and the 9 shaded cells are the Interest area)

3. Dynamic Load Balancing Model

3.1 Characteristics of MMOG Server Load

The purpose of Dynamic load balancing is to minimize the load differences by evenly controlling the loads between servers by real time monitoring and analyzing the state of the game server. For real time analysis of the server load, first it is important to understand the critical factors that affect the server load of MMOG Server.
For real time analysis of the server load, first it is important to understand the critical factors that affect the server load of MMOG Server.
Firstly, the process loads in the server, which are known as the W(work load), due to the management and process of entities. Secondly, C(communication load) due to the interaction of entities in boundary regions in multiple servers.
Lastly, M(management load) due to domain management in the virtual worlds.[1,3]. Management load is not directly related to the numbers of users. It is a default load generated by the allocated game space and is a very small load compared to the work load and communication load. The boundary domains should be remained in straight line so that an average communication load can be minimized in boundary domains.

Work load and management load of MMOG Server load is generated definitely and proportionally according to the numbers of users and scale of the server area, but the communication load is varied by how the boundary area is defined.
Consequently, this statement indicates a method for minimizing average of *C*, which is a primary factor of loads in communications with other servers under the statement of "distribution of entities in virtual world is unpredictable." If we assume that distribution of entities in virtual world is uniform, then *C* can be minimized when the boundary between partition regions is maintained in straight line and such characteristic improves an application efficiency of managing technique and dynamic load balancing model in virtual world. Fig.4 illustrates the case of 2 to 4 of partitions in virtual world.

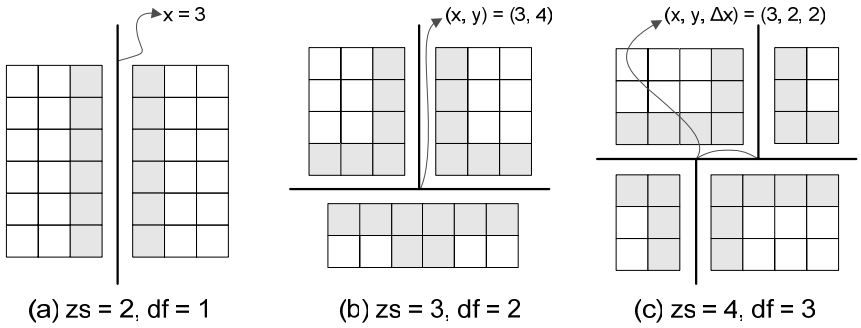


Fig. 4. Minimization of average of C

3.2 Server Load Definition

As defined in 3.1, through work load, communication load and management load, i^{th} server load L_i can be

$$L_i = aW_i + bC_i + cM_i \text{ with } a + b + c = 1. \quad \textbf{(1)}$$

In this Formulas, the terms *a*, *b*, and *c* are weight factors W_i, C_i, M_i, respectively and their sum must be 1. Through using Formulas.1, we are able to acquire a process efficiency enhancement such as an enhancement of server's response time if we maintain the managing region of a server as an optimized state. The dynamic load balancing model for enhancing the process efficiency of a server can be defined by applying reflexive 4-partitioning method in the previous section. If we divide the virtual worlds using 4-partitioning method, we get reflexive load balancing for 4 partitioned regions in maximum. If dynamic load balancing for maximum 4 existing partitioned regions is defined, an efficient model is also defined without consideration of size of virtual world or number of servers. Fig.5 illustrates reflexive operation for dynamic load balancing in reflexive 4 partitioning method. In dynamic load balancing,

it is started with *Depth 0*, *Depth1*, … , *Depth n*, reflexively, and optimized load deviation in each depth uniforms the load deviation in between servers.

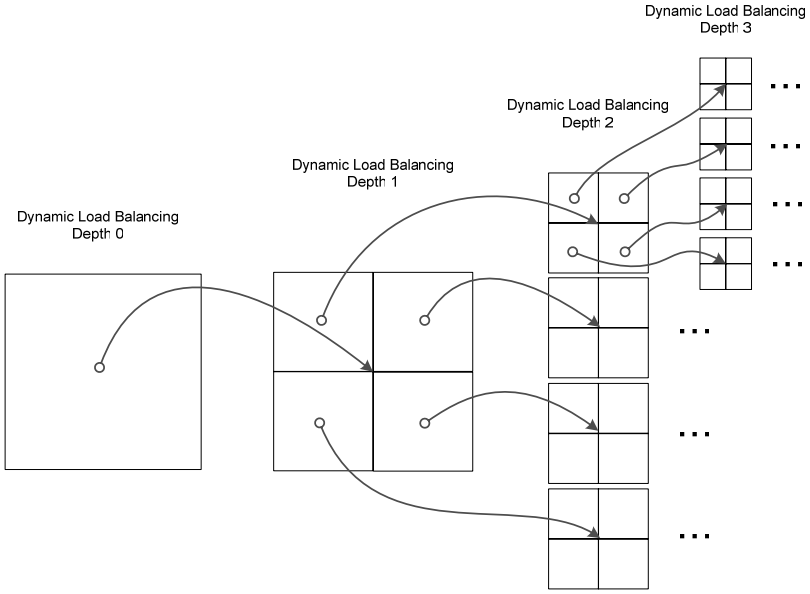


Fig. 5. Reflexive operation of load balancing.

Because finding an optimized server processing region is to maintain each partition with uniform loads, this can be restate that minimization of standard deviation in load of the partition in i^{th} server L_i. Thus, if we set the number of partition in a specific depth as $d\ (\leq 4)$ and an average of load L_{ik} in k^{th} partition as L_k^*, then standard load deviation SD_k for each partition is

$$SD_k = \sqrt{\sum_{i=1}^{d} (L_{ik} - L_k^*)^2 / d} \quad with \ \ d\ (\leq 4) \tag{2}$$

At this time, k^* can be obtained by minimizing SD.

$$\min(\ SD_k) = SD_{k^*} = \sqrt{\sum_{i=1}^{d} (L_{ik^*} - L_{k^*}^*)^2 / d} \tag{3}$$

3.3 Simplified load searching technique

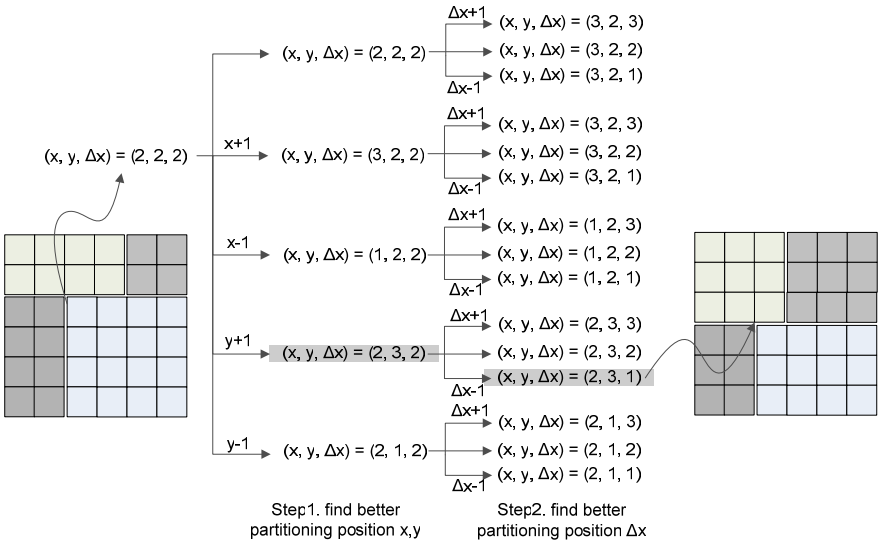


Fig. 6. Simplified load searching technique.

For actual implementation for such dynamic load balancing model, the load deviation of entire virtual world is inspected and then the partition k^* is found using Formulas.3. However, it is difficult to instantly find optimized partition k^* when the virtual world is getting larger. Thus, we believe that application of optimized partition searching techniques that sequentially searches for next optimized partition based on current partition will enhance the application possibility and processing efficiency of proposed dynamic load balancing model [4]. Fig.6 illustrates a Simplified load searching technique based on our idea.

4 Experimental result

In this section, we present experiment for the dynamic load balancing architecture that includes both the distributed MMOG server architecture in section 2 and the dynamic load balancing model in section 3. We used 25 x 25(cell) sized game world and distributed 5000 virtual clients in a uniform distribution, a skewed distribution, and a cluster distribution respectively [7]. We carried out the experiment with the following conditions.

- Because most of Distributed MMOG Server load occurred at network process, management load is ignored by setting c=0 in Formula.1.

- Work load and communication load is defined as the number of network transmissions created by one avatar. So, when the Gateway Server undertakes network broadcasting, the work load is defined by the number of users within split areas.
- Also, the communication load is the total number of existing servers (excluding the server that itself exists in) within AOI of each avatar. Therefore, *a=0.5*, *b=0.5*, and we experimented i^{th} server load as Formula.4.

$$L_i = 0.5 \times W_i + 0.5 \times C_i \quad \textbf{(4)}$$

4.1 Uniform Distribution

With uniform distribution, the measured server load decreased to a narrow range from the point when dynamic load balancing was applied, but the server response time that shows real game server performance was increased by a narrow range or showed a similar level. The result of this experiment shows that dynamic load balancing can even increase the server load in a condition where users are distributed uniformly.

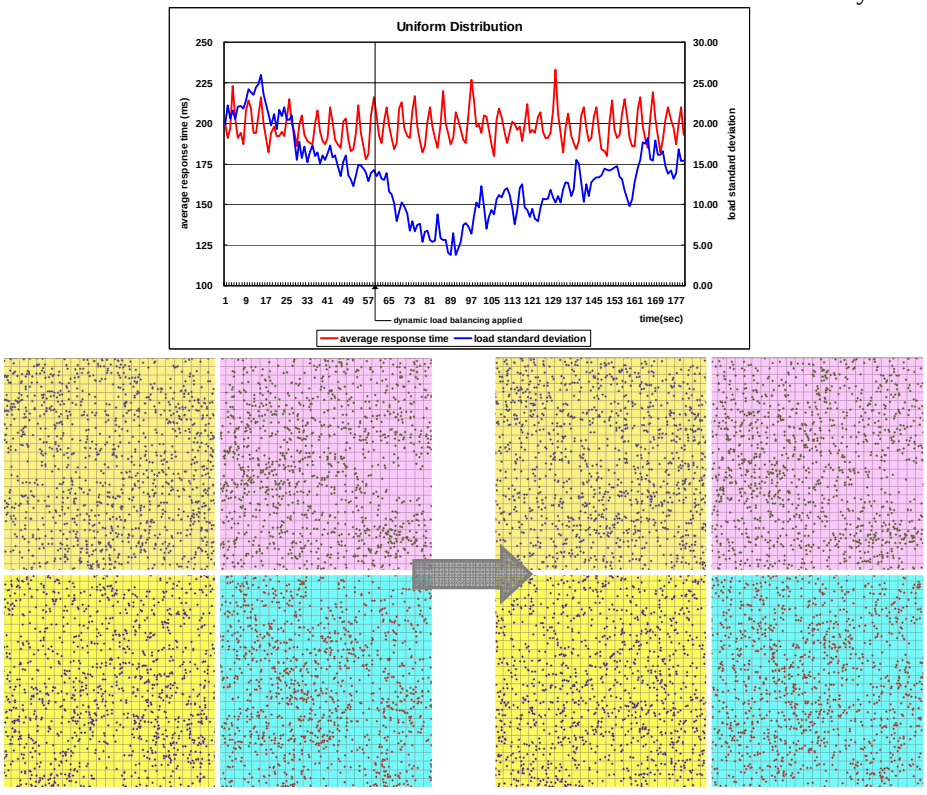
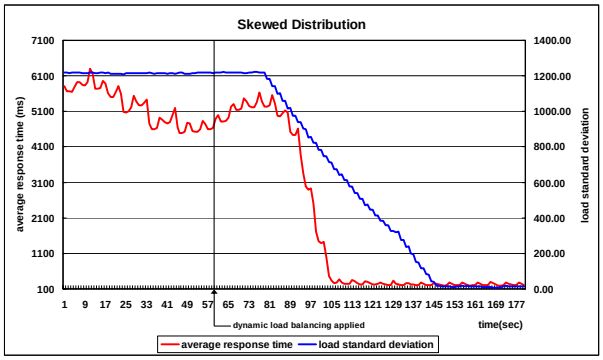


Fig. 7. Changes of Server load evaluation value (blue line) and response time (red line) when users are distributed uniformly

4.2 Skewed Distribution

With skewed distribution, the measured server load decreased by a large range from the point where dynamic load balancing was applied, and the server response time also decreased by a narrow range. It means that dynamic load balancing improves general server performance when users are concentrated in a specific game space. Skewed distribution of Game clients is often seen in real game situation.



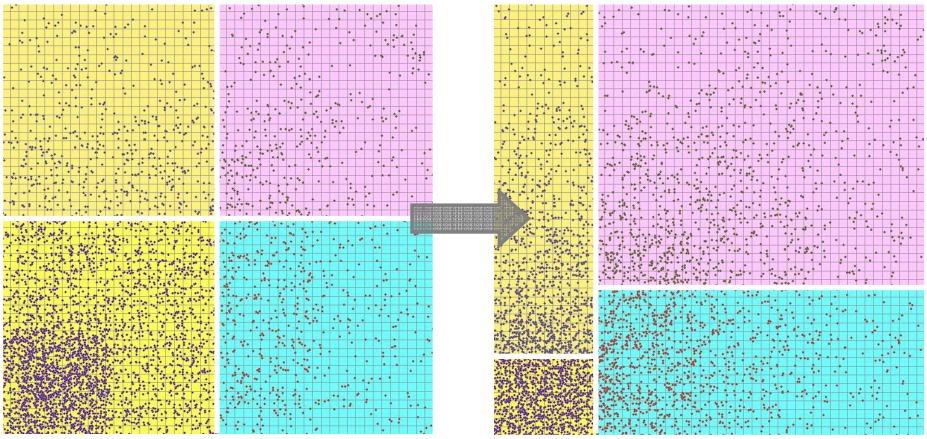


Fig.8 . Changes of Server load evaluation value (blue line) and response time (red line) when users are in a skewed distribution

4.3 Clustered Distribution

The clustered distribution case showed that server load evaluation value and server response time decreased at the same time, which means that these results were similar

to the skewed distribution case. Especially, estimated load evaluation value and response time can be verified that dynamic load balancing model shows better performance with a skewed distribution than a clustered distribution. The results of this experiment indicate that, with a real MMOG game, this research can be applied efficiently in a situation where game users were frequently crowded in one area.

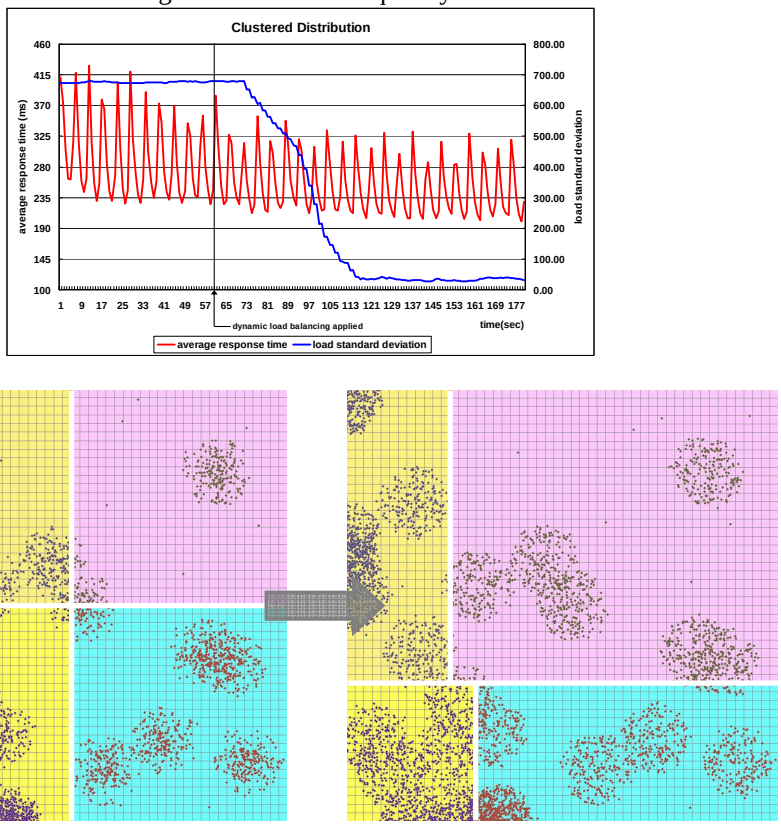


Fig.10. Changes of Server load evaluation value (blue line) and response time (red line) when users are in a clustered distribution

5 Conclusion

In this paper we present the dynamic load balancing architecture which includes both distributed MMOG server system and dynamic load balancing model. This research was applied to real MMOG game and verified server performance through the simulation of a manifold situation that used massive virtual clients. As shown in the experiment's results, the server system efficiency was increased when game clients were in a skewed or clustered distribution in a game world that is similar to a real

game situation, and our server architecture supported server load balancing efficiently which changed in real time. Also, as shown in Fig.11, this research was verified by applying the research products to a commercial game product.

However, although the simplified load searching technique in section 3.3 decreased loads concerned to searching optimized world partition, it sometimes exposed the problem of abnormal partition result for local optimization problem. Nevertheless, the positive results of our experiments show that the load balancing architecture applied to real MMOG server and can also expand its effect to general networked virtual environment such as military training, business etc. In future work, we will improve a real-time load searching technique and apply various shapes of cell such as a brick or hexagon.

Fig.11. Applied images in Elma (commercial on-line MMOG game contents)

6 Acknowledgement

The work presented in this paper has been supported in part by the Korea Ministry of Information and Communication's project: Development of Next-Generation Online Game S/W Technology.

References

1. J. Lui, M. Chan : An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems. IEEE Transaction on Parallel and Distributed Systems(2002)
2. Ta Nguyen, Binh D, Suiping Zhou: A Dynamic Load Sharing Algorithm for Massively Multiplayer Online Games. ICON(2003)131-136
3. J.C.S. Lui, M.F. Chan., K.Y.So., T.S. Tam: Balancing Workload and Communication Cost for a Distributed Virtual Environment. Fourth International Workshop on Multimedia Information Systems(1998)
4. D.Min, E. Choi., Donghoon Lee., B. Park: A Load Balancing Algorithm for a Distributed Multimedia Game Server Architecture: Proceedings of IEEE International Conference on Multimedia Computing and Systems(1999)882-886
5. Beatrice Ng, A. Si, R. Lau, F. Li: A Multi-Server Architecture for Distributed Virtual Walkthrough. ACM VRST(2002)163-170
6. Weyten L, De Pauw W: Quad list quad trees: a geometrical data structure with improved performance for large region queries. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(1989)Volume 8, Issue 3, 229-233
7. YungWoo Jung, et al: VENUS: The Online Game Simulator using Massively Virtual Clients. Lecture Notes in Computer Science, Springer-Verlag(2005) Volume 3398
8. Hunjoo Lee, Taejoon Park: Design and Implementation of an Online 3D Game Engine. Lecture Notes in Computer Science(2004) Volume 3044
9. Microforte: Bigworld game engine. http://www.bigworldgames.com/
10. Emergent: Gamebryo game engine. http://www.emergent.net/
11. Jung Youl Lim, Jin Ryong Kim, Kwang Hyun Shim: A Dynamic Load Balancing Model For Networked Virtual Environment Systems Using an Efficient Boundary Partition Management.IEEE The 8th International Conference on Advanced Communication Technology(2006)
12. Bjorn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins: Peer-to peer support for massively multiplayer games, INFOCOM (2004)
13. Stefan Fiedler, Michael Wallner, Michael Weber: A communication architecture for massive multiplayer games. The 1st workshop on Network and system support for games (2002)