An Efficient Load Balancing Mechanism in Distributed Virtual Environments

Su Min Jang and Jae Soo Yoo

ABSTRACT—A distributed virtual environment (DVE) allows multiple geographically distributed objects to interact concurrently in a shared virtual space. Most DVE applications use a non-replicated server architecture, which dynamically partitions a virtual space. An important issue in this system is effective scalability as the number of users increases. However, it is hard to provide suitable load balancing because of the unpredictable movements of users and hot-spot locations. Therefore, we propose a mechanism for sharing roles and separating service regions. The proposed mechanism reduces unnecessary partitions of short duration and supports efficient load balancing.

Keywords—Distributed virtual environments, load balancing, partitioning algorithm, scalability issue.

I. Introduction

An important requirement of a distributed virtual environment (DVE) system is to provide its users with realistic interaction. This necessitates that the system has good load balancing. As the number of users increases, the DVE system faces some challenging problems, such as overhead, bottlenecks, and so on. There are several research works which partition the global game space into sub-spaces to solve such problems. In [1], microcells were proposed, which can be dynamically assigned to a set of servers to redistribute the load on servers. However, frequent re-mapping and region migration in the method leads to high overhead for servers. Also, in [2], a locality aware load balancing method was proposed which reduces cross-server communication. Though it especially considers awareness of spatial locality in a virtual space, the method also leads to frequent region migrations. Therefore, we propose a new mechanism for sharing roles and separating service regions (SRSS) which reduces unnecessary partitions of short duration.

II. SRSS Mechanism for Load Balancing of a DVE Server

To redistribute the load over several servers, a metric is needed to determine the load on servers. We propose the queue length for request packets (LRP) as the metric. The LRP is the number of tasks in the CPU queue residing in a process. Table 1 shows the three server status levels: light load, normal load, and heavy load. The upper threshold and lower threshold are denoted by T_{up} and T_{low}, respectively.

We propose new partitioning condition parameters for consideration of historical data of players. We use a set of cells (such as 4×4 cells in Fig. 1) as the unit of historical data. The set of cells is called an *upper grid*. This is the minimum size for partitioning virtual space. Figure 1 shows the proposed parameters for partitioning virtual space. The vertical partition S1, which assigns similar areas to two servers, is better than the horizontal partition S2. Also, partition S4 is better than partition

Table 1. Load status of server.

Status	Criteria
Light load	$\text{LRP} \leq T_{low}$
Normal load	$T_{low} < \text{LRP} \leq T_{up}$
Heavy load	$\text{LRP} > T_{up}$

Manuscript received Dec. 12, 2007; revised May 19, 2008; accepted June 25, 2008.

This work was supported by the Korea Research Foundation Grant funded by the Korea government (MOEHRD) (The Regional Research Universities Program/Chungbuk BIT Research-Oriented University Consortium.) and also supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. R01-2006-000-1080900).

Su Min Jang (phone: + 82 43 261 3230, email: jsm@cbnu.ac.kr) and Jae Soo Yoo (email: yjs@cbnu.ac.kr) are with the Department of Computer and Communication Engineering, Chungbuk National University, Cheongju, Rep. of Korea.

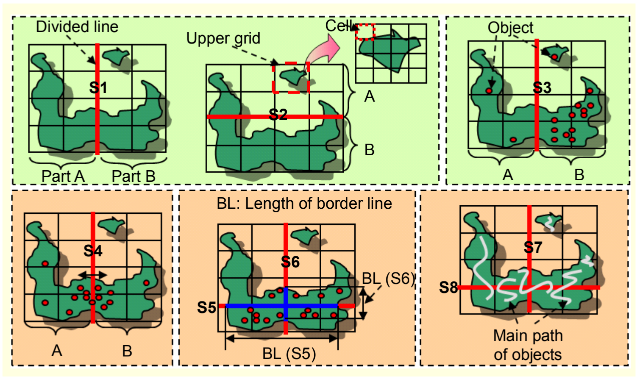


Fig. 1. Proposed five parameters for optimized partitioning.

S3 in terms of evenly partitioning objects. Most objects in partition S4 were concentrated near the line of partition. Such objects have a high probability of migration from the current server to the other neighboring server. Therefore, the partition S4 leads to a high cost of server migration (SM). It was better to divide the virtual space at the point which yields the smallest number of objects near the line of partition. Also, the longer the border line, the higher the cost of SM. So, partition S6 is better than partition S5. Further, we consider the main paths of objects. Partition S7 is better than partition S8, for which the many main paths of the objects are cut by the line of partition. The main paths of objects are generated by summarizing the trajectories of objects based on the upper grid. However, because the partition procedure using historical data in real-time involves a high overhead, we partition the virtual space according to the proposed parameters when DVE systems are initiated. We evaluated our method except the main path of objects in [3], which provides an optimized equation for partitioning the virtual space.

To solve problems of hot-spot location that result from unpredictable movements of a massive number of users, we propose an SRSS mechanism with two levels; level-SR (sharing roles) and level-SS (separating service regions). The SRSS mechanism initially applies level-SR to heavily loaded servers. If it cannot solve the overload problems, level-SS is then applied. The first level, level-SR moves the parts of modules from heavily loaded servers to lightly loaded servers, without repartitioning virtual space. DVE servers have three main modules *Packet_Receiver,* which receives requested queries of users, *Packet_Processor,* which processes requested queries of users, and *Packet_Sender,* which sends results to users. The *Packet_Receiver and Packet_Sender* modules (*Packet IO* module) use a substantial proportion of the total run-time of the DVE server. Also, the *Packet IO* module does not need synchronization among servers. Therefore, we use it for sharing functions of level-SR. Figure 2(a) shows the procedures of level-SR. The following procedure is repeated

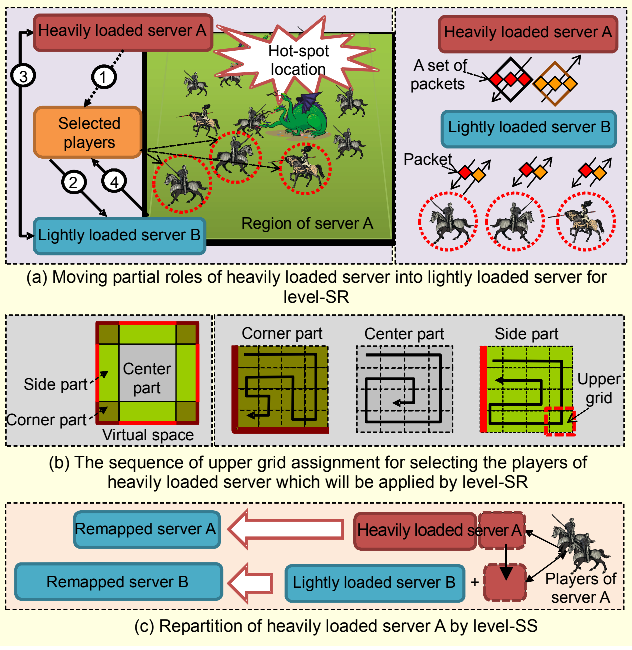


Fig. 2. Example of the proposed SRSS mechanism.

until server A has the status of a normal load.

① The heavily loaded server A initially selects players to be applied by level-SR (*SRPlayers*) based on the probability of server migration. *SRPlayers* disconnect from heavily loaded server A.

② *SRPlayers* connect to lightly loaded server B. Server B receives the request packets from individual *SRPlayers.*

③ Server B combines the received request packets into one packet and sends it to server A. Server A sends a set of result packets of *SRPlayers* to server B.

④ Server B delivers the result packets to individual *SRPlayers*.

Figure 2(b) shows the assignment sequence of upper grids for selecting players to be applied by level-SR. Sequences are made according to the probability of server migration and locations of the server's virtual space, namely, the corners, the center, and the sides. However, when level-SR is unable to solve the hot-spot location problems, we finally apply level-SS to the server. Level-SS involves placing regions of the overloaded server in a bin for assignment to the server with the lightest load with the sequence of upper grids used at level-SR as shown in Fig. 2(c). The SRSS method is executed in sub-linear time $O(gn)$, in terms of the number of upper grids(g) involved in a server, and the number of server participants (n).

III. Performance Evaluation

We evaluated our mechanism in single-server mode and multiple-server mode under various conditions.

ETRI Journal, Volume 30, Number 4, August 2008 Su Min Jang et al. 619

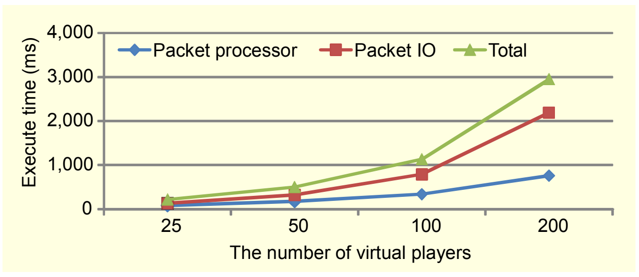
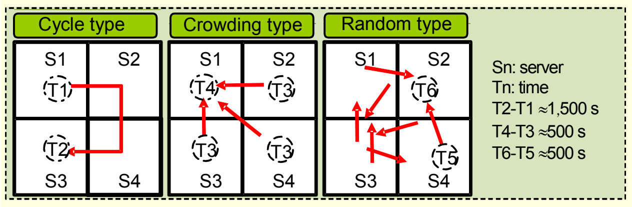


Fig. 3. Weight of Packet IO modules in single-server mode.



(a) Three movement scenarios of virtual players

	Cycle	Crowding	Random
Lightest	21	11	8
SRSS	0	3	1

(b) Frequency of total region partitions

	Cycle	Crowding	Random
Lightest	1,890	5,510	1,080
Static	3,160	17,500	1,720
SRSS	510	1,280	670

(c) Maximum delay time in four servers (ms)

Fig. 4. Frequency of total region partitioning and the maximum delay time in a multiple server mode.

In the experimental environments, virtual players (VP) are able to randomly move, seek monsters (within 5×5 cells), and fight monsters, which exhibit simple AI actions (move, avoid, attack, path finding). VPs have a screen view (8×8 cells) and can have a special destination for creation of a flocking. VPs periodically send location data and action data (the packet size is 1,024 bytes). VPs receive the status data of neighbors (8×8 cells) from the server. The size of the virtual space of a server is 1k×1k cells (single-server mode) or 10k×10k cells (multiple-server mode). Each server periodically updates a player's location data and a monster's AI actions. The server periodically sends players updated data about the region. There are eight Pentium IV 2.0 GHz LINUX systems each with a main memory of 1,024 MB connected with 100 Mbps Ethernet.

Two other load balancing methods are considered in comparison. *Lightest* is a dynamic load balancing algorithm that attempts to optimize the cost of remapping by prioritizing the redistribution of load to a single server. The regions of an overloaded server are placed into a bin for assignment to the server with the lightest load. *Static* is a fixed partitioning method. The regions of servers are fixed with equal shares of the virtual space. This method is used as a baseline comparison with our SRSS mechanism (dynamic partition).

To evaluate the weight of the *Packet IO* module in single-server mode, we varied the total number of virtual players from 25 to 200 as shown in Fig. 3.

When the number of virtual players increased, the total execution time increased sharply from 502 ms for 50 players, to 2,950 ms for 200 players. The execution time of the *Packet IO* module for 200 players is 2,188 ms, and the total execution time is 2,950 ms. The *Packet IO* module has a heavy weight (74%) on the load of the DVE server. Therefore, moving the *Packet IO* module from servers with heavy loads to servers with light loads reduces the load on heavily loaded servers. To evaluate the frequency of region partitioning and the maximum execution time of DVE servers in multiple-server mode, four regions distributed over four servers were configured as shown in Fig. 4(a) with 100 virtual players evenly distributed. There are three movement types, namely, cycle, crowding, and random, by which virtual players move toward their destination. Figure 4(b) shows that the frequency of total region partitions for the *Lightest* method for cycle movements is 21. However, the SRSS method has no region partitioning, because of shared modules at level-SR. Also, servers for the SRSS method have a shorter maximum delay time than the *Lightest* and *Static* methods, as shown in Fig. 4(c), because they reduce unnecessary partitions.

IV. Conclusion

This paper presented a load balancing mechanism which reduces unnecessary partitions, using new partitioning parameters, such as the number of objects near the line of partition, length of the line of partition, and the main path of an object. Simulation results demonstrate that our method is more efficient than the existing methods. In the near future, our mechanism will be applied to real applications for analysis of its performance.

References

[1] B De Vleeschauwer, B Van Den Bossche, and T Verdickt, "Dynamic Microcell Assignment for Massively Multiplayer Online Gaming," *Proc. Net Games*, Oct. 2005, pp. 1-7.

[2] J Chen et al., "Locality Aware Dynamic Load Management for Massively Multiplayer Games," *Proc. PPoPP*, June 2005, pp. 289-300.

[3] S.M. Jang and J.S. Yoo, "An Efficient Distributed MMOG Server Using 2Layer-Cell Method," *Proc. Entertainment*, vol. 4469, June 2007, pp. 864-875.