ATLANTIS PRESS

Advances in Computer Science Research (ACRS), volume 54

International Conference on Computer Networks and Communication Technology (CNCT2016)

A Dynamic Load Balancing Method for Reducing Migrations in DVE Systems

Wei ZHANG [1,*], Hang-jun ZHOU [2] and Zhao-ning ZHANG[1]

[1] College of Computer, National University of Defense Technology, Changsha, China

[2] Department of Information Management, Hunan College of Finance and Economics, Changsha, China

Keywords: DVE Systems, Multi-server infrastructure, Load Balancing, Migration.

Abstract. Adopting multi-server infrastructure is an important way to improve the scalability of distributed virtual environment (DVE) system. Under this infrastructure, maintaining balanced load among multiple servers is of crucial to keep system usability. Existing load balancing methods mainly focus on the efficiency and effectiveness of the load balancing process, that is, how to reduce the number of overloaded server under less time cost, but ignore the influence of the client migrations during the running time. In this paper, we propose a new dynamic load balancing method based on the calculation of cell migration priority, which can effectively reduce the probability of client migrations between multiple servers, thus improving the overall system performance. The experimental results proved the effectiveness of the proposed method.

Introduction

With the rapid development of parallel and distributed simulation technology [1] [2] and network technology, distributed virtual environment system [3] has been widely used in many online activities such as distance education, online game and military training. With the continuous increasing of the scale of DVE system, the traditional single-server architecture has been difficult to meet the needs of user's interaction efficiency. Therefore, multi-server infrastructure virtual environment system has gradually attracted the academic and industry attention. Unlike a single server architecture, in a multi-server architecture system, multiple geographically distributed servers are connected through a high-speed network and served together for users. Each server is responsible for maintaining a small area in the virtual world where users and activities in this area are handled by this server. When the virtual world expands or the number of users increases, some new servers can be added to increase the system's service capability, thus greatly improving the scalability of virtual environment system.

In a multi-server DVE system, to maintain the usability of the system, we need to ensure that the load of each server does not exceed its service capability. However, due to the constant movement of the user-controlled entities and the occurrence of some hot events in the virtual world, the number of users in some area could be much higher than the number of users in other areas. As a result, the server which provides service for the high load area may be overloaded. At this time, we need to adopt a load balancing approach to migrate some part of the load to other servers. The traditional load balancing methods mainly consider the efficiency and effectiveness of this process, hope to achieve a balanced load between servers with a small time cost, but neglect the influence of the load balancing result on the migration probability of the clients. In fact, if the clients migrate frequently among servers, on the one hand it will increase the

communication overhead between servers, on the other hand it will affect the stability of load balancing results. If the load balancing process can take into account this factor, we may effectively reduce the probability of client migrations.

In this paper, we propose a dynamic load balancing method that can reduce client migrations. In the process of load balancing, the candidate cells that can be migrated are prioritized and sorted, and the cells that can reduce the overall migration probability of the system are selected and migrated from the high-load server to the low-load server, thus reducing the probability of client migration between multiple servers and improving the usability of the system on the basis of guaranteeing balanced load among servers.

The rest of this paper is organized as follows: The second part reviews the existing load balancing methods, the third part describes the dynamic load balancing method presented in this paper, the fourth part shows and discusses the experimental results. Finally, the fifth part concludes the paper.

Related Work

The load balancing methods in multi-server infrastructure DVE system can be divided into two categories: load balancing methods for global optimization and load balancing method based on system state. Load balancing methods for global optimization [4] [5] are to calculate an optimal load distribution result for all servers but ignore the current load of each server. Although this kind of method can obtain the optimal load balancing result, the new allocation scheme may have a big difference from the existing state, which makes the adjustment process to carry out a large number of load migration, and introduces a large system overhead.

Load balancing method based on system state [6] is to find a solution to adjust the load distribution according to the current load of servers, and try to decrease the number of the overloaded server. This kind of method can effectively reduce the system overhead in load balancing process. Load balancing method based on system state can also be divided into centralized load balancing method and distributed load balancing method. Centralized load balancing method [7] is to collect the global information in a central computing node and run a centralized program to get the adjustment plan in the central node. This method can achieve good load balancing result, but when the system scale is large, the process of collecting information and computing becomes the bottleneck in the whole load balancing process. The distributed load balancing method is running an independent process at each server according to its own state. Compared with the centralized load balancing method, distributed load balancing method [7] has better efficiency and scalability, but it is not easy for the distributed process to get an optimal result. In order to balance the efficiency and effectiveness, a load-balancing method based on heat diffusion is proposed in [8]. The load-balancing process between servers is controlled by a mechanism similar to heat diffusion, and improved results can be obtained in this process with small time cost.

There is also a class of load balancing methods which consider the delay between clients and servers in the load balancing process, that is, try to match the clients to servers which has smaller communication delay with them. In [9], the authors generalize this problem into a zone-mapping problem, and proves that the problem is NP-hard and cannot be solved optimally in polynomial time. To this end, they designed some heuristic search methods [10] which can obtain better results in a relatively short period of time. In [11], the authors divide the whole load balancing process into the

cutting stage and the matching stage, and propose a load balancing method based on balanced cutting, which can obtain the optimal mapping result based on balanced cutting in polynomial time. In [12], the authors propose a load-balancing method based on irregular cutting to construct some voronoi regions according to the locations of the clients to capture the dynamic characteristics of the entities in the virtual environment.

In order to reduce the client migrations, in [13], the authors divide the virtual world into several connected regions, and proposes a dynamic matching method to maintain region connectedness during runtime. In [14], the authors propose a load balancing method based on comprehensive factor calculation, which aims to reduce the probability of client migrations by considering the cell load and the distance between the cell and the target region.

Dynamic Load Balancing Method to Reduce Migrations

In this section, we first describe the system model used in this paper, and then we will describe the main dynamic load balancing process.

System Model

In a multi-server infrastructure distributed virtual environment system, the whole virtual world is commonly composed of many basic areas in order to facilitate the distribution of virtual world among multiple servers. In this paper, we divide the entire virtual world into a large number of square areas, which we call cells, and assume that a cell is the basic unit of migration between servers in dynamic load balancing process. We use CS to represent the set of all cells, SS to represent the set of all servers, and ES to represent the set of all clients. Given a cell $C_i \in CS$, if the area and the clients it contains are served by a certain server S_a, we call the cell C_i is matched to the server S_a.

We divide the whole running time of a DVE system into many cycles. For a given cell C_i, we use $l_k(C_i)$ to represent the workload in cycle t_k, that is the number of clients on the cell. We use function $M_k(C_i)$ to represent the mapping server of cell C_i in cycle t_k. If C_i is mapped to server S_a, then $M_k(C_i) = S_a$. For server S_a, we define a region $R_k(S_a)$ as a set of all cells mapped to server S_a in cycle t_k.

Definition 1 The region $R_k(S_a)$ is defined as

$$R_k(S_a) = \{C_i, where\ M_k(C_i) = S_a\} \tag{1}$$

Definition 2 The load of $R_k(S_a)$ is defined as

$$L_k(S_a) = \sum_{\forall C_i \in R_k(S_a)} l_k(C_i) \tag{2}$$

We use $O_k(S_a)$ to represent if server S_a is overloaded in cycle t_k. If the load exceeds the threshold, $O_k(S_a)$ is set to 1, otherwise it is set to 0. Then we can define the overall system overload value.

Definition 3 The overall system overload value O_{all} is defined as

$$O_{all} = \sum_{\forall S_i, \forall t_k} O_k(S_i) \tag{3}$$

Now we describe the migrations, for a given client E_i, we also use function $M_k(E_i)$ to represent the mapping server of client E_i in cycle t_k. We use $mg_k(E_i)$ to record the

migration events. If client E_i migrated to another server in cycle t_k , i.e. $M_k(E_i) \neq M_{k+1}(E_i)$, $mg_k(E_i)$ is set to 1, otherwise it is set to 0. We use MG_{all} to represent the total number of migrations throughout the run-time.

Definition 4 The total number of migrations MG_{all} is defined as

$$MG_{all} = \sum_{\forall E_i, \forall t_k} mg_k(E_i) \tag{4}$$

Therefore, the first goal of dynamic load balancing method is to minimize overall system overload value as far as possible. On this basis, it will try to minimize the total number of client migrations. In the next section, we will present the dynamic load balancing process presented in this paper.

Dynamic Load Balancing Method

The process of dynamic load balancing is usually to check whether there is an overloaded server in the system and to migrate some of the load from the overloaded server to a server with a relative low load when overload situation occurs. From our system model description we can see that in the process of migrating, choosing proper cells is the core stage of this process, because migrating different cells will result in different probabilities of client migrations in the following cycles. In this paper, we will introduce two rules to support the cell selection process, they are boundary line rule and square rule.

The boundary line rule is to reduce the length of the borderline between the original region and the target region as much as possible during the cell migrating process. In fact, in a multi-server distributed virtual environment, the probability of client migrations between different servers is proportional to the total length of the boundary lines between servers. In other words, the longer the boundary line is, the greater the probability that the client migrate between different servers is, and vice versa. Therefore, we need to reduce the length of the boundary line in the load balancing process. To facilitate calculation, we introduce the definition of boundary line adjustment values.

Definition 5 The boundary line adjustment value of transferring C_i from S_a to S_b in cycle t_k is defined as

$$BA_{k,a,b}(C_i) = BL_k(S_a, C_i) - BL_k(S_b, C_i) \tag{5}$$

In the definition, $BL_k(S_a, C_i)$ represents the length of the boundary line between the client and the server. The value ranges from 0 to 4. When the adjustment value is positive, it indicates that the migration increases the overall length of the boundary line, while the negative value indicates that the migration reduces the overall length of the boundary line. For example, if there is only one shared edge between a cell and the original region and three shared edges between the cell and the target region, the adjustment value is -2, which means that the migration will reduce the overall length of the border line by 2. Therefore, when sorting all the potential migration cells, we need to sort the boundary line adjustment value from low to high, and migrate the cells with lower boundary adjustment values firstly.

The square rule is trying to change the shapes of the original region and the target region and make them closer to squares as much as possible. This is because the square has a low value of ratio of the perimeter to the area compared to most other shapes, that is, it has a shorter border line in the same size of the area. For example, for a square area, if we want to divide it into four parts. Then if we divide it into four small square partitions, the total border line will be twice the length of the side, and if we cut it

horizontally and divide it into 4 rectangles from top to bottom, the total border length becomes three times the length of the side. The reason is square has a low value of ratio of the perimeter to the area compared to rectangle. In order to facilitate the calculation, we introduce the definition of the square index and the square adjustment value.

Definition 6 The square index of C_i to S_a in cycle t_k is defined as

$$RI_{k,a}(C_i) = Max(DV_k(S_a, C_i), DH_k(S_a, C_i)) \Big/ \sqrt{|R_k(S_a)|} \tag{6}$$

The square index is calculated by dividing the maximum value of the distance between the cell and the center point of the region in the horizontal and vertical directions by the square root of the area, where the square root of the area represents the length of the side if we change the shape of the region to a square. So the square index represents the hindrance of this cell when changing the shape of the region to square. The bigger the hindrance is, the more the cell should be migrated to other servers.

Definition 7 The square adjustment value of transferring C_i from S_a to S_b in cycle t_k is defined as

$$RA_{k,a,b}(C_i) = RI_{k,a}(C_i) / RI_{k,b}(C_i) \tag{7}$$

The square adjustment value is defined as the quotient of the cell's square index in the original region to the cell's square index in the target region Therefore, we prefer to migrate the cell with a larger square adjustment value.

The above-mentioned borderline adjustment value and square adjustment value are used to determine the priority of the potential cells. Now we will introduce a specific distributed load balancing process. First, in the initial stage, we cut the virtual world into many regions with similar load, and assign each of them to one server. At runtime, each server periodically checks its load, initiating a dynamic load balancing process when it is overloaded and trying to migrate some cells to neighbor servers. The overall process is shown in Figure 1.

Algorithm 1.

1: ***if*** $L_k(S_x) > L_{th}$ ***then***
2: *receive load situation from neighbor servers*
3: ***for*** *every neighbor server* S_a
4: ***if*** $L_k(S_a) < L_k(S_x)$ ***then***
5: $MigrationLoad = (L_k(S_x)-L_k(S_a))/Degree(S_a)$
6: $PCells_{x,a}=\{C_i, where\ M_k(C_i)=S_x\ and\ C_i\ can\ be\ transferred\ to\ S_a\}$
7: *Order cells in* $PCells_{x,a}$ *in ascending order of* $BA_{k,x,a}(C_i)$
8: *Further order cells in* $PCells_{x,a}$ *whose* $BA_{k,x,a}(C_i)$ *are same in descending order of* $RA_{k,a,b}(C_i)$
9: ***while*** $MigrationLoad > 0$ ***do***
10: *select the best Cell* C_m *from* $PCells_{x,a}$
11: *transfer* C_m *from* S_x *to* S_a
12: $MigrationLoad = MigrationLoad - l_k(C_m)$
13: ***end while***
14: ***end if***
15: ***end for***
16: ***end if***

Figure 1. Distributed load balancing process

In Figure 1, the overloaded server first obtains the loads of neighbor servers. If neighbor servers can receive some load, a load balancing process is started. In this

process, the total number of loads to be migrated is firstly calculated, which is equal to the difference between the loads of two servers divided by the node degree of the neighbor server. After which all the cells that can be migrated are collected in the selected migration cell set. Then we first calculate the boundary line adjustment value and square adjustment value for all cells in the set, and sort the boundary line adjustment value from low to high. For cells with the same boundary adjustment value, we sort them in the descending order of square adjustment value. Finally, the cells are migrated to the neighbor server one by one from the set until the load constraints are met. Since we take into account the client migrations in the process, the overall probability that clients migrate across multiple servers is decreased. In the next section, we will demonstrate the performance of the proposed load balancing method.

Experimental Results

To verify the effectiveness of this method, we constructed a 100*100 simulated DVE system. In this system, we divide the whole virtual world into 10,000 cells with size of 1*1, and divide the 10,000 cells into 100 regions and mapped them to 100 servers. We place 2,000 to 10,000 clients in the virtual world and make them walk around in the virtual world. A screenshot of the simulated DVE system is as shown in Figure 2.

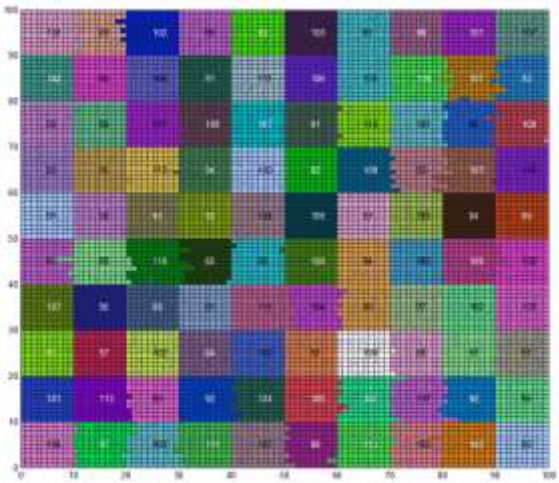

Figure 2. A screenshot of the simulated DVE system

To simulate the behavior of a user in a virtual environment, each client randomly selects a direction and speed while walking, and continues for up to 40 cycles in that direction and speed. In addition, in order to avoid the impact of virtual environment boundaries on the system, we assume that the top and bottom boundaries of the virtual world and the left and right boundaries are connected together, that is, if the client continue to move up at the top cell of the virtual world, it will reach the corresponding bottom cell, so we can ensure that each cell has four adjacent neighbor cells. We simulate the virtual environment system for 1,000 cycles, and get the statistics of the migrations. The results are shown in Figure 3.

In Figure 3.a, the x-axis represents the number of clients placed in the virtual world, varying from 2,000 to 10,000, and the y-axis represents the average number of client migrations per cycle during the entire run time. The blue bars represent traditional load-balancing methods that do not consider migrating factors, while the green columns represent the dynamic load-balancing approach presented in this paper. From the results, we can see that this method can effectively reduce the total number of client migrations

in all cases from 2,000 to 10,000, and with more clients our approach can get more improvement. The improvement ratio can reach more than 20%.

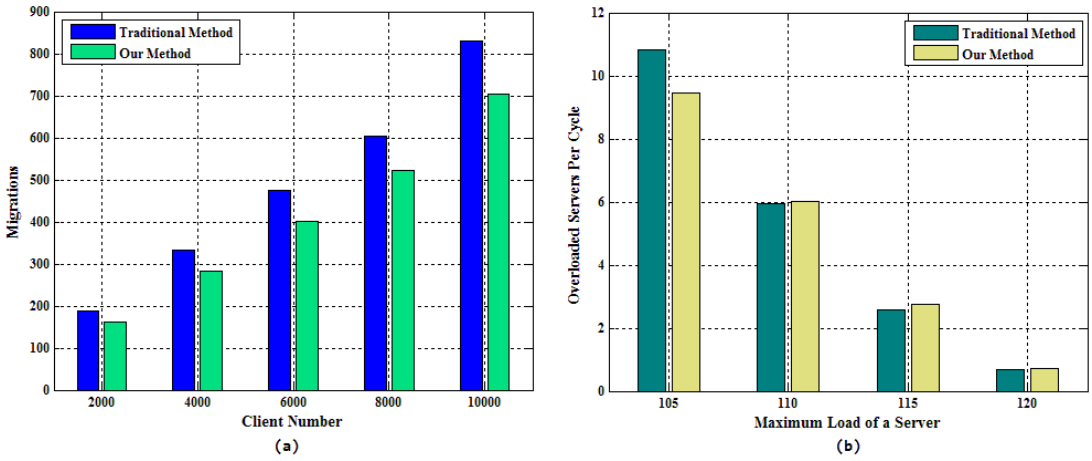

Figure 3. Comparison results of migration times and overloaded servers

Since then, we compared the statistics of overload servers of different methods. We placed 10,000 clients in the virtual world and also run 1,000 cycles. The results are shown in Figure 3.b. In Figure 3.b, the x-axis represents the maximum load that can be accommodated by a single server, varying from 105 to 120, with the y-axis representing the overloaded servers occurring per cycle. The green bars represent traditional load balancing methods, and the yellow bars represent the dynamic load balancing approach presented in this paper. From the results, we can see that compared with the traditional method, the results of the method of this paper are quite the same. The above experiments show that the dynamic load balancing method proposed in this paper can effectively reduce the probability of client migrations while guaranteeing balanced load during run-time, thus improving the usability of the virtual environment system.

Conclusions

In a multi-server infrastructure DVE system, to avoid the impact of overloaded servers on system usability, it is necessary to run a dynamic load balance process among multiple servers at runtime. In the process, the efficiency of the algorithm and the effectiveness of the result are two factors which need to be considered as priority. Existing methods mainly aim at these two factors. However, the effect of load balancing on client migration probability should also be taken into account in this process. If the client frequently migrates among multiple servers, on the one hand it will increase the communication overhead between servers, on the other hand it will affect the effectiveness of load balancing results. Therefore, in this paper, we propose a new dynamic load balancing method, which can be added to the process of traditional load balancing optimization process, so as to reduce the probability of client migrations without hampering the efficiency and effectiveness of existing load balancing methods, thus improving the usability of large scale distributed virtual environment system.

Acknowledgement

The work described in this paper was supported by the National Natural Science Foundation of China (Grant No. 61303187).

References

[1] Tang Y H, Zhang B D, Wu J J, et al. Parallel architecture and optimization for discrete-event simulation of spike neural networks[J]. Science China-Technological Sciences, 2013, 56(2), pp. 509-517.

[2] Hou B, Yao Y, Wang B, et al. Modeling and simulation of large-scale social networks using parallel discrete event simulation[J]. Simulation-Transactions of The Society for Modeling and Simulation International, 2013, 89(10) , pp. 1173-1183.

[3] Stytz M R. Distributed virtual environments[J]. IEEE Computer Graphics and Applications, 1996, 16(3): 19-31.

[4] Berger M J, Bokhari S H. A partitioning strategy for nonuniform problems on multiprocessors[J]. IEEE Transactions on Computers, 1987, 100(5): 570-580.

[5] Simon H D. Partitioning of unstructured problems for parallel processing[J]. Computing Systems in Engineering, 1991, 2(2-3): 135-148.

[6] Lee K, Lee D. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution[C]. Proceedings of the ACM symposium on Virtual reality software and technology. ACM, 2003: 160-168.

[7] Lui J C S, Chan M F. An efficient partitioning algorithm for distributed virtual environment systems[J]. IEEE Transactions on parallel and distributed systems, 2002, 13(3): 193-211.

[8] Deng Y, Lau R W H. Heat diffusion based dynamic load balancing for distributed virtual environments[C]. Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology. ACM, 2010: 203-210.

[9] Ng B, Si A, Lau R W H, et al. A multi-server architecture for distributed virtual walkthrough[C]. Proceedings of the ACM symposium on Virtual reality software and technology. ACM, 2002: 163-170.

[10] D. Ta, S. Zhou, W. Cai, X. Tang, and R. Ayani, "Efficient zone mapping algorithms for distributed virtual environments," 23rd Workshop on Principles of Advanced and Distributed Simulation, 2009, pp. 137–144.

[11] Zhang W, Xi M, Zhou H, et al. An optimal mapping algorithm based on balanced load cutting for DVE systems[C]. Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on. IEEE, 2009, 2: 429-432.

[12] W. Zhang, H. Zhou, Y. Peng, "An Irregular cutting based distributed mapping algorithm for DVE systems," Computational Intelligence and Industrial Applications, 2009. PACIIA 2009. Asia-Pacific Conference on. IEEE, 2009, pp. 215-219.

[13] Zhang W, Zhou H. A dynamic mapping method to keep region connectedness in DVE systems[C]. Information Technology, Networking, Electronic and Automation Control Conference, IEEE, 2016: 338-341.

[14] Zhang W, Zhou H. A Dynamic Mapping Method for Reducing Migrations in DVE Systems[C]. To appear in IAEAC 2017.