An Algorithm for Allocating DVE Environment with Objects Consideration for Heterogeneous System

Amany Sarhan
Associate Professor at
Computers and Control
Engineering, Tanta University,
Egypt

ABSTRACT
Distributed Virtual Environment (DVE) systems model and simulate the activities of thousands of entities interacting in a virtual world over a wide area network. These systems are composed of many servers each of which is responsible to manage multiple clients who want to participate in the virtual world. Each server delivers the information updated from different clients to other client in virtual world. Previous algorithms were proposed for balancing the workload among the servers of the DVE. However, these algorithms did not take into consideration active objects found in the virtual environment which affects the calculations of system cost. They also assumed homogenous environment where all servers have the same capabilities and all links have the same speed. This paper presents a partitioning algorithm that takes into account the active objects and a modified object layering algorithm that concentrates only the boarder to improve the performance (total cost of the system and execution time) of Distributed Virtual Environment. This paper also generalizes the system to be heterogeneous in servers' speed and link capacity. The evaluation results show that the performance of the allocation algorithm is significantly improved where the total system cost was reduced.

General Terms
Distributed system, Partitioning algorithm, Distributed virtual environments.

Keywords
Distributed virtual environment, scalability issue, partitioning algorithm, load balancing, communication reduction, linear optimization.

1. INTRODUCTION
Virtual environments are gaining more popularity with the wide usage of computer and Internet applications. It can be used in many areas like game and learning [1]. There are many types of the virtual environment; the first one is called Collaborative Virtual Environments (CVEs) describes virtual environments that involve more than one user, with avatars interacting with each other. With increased bandwidth and more available Internet access, virtual environments that allow for greater multi-user interactivity have become more widely available in recent years [1]. The second type is called Immersive Virtual Environments (IVEs) which perceptually surrounds the user in a way that increases the user's sense of being a part of it. IVEs requires special equipments such as a head mounted display or project equipment located in a room or any place. The third type is called Virtual Reality (VR) which gives the ability to see the real world and the virtual world at the same time [1]. The last type is called Distributed Virtual Environment (DVE). It is a simulated world which runs not on one computer system but on several computers. The computers are connected over a network (mostly the Internet) and users of those computers interact with each other and with environment in real time, sharing and altering the same virtual world [1].

There are two possible architectures for implementing a DVE system: 1) single server distributed virtual environment architecture (SSDVE) and 2) multiple servers distributed virtual environment architecture (MSDVE). The size of the virtual world and the average number of users of this virtual world determines which architecture to use. Under SSDVE architecture, all clients are connected to a powerful single and dedicated server. To guarantee that all users have the same consistent view of the virtual world, the system should report any action or activity generated by any user in real time to all users. Sometimes to minimize the communication cost, the system only deliver these changes to those users who can be affected by this new activity in their local view of this virtual world. SSDVE architecture is only suitable for a small scale DVE system, for example, a virtual world with a small number of objects and a small number of participating users.

In an MSDVE architecture, multiple servers are used where each server is responsible for handling a subset of the virtual world (e.g., some number of users and some number of objects in the virtual environment), as well as the communication of its attached users and the communication between servers. It is important to point out that, in order to keep the view consistency among the participating clients, some form of server-to-server communication is necessary. Large scale distributed virtual environments (DVEs) have become a major trend in distributed applications. Peer-to-peer (P2P) architectures have been proposed as an efficient and truly scalable solution for these kinds of systems. However, in order to design efficient P2P DVEs these systems must be carefully design to obtain good distribution of the environment resposibilities. Therefore, a DVE system designer has to consider the issue of balancing the computational workload among different servers and reducing the communication cost between different servers [2,3,4].

There are many problems in DVE like awareness problem [5], discard information [6], latency problem [5,7] , QOS problem [5], multicast protocol problem [7], unreliable transport mechanisms problem [7], avatar migration problem [8], and partitioning problem [2,3,7,9,10,11]. There are many ways try to

solve the partitioning problem in order to minimize the impact of network traffic on the performance of the DVE system. Lui and Chan [2,9] have shown the key role of finding a good assignment of avatars (users) to servers in order to ensure a minimum network traffic in DVE system. However, their algorithm did not take into consideration the active objects found in the virtual world which causes an additional cost to the system on both sides: computation and communication.

The main contribution of this paper is to propose enhanced DVE allocation algorithms with objects taken into consideration and a modified layering algorithm that decreases the execution time for the partitioning process. This will allow the dynamic usage of the algorithm when it is used on run-time load changes to adjust the workload and communication cost of the system. The paper also deals with heterogeneous system constrains, where the servers' speeds and communication link capacities are heterogeneous by modifying the cost equations of the system to accommodate this assumption.

The paper is organized as follows: section 2 discusses the DVE system architecture. Section 3 contains the proposed object allocation algorithms. Section 4 introduces the Modified Object Layering Partitioning (M-OLP) algorithm. Section 5 discusses the problem of using heterogeneous environment and presents a modification of the balancing condition equations. Experimental results are given in section 6 with their explanation. Finally conclusions are drawn in section 7.

2. DISTRIBUTED VIRTUAL ENVIRONMENT SYSTEM

2.1 System Architecture

The architecture of the DVE system is shown in Figure 1 which shows a DVE system with multiple servers. Each server is responsible of handling the activities of multiple users. All the multiple servers view and access the same virtual environment that contains avatars and objects.

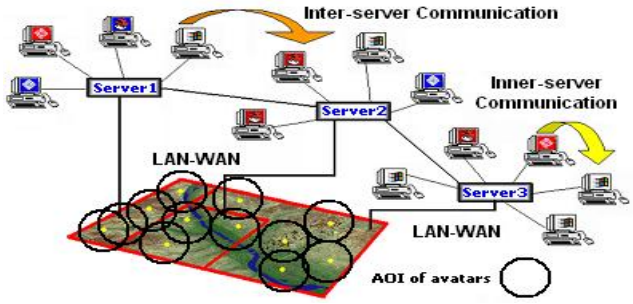

Fig. 1: Multiple server architecture for a DVE system

System users communicate via inter-server communication, if each of them is the responsibility of different server, or via inner-server communication, if both of them were the responsibility of the same server as shown in figure. The inner-server communication is too small compared to external communication so it will be neglected in the cost equation as it is performed internally on the server.

2.2 Avatar and Area-of-Interest (AOI)

The avatar is a 3D active object that represents a user of the system of the virtual world. In order to provide the interactive capability of a user, the avatar can move in a virtual world. The user uses his/her avatar to communicate with other avatars (or other users in the virtual world) or use his/her avatar to access any 3D objects, such as books, chairs, glasses, etc., in the virtual environment.

Since an avatar can move and interact with any static or dynamic 3D objects within the virtual world, a DVE system needs to transfer the information for this activity to other avatars so as to keep the information of the virtual world consistent. In general, each avatar only needs to know those activities that happened near his/her vicinity only. Figure 2 illustrates the area-of-interest (AOI) concept of different avatars. We use a circle to represent the AOI of each avatar as in [2,9].

Let AOI(Ai) be the area-of-interest of avatar A_i. From the figure, we can see that AOI (A_4) € AOI(A_1). Therefore, the DVE system has to send avatar A1 any activity generated by avatar A_4. On the other hand, if there is any activity happened within the intersection of AOI(A_1) and AOI(A_3), then the DVE system only needs to send to avatars A_1 and A_3 about this activity.

Since AOI(A_5) or AOI(A_6) does not intersect with the AOI of A_1 to A_4, the DVE system does not have to send to avatars A_1, A_2, A_3, and A_4 of any activity generated by avatars A_5 or A_6 [2,3].

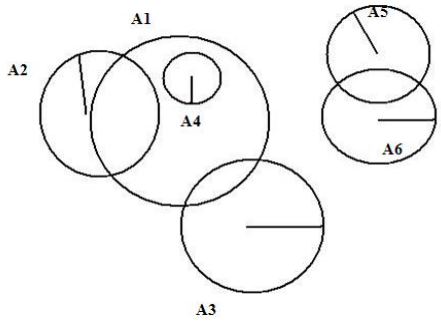

Fig. 2: Area-of-interest (AOI) of avatars

3. THE PROPOSED OBJECT-DRIVEN ALLOCATION ALGORITHMS

The main problem in the area of multiple servers of virtual environment is how to distribute the avatars to the servers such that they are balanced and the total system cost is minimized. In previous work, Recursive Bisection Partitioning (RBP), Layering Partitioning (LP), and Communication Refinement Partitioning (CRP) algorithms were presented [2]. However, their work was primitive as it did not consider the active objects available in the environment when partitioning the system to the servers. This causes additional workload and communication cost of the system as we will show by example.

In our work we intend to modify the previous allocation algorithms by taking into account the objects during the steps of allocation. We will also modify the layering algorithm to consider only the boarder avatars to minimize the allocation process execution time. Finally, we will extend the whole model to be suitable for heterogeneous system where the servers and

network links speed are not equal. The proposed algorithms will be described in the following subsections.

3.1 Object Recursive Bisection Partitioning (O-RBP) Algorithm

In this step, the main idea of the Object Recursive Bisection Partitioning (O-RBP) algorithm is to divide the avatars and objects in the virtual environment into p groups where p represents number of servers. The difference between the avatars and the objects is that the avatars send and receive messages if there is any action between them but the objects receive messages only [10].

The idea of considering the objects was given in [10]; however, they did not introduce the equations required to compute the communication and workload costs. Here, we add the cost of objects to the cost equations to be as follows:

$$C_P^W = \sum_{J=1}^{P}\Big(\Big|\sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - \omega * |J=1P(o\acute{\imath}=V/\omega o\acute{\imath} + \Psi o\acute{\imath} - \omega *|)\Big) \quad (1)$$

Where:

$$\omega^* = \Big(\sum_{i=1}^{n}[\omega(a_i) + \Psi(a_i)] + \sum_{i=1}^{n}[\omega(o_i) + \Psi(o_i)]\Big)/p$$

is the computational workload per server under the perfectly balanced workload partition strategy. The communication cost between the servers is calculated using equation 2 with considering the AOI of each avatar. Specifically, if an avatar a_i lies within the AOI of another avatar a_j, so avatar a_i performs any action, the DVE system must send this new information to avatar a_j and the object o_i if exists.

Let P be given a partition strategy which divides V into P partitions: $\{V_1, V_2 \ldots V_P\}$ and thus we assign partition V_i to server S_i. Let $\zeta(a_i, a_j)$ denote the amount of information exchanged (in unit of bit) from avatar a_i to avatar a_j and $\zeta(a_i, o_i)$ denote the amount of information exchanged (in unit of bit) from avatar a_i to the object o_i.

Let $\phi_{S_i,S_j}()$ be a non decreasing function that determines the amount of information exchange (in unit of bits) to the communication cost server from S_i to S_j in the DVE system. The communication cost between partition V_l and V_m (for $l \neq m$), denoted as C_{lm} [5], can be expressed as:

$$Clm = \sum_{a_i \in V_l}\sum_{a_j \in V_m}\{\phi_{S_l,S_m}(\zeta(a_i, a_j))\} + \sum_{a_j \in V_m}\sum_{a_i \in V_l}\{\phi_{S_m,S_l}(\zeta(a_j, a_i))\} + \sum_{a_i \in V_l}\sum_{o_j \in V_{lm}}\{\phi_{S_l,S_m}(\zeta(a_i, o_j))\} \quad (2)$$

The first term of the above equation represents the communication cost for transmitting information of the updates from partition V_l to V_m between two avatars. The second term represents the communication cost for transmitting information of the updates from partition V_m to V_l between two avatars and the third term represents the communication cost for transmitting information of the updates from partition V_m to V_l from avatar to object.

Let C_P^L be the communication cost of a given partition strategy P, where C_P^L is computed as:

$$C_P^L = \sum_{l=1}^{P}\sum_{m>l}^{P} C_{lm} \quad (3)$$

Therefore, C_P^L represents the total server-to-server communication cost of a given partition P. This assumption can be easily relaxed and can be included in the total cost C_P as in [2]. The overall cost of the partition strategy P (C_P) can be expressed as:

$$C_P = W_1 C_P^W + W_2 C_P^L \quad (4)$$

W_1 and W_2 represent the relative importance of the computational workload cost and the communication cost, respectively ($W_1 + W_2 = 1$) [2]. Finally, the DVE partitioning problem is to find an optimal partition P* such that:

$$C_P^* = \min_P\{C_P\} \quad (5)$$

The following steps are used to calculate partitioning of objects and avatars to the servers using the O-RBP algorithm:

Step1: Let the first server contains all cells while the second server is empty.

Step2: Move a single cell (containing avatars and objects) to the second server. Then calculate the cost of the system (containing avatars and objects) as given by equation 4.

Step3: Move another cell to the second server.

Step4: Repeat step 3 until all cells move to the second server.

According to the minimum calculated cost, the loading of the servers is calculated. Note that CPRBP (0) and CPRBP (N) represent the two extremes of the highest load imbalanced cost (i.e., all cells are assigned to one server and there is no server-to-server communication). So, CPRBP (0) and CPRBP (N) aren't taken into consideration. Note that; for a larger number of P, we can first use the bisection partitioning algorithm shown before, after that a partition is chosen that has the lowest cost and finally apply the Recursive Bisection partitioning algorithm again [2,10].

3.2 Object Layering Partitioning (O-LP) Algorithm

The main idea of the Object Layering Partitioning algorithm (O-LP) is to minimize the workload cost calculated from the O-RBP algorithm. This can be done by labeling each avatar and object using a server number. The label (or server number) serves as a possibility of moving the corresponding avatar and object to a new partition which has that server number [2,10]. The goal of this algorithm tries to minimize the workload cost of the system. We can apply this algorithm more than once to move avatars and objects to reduce the workload cost. The following are the steps of the O-LP algorithm:

Algorithm:

Begin
For $v_i \in V_{LP}$ do {
/* Create edges and mark those nodes along the partition boarder as connected */
For $v_j \in V_{LP}$ where $i \neq j$, do {
If v_j is within the AOI of v_i then {
Create an edge e_{ji} in E_{LP} ; /* e_{ji} is an edge between v_j and v_i where v_j , $v_i \in V_{LP}$ */
Set the weight of $e_{ji} = z(v_j , v_i)$;
If (server_num[v_i] ≠ server_num[v_j]) then {
Connected[v_i] ≠connected[v_j] = true; } }
If oj is within the AOI of vi then {
Create an edge eji in ELP ; /* eji is an edge between oj and vi where oj , vi ∈VLP */
Set the weight of eji = z(oj , vi);
If (server_num[vi] ≠ server_num[oj]) then {
Connected [vi] ≠connected [oj] = true;}}}}
For $v_j \in V_{LP}$ do {
/* Create edges and mark those nodes along the partition Boarder as connected */
For $v_i \in V_{LP}$ where $j \neq i$, do {
If v_i is within the AOI of v_j then
Create an edge e_{ij} in E_{LP} ; /* e_{ij} is an edge between v_i and v_j where v_i , $v_j \in V_{LP}$ */
Set the weight of $e_{ij} = z(v_i , v_j)$;
If (server_num[v_j] ≠ server_num[v_i]) then {
Connected [v_j] ≠ Connected [v_i] = true;} }
For oi ∈ VLP where j ≠ i, do {
If oi is within the AOI of vj then
Create an edge eij in ELP ; /* eij is an edge between oi and vj where oi , vj ∈VLP */
Set the weight of eij = z(oi , vj);
If (server_num[vj] ≠ server_num[oi]) then {
Connected [vj] ≠ Connected [oi] = true;} } }
For all $v_i \in V_{LP}$ do{ /*connect the remaining nodes*/
If (connected[v_i] = false) then {
If ((there exists a node v_j which is a neighbor of v_i) /* v_j is a neighbor of v_i if e_{ij} exists */
And (connected[v_j] = true)) then
Connected[v_i] = true;
If (connected[v_i] = false) do {
Find a nearest node v_k such that connected[v_k] = true and server_num[v_j] = server_num[v_k];
Create an edge $e_{ik} \in E_{LP}$;
Set weight of $e_{ik} = z(v_i , v_k)$;
Connected[v_i] = true;}
If (connected[v_i] = false) do {
Find a nearest node $v_k \in G_{LP}$ such that connected[v_k] = true
Set weight of $e_{ik} = z(v_i , v_k)$;
Connected[v_i] = true;}}
If (connected[vi] = false) do {
Find a nearest node oj ∈ GLP such that connected[oj] = true
Set weight of eik = z(vi , oj);
Connected[vi] = true;}}}
End

3.3 Object Communication Refinement Partitioning (O-CRP) Algorithm

The objective of the Object Communication Refinement Partitioning (O-CRP) algorithm is to reassign nodes (avatars and objects) to another partition so as to reduce the server-to-server communication cost. The boarder nodes only (avatars and objects) under the constraint (external weight is larger than the internal weight) are moved to another partition [10].

3.4 Importance of considering objects during partitioning

To illustrate the importance of considering the objects during the partitioning process, we will use a given environment where the number of avatars was assumed to be =21 (represent it with a circle), number of objects=12(represent it with a square), the number of servers =3, and the number of cells =9. Let the workload weighting (W_1) and the communication cost weighting (W_2) =0.5. Assume the workload for maintaining an avatar and object is 10 and that the three servers have the same speed. The environment to be partitioned with avatars and objects distributed is as shown in Fig. 3 which is required to be partitioned over 3 identical servers.

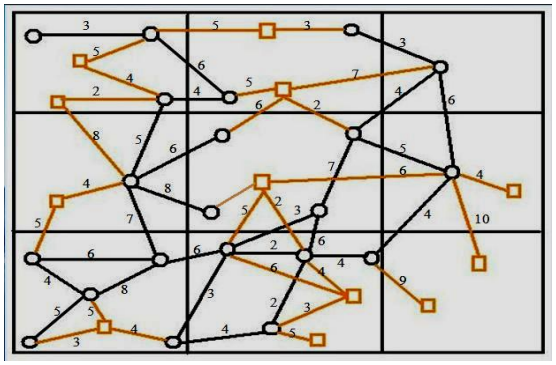

Fig. 3: Environment containing 21 avatars and 12 objects

First, we will use the Lui and Chan algorithms [2] which omit the objects from the environment before partitioning. Then, we will see the effect of omitting the objects from the partitioning on the total system cost as follows.

3.4.1 Allocation considering only the avatars (Lui & Chan algorithms)

When we use Lui and Chan algorithms [2], we only consider the avatars available in the virtual world so the cells partitioned will be seen by the algorithms as shown in Fig. 4 with the objects omitted from Fig. 3.

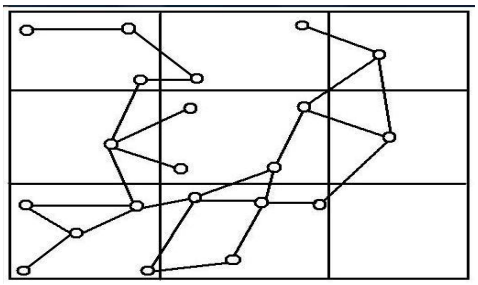

Fig. 4: Environment containing 21 avatars after omitting the objects from the environment in fig. 3

The first step divides the number of avatars into partitions to arrive to minimum cost of the system as follows:

1. Assign the first cell only to the first server and remaining cells to the second server then calculate the total cost of the system.

2. Repeat this step but assign two cells to the first server and the remaining cells to the second server and so on until we reach the minimum cost of the system.

3. Divide the first server which contains the larger number of avatars into two partitions as follows:

   3.1 Assign the first cell to the first server, three cells to the second server and the remaining cells to the third server then calculate the total cost of the system.

   3.2 Assign two cells to the first server, two cells to the second server and the remaining cells to the third server then calculate the total cost of the system and so on.

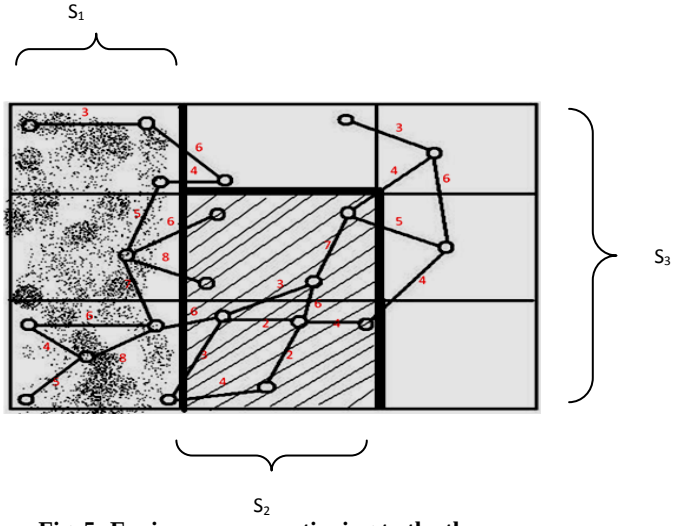

Fig. 5: Environment partitioning to the three servers

The minimum cost was reached when the first server contains nine avatars (three cells), the second server contains eight avatars (two cells) and the third server contains four avatars (four cells) as shown in Fig. 5.

The total cost of the system partitioned as above is:

$$C_P = W_1 * C_P^W + W_2 * C_P^L$$

Workload variance:

$$C_P^W = \sum_{J=1}^{P}(\left|\sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - \omega^*\right|)$$

$$C_P^W = 10 *(|9 - 7| + |8 - 7| + |4 - 7|) = 60$$

Communication cost:

$$C_P^L = C_{12} + C_{21} + C_{13} + C_{31} + C_{23} + C_{32} = 69$$

$$C_P = 0.5 * 60 + 0.5 * 69 = \underline{64.5}$$

When applying the LP algorithm [2], we need to move two avatars from the first server to the third and move one avatar from the second server to the third as shown in Fig. 6. The total cost of the system partitioned is:

$$C_P = W_1 C_P^W + W_2 C_P^L = \underline{45.5}$$

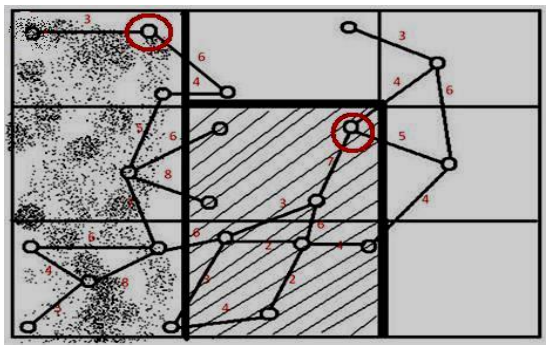

Fig. 6: The avatars that should be moved to the third server

When we moved this avatar, the partitioning becomes as shown in Fig. 7 and 8. The total cost of the system partitioned is: $C_P = \underline{33}$.

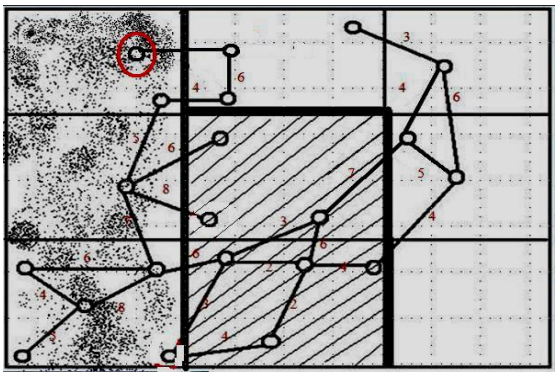

Fig.7: Another avatar should be move to the third partition

Finally we apply the CRP algorithm [2] to minimize the communication cost. To reduce the communication cost of the system, we should move the circled avatars between the two servers as shown in Fig. 9 and 10. The total cost of the system becomes: $C_P = \underline{27}$. So, the total cost of the system after applying the three algorithms is $\underline{27}$.

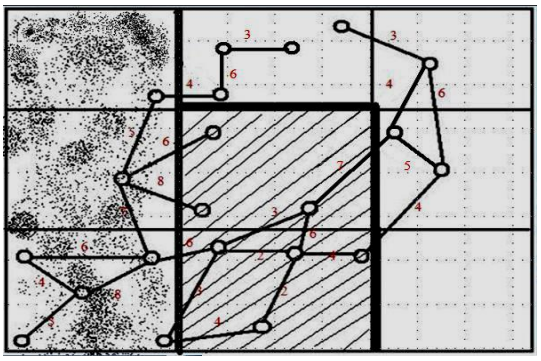

Fig. 8: The partitioning after applying LP algorithm

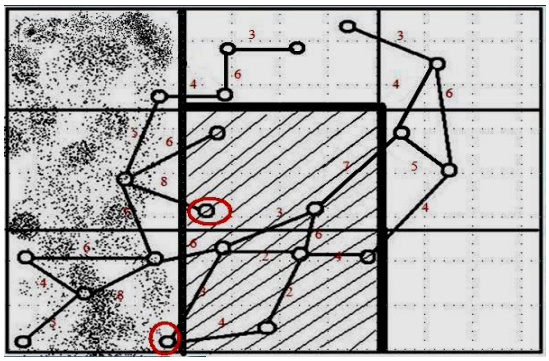

Fig. 9: The avatars that should be moved to apply CRP algorithm

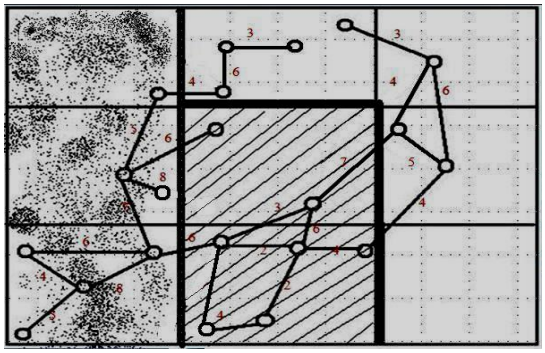

Fig. 10: The final partitioning after applying CRP algorithm

3.3.2 Computing the effect of objects on the cost

When we return the objects to the cells, the actual partitioning of the cells to the three servers is as shown in Fig. 11.

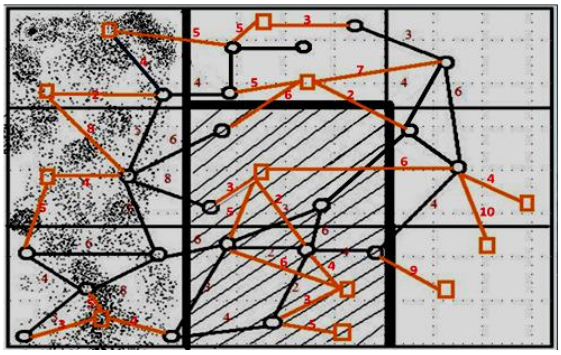

Fig. 11: The partitioning after returning the objects to cells

The actual load and communication of the servers should be recomputed after considering the effect of objects in the system as:

$$C_P^W = \sum_{J=1}^{P} \left(\left| \sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - \omega^* \right| \right)$$

$$C_P^W = 10 * (|13 - 7| + |11 - 7| + |9 - 7|) = 120$$

$$C_P^L = C_{12} + C_{21} + C_{13} + C_{31} + C_{23} + C_{32}$$

= (6+6+3)+(6+8+3+4)+(4)+(4+5)+(6+4+4)+(4+6) = 73

C_P = 0.5 * 120 + 0.5 * 73 = **96.5**

From these results, it is clear that neglecting the effect of objects during the partitioning leads to an increase in the actual system cost (from 27 to 96.5). This confirms the effectiveness of the proposed algorithms in reducing the system cost.

3.5 Applying the proposed algorithms on an illustrating example

In the illustrating environment example, we will use the same environment described in Fig. 3. We assume the workload for maintaining an avatar or an object is 10. The three servers have the same speed.

The first step is to apply the proposed O-RBP that divides the number of avatars and objects into partitions to arrive to minimum cost of the system as discussed below.

1. Assign the first cell only to the first server and remaining cells to the second server then calculate the total cost of the system.
2. Repeat this step but assign two cells to the first server and the remaining cell to the second server and so on.
3. The minimum cost is obtained when the first server contains 13 avatars and 6 objects (four cells) and the second server contains 8 avatars and 6 objects (five cells).
4. Divide the first server which contains the larger number of avatars into two partitions.
   - 4.1 Assign the first cell to the first server, three cells to the second server and the remaining cells to the third server then calculate the total cost of the system.
   - 4.2 Assign two cells to the first server, two cells to the second server and the remaining cells to the third server then calculate the total cost of the system and so on.

The final result after partitioning to the three servers, as shown in Fig. 12, where the first server contains four avatars and three objects (two cells), the second server contains nine avatars and three objects (two cells) and the third server contains eight avatars and six objects (five cells).

The total cost of the system partitioned is:

$$C_P = W_1 * C_P^W + W_2 * C_P^L$$

$$C_P^W = \sum_{J=1}^{P} \left(\left| \sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - \omega^* \right| \right)$$

$$C_P^W = 10 * (|7 - 11| + |12 - 11| + |14 - 11|) = 80$$

$$C_P^L = C_{12} + C_{21} + C_{13} + C_{31} + C_{23} + C_{32}$$

= 7+(7+5)+(5+4+6)+(6+6+8)+(5+2+4)+(3+4) = 72

C_P = 0.5 * 80 + 0.5 * 72 = 76

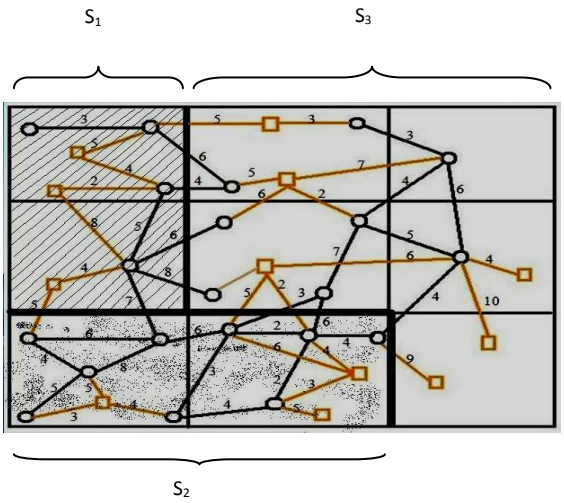

Fig. 12: Partitioning the system into three servers considering the objects

The minimum cost here is 76. When applying the O-LP algorithm, we need to move one avatar from the second server to the first and move three avatars from the third server to the first. We can't move the avatar from the second server to the first because the external weight is less than the internal weight as shown in Fig. 13.

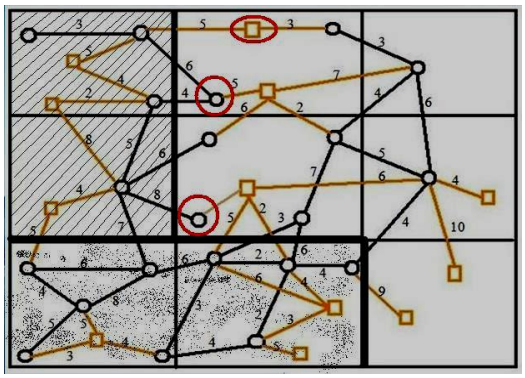

Fig. 13: The avatars should be moved to the first server

After applying O-LP algorithm as shown in Fig. 14, the cost becomes C_P= 41. We do not need to apply O-CRP in this example because the external communication weights of the avatars and objects are less than the internal weights. So the final partitioning is given in Fig. 15 with cost = 41 which is much smaller than the portioning produced by [2].

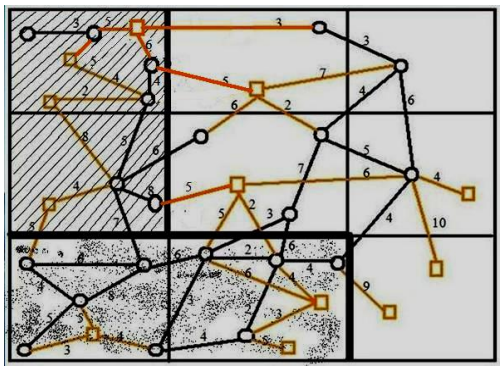

Fig. 14: The system after applying LP algorithm

4. THE MODIFIED OBJECT LAYERING PARTITIONING (M-OLP) ALGORITHM

This algorithm is based on the object layering algorithm (O-LP) with the aim to reduce the execution time of the algorithm which makes it suitable for making any improvements in the partition if the environment status is varied.

The proposed M-OLP algorithm, in order to reduce the partitioning time, will only consider the boarder avatars and objects when performing the movement from one partition to the other. The boarder avatars/objects are defined to be the avatars/objects in the partition that communicate with the avatar or/and objects in another partition. The non boarder avatars/objects will not be checked for possible movement.

The steps of the M-OLP algorithm are given below:

01. Begin
02. Initialize variables max_par=0; min_par=0
03. For each avatar a_i, create a node v_i in G_{LP};
04. For each $v_i \in G_{LP}$, do {/* initiate */
05. Initialize variables Connected [v_i] = false;
06. Initialize variables server_number[v_i] = k where $v_i \in V_k$ and $1 \leq k \leq P$;
07. /* note that the server index k for V_k can be obtained from the output of the RBP algorithm */ }
08. For each object o_j, create a node o_j in G_{LP};
09. For each $o_j \in G_{LP}$, do {/* initiate */
10. Initialize variables Connected [o_j] = false;
11. Initialize variables server_number[o_j] = k where $v_i \in V_k$ and $1 \leq k \leq P$;}
12. While (max_par - min_par or optimization reach>0) {
13. For $v_i \in V_{LP}$ do {
14. /* create edges and mark those nodes along the partition boarder as connected */
15. For $v_j \in V_{LP}$ where $i \neq j$, do {
16. If v_j is within the AOI of v_i then {
17. Create an edge e_{ji} in E_{LP}; /* e_{ji} is an edge between v_j and v_i where $v_i, v_j \in V_{LP}$ */
18. Set the weight of $e_{ji}=\zeta(v_j, v_i)$;
19. If (server_num[v_i] ≠server_num[v_j])Then {
20. Connected [v_i] = connected [v_j] = true ;}}}
21. For $o_j \in V_{LP}$ where $i \neq j$, do {
22. If o_j is within the AOI of v_i then {
23. Create an edge e_{ji} in E_{LP}; /* e_{ji} is an edge between o_j and v_i where $v_i, o_j \in V_{LP}$ */
24. Set the weight of $e_{ji}=\zeta(o_j, v_i)$;
25. If (server_num[v_i] ≠server_num[o_j])Then {
26. Connected [v_i]= connected[o_j] = true; }}}}
27. For $v_j \in V_{LP}$ do {

28. ***/* Create edges and mark those nodes along the partition boarder as connected */***
29. ***For v_i Є V_{LP} where j ≠ i, do {***
30. ***If v_i is within the AOI of v_j then {***
31. ***Create an edge e_{ij} in E_{LP} ; /* e_{ij} is an edge between v_i and v_j where v_i , v_j ЄV_{LP} */***
32. ***Set the weight of $e_{ij} = z(v_i , v_j)$;***
33. ***If (server_num[v_j] ≠ server_num[v_i]) then {***
34. ***Connected[v_j] ≠Connected[v_i] = true; }}}***
35. ***For o_i Є V_{LP} where j ≠ i, do {***
36. ***If o_i is within the AOI of v_j then {***
37. ***Create an edge e_{ij} in E_{LP} ; /* e_{ij} is an edge between v_i and v_j where v_i , v_j ЄV_{LP} */***
38. ***Set the weight of $e_{ij} = z(o_i , v_j)$;***
39. ***If (server_num[v_j] ≠ server_num[o_i]) then {***
40. ***Connected[v_j] ≠Connected[o_i] = true; }}}}***
41. ***Get max_par , min_par then check optimization reach***
42. ***Move avatars***
43. ***Move objects***
44. ***End***

5. WORKING UNDER HETEROGENEOUS ENVIRONMENT

The previous work assumed that the system components are homogenous in both server and network link speed. In real life, this is almost impossible. Thus, an extension of the balancing equations should be made to improve the allocation algorithms performance that aims to achieve the balance of the system components.

In this section we will extend the above work to include the heterogeneity of the system. First, we will assume that the P servers are different in speed. This assumption will affect the distribution of avatars and objects to the servers. High speed servers can be assigned larger number of avatars and objects than the lower speed servers without violating the partition workload balance's condition.

To illustrate how this heterogeneity will affect the distribution of avatar and objects to the servers, let's assume that we have 30 avatars and 3 identical servers (as assumed in [2,10]). The average distribution of avatars workload to the servers is computed using the equation:

$$Rv_i = \sum_{J=1}^{P} \left| \sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - R\omega^* \right| \qquad (6)$$

Where:

Rv_i : represents relative avatar's average.

$R\omega^*$: : represents relative server$_i$ speed and is computed as:

$$R\omega^* = \left(\sum_{i=1}^{n} [\omega(a_i) + \Psi(a_i)] * \frac{speed_i}{\sum_{j=1}^{p} speed_j} \right)$$

Where $speed_i$ represents server$_i$ speed and $\sum_{j=1}^{p} speed_j$ represents the sum of servers' speeds.

Thus, the average number of avatars in identical servers case computed as in Eq. (1) = 30/3 = 10 avatars.

When the servers are not identical in speed and their relative speeds are: 1, 2 and 3 for S_1, S_2 and S_3 respectively, the previous average computation will not be correct as the computation time of handling 10 avatars on the first server is 3 times the computation time of handling them on the third server.

According to this new assumption, the average number of avatars computed by the following equation is not correct:

$$C_P^W = \sum_{J=1}^{P} \left(\left| \sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - \omega^* \right| \right)$$

Where $\omega^* = \sum_{i=1}^{n} [\omega(a_i) + \Psi(a_i)] / p$ is the computational workload per server under the perfectly balanced workload partition strategy.

It should be modified to include the variation in the relative speed as follows:

$$Rv_i = \sum_{J=1}^{P} \left| \sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - R\omega^* \right| \qquad (7)$$

$$R\omega^* = \left(\sum_{i=1}^{n} [\omega(a_i) + \Psi(a_i)] * \frac{speed_i}{\sum_{j=1}^{p} speed_j} \right) \qquad (8)$$

So, the workload cost is computed using the following equation:

$$C_P^W = \sum_{J=1}^{P} \left(\left| \sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - Rv_i \right| \right) \qquad (9)$$

Where $Rv_i = \sum_{J=1}^{P} \left| \sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - R\omega^* \right|$

is the computational workload per server under the perfectly balanced workload partition strategy.
Thus if we partitioned the 30 avatars on the above three servers considering the variation in their speed as 5, 10 and 15 avatars for the three servers respectively, the variance in workload will be 0 which is the optimal case. This is computed as follows:

C_P^W = 10*{|(5–30)*1/6|+ |(10 – 30)*2/6| + |(15 – 30(*3/6|

= 10*{|5 – 5| + |10 -10| + |15 – 15|}

= 10 * {0 + 0 + 0} = 10 * 0 = 0

5.1 Illustrating Example

We perform the same steps of the O-RBP on the environment described above in Fig. 3 assuming that the servers are heterogeneous and the relative servers' speed is 3:2:1 (rather than equal in the previous sections).

After distributing the avatars and objects to the three heterogeneous servers, the minimum cost was found to be =59 (where the first server contains 19 avatars and objects, the second server contains 9 avatars and objects and the last server contains 5 avatars and objects) as shown in Fig. 15.

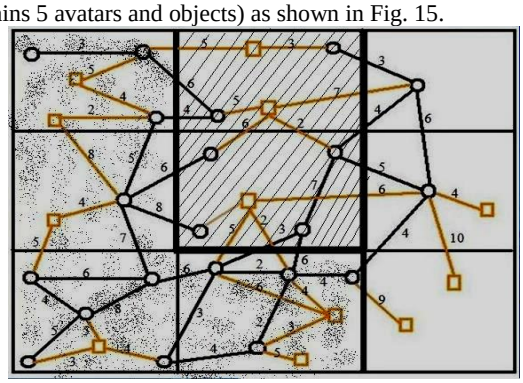

Fig. 15: System containing three servers

When we apply the O-LP algorithm, we can move two avatars from the first server to the second server to reduce the workload cost by two steps as shown in Fig. 16.

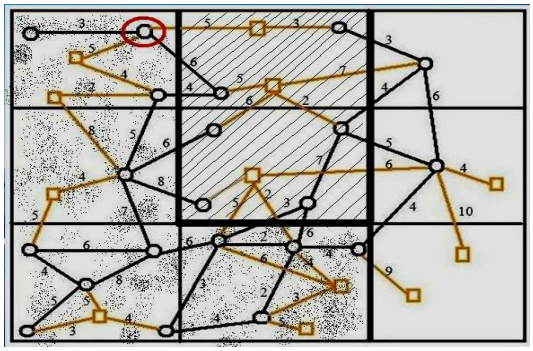

Fig. 16: Assigning the avatar which moves to the second server

The total system cost is computed to be:

$C_P = W_1 * C_P^W + W_2 * C_P^L$

$C_P^W = \sum_{J=1}^{P}(\left|\sum_{a_i=V_J} \omega(a_i) + \Psi(a_i) - \omega^*\right|)$

C_P^W = 10*(|(17-33)*3/6|+|(11-33)*2/6|+|(5-33)*1/6| = 10

$C_P^L = C_{12} + C_{21} + C_{13} + C_{31} + C_{23} + C_{32}$

=(3+4+6+5+2)+(5+4+6+8+3)+(4)+(4)+(3+4)+(3+5) = 69

C_P = 0.5 * 10 + 0.5 * 69 = 39.5

After moving the avatar, the system partitioning is shown in Fig. 17. We can also move another avatar to reduce the workload cost even more. After we move another avatar, we calculate the total cost of the system again. C_P = 38.

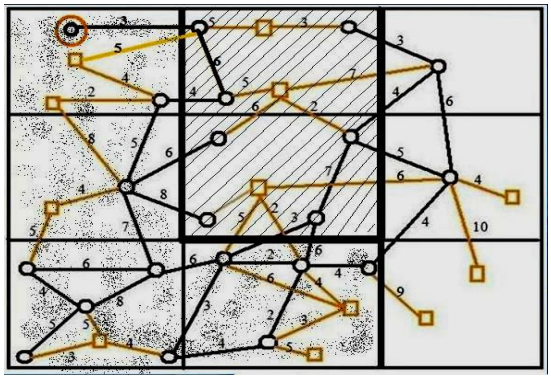

Fig. 17: Moving another avatar to the second server

This is the final partitioning on the three server generated by O-RBP and O-LP algorithms. When we apply O-CRP, the same system is produced as most of avatars and objects communicate in the same server larger than they communicate with other avatars and objects in other servers.

We note that the third server now has fewer cells while the first server has more cells than produced when the speeds were equal as the third server speed now i1 1/3 of the first server. However, this does not violate the balance of the system as each server will handle a number of avatars and objects proportional to its speed.

5.2 Link Speed Heterogeneity

The second parameter to consider in order to generalize the partitioning cost equations is the network link speed. Equation (2) that was used to compute the communication cost between partition Vl and Vm (for l≠ m), denoted as C_{lm} [5], can now be expressed as:

$$C_{lm} = \sum_{a_i \in V_l} \sum_{a_j \in V_m} \{\phi_{S_l,S_m} \frac{(\xi(a_i,a_j))}{LS_{l,m}}\} + \sum_{a_j \in V_m} \sum_{a_i \in V_l} \{\phi_{S_m,S_l} \frac{(\xi(a_j,a_i))}{LS_{m,l}}\} + \sum_{a_i \in V_l} \sum_{o_j \in V_m} \{\phi_{S_l,S_m} (\frac{z(a_i,o_j)}{LS_{l,m}})\} \quad (10)$$

The first term of the above equation expresses the communication cost for transmitting information updates from partition V_l to V_m between two avatars on a link with speed $LS_{l,m}$. The second term expresses the communication cost for transmitting information updates from partition V_m to V_l between two avatars on a link with speed $LS_{m,l}$ and the third term expresses the communication cost for transmitting information updates from partition V_m to V_l from avatar to object on a link with speed $LS_{l,m}$.

Let C_P^L be the communication cost of a given partition strategy P, where C_P^L is computed as:

$$C_P^L = \sum_{l=1}^{P} \sum_{m>l}^{P} C_{lm} \quad (11)$$

Therefore, C_P^L represents the total server-to-server communication cost of a given partition P. This assumption can be easily relaxed and can be included in the total cost C_P. The overall cost of the partition strategy P, denoted by C_P, can be expressed as

$$C_P = W_1 C_P^W + W_2 C_P^L$$

5.3 Experimental Results

In this section, the performance of the proposed algorithms is compared to the performance of the previous partitioning algorithms in [2]. We will investigate the effect of the number of objects on the cost of the system and the effect of number of avatars on the execution time of the partitioning algorithm under the different distribution scheme of the avatars and objects in the environment.

We will use three different methods to generate the position of each avatar in the virtual world. These methods are a) Uniform Distribution, b) Skewed Distribution, c) Clustered Distribution [2] as shown in Fig. 18.

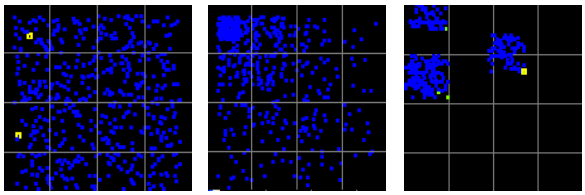

Fig. 18 a) Uniform Distribution, b) Skewed Distribution, c) Clustered Distribution

We have developed a simulation tool (written in c# 2008 express edition), intel@ coreTM 2 Duo processor T6400 (2.0 GHZ) ,320GB (5400RPM) hard Drive , and 4096MB DDR2 SDRAM (2 Dimm).

5.4 Effect of number of objects on the system cost

A virtual world with a dimension of 4 * 4 units is used in this experiment. The total number of avatars in this virtual world is equal to 100 and the number of servers P is equal to three. The workload weighting (W_1), and the communication cost weighting (W_2) are set to 0.5. The diameter of the AOI of each avatar is equal to 30. We change the number of objects from 0 to 100 to show the effect of neglecting the objects on the cost.

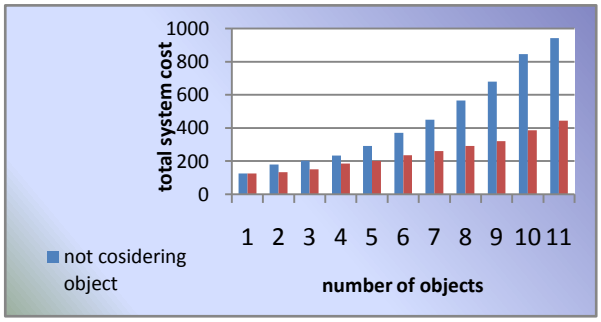

Fig. 19: System cost (CP) under variable number of objects using uniform distribution

We compare the proposed algorithms that consider the objects to the previous algorithm that neglects the objects [2]. Figures 19, 20 and 21 show the experimental results under the uniform, skewed, and clustered location distributions, respectively.

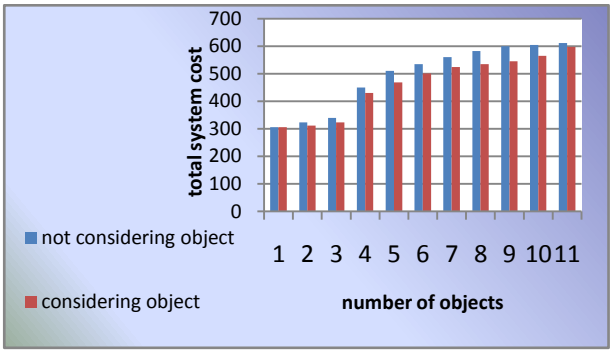

Fig. 20: System cost (CP) under variable number of objects using skewed distribution

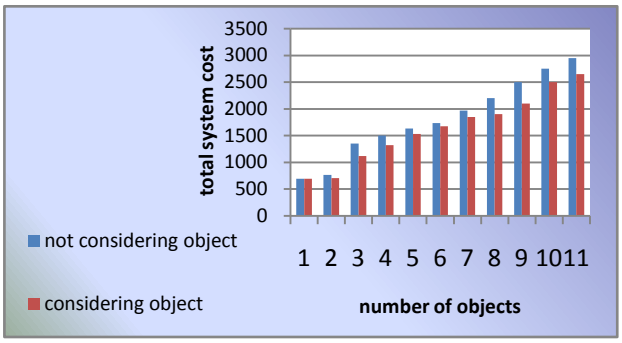

Fig. 21: System cost (CP) under variable number of objects using cluster distribution

These figures show that the total cost of the system decreases for any number of objects using the proposed algorithms than the cost if we did not consider the objects during the partitioning (using Lui and Chan algorithms in [2]) at the different types of distributions of objects. However, the system cost of the uniform distribution is less than the skewed and clustered distribution as the other two distributions concentrate the objects at certain cells making the workload high for the server handling them.

5.5 Effect of Number of Avatars on Execution Time

A virtual world with dimension 4 * 4 units is used in this experiment. The number of avatars is changed between 50 and 500 and the number of servers P is three. Both the workload weight W_1, and the communication cost weight W_2 equal 0.5. The diameter of the AOI of each avatar equals 30. In this experiments we study the effect of using the modified O-LP (MO-LP) on the execution time required for the partitioning process compared to the time required using the previous LP algorithm developed by [2].

Figures 22, 23 and 24 illustrate the execution time required for the partitioning process under the uniform, skewed, and clustered objects location distributions, respectively using the modified layering algorithm and the old layering algorithm by Lui and Chan.

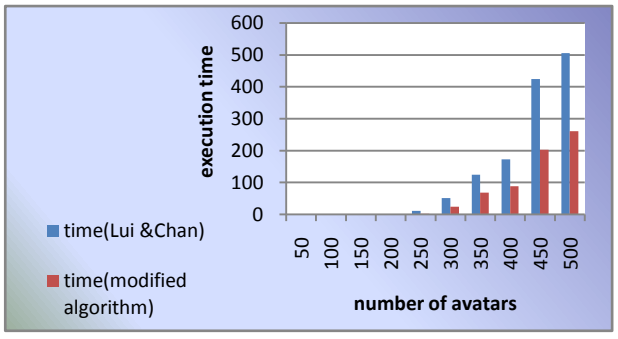

Fig. 22: Execution time under variable number of avatars using uniform distribution

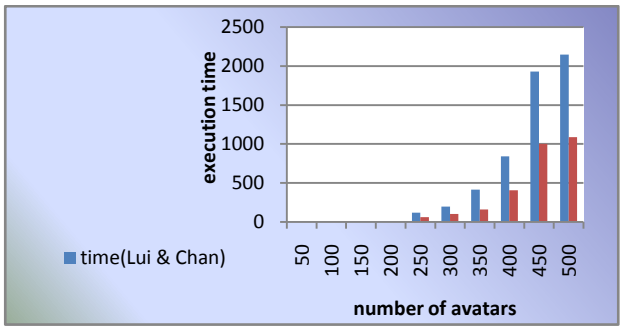

Fig. 23: Execution time under variable number of avatars using skewed distribution

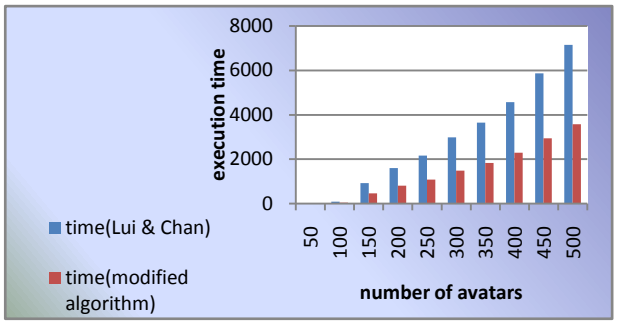

Fig. 24: Execution time under variable number of avatars using cluster distribution

The above figures show that the execution time using the MO-LP algorithm is less than the execution time of Lui and Chan algorithm (about 40%) for all cases of objects distribution. This is due that we only consider the boarder avatars and objects when moving an avatar or an object which reduces the execution time of movement. These figures also show that the algorithm takes less execution time in uniform distribution of objects and avatars than in skewed and clustered distributions

6. CONCLUSIONS

This paper discussed the partitioning problem in DVE. To build a scalable DVE system, we use multiple servers DVE architecture. Under the MSDVE architecture, there is a necessity to balance the computational workload and, at the same time, reduce the communication cost of a DVE system. The proposed algorithms that consider the objects for solving the partitioning problem in distributed virtual environments were presented. The performance results extracted from the experiments conducted for testing the efficiency of the algorithms show the importance of considering the objects when allocating the virtual environment to the servers. We also proposed a modified object layering algorithm to reduce the execution time of the partitioning process. The results show that the proposed algorithms reduce considerably the total system cost while achieving the balance among the servers.

The heterogeneity of the system was the third issue handled in this paper. Previous work assumed homogeneity of the servers and network link speeds. Neglecting this heterogeneity was proven to lead to imbalance of the system components. System balancing equations were modified to be suitable for heterogeneous system. This extension enables the allocation algorithms to be used on heterogeneous systems by modifying balancing equations and conditions used in the algorithms.

7. REFERENCES

[1] Dalgarno, B. and J. W. Lee, M. (2011) hat are the learning affordances of 3-D virtual environments?. British Journal of Educational Technology, 41:1, 10–32.

[2] Lui, J.C. and Chan M. (2002) An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems. IEEE Trans. Parallel and Distributed Systems, 13:3, 193-211.

[3] Morillo, P., Ordun J.M., Ferna´ndez, M. and Duato, J. (2005) Improving the Performance of Distributed Virtual Environment Systems. IEEE Trans. Parallel and Distributed Systems, 16:7, 637-649.

[4] Wang, L., Laszewski, G., Kunze, M., Tao, J., and Dayal, J. (2010) Provide Virtual Distributed Environments for Grid computing on demand. Advances in Engineering Software, 41:2, 213-219.

[5] Morillo, P., Rueda, PS., PJ. Ordun J.M., and Duato, J. (2007) A latency-aware partitioning method for distributed virtual environment systems. IEEE Trans. Parallel and Distributed Systems, 18:9, 1215–1226.

[6] Zhou, S., Cai, W., Lee, B., and Turner, S., (2004) Time-space consistency in large-scale distributed virtual environments. ACM Transactions on Modeling and Computer Simulation (TOMACS), 14: 1.

[7] Morillo, P., Rueda, S., Orduña, J.M. , and Duato, J. (2010) Ensuring the performance and scalability of peer-to-peer distributed virtual environments. Future Generation Computer Systems, 26:7, 905-915.

[8] Huang, J., Du, Y. and Wang, C. (2003) Design of the Server Cluster to Support Avatar Migration. IEEE Computer Society, 1087-1092.

[9] De Grande, R.E., Boukerche, A. and Ramadan, H.M.S. (2011) Decreasing Communication Latency through Dynamic Measurement, Analysis, and Partitioning for Distributed Virtual Simulation. IEEE Transactions on Instrumentation and Measurement, 60:1, 81 – 92.

[10] Bouras, C., Giannaka, E. and Tsiatsos, T. (2007) An Object Driven Partitioning Approach for Distributed Virtual Environments. IEEE computer society, 0-7695-3049-4.

[11] Morillo, P. and Ferna´ndez, M. (2003) A GRASP-Based Algorithm for Solving DVE Partitioning Problem. Proceedings of 2003 Int'l Parallel and Distributed Processing Symp. (IPDPS 2003), April.