Partitioning of Distributed Virtual Environments Based on Objects' Attributes

C. Bouras$^{1, 2}$, E. Giannaka$^{1, 2}$, Th. Tsiatsos$^{2, 3}$
^{1} Computer Engineering and Informatics Department, University of Patras, Greece
^{2} Research Academic Computer Technology Institute, Patras, Greece
^{3} Informatics Department of the Aristotelian University of Thessaloniki, Greece
{bouras, giannaka, tsiatsos}@cti.gr

Abstract

The partitioning constitutes one of the basic problems that need to be handled in Distributed Virtual Environments for maintaining consistency among users' view of the virtual scene and for optimizing the performance of the system. This paper presents an approach for handling the partitioning problem, which is based on objects' attributes. In particular, the approach proposes a technique, which takes into account the degree of interaction that the objects of the virtual environment carry for predicting future avatar behavior. This prediction of avatars' behavior based on objects' attributes defines the initial partitioning of the virtual space.

1. Introduction

Distributed Virtual Environments (DVEs) tend to become a de-facto solution for large-scale applications [1] and need to address more serious problems than the ones of single user virtual reality systems [6]. These problems are related, among others, to the control of network traffic, latency, and reliability. One of the key issues in the design of a scalable DVE system is the partitioning problem. The partitioning problem is related to the efficient assignment of the workload to the servers of the system. To this direction a lot of research work has been done and many algorithms and techniques have been proposed. In particular, the approach presented in [4] proposes a heuristic search based on Ant Colony Systems (ACS). Another approach [3] is to logically partition virtual environments by associating spatial, temporal, and functionally related entity classes with network multicast groups. Another different approach rejects dynamic concepts associated to avatars like *AoI* , aura or locale [5]. Finally, Lui and Chan [1] have described the importance of finding a good assignment of the participating clients to the available servers for managing the workload and the communication cost and for achieving a better networking performance. This partitioning algorithm currently achieves very good results for DVEs [2].

The approach presented in this paper aims to achieve an optimal initial partitioning of the virtual environment by exploiting objects' characteristics. In particular, the proposed algorithm on the one hand tries to balance the workload among the participating servers and on the other hand tries to minimize future communication need among them. Both the balancing of the workload as well as the prediction of the future state of the world take into account objects' attributes as well as the impact that these attributes have on users' behavior within the virtual environment.

The paper is structured as follows: Section 2 introduces the "degree of interaction" attribute, which is used for the partitioning approach. Section 3 presents the partitioning algorithm, while the Section that follows presents some experimental results. Finally, Section 5 concludes the paper and presents the planned next steps.

2. Degree of Interaction

In a virtual environment, users have the ability to communicate with each other as well as with the objects of the world for achieving the goals that each of the environment aims to support. The interactions that take place within a virtual environment are significantly affected by the number of actions allowed and supported by the system. In a world where only avatars exist, the interactions are mainly driven by the social behavior established between the participating users, which usually follow the basic principles of real world interactions. However, in a virtual environment, which, consists of objects as well, the behavior can be driven by the purpose of these objects and their role for achieving the goals that each environment aims to support. In particular, objects that allow a number of actions to be performed on them have a higher possibility of constituting a point of interaction with the users of the session. The number of actions that can be performed on an object may vary and can concern the modification of its location, size, shape, color, texture, etc. We define as object's degree of interaction (DoI) the number of actions that can be performed on each object. The prediction technique proposed encounters this degree of interaction for being able to predict the avatars' distribution within the virtual environment after a certain period of time.

3. Partitioning Algorithm

The partitioning approach presented in this paper considers that the initial partitioning of the virtual environment and its assignment to available servers takes place when the world is empty, which means that the world is comprised only by its objects and there are no users connected to the platform. In this initial state the virtual world is divided in equal-sized cells. The number of cells is related to the average diameter D of avatars' Area of Interest (AoI). Initially, all cells rely in one server. Each cell can comprise a number of objects. The workload for handling each cell depends both on the number of objects of this cell as well as to the DoI of these objects. In particular, the number of objects increases the workload that each cell introduces to the server, while DoI "forecasts" the workload that each object will introduce to the server when users will interact with it. Furthermore, DoI could be considered as indicator for avatars' behavior (in terms of motion and interaction) within the virtual environment. In particular, DoI value of an object could be used as prediction parameters in terms that when a number of objects is within the AoI of an avatar then the possibility of the avatar in selecting the object that it will visit and interact with is strongly related to the DoI of these objects. Furthermore, in the vast majority of virtual environments there are certain points/areas ("entry points"), where the user is transferred when entering the world or when transported from one place to another. We refer to these entry points as "hot spots", in the sense that in these areas there is often a high concentration of avatars. These areas either empty or not, add workload to the servers and should be therefore cautiously handled.

When avatars enter the virtual environment or are transported from one point to another (hot spots) and given their AoI diameter D, will most likely move towards neighboring cells of the hot spots. Thus, based on this observation and in combination to the effect that DoI plays in avatars' behavior (motion and interaction), the algorithm, when distributing cells to the existing servers tries to keep the hot spots and its neighboring cells to the same server so as to minimize the communication cost among the servers that avatars' movement and interaction would introduce, if these cells relied in different machines. At this point it should be mentioned that the rational of the algorithm is to mark the "predicted" crowded places within the virtual environment and assign them to the available servers so that workload will be balanced. For handling cases where the number of hot spots within the virtual environment is less than the number of available servers, we introduce the term of the "virtual" hot spot. The word "virtual" indicates that the selected cell is not an actual entry point to the virtual world, but it introduces some heavy workload to the server. The incorporation of virtual hot spots allows us to distinguish the crowded places of the virtual world and contribute to a more balanced distribution of workload.

Based on the main concepts of the partitioning approach, described above, this subsection presents the partitioning approach in algorithmic terms.

```
1. Divide the virtual world into $N$ disjoint cells of area $D^2$, where $D$ is the average diameter of the Area of Interest ( $AoI$ ) of the avatars
      /* Create a graph for the virtual environment*/
2. For each cell create a node and calculate {
      o the number of active, static, multi-user objects $Nob_i$ that reside in this cell
      o for each object $ob_i$ calculate the workload it introduces to the server based on its $DoI$ as follows: $ob_j\_cw \times doi\_ob_i \div 5$
      o Add the workload of each object to the total workload of each cell: $C_i\_cw = C_i\_cw + ob_j\_cw$ }
3. Find cells which constitute entry points and mark them as hotspots:
      if ( $C_i.kind = entry$ ) { $C_i.hot\_spot = true$ }
4. for ( $C_i.hot\_spot = true$ ) {
      o find its neighbors $C_j$ and create an edge: addEdge( $C_i, C_j$ )
      o mark neighbors: $Cj.neighbor = true$ }
5. Calculate the total number of hot_spots
```

```
6.If ( $hot\_spot\_num < Server\_num$ ) {
/* Create additional "virtual" hot spot areas from cells that are not neighbors of hot_spot cells based on the maximum workload */
    o For ( $Server\_num - hot\_spot\_num$ ){
        ▪For ( $C_i.neighbor != true$ ) {
            Select $\max C_i\_cw$
            Mark $C_i$ as hot spot: $C_i.hot\_spot = true$ }}
    o Find its neighbors and create an edge: addEdge( $C_i, C_j$ )
/*Begin with the assignment of cells to the available servers using round robin */
7.for ( $i = 0;\ i < Server\_num;\ i++;$ ) {
    o assign one hot spot $C\_hs$ to each server }
8.for ( $i = 0;\ i < Server\_num;\ i++;$ ) {
    o for ( $C\_hs_i$ ) {
    o while ( $C\_hs$ has neighbors) {
        ▪assign the neighbor of $C\_hs$ for which $\max C_i\_cw$ } } }
9.for (remaining cells) {
    o select cell with $\max C_i\_cw$
    o find neighbors of $C_i$
    o while ( $C_i$ has neighbors $C_j$ ) {
        ▪find server number of $C_j$
        ▪calculate server workload: $s\_cw_j$
        ▪select $\min(s\_cw_i)$ }
        ▪set $server[Server\_num]$ as $candidateServer$ }
    o else for ( $i = 0;\ i < Server\_num;\ i++;$ ) {
    o calculate server workload: $s\_cw$
    o select $\min(s\_cw)$
    o set $server[Server\_num]$ as $candidateServer$ }
10.for (remaining cells) {
    o select cell with $\max C_i\_cw$
    o assign cell to $candidateServer$ }
```

4. Experiments

In this section we examine two different cases (scenarios) for the same virtual world, where the entry points (hot spots) and the virtual hotspots are differently placed within the environment. For each of these cases we present the initial state of the virtual environment, the final distribution of cells to servers after the application of the proposed algorithm as well as the steps followed for each scenario.

We consider a small-scale virtual environment with dimensions 4x4. The objects of the virtual environment are marked with the cycles, while the value of these cycles represents the DoI value for each object. We consider that the average workload for an object is 5, the average diameter D is 1 and the number of servers available equals to 3. For the first scenario, the process that takes place is the following: We note that the number of hot spots available is 2 (C0 and C15), while the number of existing servers is 3. Thus we need to create an additional "virtual" hot spot (steps 9-10 of the partitioning algorithm). From the remaining cells we select the ones with the highest workload, which is C13 (steps 11-12 of the partitioning algorithm). Since we have created at least one hot spot to be assigned to each server, we start the assignment of the hot spots cells to the servers available (using round robin) (steps 13-14 of the partitioning algorithm).

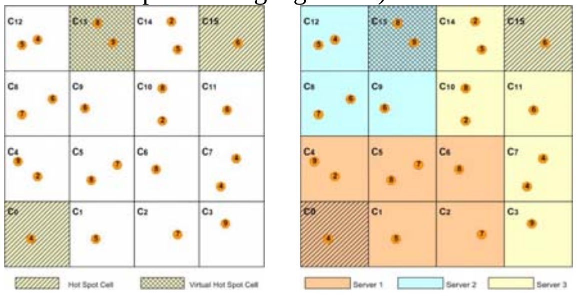

Figure 1: Initial and Final State for Scenario 1

Steps	Server 1	Server 2	Server 3
1	C_0	C_{15}	C_{13}
2	C_5	C_{10}	C_8
3	C_4	C_{14}	C_{12}
4	C_1	C_{11}	C_9
5	C_6		
6		C_7	
7	C_2		
8		C_3	
Workload	**50.0**	**46.0**	**45.0**

Table 1: Workload Assignment for Scenario 1

When this step is completed, for each of the hot spot neighbors, we start moving the cells in regard to the workload. When all neighbors are assigned to the servers, we continue with the assignment of the remaining cells, those that are neither hot spots nor neighbors to hot spots (C2, C3, C6 and C7). For this assignment, which also takes place in a cell-by-cell way, we select the cell with the maximum workload and we assign it to the server with the minimum total workload. This process takes place until all remaining cells are assigned to a server (steps 18-24 of the partitioning algorithm).

The second application of the algorithm (scenario 2) presents the case where the available hot spots and virtual hot spots share common neighbors. In this case, we calculate the total number of neighboring cells for each hot spot and we give priority (when assigning the neighbors) to the hot spot with the smaller number of neighbors (so as to ensure that workload will be balanced and all hot spots are assigned with neighbors).

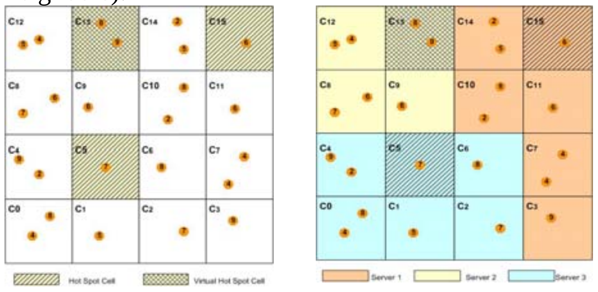

Figure 2: Initial and Final State for Scenario 2

Steps	Server 1	Server 2	Server 3
1	C_{13}	C_4	C_{10}
2	C_8	C_0	C_{15}
3	C_{12}	C_5	C_6
4	C_{14}	C_9	C_7
5		C_1	C_{11}
6			C_3
7		C_2	
Workload	46.0	48.0	47.0

Table 2: Workload Assignment for Scenario 2

As it can be extracted by the results of the algorithm, this approach achieves a relatively balanced workload among existing servers in all cases examined. In the initial state of the virtual world the workload of each server is based on the objects it contains. However, the algorithm by taking into account the degree of interaction of these objects goes one step further and partitions the space based on the workload that each of these objects will introduce when users will connect to the system. Furthermore, even though the algorithm does not take into account any communication among the servers during this initial partitioning of the virtual space, though, the incorporation of the “hot spot” entities as well as the placement of their neighbors to the same servers acts as a “forecasting” action for minimizing the communication cost when users will connect to the system.

5. Conclusion and Future Work

This paper presented an approach for the initial partitioning of a virtual environment. The partitioning takes place when users are not yet connected to the system and is based on objects’ attributes. In particular, the presented algorithm takes into account the “importance” of the objects available and based on this makes a prediction on avatars’ behavior when they will enter the virtual space. The results of the experiments conducted indicate that the algorithm, by incorporating objects’ attributes achieves a relatively balanced distribution of the workload to the available servers, while in addition the partitions created “forecast” the minimization of communication cost among servers when avatars will move from one partition to another. In addition, even though the presented algorithm cannot be applied (in its current version) to persistent virtual environments, however, with minor modifications, could be applied for the further partitioning of overloaded areas (or cells) within a persistent environment.

Some of our planned next steps include the incorporation of avatars after a given period of time and the application of this algorithm for testing the algorithms’ performance when there is also communication among users and for defining the time intervals that rebalancing will be needed. Furthermore, experiments are planned for the application of the algorithm in large-scale virtual environments. Finally, the proposed algorithm will be upgraded for use in persistent virtual environments.

6. References

[1]John C.S. Lui, M.F. Chan: An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems, IEEE Transactions on Parallel and Distributed Systems, Volume 13, No. 1, (Jan 2002).

[2]P. Morillo, J.M. Orduna, J. Duato: On the Characterization of Distributed Virtual Environment Systems, Proceedings of European Conference on Parallel Processing Euro-Par’2003, Klagenfurt, Austria, (August 2003).

[3]Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Donald P. Brutzman, Paul T. Bar-ham: Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments, VRAIS, p. 2, Virtual Reality Annual International Symposium (VRAIS’95), (1995).

[4]P. Morillo, M. Fernandez y J.M. Orduna: An Evolutive Approach to the Partitioning Problem in Distributed Virtual Environment Systems, XIV Jornadas de Paralelismo, p.p. 299-304, Madrid, Spain, (September 2003).

[5]P.T.Tam: Communication Cost Optimization and Analysis in Distributed Virtual Environment, M. Phil second term paper, Technical report RM1026-TR98-0412, Department of Computer Science and Engineerin, The Chinese University of Hong Kong, (1998).

[6]N. Pryce: Group Management and Quality of Service Adaptation in Distributed Virtual Environments, 4th UK VR-

SIG Conference, Brunel University, Uxbridge, UK, (November 1997).A.B. Smith, C.D. Jones, and E.F. Roberts, “Article Title”, Journal, Publisher, Location, Date, pp. 1-10.