Design of the Server Cluster to Support Avatar Migration

Jiung-yao Huang* and Yi-chang Du
Department of CS&IE
Tamkang University, Tamsui 251, Taiwan
*E-mail: jiungyao@ms45.hinet.net**

Chien-Min Wang
Institute of Information Science
Academia Sinica, Nankang 115, Taiwan
E-mail: cmwang@iis.sinica.edu.tw

Abstract

In this paper, we identified an important issue when supporting a large scale networked virtual environment(NVE) with a server cluster. This issue is similar to the process migration issue on the parallel computing study and we refer it as the avatar migration problem. That is, when an avatar of an NVE is moving from one region managed by a server to another region managed by a different server, the client site may perceive abrupt screen change due to the different contents managed by these two servers. This paper proposes equations to solve this problem and elaborates the proposed avatar migration mechanism with state diagrams. The implementing architecture is also given in this paper. Our experiments that successfully show the efficiency of the proposed mechanism are given at the last.

1. Introduction

"Scalability and interest management" is an important research topic on the Networked Virtual Environment(NVE). One of the essential concepts to achieve the scalability of the NVE is to partition the clients among multiple servers and each server will manage a moderate number of participants. Singhal and Zyda[1] pointed out that this partition can be based upon the geographical location of the participants, or the coordination of the avatars within the virtual world. Furthermore, the users within a server can be further classified so that the messages flowing among them are filtered based upon their respective interests, called Area-Of-Interest(AOI). This AOI approach can increase the number of simultaneous participants by reducing network bandwidth requirement. Moreover, the deployment of multiple servers will also significantly affect the scalability of the simultaneous participants.

For the AOI technique, it can be performed either by logical separation or by physical division based upon their corresponding avatars' coordinates. The logical separation is to compute the message interest scope of every avatar on-the-fly and exchange the messages among the users when their interest scopes are intersected. For example, MASSIVE used the Aura[2] to compute the awareness of an avatar. Different quantities of messages will be transmitted among clients depending upon the awareness of each avatar. The physical division is to partition a virtual world into cells beforehand. AOI[3] of an avatar is then dynamically computed based upon the cell that it resides in. AOI represents the ambit in the virtual world that a client is interested in receiving messages. Therefore, how to determine the AOI of a client to filter the messages and, hence, reduce the network bandwidth and computational load is the key to build a scalable NVE.

Locale[4] by the Spline system proposed a different approach to filter the messages and, therefore, control the data flow. Each locale has its own coordinate system and the entire virtual world is constituted of numbers of Locales. The clients within the same Locale will then receive messages from each other. PARADISE[5] proposed a "projection aggregation" approach that consists of organization aggregation and grid aggregation. This approach provides a high-level method to establish the AOI based on the general spatial area and entity class. Interleaved Squaring method[6] is another approach of the spatial culling method for the data flow management. The Interleave squaring method partitions the virtual world into interleaved squares. This approach aims to adopt the advantage of uniformly updating in AOI change provided by the hexagonal approach[3], while without its complexity in computing the resident cell.

Among the previously proposed AOI techniques, the fixed-sized and -shaped spatial culling method is a popular approach among them. It partitions the virtual world into cells of regular sizes and shapes first. The interest management is then computed by estimating the number of the cells that compose the AOI of an avatar. In this way, the computation load of the interest

Proceedings of the IEEE Virtual Reality 2003 (VR'03)
1087-8270/03 $17.00 © 2003 **IEEE**

management is alleviated while the network bandwidth requirement is reduced. However, this approach raises a problem when an avatar migrates from one server to another. We refer it as "the avatar migration issue". When an avatar locomotes to an adjacent region that is managed by another server, the client site may perceive abrupt scene change due to different contents on that server. This paper aims to study this problem and attempts to propose a general solution to it. In the following sections, the discussion of avatar migration issue for the AOI-based server cluster is given first. The proposed avatar migration mechanism then follows. The architecture and implementation of the server cluster is presented afterwards. Finally, the performance of the designed server cluster is discussed.

2. The issue of avatar migrating among server cluster

There are four issues that will affect the scalability and performance of the networked virtual environment[1]. Among them, controlling visibility of data and changing the net-VE network software architecture are two issues related to the server site. In addition, AOI is the technique that is often used to solve the issue of "controlling visibility of data". Hence, previous researches were focused on the server cluster with the AOI technique to reduce the message flow in the network.

However, care must be taken when the AOI technique is applied to the server cluster system. Since the entire virtual world is partitioned into different regions and managed by respective servers, the migration of an avatar from one server to another is an important issue for the multiple server approach of the networked virtual environment. The simplest solution is to discharge the avatar from the original server first and then login it to another server. However, this approach has a serious problem of abrupt visual effect on the client site. That is, the user will suddenly perceive an empty virtual world followed by a gradual emersion of other avatars. For example, as illustrated in Figure 1, avatar *a* is moved from region β, which is managed by server β, to region χ, which is managed by server χ. When avatar *a* is within region β, its controlling client is able to receive the status information of avatar *b*. When avatar *a* is moved to region χ, the visible avatars on the client site *a* should become avatars *f* and *g* while avatar *b* should be dropped out from the display.

Hence, in order to provide a smooth transition of visual effect on the client site, the status information of the avatar must be transmitted from one server to another progressively. Since the client site continuously receives avatars' update messages from the server site, the client site must be able to receive messages from both servers when its avatar is migrating between these two servers. In this way, the avatars will not suddenly disappear when their managing server is disconnected to the client and another set of avatars will not suddenly emerge after a new server is connected. This paper identifies this issue into problems as follows:

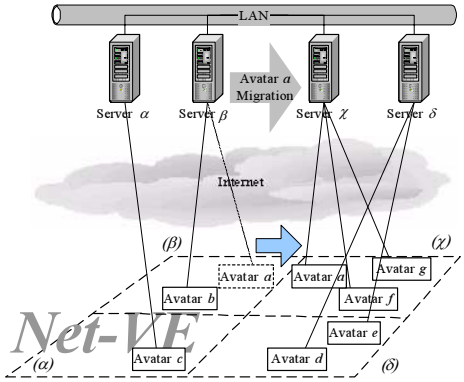

Figure 1. Illustration of the avatar migration problem

1. On the server site, how should we instantaneously migrate an avatar's status from one server to another without delay?
2. On the client site, how should we eschew the user from perceiving abrupt scene change and frame delay while migrating the avatar?

This avatar migration issue on the sever site is similar to the process migration[7] of the distributive computing. However, the avatar migration issue is more complicated than the process migration problem. Since the avatar is interactively controlled by the user, the motion of the avatar within the virtual world is unpredictable and its migration process must also be executed interactively.

3. The mechanism for the avatar migration

3.1. The proxy object and the shadow object

In order to provide a seamless avatar migration from one server into another, we overlap the cells on the border of the region managed by a server with its adjacent server. Based upon the spatial AOI technique, each server of the server cluster for NVE manages a region of the virtual world. In addition, each region is composed of a set of adjacent cells. By overlapping the border cells of two adjacent servers, we can use these border cells as the buffer zone for avatar migration. We call these two servers with overlapped cells as the logically chained server of each other. For example, as illustrated in Figure 2, let sub-region B be the border region that is managed by both server *m* and server *n*. Hence, the region managed by server *m* is A∪B and region B∪C is managed by server *n*. Server *m* is the chained server of server *n*, and vice versa.

As shown in Figure 2, when an avatar enters the

border, its AOI will cover the cells from both chained servers. Both servers will be aware of the existence of that avatar and a copy of its information must then exist in both chained servers. We model the copy of avatar information on the station server as the **proxy object** and the avatar information on the migrating server as the **shadow object**[8]. Hence, an avatar controlled by the client is represented as a proxy object on the server site and, when an avatar reaches the border district of a server, its shadow object is then created on the migrating server.

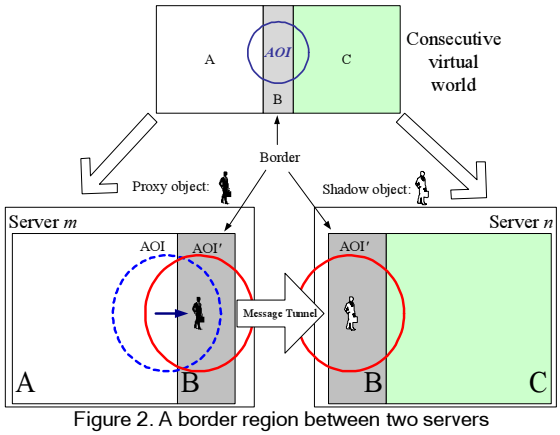

Figure 2. A border region between two servers

The proxy object and the shadow object provide a seamless mechanism to enable the avatar to perform migration among the chained servers. For example, as shown in Figure 2, when an avatar enters the border area, its AOI is updated to AOI'. While an avatar is in the border area, this situation signifies the possibility that it will migrate into a chained server. To provide a smooth migration, this chained server needs to have this avatar's status information when it is in the border. Hence, when the server is aware of the event of an avatar entering the border and it will notify its chained server, a shadow object of that avatar is then created by the chained server. Since the shadow object is a replicate of a proxy object, while an avatar remains in the border, the station server will automatically copy the proxy object's information to its shadow object on the chained server. The chained server will then forward the received information to the clients within the spatial AOI of this shadow object. In this way, the clients on the chained server can be aware of the existence of this migrating avatar, and, similarly, it will receive other clients' message with the help of its shadow object. When this avatar eventually migrates into the chained server, it is then taken over by the chained server immediately since its information already exists in that server.

The shadow object is similar to a clone of that proxy object on the chained server. With the help of the shadow object and the proxy object, we can model the interest management that supports the avatar migration as follows. Let O_i represent the proxy object of an avatar with index i on server m, which is controlled by client c_i, and $\acute{O}_i$ is its shadow object on the chained server n. Then, the interest management for the proxy object O_i can be modeled by Eq.(1).

$$F(AOI(O_i))=\{ c_k \mid \forall O_k \propto AOI(O_i) \text{ and } \forall \acute{O}_k \propto AOI(O_i)$$
where O_k is the proxy object and $\acute{O}_k$ is the shadow object in server m } ..... (1)

Eq.(1) computes a set of clients whose avatars are in server m and a set of clients whose shadow objects are in server m but are managed by server n. For example, as shown in Figure 2, Eq.(1) computes the set of clients whose proxy object or shadow object is located in the region of $AOI \cup A$ and $AOI \cup B$. In addition, the interest management for the shadow object $\acute{O}_i$ can be modeled by the following Eq.(2).

$$F(AOI(\acute{O}_i))=\{ c_k \mid \forall O_k \propto AOI(\acute{O}_i)$$ where O_k is the proxy object in server n that is within AOI of shadow object $\acute{O}_i$ but is not located in the border} ..... (2)

Eq.(2) computes the set of clients whose avatar is in server n but are not within the border. For example, Eq.(2) will derive a set of clients who have their proxy objects located in region $AOI \cup C$ in Figure 2. Hence, Eq.(1) and Eq.(2) together provide a mechanism to perform interest management with spatial AOI technique on a server cluster. The detail mathematical proof of the correctness and completeness of Eqs.(1) and (2) is given in [9].

3.2. The state diagram

The proposed avatar migration mechanism can be best described by the state diagram. For the legacy server cluster for NVE, the virtual world is partitioned into several regions that are managed by different servers and there is no border area between two adjacent regions. Hence, the migration of an avatar from one server into another can be illustrated by Figure 3.

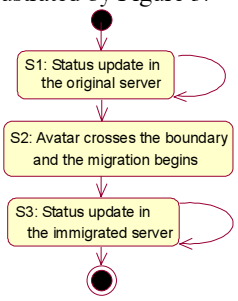

Figure 3. The avatar migration without the border area

The server performs avatar migration immediately after this avatar across the regional boundary between two servers. However, since this situation is similar to the process migration for the distributed computing[7], the avatar information is transformed to another region that is

managed by a different server. The client site needs to remove its contents from the original server and wait for the information from the immigrated server to display. This situation will perceptually cause the client site's display to produce an abrupt view change. That is, the display will first exhibit a blank screen, followed with gradually emerging of new objects and avatars.

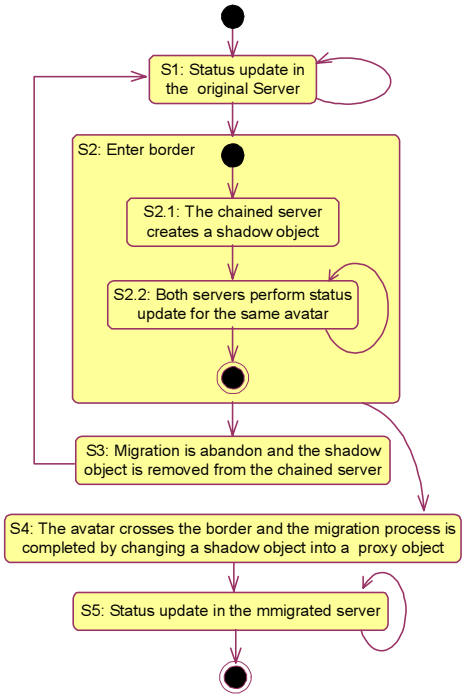

Figure 4. Avatar migration with the help of the border

To solve this problem, the concept of border is proposed and an extra state is introduced to represent the work when the avatar is in the border. As depicted in Figure 4, state S2 represents the situation when the avatar enters the border. State S2 is further modeled by two sub-states, S2.1 and S2.2, to represent the process after the avatar enters the border. The avatar migration process is completed when the avatar crosses the border as shown in state S4 in Figure 4. This state simply changes the shadow object in the immigrated server into a proxy object and removes the proxy object from the original server. However, another state is required to model the situation that the avatar abandons the attempt of immigration as illustrated in state S3. In this case, the server simply removes the shadow object in the chained server and returns to status update state, i.e. S1.

4. The architecture

4.1. The architecture

The proposed server cluster is implemented in the master/slave hierarchy as illustrated in Figure 5, which contains the following three types of servers:

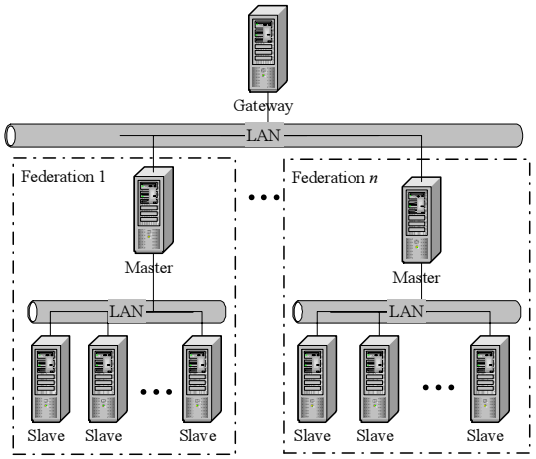

Figure 5. The server cluster hierarchy

1. Gateway server: It is the portal server of the server cluster. It directly communicates with the Master server to log the status of each virtual world. There is only one such a server for NVE.
2. Master server: It is a coordinator of a virtual world. It contains the partitioning information of the virtual world and records which region is managed by which Slave server. It is also responsible for coordinating the communication among the chained servers during the avatar migration process.
3. Slave server: This server actually manages the avatars within a virtual world. However, it serves the avatars only within its governable region. Its main function is to transmit the status update information of an avatar to other participants according to the spatial AOI technique. Hence, it is an AOI manager and is fully responsible for the avatar migration issue.

Notice that each virtual world only can have one Master server but have several Slave servers depending upon the partition of the virtual world. In addition, this server cluster is designed based upon the specifications of High Level Architecture(HLA), which is IEEE 1516 standard[10], and complied with the implementation of Run Time Infrastructure(RTI) by Defense of Modeling and Simulation Office (DMSO)[11], USA. That is, the Gateway server is similar to RtiExec of DMSO RTI and the Master server is functionally similar to FedExec of DMSO. However, the designed server cluster extends FedExec into Master/Slave hierarchy to implement the Data Distribution Management(DDM) service with the avatar migration capability[12]. A Local RTI Component(LRC) that follows the HLA interface specification is implemented on the client site.

One of the important issues to design a server cluster is the communication mechanism among the interconnected servers. Two de facto standards of the communication layer for the server cluster are Message Passing Interface(MPI)[13] and Fast Message(FM)[14].

The MPI is a specification for message passing library and it is designed for the applications in distributed memory architecture, message passing environment, and parallel computation. Its deign goal is to provide a practical, portable, and effective message passing interface and is widely adopted on massive parallel computing system, Symmetric Multiple Processors(SMP) cluster, workstation cluster, and heterogeneous networking environment.

FM takes an alternative approach to provide a highly efficient and portable communication layer for the server cluster. Since FM supports a set of basic point-to-point low-level communication functions, FM can achieve low latency communication among interconnected hosts. Due to this intrinsic nature, most of the MPI libraries were designed on top of FM as its bottom layer.

Based upon the intrinsic feature and the communication efficiency of FM, this research employs FM to design the server cluster. Further, by adopting FM as the communication layer, the designed server cluster can be portable among different computing platforms that support FM. Among publicly available FMs from the Internet, FM of Federated Simulations Development Kit(FDK)[15] from Modeling & Simulation Research & Education Center, Georgia Institute of Technology, is chosen due to its availability of source code and variety of supported platforms. Hence the infrastructure of the implemented server cluster is as shown in Figure 6.

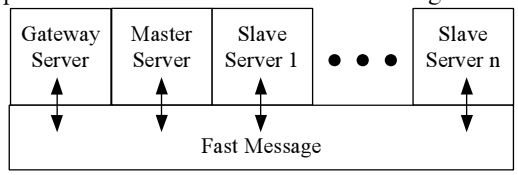

Figure 6. The infrastructure of the server cluster

4.2. The interleaved-squaring AOI technique

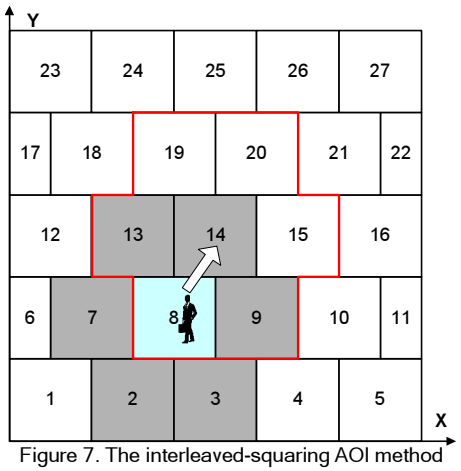

Figure 7. The interleaved-squaring AOI method

The spatial AOI implemented in this server cluster is called the interleaved-squaring spatial AOI method.[6] This AOI method is the improvement of the hexagon topology[3] by merging its stability with the simplicity of the squaring method[16]. That is, as illustrated in Figure 7, the interleaved-squaring method is the AOI technique that integrates the benefit of constant cell number of AOI from the hexagonal topology and the gain of simple cell partition from the squaring topology. For the interleaved-squaring spatial AOI technique, the virtual world is first partitioned into a set of fixed size of squares, or cells. The cells are then interleaved by half, row-by-row as illustrated in Figure 7.

Each cell S_i of the interleaved-squaring AOI method satisfies the AOI model defined in [9], i.e. $S_i \cap S_j = \phi$, $\forall\ i \neq j$. Hence, the AOI for an avatar on cell S_8 is $AOI_8 = \{S_2, S_3, S_7, S_8, S_9, S_{13}, S_{14}\}$. When the avatar moves to cell S_{14}, its new AOI is then updated to $AOI_{14} = \{S_8, S_9, S_{13}, S_{14}, S_{15}, S_{19}, S_{20}\}$. Three cells, i.e. cells S_2, S_3, S_7, are removed from the new AOI set and three cells, i.e. cells S_{15}, S_{19}, S_{20}, are added. [6] shows that the interleaved-square culling method guarantees that the AOI always removes 3 cells and adds 3 cells to construct a new AOI no matter which direction the avatar is locomoted.

4.3. The algorithm

The server cluster is implemented on Red Hat Linux 7.2 platform. The operation of the server cluster can be outlined into two phases, which are the Initialization phase and the Run Time phase. The Initialization phase is mainly executed in the master server that partitions the virtual world into a set of interleaved squares as illustrated in Figure 7. These interleaved squares are then clustered into several regions based upon their spatial relationship and each region is assigned to a slave server. The neighboring relationships among the slave servers along with the border information between two chained slave servers are also recorded in this phase.

At the run time phase, the slave server creates a proxy object for each avatar residing in its managed region. On the client site, a local object is created for its controlled avatar whereas each remotely controlled avatar is modeled as a clone object as shown in Figure 8. When an avatar in the slave server *m* enters the border, server *m* will alert this event to the master server through the dotted line 2. The master server will then search its record for the chained slave server, says server *n*, of the slave server *m*. The connection information of the slave server *m* is then passed to the chained server *n* through the dotted line 3. Upon receiving this message, the chained slave server *n* sets up a data channel with the slave server *m*, as illustrated by dotted line 4. The slave server *m* then uses this data channel to pass the information of proxy object *a* to server *n* as illustrated by dotted line 5. Accordingly, server *n* creates a shadow object *á* for proxy object *a*. This dotted line 5 is also used

by server *m* to perform status update for proxy object *a* later on.

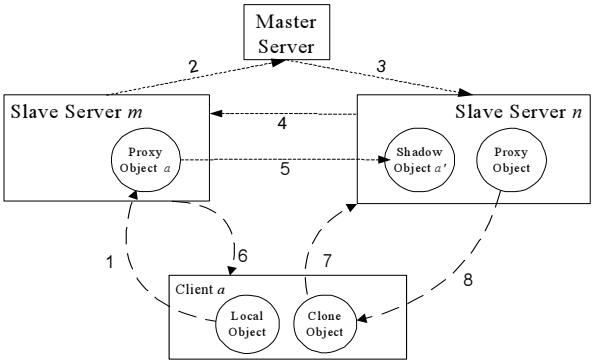

Figure 8. Avatar migration process of enter the border

Furthermore, the chained slave server *n* will also provide its connection information to the slave server *m* through dotted line 4. This connection information is then passed to client *a* so that client *a* can set up a link with server *n* as illustrated by dotted lines 6 and 7. Hence, the chained slave server *n* will then forward the information of the avatars within AOI of shadow object *á* to client site *a*, through dotted line 8, so that client *a* can propagate respective clone objects.

5. The experiment and performances

The definition of shadow object solves the problem of abrupt frame change on the client site, which is one of the two essential issues for the avatar migration problem on the server cluster of an NVE. In addition, Eq. (1) and Eq. (2) ensure the consistency of scene change in the process of avatar migration between two slave servers. However, the fluency of the frame display heavily depends upon the efficiency of these two equations. That is, the efficiency of Eq(1) and Eq.(2) decides if both slave servers can promptly forward avatar's information in the border to the clients managed by different slave servers and, similarly, forward other avatars' information to the client whose avatar is in the border. Subsequently, the client site will not perceive the discontinuity of the frame update caused by the avatar migration between two slave servers. Hence, experiments are designed and conducted to verify the efficiency of the proposed mechanism. However, the performance issue involves the communication latency among the server cluster and the network latency between the client and the server cluster. Since the later one is heavily affected by the availability of Wide Area Network(WAN) and it is an unpredictable factor, our following experiments focus on studying the communication latency among the server cluster.

There are two operations that affect the performance of the avatar migration. One is the re-computation of AOI when the avatar is moved to a new position and the other is the status update when the avatar stays in the border. Specifically, the later operation is decided by Eq.(1) and Eq.(2) and the efficiency of these two equations is affected by the number of avatars per cell. Hence, our study of the avatar migration efficiency is to trace the locomotion of a specific avatar when it is navigating from one server into another, as shown in Figure 9. By varying the number of avatars within a cell and evaluating the computation time that the server cluster spends on each step, this experiment aims to verify the efficiency of the presented avatar migration mechanism.

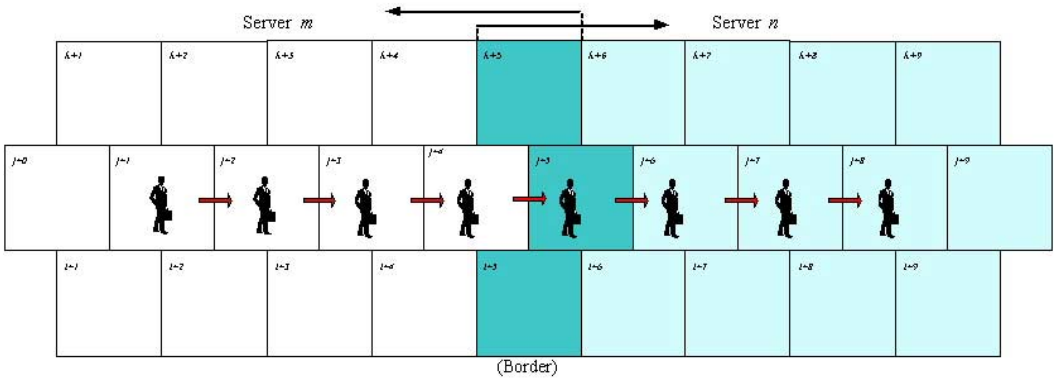

Figure 9. The scenario of the conducted experiment

In order to reduce the complexity of the simulated scenario, the experiment is based upon the following assumptions.

1. The interaction in the simulated scenario is simplified. We only consider the navigation efficiency when an avatar performs migration. The interaction between the avatar and the environment is not an issue here.
2. Only the avatar that is traced to collect data is allowed to travel from one cell into another. All of the other avatars are confined to perform status update in the cell where they are located. The reason of this restriction is to maintain the population of avatars in each cell as well as in the border. In addition, with the help of this assumption, the accuracy of the experimental data will not be diminished due to the variation of the avatar number.

3. Both the server cluster and the client are located in a same local area network. Since the communication latency between the client and the server is not considered in our experiment, this setting will reduce the inaccuracy of the experimental data caused by the network.
4. The size of data packet for the status update between the client and the server is 80 bytes. This data packet contains a data header and the positional information of an avatar.
5. The size of data packet transmitted between the master server and a slave server is 20 bytes. This sizing of the data packet includes the cell index and all other information for a slave server to notify the master server and, consequently, the master server to locate a chained slave server.
6. The size of the data packet communicated between two chained slave servers is 36 bytes.

The hardware setup of the experiment on the server site includes one Intel Pentium III 933 with 256MB memory as the master server and two Intel Pentium III 866 with 256 MB memory to be the slave servers. All of these three servers run on Red Hat Linux 7.2. The result of the experiment is shown in Figure 10.

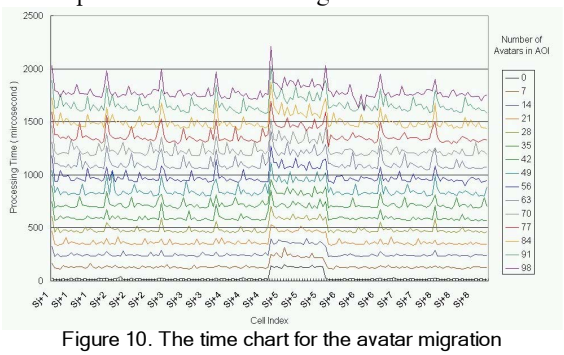

Figure 10. The time chart for the avatar migration

Figure 10 displays the time chart when an avatar locomotes from cell S_{j+1} in server *m* to cell S_{j+8} in server *n*. Compared to the normal situation, the server cluster spends extra process time when an avatar is in a borer cell S_{j+5}. That is, the processing time on cell S_{j+5} is more than on any other cells. This overhead is induced by the coordination between the master server and the slave server and the message communication between two slave servers. Referring to Figure 8, this overhead is the processing time that the slave server *m* copies the information of Proxy object *a* to its Shadow object *á* through dotted line 5. Furthermore, the peak time between cell S_{j+4} and S_{j+5} is the time for the server cluster on Figure 8 to execute the protocol illustrated by dotted lines 2, 3, 4 and 5. Notice that, this processing time also includes the time required for the slave sever *n* to create a Shadow object.

To further explore the efficiency of the protocol for the avatar migration mechanism, another experiment is conducted. This further experiment omits the status update functions, i.e. Eq.(1) and Eq.(2), when we perform the avatar migration. That is, the AOI update and the avatar migration mechanism are the two major issues considered during the experiment. The result of this experiment is shown in Figure 11.

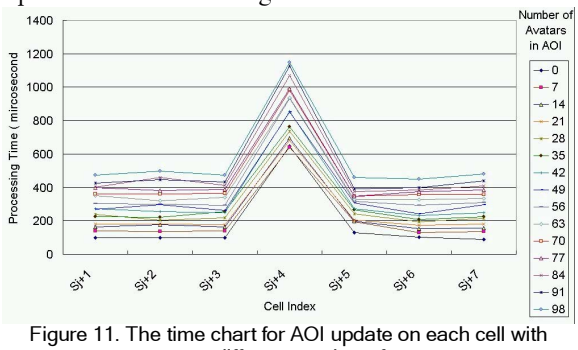

Figure 11. The time chart for AOI update on each cell with respect to different number of avatars

Figure 11 exhibits a high peak of processing time when an avatar enters the border cell S_{j+5} from cell S_{j+4}. As illustrated in Figure 8, except for re-computing its AOI when an avatar enters a border cell, a data channel is established by the protocol of dotted lines 2, 3, 4 and 5. Furthermore, two extra procedures are performed in the chained slave server, which is to create a Shadow object and to wait for connection from the client as illustrated by dotted lines 6 and 7 in Figure 8. However, the times for dotted lines 6 and 7 are not counted in our experiment since they rely heavily on the availability of the Internet that contain many uncontrollable factors. All of these operations cause overhead to the server cluster. This overhead ranges from 1.6 times, when AOI contains 98 avatars, to 6 times, when AOI contains 0 avatar, of the non-border cell's AOI update. This overhead becomes insignificant as the number of avatars within an AOI increases.

From the above experiments, we know that the avatar migration produces overhead when an avatar enters the border. By discarding the time for AOI update, this overhead is 550μs when AOI contains 0 avatar and it is increased to 650μs when AOI contains 98 avatars. This overhead includes the transmission time of FM. The study of the performance of FM[12] on different sizes of data packet. We learn that FM requires 60μs to transmit a 36 bytes of data. That is, the communication overhead to complete the steps of dotted lines 2, 3 and 4 in Figure 8 is 180μs. Since it requires 70μs to transmit 80 bytes of data through the step of dotted line 5 in Figure 8, we can conclude that, when an avatar enters the border, the protocol overhead is 260μs. This overhead is negligible when the networking environment between the client and the server cluster is Internet.

6. Conclusion and future works

This paper defines the issue of the avatar migration problem on the server cluster-based Networked Virtual Environment which employs the spatial Area-Of-Interest(AOI) technique. The interest management method along with the spatial AOI technique is a conventional approach for the networked virtual environment to increase the number of simultaneous participants without exceeding the supporting network bandwidth. This approach requires the entire virtual world to be regularly partitioned into cells of fixed size and shape beforehand. Each server of the server cluster manages only a set of adjacent cells that constitute a region of the virtual world. The interest management is then computed on each server by estimating the number of cells that compose the AOI of an avatar to perform message filtering. In this way, the computation load on each server is alleviated and the network bandwidth requirement is reduced.

However, when the avatar migrates from one server into another, the client site will perceive abrupt scene change due to different content on each server. This issue can be solved in two ways. First, how does the server cluster promptly transfer the avatar information from one server to another without delaying the message transmitting to the client site? Secondly, how can we avoid an abrupt scene change on the client site display when the avatar is performing the migration? This paper proposes a mechanism to solve this problem and the experiments successfully certify the efficiency of the presented avatar migration mechanism. Our experiments show that, although the avatar migration mechanism will induce communication overhead when the server cluster manages the message forwarding operation, the overhead is insignificant with the increase of avatars on the border. Our experimental result also suggests that this overhead can be further reduced by improving the performance of FM for the server cluster.

A further study of the avatar migration mechanism is to integrate this technique with the load balance and fault tolerance capabilities for the server cluster. The load balance capability on the server cluster is to research a method to balance the workload on each server. The workload for the server cluster of NVE is to process the status message of avatars. Hence, the avatar migration mechanism can be adopted to migrate avatar(s) from an overloaded server to increase the overall performance of the server cluster. However, this approach will create another research topic on the spatial AOI technique, which is the dynamical partition of the virtual world. The dynamically partitioning method allows each server of the server cluster to manage a different size of region in the same virtual world. The size of the managed region depends upon the number of avatars within that region. This capability can greatly enhance the scalability of NVE. Similarly, the fault tolerance capability of the server cluster involves the issue of migrating avatars to another server when a fault is detected. The avatar migration is the tool to migrate the avatar when a fatal fault of its server is detected.

7. References

[1] S. Singhal and M. Zyda, *Networked Virtual Environment – Design and Implementation*, Addison-Wesley Pub Co., MA, USA, (1999), pp.181-248.

[2] C. Greenhalgh and S. Benford, "Massive, A Collaborative Virtual Environment for Teleconference", ACM Trans. on Computer Human Interfaces(TOCHI), Vol. 2, No. 3, September 1995, pp.239-261.

[3] M.R. Macedonia, M.J. Zyda, D.R. Prat, D.P. Brutzman, and P.T. Barham, "Exploring Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments", Proc. 1995 IEEE Virtual Reality Annual International Synposium(VRAIS95), North Carolina, March 1995, pp.2-10.

[4] J. W. Barrus, R. C. Waters, and D. B. Anderson, "Locales and Beacons: Supporting Large Multiuser Virtual Environments", IEEE Computer Grapics and Applications, November 1996, Vol. 16, No. 6, pp.50-57.

[5] S. K. Singhal and D. R. Cheriton, "Using Projection Aggregations to Support Scalability in Distributed Simulation", IEEE Proc. of International Conference on Distributed Computing Systems, May 1996, pp.196-207.

[6] Y.J. Cheng, "Study and Design of the Spatial Culling Technique for the Multiple Participants Virtual Environment", Master thesis, Institue of CS&IE, Tamkang University, June 1998.

[7] Dejan S. Milojičić, et al., "Process migration", ACM Computing Surveys, Vol. 32, Issue 3, September 2000, pp. 241-299.

[8] J.Y. Huang, Y.C. Du, and C.M. Wang, "Design of the Server Cluster for the Scalable Networked Virtual Environment", 2001 National Computer Symposium, Taipei Taiwan, December 2001.

[9] J.Y. Huang, Y.C. Du and L. Hui, "The Avatar Migration Mechanism for the Large Scale of Networked Virtual Environment", Submitted to ACM TOMACS, July 2002.

[10] "IEEE Standard for Modeling and Simulation (M&S), High Level Architecture (HLA) - Federate Interface Specification", IEEE Std 1516.1-2000, 2001.

[11] "Chapter 2. RTI Synopsis", RTI 1.3-Next Generation Programmer's Guide Version 4, June 2001. Available at https://sdc.dmso.mil/

[12] Y.C. Du, "Study of the Avatar Migration Problem for Networked Virtual Environment", Master Thesis, Department of CS&IE, Tamkang University, June 2002.

[13] The Message Passing Interface (MPI) standard, http://www-unix.mcs.anl.gov/mpi/

[14] Fast Messages (FM), http://www-csag.ucsd.edu/projects/comm/fm.html

[15] FDK - Federated Simulations Development Kit, http://www.cc.gatech.edu/computing/pads/fdk.html

[16] D. J. VanHook and J. O. Calvin, "Data Distribution Management in RTI 1.3", Simulation Interoperability Workshop, Spring 1998, paper no. 206. Also available at http://dss.ll.mit.edu/dss.web/98S-SIW-206.html