A Scalable HLA-Based Distributed Simulation Framework for VR Application

Zonghui Wang, Jiaoying Shi, and Xiaohong Jiang

State Key Lab of CAD&CG
Zhejiang University, HangZhou, P.R. China, 310058
{zhwang, jyshi, jiangxh}@cad.zju.edu.cn

Abstract. HLA-based Distributed Simulation technology is employed widely in Virtual Reality (VR) applications, such as military simulation, internet games, roaming etc. To support larger number of participants, scalability is becoming a key issue of VR applications. In this paper, we explore the characteristics of distributed simulation, and analyze the scalability of servers and participants, and classify our approach to improve scalability of VR applications into three aspects: a three-tier node management mode to accommodate more participants, an efficient management of servers to manage scalable number of clients and software development interface to achieve reusability and interoperability of VR applications. We present our middleware platform, HIVE, providing a scalable HLA-based distributed simulation framework for VR applications, on which users can develop VR applications easily and quickly. Then we give the method and view of application integration with HIVE. Finally an experimental demo is given.

1 Introduction

With the development of network technology and the increasing requirements of applications in many fields, traditional Virtual Reality (VR) applications which run on only a machine are required to run on network to meet the requirement of large scale participants and scenes. Distributed Simulation technology is employed widely in large scale VR applications.

VR Juggler [1] only provides non-distributed VR application development environment. Existing distributed VR application frameworks include MR Toolkit [2], NPSNET [3], MASSIVE [4], Bamboo [5], DIVE [6], AVOCADO [7], ATLAS [8], etc. Large scale VR applications such as military simulation, internet game, roaming, surgery train etc, have the trends as the following:

(1) They should have a scalable architecture to accommodate more participants.

(2) They should support the collaboration work of large scale distributed nodes to run efficiently to provide services to more participants.

(3) They should be developed quickly and easily with unified specification and development interface.

To fulfill these demands, the scalability of system architecture of VR applications must be improved. There are two parts in network application systems:

Z. Pan et al. (Eds.): ICAT 2006, LNCS 4282, pp. 361–371, 2006.

servers and participants. The key issue to improve scalability, which is related to the design of system architecture, is to improve the capacity of the system and the efficiency of servers group. Most previous systems adopt one or more of the following architectures: peer/peer, client/server, or peer/server. In those systems with servers, resource management and workload balance of servers is very important, which affects the scalability of servers directly. In another side, VR applications should be open, and developers could use the unified interface to develop their own client. In conclusion, there are three factors leading to the architecture scalability of VR applications: node management model, management of servers and software development interface.

Our purpose is to build a general middleware platform, including the APIs and back-ends. On the platform, users can develop VR applications easily and quickly, concentrating more on the logic of application, and spending less time on the infrastructure of application, such as data distribution and management, network architecture [9], etc. In this paper, we propose HIVE, which gives a middleware platform for VR applications development. HIVE has a three-tier node management model: global tier, group tier and client tier. HIVE employs a kind of management of servers to improve the work efficiency of servers. To achieve the reusability and interoperability of VR applications [10], we employ the user interface specification of HLA (Draft 1.3) [11] as reference, which was defined by US DoD, and became IEEE P1516 Standard [12] in 2000.

The remainder of the paper is organized as follows. In section 2, we discuss the key issues on scalability of VR applications. Section 3 depicts the implementation of HIVE in detail. In section 4, we describe VR applications' structure and its integration with HIVE platform. An experimental demo is presented in section 5. Finally we summarize our work in section 6.

2 Key Issues on Scalability of Architecture

Scalability is an attribute of multiprocessor system. What is Scalability? After examining the formal definition of scalability, Mark D.Hill fails to find a useful rigorous definition of it, and thinks that the notion of scalability is intuitive, and use of the term adds more to marketing potential than technical insight [13]. In opinion of Dean Macri, scalability refers to the challenge of making a game that runs acceptably across system configurations that may vary in features, performance, or both [14]. But he only provides his scalability approach in the processor and the graphics subsystem.

In this paper, scalability means the ability to adapt to the system resource changes. In VR applications, high scalability means that it can support large number of users to join in simultaneously and interact with others and objects naturally in virtual world.

Because what we build is a general middleware platform for VR applications development, we don't touch the aspects which are related with applications in practice. We explore the ways to improve scalability of VR application only from system architecture. There are three corresponding factors:

Large number of nodes support. Since distributed simulation contains large number of participants, and also contains many servers to provide the services, how to organize them is one of the main problems of improving scalability.

High efficient servers run. While the number of participants becomes larger, the workload of servers also becomes heavier. If the servers run more efficiently, the simulation could contain more participants. So management of servers is one of key aspect of improving scalability. To make full use of servers, it is very important to deal with participants division and migration to balance the workloads.

Openness of platform. It is important that the simulation platform should have good openness to improve its scalability. It means that the simulation platform should comply with unified specification and be open to the third party, and support developers to develop their own application. More over, the simulation platform should provide the software development kit, on which user could develop the VR application quickly and easily.

According to these three factors, our approach focuses on:

Node management model. Node management model defines the organization structure of nodes, and classify the nodes and give different functions to them. There are three kinds of node management model of network applications. The first is centralization mode, i.e. client/server model. In this mode, the efficiency of mode management is very high because of the coordinating of server. But when the number of the client is large, the server's burden of workload and network interface would be very heavy. The second is distribution mode, i.e. peer to peer model. In this mode, there is no bottleneck of server, and the architecture of system is scalable. But when the number of the client is large, the efficiency of node management is very low without the coordinating of servers. The third is hierarchy mode, i.e. multiple levels of client/server model. In this mode, the efficiency of node management is high, and it supports large number of node. But the implementations of hierarchy applications are very complex.

To get high efficiency of node management and accommodate large number of node, we adopt a three-tier node management model.

Management of servers. The function of servers is to provide services to clients. Management of servers contains server resource management, workload allocation and dispatch. The servers run more efficiently, the system will support more services to more clients, and will be more scalable.

To improve the efficiency of servers run to support large number of participants, we provide two kinds of client distribution.

Software development interface. Software development interface contains the specification of services which the simulation platform provides and the library of them.

To make the VR applications reusable and interoperable, we employ the interface specification of HLA as reference. And to provide more convenience to developers, we provide the software development kit (SDK).

3 HIVE

HIVE is a Middleware Platform, containing the following:

HIVE back-ends. HIVE back-ends consist of three parts, Directory Manager (DM), Group Agent (GA) and Group Manager (GM). Group Manager provides the global services, such as GM/Client resource identity service, monitor and control, workload balance, etc. Group Agent provides GM management to create/manage/destroy GM, following the instructions of DM. GM provides HIVE services to Client.

HIVE library. HIVE library provide the interfaces of HIVE services. Client implements the VR application using the HIVE library.

Figure 1 illustrates the network architecture of HIVE.

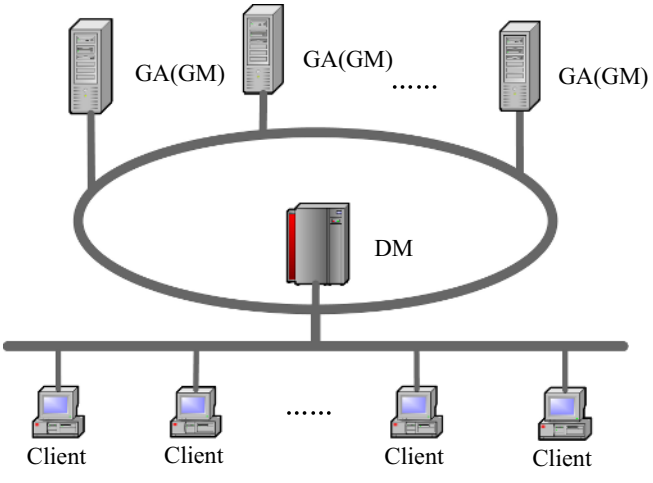


Fig. 1. Network architecture of HIVE. "GM" means Group Manager. "GA" means Group Agent. "DM" means Directory Manager.

3.1 Node Management Model

We present a three-tier mode management model: The first tier, named global tier, contains Directory Manager and Group Agent. Group tier is the second tier, consisting of Group Managers. Client lies in the client tier, which is the third tier.

A critical concept in HIVE architecture is "routing tree". All the nodes in VR applications are organized as a hierarchical tree - routing tree. Routing tree has three levels. The root of it is DM. The children of DM, in the second level, are GMs. The third level contains clients. DM maintains and updates the routing tree dynamically, and each GM has a copy. According to the routing tree, DM and GMs can find any client's owner GM to deliver messages, which is the parent node of the client.

Directory Manager (DM). Directory manager is the process that will be launched first. The architecture of DM is illustrated in Figure 2. It provides the global services: client and GM join / leave, client and GM resource monitor, client migration, workload computation and balance, client distribution, maintaining the "routing tree" and other global service, such as "global time query", "name resolve", etc.

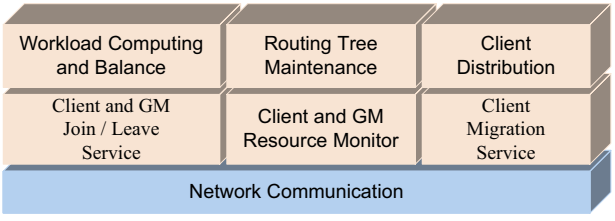


Fig. 2. Architecture of Directory Manager

Group Agent (GA). Group Agent is a daemon process. GA keeps communication with DM, sends the "heartbeat" to the DM to indicate that the GA is working. GA also launches the GM according to the "Create GM" instruction from DM, and destroys the GM according to the "Destroy GM" instruction.

Group Manager (GM). A Group Manager specifies an application. Figure 3 illustrates the architecture of GM. GM maintains the list of the clients which are assigned by the DM. It also communicates with and provides services to the clients, such as time management service, user management service, ownership management service and object distribution management service according to the clients, which will be described later. To provide services, GM keeps all the basic information of the clients, such as "Subscribed List", "Published List", "Update List" and "Discovered List", and updates them. In client message delivery, GM retransmits the messages from source client to destination client. If the destination client is maintained by another GM, the messages are delivered to that GM first, and then retransmitted to the destination by that GM. Message queue is the buffer of messages, which enables processing messages asynchronously.

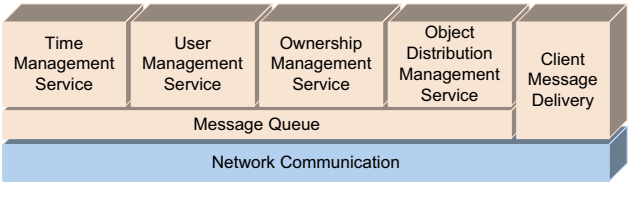


Fig. 3. Architecture of Group Manager

Client. Client can act as various roles in VR, such as the participant, the scene object, or just a passive viewer [15], etc. Figure 4 shows the architecture

of client. HIVE Client consists five parts to provide services to VR applications. User management contains the services such as creating / destroying application, joining / leaving application, etc. Object Distribution Management provides the services such as publishing / subscribing object class, registering / discovering object, updating / reflecting object, publishing / subscribing interaction, sending / receiving interaction, etc. Ownership management provides the services such as requesting / divestiture ownership, etc. time management contains the services such as querying Time, advancing Time, etc. Callbacks are the interface of event notification services. When an event comes, the corresponding callback will be called.

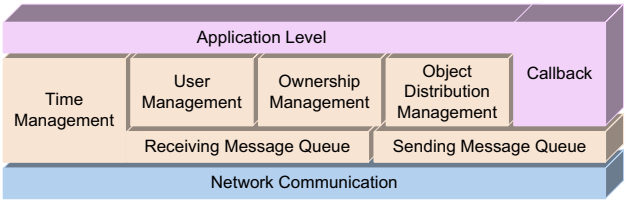


Fig. 4. Architecture of Client

On client side, there are two message queues. One is the sending queue, which is the queue for messages asking for a service. The other is the receiving queue, which buffers the messages and delivers them in proper time with the help of time management.

3.2 Management of Servers

In HIVE, resource management is mainly done by DM. GMs join/leave the DM, and then DM accepts them via GM Join/Leave Service. After GMs join, GM Resource Monitor will monitor and control them. When a client applies for a GM request to get HIVE Service, GM Resource Allocate Service will compute the workload of all GMs, and then select an appropriate GM for the client according to the GM Allocating Strategies. Figure 5 shows the workflow of resource management of HIVE.

We provide two kinds of methods of client distribution: one is that participants are grouped and assigned to a GM according to the result of workload balance of DM, no matter where it is located during initialization or where it goes later. And participant will not be migrated to other GM except when the GM is going to exit or it needs to reduce workload if a new GM joins. This method is helpful when participants of the application move within a wide area and move back and forth quickly, and the states of them don't update frequently. The other is that each GM is in charge of a partition of the virtual world. In this case, when participant joins the DM, the DM will find an appropriate GM according to its original location. When a participant moves to another partition, he will be migrated to the GM that is in charge of that partition. This method is

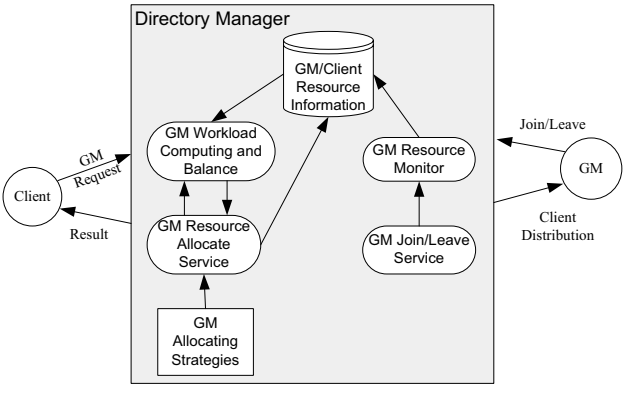


Fig. 5. Resource management of HIVE

useful when participants of the application will not move very frequently, but its frequency of state update can be high.

3.3 Software Development Interface

For convenience of VR applications development, we provide HIVE SDK. HIVE SDK includes APIs specifications and library files to be linked and run. To achieve the reusability and interoperability of VR applications, we employ the interface specification of HLA as reference. The APIs are classified into four parts. We adopt federate management of HLA as our user management, and adopt Declaration Management, Object Management and Data Distribution Management of HLA as our Object Distribution Management, and adopt Ownership Management and Time Management of HLA as our Ownership Management and Time Management.

4 Application Integration

There are two parts in the structure of a VR application. The first part is application logic, which is the content of a VR application. The second part is data presentation, which is the way would be used to show the result of a VR application. The data presentation can be text, 2D graphics or 3D graphics, etc.

Figure 6 illustrates the process flow of HIVE client. First, it selects an avatar from the model database, and loads the scene of the VR application, and then joins in the VR world. Then it goes into the VR application loop. The user interface of the client reflects the changes of dynamic information. Finally it resigns from the VR world.

Figure 7 illustrates the relationship of HIVE back-ends and HIVE applications.

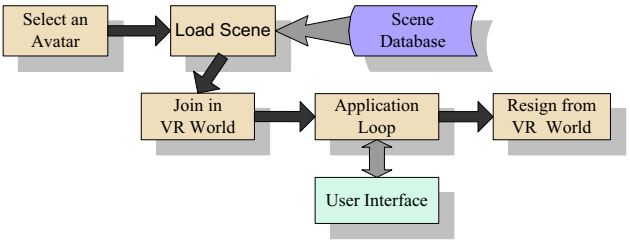


Fig. 6. Process flow of HIVE client

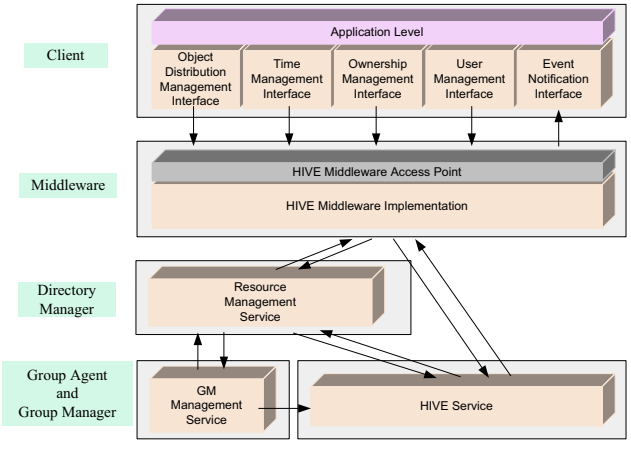


Fig. 7. Relationship of HIVE back-ends and HIVE applications

5 An Experimental Demo

We implement an experimental Demo, Navigator, to demonstrate how a VR application can be developed on HIVE. In Navigator, We use roaming as the application logic, and implement different 3D graphics as data presentations in different clients. Navigator loads the scenes specified by the user, then participants can select 3D models as the avatars of themselves to join in Navigator. After that, the participants can roaming around the scene, and interact with other participants simply. Passive viewer is also supported, who has no avatar of itself in the virtual world but can watch the scene and avatars of other participants.

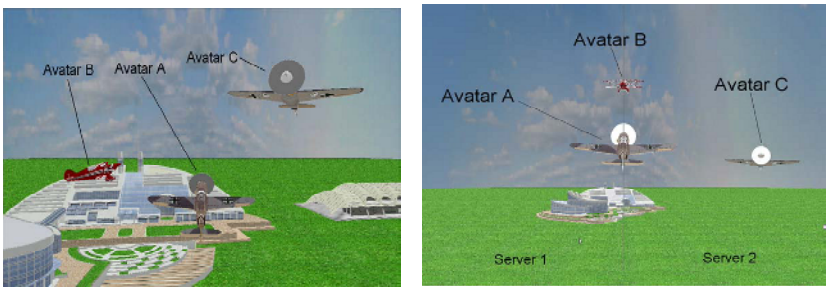
As the general implementation, Navigator Client use OpenGL to render the scene. Figure 8(a), a snapshot of a running Navigator Client, shows that three avatars (A, B and C) join in it, and A is the avatar of the participant who is viewing the window.

In some cases, large screen display and high resolution of scenes are required. To meet the demands, we employ multi-screen tiled display using MSPR [16] as

the data representation. MSPR is a retained-mode based multi-screen parallel rendering system that offers the programmers with a OpenGL-like API. MSPR system could render the scene among multiple computers in parallel.

In Navigator client with MSPR, we use three machines as a client to get a two-screen tiled display. The application logic part is run on one of the machines, and the other two are used as the rendering servers. Figure 8(b) illustrates the result of Navigator Client with MSPR. Avatar A is at the center of the client view, so one half of it is rendered by Server 1 and the other half is rendered by Server 2. Avatar B is flying from the region of Server 1 to the region of Server 2 and at the joint of them, so the image of it is also divided into two screens. But the Avatar C is rendered by Server 2 because it is fully in the region of it.

Navigator is an experimental demo. More advanced and complex applications can be developed on HIVE to meet various demands of users. Currently, we are constructing more experiments and making tests on HIVE to collect results for further performance study and improvement.



(a) Navigator client (b) Navigator client with MSPR

Fig. 8. *Navigator client*: three avatars A, B and C join in Navigator simultaneously, roaming and interacting with each other. It is from the view of Avatar A in the third view mode, so it can see avatar A itself which is always in the center of the view. *Navigator client with MSPR*: two-screen Tiled Display. The two images are captured from the two rendering servers respectively from the view of Avatar A.

6 Conclusions and Future Work

As the increasing demands of accommodating larger number of participants in VR, the scalability of participant and scalability of server are becoming the key issue. In this paper, after analyzing the requirements, we draw a conclusion that there are three fundamental factors which affect scalability of VR: node management model, management of servers and software development interface. We present a general middleware platform, HIVE, on which users can develop the scalable VR applications conveniently. HIVE has a three-tier node management model architecture to accommodate more participants, and provide an efficient management of servers to manage scalable number of clients. We employ HLA

as the reference to achieve reusability and interoperability of VR applications and provide the software development interface.

To make HIVE more efficient and practical, we will continue to explorer the other aspects of VR application, such as communication model, scene object management and rendering system interface.

Acknowledgments

This work is supported by National Grand Fundamental Research 973 Program of China under Grant No.2002CB312105 and the Key Project of Zhejiang Province under Grant No.2003C21012.

References

1. Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., Cruz-Neira, C.: VR juggler: A virtual platform for virtual reality application development. Proceedings of IEEE Virtual Reality, Yokohama, Japan, March 2001, 89-96.
2. Shaw, C., Green, M., Liang, J., Sun, Y.: Decouple simulation in Virtual Reality with the MR Toolkit. ACM Transactions on Information Systems 11, 3 (1993), 287-317.
3. Macedonia, M.R., Zyda, M.J., Pratt, D.R., Barham, P.T. Zeswitz, S.: NPSNET: A Network Software Architecture for Large Scale Virtual Environments, Presence, 1994. 3, 4, 265-287
4. Greenhalgh, C. Benford, S.: MASSIVE: A Virtual Reality System for Teleconferencing, ACM Transactions on Computer Human Interfaces, Volume 2, Number 3 (1995), 239-261.
5. Watson, K.,Zyda, M.: Bamboo - a portable system for dynamically extensible, real time, networked, virtual environments. In 1998 IEEE Virtual Reality Annual International Symposium, 252-260.
6. Frcon, E., Stenius, M.: DIVE: A scaleable network architecture for distributed virtual environments. Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments), 5(3) (1998), 91-100
7. Tramberend, H.: AVOCADO - A distributed Virtual Environment Framework. Proceedings of IEEE Virtual Reality 1999. Houston, Texas. 14-21.
8. Dongman L., Mingyu L., Seunghyun H.: ATLAS - A Scalable Network Framework for Distributed Virtual Environments, proceedings of ACM CVE'2002 , 47-54.
9. Wilson, S., Sayers, H., and McNeill, M.D.J.: Using CORBA Middleware to Support the Development of Distributed Virtual Environment Applications", Proceedings of 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen, Czech Republic, 98-105. (2001)
10. U.S. Department of Defense (DMSO): High level architecture run-time infrastructure programmer's guide, Version 1.3 v5, http://www.dmso.mil. (1998)
11. U.S. Department of Defense (DMSO): High level architecture rules, High level architecture federate interface specification, High level architecture object model template specification Version 1.3, http://www.dmso.mil. (1998)
12. Simulation Interoperability Standards Committee (SISC) of the IEEE Computer Society.: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-IEEE Std 1516-2000, 1516.1-2000, 1516.2-2000. New York: Institute of Electrical and Electronics Engineers, Inc., 2000.

13. Mark, D.H.: What is Scalability? ACM SIGARCH Computer Architecture News, 18, 4(1990), 18-21.
14. Dean, M.: The Scalability Problem, ACM Queue: Game Development, 1, 10 (2004), 66-73
15. Steve, B., Chris, G., Tom, R., James, P.: Collaborative Virtual Environment, Communications of the ACM, Vol. 44, No. 7 (2001),79-85.
16. LI, C., Jin, Z. F., Shi, J. Y.: MSPR: A Retained-Mode Based Multi-Screen Parallel Rendering System. In Proceeding of the 4th International Conference on Virtual Reality and its Application in Industry, Tianjin, China, 5444, 173-180 (2003)