HIVE: a Highly Scalable Framework for DVE*

Zonghui Wang, Xiaohong Jiang, Jiaoying Shi
State Key Laboratory of CAD&CG
Zhejiang University, Hangzhou, P. R. China
{zhwang, jiangxh, jyshi}@cad.zju.edu.cn

Abstract

With the increasing requirements for Distributed Virtual Environment (DVE): supporting larger number of participants and providing more smooth roaming and interactions, scalability is becoming a key issue. In this paper, we explore the scalability of participants and the scalability of servers, and mainly focus on three aspects: system architecture, communication model and interest mechanism. We present our middleware platform, HIVE, providing a variety of services such as data distribution, communication, event notification, etc. To achieve the reusability and interoperability of DVE applications, the interface specification of High Level Architecture (HLA) is employed as the reference. HIVE also contains the back-ends, which the middleware services depend upon. On HIVE, users can develop scalable DVE applications easily and quickly, concentrating on not the detail of distribution but the application logic. Finally an experimental demo on HIVE is given.

1. Introduction

Distributed virtual environment (DVE), as the combination of network and virtual environment, is a software system that allows users who are geographically distributed to interact with others and objects in a real-time mode.

With the development of virtual reality, network and software development technology, DVE applications have the trends as the following:

- They should support the collaboration work of large-scale distributed nodes to accommodate more participants
- As the scale of DVE system becomes larger, low responsiveness and high consistency [1] should be balanced to enhance the immersive feeling of participants
- They should be developed quickly and easily

To fulfill these demands, the scalability of participants and the scalability of servers must be improved. The key issue, which is related to the design of system architecture, is to reduce the exchange messages as many as possible [2] among the participants and servers. Most previous systems adopt one or more of the following architectures: peer/peer, client/server, or peer/server with multicast [2][3]. To avoid the delivery of unnecessary messages, interest management with message filters is employed [4][5]. Another issue is to limit the bandwidth requirements of DVE applications. Some communication techniques such as packet compression and aggregation [1] are employed. In conclusion, there are three fundamental factors leading to the scalability of DVE applications: system architecture, communication model and interest mechanism.

Our purpose is to build a general middleware platform, including the APIs and back-ends. On the platform, users can develop applications easily and quickly, concentrating more on the logic of application, and spending less time on the infrastructure of DVE, such as data distribution and management, network architecture, communication protocol configuration, event notification, etc. In this paper, we propose HIVE, which gives a middleware platform for DVE applications development.

2. System Architecture of HIVE

HIVE is a Middleware Platform, containing HIVE back-ends and HIVE library. 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, GM/Client resource 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 provide the interfaces of HIVE services. Client implements the DVE application using the HIVE library.

We present a three-tier framework: 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.

Directory manager is the process that will be launched first. It provides the global services: client and GM join / leave, client and GM resource monitor, client migration, workload computation and balance, client distribution and other global services such as "global time query", "name resolve", etc.

Group Agent is a daemon process. When the node on which the GM will run is booted, the GA process is launched. GA keeps the communication with DM, sending the "heartbeat" to the DM to indicate that the GA is alive, i.e., the node is alive. GA also

* This paper is supported by 973 Program No. 2002CB312100 and Key Project of Zhejiang Province No. 2003C21012.

IEEE Virtual Reality 2004 March 27-31, Chicago, IL USA
0-7803-8415-6/04/$20.00©2004 IEEE.

launches/destroys the GM according to the instructions from DM.

A Group Manager specifies an application. GM maintains the list of the clients which are assigned by the DM. It also communicates with and provides services to the clients, just as the agents of them.

Client can act as various roles, such as the participant of DVE, the scene object of DVE, etc. For convenience of DVE application development, we provide a set of APIs. To achieve the reusability and interoperability of DVE application, we employ the interface specification of HLA [6] as reference. The APIs are classified into four parts: time management service, user management service, ownership management service and object distribution management service.

3. Communication model and Interest mechanism

There are two kinds of communication models in HIVE. The control messages are delivered by UDP with verification, and the update messages are delivered by IP multicast.

The control messages are essential to the DVE, so it should be transmitted reliably. The traditional method is to communicate via TCP. TCP consumes considerable system resources. Especially, when a DVE application runs on the Internet and the number of participants is large, TCP is impractical. The transmission characteristic of UDP is real-time and unreliable. We use a scheme having the advantages of TCP and UDP: UDP with verification. Based on UDP, Some mechanisms, such as "Acknowledgement", "Timeout/ "Retransmission", are added to ensure that the messages can be delivered to the destination successfully. The arrival of messages out of order will not lead to the improper result, Because there is a timestamp in the message and time management of HIVE will ensure proper time advancing.

To reduce the update messages to minimum, we exploit three levels of message filtering mechanisms:

The first level is multicast IP. The whole virtual space is divided into many regions according to some specified strategies in different virtual environment, and each part will be assigned with a unique multicast IP.

The second level is object class ID. Any object in DVE has its object class ID.

These two levels of filtering mechanism are implemented in the multicast with filtering; the update messages are filtered by the multicast IP of the region and object class ID of the object class.

The third level is Object Instance ID. Any object instance in the virtual environment has its unique instance ID. When the update message is delivered to the user application process, it will be extracted to get the object instance ID. And if the object instance ID is not in the discovered object list, the message is invalid and will be discarded.

4. An experimental demo

We implement an experimental Demo, Navigator, to demonstrate how a DVE 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.

5. Conclusions and future work

In this paper, we present a general middleware platform, HIVE, on which users can develop the scalable DVE applications conveniently. HIVE has a three-tier system architecture with scalable number of servers and clients. In HIVE, a hybrid communication model containing UDP with verification and IP multicast is used to deliver messages as effectively as possible. Interest mechanism is designed to filter unnecessary messages, which can relieve the burden of network.

To make HIVE more robust and efficient, we will continue to improve error tolerance of the services as well as communication performance, and explore the strategy of workload balance.

References

[1] Jouni Smed, Timo Kaukoranta, HarriHakonen, "A Review on Networking and Multiplayer Computer Games", TUCS Technical Report NO 554, April 2002.

[2] Dongman Lee, Mingyu Lim, Seunghyun Han, "ATLAS – A Scalable Network Framework for Distributed Virtual Environments", ACM CVE'02 September-October, 2002, pp. 47-54.

[3] Tainchi Lu, Chungnan Lee, Wenyang Hsia, "Supporting Large-Scale Distributed Simulation Using HLA" ACM Transactions on Modeling and Computer Simulation, Vol. 10, No. 3, July 2000, Pages 268-294.

[4] Manuel Oliveira, Jesper Mortensen, Joel Jordan, Anthony Steed, Mel Salter, "Considerations in the Design of Virtual Environment Systems: A Case Study", ADCOG 2003, HKSAR.

[5] Wentong Cai, Percival Xavier, Stephen J.Turner, Bu-Sung Lee, "A Scalable Architecture for Supporting Interactive Games on the Internet", Proceedings of the sixteenth workshop on Parallel and Distributed Simulation, ACM, Washington, DC, May 2002, pp.60-67.

[6] 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 Draft, http://www.dmso.mil.