COVE: A Design and Implementation of Collaborative Object-Oriented Visualization Environment*

Hyung-Jun Kim, So-Hyun Ryu, Young-Je Woo, Yong-won Kwon, and Chang-Sung Jeong

Department of Electronics Engineering, Korea University, Anamdong 5-ka
Sungbuk-ku, Seoul 136-701, Korea
FAX: +82-2-926-7620, Tel: +82-2-3290-3229
hjkim@snoopy.korea.ac.kr
csjeong@charlie.korea.ac.kr

Abstract. In this paper, we present a collaborative visualization environment(COVE).Our COVE provides not only collaborative but also paralleled computing environments based on distributed object model at once. It is built as a collection of concurrent objects which interact each other and consist of two types of objects : collaborative object and application object, which are used to construct collaborative and paralleled computing environments respectively. Collaborative objects enable COVE to execute various collaborative functions, while application objects enable it to execute various visualization modes in a parallel computing environment. COVE provides a flexible and extensible framework by plugging the proper application objects into COVE, and making them interact with one another through collaboration objects. COVE is built on DOVE(Distributed Object-oriented Virtual computing Environment), a new parallel programming environment based on distributed object model. In DOVE, virtual environment is constructed as a collection of concurrent objects, each of which has its own computing power, interacts with one another by remote method invocation and those objects can be handled as the same way as local objects. Also, heterogeneity, object group, multiple method invocation to object group, object life management,and naming service of object manager are supported to provide a transparent programming environment for parallel and distributed application. We designed collaborative work manager, session manager and application manager for managing cooperative work and ray casting algorithm is adapted for visualization algorithm. Our implementation result shows that various DOVE functionalities make COVE more extensible, scalable and efficient in distributed computing environment.

* This work has been supported by KIPA-Information Technology Research Center, University research program by Ministry of Information & Communication, and Brain Korea 21 projects in 2003

J. Favela and D. Decouchant (Eds.): CRIWG 2003, LNCS 2806, pp. 42–57, 2003.

1 Introduction

During the last decade, increases in computing power of desktop computers and high-speed computer networks made it possible to build multimedia collaborative application for multi users. Such applications can solve the restriction of space for data storage, memory, and computing power, and support co-working and exchanging information between remote users. So the need of multimedia collaborative application is increasing in various field. So far, some kinds of multimedia collaborative applications supporting multi users have been developed by using several different programming paradigm, primarily socket programming model. But recently, object-oriented programming model which solves problem by interactions of objects is prevalent and distributed object models such as OMG CORBA [5], JAVA/RMI [6] and DCOM [7] have been spotlighted to tackle the problems inherent in distributed computing on a heterogeneous environment. Distributed object model provides an easy programming environment by supporting transparency of distributed objects, plug and play of software as well as the advantages of object oriented programming such as reusability, extensibility, and maintainability through abstraction, encapsulation and inheritance. However, it lacks some functionalities for distributed and parallel applications, since they are based on client-server model. It does not support group operations, and has some difficulties in implementing efficient parallelism by using asynchronous communications.

In this paper, we present a new collaborative visualization environment, COVE(Collaborative Object-oriented Visualization Environment) which enables multiple remote users participate in and discuss cooperative visualization work together. COVE is built based on our DOVE(Distributed Object-oriented Virtual computing Environment) [18], new parallel programming environments called based on distributed object model. DOVE build a virtual environment as a collection of concurrent and autonomous objects interacting with one another via method invocation. It appears to a user logically as a single virtual computer for a set of heterogeneous hosts connected by a network as if objects in remote sites reside in one virtual computer.

We designed COVE as a collection of collaborative work manager, session manager, Front-End and Collaborative application on the basis of DOVE for managing cooperative work and ray casting algorithm is adapted for collaborative visualization application.

The outline of our paper is as follows: In section 2, we describe previous works which are related to our work. In section 3, describe COVE architecture based on distributed object model. In section 4, we present our implementation result of COVE which use ray casting as a visualization algorithm. Finally, a conclusion will be given in section 5.

2 Related Work

In this section, we describe several existing distributed computing systems and heterogeneous collaborative environments based on them.

2.1 Distributed Computing Environment

Distributed programming model can be broadly classified into message passing model and distributed object model. In message passing model, a program is divided into components or tasks, which may run on different nodes of machines. The tasks communicate with each other by explicitly sending and receiving messages. In distributed object model, often called method invocation model, a program consists of distributed objects which interact with one another by method invocation. PVM [9] and MPI [8] are distributed programming systems based on message passing model, while CORBA [5] and Legion [10] based on method invocation model.

MPI is a single standard programming interface mainly designed for developing high performance parallel applications with emphasis on a variety of communication pattern and communication topology. However, MPI lacks in functionalities such as process control, resource management and fault-tolerance. PVM is one of the most widely used distributed computing systems based on the message passing model, and connects together separate physical machines into a virtual computer by providing process control, simple message passing and dynamic process group management. In PVM, a daemon process, which runs on each host of a virtual machine, is used not only as process controller but also message router, which may result in communication bottleneck as all tasks heavily depend on daemon processes.

Legion is an architecture based on a distributed object model and designed to build system service which provides a virtual machine using shared object, shared name space, fault-tolerance. Legion uses data flow model as parallel computation model, and parallelisms are implicitly supported by the underlying runtime system. However, management of data dependency graph for every invocation as well as scheduling nodes of the graph may incur additional computation overhead, and no support for object group may cause communication inefficiency. CORBA is a vender-independent standard which aims at interoperability and portability of distributed applications. CORBA defines a distributed object model for accessing distributed objects. It includes an Interface Description Language, and a specification for the functionality of run-time systems that enable access to objects. But CORBA is based on client-server model rather than a parallel computing model, and hence it is not adequate to provide a virtual machine. Also, it is not well suited to distributed applications where performance requirements demand asynchronous communication and group operations.

2.2 Previous Collaborative Environments

In **CVE** system, multiple users can interact with each other in real-time, even though they may be physically located in different places around the world. This virtual environment provides users a sense of realism by incorporating realistic 3D graphics and stereo sound into the computer-human interface to create an immersion experience. CVE system gives the users a shared sense of space, and shared sense of time. It also provides users with the natural ways of communications and interaction.They achieve their collaborative works with several their

own unique characters and purposes. For example, In **SmartCu3D** [1] system, an Internet CVE system, distributed users communicate one another with Behavior Based Interaction Management. **DISCOVER** [2] is CVE project to train teams for emergencies on ships or rigs. **SciCentr** [3] CVE system is intended to become a meeting place, a workplace, and a showcase for the power of the Internet as a medium for informal science education aimed toward teens and young adults. **Industrial tele-training CVE** [4] is designed for training the operation of ATM switch equipment. These CVEs use specific 3D visualization and immersion effect to help user's collaborative work. However,it is not easy to extend them to other area of part because they have developed with specific their purpose. And they do not have accelerate operation for Problem Solving which need High Performance Computing power.

Shastra [11,12] is a scientific groupware developed at Purdue University. It is composed of multi-layer architecture, where each layer has the responsibility of connection, communication and collaboration, and supports several tools for useful scientific collaboration. However since Shastra is based on DCE(Distributed Computing Environment), it is neither completely object-oriented nor suitable for applications that use distributed object.

MAESTRO [13] is a distributed multimedia collaborative environment developed at Postech University. MAESTRO provides a rich multimedia collaborative service API which can be used to develop a variety of multimedia applications easily. MAESTRO API and its underlying service have been modeled and implemented using the distributed object-oriented approach. CORBA has been used to design MAESTRO multimedia collaborative services and Orbix has been used for its implementation.

Sieve [14] is Java-based collaborative modular visualization environment (CMVE) for construction of interactive visualization. Sieve provides many benefits to data-analysis through data-flow network creation. It supports data analysis over the Web by multiple users' collaborating in real-time. Furthermore, Sieve demonstrates several experimental techniques for using the JavaBeans component architecture to build collaborative interactive systems.(ex.Interaction, Module Configuration, Collaboration and Persistence and Publication) Sieve presents the user with a large, scrollable workspace onto which data sources, processing modules, and visualization components may be dropped, linked, and edited. Modules representing data sources may be written for a wide range of raw data formats and sources. These may include objects which parse files retrieved over the Web, objects which access SQL or other databases, and objects which retrieve data from remote servers using CORBA, Java's RMI library, or proprietary protocols. It has good portability for heterogeneous platforms and Object Oriented Model because it uses Java and CORBA. However it does not support High performance Parallel computing. So this system is not good for the task modules which need high computing power.

CoVis [15] is Collaborative Visualization Learning system with remote high school students, teachers and scientists. It was developed at Northwestern University by funding of NSF. Participating students study atmospheric and environmental sciences through inquiry-bases activities. Using state of the art sci-

entific visualization software, specially modified to be appropriate to a learning environment, students have access to the same research tools and data sets used by leading-edge scientists in the field. The CoVis Project provides students with a range of collaboration and communication tools. The merit of this system is to support various and helpful functionality for Science Studying. This provides: desktop video teleconferencing ; shared software environments for remote, real-time collaboration; access to the resources of the Internet; a multimedia scientist's notebook; and scientific visualization software. However, this is too specified to only Science Learning. it is not easy to extend and apply to other goal and area. So, it is not suitable to generic collaborative framework that supports collaboration of ordinary application to multi-user.

CSpray [16] stands for Collaborative Spray rendering and is an extension of Spray, a visualization application, into a collaborative environment. Spray rendering is a framework which has been developed for visualization using a spray can; cans are filled with smart paint particles(sparts) that are sprayed into the data to highlight interesting features. Features are displayed when sparts become activated and leave visualization objects in their path. In order to support collaboration, CSpray is obtained several features. In CSpray, several users in a visualization session may analyze a set of distributed data by creating spray cans loaded sparts. Spray cans may be made public or kept private. Public cans are visible and accessible by other participants. Participants can see the spraying action of other users in their local window. More than one participant may be spraying any given time. Once in a while, the attention of the entire group may be directed at what one of the participants is doing. Participants may also join and leave the session at any time. However, CSpray is based on a message passing paradigm, so it has no virtues of object oriented paradigm.

Recently, WWW has become a popular word for application programmers. And there have been another approaches to develop collaborative applications on WWW environment. JETS(Java Enabled Telecollaboration System) which was developed at University of Ottawa is one of those approaches on WWW environment. JETS [17] is a collaboration system based on Java applet. Because Java applet is a mobile code, JETS has the advantages of no previous download or installation. What user in the client part should do is just browsing and navigating through web page. However JETS has the problems such as security restriction and performance problem of Java applet.

3 COVE Architecture

3.1 Overall Architecture

COVE(Collaborative Object-oriented Visualization Environment) is a framework for collaborative visualization application which has three layered architecture. It consist of three layers such as Collaborative Application(CA) layer, Collaboration Environment(CE) layer and Parallel Computing Environment(PCE) layer.(see Figure 1.) CA layer provides visualization and computation application modules such as ray caster, ray tracer, image processing module and learning

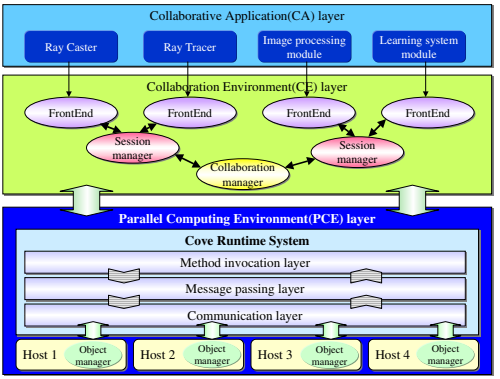

Fig. 1. Overall COVE architecture

system module. These modules can be easily plugged into a FrontEnd object which is provided by CE layer. Using this module plug-in feature, user can easily construct flexible collaborative application. CE layer provides several useful collaborative components to construct more efficient and easy-to-use collaborative visualization environment:*FrontEnd*, *Session manager* and *Collaboration manger*. PCE layer provides feature of distributed object model and consist of three sublayer such as *method invocation layer*, *message passing layer* and *communication layer*. method invocation layer supports functionality of remote method invocation and it uses function of message passing layer for sending and receiving an invocation request message and result reply. Communication layer provides various communication protocols to upper two layers. PCE layer also manages usable computation resources. For the purpose of managing computation resources, PCE layer uses special distributed object called *Object Manager*. Object Manager exists per Host, and provides crucial services such as object creation and naming service. PCE layer implementation is based on DOVE [18].

3.2 Parallel Computing Environment(PCE)

Distributed Object Model. COVE is based on distributed object model which consists of several distributed objects interacting with each other using method invocation mechanism. Distributed object is divided into *interface object* and *implementation object*. (See figure 2.) The interface object is distributed to applications which are to use the distributed object, and it provides interaction point to its corresponding implementation object. Users can issue a method invocation to the distributed object by invoking the method of its interface object, a local representative of the distributed object. Method invocation to the interface object is converted to the invocation message by *stub object*, and sent to the corresponding implementation object by *COVE run-time system*. On the opposite side, the receiver's run-time system unmarshals the invocation message, and invokes the appropriate method of the target implementation object

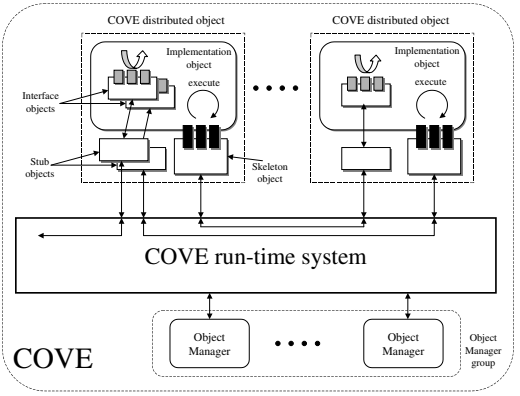

Fig. 2. Distributed object model

through *skeleton object*. A reply message is sent from the implementation object back to the interface object, and returned like a normal function call. This mechanism, which is called *remote method invocation*(RMI), allows transparent access to the object irrespective of whether it resides in local or remote site. In COVE, distributed object behaves either as client or server to interact with other distributed objects. In other words, it executes the implementation object for incoming RMI requests, while during the execution of the implementation object, it generates a remote invocation to the other distributed object using the interface object as a client. In COVE, *object manager* exists per host and it provides a set of indispensable services, such as object creation, deletion and naming services, to build a transparent and easy-to-use clustered computing environment. A set of object managers constitutes a single object group which determines the domain of the virtual parallel environment that might encompass a huge number of machines and networks. The main features of COVE object model are as follows.

Concurrency Enhancement Scheme. COVE provides three kinds of RMI to support various synchronization modes during data exchange between remote objects: *synchronous*, *deferred synchronous* and *asynchronous* RMIs. In synchronous RMI, sender is blocked until the corresponding reply is arrived. In deferred synchronous call, the sender can do other work immediately without awaiting the reply of the RMI, but later at some point must wait for the reply in order to use the return values. In asynchronous RMI, the sender can proceed without awaiting the reply similarly as in deferred synchronous one, but the corresponding *upcall* method is invoked on the arrival of its reply. These synchronization schemes are fully supported by multilayered architecture in PCE layer. In distributed system, group communication pattern is often used, since it provides simple and powerful abstraction. In COVE, group communication mechanism is supported by introducing a new construct, *object group*, as means

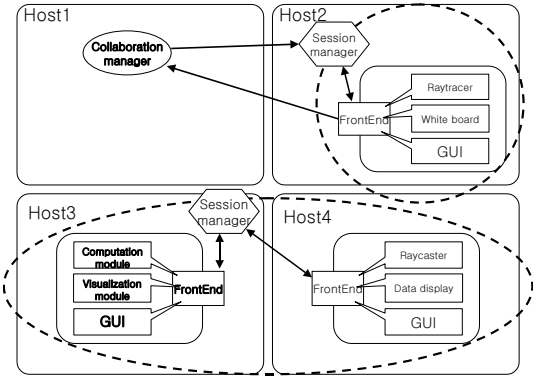

Fig. 3. Components of Collaborative Environment

of grouping objects and naming them as one unit for RMIs. An interface object can be bound to its corresponding object group, and a RMI issued on the interface object is transparently multicast to each implementation object in the group. Interface object to the object group has the same interfaces as the one to the single object, and provides an interaction point with multiple objects in the object group so that user treats it just like a single object. Therefore, the concept of object group allows users to do more simple programming, and to have chance to get better performance if the underlying communication layer supports multicasting facilities.

3.3 Collaborative Environment (CE)

COVE(Collaborative Object-oriented Visualization Environment) is a framework for collaborative visualization application. Users on remote site can participate in cooperative visualization work, exchange various information they have and discuss current visualization work in COVE. All those collaborative functionalities of COVE is implemented as several components in CE layer. Each component in CE is designed as distributed object which is supported by PCE layer and communicates with each other by remote method invocation. Each object has its own name provided by PCE layer and this makes it possible to distinguish all objects from each other guaranteeing its unique existence in COVE environment.

To develop COVE, we need to define several collaborative components to construct more efficient and easy-to-use collaborative visualization environment. The first, we need a CollaborationManager managing and controlling entire collaborative environment. The second, we need a SessionManager managing a collaborative visualization session. And the third, we need a FrontEnd which can interact with user via GUI and manage visualization and computation module according to user interaction directly. FrontEnd has a standard interface for connections with various visualization and computation modules. So the user

can easily plug in FrontEnd with the modules which serve the purpose of collaborative work, such as ray caster, ray tracer, wire frame viewer and so on. The relationship between above three COVE components are shown in figure 3. Design and implementation details for COVE components are as follows.

CollaborationManager. CollaborationManager is a COVE component, which manages and controls entire collaborative environment. It maintains information about all other components and provides object registration service, session creation service, and environment information service in COVE. Also, it maintains and updates all SessionManager alias and FrontEnd alias as a list. Therefore CollaborationManager has the responsibility for registering and deregistering FrontEnd, creating SessionManager, terminating SessionManager, and sending session list and user list to FrontEnd requesting them. Besides above functionalities, it provides mechanism to guarantee unique object alias for each component in COVE environment.

SessionManager. SessionManager is a COVE component managing and maintaining all information concerning session as well as joining and leaving session service. All information about remote user joining the session is stored in SessionManager and data distribution service and access control service are achieved by contacting SessionManager. To distribute visualization data and result data to all other remote users more fast and efficiently, each SessionManager has a FrontEnd group consisting of FrontEnds contained in the same session. SessionManager registers FrontEnd alias to its FrontEnd list and also add it to FrontEnd group whenever joining-session request is received by a FrontEnd. When data distribution service is requested by a FrontEnd, SessionManager receives and sends requested data to all remote users in a FrontEnd group at once by multicasting which PCE layer support. When designing collaborative application supporting multi-users, we should add data access control to it. For access control service in COVE, we adopted floor control pattern. Floor control is a manner to determine the access order to shared data of each FrontEnd. We implemented several modes of floor control, such as round-robin mode, baton-passing mode and etc so that user can choose whatever floor control he wants.

FrontEnd. FrontEnd is a COVE component which interacts with user via GUI and manage visualization and computation modules according to user interaction. It contains GUI for user interaction such as creating and joining a session, rendering image, exchanging visualization data, choosing flow control and etc. When a user interaction is retrieved from GUI, FrontEnd processes it by calling predefined callback function. FrontEnd supports efficient way for data exchanging between remote users. When a FrontEnd joins a session, it is simultaneously added to FrontEnd group. By joining FrontEnd group, one FrontEnd can send and receive collaboration data via group method invocation. This increases communication performance in a collaborative application as the same data should be transferred repeatedly to all users unless there is group method invocation.

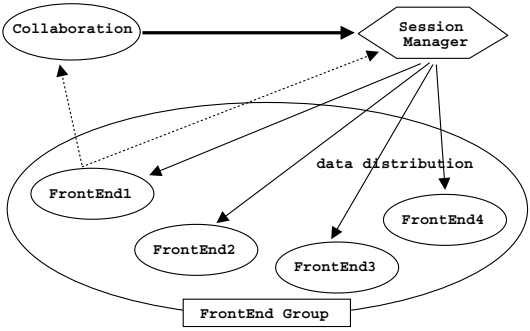

Fig. 4. Session manager: FrontEnd1 requests new session creation to CollaborationManager and CollaborationManager creates a new SessionManager and session. FE1 joins created session and it becomes a session leader. When a FrontEnd participates in one session, it is added to FrontEnd group of that session, and data distribution to all FrontEnds can be done at once by using FrontEnd group which support multicasting

Besides this, FrontEnd manages visualization and computation modules for cooperative work. In COVE, these modules are developed regardless of COVE environment. All collaboration-related processes such as data exchanging and task distribution are assigned to FrontEnd. Various modules are simply plugged-in to FrontEnd and this makes COVE more extensible and easy-to-use for various kinds of visualization applications.

3.4 Collaborative Application (CA)

CA layer provides various modules for collaborative application such as ray caster, ray tracer, image processing module, learning system module and so on. These modules can be easily plugged into a FrontEnd object which is provided by CE layer. Using this module plug-in feature, user can easily construct flexible collaborative application. In this paper, Ray casting is chosen as an application visualization algorithm. We designed three collaborative visualization mode in Ray Caster of COVE application - parallel mode, rotated mode, multiple mode. Each mode uses extended space-leaping method [21] to accelerate rendering speed and is designed to have its own unique feature for fast and efficient rendering in collaborative environment. In COVE, we currently provide only simple text chatting functionality for communications between remote users. Adding more complicated and enhanced communication system to COVE is left as future work.

4 Implementation and Experiment

4.1 Ray Casting Application Implementation

For the experiments and evaluation, we developed a collaborative volume rendering application which uses ray casting algorithm on COVE(Its GUI is seen

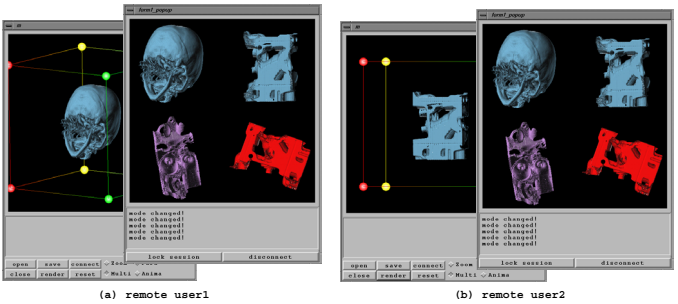

Fig. 5. Interface of Collaborative Visualization Application on COVE: (a)shows windows of one remote user and (b) shows of another remote user

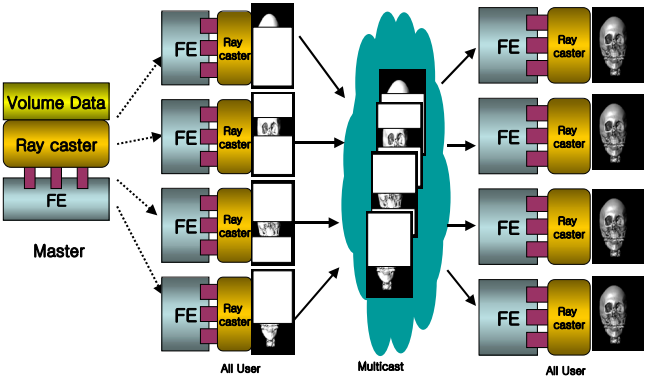

Fig. 6. Illustration of Parallel Visualization

at the Fig. 5). This application has three rendering modes such as parallel visualization mode, rotated visualization mode and multiple visualization mode. detail of each mode is as follow.

Parallel Visualization Mode. As ray casting is a highly time consuming process, many ray casting acceleration techniques and parallel algorithms have been developed. Parallel visualization mode is designed to render one image as fast as possible by dividing task into small subtasks and then assigning them to all remote users participating collaborative work. For fast and efficient ray casting, we used extended space-leaping technique and image based parallel algorithm.

In parallel visualization mode, only rendering leader have right to render volume image. That is to say, when rendering mode is set to this mode, only

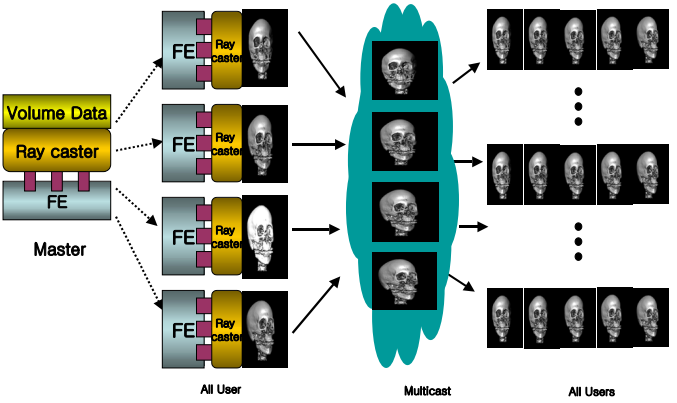

Fig. 7. Illustration of Rotated Visualization

leader can choose viewing direction and render image by pushing render button. Other remote users can do nothing but provide their computing resources to parallel collaborative work and examine the received result image. This is for high performance of parallel rendering process by preventing remote users' individual usage of computing resources.

Our parallel visualization consists of three phases as follows: In the first phase we find active pixels and active depths by using forward projection as follows: Initially, the volume data is partitioned among various processes. Each process finds non-empty voxel runs for its partitioned volume slice, and then executes forward projection using line drawing algorithm for each voxel run, finally returning active pixels to the master process. In the second phase, for job assignment, the active pixels obtained in the first phase are distributed among participators on COVE from the leader of the session. The value of each active pixel on the screen is calculated as follows: Generate a ray through the active pixel into the data space. Starting at the nearest active depth where the ray intersects non-empty voxel, follow the ray while sampling the volume at constant interval. Accumulate the color and opacities of these sampled values. Stop following the ray when it is known that it cannot significantly change its value, or when it intersects the farthest active depth. In the third phase, the resulting partial images obtained from the second phase are merged to yield the final image in the master process.

Rotated Visualization Mode. Ray casting algorithm visualizes 3-dimensional feature of an object on the screen. When visualizing 3-dimensional object, it is necessary to display the same object from various viewing direction by incrementally rotating it. Animated visualization mode is designed to produce multiple images of one object from different viewing direction at once. All remote users are assigned his own viewing direction from leader and render images corresponding to given viewing direction simultaneously and then result images are exchanged

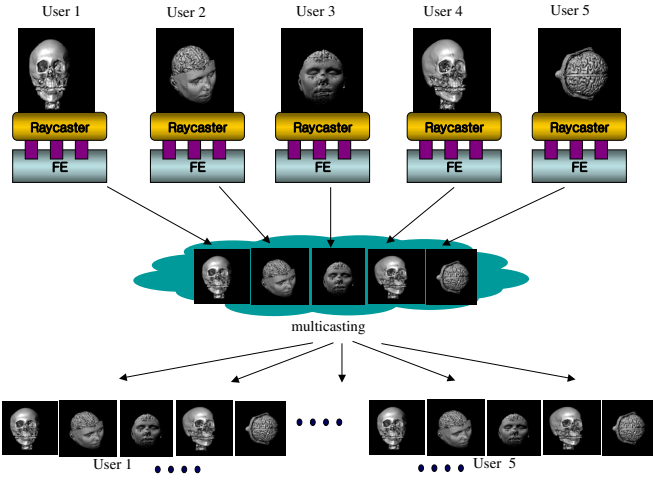

Fig. 8. Illustration of Multiple Visualization

to each other. This mode enables remote users see incrementally rotated images within constant time regardless of the number of images.

In animated visualization mode, only rendering leader have right to render volume image as parallel visualization mode. This is for high performance of rotated rendering process by preventing remote users' individual usage of computing resources.

Multiple Visualization Mode. Both parallel visualization mode and rotated visualization mode restrict remote users' rendering right except rendering leader to use remote computing resources more efficiently. But in collaborative visualization environment, it is necessary to allow all remote users to render individual images. Individual image means different data sets, different viewing direction, different shading effect and so on. Also rendering processes on each remote site can be implemented independently in time by user interaction. Multiple visualization mode is designed to allow all remote user to have his own data set and choose his own viewing direction he is interested in. In this mode, whenever each remote user render an image, rendered result image is sent to all other users to share various kind of result images. More detailed rendering process for this mode is as follows:

When rendering mode is set to multiple visualization mode, all collaborative participants can examine various images of different data sets, shading effects and from various viewing direction simultaneously. And this makes it possible that all users can share distributed information on remote sites.

Table 1. Machine specifications

Machine type	M_1	M_2	M_3	M_4
Model	Pentium IV PC	USparc1	O_2	$Octane$
CPU	P IV	UltraSPARC	MIPS R10000	
Clock(MHz)	1740	143	150	250
Memory(MBytes)	1024	128	128	512
OS	Linux 2.2	Solaris 2.5	IRIX 6.3	IRIX 6.5

Table 2. Measurement of relative performance with respect to M_1 for ray casting

machine i	M_1	M_2	M_3	M_4
OS (spec.)	Linux (PIV-1.7G)	Solaris2.5 (USparc1)	IRIX6.3 (O_2)	IRIX6.5 (Octane)
running time	103.01	413.812	205.390	128.690
relative perf.	1.0	0.249	0.502	0.800

Table 3. Performance results of parallel ray casting on COVE

number of machines		1(M_1)	2($M_{1,4}$)	4($M_{1,2,3,4}$)	8($M_{1,1,1,1,2,2,3,4}$)	11($M_{1,1,1,1,1,1,1,1,2,2,3,4}$)
expected speedup		1.0	1.8	2.551	6.351	9.551
COVE	time (sec)	103.01	70.01	50.012	20.844	13.912
	speedup	1.0	1.471	2.060	4.942	7.404
	efficiency (%)	100.0	81.72	80.75	77.81	77.52

4.2 Experimental Results

We implemented the parallel rendering on COVE with participators which consists of 12 heterogeneous machines, two Ultrasparc1, one SGI O2, a SGI Octane, eight Pentium IV PCs running Linux connected by 100 Mbps Ethernet. Our test data set is a 256 x 256 x 225 human head, and image screen measures 1024 x 1024 pixels. The details of hardware and software information for each machine are shown in table 1.

Since each machine has different computing power, we have measured the relative performance with M_1 as reference machine by comparing the execution time of the identical sequential ray casting program on each machine. Then, the expected speed up is computed as a sum of each relative performance of participating machines. The relative performance of the machines obtained by executing the identical sequential ray casting program is shown in table 2. Table 3 shows the execution time, speed up and efficiency of ray casting according to the number of machines. The efficiency represents the ratio of speedup with respect to expected speedup. As the number of machines increases, the parallel algorithm shows relatively good speed up with efficiency around 80% without degrading its performance due to the communication overhead. The image which is made through this parallel visualization is scattered to every participator. So, they can see the image much faster using COVE's parallel visualization than using their own single machines.

5 Conclusion

In this paper, we have presented a COVE(Collaborative Object-Oriented Visualization Environment) which provides a flexible and extensible framework for collaborative visualization by integrating collaborative and parallel computing environments based on distributed object model. It has been built as three layers : Parallel Computing Environment(PCE), Collaboration Environment(CE), Collaborative Application(CA).

PCE has been designed to provide an easy-to-use transparent distributed and parallel programming environment for networked heterogeneous computers. Besides the traditional client-server model, it offers a peer-to-peer parallel model of computation by considering a parallel application as a collection of distributed objects which interact with each other. Efficient parallelism is supported by two concurrency enhancement schemes which support various types of synchronization methods and RMI for object group. CE has been designed to provide an efficient collaborative environment by designing collaborative objects such as FrontEnd, Session Manager, and Collaboration Manager, while CA provides users for application objects implementing specific functions. The plug-in of different application object into collaborative object, FrontEnd, allows application developers to easily construct a collaborative environment for diverse applications. Therefore, COVE can provide a flexible and extensible collaborative environment for not only visualization but also other many applications such as image processing, distant learning, etc. Three visualization modes are designed and implemented to support a fast and flexible analysis of visualization data. Parallel visualization mode has been designed for the fast generation of a volume image, rotated visualization mode for the generation of animated volume images, multiple visualization mode for the generation of different volume images. We have shown the experimental result for parallel visualization mode on COVE by executing the parallel ray casting algorithm rotating the volume image on COVE.

References

1. Weihua Wang,Qingping Lin, Jim Mee NG, Chor Ping Low: "SmartCU3D: a Collaborative Virtual Environment System with Behavior Based Interaction Management", VRST'01, ACM, Nov 2001.
2. Susan Turner, Phil Turner, Liisa Dawson and Alan Munro: "DISCOVERing the Impact of Reality", CVE 2000, San Francisco, ACM, 2000.
3. Margaret Corbit, Bonnie De Varco: "SciCentr and BioLearn: Two 3D Implementations of CVE Science Museums", CVE 2000, San Francisco,ACM,2000.
4. J.C. de Oliveira, S. Shirmohammadi, and N.D. Georganas: "Collaborative virtual environment for industrial training", Virtual Reality 2000,IEEE, 2000
5. Object Management Group Inc., The Common Object Request Broker: Architecture and Specification, OMG Document Revision 2.2, February 1998.
6. T. B. Downing, Java RMI: Remote Method Invocation, IDG Books worldwide, 1998.

7. E. Frank and III. Redmond, DCOM: Microsoft Distributed Component Object Model, IDG Books worldwides, 1997.
8. MPI Forum, MPI: A Message-Passing Interface Standard, International Journal of Supercomputer Application 8, No. 3, 1994.
9. A. Geist, A. Beguelin and et al., PVM 3 User's guide and Reference manual, ORNL/TM-12187, September 1994.
10. M. Lewis and A. Grimshaw, The Core Legion Object Model, University of Virginia Computer Science Technical Report CS-95-35, August 1995.
11. Anupam,V. "Shastra – An Architecture for Development of Collaborative Applications" Thesis for the degree of Doctor, Dept. of Computer Science Univ. of Purdue, 1995.
12. Anupam,V. and Bajaj,C., "Collaborative Multimedia Scientific Design in Shastra", Proc. of the ACM Internation Conference on Multimedia, ACM Press, August 1993.
13. T.H.Yun, J.Y.Kong and J.W.Hong, "Maestro: a CORBA-based Distributed Multimedia System", Proc. of 1997 Pacific Workshop on Distributed Multimedia Systems, Vancouver, Canada, July, 1997, pp. 1–8.
14. Philip L. Isenhour,James Bo Gegole, Winfield S. Heagy, Clifford A. Shaffer, "Sieve: A Java-Based Collaborative Visualization Environment", IEEE Visualization '97 Late Breadking Hot Topics Proceedings, Oct 22–24, 1997, pp. 13–16.
15. CoVis Project URL: http://www.covis.nwu.edu/
16. Alex Pang, Craig Wittenbrink "Collaborative 3D Visualization with CSpray", IEEE Computer Graphics, Vol. 17, No. 2, 1997, pp. 32–41
17. Shirmohammadi, S. and Georganas, N., "JETS: a Java-Enabled Telecollaboration System", Proc. IEEE ICMCS, IEEE Computer Society Press, Los Alamitos, Calif., 1997, pp. 541–547.
18. C. S. Jeong and H. D. Kim, "DOVE: A Virtual Programming Environment for High Performance Parallel Computing,"
19. J. Frey, S. Graham, C. Kesselman: "Grid Service Specification. S. Tuecke, K. Czajkowski, I. Foster," Open Grid Service Infrastructure WG, Global Grid Forum, Draft 2, 7/17/2002. Lecture Notes in Computer Science, May 2000, pp. 12–21.
20. I. Foster, A. Roy, V. Sander: "A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation," 8th International Workshop on Quality of Service, 2000.
21. S. U. Jo and C. S. Jeong, "A Parallel Volume Visualization Using Extended Space Leaping Method," Para 2000, Norway, July 2000, pp. 398–403.