A Web-Based Distributed and Collaborative 3D Animation Environment

Tain-chi Lu, Chung-wen Chiang, and

Chungnan Lee

Institute of Computer and Information Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan, ROC
e-mail: tclu@cie.nsysu.edu.tw

Tong-Yee Lee

Department of Computer Science and Information Engineering
National Cheng-Kung University
Tainan, Taiwan, ROC

Many applications on the Web require active processing and coordination of services. In this paper, we describe the design of a distributed 3D animation system built by integrating the Java language, Parallel Virtual Machine (PVM) software, collaborative mechanism, 3D computer graphics, and the Web technologies. To achieve the collaborative cooperation and functional independence, session control and system agents are devised in this system. In particular, we propose a simplified collaborative group definition, collaborative policies, and the state of participants to dynamically manage participants in this Web-based distributed environment. Based on our proposed mechanism, the server can efficiently determine the status of collaboration activities.

INTRODUCTION

In addition to using Java applets for visualization and information exchange, Java [1] gains popularity as a language for parallel computing on the Web. Though it is possible to use Java alone as a language for the parallel computing of the scientific application [2], it is worthwhile to incorporate Java applets with the existing distributed computing environments, such as PVM [3], MPI, etc. The marriage of Java language and existing computing environment provides a quick and reliable Web-based distributed computing environment.

3D animation has many important applications such as film industry, scientific visualization, etc. However, it suffers from the lack of collaborative environment and needs large computation power. With collaboration capability, it can help people work together and perform common tasks in a shared environment. A collaborative environment must satisfy the computer supported cooperative work (CSCW) requirements - sharing an information space to design an effective technical system. To solve the computation needs for the generation of photorealistic 3D animation, we employ PVM as the distributed computing technology. The proposed system is an interactive, distributed 3D animation system across the Internet with the emphasis on the collaboration among different participants.

AN OVERVIEW OF SYSTEM ARCHITECTURE

To generate photorealistic image, we exploit raytracing algorithm which is very computationally intensive. Our distributed animation system is a client-server architecture. In the client site, we use HTML incorporated with Java applet to implement a front-end Graphical User Interface (GUI). With this arrangement, the WWW browser accepts user's service request and then evokes a Java applet execution. This Java applet will send the environment's parameters to the Web server for further processing. The system architecture is illustrated as Figure 1. A user first accesses our service from any WWW browser. Then a Java byte-code encapsulating the front-end GUI is shipped to the client site. From the front-end GUI, the end-user can configure the PVM heterogeneous computing environment by adding or deleting computing hosts. The Java applet establishes a reliable socket connection with a remote PVM daemon and sends the system configuration to the PVM daemon. This PVM daemon later will execute pvm_addhosts routine to form a virtual parallel machine. Similarly, we send the graphics rendering information to the parallel raytracer or send back the rendering result to the WWW browser through the same socket connection. The socket connection in the Web client uses a specified port to communicate with server socket. We assemble the client socket and server socket into a communication agent. The characteristics of the communication agent is functional independence. Both sockets just send the signals or messages to the communication agent. Other agents like interface agent, session control agent, and token management agent, will be explained in the next two sections.

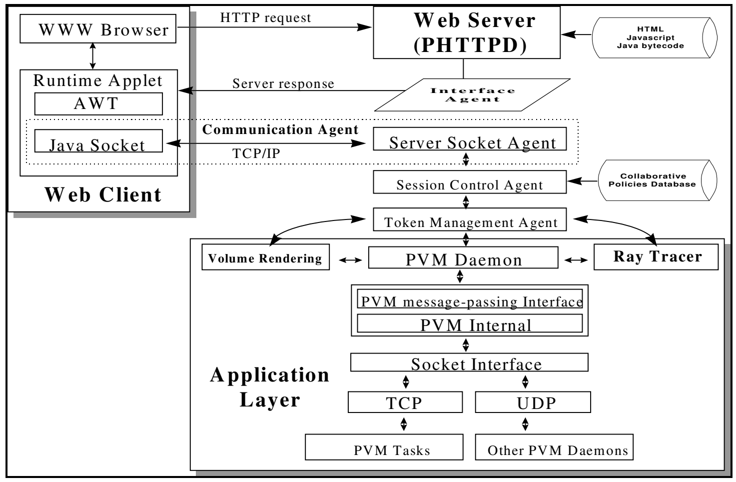

Figure 1. The high level block architecture of the distributed 3D animation system

COLLABORATION MODEL

Collaboration [4] means that a group of users work together on the same problem. In such a situation, awareness of individual and group activities is an important issue, especially for a distributed environment. Awareness is fundamental to coordination of activities and sharing of information. Collaboration control mechanism regulates how multiple users assemble and interact over the shared data.

Session Control

Users can login into a collaboration group by sending an HTTP request. He or she can either join an existing group or create a new group. A group of collaboration lasts until the group is terminated by the system or when nobody has a desire to be the leader. In this subsection, we propose a simplified collaborative group definition (SCGD) [4] that is a simplified description of the syntax, or form, of individual statements. Table 1 shows a set of language specification, which is in the Backus-Naur form, for the SCGD. The syntax provides a description for efficiently collaborative session in the system.

Table 1. Simplified collaborative group definition (SCGD)

Item	Definition
1	<Group> ::= GROUP <Group-name> <Topic> <State> <Group-work-attributes> <Group-manager>
2	<Group-name> ::= ID
3	<Topic> ::= <Statement>
4	<State> ::= "Private(Invite only)" | "Public"
5	<Group-work-attributes> ::= <Participant> | <Group-work-attributes> + <participant> | <Group-work-attributes> - <Participant>
6	<Group-manager> ::= <Participant> <Manager-candidates>
7	<Manager-candidates> ::= <Participant-name-list>
8	<Statement> ::= CHAR | ϕ
9	<Participant> ::= <Participant-name> <Participant-attributes>
10	<Participant-name> ::= ID
11	<Participant-name-list> ::= <Participant-name-list>, <Participant-name>
12	<Participant-attributes> ::= <Specialty> <Responsibility> <Location>
13	<Specialty> ::= <Statement>
14	<Responsibility> ::= ITEM
15	<Location> ::= <Statement>

Collaboration Policies and Mechanism

Session control encompasses the management of participation, authentication, and presentation of coordinated user interfaces. The session manager provides a conduit for control. In the following we define collaboration policies and mechanisms through some operators operating on the set and its elements. A group $G = \{x | x_i \in G, i = 1, 2, 3, \ldots, n\}$ is defined as an ordered set with finite nodes, each node is a variable that has state and information.

Collaboration policies are defined as a set of unary or binary operations on G_i and its elements, i = 1, 2, 3, …, n. We list these operators in the following:

Empty (G_i) : When a group G is empty, it means that the group doesn't have a leader or any participant. The session control agent will terminate an empty group.
Leader (G_i, x) : It assigns x to be the group manager of the set. x will be placed in the first element in the ordered set.
Participant (G_i) : Participant (G_i) = $\{ x_i \mid x_i \in G_i$, i = 1, 2, 3, …, n$\}$. The operator of element is to find out

the participants in a group.
Join_permission (G_i , x) is a mapping $G_i \times x \longrightarrow$ True or False. When a new participant wants to join a group G, he must negotiate the group manager and wait for the response. If the mapping is true, then x is allowed to join the existing group G_i ; otherwise x is not allowed to join the group G_i .
Create (G_i , x) : A user can create a new working group in the collaborative system. The group information includes group name, topic, state, group work attributes, and manager.
Append (G_i , x) = $G_i \cup \{ x \}$ When x is allowed to join the group G_i , it is appended to the last element.
Participant_information (x_i) : The information of a participant includes the name, specialty, responsibility, and location.
Leave (G_i , x_i) = G_i - { x_i } To leave a group, a user must inform the group manager. If the group manager wants to leave, the rule to choose a new group manager is described as follows : ● The group manager should find a new group manager among other participants. ● If there is a participant that wants to be the group manager, this group will be chaired by this new manager; otherwise, the group is terminated.

To keep track of the state of each participant is important to manage the participant in the collaborative model. We use the *Finite State Machine* concepts developed on sets and functions to maintain the state information of each participant. By using the concepts, the collaborative environment in server mode can simply be an input-output device. The system agent concept can be the media to be transferred from input to output.

SYSTEM AGENTS

An agent is the actual site of media interaction for a user [5]. It may actually implement media interaction functionality, or may use orthogonal abstract mechanisms to achieve the same effect. All agents are programmed in Java and executed as byte-code by Java Virtual Machines (JVM). These agents supporting team work at the same time in different places are within synchronous distributed interaction or real time interaction. There are four classes of agents in the system - namely, interface agent, communication agent, session control agent, and token management agent. The interface agent provides intuitive methods for session, interaction, and coordination control, and supports media-rich communication. It accepts requests from different modalities and packs user's requests in a form that will be transmitted to session control agent for further processing by communication agent. The communication agent is responsible for communication among Web clients and Web servers. Communication agents not only can employ network protocols such as TCP/IP to transmit messages among themselves, but they also retrieve data holdings in various native formats and dictate how the session control agent interpret and process these messages.

The session control agent provides event-driven notification services by requesting collections according to user scripts. The session control is in charge of client registration, request submission, result delivery, and delivering locked tokens' status to token management agent etc. In addition, the session control agent is responsible for establishing and maintaining groups of users, and coordinating the activities of the various modules that are operating in the conference mode. It regulates how multiple users assemble and interact over shared data.

In the system, we use the term 'token' to include resources of all kinds such as rendering database. A fixed number of tokens in a form of object are used to communicate and share among the processes of the system. Each token can be held either a group or by the token management agent which maintains the token pool. A situation like this, where several participants from the same group access and manipulate the same data (token) concurrently, and the outcome of the execution depends on the participant order in which the access takes place. To guard against the situation above, we use some control mechanism to ensure that only one participant at a time can manipulate the data. In our implementation, a manager is selected among the members of the same group.

PARALLEL RENDERING ALGORITHM

The efficiency of the parallel computing is critical for the computationally intensive applications, such as ray tracing, radiosity, etc. In this section, we first discuss the global distributed control (GDC) algorithm, which has ability to provide a dynamic ability to adjust load imbalances and to provide fault tolerance.

In the past, we have exploited GDC to balance raytracing on the multicomputer system. GDC method is a kind of decentralized parallel computation environment. Applying the GDC method to distributed computing environment is in [6, 7] where detailed results are available. Here, we present some new experimental results of the fault tolerance ability in our system.

Fault Tolerance

Because ring's construction is so weak that any lost message may incur a deadlock. Thus, it is important to incorporate the fault tolerance into the GDC's environment. The process of fault tolerance shown in Figure 2 is described as follows:

When a processor P_i finishes its job and is idle. Then P_i will visit P_{i+1} to ask for a new scanline to render. There are two cases to consider. One is when P_{i+1} is busy that can not response to the request of P_i. The other is when P_{i+1} is crashed.
In order to distinguish whether the processor P_{i+1} is crashed or busy, we must set the timeout. If P_{i+1} does not response the request more than three times, we can regard that P_{i+1} is crashed .
If P_{i+1} is crashed, P_i will send a message to the master and ask master for checking the global scanline table. Then master will notify P_i to take over the incomplete tasks that were owned by the process P_{i+1}. The information include how many scanlines P_{i+1} has not finished and the processor identifier of P_{i+1}'s next processor. At present, P_i can replace P_{i+1} to connect with the P_{i+1}'s next processor. So it looks like a satellite to protect the planet when the planet cannot normally operate.

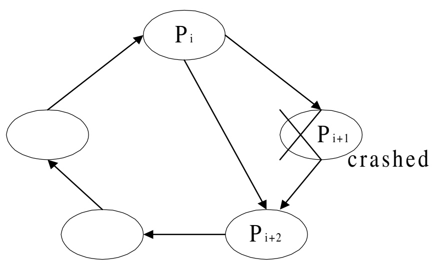

Figure 2. The fault tolerance mechanism of GDC

Table 2. The performance of fault tolerance after deleting two Sparc 20 from PVM's console

	Sparc 2	Sparc 10	Sparc 20	Total relative power	Performance
No of workstations	**3**	**2**	**4**	**-**	**-**
Relative power	**0.77**	**1**	**1.96**	**-**	**-**
Before deleting hosts	**3*0.77**	**2*1**	**4*1.96**	**12.15**	**14 secs**
After deleting hosts	**3*0.77**	**2*1**	**2*1.96**	**8.23**	**19 secs**
Degradation	**0**	**0**	**3.92**	**3.93**	**42.85 %**

For this fault tolerance scheme, the crashed process can be successfully replaced by the preceding processor without paying too much overhead. The other processors can continue their jobs without spending the extra time to modify their environment's parameters. To test fault tolerance ability, we conducted an experiment performing ray tracing with 9 workstations using PVM. After the master process gathered 300 scanlines from the slave processes, two workstations (i.e. two Sparc 20) are deleted from PVM console. In the master process, we adopted the blocking receiving within timeout. So when the given hosts were deleted, the master process can still receive the rendering result from other slave processes. The ball image is used as the testing scene. The results in Table 2 show that the rendering process is still running after two workstations are deleted, thus the fault tolerance ability is demonstrated.

SYSTEM DESCRIPTION

The system consists of four parts: namely, the collaborative environment, the PVM console, the Rendering console, and the Display console. We will briefly describe each part in some details :

The interface of collaborative environment as shown in Figure 3 provides a login circumstance for users to create a new group or join an existing group. When a user participates in a group, the interface agent will display the group information in the applet (GUI frontend). In addition, users can use the "talk" or "white board" media to communicate with other participants in this group. We adapt both the broadcasting and single user message passing methods to talk with other users. Two types of communication are provided, namely, private and public modes.

The PVM console allows users to add or delete a set of hosts. The user can interactively configure his/her virtual machine according to their computing requirement. After configuring their parallel virtual machine, the user can proceed to the "Rendering" console for ray tracing images. The rendering console provides options to dynamically control the process of raytracing, Filename, Camera position, Light source, Background, Rotation, Zoom, Render, Record, and Help.

Users can display a static rendered image in the browser or an animation sequences in an internal viewer. The interface agent will pop a window as an internal viewer to display the contiguous images. With regard to the image compression format, a static rendered image is compressed in a JPEG format. Due to the limitation of the network bandwidth and the time-consuming computation in raytracing, after finishing a rendered image, token agent will send a signal to announce interface agent to display this rendered image. So user can view the animation sequences frame by frame before finishing the rendering of an animation sequences. After finishing all frames' rendering, MPEG format is adapted to compress the animation sequences and interface agent will display this MPEG file in the internal viewer. A sample of display is shown in Figure 4.

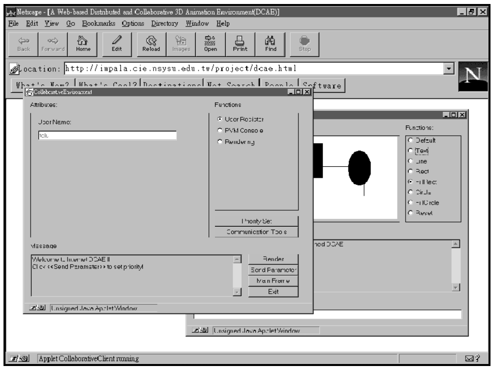

Figure 3. The user interface of collaborative environment

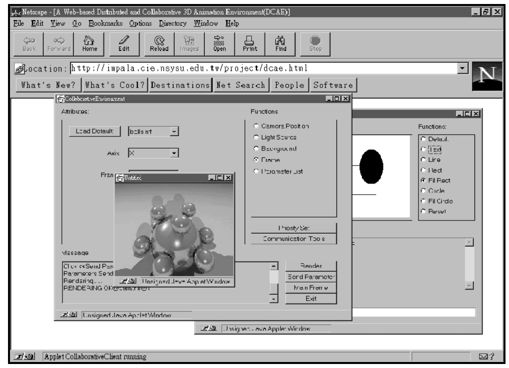

Figure 4. The animation display screen

CONCLUSIONS

In this paper, we have described a distributed and collaborative system using the Web browser, PVM, agents, Java and Java socket classes. It provides a collaborative and distributed computing platform for users to design, discuss, compute, and visualize the 3D animation over the Web. We have proposed a simplified collaborative group definition, collaborative policies, and the state of participants to dynamically manage participants for the collaborative model. Based on the proposed mechanisms, the server can efficiently determine the status of collaboration activities.

REFERENCES

1. J. Gosling, F. Yelin, and the Java Team. *The Java Application Programming Interface.* Addison-Wesley Developer Press, Sunsoft Java Series, 1996.
2. Bryan Carpenter, Yuh-Jye Chang, Geoffrey Fox, Donald, and Xiaoming Li, "Experiments with HPJava," Java for Science and Engineering Computation, in http://www.npac.syr.edu/projects/javaforcse/javameettalks.html.
3. V. S. Sunderam, "PVM: A Framework for Parallel Distributed Computing," Concurrency: Practice and Experience, Vol. 2 No. 4, pp. 315-339, Dec. 1990.
4. Eve M. Schooler, "Conferencing and collaborative computing," Multimedia Systems, pp. 210-225, 1996.
5. Taizo Miyachi and Norio Shiratori, "A Multi-Agent Collaboration System in Planning with Reduced Coordination Cost," in: Proc. The 11th International Conference on Information Networking, Vol. 1, 3D-1.1~3D-1.8, 1997.
6. T.Y. Lee, C.S. Raghavendra and J.B. Nicholas, "Experimental Evaluation of Load Balancing Strategies for Ray Tracing on Parallel Processor," to appear in: Integrated Computer-aided Engineering Journal 4 (1997).
7. Chungnan Lee, Tong-yee Lee, Tainchi Lu, and Yao-tsung Chen, "A World-Wide Web Based Distributed Animation Environment," the Journal of Computer Networks and ISDN Systems, a special issue on Visualization and Graphics on the WWW, 1997.
8. E. Haines, "A Proposal for Standard Graphics Environment," IEEE Computer Graphics and Applications, 7(11):3-5, Nov 1987.