協同分散式之視覺化環境應用在三維繪圖場景
A Distributed and Collaborative Visual Environment for Rendering 3D Model Scenes

盧天麒, 江傳文, 李宗南, 林明堂

國立中山大學資訊工程研究所
Institute of Computer and Information Engineering
National Sun Yat-Sen University, Kaohsiung, Taiwan
ROC
cnlee@mail.nsysu.edu.tw

李同益

國立成功大學資訊工程系
Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, ROC
tonylee@mail.ncku.edu.tw

摘要

本文中詳細的描述了整個系統架構之設計與實現，伺服器端的協同控制和共享資料機制，及分散式異質計算環境平台, 藉由本系統之建制，可進一步實現虛擬實境在全球資訊網上的應用範疇。

關鍵字 :爪哇語言, 協同性, 分散式計算

Abstract

In this paper, we discuss underlying design concept, issues and the current applications. We use performance indices to measure parallel performance of a program and present some experimental results.

Keyword: Java, Collaborative, Distributed Computing

1. Introduction

Today, many grand-challenge problems have being studied by the scientists in the world. These researches require the use of large amount CPU time as well as depend heavily on collaboration that combines expertise from different sub-discipline at multiple, geographically distributed locations. In such circumstance [1], it is very essential to develop a proper working infrastructure that supports the distributed computing as well as group collaboration [2]. The graphics visualization has been exploited to facilitate knowledge discovery in science. However, there is little work to address the need of such integrated infrastructure offering distributed computing and collaborative facilities. Visualization is a very computationally intensive and is a kind of artwork requiring multiple-user efforts. We sense these requirements and have been involved in design of a distributed and collaborative infrastructure since 1995. For distributed computing requirement, it must provide sophisticated mechanisms to efficiently manage all resource, to intelligently schedule computing-power, and to robustly tolerate unexpected faults in the system. In our system, we exploit PVM [3] technology to develop our distributed applications, and also devise an efficient scheduling and a solid fault-tolerant scheme. For collaborative functionality, our system satisfies computer supported cooperative work (CSCW) requirements - sharing an information space to design an effective technical system [4]. With such a collaborative capability, people can work together and perform common tasks in our environment. A framework with collaboration is believed to be a better scenario to solve the problem with common interest.

The access of WWW network resources and use of WWW browser to obtain information has grown tremendously in the recent years. Coupled with the advent of Java, WWW becomes an active document capable of securely executing code distributed to the clients. Therefore, the Web-Java has been recognized as the standard user interface for applications [5]. With the perception of quickly immense acceptance on Web-Java technology, the DCVE system is now a set of Web-Java servers and applets which allow dynamic collaborative environments, and a set of non-Web applications which are easily integrated into our system via the Web-Java interface.

In Section 2, we first overview the related work, and then present an overview of our DCVE system architecture namely: collaborative environment, PVM distributed clustering environment, server fault tolerance, graphic editor and collaborative toolkit. The distributed graphics server performance is analyzed and discussed in Section 3. A short description of system functionality of the DCVE is given in Section 4. Finally, concluding remarks are drawn in Section 5.

2. DCVE Architecture

2.1 Previous Work

There are multiple ongoing projects in the web to develop the collaborative environment. TANGO [6] is a Java-based collaborative system for the World Wide Web and is aimed at creation of shareable information

spaces. The project PageSpace [7] is targeted at supporting networked applications which require interaction between distributed software components and active processing. It is based on the Internet and the World Wide Web but introduces a notion of active Web-pages that are capable of executing code. Juice [8] is a new technology for distributing executable content across the World Wide Web. Juice differs from Java in several important aspects that allow it to outperform Java in many "downloadable Applets" applications. Project SHASTRA [9] is directed at research in CSCW based geometric modeling, simulation, interrogative visualization and design prototyping environments.

2.2 Collaborative Environment

Collaboration 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. The infrastructure must provide flexible collaboration control methods to initiate and terminate collaborative sessions, to join or leave ongoing sessions, and to invite a new participant in a collaborative environment.

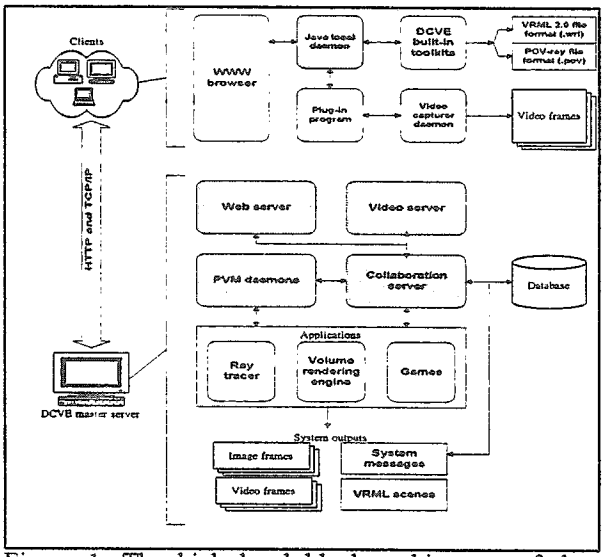

Figure 1. The high level block architecture of the DCVE system

In 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 distributed ray tracer 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.

In the following we define collaboration policies and mechanisms through some operators operating on the set and its elements. A group $G = \{x_i \mid x_i \in G, i = 1,2,3,...,n\}$ 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 Table 1.

Table 1. The collaborative operators of the DCVE server mechanism

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 $G_i \times x \rightarrow$ 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.

2.3 Distributed Environment

Following the success of Oak Ridge National Laboratory's Parallel Virtual Machine (PVM), it allows an interconnected collection of independent heterogeneous computers to appear as a single virtual computational resource. The PVM platform provided a standard interface to support common parallel-processing paradigms, such as message passing and shared memory. However, PVM does not provide capability to achieve good load balancing and tolerate occasional faults. These two issues are very essential in distributed environment to ensure high utilization of computing resource and functional correctness of the applications. In DCVE, we design an efficient load balancing scheme as well as fault-tolerance scheme. We have presented a new dynamic load balancing strategy, called global distributed control. The global distributed control not only has the ability to dynamically adjust the load, but also has the fault tolerance ability. The detailed description of GDC algorithm is in [10].

Conventionally, to exploit PVM technology, users have to use command-driven PVM console to add or delete computers in the parallel virtual machine. It is neither efficient nor user friendly. Alternatively, users can use an XPVM built on the X-windows system to manipulate the PVM system. However, it is limited to some machine architectures, like SUN SPARC, HP, and DEC workstations. As a result, it is difficult to build a portable distributed environment. This motivates us to establish a portable user interface, and thus for this purpose, we use Web-Java technology to design the GUI for PVM.

2.4 Collaborative Toolkit

The objective of the collaborative toolkit is to improve A\V communication among clients in the DCVE. The system provides the following media types.

- **Whiteboard**

It is used as a sketching tool, and facilitates to the generation and display of simple 2D sketches. Drawing can be moved to and from different canvases using interface facilities. Each participant may mark up the whiteboard by using simple drawing tools, or may enter text.

- **Chatting Room**

It is used as a text notepad. A collaborative session consists of text chatting and allows a group of collaborators synchronously communicate using text. We adapt the broadcasting and single user message passing method to talk with other users.

- **Video Conferencing**

A plug-in Video component provides frame grabbing, video display, and the use of various video formats in DCVE. The video frames are transmitted and exchanged among collaborative participants to allow a simple video conferencing.

3. Performance Metric and Analysis

Performance indices [11] are used to evaluate the characterization of a program quantitatively and to locate potential bottlenecks in parallel computation. Moreover, it can help users to identify pitfalls of their parallel program in advance. The traditional analysis of parallel program performance tends to use statistical inference to trace the performance projection. They make a conclusion by analyzing a vast of information. In this section, we describe the performance indices and inspect these indices from which it may benefit program performance. We use performance indices to measure the performance of a parallel ray-tracing program.

Workstations may have different configuration in terms of speeds of CPU, memory accesses, and I/O systems. The sequential program doesn't need to be executed under the network, hence the performance of a sequential program is only influenced by two factors - the rendering algorithm and machine's power. Therefore, the elapsed time of a sequential program is easily to identify and estimate. On the contrary, the parallel program needs to consider the network's bandwidth and the work load in the distributed computing environment.

The execution time of a sequential program is defined as T_{seq}, which is different for each machine. The more power machine will gain a smaller T_{seq}. The elapsed time of a parallel program is defined as $T_{parallel}$. When we execute the same program at several machines at the same time, the elapsed time of the program will not be the same. $T_{parallel}$ depends on the current state of the cluster computing environment. We define $T_{parallel}$ as follows :

$$T_{parallel} = \max_{i=1\sim n} (T_{host\_i}) \qquad (3.1)$$

T_{host_i} is the execution time of node i in a distributed environment. The more hosts we add in the parallel virtual machine, the smaller T_{host_i}, due to the less work load for each host. Therefore, the total $T_{parallel}$ is to find a maximum value of T_{host_i}.

In the master-slave model, a master is responsible for spawning, initialization, collection and display of results. The master does not response for the computation. In the following, we define equations for the master process.

$$T_{master} = T_{comm} + T_{data} + T_{pvm} \qquad (3.2)$$

where T_{master} is the total execution time of the master process; T_{comm} is the communication time between the mater process and slave processes; T_{data} is the time to pack and unpack the active messages in the

active buffer; T_{pvm} is the degradation time caused by the PVM daemon. The communication time T_{comm} can be further decomposed into two components – send and receive blocking. That is

$$T_{comm} = T_{send\_slave} + T_{receive\_slave}(r) \quad (3.3)$$

T_{send_slave} includes the time to send data to the active message buffer in the master and to the slave. T_{send_slave} is the send blocking time. PVM uses several methods to receive messages in a task. In a blocking receive mechanism, a master has to wait for messages from slave processes. On the contrary, in a nonblocking mechanism, a master does not have to wait for the message. The variable r is used to represent the receive routine, it can be blocking, nonblocking, and timeout receive routine. For example, $T_{receive\_slave}$ (blocking) represents the elapsed time that uses the blocking receive routine to receive the messages from the slave processes.

T_{data} is the time to pack and unpack the active message buffer with arrays of data type. T_{data} can be further decomposed into pack and unpack routines.

$$T_{data} = T_{pack} + T_{unpack} \quad (3.4)$$

The slave programs perform the actual computation. Their work load are assigned by the master. The equations from the slave processes are defined as follows:

$$T_{slave\_i} = T_{comm\_i} + T_{comp\_i} + T_{data\_i} + T_{pvm} \quad (3.5)$$

where T_{slave_i} is the total elapsed time of the i-th slave; T_{comm_i} is the i-th slave's communication time; T_{comp_i} is the i-th slave's computation time; T_{data_i} is the time to pack and unpack the active messages of the i-th slave. T_{pvm} is the degradation time caused by the PVM .

In order to derive the correct speedup from the $T_{parallel}$ and T_{seq}. We define a relative computing power for each machine. Generally, we select a machine to be the based computing power. Then, we can derive other machines' relative computing power by comparing the based computing power. The total computing power, $p_{machine}$ is defined as follows :

$$p_{machine} = \sum_{i=1}^{n} p_{host\_i} \quad (3.6)$$

We use the parallel ray tracing program [12] to illustrate the use of the indices. The experimental environment is a non-dedicated heterogeneous environment, which consists of four types of Sun workstations: SPARC 2, SPARC 10, SPARC 20, and ULTRA SPARC 1. The public domain PVM is used as the parallel computing platform. Based on the performance metrics proposed in the previous section, we measure the total elapsed time, communication time, message manipulation time, and PVM degradation time from the master process. The unit of the elapsed time (Z axis) is in microsecond. We compared the execution time for computation and communication in Figure 2. As we can see the computation time is much larger than the communication. Hence the problem is good for parallel computing.

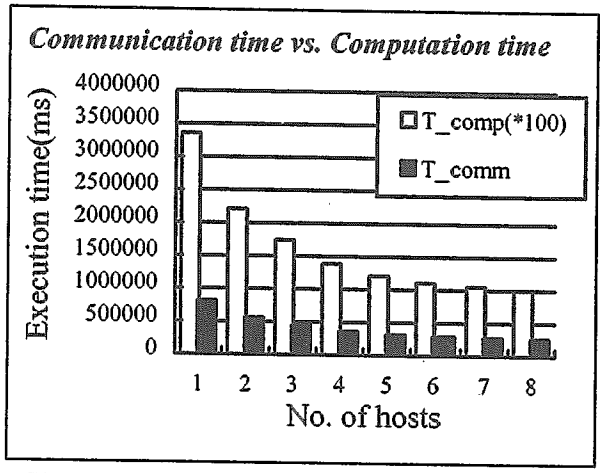

Figure 2. The comparison with the communication time and computation time of the slave process. The execution time for T_{comp} is scaled down 100 times in the figure.

Due to limitations of memory size, cache size, and other hardware components, the power weights usually change with different workstation types. Unfortunately, the relative computing power among a set of heterogeneous work varies with applications. Hence, one workstation may run an application program faster than another workstation, but slower for another application program.

Table 2. The relative power of different machine platform

Machine platform	p_{HP7} (HP 715/33)	p_{sparc} (Sparc 2)	p_{sparc1} (Sparc 10)	p_{sparc} (Sparc 20)	p_{ultra1} (Ultra Sparc 1)	p_{mach}
Relative computing power	1	0.442	0.929	1.347	2.586	9.369
No. of hosts		2	2	3	1	8

The computing power of each machine might vary significantly in the heterogeneous environment. Using the speed of the dedicated workstation, we can calculate the speeds of the other workstations in the clustering environment. In order to quantify the

relative computing power among the workstations in a heterogeneous network system, for each workstation, a computing power weight with respect to ray-tracing is delineated in the Table 2.

4. System Overview

The system integrates Java, 3D computer graphics, parallel computing and WWW technologies to provide a 3D distributed and collaborative environment [13]. With the PVM software support, the computationally intensive graphics application can be executed on a set of machines in the networked environment. The WWW technology allows our system to be accessed via the Web from different platforms. The system consists of five parts: namely, the collaborative environment, the PVM console, the rendering console, collaborative toolkit and the display console. We will briefly describe each part in some details.

The interface of collaborative environment provides a login subwindow for users to create a new group or join an existing group. A new user can register into our system. For example, to have a new nickname, password, and personal information. In Figure 3, when a user participates in a group, the interface agent will display the group information in the applet (GUI frontend), including group name, group manager name, group topic, all participants' name and the current group messages. Therefore, users can type their user id and password to enter the system.

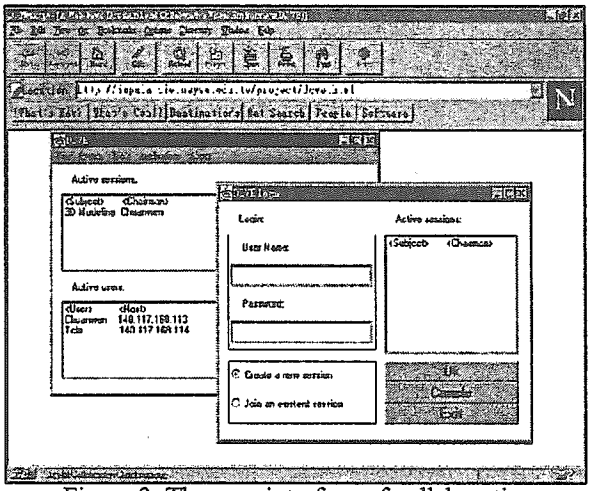

Figure 3. The user interface of collaborative environment entrance

The PVM console provides a GUI front-end for users to add or delete a set of hosts in an interactive manner. Before adding hosts to the virtual parallel machine, a user can click the "Initial'' button to list all available computing resources in the environment. In addition, we provide an extra subwindow to show the execution status of the PVM system.

After configuring their parallel virtual machine, the user can proceed to the "Rendering" console for ray tracing images. In Figure 4, 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 message to signal interface agent to display the rendered image. So user can view the animation sequences frame by frame before finishing the rendering of an animation sequences.

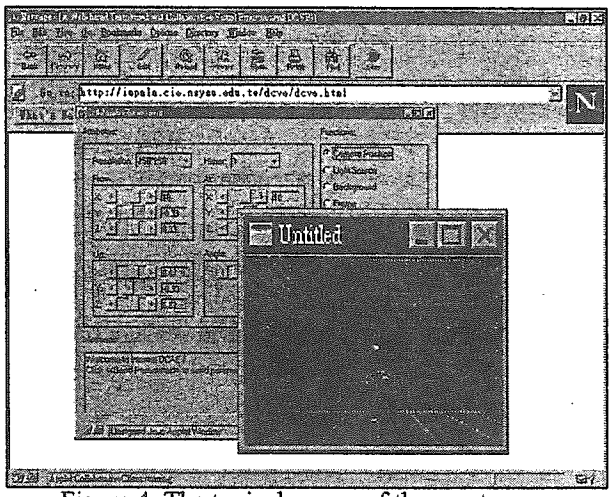

Figure 4. The typical screen of the ray-tracer

5. Conclusion

In this paper, we have described a distributed and collaborative system by 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 agents dynamically manage participants for the collaborative model. Based on the proposed mechanisms, the server can efficiently determine the status of collaboration activities. The system architecture and some applications are described. Some performance metric and analysis are proposed to analyze the potential performance of a program to be parallelized.

Acknowledgment

This research was supported in part by the National

Science Council of Taiwan, R.O.C., under contracts NSC-87-2213-E-110-013 and NSC-87-2213-E-006-012.

References

1. V. Anupam et al., "Scientific Problem Solving in a Distributed and Collaborative Environment," Journal of Mathematics and Computers in Simulation, 36, 1994, 433-542.
2. A. Mani Chandy et al., "A World Wide Distributed System Using Java and Internet," 1996.
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. V. Anupam and C. Bajaj, "Shastra : An Architecture for Development of Collaborative Application for Development of Collaborative Applications," International Journal of Intelligent and Cooperative Information systems(IJICIS), 3, 2, 1994, 155-166.
5. J. Gosling, F. Yelin, and the Java Team. *The Java Application Programming Interface.* Addison-Wesley Developer Press, Sunsoft Java Series, 1996.
6. L. Beca, et al. "Tango - a Collaborative Environment for the World-Wide Web," in http://trurl.npac.syr.edu/tango/papers/tangowp.html.
7. P. Ciancarini, A. Knoche, R. Tolksdorf, and F. Vitali, "PageSpace : An Architecture to coordinate Distributed Applications on the Web," 5th International World Wide Web Conference, May 6-10, 1996, Paris, France.
8. M. Franz and T. Kistler, "Introducing Juice," University of California, in http://www.ics.uci.edu/~juice/.
9. V. Anupam and C. Bajaj, "Shastra : An Architecture for Development of Collaborative Application for Development of Collaborative Applications," International Journal of Intelligent and Cooperative Information systems(IJICIS), 3, 2, 1994, 155-166.
10. 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).
11. G. Casciola, S. Morigi, "Graphics in parallel computing for rendering 3D modelled scenes," Parallel Computing, Vol. 21, pp. 1365-1382, 1995.
12. C.N. Lee, T.Y. Lee and T.C. Lu, "Comparisons of Load Balancing Strategies for Ray Tracing on Network," In: Proc. International Computer Symposium (ICS'96), Taiwan, R.O.C., 1996.
13. C.N. Lee, T.Y. Lee, T.C. Lu, and Y.T. 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.