Design and Implementation of a Scalable Collaborative System for Industrial Design

Jinmo Yang, Sungyoung Lee and Byeong-Soo Jeong

School of Information and Electronics , Kyung Hee University

449-701 Seo Cheon Ri 1, Ki Heung Eup, Young In Si, Kyung Ki Do, Korea

(isoli@oslab.kyunghee.ac.kr, sylee@oslab.kyunghee.ac.kr, jeong@nms.kyunghee.ac.kr)

Abstract

This paper describes our experience with the design and implementation of a collaborative system framework that aims at developing specific applications such as 3D animation, computer game, and industrial design. The proposed collaborative system enables users, who may be far removed from each other geographically, to do collaborative work in a single virtual space. The system basically adopts client/server architecture, in order to develop a platform independent, scalable and easily portable collaborative system framework. In order to achieve this goal, the server was implemented on Java platform and it adopts a hybrid architecture in order to take advantage of both centralized and distributed architecture. Thus, the system is functionally divided into several modules, such as User Manager Server (UMS), Session Manager Server (SMS), and Information Server (IS). The role of UMS is to manage the information about users who participate in collaborative work, SMS is to support the conferences in the system, and IS is to provide the means of connection among the UMSs. For the user's convenience, the client was implemented on Visual C++ in Windows environments. It also supports several collaborative tools, such as 3D Studio Max which is extended to work for distributed environments, chatting, and white board features.

1. Introduction

With the improvement of Internet, multimedia, virtual reality, and high speed processor technology, systems which can provide a collaborative working environment in virtual shared space are being made available. Collaborative systems have the potential to improve productivity where groups of users work together, such as industrial design, 3D animation, team diagnosis, and teleconferencing. However, since in such applications, several groups of geographically dispersed people may jointly work on multiple shared data artifacts, building collaborative applications is fraught with difficulties. The collaborative system developers should be concerned with technical issues such as synchronization, concurrency, communications, registration and more [11].

Even though each collaborative application reveals its own specific requirements, most collaborative applications commonly require several functionalities [5]. First, for better coordination among users, there should be mechanisms so that users are generally aware of what other participants are doing. Second, a concurrency control mechanism should be provided for keeping the shared data consistent even though users may attempt to make simultaneous, conflicting changes. Third, because of the interactive nature of a collaborative application, it should ensure interactive response time to users' actions while maintaining shared data consistency. Fourth, since the number of users in a collaborative work environment tends to increase, it should provide scalable server performance without regard to the number of clients. Last, it should provide an efficient mechanism to store shared data persistently since collaborative work tends to be accomplished over a long period.

Collaborative system architectures that are currently being implemented are generally classified into the three categories of centralized, distributed, and hybrid [3]. Centralized architecture gives the advantage that shared data can be easily maintained and overall

* This work was supported by Core S/W Technology Development Program (#NS-01-08-A-24) sponsored by Ministry of Science and Technology in Korea.

implementation is somewhat easy compared to other approaches. But there is the possibility of server bottleneck. Again, distributed architecture can eliminate sever bottleneck by distributing workloads, but maintaining shared data consistency in distributed architecture is very difficult. The hybrid approach intends to take advantage of both approaches. It replicates shared data objects in distributed architecture, but meta data information about shared data objects is managed in a centralized manner.

In this paper, we present our experience of designing and implementing a collaborative system intended to support industrial design (including 3D design) processes. Our design goal is to provide platform-independent, highly scalable, and easily portable framework for collaborative applications. Our system adopts a hybrid model that is a compromise between centralized and distributed alternatives. Overall system implementation represents client-server architecture and the server part is functionally divided into several modules in order to extend easily with workload increase. The server part consists of three modules: the User Manager Server is responsible for preprocessing of collaborative work, such as user connections, message communication, and event management; the Session Manager Server deals with project schedule management and session management; and the Information Manager Server provides information exchange between User Manager Servers. The client part mainly handles user interaction and is implemented by Visual C++ in Windows environments while focusing on user friendliness. As collaborative application tool, we implement chatting and white boards. We also extend 3D Studio Max to distributed environments by using a plug-in mechanism.

The rest of the paper is organized as follows. In section 2, we briefly summarize previous work on collaborative system implementation. Section 3 and 4 describe the implementation details of our system, Collaborative System Server and Client parts respectively. Section 5 introduces experimental results to evaluate server performance. Finally, section 6 presents some concluding remarks and directions for future work.

2. Related Work

Collaborative applications can be classified according to the patterns of collaboration and the degree of geographical distribution between collaborating users as in Figure 1. First of all, by way of interaction in collaborative groups, systems are divided into *Synchronous* and *Asynchronous*. Synchronous collaborative applications, such as teleconferencing, require that each participant's activity should be synchronized in a real time domain. Asynchronous collaborative applications, such as E-mail, can be accomplished without synchronization. According to the geographical dispersion of collaborative users, collaborative systems are classified into *Co-located* and *Remote*.

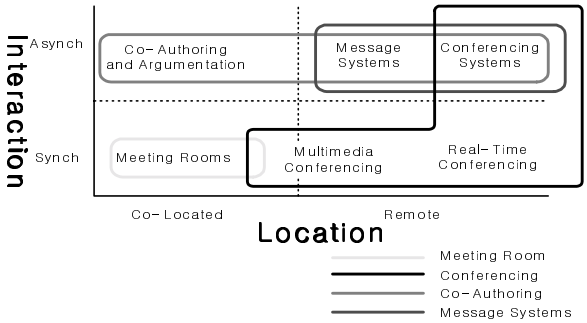

Figure 1. Classification of Collaborative Systems

Many experimental implementations for collaborative applications have been done during the last few years. Some representative examples are Habanero [3], GroupKit [12], and EGRET [7] which have influenced our system design to some degree. Habenero is a collaborative system framework implemented by Java. It provides an environment where Java objects can be shared through the Internet and also provides API for collaborative application development. By using Java, Habanero makes it possible to develop platform independent collaborative applications easily. However, it has the deficiency that existing PC application software cannot be easily integrated.

GroupKit is a groupware toolkit for building real-time conferencing applications developed at the U. of Calgary. Its run time architecture reveals replicated forms across a number of machines as in Figure 2. The session manager and conference process are replicated processes, one per each participant. The session manager provides both a user interface and a policy dictating how

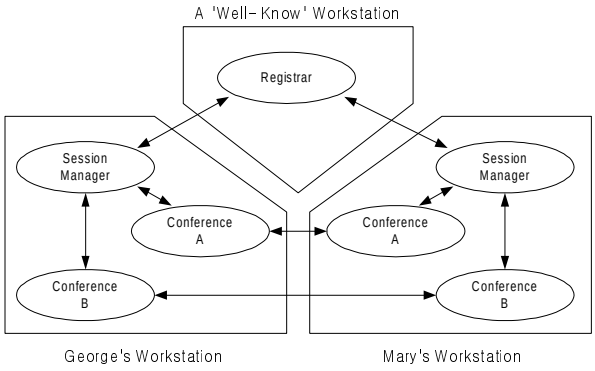

Figure 2. GroupKit's Run Time Architecture

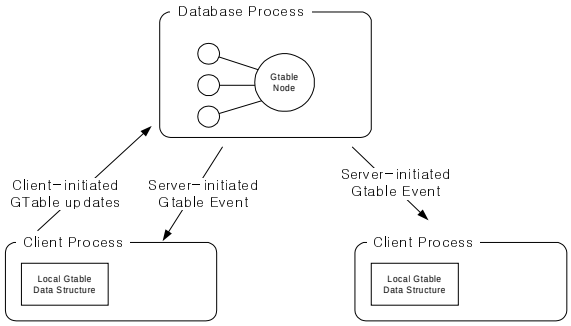

Figure 3. EGRET's Gtable Architecture

conferences are created or deleted.

The registrar is a centralized process that acts as a connection point for a community of conference users. It provides the current state information for conferences to collaborating users by maintaining a list of all conferences and their users. The major contribution of GroupKit implementation is that it formalizes the design and implementation of general groupware conferencing toolkits and provides a basis for generalizing existing application or toolkit features.

EGRET is a collaborative system framework for exploratory collaboration that was developed at the U. of Hawaii. In order to provide exploratory services in a responsive interactive environment, EGRET implements a dynamically extendable framework that consists of Multi-Server, Multi-Client, and Multi-Agent. As for database management, it uses client-server database architecture. To reduce server workloads, it tries to distribute data processing workloads to clients if possible. As for shared data management, it uses a hypertext mechanism where data is accessed by predefined links and also it defines the Gtable structure in order to efficiently process hypertext data.

3. Collaborative System Server

Generally, collaborative systems are comprised of two parts, a server part that deals with session management for collaboration and a client part that actually performs collaboration work by connecting the server. Our collaborative system is also implemented by separating it into two parts. In this section, we describe the implementation details of the server part.

3.1 Server Model

As previously mentioned, our collaborative system consists of server part and client part. Our collaborating server is functionally divided into independent sub servers, such as Information Server (IS), User Manager Server (UMS), and Session Manager Server (SMS). Such server architecture aims for scalable system architecture.

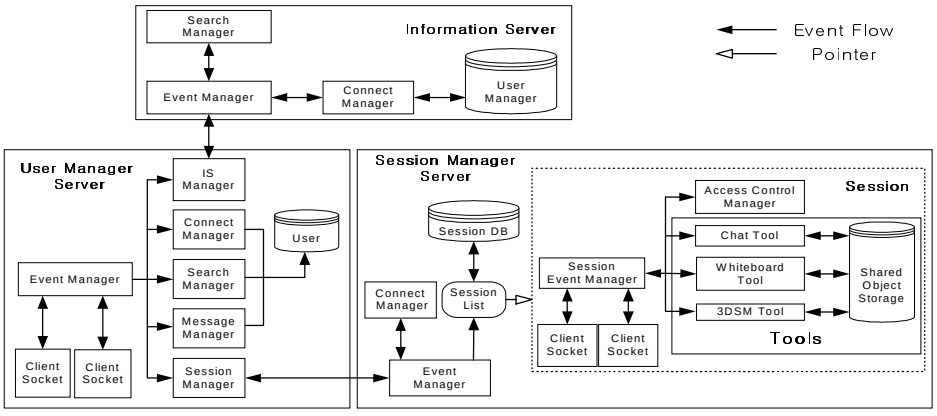

Figure 4. Collaborative System Server Architecture

Figure 4 shows this server architecture. UMS provides several services that are needed before performing actual collaboration work, such as user connection, processing requests for session creation, and message exchange. SMS deals with managing collaborative work schedules and sessions that are currently activating. It also provides event handling services that are needed for shared date management. IS is provided for system scalability. As system load increases, UMS and SMS can be multiply configured as in Figure 5. IS supports information exchange between UMSs.

As you may see in Figure 4, UMS handles all kinds of communication between server part and client part. After receiving service requests from users, it checks the user's authentication and provides a session by contacting SMS. SMS not only provides session service to users via UMS but also controls consistency of shared data by using an optimistic locking method. For system scalability, each server module can be replicated and run on different machines as in Figure 5. Server modules are implemented by using platform independent Java (JDK 1.2.1) and carefully packaged for easy maintenance.

The infrastructure of collaborative server part is comprised of a communication module and event handling module. A communication module supports interconnection between server part and client part by way of an event processing module. Service requests from each client are transferred by the form of event. Such events are passed and processed on each event handling routine according to event type. Communication modules are implemented by using TCP sockets and Java RMI. For extendibility and processing efficiency, we have devised our own event management structure and define a packet type that will be explained in the next section.

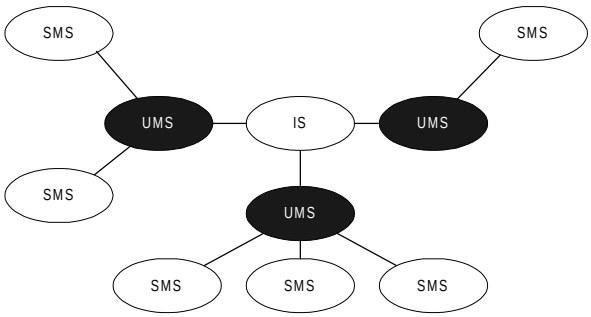

Figure 5. Extended Collaborative System Architecture

3.2 Event Processing Model

Event processing modules are responsible for handling all kinds of events that occur from a client part or within a server part. An event processing structure is designed for effectively supporting the interactive response requirements of collaborative systems. For fast response, non-conflicting events are processed in parallel. This also makes it possible for newly defined events to be easily added for other collaborative applications. Such an event processing structure is a main body of our collaborative system and commonly used over UMS, SMS, and IS. An event processing structure consists of CEvent Class that defines event type, Event Thread Class that actually processes events, and Event Monitor that implements a priority based event queue.

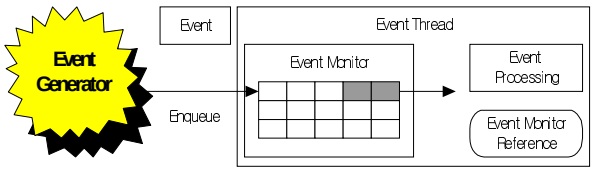

Figure 6. Event Processing Model

In order to reduce resource usage, Event Thread enters a sleep state when the event queue (Event Monitor) is empty. For the purpose of processing multiple Event Threads simultaneously, Event Generator can be an Event Thread. That is, Event Thread can propagate events to other Event Threads by Event Monitor Reference. As in Figure 7, one Event Thread can refer to multiple Event Monitors and provides an event dispatch function to Event Monitors of Event Threads that process different events so that non-conflicting multiple events can be executed in parallel.

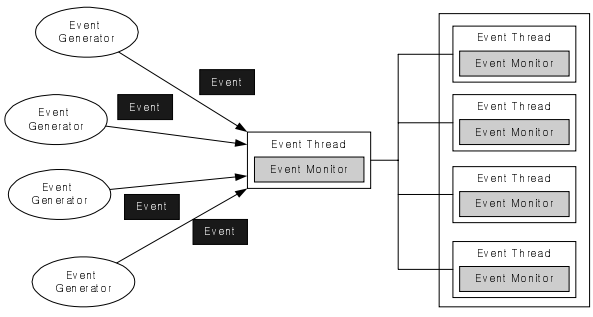

Figure 7. Extended Event Processing Model

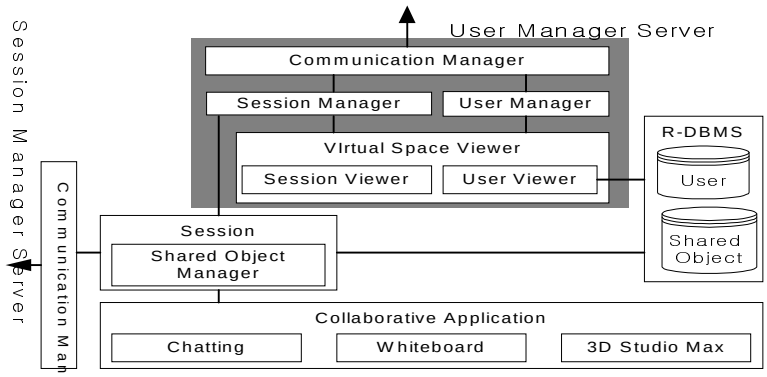

Figure 8. Collaborative System Client

4. Collaborative System Client

While contacting the server for project schedule information and current status of collaborative work, the client system supports collaborative design processes where multiple participants are working together. For this purpose, client system is divided into two stages, preparing stage modules for collaborative work and collaborating stage modules as in Figure 8. In the preparing stage, client modules provide current status information of collaborative work to users by communicating with USM in the server part. Such information is transmitted through the Communication Manager, processed by the Session Manager and User Manager, and finally shown to users via user interface. The User Manager provides information about the connection status of other users and handles message transfer services between collaborating users. The Session Manager provides the information about current collaborating work status and handles session services, such as session creation and session join. The Virtual Space Viewer is a user interface module that outputs user information and session information.

The Collaborating stage covers the performance of actual collaborating work after a joining session by connecting with SMS while updating sharing data objects with other collaborating users. For joining session, user authentication is needed from the User Manager Server and a session module handles session join after receiving the required information from SMS. Collaborative Application modules provide user interface for collaborative application programs. Current applications include Chatting, Whiteboard, and 3D Studio Max.

5. Conclusion

This paper describes our experience with the design and implementation of a scalable collaborative system framework that aims at developing collaborative applications for industrial design. The overall system architecture adopts basically client/server architecture and the server part is functionally divided into several modules, such as UMS, SMS, IS, and communication modules for the purpose of extendibility and scalability. We evaluate server performance by experimenting with a collaborating work environment while varying system workloads, including the number of users and the event frequency. Experimental results show that our multiple sever approach can be well scaled with the increase of system workloads and easily extended to integrate other collaborative applications.

For future work, we are planning to further investigate issues concerning data consistency and system performance. As for data consistency in shared data management, we need to consider different levels of granularity depending on application characteristics. As for system performance, we need to devise a mechanism so that system workloads can be well balanced among multiple servers by monitoring their workloads.

Reference

[1] S. Bhola, G. Banavar, and M. Ahamad, “Responsiveness and consistency tradeoffs in interactive groupware”, In *Proceedings of 7th ACM Conference on Computer Supported Cooperative*

Work, November 1998.

[2] S. Bhola, B. Mukherjee, S. Doddapaneni, and M. Ahamad, “Flexible Batching and Consistency Mechanisms Building Interactive Groupware Applications”, In *Proceedings of the 18th ICDCS*, 1998.

[3] Annie Chabert, Ed Grossman Larry Jackson, and Stephen Pietrovicz, “NCSA Habanero: Synchronous Collaborative Framework and Environment”, 1998

[4] P. Dewan and R. Choudhary, “A Flexible and High-Level Framework for Implementing Multi-User User Interfaces”, In *ACM Transactions on Information Systems*, vol. 10, no. 4, pp. 345-380, Oct. 1992

[5] S. Greenberg and D. Marwood, “Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface”, In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 207-217, Chapel Hill, North Carolina, October 22-26, 1994, ACM Press.

[6] J.H. Lee, A. Prakash, T. Jaeger, and G. Wu, “Supporting multi-user, multi-applet workspaces in CBE”, In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'96)*, pp. 344-353, 1996

[7] Philip M. Johnson, “Experiences with EGRET: An exploratory group work environment. Collaborative Computing”, January 1994

[8] J. Munson and P. Dewan. "A Concurrency control Framework for Collaborative System", In *Proceedings of the 6th ACM Conference on Computer Supported Cooperative Work*, 1996.

[9] M. T. Ozsu and P. Valduriez, “Principles of Distributed Database Systems”, *Prentice-Hall*, pp. 327-329, 1998.

[10] A. Prakash and H.S. Shim, “DistView: Support for Building Efficient Collaborative Applications using Replicated Objects”, In *Proceedings of CSCW '94*, ACM Press, New York, pp. 153-64, 1994.

[11] Tom Rodden, “A Survey of CSCW Systems”, *Interacting with computers*, vol. 3 no. 3, pp. 319-352, 1991.

[12] M. Roseman and S. Greenberg, “Building Real Time Groupware with GroupKit: A Groupware Toolkit”, In *ACM Transaction On Computer Human Interaction*, vol. 3, no. 1, March 1996

[13] G. Smith and T. Rodden, “SOL: A Shared Object Toolkit for Cooperating Interfaces”, *Technical Report CSEG/7/1995*, Lancaster University, 1995.