May 23, 2001
RT0415
Computer Science 5 pages

Research Report

Agent Server Technology for Web Applications

Gaku Yamamoto and Hideki Tai

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Agent Server Technology for Web Applications

Gaku Yamamoto and Hideki Tai

yamamoto@jp.ibm.com, hidekit@jp.ibm.com

IBM Research, Tokyo Research Laboratory

Abstract

Web applications providing services customized for individual users have become common. Such applications require very high performance though the available development time is very short. Therefore, we need a very flexible and high performance platform for such customized Web applications. In this paper, we introduce an agent server technology based on an "agent-oriented programming model" for developing Web applications and show how the technology addresses issues in such Web application development.

1 Introduction

Web applications providing services customized for individual users have become common. Some of them provide notification services as well as other customized services. However such Web applications are complex and the development time has become brief. In the Internet world, a system may easily have hundreds of thousands of users. The load on such a system can be high because it manage users' data at the server side. These applications require very high performance. We have developed an agent server platform named "Caribbean" for developing such Web applications [1,2,3]. The agent server is an application server based on an "agent-oriented programming model" [5]. Although most studies on agent-oriented programming models addresses distributed systems, the approach is applicable to developing applications which run on the application server. In the agent server, agents are created at the server side. They are responsible for given roles and for performing their tasks to meet their design objectives. In a typical application, each agent works for an individual user. Since modern servers have several gigabytes of memory, most agents can be kept in memory, and an agent server can achieve very high performance.

We clarify some issues in Web application development in Section 2. A model of an agent server and the required technologies are described briefly in Section 3. Section 4 explains how these technologies address the issues. Related work and our conclusions are discussed in Sections 5 and 6, respectively.

2 Issues in Developing Web Applications

Our target applications are not simple. 1) Our focus is applications that provide services customized for individual users. Therefore, they have to manage users' data. Services for a user usually read and update the user's data. 2) Though the application may be complex, the development time is very short in the Internet world. In a typical case, it is only three months. 3) The application specification is not fixed prior to the start of programming. Therefore, the specification may be modified frequently. 4) These applications are server applications which perform multiple tasks concurrently. The models for these applications must represent concurrent tasks in a natural way. 5) The number of users for Web applications on the Internet can be very large. It may be in the hundreds of thousands. Each task may require reading and updating a user's data, so a high performance platform for Web applications that perform tasks while reading and updating individual user's data is needed.

3 Agent Server

3.1 Model

In our agent server architecture, a user has his own agent at an agent server. An agent is responsible for given roles and performs its tasks to meet its design objectives. In typical applications, an agent is created for an individual user. An agent receives messages from outside programs or other agents in the same server, and it performs tasks using its own data. An agent also uses "services" such as a wrapper for a database. Data kept by each agent must not be lost, so the agents need to be persistent. An agent remains active on a server until it is explicitly removed.

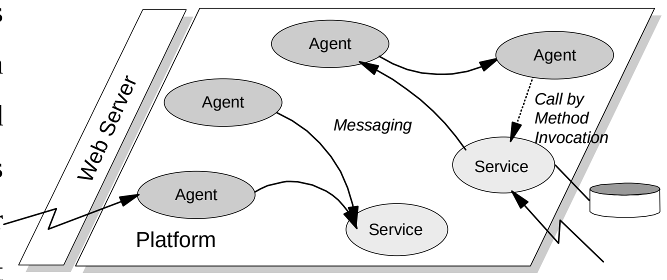

Fig 1 Agent Server

3.2 Runtime Support

Some significant points of an agent server's runtime support are 1) a mechanism for managing very large numbers of agents in spite of physical memory limitations, 2) a mechanism for agent persistence, and 3) a mechanism for scheduling the activities of agents in order to maximize their performance. Basically, agents should be kept in memory, however, sometimes the amount of memory occupied by agents will exceed the available physical memory. Therefore the runtime support has to manage agents beyond the physical memory limitations. In the example of the Caribbean agent server, the Agent Swapping Mechanism swaps agents in and out between memory and disks for limiting the memory usage. Also, a system might fail because of a problem. Even in such cases, the data of the agents must be recovered. The Caribbean runtime support writes a snapshot of the agent onto disk when the agent requires it. Swapping an agent in or out and writing a snapshot of an agent causes disk access. Therefore, the number of disk accesses should be reduced in order to improve the performance. Caribbean's Logger mechanism gathers several snapshot requests and reduces the number of disk access. And Caribbean's Agent Scheduler assigns threads in a thread pool to agents so that the number of swaps is minimized.

4 Solution for the Issues

4.1 Application Development

The Agent-oriented Programming Model is flexible and can represent applications in natural way [5]. We have used the agent server to develop real portal systems providing financial information services. From these experiences, we believe that it is easy for application programmers to understand this model. In the projects, they understood the model very easily and quickly, and they started the rough design of the applications quickly.

While they were developing the applications, the specifications of the applications were modified very frequently. They had to modify the programs many times, but they could do so quickly. We believe that they could understand the scope of the modifications because the roles of the agents were very clear. Sometimes they had to change the storage schemes for the users' data, but they also could do this quickly because they only had to change the definition of instance variables of the agents.

A Web application is a server application that performs multiple tasks concurrently. So programmers have to take care when using multithreaded programming that requires advanced skills. In Caribbean, although multiple agents run on a server simultaneously, each agent is not invoked concurrently since messages for the agents are serialized. Therefore programmers do not need to take care of concurrent processing of a single agent.

4.2 Performance

In this section, we briefly show the results of a performance evaluation of an agent server. The detailed description appeared in [4]. The brief description of the benchmark test which we used is as follows. The server manages users' data. Each user's data consists of 50 strings of 10 characters and an integer used as an update counter. All data was created before the start of the test. There are two operations for each user, an inquiry operation and an update operation. The former operation only reads the user's data. The later operation reads the user's data and increments the user's update counter. The number of users was 100,000. The percentages of inquiry and update were set to 100% inquiry, 50% inquiry and 50% update, and 100% update in the three tests. Table 1 shows the results of performance comparisons between an agent server approach and an existing approach using a standalone program, JDBC, and DBMS. In both cases, all users' data is kept in memory.

Table 1 Results of The Benchmark Test (unit: processes per second)

	100% inquiry	50% inquiry 50% update	100% update
The Agent Server Approach	22,301.5	2,077.9	1,069.33
The DBMS Approach	168.05	125.4	100.61

Pentium III 600 MHz CPU
1 GByte Memory
Windows NT4.0
IBM JDK1.1.8

From the results, we see that the agent server approach is much faster than the DBMS approach. The reason is that referring to an user's data is done by referring directly to objects in the agent server. On the other hand, an application server has to copy each user's data from the memory cache of a DBMS by using inter-process communication in the DBMS approach.

5 Related Work

Agent server technology which manages users' agents is included in the concept of mobile agent technologies such as Telescript[6] and Aglets[7]. In mobile agent technology, an agent server is the server to which a mobile agent comes and where it performs tasks by accessing stationary agents. However, the agent server technologies of the mobile agent are focused on security, agent transferring mechanisms, and program management mechanisms. The agent server technology described in this paper focuses on performance, scalability, and reliability, as well as flexibility, because these issues are central for these systems.

Agent server technologies can greatly improve performance when a server computer has a sufficiently large memory. On that point, our technology is similar to the main memory database management systems (MMDBMS) [8]. Since MMDBMS keeps all data in memory, it can ignore the mechanisms for managing physical storage, allowing it to improve performance a great deal. Moreover, the technology can be applied to the cache mechanism accompanying an application server [9]. MMDBMS provides applications with a standard data access interface such as SQL. This means an application can search its data by specifying a complex search condition, but it incurs the overhead of using SQL. On the other hand, our agent server provides an application development framework specialized for Web applications. Although it provides no search engine like DBMS, an application can access data by referring to the instance variables directly.

6 Conclusion

In this paper, we have introduced an agent server to develop a Web applications. A model of the agent server is based on an "agent-oriented programming model." The model is easy for programmers to understand and it provides a flexible framework. Therefore, it can solve problems in developing Web applications, such as short development periods and frequent specification changes. Moreover, since each agent is independent, the model can represent concurrent tasks in natural way. Since each agent is persistent, programmers can manage users' data easily. Another important issue is performance. Web applications must be capable of very high performance. Fortunately, modern computers have very large physical memory, and by using this memory efficiently, systems can achieve the required high performance. As we showed in Table 1, our agent server does this. However the agent server is efficient for managing non-shared data such as users' data, though it has some limitations on managing shared data, such as product information in an electronic mall. We should use an agent server with a DBMS and other tools for developing such kinds of Web application.

References

[1] IBM alpha Works Caribbean、<http://www.alphaworks.ibm.com/tech/caribbean>

[2] H. Tai and G. Yamamoto: An Agent Server for the Next Generation of Web Applications, *The 11th International Workshop on Database and Expert Systems Applications (DEXA-2000)*, IEEE Computer Society Press, Sep. 2000

[3] G. Yamamoto and H. Tai: Architecture of an Agent Server Capable of Hosting Tens of Thousands of Agents, IBM Research, Research Report RT0330 (1999)

[4] G. Yamamoto and H. Tai: Performance Evaluation of An Agent Server Capable of Hosting Large Number of Agents, Autonomous Agents 2001, IBM Research, Research Report RT0381 (2001)

[5] N.R. Jennings and M. Wooldridge: Agent-Oriented Software Engineering, in *Handbook of Agent Technology*, J.M. Bradshaw,ed., AAAI/MIT Press. (to appear)

[6] J.E. White: "Mobile Agents," Software Agents, J.M. Bradshow, ed., AAAI Press/MIT Press, 1997

[7] D.B. Lange and M. Oshima: *Programming and Deploying Java Mobile Agents with Aglets,* Addison-Wesley, Boston, 1998

[8] D.J. Dewitt, et al: Implementation Techniques for Main Memory Database Systems, *Proceedings of ACM SIGMOD Conf.*, 1984

[9] TimesTen, <http://www.timesten.com>