2003－MPS－45 (4)
2003／6／24

Specification of a Parallel I/O Control Agent for Large-Scale Simulation Users

Yuki Matsuoka * Hitomi Matsuyama * Mika Koganeyama * Chiemi Watanabe *
Yutaka Ueshima † Kazuki Joe *

Abstract

An integrated management system for large–scale simulations saves the output data of simulation results without considering the amount of empty space on the disks in the data servers. The present problem is inefficient operations on the data servers because each the disk sizes on the data server are unbalanced. We design and implement a prototype parallel I/O control agent for large–scale simulation data servers which supervises the data servers, makes suggestions to users and moves data files between disks to keep disk sizes well-balanced.

1 Introduction

Large-scale and extremely precise simulations are performed with progressive parallel plasma code on the massively parallel computers at JAERI KANSAI Research Establishment Advanced Photon Research Center (JAERI KRE APRC). The integrated management system called the "P–cube support system" is used to execute large–scale simulations.

However users must understand the simulations in detail to operate the complicated system. Otherwise, simulation results cannot be saved to a data server and are transmitted to other servers ineffciently.

Considering the above background, we began research on the development of an autonomous agent system which performs support operations and gives suitable advice to users[1]. An Agent is a system which carries out rational operations using their own decision-making. Agents consider and execute plans to achieve goals autonomously. One of the necessary functions the agent system must provide is parallel I/O control capability. The P-cube support system does not consider variable disk sizes on data servers when simulation calculation results are saved. Since the size of each disk can be different, the data servers are often used inefficiently. We design and implement a parallel I/O control agent which supervises the data servers and balances disk sizes autonomously[3].

We have previously implemented a prototype of the agent with minimal capabilities[2]. It is difficult for the prototype to communicate with other agents interactively and to extend its capabilities because it was implemented on the same server. In this paper, we design a parallel I/O control agent for large-scale simulations at JAERI KRE APRC. We divide the functions of the agent into several roles, and implement each role extensibly using JSP, servlets, and JavaBeans

This paper is structured as follows. In section 2, we explain the structure of the data servers. In section 3, we propose the parallel I/O control agent. Finally, we conclude in section 4.

2 Structure of the data servers

2.1 Environment

The parallel I/O control agent manages two data servers and two graphic servers. In this paper, we discuss agent implementation issues for the data servers only. We explain the structure of the data servers in JAERI KRE APRC below.

There is a massively parallel computing environment for the execution of large–scale calculations, and a developing–computing environment for middle–scale or small–scale calculation at JAERI KRE APRC. There are some data servers used for job I/O in the both of environment.

The data server on the massively parallel computing environment consists of eighteen sets of AlphaServerES40-4 which are called "sscmpp2" to "sscmpp19". Each node is connected to Compaq StorageWorks MA8000 high-speed storage via FibreChannel. The disk storage consists of RAID5, and the total capacity of the storage is about 15TB. The disk areas on the storage are divided "work01" to "work90".

The data server on the developing–computing environment consists of four sets similar to those above called "sscmppl2" to "sscmppl5". The way to connect to each node and the structure of the disk storage are identical. The disk areas on the storage are

*Nara Women's University, Graduate School of Human Culture
†JAERI, KANSAI Research Establishment,Advanced Photon Research Center

divided into "work01" through "work16".

2.2 A problem in the data server

The data servers are used not only by the P-cube support system users but also by many users in various research groups (about 120 people). Each user can use the entire disk area on the data servers. Some users save their data on the same disk area, so the usage ratio of each disk is not uniform leading to imbalance.

Simulation calculations using the P-cube support system are distributed to multiple productive nodes. The results on each node are saved to multiple disk areas on the data server in parallel. Although each disk area is unbalanced before saving the output data, the system saves them blindly, even if the save operations make the disk imbalance worse. If the disk size exceeds a given value, the system fails to save the output data, the simulation execution cannot continue, and operation is interrupted.

Therefore, we have designed and implemented agents which monitor the disk usage periodically and move data from higher load disk areas to lower load ones so that the disk sizes are kept balanced automatically.

3 Parallel I/O control agent

3.1 Structure of the agent

The parallel I/O control agent consists of the following three agents as shown in Fig.1.

- Data Server Supervising Agent
  Dynamically checks whether the disk sizes are below the upper bound value[4].

- Disk Size Balance Adjusting Agent
  Calculates the disk sizes after moving the data files to help maintain balanced disk sizes.

- User Response Confirmation Agent
  Receives requests from users for performing balancing operations for the users.

We implemented prototypes of the above agents before running on the real data server. We create a virtual data server as shown in Fig.1 which is temporary for the real data server. We create sixteen directories (called work01–work16)in the virtual data server so that each directory has subdirectories for research groups and users. Files with various sizes are put in user directories at random.

We explain the capability and the implementation of each agent below.

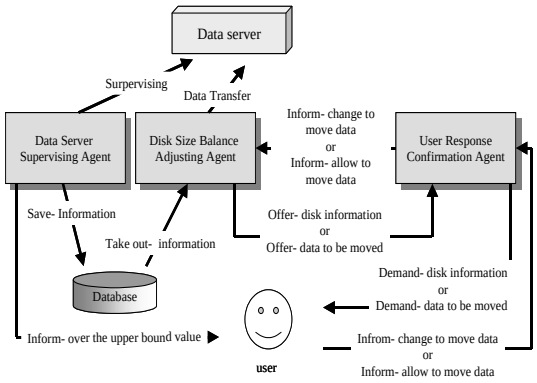

Figure 1: Structure of the parallel I/O control agent

3.2 Data Server Supervising Agent

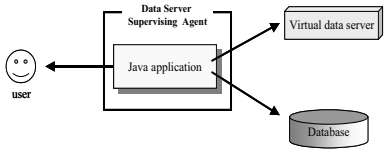

Figure 2: Implementation of the Data Server Supervising Agent

The Data Server Supervising Agent is implemented as a Java application of Fig.2. The agent supervises temporary directories and user directories in the virtual data server periodically, and checks whether directory sizes are below a given upper bound value, and sends e-mails to users if directory sizes are over the upper bound value. These processes are implemented as described below.

- Supervising the virtual data servers
  The agent issues an external command (the "du" command) periodically to supervise the directory sizes in the virtual data servers. The external command is implemented using the "exec" method of the java.lang.Runtime class. The agent gets each directory size in the order of work01 to work16. The agent also gets user directory sizes and file sizes.

- Saving information about the virtual data servers to the database.
  The agent saves information about the virtual data servers obtained by the supervising process to the database. The database is connected via the JDBC driver. Information about temporary

directories and user directories are managed the management table.

- Sending e-mails to users
  If temporary directory sizes are over the upper bound value, the agent sends e-mails to the owners. A URL which links to the login page created by the User Response Confirmation Agent is provided in the e-mail. The JavaMail API is used to send the e-mails.

3.3 Disk Size Balance Adjusting Agent

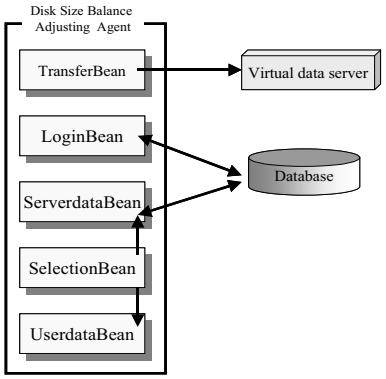

Figure 3: Implementation of the Disk Size Balance Adjusting Agent

The Disk Size Balance Adjusting Agent is constructed with Java program components called Beans as shown in Fig.3. The agent receives a request from the User Response Confirmation Agent and calculates and maintains optimal directory sizes according to the user's request. The agent performs three kinds of processes; it finds candidate files to be moved, selects the files to be moved according to user preference, and moves the files. We explain the processes below.

- Find candidate files to be moved
  When a user receives a request from the Data Server Supervising Agent as described above, and accesses the URL in the e-mail. The Disk Size Balance Adjusting Agent checks whether user name and password inputted via the Web match the user name and password in the database management table (LoginBean). If authentication is successful, the agent gets temporary and user directory sizes, and file names and sizes from the database (ServerdataBean), and is invoked to find candidate files to be moved (SelectionBean). The agent calculates the average of the current temporary directory sizes. The candidate files to be moved are found so that each temporary directory size approaches the average size. Also the candidate files are determined by the user for the "over used" temporary directory. Assuming that the candidate files are moved to another temporary directory whose size is less than the average, the candidate files are reserved for the notification to the referencing user. These processes are repeated until the size of the "over used" temporary directory size is below the average.

- Select the files to be moved according to the user's preference
  It receives a notification from the User Response Confirmation Agent (UserdataBean). If it indicates that the user wants to change the file sizes and/or the target directories to be moved, the agent selects the candidate files (SelectionBean).

  – When changing the target directories: the agent calculates the directory sizes after the candidate files are moved.
    If the target directory sizes are below the average, the target directories are accepted. Otherwise the agent finds other candidate directories and informs the user of the new candidate.

  – When changing the file sizes: the agent calculates the directory sizes after the candidate files are moved.
    If the directory sizes are less than the average, the requested file sizes are accepted. Otherwise the agent finds other file sizes and informs the user of the new sizes.

  The agent informs the User Response Confirmation Agent of the above processes.

- Move files.
  When the agent receives the user's permission to move the files from the User Response Confirmation Agent, it performs the following processes (TransferBean):

  1. The target files are moved to the target directories. If files with the same names as the target files exist, the target file names are changed by the agent.
  2. To guarantee transparent operation for the users regardless of file transfer, the agent creates symbolic links.

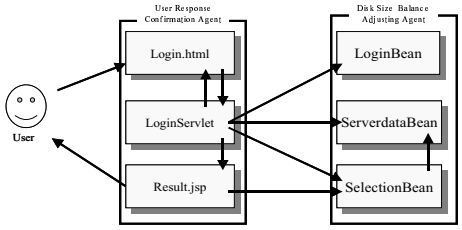

Figure 4: Implementation of the User Response Confirmation Agent : the first action

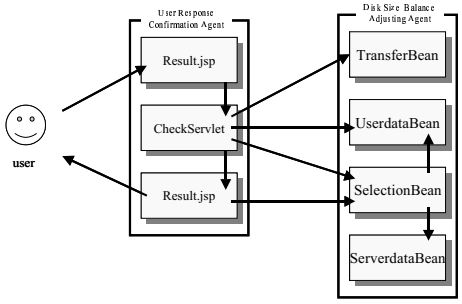

Figure 5: Implementation of the User Response Confirmation Agent : the second action

3.4 User Response Confirmation Agent

The User Response Confirmation Agent consists of HTML pages, Java servlets, and JSP pages in Fig.4 and Fig.5. The agent acts according to the user's requests. There are two cases when the agent is invoked as described below.

- The user's first access to the URL in the notification E-mail (Fig.4):the agent informs the Disk Size Balanced Adjusting Agent of the user name. If the agent can confirm that the user can log on, the agent gets the user information from the Disk Size Balance Adjusting Agent about currently available directories and candidate files to be moved. If the agent cannot confirm that the user can log on, the agent displays the login page again.

- The user's modification of file sizes and/or target directories (Fig.5): When the agent receives the user's modification information for the file movement, it informs the Disk Size Balance Adjusting Agent of the changed target directories and/or file sizes to be moved. When the agent receives the user's permission for the file movement, it informs the Disk Size Balance Adjusting Agent of the permission. After the actual file transfer, the agent receives the completion notification from the Disk Size Balanced Adjusting Agent.

4 Conclusion

In this paper, we proposed the parallel I/O control agent for large–scale simulations on the data servers at JAERI KRE APRC, and explained prototype implementation issues of the agent. As the result, it turned out that any agent could communicate with other agents interactively and we could extend the agent functions easily. By extending the JavaBeans described in this paper to EJB (Enterprise JavaBeans), the agent can communicate with other agent on other servers.

There are two more efficient agents to construct. The first subject is a knowledge–base for individual users. Therefore, we require more supervising information than was explained in section 3.2, and we need to create user profiles. The second subject is algorithms which make good use of the knowledge–base for each user.

References

[1] H. Matsuyama, Y. Matsuoka, M. Koganeyama, Y. Ueshima, K. Joe: "Design and Implementation of an Automatic Learning Agent System for Large–scale Simulation Cycles", IPSJ SIGMPS, 2002–MSP–42, Vol.2002, No.114, pp.17–20 (2002).

[2] Y. Matsuoka, H. Matsuyama, M. Koganeyama, Y. Ueshima, K. Joe: "Parallel I/O Control Agent for Large–scale Simulation Database", IPSJ SIGMPS, 2003–MPS–43, Vol.2003, No.20, pp.1–4 (2003).

[3] J. Cao, D.J. Kerbyson, and G.R. Nudd: "High Performance Service Discovery in Large–Scale Multi–Agent and Mobile–Agent Systems", International Journal of Software Engineering and Knowledge Engineering, Special Issue on Multi–Agent Systems and Mobile Agents, World Scientific Publishing, 11(5), 621–641, 2001.

[4] J. Cao, S. A. Jarvis, D. P. Spooner, J. D. Turner and G. R. Nudd: "Performance Prediction Technology for Agent–based Resource Management", 11th IEEE Heterogeneous Computing Workshop (HCW02), Marriott Marina, Fort Lauderdale, Florida, 15–19th April 2002.