APPLICATION SERVER AND PUSHING TECHNOLOGY ON COACK-II

Takashi Kosuge, Isamu Abe, Junichi Kisiro, Kazuyuki Nigorikawa, Masakatsu Mutoh*, and Shin-ichi Kurokawa
High Energy Accelerator Research Organization (KEK), 1-1 Oho, Tsukuba, Ibaraki 305-0801, Japan
*LNS Tohoku Univ. Taihaku, Sendai, Miyagi, Japan

1 INTRODUCTION

The COACK-II (Component-ware Oriented Accelerator Control Kernel)[1] is a very powerful and flexible system. Many client computers that run with many kinds of operating systems use it [1][2][3]. We must update and maintain the COACK-II applications on these client computers.

The push system is required on the application server to reduce the application labor, upgrade and maintenance times

We are developing a new program push system for the COACK-II and a system is required that:

* It must be able to control computers that are running different kinds of operating systems.
* It must be simple.
* It must be updateable.

At this time we have made the first-version programs (including the test version) of the system. We describe here the details and how the system actually works.

2 OVER VIEW

2.1 Layout of the application push system

The application push system consists of 3 types of program the push commander, the kicker and the application manager. (Fig. 1)

The push commander works on the application server, and controls the updating of COACK-II application programs. We are developing the push commander with Visual Basic.

The kicker works on a client computer. When the kicker receives a request from the push commander, it executes the application manager. We have prepared a DCOM version and a TCP/IP socket version kicker. We are going to make compliant different kinds of operating systems by using these types of kickers.

If the kicker starts up the application manager, the application manager tries to connect to the push commander with the TCP/IP socket. We made the application manager with Perl[4][5]. It is possible to write a program that does not depend on the operating system.

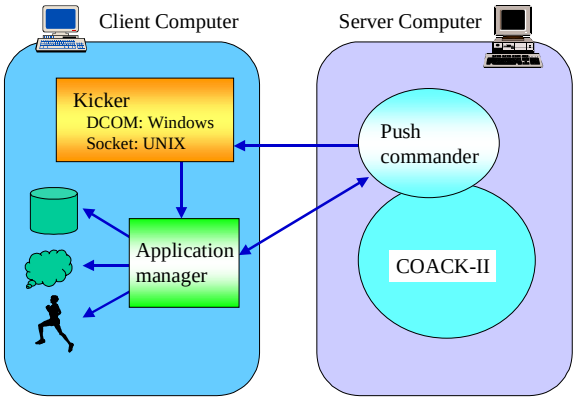

Figure 1: Composition of the system

2.2 Procedure of running programs

We can manage the client computers from the application server with this system. The procedure of the three programs is shown Fig. 2.

First, if we choose the client computer and push the connect button, the push commander connects to the kicker with the DCOM or TCP/IP socket. Then the push commander sends the "call ID" to the kicker and waits for a connection from the application manager.

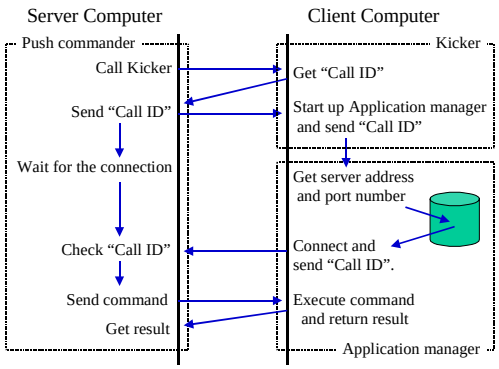

Figure 2: Procedure of programs

[1] This project is supported by JST (Japan Science & Technology).

Next, the kicker starts up the application manager and sends the "call ID" to the application manager.

Then, the application manager tries to connect to the push commander with the TCP/IP socket. The application manager knows the application server and TCP/IP socket number beforehand. An easy security mechanism with a call-back style connection has been made.

Finally, the push commander checks the "call ID" from the application manager.

The application manager works according to the commands given by the push commander after a connection had been established.

3 DETAILS OF THE PROGRAMS

3.1 The push commander

The push commander runs on the application server with Windows NT. The push commander is being written with Visual Basic. The present push commander is a test-version program. Thus, it can not connect all the client computers at the same time. We are planning to develop a multi-connect type push commander in the future, which will be connected to COACK-II with DCOM. After which we plan to use databases through COACK-II.

3.2 The kicker

The kicker runs on the client computer. It only starts the application manager and sends the "call ID". We prepared 2 types of kickers: the one is a DCOM-based kicker, the other a TCP/IP socket-based kicker. The DCOM-based kicker is used for Windows, and receives requests from the push commander with DCOM. The TCP/IP-based kicker is used for UNIX and other operating systems.

3.3 The application manager

The application manager runs on client computers. If the kicker starts it, it will connect to the push commander with the TCP/IP socket. If the connection is already established the application manager works according to the text-based commands from the push commander.

The applications of COACK-II are installed on resolved directories on the client computer. We can install the COACK-II application programs through the application manager (Fig. 3).

The application manager has a script mechanism and the script files are also installed on a resolved directory. If we update the script files using an application manager mechanism, we can update it dynamically.

We made the application manager with Perl. We must install Perl on the each client computer first, although the same application manager programs are able to run on many kinds of operating systems without changing the programs.

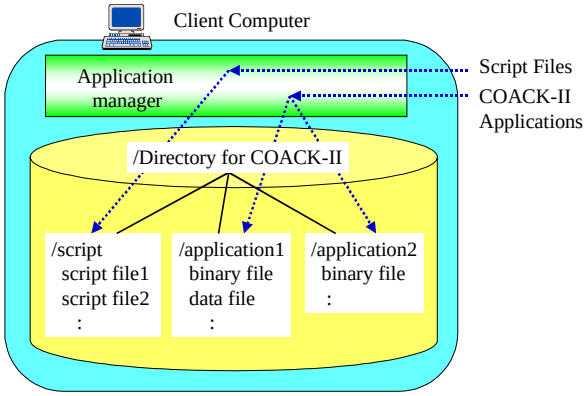

Figure 3: Application manager

The script mechanism is issued by the "eval" function of Perl. If we set the strings into the variables and call the "eval" function, the string will be executed by the "eval" function, just the same as when using the Perl commands.

Here, the application manager finds the script file and executes it if the command from the application manager is not a common command on the application manager.

If an error occurs in the "eval" function, it only returns an error code to the main program. Program termination by the error does not occur in the "eval" function. Then, the application manager will not crash if we install the erroneous script files.

4 ACTUAL OPERATION

The actual running state of the push commander is shown in Fig. 4. If we set the IP address of the application manager on the push commander and push the connect button, the connection will be established, as described above. Then, the COACK-II application list of the client computer will appear on the list box of the push commander. Here, the push commander sends the "ListPrograms" command to the application manager and

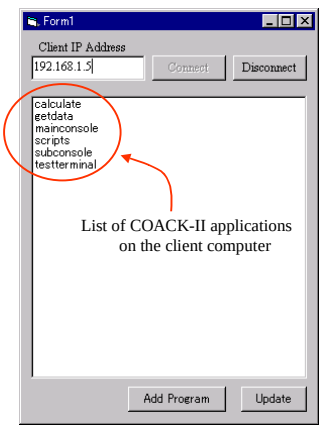

Figure 4: Push commander

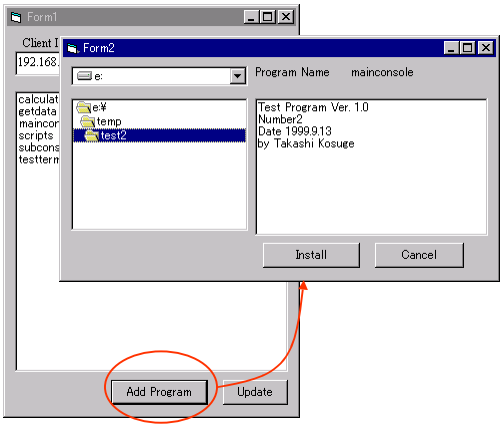

Figure 5: Transferring binary files

shows the application list that is sent by the application manager.

The system also has a binary transfer mechanism. (Fig. 5) This mechanism is used for transferring COACK-II applications to the client computer. In this case, the push commander tells the directory to install and send the application name to the push commander. Then, the push commander sends the spliced binary data to the application manager.

5 ADDITIONAL FUNCTIONS

We can easily and dynamically add functions to the application manager, as explained above. It depends on the adoption of Perl. Perl is very powerful, and we can use its functions easily. We are now developing some additional functions with the advantages of Perl. The functions are introduced here.

5.1 Accessing the ftp server

We are developing an ftp client function for the application manager, allowing installation of COACK-II applications from the application server. However, it is required that the push commander runs on the application server. If the application manager has an ftp client function, we can divide the push commander and application server. In addition, a specific program is not required on the application server; instead, only a generic ftp server is required.

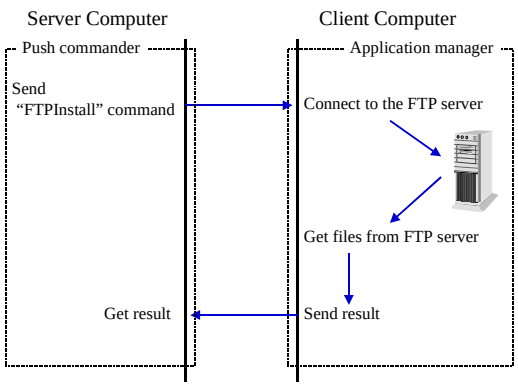

Figure 6: Install programs with FTP

We are planning to use the Perl ftp module to add the ftp client function to the application manager. Beforehand, we must install the Net::FTP[6] module on the client computer allowing easy use of ftp functions. The actual procedure of the mechanism is shown in Fig. 5.

5.2 Administration of client computers

We are planning to prepare a mechanism that can administer COACK-II applications on client computers easily, and we are going to use Perl for developing the mechanism.

Each configuration file will be put on a directory of each COACK-II application. We will be able to change the settings of the COACK-II client programs by editing each configuration file through the application manager.

We will make the mechanism with the script system of the application manager.

6 CONCLUSION

A program push system for COACK-II with very simple mechanisms has been developed. A system which does not depend on the operating system, that can be dynamically updated with Perl. Our pushing technology became an important part of COACK-II.

REFERENCES

[1] I.Abe, et al., COACK-II Project on Accelerator Control Kernel Development, this conference.
[2] K.Nigorikawa, et al.,GUI and I/O Interface for COACK-II, this conference.
[3] http://ninja.kek.jp/COACK2/
[4] Randal L. Schwarts, and Tom Christiansen, "Learning Perl Second Edition", 1997, O'Reilly & Associates, Inc.
[5] Larry Wall, Tom Christiansen, and Randal L. Schwarts with Stephen Potter, "Programming Perl Second Edition", 1996, O'Reilly & Associates, Inc.
[6] Sriram Srinivasan, "Advanced Perl Programming", 1997, O'Reilly & Associates, Inc.