韓國 컴퓨터情報學會 論文誌
第11卷 第5號, 2006. 11.

2006-11-5-2-9

교환기 소프트웨어 개발을 위한 호스트 기반 데이터베이스 시뮬레이터의 구현

박 영 호*, 이 호**

An Implementation of the Host-based DBMS Simulator for Developing Switching System Software

Young-Ho Park*, Ho Lee**

요 약

교환 시스템을 구동하는 대규모 소프트웨어에서는 여러 기능간의 상호 데이터 교환과 그 처리를 위하여 실시간 데이터베이스의 사용이 필수적이다. 교환기용 DBMS 소프트웨어 개발에서는, 데이터베이스 질의어를 포함하는 응용 프로그램을 우선 호스트 컴퓨터상에서 개발하고, 이렇게 개발한 응용 프로그램을 나중에 교환기 본체에 로딩하여 그 기능을 시험한다. 호스트 컴퓨터 상에서 데이터베이스 응용 프로그램을 개발할 수 있도록 하기 위해 연구 개발한 것이 호스트 기반 DBMS 시뮬레이션 시스템(HDBMS) 이다. 본 논문에서는 HDBMS의 역할과 기능, 시스템 구성, 시스템 구현을 위한 기술적인 세부 사항에 대해 우리가 연구한 내용을 소개한다.

Abstract

For such large-scale software as for operating a switching system, the use of real-time databases is essential for data exchanges among various functions and their data processing. Under the environment of developing the DBMS software for a switching system, the application program including database manipulations is first developed on a host computer and then the developed program is loaded into a switching system for its tests. To make it possible for DBMS manipulation software to be developed on a host computer rather than a switching system itself, we developed a host-based DBMS simulation system(HDBMS). In this paper, we presented the roles and functions of HDBMS, its system structure, and the technical details for implementing HDBMS.

▸ Keyword : DBMS, HDBMS, DBMS Simulator

• 제1저자 : 박영호
• 접수일 : 2006.10.19, 심사일 : 2006.10.25, 심사완료일 : 2006.11.18
* Prof. of Dept. of Multimedia Science, Sookmyung Women's Univ.
** Prof. of Dept. of Computer Information Security, National College of Korea Rehabilitation and Welfare

Ⅰ. Introduction

Switching systems are the systems which function by means of interactions of the multiple execution blocks which belong to the call processing and administration & maintenance software working on distributed subsystems[1], the functions of which are supported by such system software as the real-time operating system[2] and real-time DBMS. When such large-scale software systems as for running switching systems are developed, it is essential that real-time DBMS should be used for performing data exchanges among various functions and their processing[3]. In the development of switching system software, the functions containing data manipulation language are usually developed on an independent host system from a target switching system. These executable blocks are later on loaded into a target switching system, in which they are run being supported by an operating system and DBMS run-time systems. For testing execution blocks on an target system, it is prerequisite that you should check out the execution blocks which may be concurrently being tested on the same target system by other users. It would be better to load the execution blocks to be tested into a target such a checkout. Moreover, with newly developed execution blocks it is probably difficult to make tests of their functions since they may not be compatible with the other existing functions in the integrated running environment. In case that a newly introduced executable blocks have fatal errors with them, they may influence the other functions and propagate troubles through the whole system.

A host-based simulation system can be an alternative to resolve above-mentioned problems with the instant tests on a target system[4].

We have developed a host-based simulation system for making tests of DBMS-related functions. The developed system is a DBMS simulator, by which application's executions containing data manipulations are simulated on a host rather than a switching system. We are supposed to develop the applications including data manipulations using this simulator and put the developed applications on tests on a real switching system. Given that a target system itself has a limitation to allowing applications to be simultaneously tested, this simulator may be a host-based simulation environment which helps make a test of an application without a target system. The simulator is designed to accomodate multiple PLDs(Processor Loaded Data) and allow the data manipulation functions using PLDs in multiple executable blocks to be tested on a host. The host-based database system named HDBMS can simulate the situation where multiple processors use multiple databases.

In this paper we make an introduction to the implementation of the HDBMS system and describes the merits of using the system as well. The target switching system to be simulated is the ETRI-developed HANbit/ACE ATM switching system[5] which has a distributed architecture composed of the subsystems[1] like OMP and CCCP00~CCCPn processors. The target system is equipped with the ETRI-developed database system named DREAM S(Distributed Real time database Management system for SPARC)[6]. In the chapter two, is made the introduction to the roles and functions of HDBMS. In the chapter three, We categorize the whole system into functions, and describe the necessities and operation characteristics of each function and relationships among them. In the chapter four, we propose a scheme of implementation and specify its propriety and possibility of extension. In the chapter five, conclusion and the direction of future research are mentioned.

II. Roles and Functions of HDBMS

2.1 Roles of HDBMS

Roles of the HDBMS are as follows.

Firstly, the HDBMS provides the applications to be tested with testing environment. There are two types of users in terms of the number of users to make tests on a host. One is a user who keeps a registered database unshared with other users and is assured of the independence of execution from them, which means that his execution is not affected by other users' executions while testing on a host. The other is one of the users who can make a test simultaneously with other users on the same host into which each user loads his database and runs his application. On having completed the execution, it is assured that each user's execution result is not affected by others.

Secondly, the HDBMS can be used as a test bed where you can test a prototype for further development of DBMS.

Thirdly, the HDBMS enhances the possibility that the real-time DBMS running on a switching system can be run on generic host systems as well.

2.2 Functions of HDBMS

The functions of HDBMS can be broken down into five categories largely. Each category in turn contains various functions.

1) A function to provide an interface to existing DREAM-S kernel

The HDBMS has the interface for integrating the existing DREAM-S kernel with it being unchanged. The interface can cope with even the expansion and modification of DREAM-S and provides the same access methods as on a target system.

2) A function to support executions of multiple processes

In order to support multiple processes, there are a single-user multi-process supporting function and a multi-user multi-process supporting function. The former can support the multiple processes which are simultaneously run by a single user. The latter is able to support the multiple processes which are simultaneously run by multiple users. For efficient management of multiple running processes, it is assured that one process performing a database manipulation is not preempted by other processes, which means that a running process is assured of being given the independent right to be completed.

3) A function to support multiple databases

There are three types of database support functions. The first one is for database dynamic loading and unloading. The second one is a function for database sharing joint and disjoint. The last one carries out confirmation of a database access right, which analyzes a query request in shared database environment.

4) A function to support the IPC with large capacity

Large data transaction is possible in the HDBMS. When a client process needs transmitting or receiving large data to or from a demon process in server, The amount of data transacted can freely be regulated by use of shared memory.

5) A function as an intelligent server

The HDBMS server takes the form of such versatile intelligent server as mentioned in the following. It can function as multiple servers, and interact through grasping users and processes, and output server's status, and recover system resources by means of a signal handler, and take in additional functions when needed.

III. Structure of HDBMS

This chapter specifies the structure of implementation of functions mentioned in the chapter two.

3.1 Overall structure

The HDBMS has a structure of multiple clients and multiple servers in order to simulate the running environment with a real switching system. Prior to loading on the HDBMS server, the executable file of application software is linked with the library that is specific to a processor they belong to. The EDML in a executable file is transferred to a server through a library while the IPC with a host system is used as

infrastructure for EDML transfer. A database kernel resides in the HDBMS sever to process the client's EDML transferred to itself. For EDML processing, it is necessary that the server's database should be loaded before it is processed. A user is supposed to set up a host database for tests using administration commands.

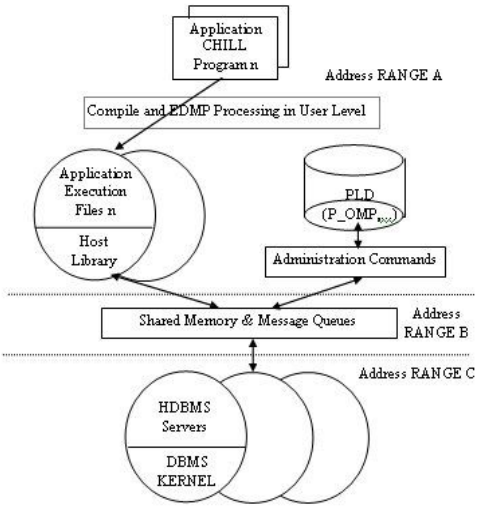

Fig. 1 Host-based DBMS Simulation Environment with Multiple Processes and Databases supported
그림 1. 다중 프로세스와 데이터베이스를 지원하는 호스트 기반 DBMS 시뮬레이션 환경

Fig. 1 shows the overall structure of the host-based DBMS simulation system and also depicts the IPC communications among processes running in different address spaces. The client implies the executable file of application software and the administration command to dynamically load a database and make it shared. Using administration commands you load a specific database and run the relevant executable file already compiled and linked with library. Now, the executable file sends the EDML embedded in it to HDBMS server using IPC and receives its responses.

3.2 Implementation characteristics

The implementation of HDBMS is characterized by the followings. The HDBMS system is organized for multiple clients and servers to support the executions of multiple execution files. It has adopted identification of an execution file to know the source of user's query request. The transmission of large data is supported by means of shared memory. Errors which may occur due to simultaneous accesses to the same database can be avoided through the locking method applied to the unit of a query. There is a function to recover system resources from faults occurring on system running.

The structure of HDBMS is divided into a client part and a server part as depicted in Fig. 1. The client is of the execution block linked with library and administration commands, while the server consists of maximum three HDBMSs including a deamon process.

IV. Implementation

In this chapter, is specified the details of HDBMS with respect to the client and the server.

4.1 Organization of HDBMS Server

Fig. 2 represents the organization of HDBMS, which consists of a module for supporting multiple user environment, and a module for dynamic loading of database, and memory spaces(DB1~DBn) for keeping the database loaded into a server, and sets of various data structures(DSs) which are necessary for managing users' accounts and access control to database. The whole HDBMS is organized for its components to call for HDBMS kernel through the database access interface.

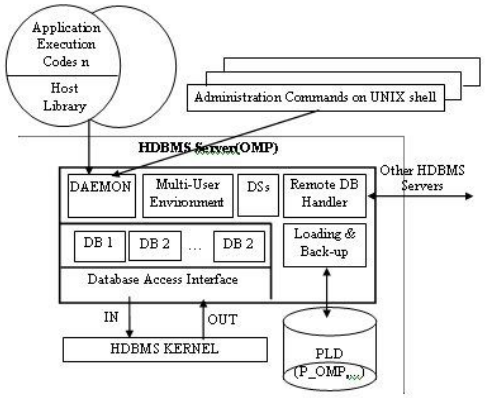

Fig. 2 Organization of HDBMS Server
그림 2. HDBMS 서버 구조

On reception of an administration command, the daemon process of HDBMS server makes the configuration be established. When the daemon process receives a request for database manipulation, it calls for database kernel and transmit its result of processing to a client.

Fig. 3 is a conceptually simplified picture representing data flows. The IN represents the data a user initially creates and transmit to server. The OUT represents the data returned as a result after being processed by a host DBMS. That is to say, Fig. 3 represents the processing result to a user's request.

The detailed explanation is given in the following. An execution file, or a client, is created when a source program is processed by EDMLP(Embedded Data Manipulation Language Processor)[3] and compiled and linked with library.

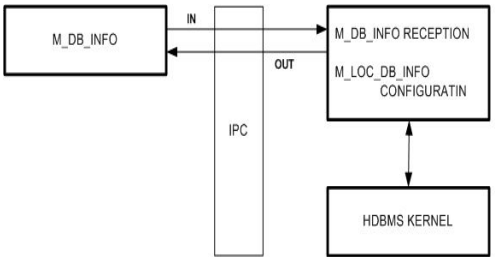

Fig. 3 Conceptual Diagram of Data Flow between Client and Server
그림 3. 클라이언트 서버간의 데이터 흐름도

The execution file created as mentioned above contains several database manipulation commands. The M_DB_INFO is a structure with the expanded information of database manipulation commands. The server receives it, and converts it into a M_LOC_DB_INFO construct and call for the HDBMS kernel. What is marked with OUT among the information returned by a kernel execution are the values which are returned after having been changed.

4.2 Messaging Mechanism

Fig. 4 depicts the process to change the form of data for using shared memory.

The intial data residing in a client is not to be directly transferred to a server but to be done by use of the IPC with a host system. The shared memory method is generally used for large data transfer owing to its high efficiency. With the HDBMS, the same method is used to transfer client's data to a server. As seen in Fig. 4, when a client sends the expanded information of a database manipulation command, it transforms the information structure into a sending form which is used on writing data in shared memory and on being read by a server.

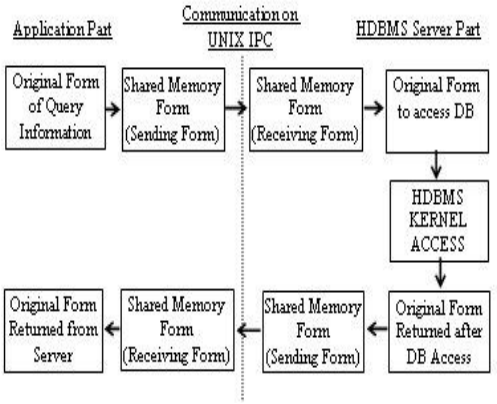

Fig. 4 Data Conversion and Communication by using Shared Memory
그림 4. 공유 메모리를 사용한 데이터 변환 및 전달

When a server receives data, it transforms its data structure into the structure used on calling the HDBMS kernel and then calls the kernel. The information, which has been changed by kernel processing, is transformed into a sending form for being returned to the very user's process which requested the database manipulation. The result is sent to a client via shared memory and the client finally delivers the result to its user. The locking method using Semaphore[7] is applied to the clients requesting for database manipulations. This method can prevent different processes from accessing the same memory address simultaneously while processing a database manipulation. In a server, the locking method is applied to reading and writing shared memory. By means of using this method, during communicating with one client, the other clients are not allowed to access the same server's resources.

4.3 The Primitives Available in an Execution File

The primitives available in an execution file, namely embedded data manipulation language(EDML), are listed in Table 1. The command format of the primitives on a host system is the same as that on a target system.

Tab. 1 Host Primitives
표 1. 호스트 프리미티브

$ SELECT : read a tuple to meet a specified condition from a given relation.
$ UPDATE : update an attribute according to a update clause.
$ INSERT : insert a new tuple into a given relation.
$ DELETE : delete a tuple to meet a specified condition from a given relation.
$ ASELECT : find an attribute in a tuple to meet a specified condition.
$ MAXI : find a maximum value among the tuples to meet a specified condition.

4.4 Administration Commands

Another type of a client except for an execution file is administration commands available on host shell to set up server's environment. After dynamically load the PLD using the administration commands, a user can confirm the result of database manipulation by running the execution file which is specified as the 1st client. The server's deamon process always keeps monitoring its inputs. If the input is an administration command then the deamon process sets up the server's environment. If it is a database manipulation command, it sends the result to a client after processing the database manipulation request. Fig. 5 shows the relationship between administration commands and EDML.

A user who uses the PLD loaded in host DBMS can be registered in server as one of either a PLD owner or a PLD user. The PLD owner means a user who first registers the PLD in a server. He can have only one PLD in a target server and has the right to register a sharing user or release it.

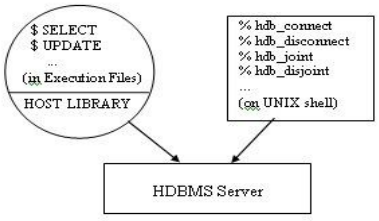

Fig. 5 Relationship between Administration Commands and EDML
그림 5. 운용 제어 명령어와 EDML 간의 관계

On the other hand, the PLD user is a user who wants to share the PLD, a registered owner for which now exists. He can also have only one PLD in a target server. The following is the explanation on implementation of administration commands and how to use them. The administration commands are shown in Table 2.

Tab. 2 Administration Commands
표 2. 운용 제어 명령어

```
% hdb_connect [target_name] [pld_name];

% hdb_disconnect [target_name] [pld_name];

% hdb_joint [target_name] [pld_name]
[sharing_user_account];

% hdb_disjoint [target_name] [pld_name]
[sharing_user_account];
```

Using commands shown in Table 2, a user can register a database in a server in the way he wants. That is, designating the server with target_name and the database with pld_name. The command with sharing_user_account is a client which can be used for allowing now registered database to be shared by other user or to recover the given right for database from him. For instance, suppose that a user whose account is userRomeo wants to use a PLD database the name of which is P_0_Romeo in a host, which is an OMP server. The process of work is like the following. The user userRomeo loads the PLD P_0_Romeo into the OMP server using a host shell command hdb_connect. The OMP server gives to the user userRomeo an access right after registering the user in the account table as the owner of loaded PLD. At this time, a user can register only one PLD in per target server. When allowing the registered P_0_Romeo shared with other user whose account is userPeter, the hdb_joint command is used. Here, the subject who gives an access right is the userRomeo who is the owner of database P_0_Romeo. The user userPeter is registered as a generic user and is allowed to share the database P_0_Romeo with its owner userRomeo. The user userPeter, who is given an access right of database, only needs running his execution block. Here, all the database manipulation requests in the execution block shares the database P_0_Romeo registered in the server to be shared with other users with sharing rights. When the user userPeter has completed accessing database, the fact of completion is informed the owner of the P_0_Romeo, userRomeo. Then, the owner userRomeo makes the database freed from sharing using the hdb_disjoint command and makes use of the database P_0_Romeo finished using the hdb_disconnect command. These commands first make the user's account which has issued a command registered in the account board. Then, they enable the database relevant to the account shared in the PLD board. When a server receives the requests of EDML embedded in an execution file, it enables those requests to use the database relevant to the now registered account.

For accommodating requests of registration, the account board consists of busy indicators, owner IDs, user IDs, user accounts, requesting process IDs, used PLD name, and PLD board index.

The procedure of executing those commands is as follows. The server in a system first knows that an incoming request is whether for a database manipulation or an administration command. In the case of administration commands, the server executes the commands requested. Then, a user can know the results of the command executions in server, since the status information including faults or normal termination is reported to a user. The executional results of the commands in server are reported to the server's monitor, too.

When the normal termination of commands, the fact is reported to a user and then the procedure is finished.

V. Results of Implementation

The HDBMS server provides the developers of switching system software with the integrated development environment, which is very close to

that of target switching system. It implies that the developers need no longer to make their tests using a real switching system because with the HDBMS it is possible that they can make tests of the software composed of large execution files on the server rather than a real switching system. We can say that, for development of switching system software, the HDBMS provides developers with more efficient testing environment than a switching system itself. Moreover, not only it has become possible that a user can make various kinds of tests on hosts in collaboration with other multiple users, but also the HDBMS has the possibility that it can be applied to developing a dynamic transaction processing system for DBMS. It will also become possible in the future to provide developers with rather comprehensive and convenient testing environment because the confirmation in data transition process can be viable by means of interacting with the IQM(Interactive Query Manager) which has the database manipulation functions under the windows environment.

VI. Conclusions

In this paper, we have made an introduction to the HDBMS simulation system, which is close to the DREAM-S which is a real-time distributed DBMS system running on the ETRI-developed ATM switching system. The developed simulation system can be used in order to make host-based tests of the embedded EDML functions belonging to the execution file for application software. The simulation system named HDBMS is a host-based integrated development environment. In this paper, we described the specific roles and functions of the HDBMS, and then the overall structure of HDBMS and its operational mechanism as well. In turn, we explained in detail the issue of system implementation which includes system organization and messaging mechanism and primitives and administration commands. The new developed host-based simulator can provide the DBMS simulation environment where multiple users can access multiple databases simultaneously. The system has also the powerful expansiveness to allow it enriched, which means it continues having been developing. The further studies in the future will touch on upgraded implementations and discuss the usability and expandability of the proposed system here.

참고문헌

[1] Andrew S. Tanenbaum, "Distributed Operating Systems", Prentice Hall, pp.223~226, 1995

[2] Y. B. Kim, "An Architecture of Scalable ATM Switching and It's Call Processing Capacity Estimation", ETRI Journal Vol.18, number 3, pp. 107~125, Oct. 1996

[3] Y. I. Yoon, S. S. Lee, Y. H. Kim, and J. H. Jo, "A Study on the Design of Asynchronous Transaction Processing in Realtime Distributed Database System" Journal of SIG on Databases, Korea Information Science Society, Vol.11, No.4, pp. 38~50 Oct. 1995

[4] Don-Gil Lee, Joon-Kyung Lee, Wan Choi, Byung-Sun Lee, and Chi-moon Han, "A New Integrated Software Development Environment Based on SDL, MSC, and CHILL for Large scale Switching Systems", ETRI Journal Vol.18, number 4, Jan. 1997.

[5] Young Ik Yoon, Yoo Mi Park, Mi Kyong Han, Seung Sun Lee, Young Ho Park, Ju Hyen Cho, and Chi Moon Han, "DREAM S: Distributed Real Time Database Management Systems for Switching Systems," APAITT '97, pp. 2.7.1~2.7.5, Mar. 13th, 1997.

[6] Young Ho Park and Yong Ik Yoon, "Techniques Managing the Variety Transactions in Switching Systems", Proceedings of International Conference

on Telecommunication(ICT '97), Vol.3, pp. 1271~1276, Apr. 2nd, 1997.
[7] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", John Wiley & Sons Inc., pp. 201~206, Sixth Edition, 2003

저 자 소 개

박 영호

2005년 한국과학기술원(KAIST) 전산학과(공학 박사)
1993년-1999년 한국자통신연구원 (ETRI) 선임연구원
2006년 9월 현재 숙명여자대학교 멀티미디어과학과 교수

이 호

2002년 성균관 대학교 대학원 정보공학과(공학 박사)
1982년~1991년 한국전자통신연구원 (ETRI) 선임연구원
2006년 9월 현재 국립 한국재활복지대학 컴퓨터정보보안과 교수