NATIONAL INSTITUTE FOR FUSION SCIENCE

Object-Oriented Design for LHD Data Acquisition Using Client-Server Model

M. Kojima, H. Nakanishi and S.Hidekuma

(Received - Nov. 11, 1997)

NIFS-TECH-6 Nov. 1997

Object-Oriented Design for LHD Data Acquisition Using Client-Server Model

Kojima M., Nakanishi H., and Hidekuma S.

National Institute for Fusion Science, Japan

LHD data acquisition system processes a huge amount of data exceeding over 600MB per shot. Fully distributed processing and the object-oriented system design are the main principles of this system. Basic elements of the LHD data acquisition system are as follows;

1. standard CAMAC system for digitizing
2. SCSI optical extender for data transfer
3. Windows NT servers for distributed data-acquisition
4. object-sharing method for cross-referencing data through the network
5. object-oriented database for data management
6. harddisk array (RAID) and magneto-optical (MO) disk jukebox for storage.

This system's wide flexibility has been realized by introducing the object-oriented method toward the data processing, in which the object-sharing and the object class libraries will provide the unified way of handling data. It is quite effective especially for the development of data processing softwares, because it enables both servers and clients to share the same procedures and data types compounds, i.e. class ,simultaneously. Object class libraries are written in Visual C++ under MFC, and network object-sharing is provided through a commercial middleware called HARNESS. As for the CAMAC parameter setup, the Java script can share the C++ class libraries and thus makes it possible on WWW homepage by relating with both the object-oriented database and the WWW server.

In LHD experiment, the CAMAC system and the Windows NT operating system are applied for digitizing and acquiring data, respectively. For the purpose of the LHD data acquistion system, a new CAMAC handling software which works on Windows NT has been developed to manipulate the SCSI-connected crate controllers.

The relationship between CAMAC-related processes are based on the client-server model in LHD. The CAMAC list sequencer runs as a server which executes the camac command lists sequentially by communicating with the corresponding driver, and any data managing process can send multiple requests arbitrarily to the server as a client.

In LHD data acquisition, CAMAC lists and diagnostic data classes are shared between these clients and servers. A lump of diagnostic data mass is managed as an object by object-oriented programming.

Keywords:

object-oriented method, LHD, data acquisition, client/server model, CAMAC, SCSI, WindowsNT, WWW, C++

OBJECT-ORIENTED DESIGN FOR LHD DATA ACQUISITION USING CLIENT-SERVER MODEL

KOJIMA M., NAKANISHI H., HIDEKUMA S.
National Institute for Fusion Science,
Toki 509-52, Japan

ABSTRACT. The LHD data acquisition system handles a huge amount of data exceeding over 600MB per shot. The fully distributed processing and the object-oriented system design are the main principles of this system. Its wide flexibility has been realized by introducing the object-oriented method into the data processing, in which the object-sharing and the class libraries will provide the unified way of data handling for both servers and clients program developments. The object class libraries are written in C++, and the network object-sharing is provided through a commercial software called HARNESS. As for the CAMAC setup, the Java script can use the C++ class libraries and thus establishes the relationship between the object-oriented database and the WWW server. In LHD experiments, the CAMAC system and the Windows NT operating system are applied for digitizing and acquiring data, respectively. For the purpose of the LHD data acquisition, the new CAMAC handling softwares which work on Windows NT have been developed to manipulate the SCSI-connected crate controllers. The CAMAC command lists and diagnostic data classes are shared between clients and servers. A lump of diagnostic data mass is treated as a part of an object by the object-oriented programming.

1. INTRODUCTION

The Large Helical Device (LHD) project is in the final phase of the construction, and it will be planed to start the plasma discharge experiments in March 1998 [1]. The LHD data acquisition and analysis system which handles over 600 MB per discharge is being developed [2]. We chose the fully distributed system which consist of multiple high performance computers with the first network, utilizing the recent progress in the personal computer and the network technology [3].

Since about 30 kinds of diagnostics will be applied in LHD experiments, we have to treat various kinds of data. To acquire, analyze, display and store such various kinds of data, we adopted the object-oriented method to reduce the burden for the development of the softwares. In this style, each lump of the experimental data is treated as a part of an independent object including the manipulation functions for itself. In order to develop distributed computing softwares with the object-oriented method, we used the C++ programming language and the software package called HARNESS. C++ is suitable for the object-oriented method. Moreover, we adopt the object-oriented database O2 along with the client-server model [4]. In LHD experiments, the CAMAC system and the Windows NT operating system are applied for the data acquisition. However, CAMAC control software on Windows NT for the SCSI-connected crate controllers did not exist in 1996. Therefore we have developed the new CAMAC handling softwares based on the client-server model.

The software development of the LHD data processing system by using the object-oriented method is described in Section 2. The virtual shared memory for distributed computing realized by HARNESS is explained in Section 3. And the utilization of the object-oriented database is introduced in Section 4. Moreover the CAMAC handling software which is newly developed is mentioned in Section 5. The summary is made in Section 6.

2. SOFTWARE DEVELOPMENT USING OBJECT-ORIENTED METHOD

We introduce the object-oriented method in order to deal with various kinds of experimental data [5][6][7]. The standard class for the experimental data is defined as a compound which consists of;

1. variables, such as setup parameters,
2. data arrays,
3. data transforming methods.

The object of the experimental data is created according to this class definition. . The advantages of this object-oriented style are;

1. We need not recognize or manage the internal data structure, but just send a message to operate/retrieve data.

2. The class library can be shared anywhere, where we can handle the objects in the same manner.

3. Objects can protect and conceal its internal data structures and variables completely, whose class definitions are easy to change or maintain.

These advantages lead to a good software portability and an easy programming. The object-oriented method is very useful to make the complicated LHD data processing system. The important point for the object-oriented method is that it can deal with experimental data as a part of an object. The object-sharing and the class libraries will provide the unified way of data handling for both servers and clients program developments. As a result, the development of the complicated distributed system along with the client-server model becomes more easier. Overview of the object structure in the LHD data acquisition system is shown in Fig. 1. All of the information handling in the object-oriented method are implemented as the message exchange between objects.

3. OBJECTS SHARING BY VIRTUAL SHARED MEMORY ON NETWORK

About 30 sets of the data acquisition server computers are connected to the network, and each unit takes care of the individual diagnostic device. By the data acquisition LAN, they are linked mutually and share the virtual memory named HARNESS [8]. It makes the virtual shared memory over the network, and the shared memory space is called as "HARNESS space". The computers on the network can share the objects on memory by HARNESS. A schematic view of the virtual shared memory by HARNESS is shown in Fig. 2. The objects of experimental data which exist on each computer's memory can be shared over network. This is one of the most important properties of the LHD data processing system. Since the HARNESS functions are available as a class library, it can directly treat the objects. Using HARNESS, users do not have to know what network protocol is used or where the server computer is. The HARNESS space makes the program simpler because only its space name is required for processes to communicate with each other. HARNESS has the function of the automatic timing synchronization to refer the data, which is necessary for the distributed computing. For example, when the server and the client processes want to communicate each other, the server process "write" an object in the HARNESS space, and the client process "read" it from the same space.

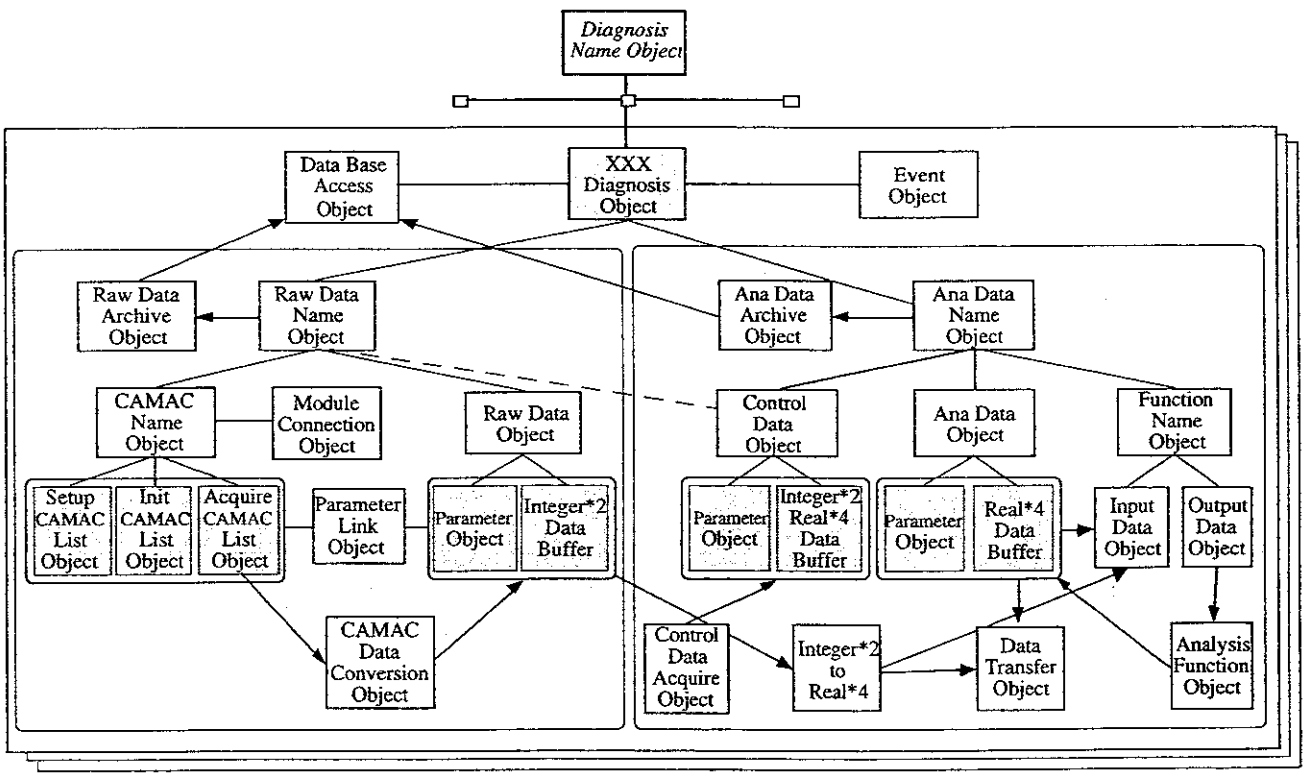

Figure 1: Overview of the object structure on the LHD data acquisition system.

Overview of the client-server communication using the HARNESS space is shown in Fig. 3.

The object-oriented data analysis is realized by HARNESS. The objects of data analysis start its calculation when all the required raw data objects have been prepared. That is to say, the reference timing for required objects are synchronized with their appearance. In case that an analysis like a plasma equilibrium calculation requires the analyzed data from other diagnostics, their data objects are taken from the HARNESS space. After the data analysis object has completed the calculation by itself, the created objects are written to the HARNESS space. Since the start of the analysis depends only on the completion of the preparation of all required objects, we do not need the complicated calculation scheduling mechanism for the sequence of the analysis. When the secondary analyzed data are calculated from the plural kinds of the primary analyzed data, the preparation mechanism of the referred data becomes simpler by using HARNESS.

4. UTILIZATION OF OBJECT-ORIENTED DATABASE

In case of an operating system with a kernel supported the virtual memory, the operating system automatically exchange the data on memory and the hard disk. Because the object-oriented database has more extensive function to exchange the objects between the main memory and the hard disk device, we do not have to be aware of the location of the objects. Since the data structure is consisted in the object, it is necessary to use the object-oriented database for storing and retrieving data.

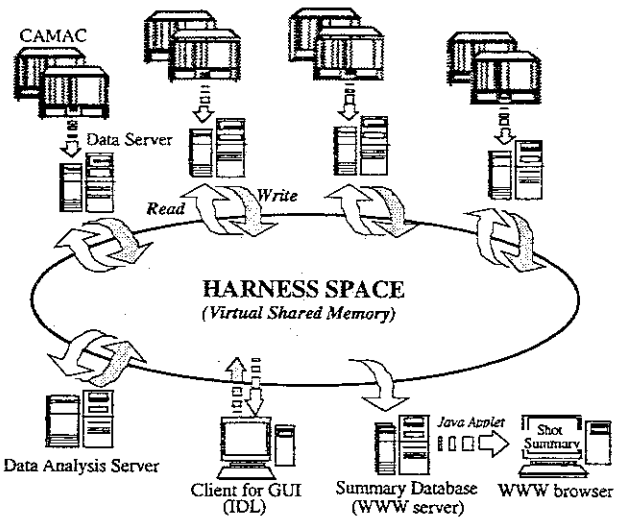

Figure 2: A schematic view of the virtual shared memory by HARNESS.

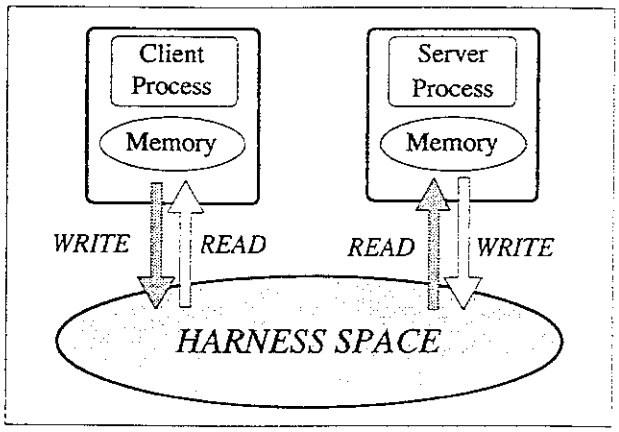

Figure 3: Overview of the client-server communication using the HARNESS space.

The advantages of the object-oriented database for storing objects are as follows [9][10]:

1. The C++ class library for the experimental data can be applied not only to the data acquisition server and the data operation client programs but also to the object-oriented database class definitions just as it is.
2. The objective programming language like the C++ can describe the class definition overlay, and enables to change the program without taking notice to follow its revisions.

The object-oriented database is also used to store the CAMAC module setting or the shot summary data. The clients can access the data stored as the objects on the database through a WWW browser. And the clients communicate with the object-oriented database through the gateway program attached to the HTTP server. The gateway program executes the language and query conversion from URL,OQL to HTML. A block diagram of the communication between the summary database sever and its clients is shown in Fig. 4. The sequence of the client-server communication is [11];

1. the client sends the uniform resource locators (URL) which includes a query written in the object query language (OQL) [12],
2. the HTTP server transfers the query to the database server,
3. the database server computer converts the requested data into hyper text markup language (HTML) format,

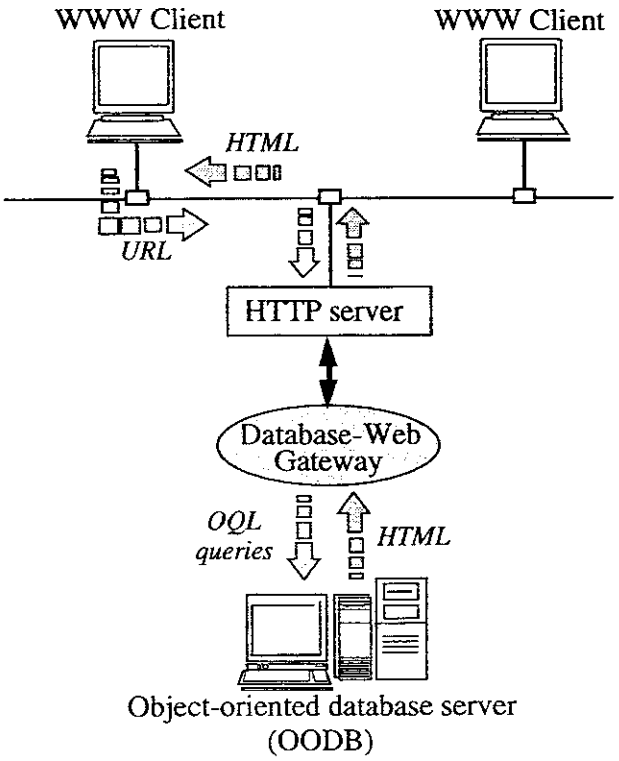

Figure 4: A block diagram of the communication between the summary database sever and the client which displays summary data.

4. the HTML data is sent to the client.

Using a WWW browser, we can access the summary database from anywhere on the internet. The Java applets could be useful to display the data in WWW browser because the displaying program can be executed locally in the browser by downloading from WWW server [13].

5. CAMAC HANDLING SOFTWARES ON Windows NT

We have newly developed the CAMAC handling software [14], in order to manipulate the SCSI-connected crate controller from Windows NT and to acquire the data from CAMAC ADCs [15]. The contents of the CAMAC control software on Windows NT are shown in Fig. 5. This software consists of the following three parts;

1. CAMAC driver: the SCSI class-driver to control SCSI crate controller,
2. CAMAC library: the application programming interface (API) to the CAMAC driver,
3. CAMAC list sequencer: the application program using the CAMAC library which manages the CAMAC command lists.

In Windows NT, the SCSI device driver has a hierarchy of several drivers [16]. The CAMAC driver is an upper driver of the SCSI port driver, and this driver translates the CAMAC commands into SCSI ones. The CAMAC driver can deal with multiple SCSI ports, and it can be used both on Intel Pentium and DEC Alpha processors.

The CAMAC library is the interface between a user program and the CAMAC driver. It is provided as the dynamic link library (DLL) in Windows NT [17]. The function names in this library are almost compatible with KineticSystems CAMAC libraries [18].

The CAMAC list sequencer works as a list-processing server process, which executes CAMAC commands of N, A, F, to control the CAMAC modules. The user application program will hand the CAMAC command lists to the list sequencer, and obtain the data. These CAMAC-related data acquisition programs were designed in accordance with so-called the client-server model. The CAMAC list sequencer runs as a server process which executes the CAMAC command lists sequentially by communicating with the corresponding driver, and any process which wants to access the CAMAC can send it arbitrary requests. The list sequencer is implemented as a background service process of Windows NT.

The list sequencer process will hand the list to the driver by one line after it received a series of command

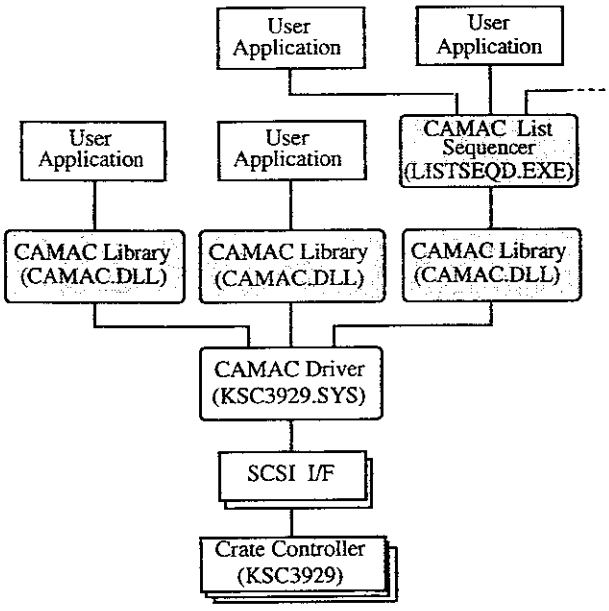

Figure 5: The contents of the CAMAC control software on Windows NT.

lists completely from the client application program. The client application executes the following processes for individual series of command lists;

1. it copies the command lists to the sequencer,
2. readout the data according to the returned status from the sequencer.

The block data transfer between the list sequencer and the application program will be executed efficiently through the double buffering mechanism.

The double buffering mechanism realizes more efficient data transfer by using two buffers because both acquiring data from the CAMAC module and transferring data between the list sequencer and the client application can be executed at the same time. The total throughput of the block data transfer is about 700kB/s between the CAMAC module and the application program. The schematic view around the list sequencer is shown in Fig. 6 and the internal block diagram of the list sequencer is shown in Fig. 7.

The behavior of the list sequencer which handles multiple requests from the plural client processes are as follows:

1. the residential primary thread of the list sequencer process is waiting for connection from a client process through the named pipe.
2. when a client process requests to execute the CAMAC command lists through the named pipe, the primary thread creates the secondary thread for execution of the command list.
3. the secondary thread executes the command list after that, it return the result of it.
4. if a CAMAC command for the block data transfer is included in the command list, the secondary thread transfer the data from the buffer inside of the list sequencer to that of the client process.
5. when another client process requests to execute the different command list simultaneously, the primary thread creates the another secondary thread for execution of the command list independently.

Thus, even if plural client processes request the list can sequencer to execute the command lists simultaneously, it handle the requests by using multi-thread operations.

The client process can obtain data only by describing the necessary CAMAC lists and requesting the sequencer to execute them. The arithmetic calculations of the

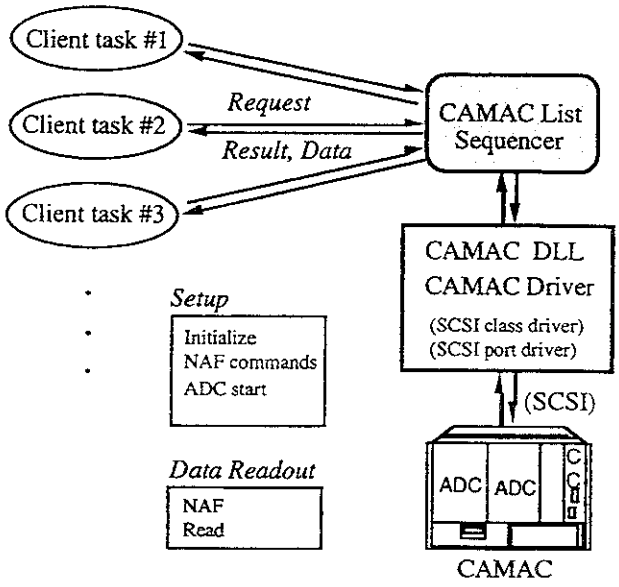

Figure 6: The schematic view around the list sequencer.

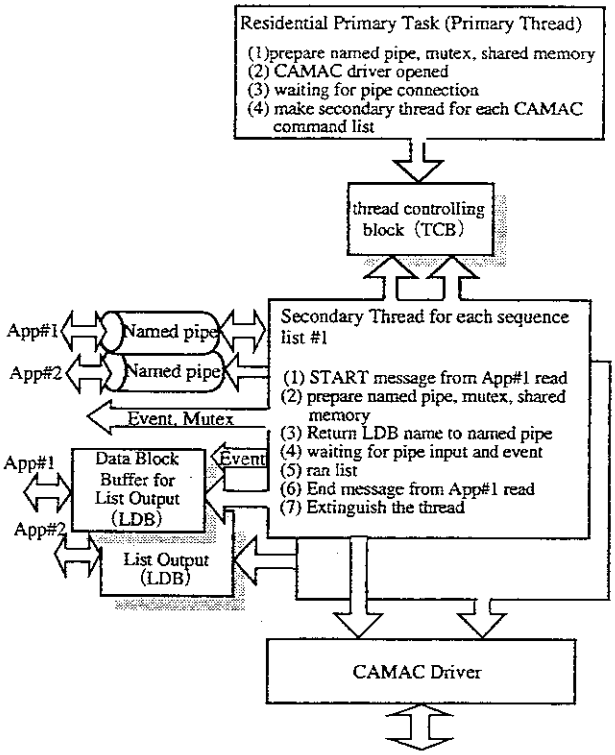

Figure 7: The internal block diagram of the list sequencer.

readout data and the iteration loops due to the CAMAC Q-response can be also carried out by the list sequencer. It can provide the independent operation for any CAMAC modules and crates.

Using this list sequencer, we do not have to describe a

Publication List of NIFS-TECH Series

NIFS-TECH-1 H. Bolt and A. Miyahara, *Runaway–Electron –Materials Interaction Studies*; Mar. 1990

NIFS-TECH-2 S. Tanahashi and S. Yamada, *Dynamic Analysis of Compact Helical System Power Supply and Designs of Its Upgrade;* Sep. 1991

NIFS-TECH-3 J. Fujita, K. Kawahata, S. Okajima, A. Mase, T. Suzuki, R. Kuwano, K. Mizuno, T. Nozokido, J.J.Chang and C.M.Mann, *Development of High Performance Schottky Barrier Diode and its Application to Plasma Diagnostics;* Oct. 1993 (in Japanese)

NIFS-TECH-4 K.V. Khlopenkov, S. Sudo, V.Yu. Sergeev, *Operation of the Lithium Pellet Injector*; May 1996

NIFS-TECH-5 Nakanishi, H., Kojima, M. and Hidekuma, S., *Distributed Processing and Network of Data Acquisition and Diagnostics Control for Large Helical Device (LHD)*; Nov. 1997

NIFS-TECH-6 Kojima, M., Nakanishi, H. and Hidekuma, S., *Object-Oriented Design for LHD Data Acquisition Using Client-Server Model*; Nov. 1997