Autonomous Decentralized File System and Its Application

Masayuki Orimo*, Shigeki Hirasawa*, Hiroshi Fujise**,
Masuyuki Takeuchi** and Kinji Mori*

*Systems Development Laboratory, Hitachi, Ltd., 1099 Ohzenji, Asao, Kawasaki 215, Japan
**Software Development Center, Hitachi, Ltd., 5030 Totuka, Totuka, Yokohama 244, Japan

Abstract

From the standpoint of system construction /expansion, it is required that a computer system is flexible and tolerant for change of the system structure and has software productivity. And a computer system is also required to be easily operated by users regardless of its structure and status. In order to achieve these requirements, the EVERUN (EVEr RUNning) computer system has been developed based on the autonomous decentralized system concept. In this paper, the distributed file system for this EVERUN computer system is proposed. This file system realizes flexible fault-tolerance, and easy expansion, operation and maintenance. The EVERUN computer system including this file system has been applied to production management systems and so on. And the effectiveness of this computer system has been verified.

1. Introduction

A distributed computer system(DCS) has been developed rapidly based on the price-performance revolution in microelectronics and the advancement of communication techniques. A DCS potentially has significant advantages, including good performance, good reliability, good resource sharing, and good expandability[1].

Recently operating systems for a DCS have been widely discussed and developed[2][3][4]. Most of these systems are experimental systems and mainly aim at attaining high system performance and resource sharing. But the widespread use of a DCS has made it become required that a DCS can continue its operation even if it partially fails and/or recovers and the structure of it changes. This means that a DCS has been required to attain fault-tolerance, on-line expandability and on-line maintainability.

The autonomous decentralized software system architecture has been proposed in order to meet these requirements[5]. This architecture has the feature that every software subsystem has autonomy to manage itself and coordinate with the other software subsystems. This coordination is attained by communicating with the other software subsystems through the Data Field(DF), in which the data circulates, and the computer unit judges whether to receive the data or not on the basis of the content of the message. In this system, there exists no specific manager and no master/slave relation among the software subsystems.

On the other hand, in actual systems, from the standpoint of the users who operate the computer system, it has been required that they can easily manage the system even if they are non-specialists of computers. And this requirements for easy operation of the system has become significant more and more especially for a DCS.

The EVERUN(EVEr RUNning) computer system has been developed in order to meet the requirements for fault-tolerance, on-line expandability, on-line maintainability and ease-to operate based on the autonomous decentralized software system architecture. In this paper, the distributed file system for this EVERUN computer system is proposed. This file system is called the EVERUN file system. This realizes the file access from the application software module independent of the system configuration. And the effectiveness of the EVERUN computer system including the proposed file system is shown through the application to the production management system.

2. System Requirements

Due to the widespread use of a DCS, a DCS has been required to meet the following three requirements.

(1) Flexible fault-tolerance

As computing systems play a larger role in everyday life, we face a rapidly growing need for fault-tolerant computing systems. Several commercial fault-tolerant computers have been developed[6]. These computers attain fault-tolerance by replicating their components such as

0-8186-2755-7/92 $03.00 © 1992 IEEE

processors, disks and input/output devices, or by replicating entire computer. These fault-tolerant computers have been applied to large scale system whose stop causes a social problem or a great deal of loss of benefit such as stock information systems. These computers aim at fault-tolerance of computer itself.

Today, computerization has been achieved not only for a large scale system but also for a medium or a small scale system. In a medium or a small scale system, not the fault-tolerance of computer itself but the fault-tolerance of important application programs which run in a computer is required. The degree of required fault-tolerance is different according to the importance of each application program. The achievement of this different level of fault-tolerance for each application program has been required. We call this flexible fault-tolerance.

(2) Ease to construct, operate and maintain

Because of the rapid change of economic situation, it has become difficult to determine the total system configuration at first. And it is desirable that the system configuration can be easily changed according to the change of economic status. That is, easy and quick construction and expansion are required. Of course, easiness is required for not only construction but also for operation and maintenance.

(3) Software Productivity

High productivity of software for a DCS has been required. And utilization of the conventional software in a DCS environment has been also required. This means that the special software for a DCS is unacceptable to software engineers.

3. Autonomous Decentralized Software System Architecture

3.1 Autonomous Decentralized System Concept

The autonomous decentralized system concept has been proposed to attain fault-tolerance, on-line expandability and on-line maintainability of not only the hardware system but also the software system[7].

This concept, based on biological analogy, has the perspective that a system almost always has faulty parts and undergoes modifications. That is, a total system cannot be previously defined. A system is defined as the result of the integration of subsystems. A system is called as autonomous decentralized system, if the following two properties are satisfied.

(1) Autonomous controllability

If any subsystem fails, the other subsystems can continue to manage themselves.

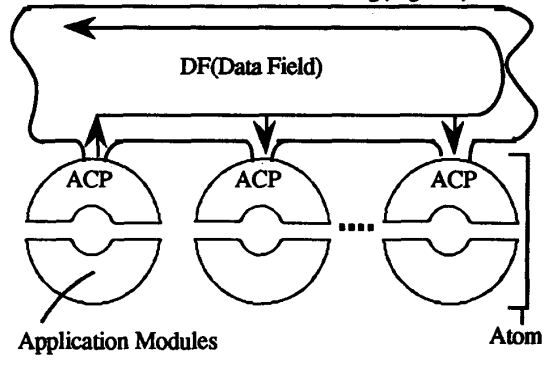
(2) Autonomous coordinability

If any subsystem fails, the other subsystems can coordinate their individual objectives among themselves.

The autonomous decentralized software system architecture has been proposed based on this concept.

3.2 Software System Architecture

The basic feature introduced in the autonomous decentralized software system architecture is the Data Field(DF) and the Atom(Fig. 3.1). The Atom is the computer unit which has its own management system software called ACP(Autonomous Control Processor) for attaining its autonomy. Each Atom coordinate with other Atoms through the DF. The DF is the field where the data circulates and physically is the local area network(LAN) connecting each computer unit. In the DF, each data has a content code which indicates its meaning(Fig. 3.2).



ACP:Autonomous Control Processor

Fig. 3.1 Autonomous Decentralized Software Structure

F	CC	SA	C	Data	CRC	F

F : Flag, CC : Content Code, SA : Sender Address
C : Control Code, CRC : Cyclic Redundancy Check

Fig. 3.2 Message Format

This feature means that the Atom broadcasts all data with their content codes into the DF and it judges whether or not to receive the data from the DF on the basis of their content codes. Each Atom independently judges what data in the DF to accept and how to process them. And each Atom broadcasts every processed result data into the DF without knowing how to be processed or which Atoms to

receive. This architecture makes it possible for every Atom to control itself and coordinate with the others without having global information on the entire system but having only local information on the content codes necessary for the software modules in the Atom itself. The content code protocol is not designed for the sender and receiver Atoms but for the data themselves.

4. EVERUN File System

In order to attain flexible fault-tolerance, ease-to construct/operate/maintain, and software productivity of a file system in a DCS, the EVERUN file system is proposed based on the autonomous decentralized software system architecture.

4.1 ACP Structure

The ACP consists of several management modules necessary for executing the application programs under the autonomous decentralized software system environment. The basic management module of the ACP is the DF management module which provides the interface with the DF. The other management modules communicate with each other based on the DF interface through this DF management module. That is, each management module of the ACP can communicate with other management modules without knowing whether these modules are located in itself or in other computer units.

The basic management modules for the EVERUN file system are the file client management module and the file server management module. The file client management module is the module for managing application programs which utilize the EVERUN file system. And the file server management module is the module for managing the file access process.

The EVERUN file functions are achieved by the coordination among the above management modules. The functions of these three management modules are shown below.

4.2 Functions of DF Management Module

The DF management module has the following three functions for achieving the message transmission between the DF and the other management modules.

(1) Registration of content codes

Each management module registers the content codes of the messages which are necessary for itself to the DF management module. And each management module cancels the content code which has been registered by itself when the message with this content code becomes unnecessary for itself. This registration and cancellation of content codes can be done at any time without stopping the operation of the computer unit. So, each management module can receive any message on the DF when the message becomes necessary to itself. And if the new module is added to the computer unit, this new module can start its process at the time when it registers the content codes to the DF management module .

(2) Receiving messages from the DF

The DF management module judges whether or not the received message from the DF is necessary for the other management modules in its computer unit on the basis of the content code of the received message and the content codes registered in itself. Then the DF management module delivers the received message to the corresponding management module.

(3) Sending messages to the DF

The DF management module broadcasts the messages, which are requested to send by the other management modules, to the DF. The DF management module also stores these messages into its receiving buffer. These messages in the receiving buffer are processed in the same way as the messages received from the DF.

4.3 Functions of EVERUN file System

The following functions are achieved by the coordination between the file client management module and the file server management module.

4.3.1 Functions for Flexible Fault-tolerance

In the EVERUN file system, each file can be replicated according to the degree of its importance in any disk device of any computer unit connected to the DF. We call this file the EVERUN file.

In conventional systems which supports the replication of the disk device, all of the files in the disk device must be replicated. But in the EVERUN file system, only the important file can be replicated according to the degree of fault-tolerance required to it(Fig. 4.1). Therefore, it is possible to efficiently utilize the disk resources in a DCS. In order to achieve this flexible fault-tolerance, the following two functions are necessary.

(1) Flexible communication between clients and servers

In this paper, the computer unit in which application programs run is called the client computer unit and the computer unit which fields requests for the file access from application programs is called the server computer unit. The client computer unit and the server computer unit may

be a same computer unit. In the EVERUN file system, both of the client computer units and the server computer units are connected to the DF. Therefore, there is no fixed relation between any client computer unit and any server computer unit.

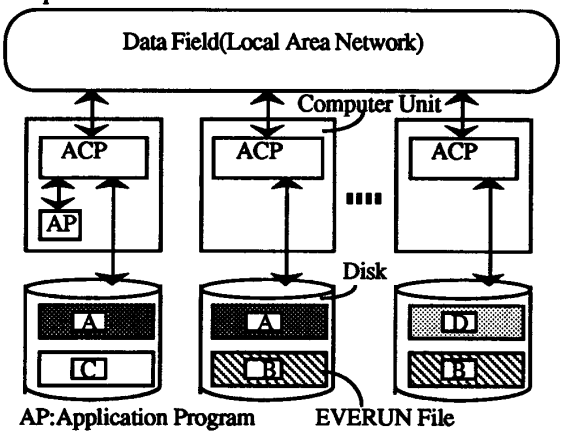


Fig. 4.1 Flexible Fault-Tolerance of EVERUN File System

A client computer unit need not know where the server computer unit managing the file to be accessed is located and whether the file is replicated or not. For the communication with server computer units, the file client management module of the client computer unit executes the following steps.

[step 1] Broadcasting the file access message with the content code indicating the content of the file to be accessed to the DF by way of the DF management module.

[step 2] Receiving the message which has the content code indicating the content of the file access result from the DF by way of the DF management module. This content code is registered to the DF management module when the file access message is broadcasted.

The file server management module of the sever computer unit executes the following steps.

[step 1] Receiving the message which has the content code corresponding to the EVERUN file belonged to its computer unit. This content code is registered to the DF management module when the EVERUN file corresponding to the files of the server computer unit was defined. This definition process is explained in the next section.

[step 2] Fielding message received in the step 1 and broadcasting the access result message with the content code indicating the content of the access result.

(2) Data consistency check function

In the EVERUN file system, each file can be replicated according to the degree of its importance. That is, the EVERUN file may consist of multiple replicated files. We call these files the physical replicated files. All of these physical replicated files are logically connected to the DF and there is no master/slave relation among these physical replicated files. Each file server management module for each physical replicated file independently receives the message from the DF, fields the received message(file access) and broadcasts the access result message to the DF. Therefore, the multiple access result messages are asynchronously broadcasted to the DF by the multiple file server management modules. In order to manage these multiple access result messages, the file client management module has the data consistency check function described bellow. This function makes it possible for application programs to treat the EVERUN file as one file, that is, makes the physical replicated files invisible to application programs.

The file client management module selects one of the multiple access result messages received from the DF. This selection is done by using the firstly received access result message and neglecting the other access result messages received after. This selection logic assures the fast response time of the EVERUN file access. And by this selection logic, application programs can continue to access the EVERUN file as long as any one of the physical replicated files survives.

There is the possibility that messages on the LAN are lost by transient external causes such as noise. Therefore, the inconsistency among the physical replicated files is possible to occur even if there is no hardware fault in the system. The possibility of the occurrence of this inconsistency is low, but this is also checked by the data consistency check function. The file client management module checks whether or not the consistency among the physical replicated files is assured when the access result message is received. If the inconsistency is detected, the file access process which caused this inconsistency is canceled and retried by the cooperation between the file client management module and the file server management modules. The quick file access response and the consistency among the physical replicated files are assured by this data consistency check function.

4.3.2 Functions for Ease-to Construct, Operate and Maintain

In a DCS, it is desirable that users can use the system

without knowing the configuration of the system. In the EVERUN file system, users can access any EVERUN file without knowing which computer unit the file is located in and whether or not the file is replicated. In order to attain easy definition, operation and maintenance of the EVERUN file, the following functions are supported in the EVERUN file system.

(1) Definition of EVERUN file

Each EVERUN file has its own logical name which is independent to the configuration of the system. And application programs access an EVERUN file by using this logical name. The definition of the EVERUN file is to define the logical name of the EVERUN file and the physical files corresponding to the EVERUN file. Multiple physical files can be assigned to each EVERUN file according to the degree of the replication of the EVERUN file.

This definition can be done at any computer unit which is connected to the DF. The information defined by this definition process is broadcasted to the DF. And the computer units which have the corresponding physical file receive this message and register the content code for this EVERUN file to the DF management module in themselves. And this information is stored in their disks. After this definition process, the above computer units act as the server computer units for the EVERUN file defined above. In the EVERUN file system, each server computer unit need not know the information about the physical files of other computer units. Therefore, if the configuration of the physical files in the system changes such as addition of new EVERUN files, there is no need to change the definition information which has been stored in each server computer unit.

(2) Monitoring of the status of EVERUN files

From the standpoint of maintenance, it is desirable to be able to recognize the location and the status of the physical files at any computer unit. In order to attain this, the EVERUN monitor has been developed. This monitor consists of two parts. One is the BIT(Built-In Tester) implemented in the file server management module and the file client management module in each computer unit. The other is the EXT(EXternal Tester) which can run in any computer unit as an application program. By the coordination of the BIT's and the EXT through the DF, the following two functions are attained.

(a) System structuring

The configuration of EVERUN files in the system has not been previously determined since each EVERUN file can be defined at any time without stopping the operation of the system. Each BIT in the file server management module broadcasts the message containing the information about the physical files managed by itself to the DF in response to the EXT's message. The EXT gathers these messages from the BIT's, and determines the EVERUN file structure in the system by integrating the content of the gathered messages.

(b) Fault detection

Each BIT in the file server management module detects the fault of the files managed by itself. Each BIT in the file client management module detects the fault of the server computer units. The BIT detecting the above faults broadcasts the message indicating this fault to the DF. The EXT receives this message and displays this information to the terminal of the computer unit in which the EXT runs.

(3) Recovering of the failed file

In the EVERUN computer system, application programs can continue to access the EVERUN file as long as one of the physical replicated files corresponding to the EVERUN file survives. This means that the partial fault of the EVERUN file may cause the inconsistency among the physical replicated files. Therefore, it is necessary to check whether or not the inconsistency between the failed physical replicated file and the other normal physical replicated file has been occurred when the failed file is repaired. In order to attain this, the following file recovering functions has been developed.

When the failed file is added to the system after the completion of its repair, the file server management module of this file broadcasts the message indicating the recover of the failed physical replicated file to the DF. This message is received by the file server management modules of the other physical replicated files composing the EVERUN file with the repaired file. Each file server management module checks whether or not the inconsistency between the repaired file and its own file exists. And when the inconsistency is detected, the content of the normal file is copied to the repaired file automatically.

4.3.3 Functions for Software Productivity

The EVERUN file system appear to users as a single sub-tree of the local file system on each computer unit. That is, users can treat EVERUN files as the same way of local files. Both of the file client management module and the file server management module are invisible to an application program, which only sees the local file system of the computer unit in which the application program

runs. One sub-tree of the local file system is identical on all computer units connected to the DF. This sub-tree is the directory for EVERUN files(EVERUN directory). Therefore, there is no need to make a special application program in order to use EVERUN files.

In order to attain the above file system view at each computer unit, the file client management module has the next functions. The file client management module checks whether or not the file which is requested to access by an application program is the file under the EVERUN directory. And in the case of the file under the EVERUN directory, the file client management module generates the message for the EVERUN file access and broadcasts the message to the DF by way of the DF management module.

4.4 Installation of the EVERUN file system to Hitachi Computer

The EVERUN file system has been installed to Hitachi's small size computer, L-700 series computer. L-700series computer consists of five processor groups, L-750, L-760, L-770, L-780 and L-790 processor groups.

L-700 series computer has Hitachi's proprietary operating system MIOS7/AS(Multiple office Information Operating System/Advanced System). The DF management module, the file client management module and the file server management module are installed on this MIOS7/AS. The DF management module is realized as a kernel part of the MIOS7/AS.

5. Application

One of the applications of the EVERUN file system is a production management system for a medium scale production system. This system is explained below.

The computerization of production process management has required not only for a large scale production system but also for a small or a medium scale production system. The production management system composed of small and medium size computers has been developed for management of the medium scale production process system from the standpoint of cost reduction and easy system construction. Fault-tolerance is also required to this medium scale management system, since the stop of the system causes stop of production. In the case of this management system, even if the system partially fails, it is required to continue the execution of important programs such as the production process tracking program which are indispensable to the management of the production process. That is, realization of flexible fault-tolerance by effectively utilizing the computer resources is required.

In this system, there are three major subsystems for on-line process control, on-line information management and software development(Fig. 5.1). The on-line process control subsystem consists of several personal computers. The functions of on-line information management subsystem and software development subsystem are attained by two L-700 series computers, respectively. The

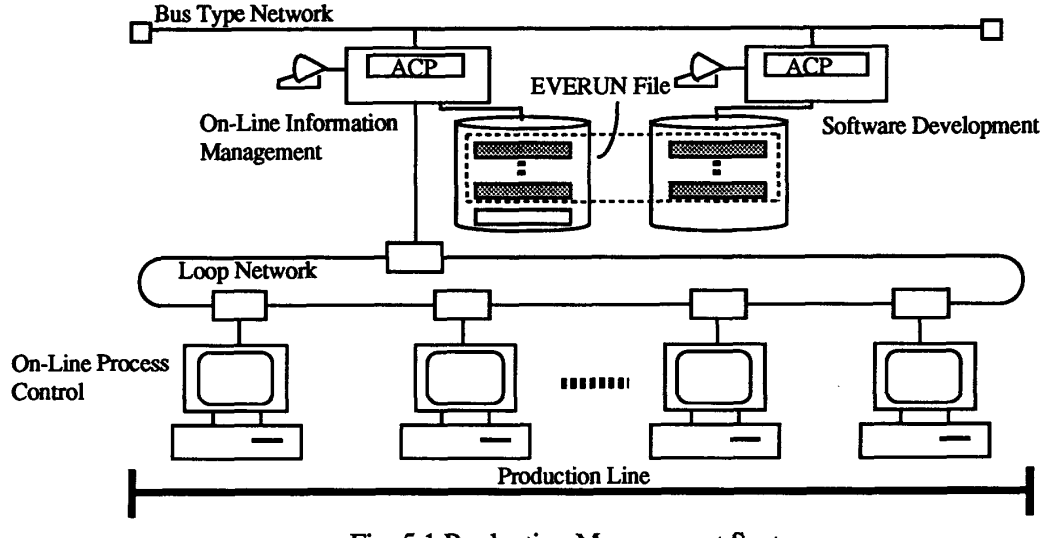


Fig. 5.1 Production Management System

on-line process control computers and the on-line information management computer are connected by the optical loop network. The on-line information management computer and the software development computer are connected by the bus type network.

For the production process management of this system, execution of the production line management function in the on-line information management computer is indispensable. And in order to execute this function, the production schedule data file, the work in process data file, the process tracking data file and the production result data file are necessary. Therefore, these four files are duplicated in the disk devices of the on-line information management computer and the software development computer as the EVERUN files. That is, fault-tolerance of the production management function of this on-line information management computer is attained by effectively utilizing the resources of the software development computer.

The application of the EVERUN file function to this system has been attained without any additional software development. There was no need to develop special programs in order to use the EVERUN file function. And the definition of the EVERUN file, that is, the definition of the degree of replication for each file, could be attained without stopping the system operation.

6. Conclusions

The EVERUN file system has been developed based on the autonomous decentralized software system architecture in order to attain flexible fault-tolerance, ease-to construct, operate and maintain, and software productivity. In the EVERUN file system, the client computer unit and the server computer unit coordinate with each other through the DF in which the data circulates. Each computer unit can act as either of a client computer unit or a server computer unit. Each server computer unit independently receive the messages necessary for themselves from the DF and broadcasts the messages containing their processing result to the DF. Each file can be replicated according to the degree of its importance. The EVERUN file functions are attained by the DF management module, the file client management module and the file server management module. The DF management module provides the DF interface. The file client management module is the module for managing application programs. The file server management module is the module for managing the file access process. These two file management modules coordinate with each other through the DF management module.

The EVERUN file system has been installed to Hitachi L-700 series computer. This computer with the EVERUN file system has been applied to the production management system, and its effectiveness has been verified.

Acknowledgements

We would like to thank Mr. Singi Domen, general manager of Systems Development Laboratory, Hitachi Ltd. for giving us the opportunity for this research.

References

1) J. A. Stankovic, "A Perspective on Distributed Computer Systems," IEEE Trans. on Computer, vol. C-33, no. 12, 1102-1115, Dec. 1984

2) S. J. Mullender, et al., "Amoeba: A Distributed Operating System for the 1990s," IEEE Computer, vol. 23, no. 5, 44-53, May 1990

3) M. Satyanarayanan, "Scalable, Secure, and Highly Available distributed File Access," IEEE Computer, vol. 23, no. 5, 9-22, May 1990

4) G. A. Champie, et al., "Project Athena as a Distributed Computer System." IEEE Computer, vol. 23., no. 9, Sept. 1990

5) K. Mori, et al., "Autonomous Decentralized Software Structure and its Application," Proc. of FJCC, 1056-1063, 1986

6) D. P. Siewiorek, "Fault Tolerance in Commercial Computers," IEEE Computer,vol. 23, no. 7, 26-38, July 1990

7) H. Ihara and K. Mori, "Autonomous Decentralized Computer Control Systems," IEEE Computer, vol. 17, no. 8, 57-66, Aug., 1984