Symfoware Server—Reliable and Scalable Data Management

● Kazunori Takata ● Masaki Nishigaki ● Katsumi Matsumoto

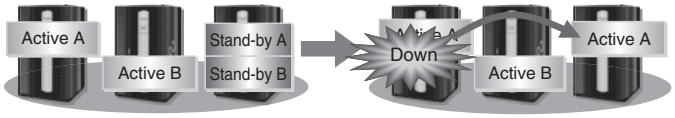
In recent years, the scale of corporate information systems has continued to expand and the amount of data and transactions to be processed has been drastically increasing. Nevertheless, these systems still need to ensure a high level of processing performance, reliability and scalability. To achieve this, Fujitsu has developed a database server fallback function and application-connection shifting function. These functions are used not only to handle potential faults but also to accommodate business growth, and are offered through Fujitsu's Symfoware Server high-reliability high-performance database software. These approaches have propelled the development of parallel-processing and cluster technologies. This paper describes Fujitsu's approach to steady database operation with the focus on a database server fallback function to achieve high performance, high reliability and good scalability in mega-scale load-sharing systems using multiple database servers.

1. Introduction

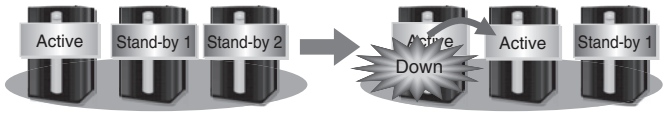
Because current ICT systems are closely related to corporate and social activities, they need to operate in a reliable way. Unexpected system shutdowns are having a greater impact than ever before. This may affect not only corporate business processes but also social infrastructure services such as financial, community, medical, logistics and communication services. Therefore, the majority of these companies consider it essential to be able to continue their business without suspending services even in the event of a fault in their systems such as a server fault.

Against this background, in developing Symfoware Server (hereafter, Symfoware), a high-reliability, high-performance database, Fujitsu has placed emphasis on the ability to continue business through steady system operations. In mission-critical systems in particular, there are needs for high-quality, continued operations and stable performance of a system. To meet these requirements, we have made substantial progress in various mission-critical systems by applying our technologies nurtured since the age of mainframes to open systems in an improved form. As a high-reliability, high-performance database for mission-critical systems, the functions of Symfoware have been enhanced. Based on its proven performance, Symfoware has been adopted in "arrowhead," the latest stock trading system of the world's highest level used by the Tokyo Stock Exchange.

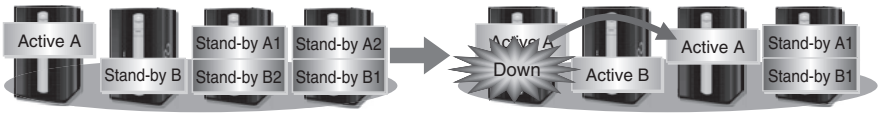
In this paper, first we will describe our approaches to date to secure stable operations. Then, we will introduce the "database server fallback function" which is used to address increasing transactions in mega-scale systems, high-speed access from application programs and scalability of systems. At the end, we will further describe how the aforementioned approaches are deployed in small and medium-scale systems as well.



(a) *N* vs. 1 stand-by type: Operation mode comprised of one stand-by node versus multiple active nodes



(b) Cascade type: Operation mode comprised of multiple stand-by nodes versus one active node



(c) *N* vs. *M* stand-by type: Operation mode comprised of multiple stand-by nodes versus multiple active nodes

Figure 1
Operation modes supported by Symfoware.

2. Approaches to stable operations to date

Within the Symfoware project, we have tried to reduce the duration from fault occurrence to the resumption of business process to ensure steady system operations even if various types of faults occur. In our approaches so far, we have sought to develop high-speed switch technologies to address unexpected system shutdowns. These technologies such as the prior start-up of a stand-by server (hot stand-by) and prior database opening function accelerate the switching process using additional stand-by server. Furthermore, we offer varied operation modes as indicated in **Figure 1** that satisfy various system requirements and budgets of customers.

On the other hand, there are requirements to ensure performance and expandability in systems that are increasingly becoming large. To address these requirements, the SharedNothing method[note 1)] is used to achieve both high performance and expandability, because it allows parallel processing without generating conflicts for database server resources (**Figure 2**). Further, by combining this method with hot stand-by operation to enhance availability, the performance, scalability and reliability of large-scale systems can be ensured.

Because the rate of database updates will drastically increase as system scale expands, the rate of write access to the logs that guarantee the transactions will increase. As a consequence, log write-access performance tends to become a bottleneck that adversely affects the performance of an application server. To address this issue, logs are divided into multiple groups based on each business application program to ensure good performance. These units of divided logs are called "log groups" and operations based on multiple log groups are called "scalable log operations." Further, by combining scalable log operations and load-sharing operations, it is possible to limit the impact of a fault to only within the concerned log group unit.

Thus, Symfoware has evolved under the concept of thoroughly protecting customers' business processes by protecting the entire system. In recent years, the importance of

note 1) An access method where database access control is matched with the node and only the node concerned can access the database under control. Performance proportional to the increase of node unit can be achieved.

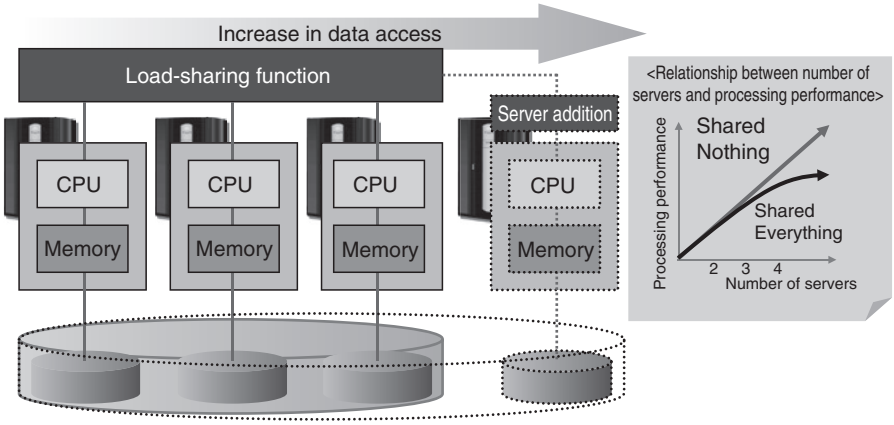


Figure 2
SharedNothing removes trade-offs between high-performance and scalability.

simultaneous processing for large-volume transactions has been increasing. Such processing is one way to make positive use of data including log information from algorithmic trading[note 2)] and data from RFID tags and various sensors. This has resulted in the further expansion of systems. To address this situation, the technologies so far developed need to be further improved.

3. Increase of transactions in mega-scale systems

A great many transactions are generated with the expansion of systems. Once a server fault occurs, a large amount of transactions will remain queued during the task interruption. This means that the cost spent for recovery after task resumption is extremely high. Therefore, it is becoming increasingly important to reduce the processing time to recover the system from failure.

In conventional load-sharing database systems, system availability has been ensured by adopting a failover function based on each database server. This method has advantages in terms of its high clarity and reliability, based on the fact that the switch is carried out by the database server unit. While maintaining this characteristic, the performance at switching has been enhanced. This has been achieved by distributing logs for database processing by Symfoware into multiple log groups and conducting simultaneous switch controls. This improvement has helped reduce the time needed for restarting the tasks. Nevertheless, the drastic increase in the amount of data to be handled by Symfoware gave rise to the significant increase in transactions. So, the two challenges mentioned below have been focused on:

1) Challenge 1: Prolonged time for switching caused by increase in number of disks

If a large amount of data is to be handled by Symfoware, the number of disks necessary to accommodate that data in a database server will also increase. In such a situation, it is essential to distribute logs that limit the writing performance into a database in order to maximize the server performance. As a consequence, a single database server needs to have a large number of disks for log groups.

On the other hand, when switching takes place from one server to another, a large amount of processing time will concentrate on a single database server or a single Symfoware instance (DBMS execution unit). Such time includes the time needed for identifying many disks,

note 2) One type of stock trading. Trading based on a computer system where buy and sell orders are repeated based on predetermined rules.

completing I/O processing on them, and for controlling these disks from the CPU and the time for synchronization. This poses a barrier to further reducing the time necessary for task resumption.

2) Challenge 2: Equipment cost accrued from preparation of stand-by servers

When adopting a switching method by each database server, it is necessary to prepare stand-by servers that take over the original. In the case of a load-sharing system comprised of multiple database servers, it is necessary to have stand-by servers of a number proportional to the number of the database server units. However, with the increase in unit number, the CPU usage ratio of the entire system will drop. Further, equipment cost increases due to the need for additional space for server installation and for a reinforced power supply and cooling systems.

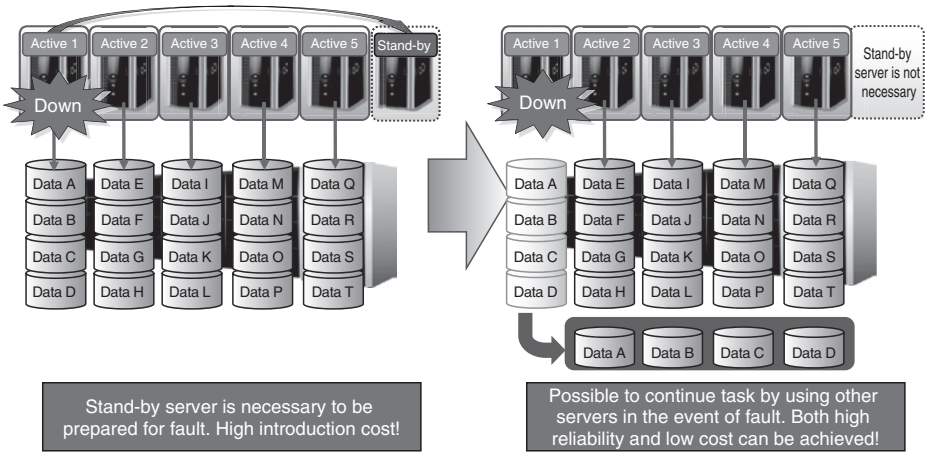
To address these challenges, in the latest development project we reinforced the following two points. We focused on the ability of business continuity in a load-sharing system that carries out a large amount of transaction processing.

- Function to switch data access role to a sound database server (fallback function of database server)
- Function to keep optimum connections from application servers to database servers all the time (application-connection shifting function)

4. Solution based on fallback function of database server

A new function is integrated Symfoware that shifts an access to log groups from the faulty database server in charge to the remaining sound database servers. We call it a "fallback function of database servers." This method can overcome the performance barrier caused by serialization in the switching process adopted in the conventional approaches that were employed for database server and Symfoware instance. We have a lot of work to succeed the log groups from the faulty server to other sound servers. By distributing the load to multiple servers and conducting them in parallel processing, the time necessary for task resumption can be reduced (**Figure 3**).

By adopting this approach, there is no need to have a stand-by server, because the remaining sound database servers succeed the processing managed by the faulty server. This enables efficient use of the CPU as an entire system.



(a) Conventional load-sharing function

(b) Configuration of load-sharing function after improvement

Figure 3
Efficient fallback mode of DB servers.

Based on the above, it became possible to both reduce the time until task resumption and minimize the amount of equipment needed to secure availability while maintaining the high-speed feature of the SharedNothing method.

5. Application-connection shifting function

With regard to application servers connecting to database servers, database access is possible even if they connect to any database server in a system based on the load-sharing configuration. Thanks to the concept of "one large database server" comprised of many servers, details of its inner structure are concealed from application servers. This makes the connection design easy.

On the other hand, there was a requirement for significant speed with regard to application server performance in this development project that focused on a large amount of transactions. In this case, it is advantageous to use the direct access method in terms of performance. In this method, the connection path is proactively controlled so that an application server should be directly connected to the database server that controls the data to be accessed (**Figure 4**).

In the conventional load-sharing system, direct access to log groups is naturally maintained even if database servers are switched. This is because the stand-by server to be taken over is fixed.

We implemented an application-connection shifting function to reinforce the fallback function of database servers. In the event of fallback, this new function shifts application connections from the faulty database servers to the new servers that will succeed the direct-access to the log groups in question (**Figure 5**). By ensuring direct accessibility of application servers to the data to be accessed (log groups), communication costs among database servers are minimized.

In this way, approaches to reinforce system performance are made even after the fallback of database servers.

6. System scalability

To realize stable operations on a continuous basis, a database should grow along with the growth of customers' businesses. To be specific, it is necessary to address the increase in the number of transactions and data volume due to

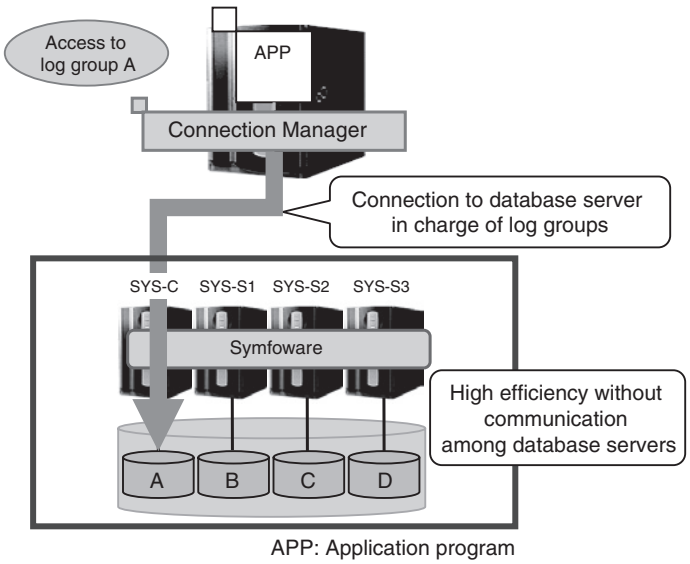


Figure 4
Connection to DB server with direct access.

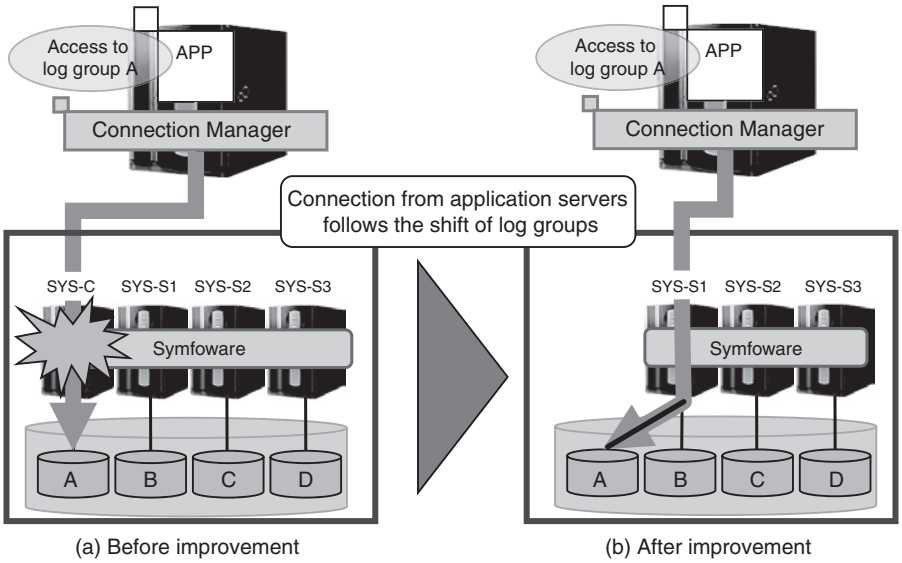


Figure 5
Reconnecting application servers to DB servers with direct access.

the integration of new businesses and day-by-day data accumulation.

Our load-sharing system can keep pace with these changes by being scaled out. In this load-sharing system, an increase in the entire processing amount can be effectively handled by adding database servers and distributing workloads.

In the SharedNothing method adopted by Symfoware, a considerable amount of data had to be reallocated to the added database servers. Because of this, it was necessary to reexamine the database configuration used in the conventional load-sharing system. In such a case, data once arranged on a disk should be extracted, the storage for the concerned tables should be reallocated to a different database server, and the data should be restored there. This procedure requires a significant amount of time and labor, leading to high risks due to potential human errors. In our approach to developing a fallback function for database servers, the solution shown below was developed to address this issue.

In this fallback function, the administrator adds new database servers to the system configuration file and changes some of the names of database servers in the file to the names of new servers. In accordance with this new configuration file, the log groups to be migrated will be succeeded by the new servers. Then, just by rebooting the system, the shift of database configuration will be completed automatically (**Figure 6**).

Important functions of corporate database products are functions for business continuity that can handle not only potential faults but also accommodate the business growth of our customers.

Even when it is hard to predict the amount of data and the rate of access after system inauguration, our method allows stable system operation for a long period, because it is easy to add storage and servers if necessary. With these features, it is easy to reconfigure databases.

7. Example of introducing mega-scale system

Excellent scalability has been secured by making the aforementioned fallback function of database servers and the application-connection shifting function. Consequently, Symfoware has been adopted in arrowhead, the latest

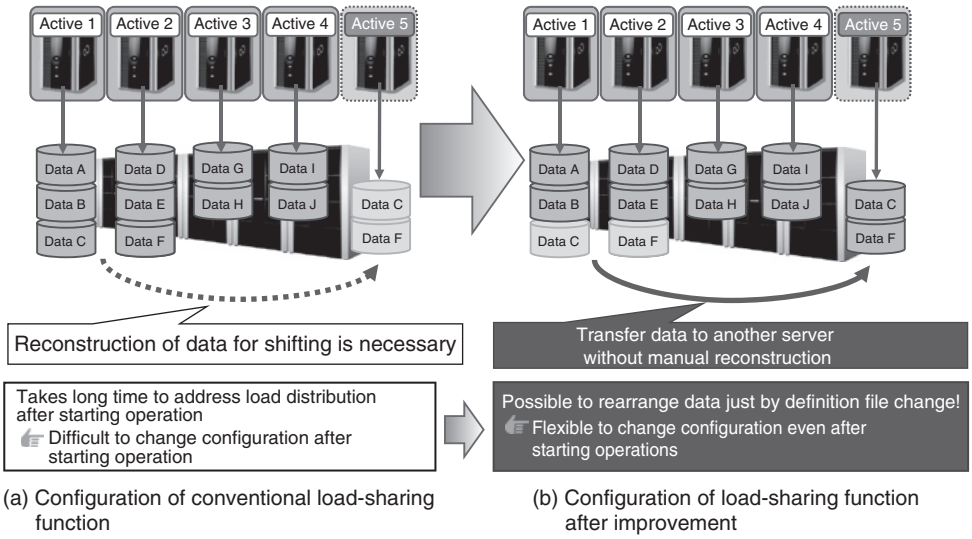


Figure 6
Expansion of DB system without data reconstruction.

stock trading system used by the Tokyo Stock Exchange, since January 2010.[1)]

arrowhead is a mega-scale load-sharing system that works by dividing its storage exceeding 1 TB into several dozens of log groups. It performs distributed processing using a group of database servers comprised of multiple PRIMEQUEST units for each disk. It is a mission-critical system that must not be stopped by any means. It is a gigantic system that truly represents Japan's technological prowess.

Symfoware ensures the availability of database functions through its fallback function and application-connection shifting function, while processing SQL commands more than 14 million times per minute, as requested by arrowhead. It also ensures a switching-over performance in the order of seconds.

Achieving both good performance and availability in such a mega-scale system could be achieved only with Symfoware that integrates the fruits of longtime commitments to these objectives.

8. Approaches to diversifying system requirements

The request for uninterrupted systems was previously only for large-scale systems owned by a limited number of large companies. But nowadays, demands for uninterrupted systems have come also from small and medium-size companies. For instance, it has become possible not only for large companies but also for those small and medium-size companies to deal directly with overseas companies. Accordingly, there are demands to have systems operate for longer or operate non-stop when working with such overseas counterparts in different time zones. Such a situation is attributable to the increasing dependency of business processes on ICT systems, regardless of the corporate scale. This is because of the adoption of the Internet and other technologies.

Therefore, there are needs to have not only the above-mentioned functions for mega-scale systems but also an affordable uninterrupted system that allows simple operations. To address these requirements, we offer a mirroring function that allows the entire database to be easily and completely duplicated. With this technology, high-reliability operations at an affordable budget are possible for customers to whom it has been difficult to introduce clusters and recovery operations.

The features of the mirroring function are as follows:

1) Simple business continuity

There is no need to cut out points of fault and select a recovery method when there is an abnormality. It is possible to continue business processes using a sound database whatever the type of the fault may be. Besides, there is no need to conduct a complicated recovery design before inauguration.

2) Low cost

It is not necessary to purchase expensive equipment, because ordinary hardware components can be used. Further, this solution is feasible with the minimum amount of resources, because mirroring is achieved by using the minimum number of logs. In addition, the initial introductory costs can be reduced because the system is simple to set up.

9. Conclusion

Based on Fujitsu's core technologies nurtured for many years, new technologies for load-sharing operations were developed with Symfoware. Application-connection shifting functions were reinforced in the load-sharing operation. In this way, both excellent performance and high reliability could be achieved in the system based on close interconnection of application servers and database servers. Further, to address diversifying requirements, we are committed to improving various types of systems in terms of scale and requirements, including mega-scale systems, in order to realize stable operations.

As we described above, Symfoware has evolved with the growth of its customers. We have made our best efforts to realize stable operations in customers' systems based on reinforcing technologies and improving operability. As a result of these efforts, Symfoware was evaluated as the number one system in the "open-type RDB software division" of the 15th customers' satisfaction study sponsored by *Nikkei Computer*.

Particularly, its proven performance in arrowhead confirmed its world-class level reliability and scalability. Furthermore, it was highly evaluated for the continuous efforts to improve it from the users' standpoint, for its ease of system development, and for its enhanced operations management.

We will continue working to offer more sophisticated products that ensure our customers' systems operate stably.

Reference

1) Tokyo Stock Exchange: TSE Equity Market Summary after arrowhead Launch. Feb. 25, 2011. *http://www.tse.or.jp/news/47/b7gje6000001cjo1-att/20110225_b.pdf*

Kazunori Takata
Fujitsu Ltd.
Mr. Takata is currently engaged in the development of Symfoware.

Katsumi Matsumoto
Fujitsu Ltd.
Mr. Matsumoto is currently engaged in the development of Symfoware.

Masaki Nishigaki
Fujitsu Ltd.
Mr. Nishigaki is currently engaged in the development of Symfoware.