Problem Detection and Automatic Recovery of Business Applications

● Toshihiro Mimura

(Manuscript received January 9, 2007)

High reliability is important for mission-critical systems, and to achieve it, advanced prevention and rapid recovery from latent problems are necessary. Fujitsu's Interstage Application Server meets these needs by providing functions that detect signs of problems and recover applications automatically. This paper describes how Interstage Application Server can be used to provide high availability for business applications.

1. Introduction

Problems in mission-critical systems can cause a substantial loss of business. Therefore, it is very important for systems to be highly available. In general, the system administrator (hereafter administrator) does the following to ensure high availability.

1) Prevents latent problems from developing

The administrator must be able to detect any signs of latent problems to prevent them from developing. The administrator uses some type of monitoring to detect early signs of a problem, for example, monitoring of an application's response time.[note)] If the response time is longer than expected, the administrator will assume there is a problem, analyze the reason for the slow response, and then take action before the problem develops.

2) Rapidly recovers the system from problems

If a system problem occurs, the administrator takes predetermined countermeasures to eliminate the problem. For example, if an application terminates abnormally, the administrator will restart the process containing that application. For rapid recovery, the administrator will try to automate problem detection and the implementation of countermeasures if possible.

note) "Application" here means an application used on an online system.

To support high-availability systems, Interstage Application Server[1)] provides functions to detect the early signs of problems, detect the occurrence of problems, and perform automatic recovery. These functions have been developed from the experience Fujitsu gained through operating a large number of mission-critical systems.

This paper describes how Interstage Application Server detects the signs of problems and automatically recovers a system from problems.

Because an increasing number of systems, particularly mission-critical systems, are based on the Java platform, this paper focuses on problems that may occur with Java applications.

2. Requirements for high system availability

As mentioned above, early detection of problems is required to prevent problem escalation. For problems that cannot be prevented, rapid recovery is required. In many cases, application crashes and freezes are caused by mistakes in resource estimation and bugs in applications.

Some concrete problems and countermeasures for these mistakes are described below.

1) Preventing latent problems from developing

- Monitoring the JavaVM resource usage

Because it is difficult to estimate the amount of memory used by a Java application, application processes may terminate or freeze due to a java.lang.OutOfMemoryError (OutOfMemory) exception. Also, an application may respond slowly due to frequent Garbage Collection (GC) operations by the JavaVM.

To prevent these problems, a function that monitors the Java heap and permanent areas (Java Heap/Perm) and the GC activity is required. By using this function, the administrator can detect signs of memory shortage and performance reduction.

- Monitoring the number of requests waiting to be processed

When the number of requests increases and the application's processing speed slows down, the application cannot process all the requests. To prevent this from happening, it is necessary to detect any abnormal increase in the number of requests waiting to be processed.

2) Rapid recovery from a problem

- Automatic recovery for abnormal termination of application processes

An application bug can cause an abnormal termination of an application. In this case, the application can often be recovered by restarting it. If the problem rarely occurs, and the continuation of business takes priority over finding the cause of the problem, the administrator will restart the application to recover it. Therefore, when an abnormal application termination is detected, to achieve rapid recovery, an automatic application restart function is required.

- Automatic recovery when an application's response is slow

Mistakes in resource estimation and application bugs can slow down or even freeze an application. However, there are many cases when an application can be recovered from these problems by restarting it. Therefore, an automatic application restart function is required to achieve rapid recovery when an application has not responded for a specified time.

- Automatic recovery when JavaVM throws an OutOfMemory exception

As mentioned above, when an OutOfMemory exception is thrown, the application usually terminates or freezes. If the process is terminated, it will be recovered by the above-mentioned automatic recovery function, but if the process freezes, some other countermeasure is required. To ensure high availability, a countermeasure to prevent processes from freezing is required. One of the solutions for this problem is to restart the application process automatically without it freezing when an OutOfMemory exception is thrown.

The next section describes how Interstage Application Server provides high availability by meeting the requirements described above.

3. How Interstage Application Server provides high availability

Interstage Application Server provides an original application management feature called a WorkUnit. By using this feature, Interstage Application Server provides functions for high availability such as functions for detecting the signs of problems and providing automatic recovery.

This section gives an overview of the WorkUnit functions and then describes how the WorkUnit detects the early signs of problems and performs automatic recovery from problems.

3.1 WorkUnits

A WorkUnit is an operation unit consisting of one or more applications. WorkUnits can be made to correspond to business operations by allocating one WorkUnit per business operation.

Each WorkUnit has its own tuning parameters or management functions to accommodate the characteristics of an application. For example, a WorkUnit consists of a number of concurrently running processes. By creating multiple processes, it is expected that the system will produce a better throughput or minimize the effect of an abnormally terminated process.

Problem detection and recovery functions are provided to ensure high availability.

A WorkUnit provides the following functions for application management (**Figure 1**).

1) Request management function

This function queues requests from clients and dispatches the requests to the appropriate applications.

2) Process management function

This function manages the activity of application processes and the status of transactions.

3.2 Preventing latent problems from developing

Interstage Application Server provides functions that detect the signs of problems. When Interstage Application Server detects a sign of a problem, it notifies the administrator that there might be an application problem. The administrator can then take action to prevent the problem from developing.

These functions are described in detail below.

3.2.1 Monitoring JavaVM resource usage

The process management function monitors the JavaVM Heap/Perm area as well as the GC execution. When the process management function detects a sign of a problem, it notifies the administrator. To determine if there is a sign of a problem, the process management function regularly samples the Heap/Perm usage and GC execution and calculates the transitions, means,

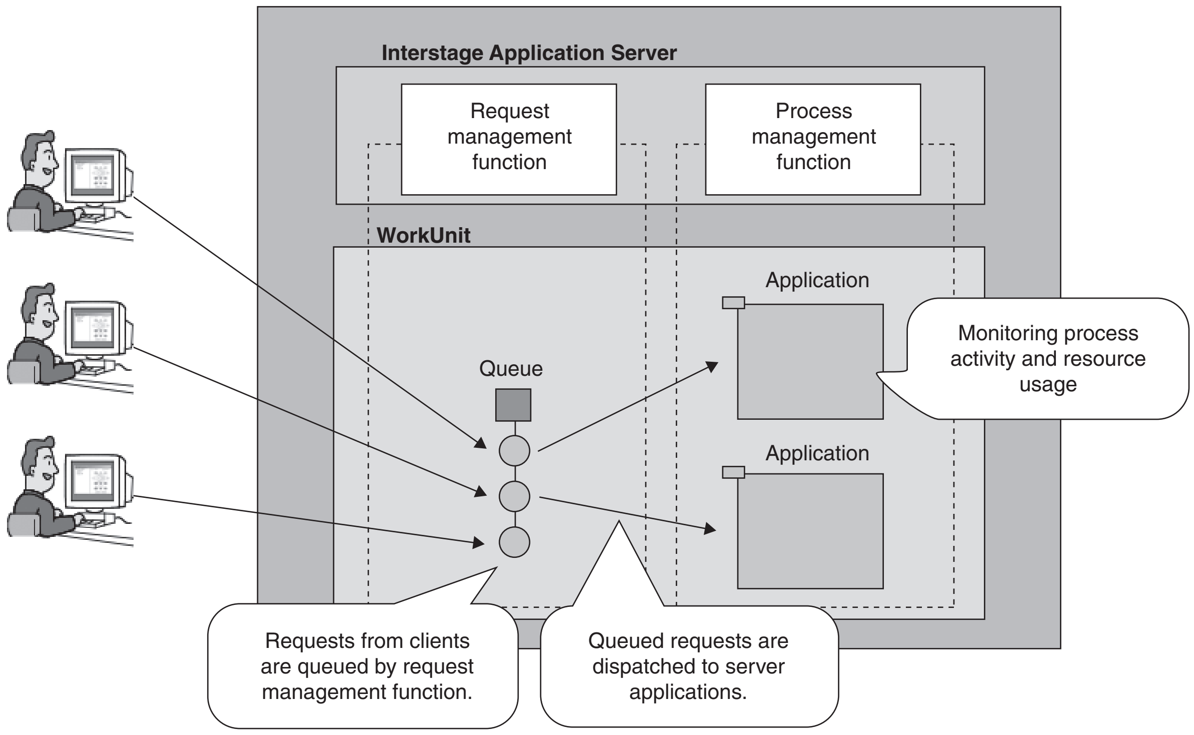
Interstage Application Server
Request management function
Process management function
WorkUnit
Application
Queue
Application
Monitoring process activity and resource usage
Requests from clients are queued by request management function.
Queued requests are dispatched to server applications.

Figure 1
WorkUnit mechanisms.

and other values.

For example, if the process management function determines that an OutOfMemory exception might occur, it outputs the following message:

"The usage rate of the Perm area is larger than 90%."

In addition, there are a number of other cases when the process management function will determine there is a problem (**Table 1**).

When the process management function detects a sign of a problem, the administrator can prevent the problem from developing by restarting the server processes. The administrator can also analyze the information about the problem and take action to remove its cause.

3.2.2 Monitoring the number of requests waiting to be processed

The request management function monitors the number of requests that are waiting to be processed by an application. If the number of requests exceeds the expected number, the request management function will interpret this as a sign of a problem and notify the administrator.

Table 1
Monitoring cases for JavaVM.

Cases	Determination of process management function
A and B A) Increase in amount of use of Perm area from the last measurement exceeds 10% of entire Perm area. B) When assuming an increase at the same rate, use rate of Perm area exceeds 90%.	Use of Perm area is increasing rapidly.
Mean value of execution time for GC for the previous three times exceeds 5 seconds(s).	GC still takes a long time to execute. Possibility of insufficient JavaVM heap space.
A and B A) Interval between executions of GC for the previous three times is less than 20 s. B) Heap usage rate immediately before GC execution is less than 65%.	Inefficient GC is executed at short intervals. There are the following possibilities: • Insufficient JavaVM heap space. • java.lang.System.gc( )^note) is called within a short period.
A and B A) Interval between executions of GC for the previous three times is less than 20 s. B) Heap usage rate is less than 65% immediately after execution of GC.	Memory shortage.
A or B A) Heap usage rate immediately after GC exceeds 95%. B) Heap usage rate exceeds 95% immediately after occurrence of GC, and heap usage exceeds that of just after the last GC execution three or more times.	Memory shortage.
Usage rate of Perm area exceeds 90%.	Shortage of Perm area.
Execution interval of GC for the previous three times is less than 20 s.	GC is executed at short intervals. There are the following possibilities: • Insufficient JavaVM heap space. • java.lang.System.gc( ) is called within a short period.
A, B, and C A) Execution interval of GC for the previous three times is less than 20 s. B) Mean value of execution time of GC for the previous three times exceeds 0.5 s. C) Frequency of GC executed by java.lang.System.gc( ) is high.	java.lang.System.gc( ) is executed at short intervals. Unnecessary java.lang.System.gc( ) is frequently called from application.

note) java.lang.System.gc( ) is a method for instructing the JavaVM to execute GC. When this method is issued frequently, the JavaVM load becomes high, causing the application to have a slower response.

The request management function receives requests from clients and dispatches them to the appropriate server. If an application is busy, these requests will be queued for processing. The request management function also monitors the number of these queued requests.

The administrator can define three threshold values for monitoring the number of queued requests (**Table 2**).

If the number of queued requests exceeds the specified monitoring number, the request management function sends a message to the administrator. The administrator can then prevent a problem from developing by analyzing its cause and taking suitable action.

3.3 Recovering from system problems

When a problem occurs in an application of a WorkUnit, Interstage Application Server detects it and automatically performs recovery. Therefore, the administrator does not have to do anything manually, and as a result, recovery is rapid and the loss of business is minimized (**Figure 2**).

The following sections describe how these functions can detect and resolve problems by illustrating the pattern of problems.

3.3.1 Automatic recovery for abnormal termination of application processes

When a WorkUnit is started, its application processes are started and the process management function begins to monitor their activity. The function detects when an application process goes down and automatically activates an alternative process. While the process management function is recovering, requests from clients are queued by the request management function. Therefore, clients will receive no errors from the server and business will continue uninterrupted.

The flow of automatic recovery is as follows (**Figure 3**).

1) An application process terminates abnormally (① in Figure 3).
2) The process management function detects the termination, identifies the application running on the terminated process, and activates an alternative application process automatically (②).

   While the application process is down, the following functions keep the loss of business to a minimum.

   - The request management function receives requests from clients regardless of the application's status (③).
   - If the application is also running on

Table 2
Threshold values for number of queued requests.

Definition	Explanation
Maximum Queuing Message	Maximum number of requests to be received. If the number of requests exceeds this value, the request management function provides a warning by outputting a message to Syslog (Solaris/Linux) or EventLog (Windows). Requests that exceed this threshold value are not received.
Queuing Message to Notify Alarm	Threshold at which to suspend monitoring of the number of requests waiting to be processed by an application. If the number of requests exceeds this threshold, monitoring of the number of requests is suspended and the request management function provides a warning by outputting a message. The message is not output again until the number of queued requests returns to below the Queuing Message to Notify Resumption value.
Queuing Message to Notify Resumption	Threshold at which to restart monitoring of the number of queued requests after the Queuing Message to Notify Alarm value has been exceeded. When the number of requests returns to below the Queuing Message to Notify Resumption value, the request management function notifies the system administrator by outputting an information message and monitoring of the number of requests is restarted.

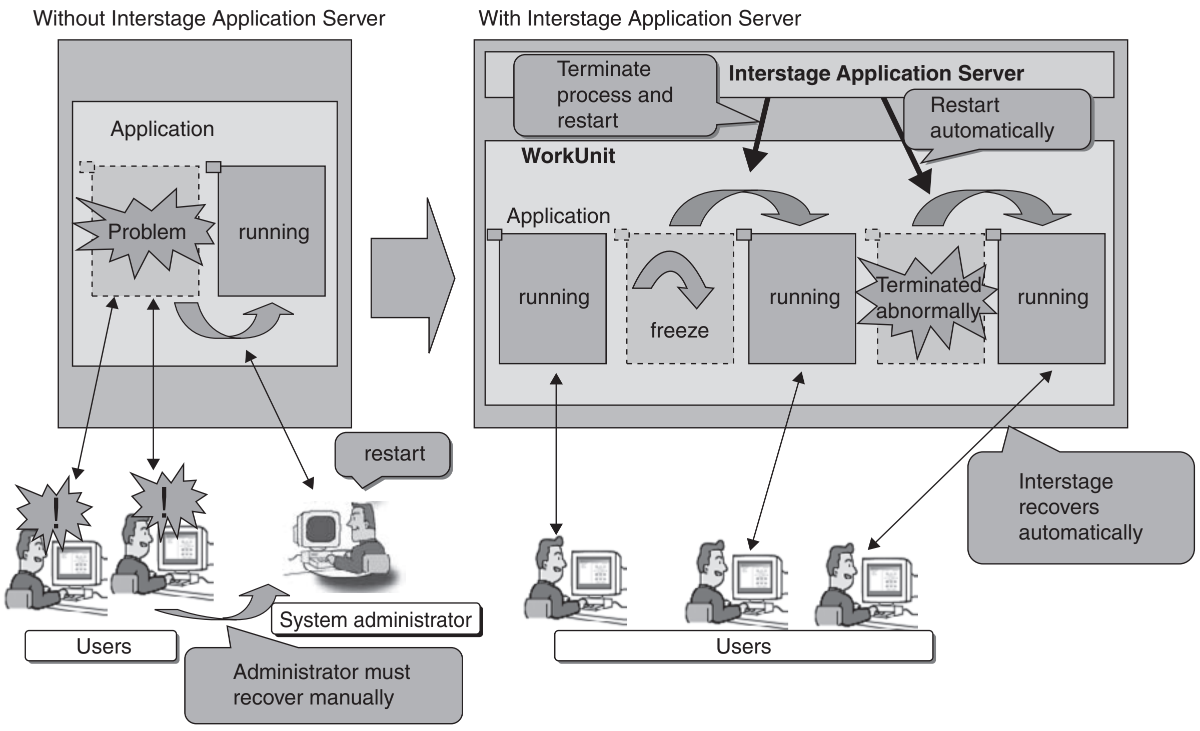
Without Interstage Application Server
Application
Problem
running
restart
System administrator
Users
Administrator must recover manually
With Interstage Application Server
Terminate process and restart
Interstage Application Server
Restart automatically
WorkUnit
Application
running
freeze
running
Terminated abnormally
running
Interstage recovers automatically
Users

Figure 2
Automatic recovery from system problems.

other processes, the remaining processes continue to receive and process requests (④).

3) The alternative application process begins to receive requests (automatic recovery is completed) (⑤).

3.3.2 Automatic recovery for slow application responses

When a bug slows down an application's response, the problem can often be resolved by restarting the application. Therefore, when the process management function detects a slow application response, it restarts the application automatically.

The process management function monitors the processing time of an application by monitoring the time from when the application receives a request to when it returns a reply. If this time exceeds the specified value, the process management function determines that the application's response is slow and terminates the relevant process. Then, the process management function automatically activates an alternative application process in the WorkUnit.

The flow of automatic recovery is as follows.

1) The application processing time exceeds the specified value in the corresponding WorkUnit.
2) The process management function detects that the specified processing time has been exceeded. Then, it terminates the relevant application process and automatically activates an alternative application process.
3) The alternative application process begins to receive requests (automatic recovery is completed).

3.3.3 Automatic recovery for resource shortages in applications (JavaVM OutOfMemory exception)

As mentioned in Section 2, when an

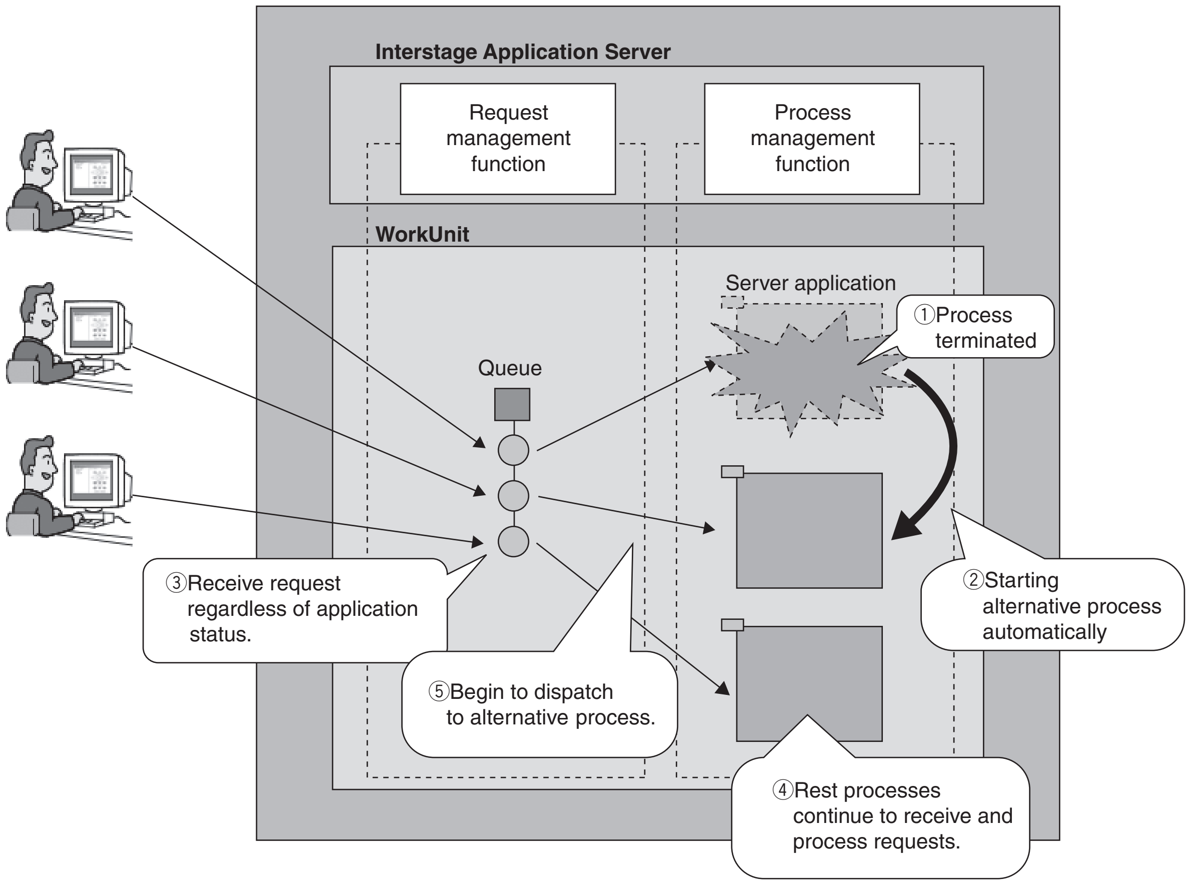
Interstage Application Server
Request management function
Process management function
WorkUnit
Server application
①Process terminated
Queue
②Starting alternative process automatically
③Receive request regardless of application status.
⑤Begin to dispatch to alternative process.
④Rest processes continue to receive and process requests.

Figure 3
Automatic recovery when application process terminates.

OutOfMemory exception is thrown from JavaVM, a countermeasure to recover without the application freezing is required. Interstage Application Server provides a function that restarts an application automatically when an OutOfMemory exception is thrown.

When an OutOfMemory occurs in a J2EE application running within a WorkUnit, the administrator can select one of two actions:

1) Return "java.lang.OutOfMemoryError" to the application.
2) Restart the application process.

If the administrator selects the "Restart the application process" property of a WorkUnit, when the process management function detects an OutOfMemory, the function terminates the process and then activates an alternative application process to recover automatically.

With this function, the system can quickly recover from a frozen process.

If the administrator selects the "Return "java.lang.OutOfMemoryError" to the application" property of the WorkUnit, the java.lang.OutOfMemoryError is returned to the application in the same way as for normal Java applications.

The flow of automatic recovery from a freeze is as follows.

1) An OutOfMemory exception is thrown.
2) The request management function detects that an exception has been thrown, terminates the process, and then automatically activates an alternative application process.
3) The alternative application process begins to receive requests (automatic recovery is completed).

4. Conclusion

In this paper, we explained how Fujitsu's Interstage Application Server ensures high system availability. We showed that the predictive monitoring and automatic recovery functions contained within WorkUnits greatly improve system availability, although there are some issues that remain to be addressed.

We described the functions for detecting problems and automatically recovering applications. The functions for detecting the early signs of problems are effective for preventing latent problems from developing. However, to prevent these problems from developing, these functions require manual actions by the administrator. Therefore, functions that automatically prevent problems are required. For example, a function that automatically extends the size of the Heap/Perm area when it is running low is required.

We believe we need to develop these functions to further improve reliability.

Lastly, we need to be aware of ways to remove factors that may cause application problems during the design and development phase. This will be a challenging task, but we will continue our work to improve the quality of our application designs.

Reference

1) Fujitsu: Interstage Application Server. *http://www.fujitsu.com/global/services/software/interstage/products/apserver/*

Toshihiro Mimura, *Fujitsu Ltd.*
Mr. Mimura received the B.S. degree in Administrative Engineering from Kinki University, Hiroshima, Japan in 1992. He joined Fujitsu Kobe Engineering Ltd., Numazu, Japan in 1992 and moved to Fujitsu Ltd. in 2003, where he has been engaged in research and development of application servers.