CICS Transaction Server for z/OS —a robust platform for Java.

By Scott Clee, Paul Cooper and Pete Seddon,
IBM Software Group

June 2007

Contents

Introduction

This white paper discusses the additional new support IBM CICS® Transaction Server for z/OS®, Version 3.2 provides for running Java™ applications in a high-volume, high-availability transaction-processing environment. The new Java support is part of the drive toward modernizing business applications and using them in flexible, new, services-based IT infrastructures such as service oriented architecture (SOA).

Figure 1 shows the IBM SOA reference architecture.

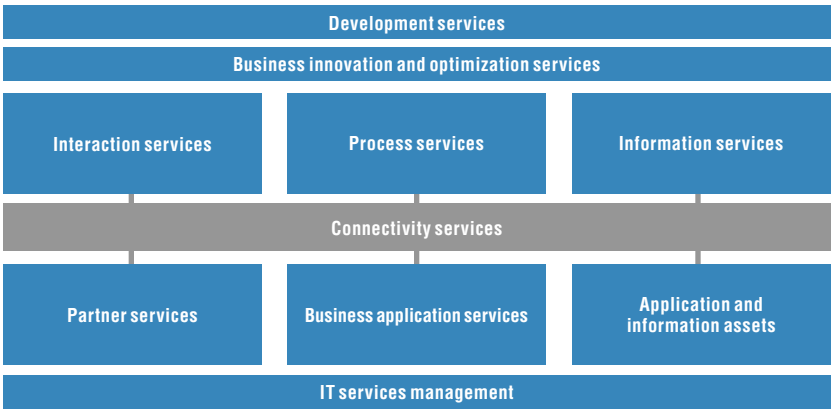

Figure 1. IBM SOA reference architecture

The new release of CICS Transaction Server includes enhancements that deliver better manageability, serviceability and usability when running Java technology-based workloads in CICS Transaction Server. These enhancements are based on real use cases and have been made in direct response to IBM clients who have requested greater control over how CICS Transaction Server manages Java workloads.

This white paper is designed to address the needs of application developers, operational support staff and anyone interested in the Java support provided by the latest version of CICS Transaction Server.

CICS Transaction Server and Java readiness

Java has rapidly grown in popularity throughout the IT industry, and for many organizations, it is now their programming language of choice. CICS Transaction Server has extended the support for Java technology-based workloads over a number of releases in response to the uptake of the Java programming model. CICS Transaction Server, Version 3.2 adds further support for Java, making this version essential for anyone already running Java workloads in earlier-version CICS environments.

From a historical perspective, the expectations that users had for CICS Transaction Server and Java have matured over time. When IBM first introduced Java support in CICS Transaction Server, it was expected that Java should behave as much like COBOL as possible from within the CICS runtime environment. Today, the expectation is that Java in CICS Transaction Server should be much like Java in other applications.

The unique characteristics of the persistent reusable Java Virtual Machine (JVM) that has been traditionally supported in earlier versions of CICS Transaction Server make it more difficult to write Java programs that run as intended in CICS Transaction Server. As a result of this and other performance-related issues, many users of Java in a CICS Transaction Server, Version 2.3 environment have elected to use the continuous JVM exclusively and have abandoned the persistent reusable JVM altogether.

Recent versions of the JVM do not include the persistent reusable JVM extensions. To make CICS Transaction Server consistent with this Java change, CICS Transaction Server, Version 3.2 also does not support the persistent reusable JVM (support is for the continuous JVM only). Support for the class cache remains unaffected.

The persistent reusable JVM was first made available for use in CICS Transaction Server, Version 2.1. It offers the ability to reset the state of a JVM between tasks to help ensure that subsequent users of the same JVM are fully isolated from states left behind by previous users of the JVM. The time taken to reset a JVM depends on there being no cross-heap references between the middleware and the application heaps within the JVM. If these references exist, the JVM scans the heap to determine if the references are in live objects. The scan process is rather slow. If the attempt to reset the JVM fails, CICS Transaction Server discards the JVM and creates a new one. These unresettable events (UREs) have been a major performance problem for some users of CICS Transaction Server and Java.

The continuous JVM was introduced in CICS Transaction Server, Version 2.3. It offers the ability to omit resetting the JVM between CICS tasks, helping to ensure that there are no performance problems due to cross-heap references. It also offers the ability to cache states between transactions to help improve performance. As a Java application-execution environment, it is more consistent with Java in other environments and on other platforms (for example, the class loading, threading, just-in-time [JIT] and garbage-collection components are the standard ones). CICS Transaction Server is still designed to ensure complete isolation between concurrently running tasks running in different JVMs in both JVM modes, but isolation issues might exist between serial tasks running in the same JVM. In practice, most CICS Java workloads now use the continuous JVM to benefit from the considerable performance advantages. With the 2007 announcement of CICS Transaction Server, Version 3.2, the focus is now firmly on continuous JVM and the operational advantages this feature confers (see Figure 2).

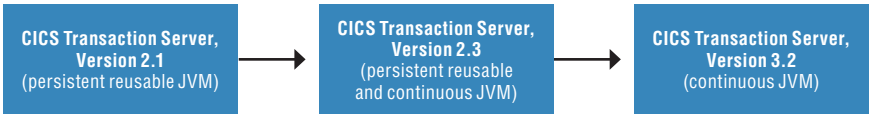

Figure 2. History of JVM support in CICS Transaction Server

From a migration viewpoint, IBM provides a tool for upgrading Java applications so that they can be used in the continuous JVM environment. The tool is called *IBM CICS JVM Application Isolation Utility*, and reports the use of a static state within an application (the main cause of isolation issues between serial users of a JVM). If the tool is used along with a JVM security-policy file, it can help the administrator ensure that isolation breaches (caches) occur in a planned way.

The tool scans all Java class files and Java archive (JAR) files for methods that attempt to modify static variables for Java applications. If the tool discovers these methods in the scanned files, it generates a report that contains the class name, method and static variable name of each relevant method. This information then enables the administrator to decide what to do with the affected fields (bearing in mind that, when a JVM runs in continuous mode, static initialization of fields does not happen on subsequent uses of that JVM).

However, the tool is not just about upgrading applications; it is also useful in general deployment scenarios for determining whether Java applications are suitable candidates for deployment in CICS Transaction Server. The tool helps the system administrator audit compiled Java programs to see whether they are likely to experience problems when deployed to a continuous JVM environment.

The Web user interface (WUI) for CICSPlex System Manager has also been made consistent with the changes to JVM support. For example, the reset option in the JVM profile has been removed and other options related to the persistent reusable JVM are no longer available because it is no longer supported.

Java manageability, serviceability and usability enhancements to CICS Transaction Server

Several manageability, serviceability and usability enhancements have been made in response to feedback from CICS and Java users.

Schedule garbage collection at specified heap usage

The garbage-collection process reclaims dereferenced storage so that it can be reallocated and reused. Java applications tend to generate large amounts of garbage that slowly consume storage that would otherwise be available for allocation. In CICS systems, Java garbage collection can occur for the following reasons:

- *The JVM has run short on storage and schedules a garbage collection.*
- *CICS Transaction Server schedules a garbage collection.*

CICS Transaction Server schedules a garbage collection to occur after a specified number of JVM uses (a value that has been configurable since the release of CICS Transaction Server, Version 2.3). Garbage collection occurs after an application has finished with the JVM, but before the task ends. This means that the cost of the garbage collection is included in the cost of the task, and application-response time suffers as a result. To address this problem, the garbage collections that are scheduled by CICS Transaction Server, Version 3.2 occur at a specified target heap utilization (the default value when this happens is 85 percent of heap utilization) and in a separate CICS system task. A value of zero implies that garbage collection is never scheduled by CICS Transaction Server, in which case garbage collection occurs only when needed within the application task. Keep in mind, though, that unscheduled garbage collections within the JVM can still occur at any time, but fine-tuning can help minimize the likelihood of this happening.

Specify a JVM timeout value

CICS users can now access a new JVM profile-idle-timeout option that specifies how long a JVM remains in the JVM pool if it becomes inactive. The range is zero to seven days (a value of zero is infinite and means that the JVM is never terminated). The default idle timeout value is 30 minutes. If the number of idle JVMs in the pool exceeds the workload requirement, CICS Transaction Server terminates them automatically. However, CICS Transaction Server does not immediately terminate all JVMs that have timed out; instead, it terminates them gradually. This capability helps ensure that a balanced capacity is maintained in the JVM pool. Those JVMs that have timed out but have not yet been terminated are still available for reuse by applications if an increase in demand occurs. If a JVM is reused, its timeout value is automatically reset. But it's important to remember that CICS Transaction Server never automatically terminates the last JVM in the JVM pool.

Being able to specify a JVM timeout value from within such a wide range provides much greater flexibility in handling workload. It also helps reduce the system-processing overhead and increase performance because the JVM pool is more-effectively managed by the system.

Preinitialize JVMs

The same administrators who experience problems with JVMs being discarded too soon typically require the ability to preinitialize JVMs so that the first CICS Java application task does not incur the processing cost and delay of starting a JVM. CICS Transaction Server, Version 3.2 includes a new system programming interface (SPI) command for creating a specified number of JVMs from the same JVM profile. It is also possible for users to write a program that initializes the required number and type of JVMs ahead of time (for example, at start-up) or at a specified time (for example, at midnight), or after a JVM pool has been phased out.

Selectively phase out JVMs by profile

CICS Transaction Server now offers users the ability to selectively phase out JVMs in the pool that have a specific profile. As with previous versions of CICS Transaction Server, it is still possible to phase out all JVMs in the JVM pool regardless of profile. The ability to phase out by profile provides much greater flexibility when managing the JVM pool. It also helps maintain workload throughput because the user does not have to take down all JVMs in the pool to implement changes to some, such as when adding new application classes and when refreshing shared Java classes (those on the shareable application class path).

Offer an easier-to-use JVM profile

The JVM profile in CICS Transaction Server, Version 3.2 is simpler than the JVM profiles used in earlier CICS versions. Some profile options (including TMPREFIX and TMSUFFIX) have been deprecated. In addition, JVM properties can be specified within the JVM profile itself, rather than in a separate properties file, helping to reduce the number of files that have to be maintained.

Additional validation checks have also been implemented to address common user errors such as the accidental use of a CICS directory entry that is not at the latest level. The JVM profile-configuration options in CICS Transaction Server, Version 3.2 are also more consistent with Java configurations on other platforms. It is now possible to use standard JVM configuration parameters that are used on other platforms.

Identify JVM profiles by name

SJ trace messages issued by CICS Transaction Server, Version 3.2 now include the JVM profile name associated with the JVM. This function helps users make changes to the correct JVM profiles in response to messages. In previous versions of CICS Transaction Server, it is not always obvious which JVM profiles are used by tasks.

Summary

This white paper has discussed how the Java enhancements to CICS Transaction Server, Version 3.2 enable businesses to run Java applications as part of modern, flexible, services-based, business-oriented IT infrastructures. These Java enhancements also help deliver better productivity and cost savings through improved manageability, serviceability and usability when running Java technology-based workloads in CICS systems. In addition, enhancements in this release enable Java in CICS Transaction Server to behave in much the same way as Java in other environments, resulting in shorter application-development lead times, more-effective system management and better user expectations.

Other enhancements include:

- *Improved Java garbage collection that can lead to faster application-response times, and ultimately, enables business processes to run more efficiently.*
- *The ability to balance JVM usage more effectively to handle Java technology-based workloads more efficiently.*
- *The ability to preinitialize Java resources before they are needed, helping to reduce processor usage and enabling organizations to manage their business systems more cost-effectively.*
- *The ability to selectively remove Java resources while processing is occurring, so that the system can continue to process valuable workload while administrative changes are made.*
- *Simpler-to-understand Java resources that are easier to identify, helping to reduce user error.*

For more information

To learn more about CICS Transaction Server and Java, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/cics

IBM United Kingdom Limited
Hursley Park
Winchester
Hampshire
SO21 2JN
United Kingdom

Produced in the United States of America
06-07

WSW11333-USEN-00