A View into the New Java Workloads in CICS Transaction Server

IBM Redbooks Solution Guide

For many organizations, the ongoing pressure to manage long-term operational costs can be a psychological barrier when considering deployment of new applications onto IBM® System z®. Yet System z is still one of the most capable and cost effective platforms for running mixed workloads, with the qualities of service on which these organizations depend. In recognition of these ongoing pressures, IBM has introduced the Value Unit Edition (VUE) pricing model to provide customers with greater flexibility to balance the capital and operating expenditure of new projects.

System z New Application Licence Charges (zNALC) give organizations the opportunity to run a new workload on System z (Figure 1) at a reduced cost, provided that the workload is a qualified "new workload" application. An example of a qualified application would be Java language business applications running in IBM CICS® Transaction Server for z/OS Value Unit Edition (CICS TS VUE) or IBM WebSphere® Application Server.

Running CICS Transaction Server for IBM z/OS® using the Monthly Licence Charge (MLC) pricing option is normally considered an operating expenditure (OpEx). This is because the expenses are charged on a monthly basis and are directly related to the CPU usage for the month. Buying IBM CICS Transaction Server for z/OS Value Unit Edition is a one-time charge licence, and is categorized as capital expenditure (CapEx). This one-time charge licence can run in a reduced price zNALC LPAR. The same principle applies to other qualifying software required to support the application.

CICS Transaction Server is part of a family of products available as Value Unit Editions. Other IBM software products that are available as Value Unit Editions include IBM DB2® for z/OS, IBM IMS™ Database, IMS Transaction Manager, and WebSphere MQ for z/OS.

Figure 1. New Java Workloads on System Z

Did you know?

CICS Transaction Server Value Unit Edition allows you to deploy new Java workloads at a fixed cost. zNALC provides a reduced price for z/OS. And for customers who have IBM System z Application Assist Processors (zAAP), Java workload can be transferred to the specialty processor, further reducing the cost of running new Java workloads under CICS Transaction Server.

CICS Transaction Server Value Unit Edition contains a version of the WebSphere Liberty Profile (Liberty) at no extra charge, running in the highly scalable JVM server architecture within CICS TS. A separate WebSphere Application Server licence is not required to run WebSphere Liberty Profile applications inside CICS TS. Deploying new Java workload into Liberty on CICS allows you to exploit many of the Liberty capabilities as well as integrating Liberty applications with the JCICS API, providing the best of both worlds in a single managed runtime.

The IBM JVM is designed to exploit the latest hardware advances in System z, including features such as Transactional Memory (for improved concurrency in multi-threaded applications) and large 1-MB pageable memory using flash. An IBM benchmark shows 40% improvement in Java workloads running on zEC12 using Java7SR3 exploiting hardware improvements compared to z196. CICS TS V5 provides support for Java7SR1 or later, enabling you to use the most efficient JVM runtime available on System z..

Business value

The Value Unit Edition (VUE) family of products provides a choice for how to budget for new projects. Previously, all new projects would primarily result in an increase in operational expenditure. Deploying into a VUE environment on a zNALC LPAR enables a proportion of the project cost to be a capital expense. In organizations where strict budgetary controls are in place, having the flexibility to choose between these two budgets can help to get new projects financed and delivered.

Customers can deploy VUE versions of products to exploit new capabilities without needing to migrate their existing installations to the latest software levels. For example, CICS TS VUE can interoperate with an existing version of CICS TS (MLC) at a lower level, without triggering the single-version-charging (SVC) window. Furthermore, customers can start by deploying the CICS TS Developer Trial, a free version of CICS that allows them to evaluate product capabilities, and later upgrade it in-place to either CICS TS VUE, or CICS TS (MLC), so there is no need to uninstall and reinstall software.

Why Java on the mainframe?

The idea that Java and the mainframe are a bad pairing is an outdated and vastly inaccurate idea. If you still hold the opinion that the two do not fit well together, it is time to think again.

Java is a language, and there are many to choose from, but not all languages get the same investment that Java has had, particularly on the mainframe. Java is also, and perhaps more importantly, a runtime platform. Java on System z has received significant and on-going investment from IBM, building the Java Virtual Machine (JVM) from the ground up, developing hardware that can be exploited by the JVM, and building core middleware technology using Java, making Java on the mainframe one of the world's most optimized Java environments.

System z is itself something special. There is no other machine that has such a vast heritage with such a profound effect on business, science, and people. From putting man on the moon, to becoming the heart of the world's financial system, or to buying cosmetics with a mobile phone from the comfort of a chair, the mainframe has continued to evolve to incorporate the latest advances in technology to deliver the most capable platform the IT industry has.

Nobody is surprised that the mainframe sits at the heart of business, processing transactions around the world, and providing the backbone to the 24x7 always-on world we have come to expect. It is estimated that 80% of the world's corporate data resides or originates on the mainframe; you might say it represents the "center of gravity" for business applications and data.

Java's popularity as a language for enterprise systems has grown steadily. And despite now being around 20 years old, eweek still ranks Java number 1 in its Top 10 Programming Languages for Job Seekers in 2014. It is a well-structured language, suitable for writing clean and robust code, where the developer can focus on the problem, take advantage of the language features, and write something that a colleague will understand decades later using the same tools that are used for writing code on distributed platforms.

Source: http://bit.ly/1hsROAE

Java also benefits from being the language of choice for many universities around the world, where it forms the basis of computer science and software engineering courses. Graduates understand it, have been trained to use it, and know its strengths. Java has an established ecosystem, which means it will not be going out of fashion any time soon. Consequently there are a huge number of open source projects providing frameworks and services for all manner of tasks. The actual number of Java programmers that exist exceeds nine million.

Plus, it performs; there might be people who can write tight assembler code to perform a very specific function faster than the equivalent COBOL or Java, but who can maintain it and for how long? The IBM Just In Time (JIT) compiler achieves some phenomenal results, and while our hypothetical assembler guru might be able to do a good job in a specific case, the JIT looks at it all and is designed to optimize code for the hardware on which it is running. As IBM invests in hardware improvements on System z, so it also enhances the JIT to take advantage of them. The result is continued performance improvements for Java applications, as IBM updates the JIT to produce more efficient code at runtime.

Some might say that Java and the mainframe did not get off to the best start, and it is certainly true that some people formed an opinion in the early years that the two were not suited to each other. You can still find these opinions in old blog posts and forums on the Internet, but they do not reflect the current state of Java on the mainframe.

It is not entirely surprising that a fledgling technology, as Java obviously was when it first came to the mainframe, did not perform as well as the long-standing applications written in COBOL to which it was being compared. Those existing applications had received years of progressive refinement and refactoring, perhaps even decades of it, so the comparison was unfavorable and some people formed a bad opinion by focusing on short-term comparisons.

Today's Java on System z runs in the IBM 64-bit JVM, exploiting features such as hardware transactional memory, large memory pages, and hardware instructions going back to the z990 introduced in 2003. It has over a decade of investment and refinement behind it; the JVM and JIT have both been enhanced in step with Java specification developments and the hardware itself has had capabilities added specifically to enhance Java workloads. IBM also introduced the z Application Assist Processor (zAAP) that runs Java workload on dedicated processors, massively reducing the long term operating cost of Java workload on the mainframe.

Comparing Java today to the Java of 10 years ago is like comparing Formula 1 racing cars that are a decade apart and asserting that the technology has not changed much in the passing years; it is not really a comparison that is worth making because they are simply not the same thing.

When and where to put Java on System z

Deciding if System z is the right place to put new Java applications has not always been the simple decision everybody would like it to be. Depending upon whom you ask, the answer can be anything from, put everything on z, or sometimes, it depends, accompanied by a long technical discussion of the pros and cons, or never. None of these answers are particularly helpful. The first leaves you wondering if the person really understands why you have asked the question, the next leaves you wondering if technology alone can provide the answer, and the last often feels like a passionate and unsubstantiated opinion.

Today, if you are asking yourself, where is the best place to deploy a new Java application, you should probably first ask yourself, "Where is the center of gravity of the existing applications and data that this new Java application will interact with?"

What do we mean by "center of gravity" when applied to technology? Imagine a typical IT environment where there are lots of disparate systems that need to interact with each other. Now consider what the typical pattern of interactions will be between the proposed new Java application and those existing systems, applications, and data. The "center of gravity" is located wherever the greatest concentration of interactions is. Positioning the new Java application on, or near, the center of gravity minimizes the architectural complexity of the solution, which in turn has a positive impact on things like response time, maintenance costs, and resiliency.

Consider a simple example. Imagine you are planning to build and deploy a new Java application. The new application will provide some entirely new capabilities, but will need to interact with the existing business services hosted in CICS that modify data hosted on the mainframe. Not only do these services provide the business function we need, but they also embody best practice and comply with our audit and governance policies, so being able to reuse them is essential.

By applying our center of gravity principle, we can quickly assess the level of interaction our new application is going to have with the existing mainframe solution. In this example, the center of gravity is clearly located in CICS on the mainframe, so what Java environment can we expect to find there? CICS TS V5 supports the IBM 64-bit JVM, providing a multi-threaded JVM server environment directly in CICS TS. If your application would benefit from some of the capabilities of the WebSphere Application Server Liberty Profile, for example, a presentation layer built using JSPs and Servlets, then you will find that the Liberty Profile is embedded in the CICS JVM server and packaged without charge as part of CICS TS V5.

A counter example would be a new Java application that interacts with one or more non-mainframe systems, such as a distributed DB2 database perhaps in combination with a third party packaged application running in an IBM SoftLayer public cloud. If there is little or no requirement for this new Java application to have access to applications or data residing on the mainframe, then the center of gravity for that application is clearly not going to be on the mainframe. In this example, the question to ask might be whether the center of gravity is nearer the distributed DB2 database or the cloud based third party application. Either way, WebSphere Application Server has deployment options in both of these environments.

The center of gravity principle works because no matter where the center of gravity is, there is an enterprise grade Java environment there to accompany it. The goal of the center of gravity principle is not to define an absolute answer, as there might be additional considerations that steer you in another direction, but to consider options that are congruent with past business decisions, and that recognize the broad range of Java capabilities available to you irrespective of their location.

Defying gravity

While the center of gravity principle helps to keep our decision making process grounded, there are some specific cases where following the principle without question might not be optimal. The most likely case where this can occur is in a mixed platform environment, where both System z and distributed servers co-exist. In mixed platform environments, the center of gravity might appear to be located in the distributed server environment, even if it has interactions with System z based services, so the logical conclusion would be to locate a new Java application somewhere in the distributed environment. Is this the right answer?

To be clear, we are not suggesting that this is a wrong answer, but it is worth stopping to consider the economics of growing a distributed server environment, particularly when System z is also present. The case for consolidation onto System z, in particular zLinux, is incredibly strong, as it breaks the economic pattern of incremental cost associated with each new distributed server.

zLinux virtualization enables new instances to be added to the environment quickly and easily, without needing to worry about floor space, power, cooling, cabling, physical maintenance, and much of the cost associated with the introduction of a physical device. Moreover, if your organization is motivated by the economics of a cloud environment, but concerned about migrating sensitive data to the cloud, then the combination of zLinux and z/OS can form the basis of an efficient private hybrid-cloud environment.

The economic argument extends further. In the same way that System z provides zAAP processors dedicated to running Java workloads, it also provides the Integrated Facility for Linux (IFL) processors. These are dedicated to running Linux on System z, providing cost and scalability benefits. Consolidation onto System z also offers advantages from the co-location of applications and data, reducing network latency by taking advantage of System z HyperSockets and simplifying the application architecture. IBM z/VM® provides an incredibly efficient virtualization environment, allowing you to reach near 100% processor utilization, even in mixed workload environments, which drives down the overall cost per transaction.

Managing costs, in particular operational expense, is a top priority for all organizations. Consolidating distributed server environments can directly improve operational expense, but if the economics for consolidation do not apply to your environment (perhaps you have already consolidated), there are still options to consider for managing operational expense on System z when deploying new Java applications onto the platform. These options, discussed throughout the remainder of this book, look at how CICS Transaction Server Value Unit Edition can be used to deploy new Java workloads using a one-time-charge pricing model.

Solution overview

The CICS Transaction Server Value Unit Edition can be deployed to address some typical business challenges, ranging from mobile enablement of existing applications, through to the development of a modern, hybrid, batch environment. All these examples share a common trait, they are new workload scenarios and they can be deployed into CICS TS VUE.

We begin with a review of the WebSphere Application Server Liberty Profile, packaged with CICS Transaction Server, and consider how Liberty can be used to modernize existing interface technologies. Liberty provides capabilities such as JSPs and Servlets, which offer developers familiar with web technologies the ability to build rich web-based interfaces that interact with existing CICS TS applications and services. For many customers, these back-end applications represent the cumulative investment in business process, governance, and audit requirements, and represent core business value in the form of intellectual property and competitive advantage. For many, these are irreplaceable, so providing new methods to interact with these CICS TS services enables businesses to continue to exploit their existing investment, and to do so using modern interfaces, languages, and capabilities available in the Liberty profile.

Associated with this Solution Guide, we recommend that you download and view the IBM Redbooks publication, An Architect's Guide to New Java Workloads in CICS Transaction Server, SG24-8225-00. In this publication, you will be provided more insight into the following activities:

- Using CICS Liberty JVM server
- Configuring Liberty in CICS TS
- Developing new application interfaces using a CICS Liberty JVM server
- Porting JEE applications into a CICS Liberty JVM server

Also mentioned in the IBM Redbooks publication, An Architect's Guide to New Java Workloads in CICS Transaction Server, SG24-8225-00, is how CICS TS can provide mobile enablement of existing back-end services, and the integration of IBM Operational Decision Manager and Modern Batch Processing. Again, download the book to get a more in-depth perspective as well as information about how to build, configure, and deploy these services into CICS TS VUE.

Usage scenarios

IBM CICS Transaction Server for z/OS Value Unit Edition runs on a zNALC enabled LPAR on z/OS. In order to deploy workload into CICS TS VUE, it must meet some simple criteria to demonstrate that it is not new Java workload, in accordance with zNALC pricing conditions outlined on the IBM website (http://www-03.ibm.com/systems/z/resources/swprice/mlc/znalc.html), or is a qualifying application, also described on the IBM website.

A range of technical capabilities can be deployed into CICS TS VUE. You will find them in section 1.5 of the IBM Redbooks publication, An Architect's Guide to New Java Workloads in CICS Transaction Server, SG24-8225-00, Note that the list is not exhaustive, and there are many other usage scenarios. However, there are some common reasons why an organization might consider deploying new workload onto VUE, and it is these that are briefly considered as "usage scenarios."

For most businesses, a large proportion of what they are doing with their IT budget can be simply categorized as either focused on increasing service agility, or driving operational efficiencies, solutions that get things done quicker, or cost less. CICS TS VUE provides both, offering a reduction in cost for new workloads, while providing capabilities that enable customers to get things done faster.

For those looking at cost savings, one obvious consideration is *consolidation*, be it that of the physical servers, or the architectural software components. Both carry a cost. As an example, consider an organization that has progressively developed an application in CICS. Such an organization might have initially developed a terminal based application, eventually replacing this with a fat-client solution running on distributed servers, perhaps also building some applications using web technology hosted on web servers and interacting with CICS applications.

The concept of consolidation can be applied in many ways to this example, bringing various benefits, depending on exactly what is being consolidated, for example:

- Consolidation to reduce architectural complexity and therefore maintenance cost
- Consolidation to reduce operational expense of managing large data centers
- Consolidation to reduce network latency and improve response times
- Consolidation of "legacy" interface layers to increase productivity and introduce new interactions

Prior to the introduction of CICS TS VUE, these examples, while valid, might have faced challenges when looking at the financial factors involved. This no longer has to be the case, as VUE changes the pricing dynamic, allowing for a fixed price to be applied to such projects. Using VUE, a customer who had the circumstances described above would be able to benefit from all of these consolidation options; minimizing architectural complexity, removing unnecessary hardware, improving the response time of applications, and exploiting new languages and technology to create rich mobile and web applications.

The second type of IT budget spending is directed towards *service agility*; doing things faster. The latest features of CICS TS, particularly those offered by the inclusion of the Liberty Profile and the mobile capabilities of CICS TS, are well suited to organizations that want to build dynamic interactions that exploit the applications and services already present in existing CICS applications. VUE enables customers to consider building these new workloads using all the features of CICS, integrating with existing CICS applications and services already in production, yet do so with a degree of separation and financial security, pricing the project to exploit capital budgets and rapidly deploy new applications that exploit the existing skill base of development and operations staff.

Consider another customer, again with an investment in CICS applications, who wants to build a "next generation" mobile application to replace their existing mobile solution. In this example, the customer has a broad range of services hosted in CICS, and wants to provide a simple interface to these services that does not require additional servers in the datacenter, is built using popular programming models for mobile development, and provides a light-weight interface for operations staff to track and monitor key performance metrics for the application.

For this customer, service agility might provide the following capabilities:

- RESTful APIs for accessing services from mobile devices, making integration quick and easy for application development
- Java servlets and JSPs to create web-based performance metrics viewable in a web browser, making the deployment of interface updates quicker and managed centrally
- A structured approach to packaging and managing application updates, so that operations staff can be confident when deploying new changes, thus speeding up deployment times

In this example, prior to CICS TS VUE, a customer who saw value in the Liberty, mobile, and applications packaging capabilities of CICS would have had to migrate their existing CICS installation to the latest level of CICS TS in order to gain access to these features. Using CICS TS VUE, a customer can deploy the latest version of CICS into a separate LPAR, yet connect to all the existing applications and services in their existing CICS environment, and also exploit the latest capabilities CICS has to offer. Not only does this bring all the capabilities of the latest technology, but it also allows the price of such a project to be delivered using the capital budget.

Although these two scenarios are somewhat trivial, they demonstrate that VUE is *not* simply about changing the price of a project from an operation expense to a capital expense, but that VUE offers a complete CICS solution to many different business problems.

Ordering information

Ordering information is shown in the following table.

Table 1. Ordering part numbers and feature codes

Program name	PID number	Charge unit description
CICS Transaction Server for z/OS Value Unit Edition V5.2.0	5722-DFJ	Value Unit VUE007
CICS Transaction Server for z/OS Value Unit Edition Subscription and Support V.1.10	5722-DFK	Value Unit VUE007

Related information

For more information, see the following documents:

- IBM Redbooks: *An Architect's Guide to New Java Workloads in CICS Transaction Server*, SG24-8225
  http://www.redbooks.ibm.com/abstracts/sg248225.html
- CICS Transaction Server for z/OS Value Unit Edition product page
  http://www-03.ibm.com/software/products/en/cics-ts-vue/
- IBM Offering Information page (announcement letters and sales manuals):
  http://www.ibm.com/common/ssi/index.wss?request_locale=en

  On this page, enter `CICS Transaction Server for z/OS Value Unit Edition V5.2.0 or 5722-DFJ`, select the information type, and then click **Search**. On the next page, narrow your search results by geography and language.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document was created or updated on November 6, 2014.

Send us your comments in one of the following ways:

- Use the online **Contact us** review form found at:
  ibm.com/redbooks
- Send your comments in an e-mail to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

This document is available online at http://www.ibm.com/redbooks/abstracts/tips1182.html .

Trademarks