Microsoft

Move mainframe TP monitors to Azure

By Larry Mead
Azure Global Engineering

June 2020

Contents

Authored by Larry Mead. Edited by Nanette Ray. Reviewed by Azure Global Engineering.

Overview

Many corporations and public agencies are looking for an easy path from their existing mainframe applications to the cloud. This path often starts in the world of transaction processing (TP) monitors, a series of models used by mainframe computers to monitor transactions from end to end. Also known as *transaction managers*, TP monitors represent an approach to high-speed online transaction processing (OLTP) that has powered banks, insurance companies, and governments for decades.

At a glance, these industrial-strength systems might look like the opposite of cloud-native architectures. However, this guide explains how mainframe workloads based on TP monitors can run in the cloud. Getting started is easy when you set up an emulation environment on Azure.

Financial institutions are among the biggest users of OLTP systems, which can process millions of updates daily for huge numbers of users. Applications based on these systems typically run on IBM mainframes that do two things particularly well—they host software to process transactions, and they display forms for user input onscreen.

For example, CICS (Customer Information Control System) is an IBM server originally developed in the 1960s to support high-speed transactions with well-defined application interfaces. Information Management System Transaction Manager (IMS TM) is one of the earliest mainframe databases and was originally developed for the Apollo space program. CICS and IMS are both types of TP monitors.

Programs that use TP monitors on the mainframe are typically developed as *reentrant*, or single user interaction, applications. That means they perform a single operation, or transaction, and then exit and are subsequently invoked again when another input for the program is received. Within that single operation, a number of functions can be performed, such as receiving input from a device, updating a database, and returning output to the device.

Mainframe OLTP systems still work, which is why they remain in use across the Fortune 500. But today's organizations want the benefits of cloud computing, including:

- Enterprise-grade security and governance on Azure.
- Virtually unlimited scale and elasticity.
- Lower total cost of ownership of x86-based cloud platforms. In some cases, the cost is ten times lower than a similar z/OS environment.
- Powerful analytics and AI workflows using Azure Cognitive Services, Azure Databricks, Azure Machine Learning, and other platform as a service (PaaS) options.
- Cloud-native development using Azure DevOps services that support a continuous integration (CI) and continuous delivery (CD) experience.
- The ease of managed services, such as Azure Monitor and Azure Cost Management, for managing and maintaining their data estate.

Azure Global Engineering works with the world's top public and private sector organizations to modernize their legacy platforms. The team has seen firsthand how a lift-and-shift migration of OLTP systems from mainframes to TP monitors on Azure can be a key step in IT modernization

efforts.

This guide tells you how to use TP monitor emulation software on Azure to run even the most complex CICS applications in the cloud. The emulation environment creates all the interfaces that the application expects to see, so it can run unchanged from within a virtual machine (VM) on Azure.

This guide is software agnostic—that is, it doesn't recommend any particular emulator. Several vendors offer TP monitoring emulators that meet different requirements. This guide can help you ask the right questions.

You may also want to work with a Microsoft Certified Partner as you plan your cloud migration. For more information, see Migration partners.

What is CICS?

CICS is the key to high-volume OLTP in mainframe applications, but unless you've worked with IBM mainframes, you might not have heard of it. CICS refers broadly to a family of application servers that was pioneered by IBM in the 1960s to solve the problem of handling multiple concurrent transaction threads. Today, people may use CICS to refer to CICS Transaction Server, a modern web-aware platform from IBM.

You can think of CICS as one of the earliest virtualization solutions, since it creates more capacity from a finite set of hardware resources. The CICS platform also gives applications well-defined interfaces for transaction management and user interaction. As a virtualization solution that encapsulates interfaces, CICS may sound a bit like a containerized application environment. In this type of container, the TP monitor handles the screens, user session state, user interactions, database, and other resource connections, such as queuing. TP monitors are a bit like web servers with transaction capabilities and screen handling built in.

Because of the abstraction layer that CICS provides for application developers, they are free to focus on the application logic rather than on the complicated process of communicating with a user and a database. Originally, programmers interacted with CICS through specific programming language extensions that supported transactions and provided access to back-end databases.

The "green screen" applications of the 1970s and later show how users interacted with these applications. With the advent of distributed computing, programmers could create interfaces that allowed web services to interact with CICS—and allowed users to say goodbye to the green screens.

When working with emulation environments on Azure, it helps to understand the following CICS concepts:

- **Screen handling.** Screen handling in CICS is based on device-independent screen definitions that developers create as Basic Mapping Support (BMS) instructions. BMS is an application programming interface between CICS programs and terminal devices, and it provides a common data format to present to programs.
- **Transactions.** CICS doesn't support message-based sessions. Nearly all CICS programs run as a single unit of work—a *transaction*—and pass state between user interactions through a special data area called the COMMAREA option. When a series of programs are invoked by CICS, the COMMAREA enables the next program to pick up where the last one left off.

- **Scalability.** To add capacity to a system, you can scale it up—that is, add processor speed—or scale it out. IBM mainframe applications scale out using a cluster, called a Parallel Sysplex, which connects multiple Central Electronics Complex (CEC) units. The CEC units are like building blocks containing the core CPU, RAM, and other components. CICS can be deployed in a Parallel Sysplex environment. This provides a specialized type of cluster, called a CICSplex, and lets you scale workloads.
- **Data.** Today, many mission-critical applications based on CICS are still in use in the financial services sector, the government, the healthcare industry, and elsewhere. Data in these systems is typically formatted as Virtual Storage Access Method (VSAM) files—another format that's been around for decades. VSAM works like a record-based file system and can be used in conjunction with TP monitors, like CICS and IMS.

What is IMS TM?

Like CICS, IMS TM is a high-performance transaction and resource management software system from IBM. Originally developed by the Rockwell Space Division at NASA, IMS TM provides an application interface to mainframe-based resources, such as databases, telecommunication devices, and messaging systems. It uses a very efficient message queue approach to handle the input and output of various resources and devices. For this reason, it continues to be used in high-volume transaction systems.

Over the years, IMS TM has evolved to support modern languages, such as Java, and web-based input devices. Although not as widely used as CICS, IMS TM continues to be used across many industries.

The following concepts apply when running an emulation environment on Azure for an IMS TM application:

- **Screen handling.** IMS TM provides a device-independent interface definition layer through the Message Format Service (MFS) interface and a concept called *distributed presentation management* (DPM).
- **Scalability.** Like CICS, IMS TM supports deployment with a Parallel Sysplex implementation.

How CICS and IMS TM differ

Today, CICS is much more commonly used than IMS TM. However, a significant number of IMS TM applications are still in use, particularly in payment processing and other high-volume transaction systems. Both systems provide many types of services, such as terminal interaction, data transfer, database query, and database updates. Both perform screen handling for 3270 terminals, store information between screen actions, and send updated screens. Both support a number of programming languages, including COBOL, PL/1, Assembler, and Java.

However, their differences matter and affect the approach used when migrating applications based on z/OS to Azure. CICS works more like an application server for the mainframe. It runs in one or more *regions*, which is where CICS programs are hosted. It uses macros and interfaces to call services that handle device input, data IO, and transactions. By contrast, IMS TM is a message-based system that provides APIs to use services that handle input from devices, databases, and transactions.

In fact, application code written for CICS is not compatible with application code written for IMS

TM. To move an application from one type of mainframe TP monitor to the other, you have to rewrite it to fit the model of the target TP monitor.

The following table compares the key features in CICS and IMS TM.

Feature	CICS	IMS TM
System type	Application server	Message-based APIs
Device input	BMS	MFS, DPM
Data IO	DB2, IMS DB VSAM	DB2, IMS DB, VSAM
Distributed transactions	Yes	Yes
Parallel Sysplex support	Yes	Yes

TP monitors on Azure

When moving OLTP applications to Azure, emulators for mainframe TP monitors run on virtual machines, a type of infrastructure as a service (IaaS). Code for CICS and IMS TM is not interchangeable, so when you move an application to a TP monitor emulator on Azure, you need to work with a vendor that supports your specific type of TP monitor. Many vendors offer TP monitors that run efficiently on VMs within an emulation environment, as Figure 1 shows. The emulation environment:

- Runs COBOL, PL/1, and other common mainframe applications.
- Runs the TP monitor and the batch jobs that use job control language (JCL).
- Supports required data sets and access methods, including VSAM.

Other requirements are provided by Azure:

- In the data tier, the mainframe database can be replaced by a managed service, such as Azure SQL Database.
- The mainframe's system management tools are replaced by Azure services and other vendors' software running in VMs.

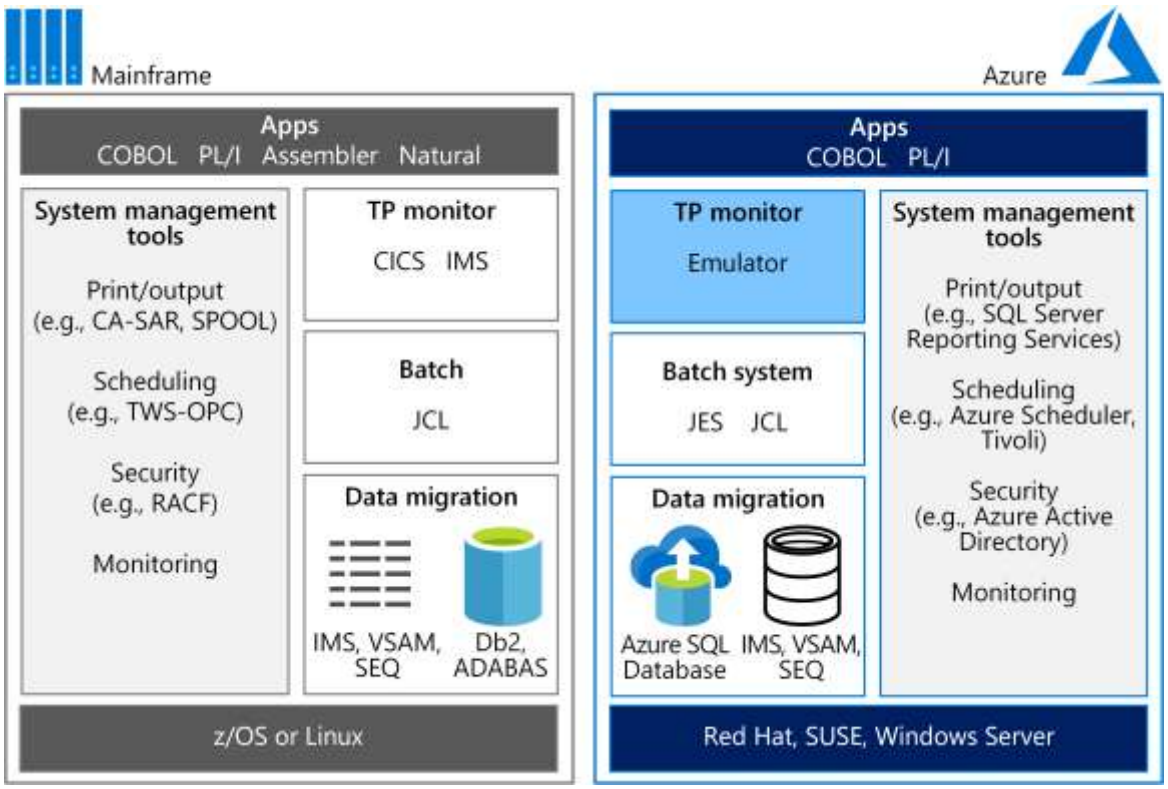

Figure 1. Side-by-side comparison of mainframe and Azure emulation environments.

Screen handling

To create a graphical user interface (GUI), CICS and IMS use a map of instructions. For example, a BMS map in CICS contains fields that prompt users to type, fields for display-only content (such as the title and header), and output fields that display the results of processing. For IMS TM, MFS or DPM does something similar. The result looks a bit like an HTML webpage. Since CICS is more common, this document focuses on BMS.

An enterprise-scale system has hundreds of screens, generally grouped by function into a mapset. An emulation environment performs the screen-handling tasks, so you don't have to create the maps. It translates the green screen instructions and returns the requested data. Emulators typically support both TCP/IP and the LU6.2 User Data communication models used by CICS and IMS.

When replatforming a legacy application, the screen-handling and form entry functionality that TP monitors provide is commonly implemented using web servers on Azure. These can be combined with database APIs, such as ADO.NET, ODBC, and JDBC, for data access and transactions.

Scalability on mainframes and Azure

When you move a mainframe application to Azure, you need to consider scalability, since mainframes and Azure handle this very differently.

For example, say you want to migrate a highly scalable application based on CICS. The mainframe provides scalability across compute resources divided into *logical partitions* (LPARs). One LPAR

equals one virtual machine on Azure. LPARs are simply an area that's been allocated on the mainframe, and each can contain CICS regions. (Note that Azure uses the term *region* to refer a geographical area with an Azure datacenter.) Mainframe applications run within a CICS region and are designated by a four-letter transaction code, such as TX01.

By contrast, scalable architectures on Azure use very different design patterns. The most popular patterns today for highly scalable applications are based on loosely coupled microservices, which are deployed in various ways. A microservices-based design makes an application especially agile by compartmentalizing units of functionality, which can then be deployed, scaled, and updated individually. Microservices are often deployed in containers that are managed by an orchestration solution, such as Kubernetes, which runs the services on as many nodes as needed to meet demand.

Figure 2 shows several applications created in different CICS regions. The functionality that a region provides is similar to a node in Azure Kubernetes Service. A specific application, such as TX03, might run as a microservice in a specific Kubernetes *pod*. A Kubernetes pod can contain one or more CICS transactions.

The benefit of this approach is that you can choose the level of granularity for a deployment. Do you want multiple CICS transactions per pod—or just one? The choice is yours. At the data layer, CICS serves as the abstraction layer for mainframe databases, such as DB2, VSAM, or IMS. On Azure, you have many more choices for the database tier.

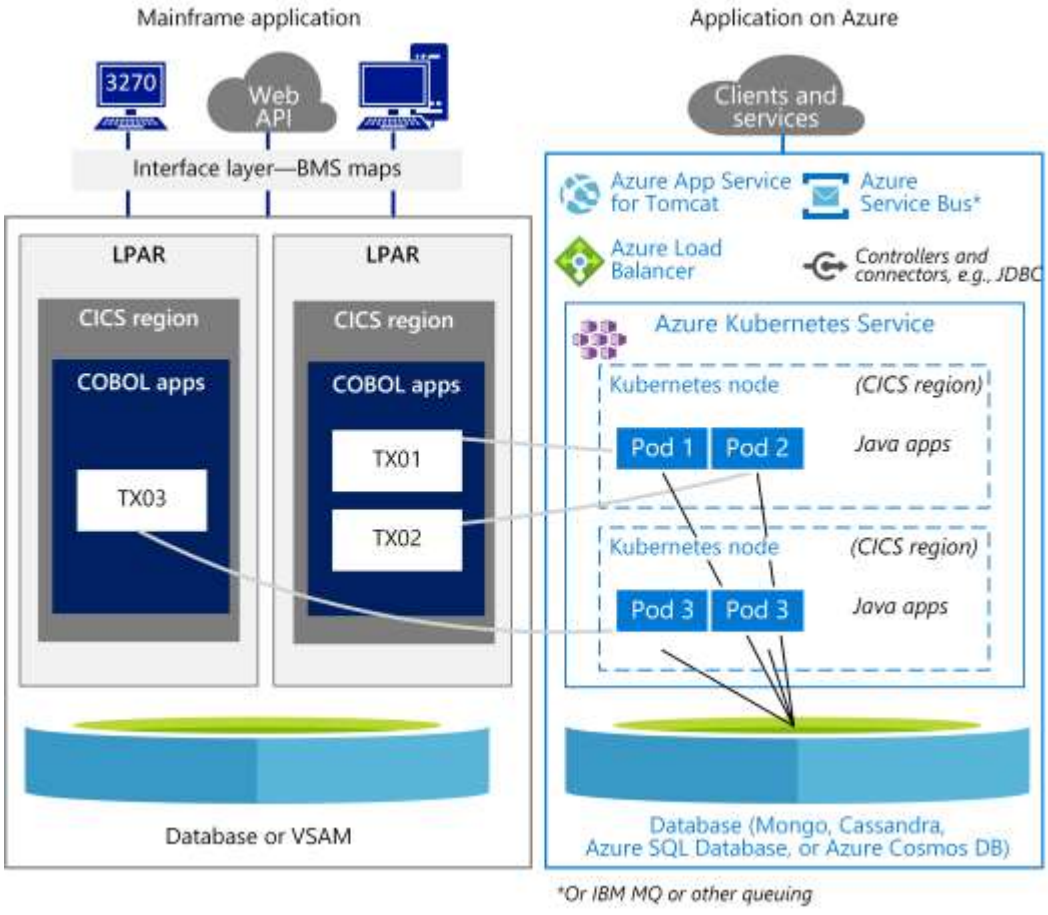

Figure 2. Comparing CICS regions to Java applications hosted in Azure Kubernetes Service.

Options for modernization

There are options for how you migrate the CICS application code to Azure. In general, the TP monitor emulation software is deployed either as a standalone runtime or as a module that is included in the development environment and the executable. When you add the module, it performs CICS for you.

The type to choose depends on whether you want to rearchitect or refactor your application when you migrate it:

- To migrate an application as is, run it in a standalone emulation environment on Azure. Standalone environments are also easy to scale up and out. You can select special-purpose virtual machines optimized for memory and compute power. You can scale out but deploy multiple instances on virtual machines and then use Azure Load Balancer to direct traffic.
- To migrate the codebase, use the code module approach, which makes it easier to convert the application to a more modern language, such as Java or .NET.

Even if you don't refactor the codebase, you can manage the existing COBOL or PL/1 application using a modern DevOps platform, such as Azure DevOps Services.

CICS systems can also be migrated to efficient container-based architectures on Azure. The goal of a migration is not just a distributed monolithic architecture in the cloud. Loosely coupled services, delivered in containers, add flexibility and resiliency to applications, and the most popular way to orchestrate these containers is Kubernetes. For example, you can run applications using Azure Kubernetes Service clusters or use Azure Service Fabric to support multiple serverless compute systems.

These services can be scaled with multiple clusters that are load balanced for greater compute power and scalability. Services can also run in more than one region and fail over to another region, if necessary.

Virtual machine considerations

The exact lineup of Azure IaaS components to use depends on the operating system you prefer and the scale and scope of the resources to be migrated. For example:

- **Windows–based VMs.** Internet Information Server (IIS), along with ASP.NET, for the screen handling and business logic. Use ADO.NET for data access and transactions.
- **Linux–based VMs.** The Java-based application servers that are available, such as IBM WebSphere Application Server and Apache Tomcat, for screen handling and Java-based business functionality. Use JDBC or other Java client library drivers for data access and transactions to such databases as Azure SQL Database or Cassandra.

PaaS considerations

Older mainframe applications were never designed for the cloud. Some level of refactoring or reengineering is usually required to make them run optimally. A migration can be a great time to replace out-of-date technologies with higher performing PaaS options.

For example, you can remove the CICS verbs in the source code with web services, such as Azure

Web Apps, and do away with the overhead of partner tools. Or you can reengineer business rules as serverless Azure Functions code or modern microservices, which decouples the code from the rules and accelerates DevOps workflows. A similar approach works for Java classes, which you can run on Azure using microservices and the container of your choice.

Summary

A lift-and-shift migration using emulation software on Azure is a quick way to modernize legacy applications. Your organization can then begin to take advantage of the massive scale and global reach of the cloud and adopt modern DevOps practices.

To go to the next step, you can rebuild applications to take advantage of efficient Azure PaaS options. A managed service, such as Azure Kubernetes Service, can support high-performance transaction processing with a cloud-native architecture based on containers or microservices.

Learn more

For more information, see the following articles in the IBM Knowledge Center:

- Introduction to CICS
- IMS Transaction Manager

In the Azure documentation, see the following:

- Azure Kubernetes Service documentation
- Mainframe application migration
- Azure migration partners