ISSN: 2754-6705

Journal of Mathematical & Computer Applications

SCIENTIFIC Research and Community

Review Article

Open Access

Containerization on z/OS: Running Applications with z/OS Container Extensions (zCX)

Chandra Mouli Yalamanchili

Software Development Engineering - Sr Advisor 2, USA

ABSTRACT

z/OS Container Extensions, or zCX in short, offer a revolutionary way to run container workloads based on Linux natively on IBM Z within the z/OS environment. This paper takes a deep dive into zCX architecture to understand its components, integration with the workload of z/OS, operational mechanics for deploying containerized applications, and unique advantages compared to standalone implementations of Linux on IBM Z.

This paper also looks into some examples of how to build and run a "Hello World" Java application and demonstrate tight integration with z/OS subsystems like CICS and DB2. The paper concludes by evaluating zCX's role in modern application development and its potential to streamline hybrid workloads.

***Corresponding author**

Chandra Mouli Yalamanchili, Software Development Engineering - Sr Advisor 2, USA.

Received: December 16, 2024; **Accepted:** December 20, 2024, **Published:** December 30, 2024

Keywords: IBM Mainframe, z/OS, z/OS Container Extensions (zCX), IBM Z, Mainframe Modernization, Containerized Workloads, Hybrid Application Development, Docker on z/OS, Linux on z/OS

Introduction

In the current age, all businesses see value in moving to the cloud infrastructure by modernizing their workloads into the cloud-native and microservices-based architecture. While the cloud infrastructure has benefits, the key challenge is ensuring these business-critical applications' security, reliability, and scalability. With the adoption of containerization technologies on IBM Z, Linux workloads can now benefit from the enterprise processing advantages of IBM Z and z/OS, such as speed, security, and reliability [1].

zCX allows z/OS programmers to use Docker-compatible Linux containers within the z/OS environment, providing unparalleled opportunities for modernizing legacy workloads while retaining all the strong benefits of the IBM Z platform [2].

This paper will address the key questions: How does zCX integrate with z/OS subsystems? What differentiates zCX from zLinux? This paper tries to point out the potential of zCX for hybrid workload modernization using practical examples and comparisons.

zCX and Its Benefits

The IBM z/OS Container Extensions (zCX) feature introduced with z/OS 2.4 enables clients to deploy Linux-based applications as Docker containers on z/OS as part of a z/OS workload on the same LPAR. IBM zCX expands and modernizes the z/OS software ecosystem by allowing applications and workloads built for Linux on Z to run on z/OS and packaging them as Docker images. Docker is a framework that packages the application and its dependencies, resulting in Docker images. Applications are deployed in the target infrastructure using these Docker images. The zCX environment supports any Docker image built for IBM Z or s390x [2].
The picture below shows the different components of the zCX instance running in a z/OS LPAR at a high level.

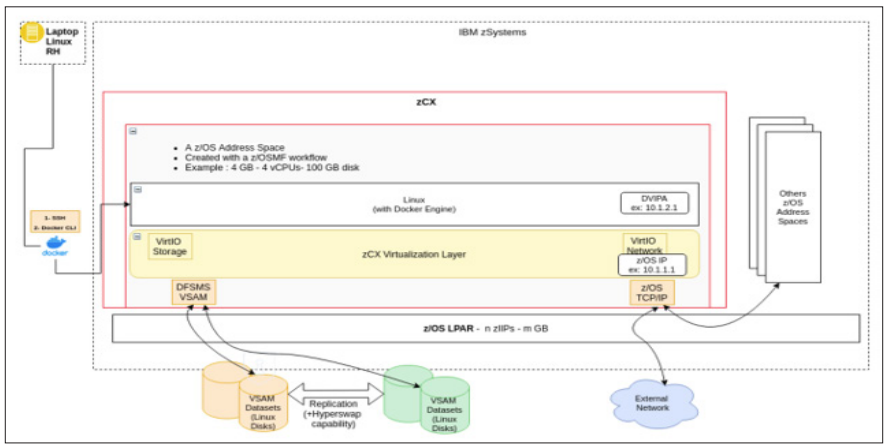

Figure 1: Overview of a zCX Instance Running in a z/OS LPAR. [3] .

As part of the zCX solution, IBM provides the tailored Linux kernel and Docker Engine that supports the Docker CLI and the containers that run the Docket images. IBM also provides the virtualization layer that bridges the communication between the Linux kernel and the z/OS, allowing the applications built for Linux to run seamlessly [2].

IBM zCX instances run like started tasks within zOS that can be provisioned and managed through the z/OS Management Facility (z/OSMF). System administrators can use z/OSMF workflows to automate the life cycle management of zCX instances. Once provisioned, the application developers or App Ops can control the Docker containers running within zCX like the standard Linux-based Docker implementation using the Secure Shell (SSH) or Docker command line interface (CLI) [1].

In later releases of z/OS, IBM has released zCX for Red Hat OpenShift, which supports RedHat OpenShift's enterprise-level container orchestration and management capabilities to z/OS through zCX foundation for Red Hat OpenShift [4].

The picture below shows how the zCX for Docker and zCX foundation for Red Hat OpenShift differ concerning the building blocks of the zCX instance.

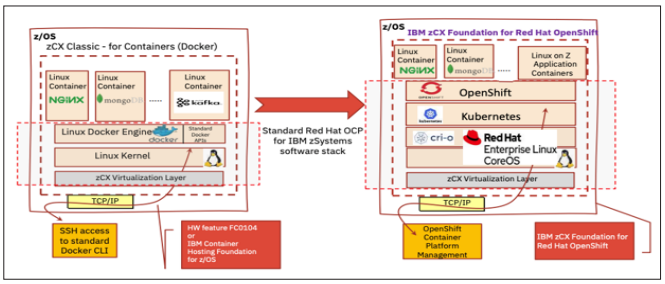

Figure 2: Visualizing the Differences in zCX Instances for Docker and Red Har OpenShift [5].

Benefits of zCX

IBM zCX allows us to bring the containerized Linux applications as close as possible to the existing enterprise z/OS applications and resources without modifying them. Below are some of the benefits available to the applications deployed to Zcx [2,4].

- Provides a tailored version of the Linux kernel that supports Docker CLI, and users can't perform most of the root operations at the Linux kernel level; this ensures the stability, security, and integrity of the Linux provided as part of zCX.
- Docker users, application developers, and App Ops can operate within zCX instances like any Linux-based environment, with minimal IBM Z or z/OS knowledge.
- Running under z/OS provides zCX access to key z/OS features like security, pervasive encryption, highly optimized parallel I/O, AT-TLS, Optimized DB2 communication, and many more.
- Co-existing within z/OS close to other address spaces like CICS, MQ, IMS, and others provides the advantage of z/OS's high-speed SAMEHOST cross-memory networking that is only available to processes running on the same LPAR.
- The same infrastructure supporting the core IBM z/OS-based workloads and the Linux-based distributed workload (running under z/OS) provides ease of operation by reducing cross-platform operational challenges.
- Applications running under zCX benefit from the z/OS Qualities of Service (QoS), giving control over workload management based on throughput and latency.
- Workloads in zCX benefit from high availability and Disaster Recovery (DR) planning through features such as IBM HyperSwap, storage replication, and IBM Geographically Dispersed Parallel Sysplex (GDPS).
- Applications within zCX can also benefit from z/OS workload management capabilities for capacity planning and tuning.
- zCX supports various languages and tools, which is ideal for hybrid cloud-native applications and modernization efforts.

zCX vs. x86 Docker Containers

IBM has evaluated zCX and x86 using a monolithic application and a micro-services version of the same application, and zCX proved beneficial both from a processing and cost perspective.

Below are the findings from IBM testing related to throughput or TPS between zCX and x86:

- Java monolithic banking application simulation using z/OS data on a z15 T01 in an IBM zCX environment could deliver on average, 44% more transactions per second per core versus a compared x86 Docker container SSL environment using z/OS data [1].
- Java microservices banking application simulation using z/OS data on a z15 T01 in an IBM zCX environment could deliver on average, 54% more transactions per second per core versus a compared x86 Docker container SSL environment using z/OS data [1].

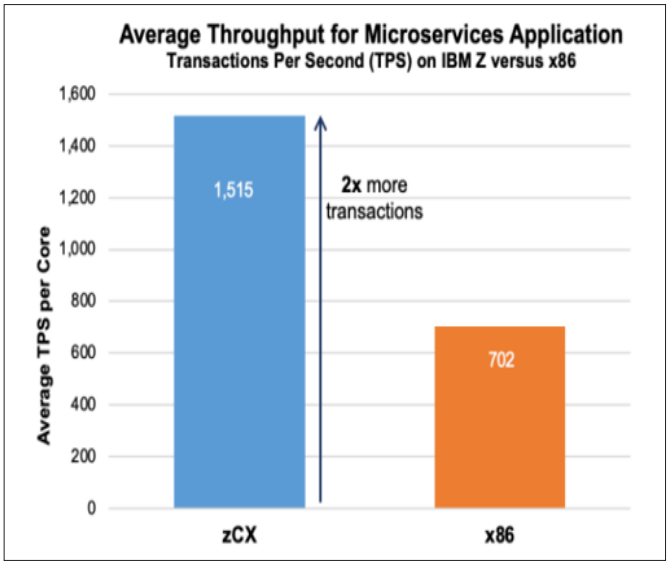

Figure 3: Average Throughput Comparison of Microservices Application Deployed in zCX Versus Docker Containers on x86 [1].

Along with high throughput, using zCX also brings huge cost savings for organizations due to the zIIP offload of workloads on zCX.

IBM also examined the acquisition and operating costs for the zCX and x86 setups discussed above, and IBM found that zCX delivers a 43% cost reduction for microservices applications and a 69% reduction in total ownership cost for the monolithic applications. Below is the picture depicting the cost comparisons for both setups [1].

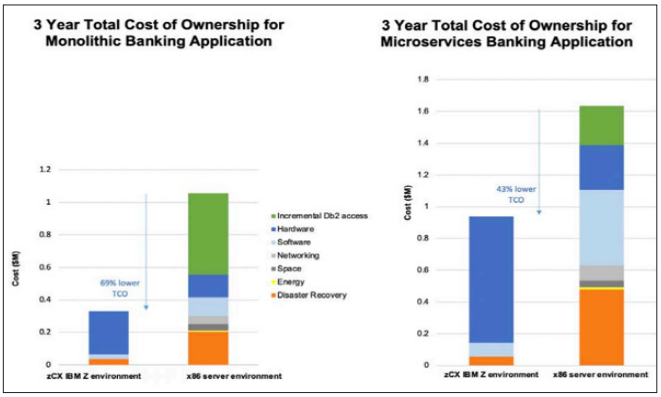

Figure 4: Three Year Total Cost Comparison Between zCX and x86 Infrastructure for Monolithic and Microservices-Based Banking Applications [1].

zCX vs. zLinux

There are other IBM Z offerings like zLinux, LinuxONE, z/VM, and others that organizations can use to host Docker containers and benefit from the high processing capabilities of IBM Z systems. Below are some differences between zCX and zLinux in different areas:

- **Integration with other z/OS Subsystems:** zCX provides close integration with other subsystems due to the 'SAMEHOST' networking feature of z/OS without the need to traverse through network adaptors or wires. In contrast, zLinux would have to integrate with any z/OS subsystems through the network layer.
- **Environment:** In the case of zCX, IBM provides a basic tailored version of Linux Kernels that supports the Docker engine and other commands needed to support Docker CLI where, as zLinux offers a full Linux distribution that functions similarly to non-Z Linux environments.
- **Management/Performance/Security:** z/OS tools help manage zCX and benefit from z/OS WLM and I/O efficiencies. In contrast, Linux-native tools manage zLinux, and zLinux uses the high performance of Z architecture and Linux-native security.
- **Processors:** zCX uses zIIPs and CPs for processing, whereas zLinux uses IFLs for processing.
- **Use cases:** zCX is best suited for Hybrid z/OS-Linux workloads where the Linux application needs to interact with other z/OS subsystems for transaction processing or data access. zLinux is best suited for hosting standalone Linux applications that benefit from high-performance IBM Z architecture.

zCX Setup Considerations

This section explores the different steps in setting up zCX for Docker and zCX for OpenShift.[2] [6].

- **Requirements:** Organizations can experiment with zCX Docker and zCX for OpenShift products for 90 days and 60, respectively. Once the trial period is complete, organizations must enable the respective products to continue using the zCX environment [1-2].
- **zCX Management:** The z/OS Management Facility (Z/OSMF) interface is used to provision, de-provision, and maintain zCX instances. System administrators must document All zCX-related input variables in the z/OSMF zCX workflow variables file to use z/OSMF to manage zCX instances [2].
- **Resource Considerations**
- **CPU:** System administrators must allocate the virtual CPUs needed for each zCX instance. Each virtual CPU is a z/OS dispatchable task within the zCX address space. zCX virtual processors can be dispatched on zIIPs or CPs based on WLM policies and CPU availability, allowing z/OS to use either zIIP or CP CPU. IBM recommends that the combined virtual CPUs across all zCX instances should not be greater than the number of zIIPs or CPs available to the particular z/OS system[2].
- **Memory:** Coming up with the optimal real and swap memory needed for zCX is critical to ensure we get the optimal I/O benefits from Linux and z/OS by avoiding memory swaps done by Linux. The zCX environment needs 1 GB of memory to work, so the total memory allocated should be at least 1 GB higher than the total virtual memory required for all containers running concurrently within zCX. IBM recommends that for the zCX instances with less than 8GB of real memory, use a 1:1 ratio of real to swap memory, and for zCX instances greater than or equal to 8 GB, use a 2:1 ratio of real to swap memory [2].
- **Storage (DASD):** Each zCX instance uses multiple VSAM linear data sets (LDS) that the z/OSMF workflow allocates during provisioning for its exclusive use. System administrators must provide high-level qualifiers and storage class details in the z/OSMF workflow variables [2]

- **Networking:** z/OS Communications Server provides network communications and network-related services like AT-TLS for the zCX workloads. z/OS communication server identifies each zCX instance through a unique application instance DVIPA called a zCX DVIPA. The system administrators must add zCX DVIPAs to TCP/IP using VIPARange with the zCX parameter. This DVIPA zCX definition enables the EZAZCX interface within the TCP/IP stack, allowing zCX containers to communicate with other subsystems running under the same z/OS system through cross-memory instead of taking network path [2].
- **Number of Containers:** zCX can host up to 1000 active docket containers, including zCX Docker CLI SSH containers that are always active [2].
- The rest of the standard z/OS features, like capacity management or resource capping through WLM, security or user management, z/OS started task support functions, and workload distribution through Sysplex Distributor, are all available to be configured and used to support the requirements of zCX container workloads [2].

Example Docker Image Setup on zCX

Below is a simple example that explores different steps to deploy a simple Java application Docker image onto zCX.

1. Getting Started with zCX
- zCX runs as a z/OS address space, providing a secure environment to execute containerized workloads. To access the instance of zCX:

a. Enable resources for zCX via the z/OSMF workflows.
b. Start the instance of zCX as a standard z/OS subsystem.
c. Connect to the zCX appliance using SSH, where the user can execute Docker commands.

2. Running a Docker Container in zCX
- For example, deploying a simple "Hello World" Java application:

a. **Application Preparation:** In this example, create a simple Java application and build and push a Docker Image to the registry that is reachable from zCX.

```
public class HelloWorld {
 public static void main(String[] args) {
 System.out.println("Hello, World from zCX!");
 }
}
```

b. **Deploy the Application:**

- SSH into the zCX instance.
- Pull the Docker image from the registry.
- Run the container:

```
docker pull <registry>/helloworld-java
docker run --name helloworld-container <registry>/helloworld
 - java
```

c.Verify Output: Monitor the logs to see the application output.

```
docker logs helloworld-container
```

Integrating zCX with other z/OS subsystems/Files

- **Subsystem Access:** z/OS Container Extensions enable seamless interaction with traditional subsystems such as CICS, MQ, and DB2 by allowing modern containerized applications to leverage these capabilities through standard APIs and protocols. Applications running in zCX, for example, can connect to CICS using APIs or RESTful services exposed by CICS. This accessibility allows easy integration of modern microservices with legacy transactional systems. Similarly, zCX applications could also communicate with IBM MQ for messaging services, enabling asynchronous interactions across distributed systems. In terms of DB2, the application on zCX can use standard options to connect to the database, such as JDBC or ODBC, to query and update data while maintaining high performance and security within the z/OS environment [2].
- **File Access:** The zCX-based applications do not have direct access to the traditional z/OS files, such as VSAM datasets or sequential files. Applications running in zCX, on the other hand, can invoke APIs exposed by other z/OS subsystems, such as CICS or Java applications running in z/OS address spaces, which handle the actual file I/O operations. This approach allows zCX workloads to leverage legacy file systems while maintaining security, scalability, and proper resource management [2].

Conclusion

zCX provides a powerful platform for modernizing applications on z/OS using containerized workloads, integrating well with mainframe resources. By simplifying the adoption of modern development practices and leveraging the strengths of the IBM Z platform, zCX empowers developers to deliver innovative solutions without compromising performance or security. Examples such as the "Hello World" application and CICS integration provide programmers with an insight into what zCX is capable of in hybrid workload modernization.

References

1. IBM Corporation (2021) Ready for the Cloud with IBM z/OS Container Extensions. IBM https://www.ibm.com/downloads/documents/us-en/10a99803f62fd90d.
2. IBM Corporation (2024) About the z/OS Container Extensions content solution. IBM https://www.ibm.com/docs/en/zos/3.1.0?topic=collections-zos-container-extensions.
3. IBM Corporation (2023) Containers for z/OS System Programmers - PART II : Interview of z/OS System Programmers: “Traditional” and “Containers” tasks. IBM https://community.ibm.com/community/user/ibmz-and-linuxone/blogs/sebastien-llaurency/2023/07/12/containers-for-zos-system-programmers-part-i-intro?communityKey=2d6a0d68-f239-4ad4-ae69-207c63ff4b61.
4. IBM Corporation (2022) Red Hat OpenShift Container Platform for IBM zCX”. IBM https://www.redbooks.ibm.com/redbooks/pdfs/sg248528.pdf.
5. IBM Corporation (2020) When & Why one would deploy Red Hat OpenShift Cluster on systems and/or on IBM zCX Foundation for OpenShift. IBM https://www.ibm.com/support/pages/system/files/inline-files/When%20and%20Why%20one%20would%20deploy%20Red%20Hat%20OpenShift%20Cluster%20on%20zSystems%20or%20on%20IBM%20zCX%20Foundation%20for%20OpenShift-fin.pdf.
6. IBM Corporation (2024) zCX Foundation for Red Hat OpenShift 1.1.0. IBM https://www.ibm.com/docs/en/zcxrhos/1.1.0.