TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Fakultät Informatik

Enablement of Kubernetes Based Open-Source Projects on IBM Z

Bachelorarbeit im Studiengang Informatik

vorgelegt von

Sarah Julia Kriesch

Matrikelnummer 303 6764

Erstgutachter:	Prof. Dr. Ralf-Ulrich Kern
Zweitgutachter:	Prof. Dr. Tobias Bocklet
Betreuer:	M.Sc. Alice Frosi
Unternehmen:	IBM Deutschland R & D GmbH

Hinweis: Diese Erklärung ist in alle Exemplare der Abschlussarbeit fest einzubinden. (Keine Spiralbindung)

TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: Vorname: Matrikel-Nr.:

Fakultät: Studiengang:

Semester:

Titel der Abschlussarbeit:

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum, Unterschrift Studierende/Studierender

Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☐ genehmige ich, wenn und soweit keine entgegenstehenden Vereinbarungen mit Dritten getroffen worden sind,

☐ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigefügt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Ort, Datum, Unterschrift Studierende/Studierender

Kurzdarstellung

Kubernetes ist eine Container-Plattform zur Orchestrierung mit unterschiedlichen Container-Runtimes, wie Docker, CRI-O und Podman für Hochverfügbarkeits-Cluster. Es gibt Hardware-Abhängigkeiten für die Software. Deshalb sollen alle Dienste und Container-Applikationen darauf basierend auch auf IBM Z Systemen laufen können. Die meisten Kubernetes-basierten Projekte sind Open-Source-Projekte. Sie haben ihre eigene Infrastruktur für automatisierte Tests mit Continuous Integration. IBM Z Hardware kann mit QEMU/libvirt als Hypervisor emuliert werden. So kann der Quellcode immer getestetet werden, so wie es auch schon für die Architektur x86 durchgeführt wird. Außerdem werden auch Fehler schneller erkannt. Das erleichert es den einzelnen Communities Software für spezielle Hardware zu entwickeln. Die Emulation soll in sämtliche CI/CD-Umgebungen der unterschiedlichen Open-Source-Projekte integriert werden können. Mainframes, wie IBM Z, verwenden die s390x-Architektur. Normalerweise haben Open-Source Communities keinen Zugriff für Tests zu solchen Systemen. Deshalb ist Emulation ein durchführbarer Weg diese Communities mit der Befähigung alternativer Hardware zu unterstützen.

Abstract

Kubernetes exists as a container orchestration platform with different container runtimes as Docker, CRI-O and Podman for clustering. There is some hardware dependency for the software. Therefore, all services and container applications should be able to run based on that on IBM Z systems, too. Most Kubernetes based projects are open-source-projects. They have their own infrastructure for automated test environments (Continuous Integration). IBM Z hardware can be emulated with QEMU/libvirt as a hypervisor. In this way, the source code can be tested continuously as it is for x86 architecture and can discover earlier possible bugs. That makes it easier for communities to develop software for special hardware. Emulation should be able to be integrated in all CI/CD environments by different open-source projects. Mainframes as IBM Z are using the s390x architecture. Usually open-source communities don't have access to such systems for builds and tests. For this reason, emulation is a viable way to enable those communities supporting alternative architecture. The first step is to understand their CI/CD infrastructure and how to integrate emulation in their configuration. The investigation on various open-source-projects can help to identify a common pattern how to integrate emulation. Hardware emulation requires a lot of performance. Therefore, minimal system requirements have to be analyzed for the emulation in the next step. This hardware emulation will added into the test environment of both open-source projects Kubernetes and Apache Cassandra then. The goal of this Bachelor Thesis is to apply and integrate emulation for s390x architecture into the infrastructure for various open-source projects. That can be reapplied for other open-source projects then, too.

Contents

Chapter 1.

Introduction

The main business introduced by IBM is the mainframe, well known as a Z system. It is possible to run Linux on it. There is a large community behind Linux and open source. Open source does not contain only Linux. There are different applications and other software developed by open-source communities. Mainframes have got a different hardware architecture than a home pc. The Z system architecture has got the name s390x and a default system x86. It should be possible to test hardware dependencies for s390x on x86 because Z systems are really expensive. Therefore, different Kubernetes-based open-source projects should be emulated for Z systems in the CI/CD test infrastructure by open-source projects. As the first step, the emulator will be chosen with the focus on functionality for Z systems on x86 architecture. After that, Kubernetes is installed in a Docker container. Tests should be able to be run on this system, too. That will be integrated into the emulation environment for an automated start. The CI/CD system should be able to execute all tests then.
The same will be done with the NoSQL database Cassandra for the Apache community to represent the whole system stack from Kubernetes until the application layer for container platforms. Another point are minimal systems requirements and minimal systems sizes. Here are different methods evaluated to minimize the system for emulation.
The goal of this Bachelor Thesis is to offer emulated Z systems for different open-source projects to test their software for hardware dependencies, so that it is possible to release new versions running on the hardware architecture s390x.

1.1. Container Orchestration

1.1.1. Kubernetes

Kubernetes[1] is an open-source project for container orchestration. That is well known as K8s, too. This project was started by Google. A Kubernetes cluster has at least one Master node and one Worker node for high availability. This container platform portable for private und public clouds. Kubernetes is available as a managed platform by different cloud providers as

[1] https://kubernetes.io/

same as different Kubernetes distributions exist to download or installations from scratch are possible. It is configurable with different container runtimes, as Docker[2], Podman[3] or CRI-O[4] as examples. The Container Runtime Interface (CRI) is necessary for managing container images, the life cycle of container pods, networking and help functions[Scho 19, p.16].

1.2. Mainframe Computers

Mainframe computers are large computers. Some of them are part of the Z series[5] by IBM. They are not only used as internet servers or for banking systems. They can handle large numbers of transactions in one second for e-commerce[Tane 14, p.56]. Such Z systems do not use the well known x86 architecture. They are built with s390x. This architecture has been developed by IBM. It has been introduced in late 2000 and supported by the Linux Kernel since late 1999[Bloc 19, p.15]. The traditional operating system for mainframes has been zOS. Linux is used as a base operating system for this Bachelor Thesis.

1.3. Hardware Emulation

Not everybody has access to expensive hardware or hardware with specific architecture. Software should be able to run on most important hardware architectures. The solution for Software Developers is hardware emulation. You can test based on hypervisors with the hardware emulation whether the software is running correctly. So you can run different operating systems and applications for special hardware in virtualization software. It is possible to enable other hardware architectures than the host has got.

1.4. Open Source Projects

1.4.1. And an even more important subsection

[2] https://www.docker.com/
[3] https://podman.io/
[4] https://cri-o.io/
[5] https://www.ibm.com/it-infrastructure/z/hardware/

Chapter 2.

Emulation

2.1. System Emulation

The System Emulation emulates a whole system with hardware, the operating system (with the kernel) and the user space (with application processes). It makes the VM really slow because so much will be virtualized.

2.2. User Mode Emulation

The User Mode Emulation does not emulate the whole system. It is possible to reproduce application processes in QEMU with a minimal system for a special application. This emulation type is working on a syscall level. An external Linux kernel will be built and the application can be mounted via a loaded Docker image in a hard disk image.

2.3. QEMU

QEMU is an open-source emulator available in most Linux distributions. It is most used for virtualizations with KVM and XEN, too. So QEMU is well tested and has got all necessary features for emulations. Additionally, you can emulate other architectures on different hardware. The open-source projects can use any Linux distribution as their base operating system then because QEMU is integrated as a package as default. QEMU does not emulate the whole hardware. That is only possible for the CPU. Therefore, QEMU is used for emulations in this Bachelor Thesis.

2.4. Emulation of different architectures

It is possible to emulate different architectures on another hardware architecture. The package qemu-use-static has to be installed then and the special architecture has to be registered in binfmt. binfmt_misc is a kernel module. You can register other architectures within that, that you can run multiple other architectures on a host. So hybrid virtualization approach is possible with different virtualization technologies as with QEMU and Docker.

2.4.1. Prerequisite for s390x on x86

Different software is necessary to run qemu or docker for multiple architectures. Therefore, docker and qemu should be installed. Additionally, qemu-user-static [1] and binfmt_misc [2] are important for running multi-architecture containers.

It is possible to use packages as binfmt-support and qemu-user-static by different Linux distributions, but it is recommended to use the latest possible version for s390x.

The kernel module binfmt_misc can be mounted with the following command:

```
# mount binfmt_misc -t binfmt_misc /proc/sys/fs/binfmt_misc
```

Latest stable releases of qemu-user-static can be found under https://github.com/multiarch/qemu-user-static/releases/. The release v5.0.0-2 is used for the project and downloaded with

`# wget https://github.com/multiarch/qemu-user-static/releases/download/v5.0.0-2/x86_64_qemu-s390x-static.tar.gz`

for the special version of qemu-s390x-static on x86. That is extracted to the directory `/usr/bin/` with the command

`# sudo tar -xvzf x86_64_qemu-s390x-static.tar.gz -C /usr/bin/`

then.

s390x binaries have to be registered for s390x. That is done with the following commands:

`# sudo -i`

and

[1] https://github.com/multiarch/qemu-user-static

[2] https://www.kernel.org/doc/html/latest/admin-guide/binfmt-misc.html

```
# echo ':qemu-s390x:M::\x7fELF\x02\x02\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x02\x00\x16:\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xff\
xff\xfe\xff\xff:/usr/bin/qemu-s390x-static:OCF' > /proc/sys/fs/binfmt_misc/register
```

Docker is configured to build only for the own architecture, which is x86 at open-source projects. The "Experimental" flag exists for new available features which are not ready for production. So you can build docker images for s390x on x86 then. That can be added with the following:

```
{
"experimental": enabled
} >> /etc/docker/daemon.json
```

Listing 2.1: This is an example of inline listing

After a restart of the docker daemon there should be listed this flag in the command

`# docker version`

. An alternative way is to export this flag as an environment variable in the shell with

`# export DOCKER_CLI_EXPERIMENTAL=enabled`

to enable it. Now it is possible to build docker images for s390x on x86 with

`# docker build -platform=linux/s390x -t image-example:s390x .`

based on a Dockerfile in the existing directory.

2.4.2. Building a s390x Kernel on x86

QEMU needs a built kernel to start a system. These packages are necessary:
bison, flex, gcc, gcc-s390x-linux-gnu, libssl-dev
One kernel has to be downloaded from kernel.org with the following command then:

`# wget https://git.kernel.org/torvalds/t/linux-5.7-rc7.tar.gz`

It can be extracted with

`# tar -xf linux-5.7-rc7.tar.gz`

. The Makefile is in the directory linux-5.7-rc7. Therefore the command

`# make ARCH=s390 defconfig localyesconfig`

has to be executed to create the default configuration for s390x and

`# make ARCH=s390 CROSS_COMPILE=s390x-linux-gnu- -j6`

for the compilation. The kernel has the name bzImage in the directory `arch/s390/boot/` then and can be used in the qemu command.

2.4.3. Optimized QEMU Command

Every additional device requires additional performance and time for starting the system. So the systems requirements had to be figured out that system requirements are minimal for every open-source project and for running tests on it. That counts for the number of CPUs, too.

The kernel option is receiving the path to the built s390x kernel. The option -m is available to add the minimal guest memory matching the system requirements of every open-source project. -nodefaults is deactivating default additional devices activated in QEMU. Only the console is necessary for receiving an output and debugging. So that is added as a device. Cassandra as a project does not need any network interface or parallelism. The option nographic is responsible for not adding any graphical interface. So we save system requirements. The option -smp is the minimal number of CPUs for the guest. The file system of containers can be loaded as a hard disk with the option -hda which is explained in every chapter of a special open-source project. That is the ideal option to mount a minimal file system for every application or system. `/dev/vda` is the partition name and rdinit is used for using Bash as a default shell.

```
/usr/bin/qemu-system-s390x -kernel bzImage -m 4G -M s390-ccw-virtio
    -nodefaults -device sclpconsole,chardev=console -parallel none -net none
    -chardev stdio,id=console,signal=off,mux=on -mon chardev=console
    -nographic -smp 3 -hda /data/kub-container.img --append
    'root=/dev/vda rw console=ttyS0 rdinit=/bin/bash'
```

Chapter 3.

Continuous Integration

Chapter 4.

Cassandra

4.1. Overview

4.2. Deployment

IBM is offering a Dockerfile[1] for Apache Cassandra with the latest version on Github. This file can be cloned to the system and will be built with the command

```
#  docker build -platform=linux/s390x -squash -t cassandra:s390x .
```

in the directory with the Cassandra Dockerfile. **squash** is an option to comprimize a Docker image and combine commands in a Dockerfile automatically. The prequisites for building s390x images on x86 are set duiring the emulation preparation. The command

```
#  docker images
```

has to show the registered Dockerimage with the name cassandra:390x then. It should be possible to integrate this Docker image into the qemu command. Therefore, a qemu-image will be created with an rounded given size besides of the Docker image in the

```
# docker images
```

command. So the command

```
#  qemu-img create -f raw cassandra.img 2G
```

can be used. This image needs any Linux file system because QEMU does not know the Docker file system. The image is formated with the command

```
#  mkfs.ext4 -F cassandra.img
```

then. For receiving the file system of the Dockerimage a directory with the name rootfs has to be crated and the command

```
# docker export $(docker create cassandra:s390x) | tar -C "rootfs" -xvf -
```

is exporting the docker image into the directory rootfs. Following transfers the content of rootfs into the image cassandra.img.

[1] https://github.com/linux-on-ibm-z/dockerfile-examples/tree/master/ApacheCassandra

```
mkdir /mnt/rootfs
mount -o loop cassandra.img /mnt/rootfs
cp -r rootfs/* /mnt/rootfs/.
```

Now it is possible to run the system with Cassandra:

```
# /usr/bin/qemu-system-s390x -kernel bzImage -m 40G -M
    s390-ccw-virtio -nodefaults -device sclpconsole,chardev=console
    -parallel none -net none -chardev stdio,id=console,signal=off,mux=on
    -mon chardev=console -nographic -smp 3 -hda
    /data/dockerfile-examples/ApacheCassandra/cassandra.img --append
    'root=/dev/vda rw console=ttyS0 rdinit=/bin/bash'
```

4.3. Start of the application and tests

A script has been written to start Cassandra and to run tests. Java and Cassandra have to be started here. The Icinga monitoring check has been used for experiments.

Chapter 5.

Kubernetes

Chapter 6.

Outlook

Chapter 7.

Summary

Appendix A.

Supplemental Information

List of Figures

List of Tables

List of Listings

Bibliography

[Bloc 19] B. Block, Ed. *Modern Mainframes & Linux Running on Them*, IBM Deutschland Research & Development GmbH, Chemnitzer Linux-Tage, 3 2019. https://chemnitzer.linux-tage.de/2019/media/programm/folien/173.pdf (Eingesehen am 07.02.2013).

[Scho 19] B. Scholl, T. Swanson, and P. Jausovec. *Cloud Native: Using Containers, Functions, and Data to Build Next-Generation Applications.* "O'Reilly Media, Inc.", 2019.

[Tane 14] A. S. Tanenbaum and A. Todd. *Rechnerarchitektur: Von der digitalen Logik zum Parallelrechner.* Pearson Studium ein Imprint von Pearson Deutschland, 2014.