Testing BPF on s390

1. Introduction

IBM Z are mainframe computers, which are descendants of IBM System/360 from year 1964. They are supported by the Linux kernel under the name "s390". This document describes how to test BPF in an s390 QEMU guest.

2. One-time setup

The following is required to build and run the test suite:

- s390 GCC
- s390 development headers and libraries
- Clang with BPF support
- QEMU with s390 support
- Disk image with s390 rootfs

Debian supports installing compiler and libraries for s390 out of the box. Users of other distros may use debootstrap in order to set up a Debian chroot:

```
sudo debootstrap \
  --variant=minbase \
  --include=sudo \
  testing \
  ./s390-toolchain
sudo mount --rbind /dev ./s390-toolchain/dev
sudo mount --rbind /proc ./s390-toolchain/proc
sudo mount --rbind /sys ./s390-toolchain/sys
sudo chroot ./s390-toolchain
```

Once on Debian, the build prerequisites can be installed as follows:

```
sudo dpkg --add-architecture s390x
sudo apt-get update
sudo apt-get install \
  bc \
  bison \
  cmake \
  debootstrap \
  dwarves \
  flex \
  g++ \
  gcc \
  g++-s390x-linux-gnu \
  gcc-s390x-linux-gnu \
  gdb-multiarch \
  git \
  make \
  python3 \
  qemu-system-misc \
  qemu-utils \
  rsync \
  libcap-dev:s390x \
  libelf-dev:s390x \
  libncurses-dev
```

Latest Clang targeting BPF can be installed as follows:

```
git clone https://github.com/llvm/llvm-project.git
ln -s ../../clang llvm-project/llvm/tools/
mkdir llvm-project-build
cd llvm-project-build
cmake \
  -DLLVM_TARGETS_TO_BUILD=BPF \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/opt/clang-bpf \
  ../llvm-project/llvm
make
sudo make install
export PATH=/opt/clang-bpf/bin:$PATH
```

The disk image can be prepared using a loopback mount and debootstrap:

```
qemu-img create -f raw ./s390.img 1G
sudo losetup -f ./s390.img
sudo mkfs.ext4 /dev/loopX
mkdir ./s390.rootfs
sudo mount /dev/loopX ./s390.rootfs
sudo debootstrap \
  --foreign \
  --arch=s390x \
  --variant=minbase \
  --include=" \
    iproute2, \
    iputils-ping, \
    isc-dhcp-client, \
    kmod, \
    libcap2, \
    libelf1, \
    netcat, \
    procps" \
  testing \
  ./s390.rootfs
sudo umount ./s390.rootfs
sudo losetup -d /dev/loopX
```

3. Compilation

In addition to the usual Kconfig options required to run the BPF test suite, it is also helpful to select:

```
CONFIG_NET_9P=y
CONFIG_9P_FS=y
CONFIG_NET_9P_VIRTIO=y
CONFIG_VIRTIO_PCI=y
```

as that would enable a very easy way to share files with the s390 virtual machine.

Compiling kernel, modules and testsuite, as well as preparing gdb scripts to simplify debugging, can be done using the following commands:

```
make ARCH=s390 CROSS_COMPILE=s390x-linux-gnu- menuconfig
make ARCH=s390 CROSS_COMPILE=s390x-linux-gnu- bzImage modules scripts_gdb
make ARCH=s390 CROSS_COMPILE=s390x-linux-gnu- \
  -C tools/testing/selftests \
  TARGETS=bpf \
  INSTALL_PATH=$PWD/tools/testing/selftests/kselftest_install \
  install
```

4. Running the test suite

The virtual machine can be started as follows:

```
qemu-system-s390x \
  -cpu max,zpci=on \
  -smp 2 \
  -m 4G \
  -kernel linux/arch/s390/boot/compressed/vmlinux \
  -drive file=./s390.img,if=virtio,format=raw \
  -nographic \
  -append 'root=/dev/vda rw console=ttyS1' \
  -virtfs local,path=./linux,security_model=none,mount_tag=linux \
  -object rng-random,filename=/dev/urandom,id=rng0 \
  -device virtio-rng-ccw,rng=rng0 \
  -netdev user,id=net0 \
  -device virtio-net-ccw,netdev=net0
```

When using this on a real IBM Z, `-enable-kvm` may be added for better performance. When starting the virtual machine for the first time, disk image setup must be finalized using the following command:

```
/debootstrap/debootstrap --second-stage
```

Directory with the code built on the host as well as `/proc` and `/sys` need to be mounted as follows:

```
mkdir -p /linux
mount -t 9p linux /linux
mount -t proc proc /proc
mount -t sysfs sys /sys
```

After that, the test suite can be run using the following commands:

```
cd /linux/tools/testing/selftests/kselftest_install
./run_kselftest.sh
```

As usual, tests can be also run individually:

```
cd /linux/tools/testing/selftests/bpf
./test_verifier
```

5. Debugging

It is possible to debug the s390 kernel using QEMU GDB stub, which is activated by passing `-s` to QEMU.

It is preferable to turn KASLR off, so that gdb would know where to find the kernel image in memory, by building the kernel with:

```
RANDOMIZE_BASE=n
```

GDB can then be attached using the following command:

```
gdb-multiarch -ex 'target remote localhost:1234' ./vmlinux
```

6. Network

In case one needs to use the network in the virtual machine in order to e.g. install additional packages, it can be configured using:

```
dhclient eth0
```

7. Links

This document is a compilation of techniques, whose more comprehensive descriptions can be found by following these links:

- Debootstrap
- Multiarch
- Building LLVM
- Cross-compiling the kernel
- QEMU s390x Guest Support
- Plan 9 folder sharing over Virtio
- Using GDB with QEMU