Container

April 25, 2024

Contents

1 Create your own container

1.1 Creating a loop device for the container file system

First, I've got the root filesystem of an alpine system using docker export (alpine.tar). After that I proceeded to create a file, format it, use it as a loop device, and finally mounting it.

```
dd if=/dev/zero of=alpine.img bs=1G count=1
mkfs.ext4 alpine.img
mkdir rootfs
sudo mount -o loop alpine.img rootfs
sudo chown -R rinri:rinri rootfs
tar xf alpine.tar -C rootfs
```

1.2 Isolating namespaces

The easiest way to isolate namespaces is to use a unshare tool that allows to "unshare" the namespaces between parent and child processes.

```
env - unshare -U -p -m --mount-proc -C -n -r -f -R rootfs /bin/sh
```

env - -> forget the environment of the host system
-U -> new user namespace
-p -> new pid namespace
-m -> new mnt namespace
–mount-proc -> mount /proc filesystem (implies -m)
-C -> new cgroup namespace. Can be made persistent and modified manually or by using cgroups tools (cgcreate, cgset, etc).
-n -> new net namespace (it's also possible to bridge the connection between hostmachine and a container)
-r -> maps the current user as the root user (uid 0) of the container
-f -> fork, run a child process, allows pid namespace isolation
-R rootfs -> use 'rootfs' directory as the root directory of the container

2 Benchmarking

I expect difference only in fileio

2.1 'sysbench cpu –threads=100 –time=60 –cpu-max-prime=64000 run'

CPU events/s	my	lxc
	2515.7900	2441.5000

Cpu events per second for measuring average cpu performance. No particular difference as expected.

2.2 'sysbench threads –threads=64 –thread-yields=100 –thread-locks=2 run'

total time, s	my	lxc
	10.0031	10.0028

No particular difference in thread locks as expected.

2.3 ‘sysbench memory –threads=100 –time=60 –memory-oper=write run‘

total time, s	my	lxc
	5.4010	5.3552

No particular difference as expected.

2.4 ‘sysbench memory –memory-block-size=1M –memory-total-size=5G run‘

Mem speed, MiB/s	my	lxc
	16224.8100	23120.7000

For some reason, lxc is faster. It’s most probably because of some background processes on my host machine since I didn’t change anything in cgroups.

2.5 ‘sysbench fileio –file-total-size=5G –file-test-mode=rndrw –time=120 –time=300 –max-requests=0 run‘

throughput read/write, MiB/s	my read	my write	lxc read	lxc write
	21.6300	14.4200	19.3300	12.8900

The small difference may be explained by additional security overhead of lxc.