Lab 4 — create simple container

Intro

- The idea of the work is to `clone()` process with flags enabling the separate namespaces for it, etc., to prepare rootfs image for this process to `chroot` into, to configure `cgroups`.

Features

- Rootfs is based on Ubuntu 20.04 base image. Sysbench is added to the container filesystem image through `init_container.sh`. On startup, `bash` shell is invoked.
- Container is created with its own namespaces:
  1. PID namespace (`CLONE_NEWPID`): The new process will have its own PID namespace. Processes in this namespace can only see the processes within the same namespace. The first process in this namespace is usually the init process, with a PID of 1.
  2. UTS namespace (`CLONE_NEWUTS`): The new process will have its own UTS namespace, which includes the hostname and domain name. This allows a process to have a different hostname inside and outside the namespace.
  3. Network namespace (`CLONE_NEWNET`): The new process will have its own network namespace. We create namespace within Bash script using `ip netns add` and further set it up. This means that it will have its own set of network interfaces, IP addresses, routing tables, and firewall rules, independent of the host and other processes. A pair of virtual interfaces is created: `veth_host` and `veth_container`, and assigned IP addresses 192.168.10.1 and 192.168.10.2, accordingly. The `container` binary simply joins existing namespace `container_network_ns`.
  4. Mount namespace (`CLONE_NEWNS`): The new process will have its own mount namespace. This means that it will have its own filesystem root directory and its own set of mount points, independent of the host and other processes.

```
artem@latitude:~/OneDrive/Inno/S23/TVL/container_lab (master) $ sudo ./build/container
[sudo] password for artem:
root@container:/# pwd
/
root@container:/# ps
    PID TTY          TIME CMD
      1 ?        00:00:00 container
      2 ?        00:00:00 bash
      5 ?        00:00:00 ps
root@container:/# ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
root@container:/#
```

Links

- This project on Github: https://github.com/ar7ch/lab4tv

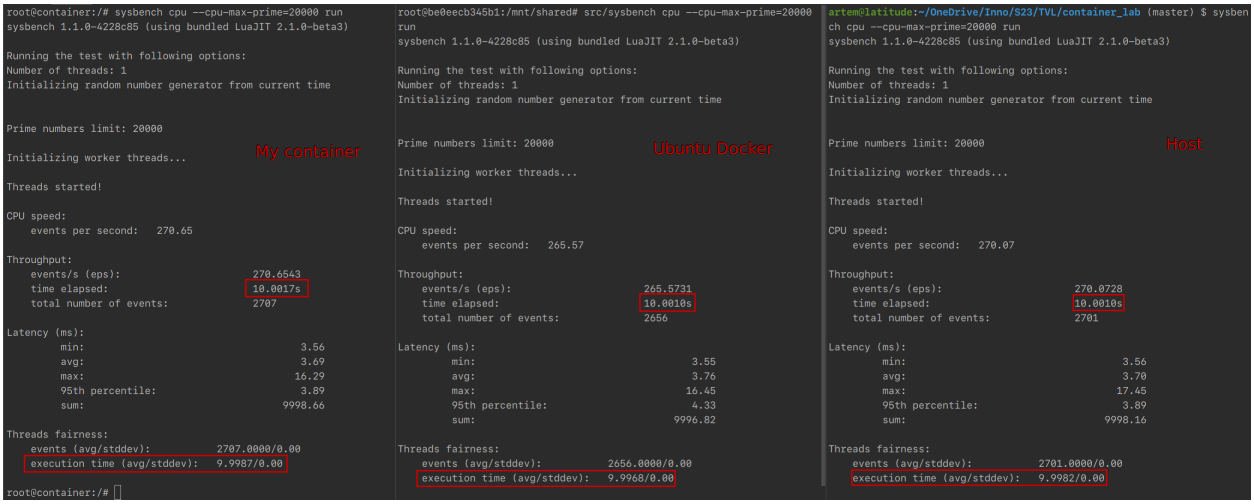

Table with metrics

	command executed	my container	Docker (ubuntu 22.04)	host machine
CPU total time	`sysbench cpu --cpu-max-prime=20000 run`	9.9899 s	9.9968 s	9.9982 s
File IO write	`sysbench fileio --file-total-size=1G --file-num=128 --file-test-mode=seqwr run`	1073741824 bytes written in 6.60 seconds (155.11 MiB/sec).	1073741824 bytes written in 5.06 seconds (202.51 MiB/sec).	1073741824 bytes written in 7.89 seconds (129.82 MiB/sec).

	command executed	my container	Docker (ubuntu 22.04)	host machine
File IO read	`sysbench fileio --file-total-size=1G --file-num=128 --file-test-mode=seqrd run`	IOPS=308201.48 4815.65 MiB/s (5049.57 MB/s)	IOPS=323803.01 5059.42 MiB/s (5305.19 MB/s)	IOPS=324396.52 5068.70 MiB/s (5314.91 MB/s)
Memory access	`sysbench memory --memory-block-size=1K --memory-total-size=4G run`	0.4148 s	0.4163 s	0.4164

Sources

1. https://man7.org/linux/man-pages/man7/namespaces.7.html
2. https://cesarvr.io/post/2018-05-22-create-containers/
3. https://github.com/akopytov/sysbench#general-syntax
4. https://docs.docker.com/storage/storagedriver/
5. https://man7.org/linux/man-pages/man8/ip-netns.8.html