Create Simple Container

Vadim Solovov

Total Virtualization

Innopolis University

2020

# Intro

The idea of the work is to create a container using linux tools like namespaces and cgroups and compare its performance with other containerization tools and with native os.

## Features

* Can work in own isolated environment
* Leaves image files with own file system after itself
* Host can reach a container
* Can be launched with one line (it is on git)

## Links

This project on Github : <https://github.com/DeadmanIQ445/ITVLab2>

# Tests

To test my container against other products and host machine I have to measure CPU, Memory, FileIO, Threading performance, because they cover almost all the metrics that are usually measured (except for network speed), because we were told so and because these guys used them already [[8](https://ieeexplore.ieee.org/document/7095802), [9](https://www.researchgate.net/publication/321957574_A_performance_comparison_of_linux_containers_and_virtual_machines_using_Docker_and_KVM)]

## commands

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Sysbench command | Why this command | What is interesting in sysbench output |
| CPU total time [sec] | sysbench --test=cpu --cpu-max-prime=20000 run | I increase cpu-max-prime because i expect to see more noticable difference between total time in different containers. Considered to be a good example, found from the following ​ source [[SysBench Example]](https://www.howtoforge.com/how-to-benchmark-your-system-cpu-file-io-mysql-with-sysbench) | total time |
| Rand Write Speed test | sysbench fileio --file-test-mode=rndwr --file-extra-flags=direct run | --file-extra-flags=direct makes sysbench not use the cache (this is the test of fileio, not memory) | Throughput:  written, MiB/s |
| Rand Read Speed test | sysbench fileio --file-test-mode=rndrw --file-extra-flags=direct run | same as Rand Write test | Throughput:  read, MiB/s |
| Threads test | sysbench threads run |  | total time |
| Memory test | sysbench memory --file-total-size=10G run |  | total time |

# Table With Metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | host machine | my container | LXC (Ubuntu 18 amd64) | Docker(Ubuntu 18) |
| CPU total time [sec] | 10.0026s | 10.0031s | 10.0029s | 10.0018s |
| File IO Write [MiB/s] | 97.63 | 80.56 | 84.14 | 65.03 |
| File IO Read [MiB/s] | 29.90 | 27.16 | 24.93 | 19.75 |
| memory access [s] | 1.0400 | 1.0422 | 1.0447 | 1.0274 |
| threads [s] | 10.0003 | 10.0001 | 10.0008 | 10.0003 |

# Explanation Why Metrics Differ

Tests of CPU, Threading, and Memory showed no difference between all the products and the host machine.

## File IO test

## However, FileIO test proved to be much more interesting. Docker is behind everyone here. And it comes as no surprise as it uses layering [[DockStorage](https://docs.docker.com/storage/storagedriver/)] and NAT[[8](https://ieeexplore.ieee.org/document/7095802)] However, docker has a slight advantage: as my container and lxc[[LXCFS](https://linuxcontainers.org/ru/lxc/manpages/man1/lxc-create.1.html)] use ext4, they needed some time before they could be benchmarked as ext4 requires “Lazy Initialization” [[EXT4](https://www.thomas-krenn.com/en/wiki/Ext4_Filesystem#Lazy_Initialization), [EXT4man](https://linux.die.net/man/8/mkfs.ext4)] which can influence benchmark results. So I didn’t have to wait some time before testing docker container.

# Sources

1. [SysBenchExample] - “How to Benchmark Your System (CPU, File IO, MySQL) with Sysbench”  
   <https://www.howtoforge.com/how-to-benchmark-your-system-cpu-file-io-mysql-with-sysbench>
2. [NSIT-USA] NSIT of the United States description of algorithm sha  
   <https://csrc.nist.gov/csrc/media/publications/fips/180/4/final/documents/fips180-4-draft-aug2014.pdf>
3. [Kim17] D.Kim et. al.“Existing Deduplication Techniques”- 2017
4. [DockStorage] docker - about storage drivers <https://docs.docker.com/storage/storagedriver/>
5. [LXCFS]<https://linuxcontainers.org/ru/lxc/manpages/man1/lxc-create.1.html>
6. [EXT4]<https://www.thomas-krenn.com/en/wiki/Ext4_Filesystem#Lazy_Initialization>
7. [EXT4man]<https://linux.die.net/man/8/mkfs.ext4>
8. <https://ieeexplore.ieee.org/document/7095802>
9. <https://www.researchgate.net/publication/321957574_A_performance_comparison_of_linux_containers_and_virtual_machines_using_Docker_and_KVM>