Create Simple Container

Maksim Surkov
Total Virtualization
Innopolis University
2019

Intro

The idea of the work is to create a small application that acts as a basic isolation layer for running programs.

Features

- Namespace isolation – programs in container will treat it as init process with pid 0
- Network bridge
- Saving filesystem as file and mounting it in container
- Cgroups limits

Links

This project on Github : https://github.com/modbrin/virt-lab-container

Tests

To test my container against other products and host machine i have to measure performance overhead of the container because that will allow to determine how much containers can be run on single machine without serious loss. Most important metrics are concerned with **latency** because general throughput in not much affected due to usage of same system components by containers. However adding abstraction mechanisms (proc namespaces isolation), some translation layers (e.g. NAT layer for network) adds up to latency increase.

commands

Metric	Sysbench command	Why this command	What is interesting in sysbench output
CPU total time [sec]	sysbench cpu --cpu-max-prime=20000 run	I increase cpu-max-prime because i expect to see more noticable difference	total time, events per second, average latency,

		between total time in different containers. Considered to be a good example, found from the following source [SysBenchExample]	
File IO Write/Read	sysbench fileio --file-total-size=18G --file-test-mode=rndwr --time=120 --max-requests=0 run	I can change file size to minimize effect of it being speed up by ram, also it provides random/sequential IO	throughput
Memory access	sysbench memory --memory-block-size=1K --memory-total-size=8G run	Allows varying block size for testing	total time, max latency
Threads	sysbench threads --threads=512 run	[SysBenchMemThread]	total time, average latency
Network	speedtest-cli	Allows for testing speed easily with same configuration (same testing server)	speed

Table With Metrics

	host machine	my container	LXC (Arch Linux x86_64 5.5.10-arch1-1)	Docker (Arch Linux 5.5.10-arch1-1)
CPU total time [sec]	10.0013	10.0015	10.0005	10.0020
CPU events per second [number]	511.87	520.26	527.00	525.83

CPU average latency [ms]	1.95	1.92	1.90	1.90
File IO Random Write [MiB/s]	11.45	8.74	10.52	9.37
File IO Sequential Write [MiB/s]	119.87	139.38	118.31	107.80
File IO Random Read [MiB/s]	165.36	159.16	158.13	143.93
File IO Sequential Read [MiB/s]	721.78	701.31	711.49	728.69
memory access 1K block total time [sec]	1.2408	1.2488	1.4672	1.4822
memory access 1M block total time [sec]	0.5270	0.5446	0.5179	0.5624
threads total time [sec]	10.3673	10.6302	10.4029	10.7325
threads average latency [ms]	264.13	282.30	285.47	320.96
network speed download/upload [Mbit]	156.36 / 299.51	-	136.76/260.58	142.00/299.91

Explanation Why Metrics Differ

Overall it's hard to measure performance and find correlations - results just seem kind of random due to such small performance overhead of containers. Difference in performance are negligible with respect to system precision error. Clear outsider in general is docker, that is because it's layering-diff mechanism requiring upper layers to be thin. Still diffs to system base provide images that are easy to create and manage but there is small overhead to it.

CPU test

CPU performance turned out to be better on containers when on host. In theory processes that run in namespaces are linked to their init process which is just regular process on host, so that may induce slowdown due to scheduler. Benchmark didn't demonstrate significant difference, all processes showed fair cpu allocation. (There are additional tests demonstrating cgroups cpu limits, they resulted in ~half performance decrease)

File IO test

Initially the tests were run with 7-GiB files and results were different, with read speeds around 6GiB/s. That is the effect of file caching as it fits into free ram. Therefore tests were changed to 18GiB and produced realistic data.
Docker with layered file system will be significantly slower, but given setup doesn't show that.
Custom container don't show difference, as tests were run on same storage just mounted by system. However if tests were run directly on image file via loop device there would be significant difference due to interfaces bottleneck.

Memory test

There are differences showing best performance on host machine and worst on docker. Privileged processes run on host are more favored by system in terms of memory allocation. However, the difference is insignificant.

Threads test

Threads test showed significant performance decrease for docker and other containers.
This is where thread scheduler is affected by not being a priority process.

Network test

Slight network performance decrease is demonstrated, this is due to using bridges and virtual network interfaces in containers.
Here docker is using host connection, but if it would be assigned subnetwork with NAT, the overhead would be significant.

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. [DockStorage] docker - about storage drivers https://docs.docker.com/storage/storagedriver/
3. [LXC] https://wiki.archlinux.org/index.php/Linux_Containers
4. [Docker] https://wiki.archlinux.org/index.php/Docker
5. [VmContainerHostComparison] IBM Research Report An Updated Performance Comparison of Virtual Machines and Linux Containers

https://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf

6. [SysBenchMemThread] Comparing performance with sysbench: memory, threads and mutexes
http://blog.siphos.be/2013/04/comparing-performance-with-sysbench-part-2/