Create Simple Container

Tasneem Toolba
Total Virtualization
Innopolis University
2022

Intro

The idea of the work is to write a simple container in C language. After running, it provides an interactive shell to communicate with it.

Features

- Separate PID namespace
- Separate UTS namespace(for networking)
- Separate network namespace
- Separate mount namespace(also, isolated file system)
- Isolated file systems supports all binaries from Alpine Linux 3.7.0
- Container filesystem in a file by using loop
- Controlling resources using cgroups

Links

This project on Github : here

Tests

To test my container against other products and the host machine, I have to measure its performance to understand the pros and cons of my solution.

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 a more noticeable difference between total time in different containers. Considered to be a good example, found from the following source [1]	Total time

Threads total time [sec]	sysbench --test=threads --thread-locks=1 --max-time=20 run	When using the --max-time option, the number to use for comparing systems is the per-request statistic.[2]	Total time
CPU total number of events	sysbench --test=cpu --cpu-max-prime=20 000 run	While older versions of sysbench limit the number of events, sysbench >=1.0 limits the total execution time by default, in which case the **total number of events** or the number of **events per second** should be used as a performance indicator instead of the total execution time [2].	Total number of events
Threads total number of events	sysbench --test=threads --thread-locks=1 --max-time=20 run	To conclude the interpretation of thread performance benchmarking, they annotate time elapsed = total time (actual time for the completion of the activity) [3]	Total number of events
File IO Write [MiB/s]	sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrw prepare sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrw run sysbench --num-threads=16 --test=fileio	In the file system I/O benchmarking, We spend time annotating and interpreting only throughput (both reads and writes) under varying loads [2], [3], [4]	Read throughput

	--file-total-size=1G --file-test-mode=rndrw cleanup		
Memory Access throughput [MiB/sec]	sysbench --test=memory --num-threads=4 -max-time=20 run	The important number to compare (given the same or similar parameters) is the throughput and operations per second [2].	Throughput
File IO Read [MiB/s]	sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrw prepare sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrw run sysbench --num-threads=16 --test=fileio --file-total-size=1G --file-test-mode=rndrw cleanup	In the file system I/O benchmarking, We spend time annotating and interpreting only throughput (both reads and writes) under varying loads[2],[3]	Write throughput
Memory Access Total Operations	sysbench --test=memory --num-threads=4 -max-time=20 run	The important number to compare (given the same or similar parameters) is the throughput and operations per second[2].	Total operations

Table With Metrics

I ran each test 5 times and took the average value for each one to be more precise.

	host machine	my container	LXC (Ubuntu 16 i386)	Docker(Ubuntu 16)
CPU total time [sec]	10.0009	10.0003	10.0000	10.0001
File IO Write[MiB/sec]	191.35	188.78	62.844	62.875
File IO Read[MiB/sec]	287.02	282.67	94.312	94.312
memory access total operations	89706804	87056448	99843978	104857600
memory access throughput[MiB/sec]	8759.64	8321.72	8136.85	8227.50
threads total time[sec]	20.0001	20.0003	3.6231	3.8231
threads total number of events	96749	95274	10000	10000

Explanation Why Metrics Differ

It should be noted that while the metrics listed above only differ from the benchmark measured on the host machine in specific parameters, the results exhibit slight variations due to the fact that Ubuntu 16 supports only version 0.4.12 of sysbench, whereas version 1.0.18 was used on the host machine. This has resulted in changes to certain algorithms used to measure performance.

CPU test

The CPU total time outcomes are nearly identical across all instances, albeit exhibiting a slight variance that could be interpreted as a measure of precision.

File IO test

When it comes to the performance of File IO, it is better on the host OS and my container, while LXC and Docker show inferior results (to some extent). The reason behind this is that LXC and Docker create overhead due to being mounted on a loopback mounted filesystem, and when processes write there, it causes significant overhead as they use loop devices, as explained in the Linux Kernel Doc.

Memory test

The results of memory access are consistent across all cases, although there is a slight variation that may be interpreted as an indication of accuracy.

Threads test

The outcomes of the thread test show discrepancies between the host and my container using LXC and Docker. I previously noted that different versions of sysbench were utilized in the tests, resulting in varying results that are now more challenging to interpret. However, I believe that the results are relatively similar across all platforms since the host kernel assigns the tasks for execution.

References

1. https://www.howtoforge.com/how-to-benchmark-your-system-cpu-file-io-mysql-with-sysbench
2. https://wiki.gentoo.org/wiki/Sysbench
3. https://minervadb.com/index.php/2018/03/27/benchmarking-cpu-memory-file-i-o-and-mutex-performance-using-sysbench/
4. https://www.alibabacloud.com/blog/testing-io-performance-with-sysbench_594709