Need for Speed: Effects of Containerization on Network Performance

Swathi Hoysala, Jeyavaishnavi Muralikumar, Ashrith Sheshan

I. INTRODUCTION

In recent times, software iterations are moving at a very fast pace. Computing environments running these softwares or libraries have to update to the newer iterations. Due to these updates, running softwares reliably when moved from computing environment to another becomes a huge problem. There was a need for technology that is platform/compute environment agnostic. This is where Container technology comes into picture. Container technology has gained tremendous popularity since it enables a fast and easy way to package, distribute and deploy applications and services.

A container consists of an entire runtime environment: an application, plus all its dependencies, libraries and other binaries, configuration files needed to run it, all bundled into one package. By containerizing the application platform and its dependencies, differences in OS distributions and underlying infrastructure are abstracted away.

However, networking between containers may come with its own overheads. Even between containers on the same host, different practices have different implications for performance (see related work) and we could find only a limited amount of evaluation of network performances in containerized and virtualized environments. Therefore, we plan to extend what we could find in order to better understand the effects of containerization on network performance.

II. RELATED WORK

It is generally accepted that containerization imposes overheads on network performance when compared to native OS-level networking. [3] makes a comparison of hypervisors and containers in terms of several system functions, including networking performance, and conclude that Docker and LXC provide comparable throughput to native networking for TCP streams, but only 60% of the bandwidth for UDP stream. They also measure performance and conclude that container networking degrades performance by a significant percentage.

[5] gives a lower level and more detailed analysis of container network performance, including throughput and performance evaluation on a variety of combinations of containers, VMs and hosts, such as containers on the same host, containers on different hosts with one on a VM and so on. They make several conclusions, such as that containers on Linux are better than those on Windows, networking overhead depends on choice of network virtualization such as veth vs MACVLAN, and that VMs may or may not be better for performance since they introduce NUMA binding but also introduce overheads. [4] also discusses some causes of performance limitations in container environments, such as the fact that the throughput of overlay networking between containers on a host will be limited by CPU.

III. INITIAL EXPERIMENTS

We ran initial set of experiments to measure TCP and UDP latencies and bandwidths. We compared the network performance of TCP and UDP by benchmarking container-to-VM communication against VM-to-VM communication. We used tools like qperf [2] and netperf [1] for our measurements.
The following environment was used to run the experiments:

- AWS - For two VMs
- VMs running Ubuntu Server 16.04
- RAM - 2GB
- Storage - 20GB
- Dokcer Version - 17.12.0-ce
- Docker image - ubuntu:latest(17.10)

The following initial observations were made:

- TCP latency of container-to-VM was comparable with that of the VM-to-VM
- TCP bandwidth of the container-to-VM was half of that of the VM-to-VM
- Reported UDP bandwidth is much higher with containerization because it somehow leads to 100 % CPU utilization. This does not happen for TCP.
- UDP latency seems to suffer by a factor of 2-3.

IV. PLAN

We plan to focus our evaluations on the following questions:

1) Do conclusions based on the evaluation of LXC [5] necessarily generalize to different container images?
2) What overhead does containerization introduce on top of virtualization?
3) Does the network performance of containers depend on the underlying OS of the host?

Based on the observations from our initial experiments we plan to evaluate

1) How can the disparity in UDP and TCP performance overhead be explained?
2) What causes the container to use 100% CPU when benchmarking UDP performance which doesn't seem to happen with TCP?
3) How does the TCP and UDP implementation on docker scale with the number of containers running on a single host?

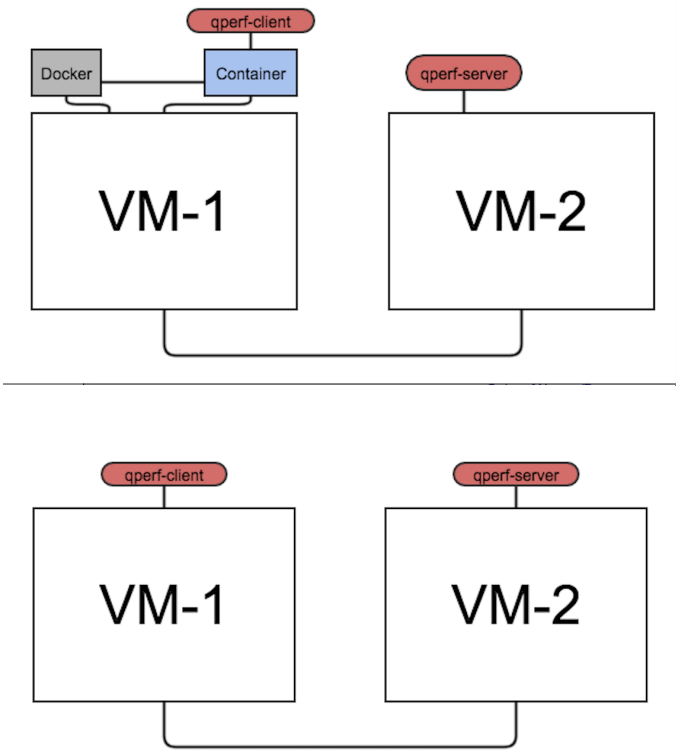

Fig. 1. Testbed architecture used in initial experiments

V. RESOURCES

We currently have a couple of hundred dollars of Amazon Web Services(AWS) credits. We did our initial benchmarking on them and we see a lot of inconsistencies with the numbers we are getting. We believe that this is because of the underlying network provided by AWS.
In order to run our experiments correctly, we need 2 servers with the following requirements:

- RAM : At least 4GB
- Storage : At least 20GB
- Network : Ethernet

In order to compare network performance with and without containers on different operating systems(OS), we need 2 servers/hardware with OS installation capabilities.

REFERENCES

[1] netperf- manual page. `https://hewlettpackard.github.io/netperf/doc/netperf.html`.

[2] Johann George. qperf-Man page. `https://linux.die.net/man/1/qperf`.

[3] Roberto Morabito, Jimmy Kjällman, and Miika Komu. Hypervisors vs. lightweight virtualization: a performance comparison. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on*, pages 386–393. IEEE, 2015.

[4] Tianlong Yu, Shadi Abdollahian Noghabi, Shachar Raindel, Hongqiang Liu, Jitu Padhye, and Vyas Sekar. Freeflow: High performance container networking. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 43–49. ACM, 2016.

[5] Yang Zhao, Nai Xia, Chen Tian, Bo Li, Yizhou Tang, Yi Wang, Gong Zhang, Rui Li, and Alex X Liu. Performance of container networking technologies. In *Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems*, pages 1–6. ACM, 2017.