Effects of Containerization on Network Performance

Swathi Hoysala, Jeyavaishnavi Muralikumar, Ashrith Sheshan

I. REVIEW

The stated goal of this project was to measure networking performance in container environments. Our initial plan was to focus our evaluation in the following directions:

1) Do conclusions based on the evaluation of LXC [3] 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?

In our previous update, we noted that TCP performance was comparable whereas UDP performance improved due to increased CPU utilization.

II. EXPERIMENTAL SETUP

The experiments reported in this checkpoint were run on the provided setup, which is identical VMs with 4-cores @ 2.83 GHz and 16 GiB of memory. We're not sure of the network between our VMs, but it shouldn't matter since we are measuring overheads anyway. Unless otherwise mentioned, the container image used was ubuntu-17.10. The container platform used was Docker version 17.12.0-ce.

III. CURRENT RESULTS

A. ICMP overhead

We checked how ping works VM-to-VM and in the container-to-VM setup and noticed the following values.

Scenario	RTT avg	RTT min	RTT max	RTT sdev
VM to VM	0.095	0.099	0.136	0.018
Container to VM	0.123	0.144	0.224	0.028

TABLE I: ICMP RTT stats; All values are in ms

All values are in milliseconds. We can see a rough 25% overhead in RTT, which we believe is due to an extra hop caused by the containerization. This is because the per hop latency is pretty much constant in both setups, as we will show later.

B. TCP and UDP bandwidth

We tried to compare the TCP and UDP bandwidth for both setups, i.e., between the VMs in one case, and by sending requests from a container on one of those VMs to the other in the second case. We used qperf [2] version 0.4.9 to obtain the following measurements in the latter case.

Image	TCP bw	UDP bw	TCP cpu usage	UDP cpu usage
centos	118 MB/s	213 MB/s	10.8%	100%
ubuntu	118 MB/s	220 MB/s	11.3%	100%

TABLE II: Bandwidth measurement from qperf on a container to a VM for packet size 2 KB

The figures reported above are for packet size 2 KB. We found, surprisingly that the TCP bandwidth was pretty much constant with varying packet size, whereas the UDP bandwidth increased. We believe that the qperf results may be normalized for some reasons, but are not sure about the reason for this behavior. Summing up the bandwidth results, we find that:

- UDP bandwidth measurement inside a container causes 100% cpu usage, leading to higher observed bandwidth than with VM-to-VM communication. This is not an artifact of using qperf, because we observed similar results while using other tools such as netperf [1]. This does not happen with TCP bandwidth measurement.
- TCP bandwidth does not seem to increase inside a container as it should with increasing packet size, whereas UDP bandwidth does (see figure). A possible explanation is that we are not using large enough packet sizes. However, we observed using our own scripts that bandwidth increase for TCP is slow for the VM-to-VM case also. We do not have an explanation at this point.

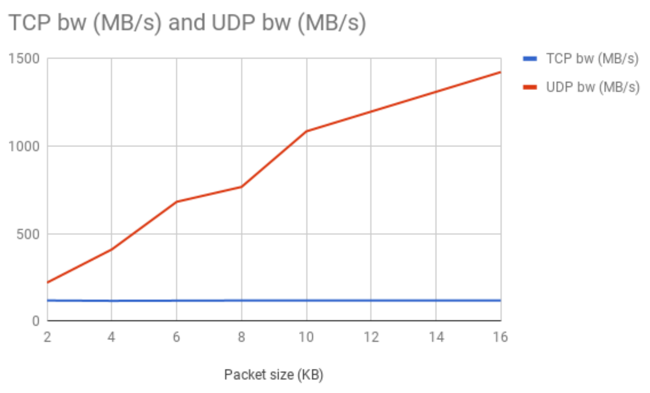

Fig. 1: TCP and UDP bandwidth w.r.t packet size, measured by sending data from a container image to another VM using qperf

TCP bandwidth is comparable to the VM-to-VM case whereas UDP bandwidth is significantly higher due to the high cpu usage. We plan to repeat this experiment with our own scripts to better understand what's happening.

C. TCP RTT

Since qperf by construction reports per hop latency, we measure TCP overhead by writing our own script to measure RTT between the two hosts. As for every other experiment, we have two setups: VM-to-VM and container-to-VM. We averaged the results over multiple round trip communications. As can be seen from the following table, the overhead was about 10%.

Scenario	Mean RTT	Std dev
VM to VM	235.289	20.2013
VM to container	259.1474	28.2726

TABLE III: TCP RTT; All values are in microseconds

IV. PLAN

Since we've been running into issues due to our lack of understanding on how network measurement tools work, we plan to conduct the rest of the experiments with our own scripts. Over the next couple of weeks, this is what we plan to do:

- Look at TCP and UDP bandwidth in the container and compare with VM-to-VM communication, as well as try to figure out why UDP throughput is so high in containers.
- Measure UDP performance overheads (RTT) in container environments.
- Measure performance impact in multi container environments, where multiple containers share a host. We expect degradation due to contention for the docker0 network interface.
- Measure performance impact in multi-flow traffic. Similar to the multi container setup, we will have multiple flows being routed through the docker0 interface.
- Measure overheads in multiple host environments, and see if they compare.

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] 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.