Testing Docker multi-host network performance

Vadim Tkachenko (https://www.percona.com/blog/author/vadim/) | August 3, 2016 | Posted In: Docker (https://www.percona.com/blog/category/docker/), MySQL (https://www.percona.com/blog/category/mysql/)

(https://www.percona.com/blog/wp-content/uploads/2016/06/docker-logo.png)In this post, I'll review Docker multi-host network performance.

In a past post, I tested Docker network (https://www.percona.com/blog/2016/02/05/measuring-docker-cpu-network-overhead/). The MySQL Server team (http://mysqlserverteam.com/mysql-with-docker-performance-characteristics/)provided their own results, which are in line with my observations.

For this set of tests, I wanted to focus more on Docker networking using multiple hosts. Mostly because when we set up a high availability (HA) environment (using Percona XtraDB Cluster, for example) the expectation is that instances are running on different hosts.

Another reason for this test is that Docker recently announced the 1.12 release, which supports Swarm Mode. Swarm Mode is quite interesting by itself — with this release, Docker targets going deeper on Orchestration deployments in order to compete with Kubernetes and Apache Mesos. I would say Swarm Mode is still rough around the edges (expected for a first release), but I am sure Docker will polish this feature in the next few releases.

Swarm Mode also expects that you run services on different physical hosts, and services are communicated over Docker network. I wanted to see how much of a performance hit we get when we run over Docker network on multiple hosts.

Network performance is especially important for clustering setups like Percona XtraDB Cluster and MySQL Group Replication (which just put out another Lab release (http://mysqlhighavailability.com/mysql-group-replication-for-5-7-14/)).

For my setup, I used two physical servers connected over a 10GB network. Both servers use 56 cores total of Intel CPUs.

Sysbench setup: data fits into memory, and I will only use primary key lookups. Testing over the network gives the worst case scenario for network round trips, but it also gives a good visibility on performance impacts.

The following are options for Docker network:

- No Docker containers (marked as "direct" in the following results)
- Docker container uses "host" network (marked as "host")
- Docker container uses "bridge" network, where service port exposed via port forwarding (marked as "bridge")
- Docker container uses "overlay" network, both client and server are started in containers connected via overlay network (marked as "overlay" in the results). For "overlay" network it is possible to use third-party plugins, with different implementation of the network, the most known are:
  - Calico network https://github.com/projectcalico/calico-containers (https://github.com/projectcalico/calico-containers)
  - Weave network https://github.com/weaveworks/weave (https://github.com/weaveworks/weave)

For multi-host networking setup, only "overlay" (and plugins implementations) are feasible. I used "direct", "host" and "bridge" only for the reference and as a comparison to measure the overhead of overlay implementations.

The results I observed are:

Client	Server	Throughput, tps	Ratio to "direct-direct"
Direct	Direct	282780	1.0
Direct	Host	280622	0.99
Direct	Bridge	250104	0.88
Bridge	Bridge	235052	0.83
overlay	overlay	120503	0.43

Calico overlay	Calico overlay	246202	0.87
Weave overlay	Weave overlay	11554	0.044

Observations

- “Bridge” network added overhead, about 12%, which is in line with my previous benchmark. I wonder, however, if this is Docker overhead or just the Linux implementation of bridge networks. Docker should be using the setup that I described in Running Percona XtraDB Cluster nodes with Linux Network namespaces on the same host (https://www.percona.com/blog/2016/06/20/running-percona-xtradb-cluster-nodes-with-linux-network-namespaces-on-the-same-host/), and I suspect that the Linux network namespaces and bridges add overhead. I need to do more testing to verify.
- Native “Overlay” Docker network struggled from performance problems. I observed issues with ksoftirq using 100% of one CPU core, and I see similar reports. It seems that network interruptions in Docker “overlay” are not distributed properly across multiple CPUs. This is not the case with the “direct” and “bridge” configuration. I believe this is a problem with the Docker “overlay” network (hopefully, it will eventually be fixed).
- Weave network showed absolutely terrible results. I see a lot of CPU allocated to “weave” containers, so I think there are serious scalability issues in their implementation.
- Calico plugin showed the best result for multi-host containers, even better than “bridge-bridge” network setup

Conclusion

If you need to use Docker “overlay” network — which is a requirement if you are looking to deploy a multi-host environment or use Docker Swarm mode — I recommend you consider using the Calico network plugin for Docker. Native Docker “overlay” network can be used for prototype or quick testing cases, but at this moment it shows performance problems on high-end hardware.

Related

Running Percona XtraDB Cluster in a multi-host Docker network (https://www.percona.com/blog/2016/0…xtradb-cluster-in-a-multi-host-docker-network/)
June 10, 2016
In "Docker"

Percona Server for MySQL in Docker Swarm with Secrets (https://www.percona.com/blog/2017/0…server-for-mysql-in-docker-swarm-with-secrets/)
April 21, 2017
In "Docker"

Measuring Docker IO overhead (https://www.percona.com/blog/2016/0…docker-io-overhead/)
February 11, 2016
In "Docker"

Vadim Tkachenko (/blog/author/vadim/)

Vadim Tkachenko co-founded Percona in 2006 and serves as its Chief Technology Officer. Vadim leads Percona Labs, which focuses on technology research and performance evaluations of Percona’s and third-party products. Percona Labs designs no-gimmick tests of hardware, filesystems, storage engines, and databases that surpass the standard performance and functionality scenario benchmarks. Vadim’s expertise in LAMP performance and multi-threaded programming help optimize MySQL and InnoDB internals to take full advantage of modern hardware. Oracle Corporation and its predecessors have incorporated Vadim’s source code patches into the mainstream MySQL and InnoDB products. He also co-authored the book High Performance MySQL: Optimization, Backups, and Replication 3rd Edition.