Evaluating Feasibility of Container Virtualization for Virtual Network Functions

M.S Thesis Defense

Uğurcan Ergün
Advisor: Dr. Atay Özgövde

Galatasaray University

4 July 2018

Agenda

- Problem
- Motivation
- Literature Review
- Methodology
- Results
- Conclusion

Problem

- Network operators starting to experience unprecedented levels of mobile data usage and network traffic.
- There are certain limitations that constrain them when trying to improve the infrastructure.
- They pursue to implement dynamic, flexible networks with various new technologies.
- Design of 5G networks implies that telecom operators are trying to approximate telecom infrastructure to cloud infrastructure

Problem

- Computers networks are still complex and hard to manage because of the proprietary and tightly coupled nature of the most network devices.
- Software defined networking try to implement the lessons learned in virtualization to the networks.
- But there are still significant amount of proprietary network appliances.
- Network function virtualization has been introduced as a solution

Motivation

- Companies that rely on cloud computing started using a different way to use software on the cloud.
- Decoupling software into independent services and running them inside a new virtualization solution called containers.
- We believe this client native approach to software and performance benefits of containers can be used to utilize network function virtualization better.

Literature Review

- Most of the works in the literature are focused on implementing network function virtualization with standard hypervisor virtualization.
- For infrastructure cloud platform Openstack seems to be a de-facto choice as it also has an NFV project.
- Few works that implemented VNFs with containers were focused on different areas such as edge computing.

Methodology

Setup

- Our focus would be measuring network performance for Kubernetes.
- We installed a Kubernetes cluster in a desktop computer.
- We picked nginx that can function as a simple network function for experiments.
- We used Kubernetes deployments for managing containers on the cluster.

Methodology

Experiment Design

- Experiment cases include different configuration of containers with different resource limits and instance counts.
- These experiments then run with automated scripts using an open source load testing tool vegeta.
- Network latency is then reported as mean, maximum and different percentile values.
- Each experiment has been run 10 times and then averaged.

Methodology

Percentile Metrics

- Percentile metrics are used for getting more accurate information about the application performance.
- They aren't affected by outlier data points contrary to arithmetic means.

Calculation of percentile metrics

First the results are sorted from minimum to maximum and then highest 100-n percent of the results are deleted. The maximum remaining result is called the nth percentile result.

Results

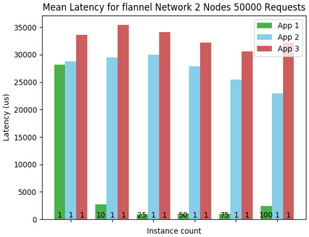
First Experiment: Configuration

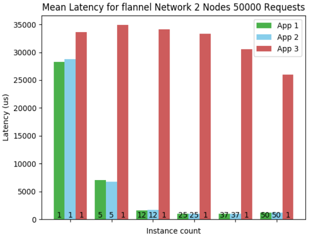
Table: Number of instances for each deployment

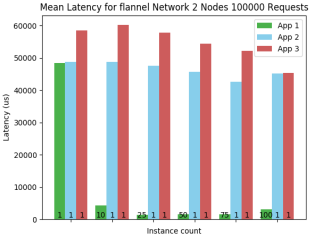
Deployment 1	Deployment 2	Deployment 3
1	1	1
5	5	1
10	1	1
12	12	1
25	1	1
25	25	1
37	37	1
50	1	1
50	50	1
75	1	1
100	1	1

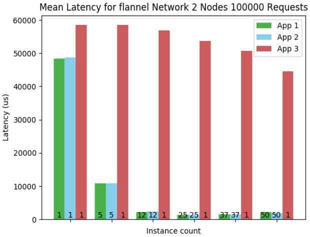
Results

First Experiment


Mean Latency for flannel Network 2 Nodes 50000 Requests


Mean Latency for flannel Network 2 Nodes 50000 Requests


Mean Latency for flannel Network 2 Nodes 100000 Requests


Mean Latency for flannel Network 2 Nodes 100000 Requests

Results

First Experiment: Discussion

- Even an aged desktop computer can run high number of containers.
- Computer might support more 100 containers.
- Applications doesn't seem to affected from co-location related latency.
- Slight amount of increase at the final steps latencies are caused by host load

Results

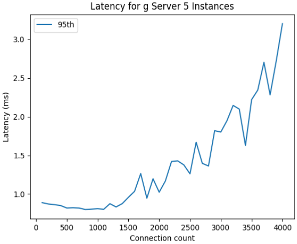
Second Experiment: Configuration

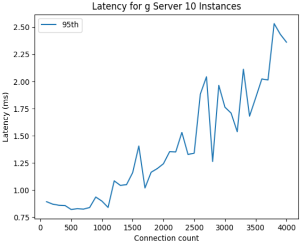
Setup	Left side	Right side
Setup A	5 * 250m	5 * 250m
Setup B	10 * 250m	20 * 250m
Setup C	1 * 1250m	1 * 2500m
Setup D	1 * 2500m	1 * 5000m
Setup E	10 * 125m	20 * 125m
	5 * 250m	10 * 250m
	3 * 500m	5 * 500m
Setup F	20 * 125m	40 * 125m
	10 * 250m	20 * 250m
	5 * 500m	10 * 500m

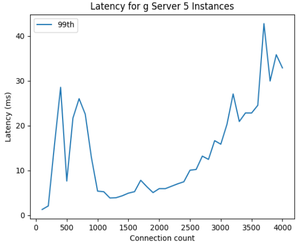
Table: experiment setups as instance count * CPU limits

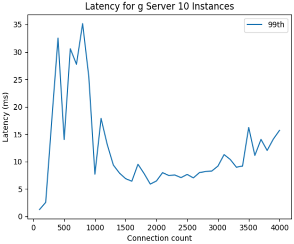
Results

Second Experiment: Setup A


Latency for g Server 5 Instances

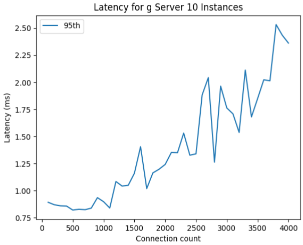

Latency for g Server 10 Instances

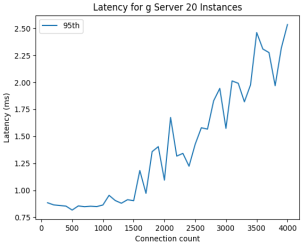

Latency for g Server 5 Instances

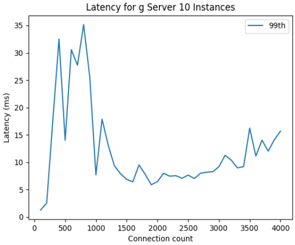

Latency for g Server 10 Instances

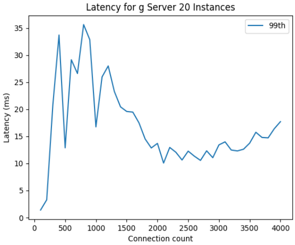
Results

Second Experiment: Setup B



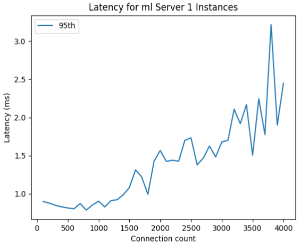


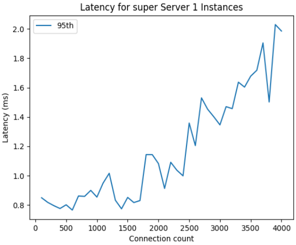


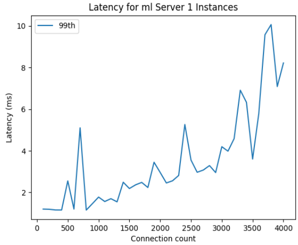


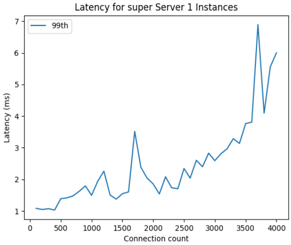
Results

Second Experiment: Setup C


Latency for ml Server 1 Instances

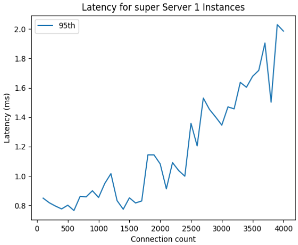

Latency for super Server 1 Instances

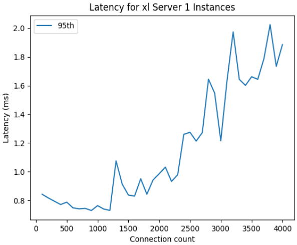

Latency for ml Server 1 Instances

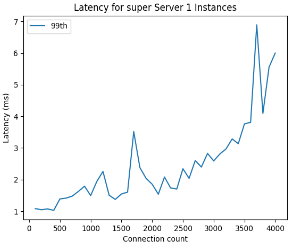

Latency for super Server 1 Instances

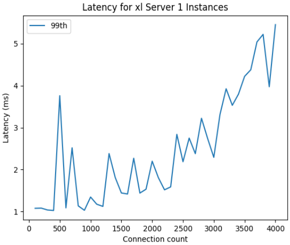
Results

Second Experiment: Setup D


Latency for super Server 1 Instances

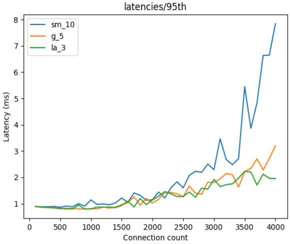

Latency for xl Server 1 Instances

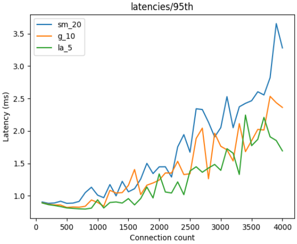

Latency for super Server 1 Instances

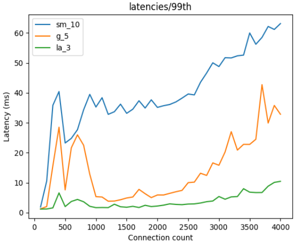

Latency for xl Server 1 Instances

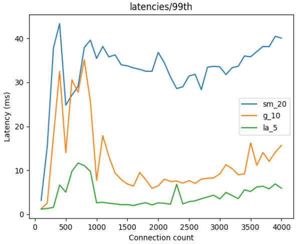
Results

Second Experiment: Setup E



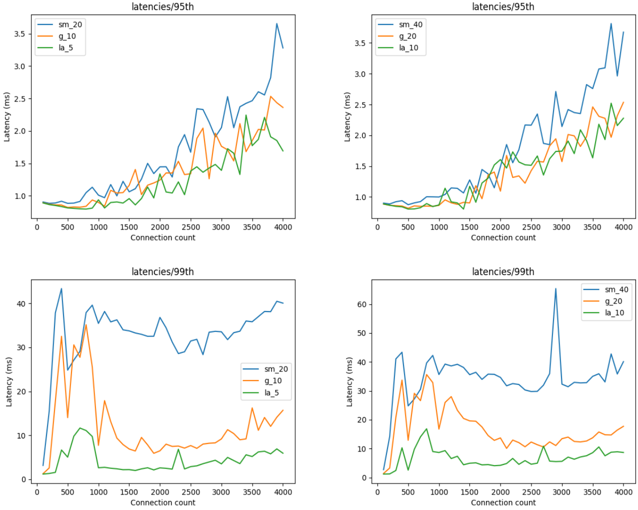






Results

Second Experiment: Setup F



Results

Second Experiment: Discussion

- Using multiple containers instead of a single one increases overhead. But some cases perform very similarly
- Major differences in latencies mostly occur in slowest 5 percent of connections.
- Balance between instance count and resource limit proves crucial for performance.
- After a certain point extra instances started to add more overhead.

Conclusions

- Fundamental technologies for implementing virtual network functions are present in container environments.
- Even for a very small percent of results there is significant peaks in latency. This may not acceptable for some network functions.
- As a solution specialized high speed networking software may be needed.
- There are still no out of box solutions for network functions in container platforms.

Thank you for listening

Questions ?