Are Kubernetes CNI solutions ready for >10 Gbit/s?

Simon Volpert, Georg Eisenhart, Jörg Domaschka
Institute of Information Resource Management
University of Ulm, Germany
{firstname}.{lastname}@uni-ulm.de

***Abstract*—The growth in variety of Container Network Interface (CNI) implementations as well as increasing Kubernetes adoption on bare metal require a thorough assessment of its performance characteristics. Specifically, the question whether CNI can saturate network links above 10 Gbit/s is of interest. This paper highlights an initial approach and central research questions towards the goal of this assessment. Therefore, we propose a method to acquire measurements that act as a groundwork for further investigations. Preliminary results show that a simple setup is not able to saturate even a 50 Gbit/s link.**

***Index Terms*—container, virtualization, benchmark, cni**

I. Introduction

With the rise of server virtualization, tightly related aspects of network virtualization became ubiquitous. Cloud computing platforms like Kubernetes and OpenStack make it easy to leverage them in order to enable connectivity among workloads, multi-tenancy and separation of concerns.

This paper narrows the scope to Kubernetes and investigates how virtual networking is realized and how it performs. We therefore pick several representative Container Network Interface (CNI[1]) implementations and compare them against each other.

While most of these technologies are adopted in (virtualized) cloud environments with limited and shared networking resources, we specifically look at their capabilities on physical hardware utilizing interfaces above 10 Gbit/s throughput. Considered metrics hereby include possible link saturation, compute resource utilization and latency. Further, Kubernetes allows for a multitude of configurations regarding connectivity among containers. This also applies to the `service` abstraction wrapping multiple pods and therefore containers, an `ingress` component handling access from outside of the cluster and network policies maintaining connectivity rules.

The core questions being worked on in the scope of this ongoing research are *(i)* whether CNI implementations can achieve the high throughput rates of their underlying physical interfaces, *(ii)* how many connections are necessary to saturate it and *(iii)* how Kubernetes network resources affect this performance.

II. Related Work

Kapočius [1] performed a thorough performance study of several CNI solutions measuring latency and throughput amongst other metrics. However, this is focused on link aggregation of 1Gbit/s links, assessing performance degradation imposed by the amount of aggregated links.

Later, Qi et al. [2] investigated inter and intra host CNI performance utilizing 25 Gbit/s network cards. Interesting metrics presented by them are CPU cycles per packet and an interface startup time latency breakdown. Yet our core questions remain unanswered. Compared to them we pursue a more thorough investigation of the CNI impact on distinct Kubernetes resources like `services`, `ingresses`, `meshes` and `policies` while scaling resources as high as possible.

The relation between network policies and CNI implementations have been investigated by Budigiri et al. [3]. While they only found a minor impact on latency, their testing environment was virtualized with a maximum bandwidth of 5 Gbit/s and focused on the CNI provider Calico.

III. Methodology

In order to retrieve meaningful reproducible data, a benchmarking workflow on a test infrastructure is built. The workflow is separated in 4 steps. At first the *(i)* participating servers are bootstrapped. Therefore an operating system is booted into memory, including its configuration. Part of this process is the bootstrapping of Kubernetes with its embedded CNI. Upon finish, a *(ii)* benchmarking workload is deployed. This workload utilizes Netperf[2] which allows the collection of measurements for bandwidth and latency. The benchmark itself is run for 10 minutes to generate sufficient data points. Afterwards, the *(iii)* data is collected and the *(iv)* infrastructure is being destroyed. Figure 1 illustrates that process. The logical test setup for the benchmarks consists of four servers interconnected with a single switch. The servers involved are a *(a)* netperf client, *(b)* netperf server, *(c)* Kubernetes master and *(d)* data sink for measurements. Figure 2 visualizes this setup.

IV. Results

Preliminary measurements results have been performed on a test setup built on physical infrastructure. The involved servers each are based on Intel Xeon E5-2630 CPUs with 2.40GHz and 256GB DDR4 RAM. Furthermore, each is equipped with a Mellanox MCX515A (50Gbit/s) network interface card. These are directly connected to a Mellanox MLX SN2100 switch

[1]https://github.com/containernetworking/cni

[2]https://hewlettpackard.github.io/netperf/

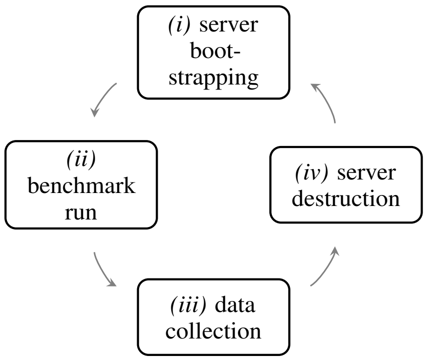

Figure 1. Measurement workflow

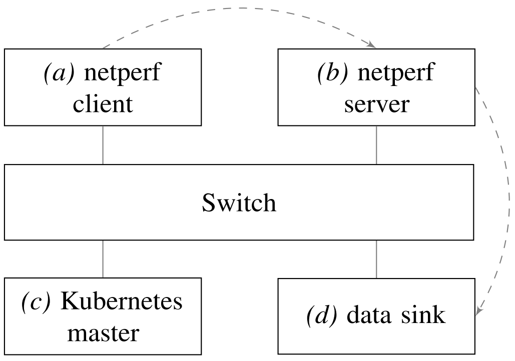

Figure 2. Logical test setup

running the Onyx 3.9 operating system. The server operating systems are Fedora CoreOS 33 with kernel 5.9. The Kubernetes distribution is a Rancher k3s with version v1.20.

In order to set measurements between two pods and two hosts into perspective, we first perform some benchmarks without any network virtualization. These baseline tests validate the sufficiency and the capabilities of the methodology. Since a single stream could not utilize the full bandwidth capacity of our setup, we parallelized this test to 3 simultaneous streams.

Upon determination and verification of this baseline performance, further measurements for pod-to-pod performance with network virtualization can be performed. Since the CNI implementation by Flannel[3] is widely adopted it is benchmarked first within the scope of this paper. Flannel can be configured to utilize different approaches for network virtualization. Here the default configuration (VXLAN) is applied. The baseline and the Flannel performance benchmark are highlighted in Figure 3.

These results yield several insights: *(i)* One TCP connection cannot saturate a 50 Gbit/s link with this specific CPU, whereas a second run with *(ii)* 3 parallel connections is able to do so. While Flannel's single performance *(iii)* is slightly oversaturating a 10 Gbit/s link, we were not able to achieve more than ~13 Gbit/s in this scenario. As we can see, latency and especially its standard deviation is significantly higher than

[3]https://github.com/flannel-io/flannel

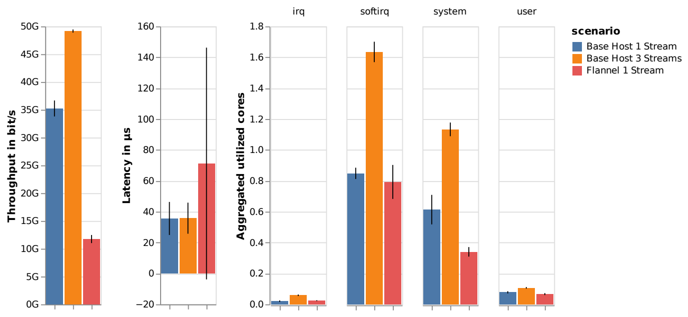

Figure 3. Preliminary results of throughput, latency and cpu utilization measurements

that for scenarios without any network virtualization. The time this process can spend in user, system and softirq time is much lower. Lower softirq directly correlate to lower throughput.

V. Conclusion and Outlook

The hardware/software combination we are currently running is capable of providing single connections with 10 Gbit/s, but is certainly quickly limited afterwards. However, we can only partly answer the introductory question whether CNI is ready for >10Gbit/s yet. For inter host connections using Flannel with VXLAN this is not the case. Thus further investigations with more parallelism, further CNI provider and different Kubernetes Resources are necessary.

Theoretically better performing eBPF [4] based solutions seem to be promising, since they are able to avoid parts of the classic netfilter stack. The impact of `service meshes` on top of CNI providers could be another interesting direction.

These measurements opened up a lot of further questions regarding bottlenecks, strategies and possible performance tunings. As a consequence we currently pursue further research to answer those questions. Ultimately, these insights shall be helpful for deciding which CNI solution fits best for specific requirements.

Acknowledgment

This work was made possible by the Institute of Information Resource Management of Ulm University, providing the necessary hardware and infrastructure.

References

[1] N. Kapočius, "Performance Studies of Kubernetes Network Solutions," in *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, Apr. 2020, pp. 1–6.

[2] S. Qi, S. G. Kulkarni, and K. K. Ramakrishnan, "Assessing Container Network Interface Plugins: Functionality, Performance, and Scalability," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 656–671, Mar. 2021.

[3] G. Budigiri, C. Baumann, J. T. Mühlberg, E. Truyen, and W. Joosen, "Network Policies in Kubernetes: Performance Evaluation and Security Analysis," in *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, Jun. 2021, pp. 407–412.

[4] S. Miano, M. Bertrone, F. Risso, M. Tumolo, and M. V. Bernal, "Creating Complex Network Services with eBPF: Experience and Lessons Learned," in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, Jun. 2018, pp. 1–8.