Performance Studies of Kubernetes Network Solutions

Narūnas Kapočius
Vilnius Gediminas Technical University
narunas.kapocius@gmail.com

***Abstract*—Containers and microservices applications are complex and their orchestration is inevitable. As of today, Kubernetes is the most popular choice for container orchestration. However, there are various Kubernetes network implementations based on different CNI (Container Networking Interface) plugins which functionalities and performance could differ. In this paper 4 CNCF (Cloud Native Computing Foundation) recommended CNI plugins are benchmarked in a physical datacentre environment. CNI plugins performance is evaluated in terms of latency and average TCP throughput for various Maximum Transmission Unit (MTU) sizes, the different number of aggregated network interfaces, and different interfaces segmentation offloading conditions. Plugins' performance is compared to baremetal baseline results.**

Keywords—Kubernetes, CNI, Container Network Interface, aggregation, teaming

I. INTRODUCTION

In recent years' microservices paradigm and containerization emerged. Microservices and containerization offer flexibility for developers, the development process is more robust and fast, systems are easy to scale, and containerization adds less computing overhead compared to traditional hypervisors. However, systems based on microservices usually are complex, made of a high number of containers and microservices communicating with each other. Because of that, the need for containers orchestrators is inevitable. The networking is a crucial part of microservices because every microservice communicates to other microservices using some kind of protocol. Kubernetes network model is based on predefined rules but the model itself does not specify the implementation of them. The most used solution in Kubernetes networking is CNI (Container Network Interface). CNI is based on built-in plugins and also allows using third-party plugins. CNI handles containers addition/removal from the network, access lists, etc. There are various CNI plugins that solve the same problem using different techniques and it is important to benchmark these plugins. In recent years' publications' performance of CNI plugins were compared in different scenarios. However, related researches focused on CNI performance comparison in physical and virtual servers environment but the impact of different MTU (Maximum Transmission Unit) sizes, network interfaces aggregation, and NIC (network interface cards) offloading was not studied in detail. In this paper 4 CNCF (Cloud Native Computing Foundation) recommended CNI plugins Flannel, Calico, Kube-router, and Weave are benchmarked in the physical data center environment. CNI plugins' performance is compared in terms of latency and average TCP throughput and is compared to baremetal baseline results.

The remainder of the paper is organized as follows. In Section II recent Kubernetes CNI plugins performance and interface aggregation studies are overviewed. The research methodology and the test environment are described in Section III, in Section IV studies results are presented and in the final section, research conclusions are given.

II. RELATED WORK

The performance of three popular Kubernetes Network solutions has been examined in [1]. The authors compare the average throughput and latency of the Kubernetes CNI plugins Flannel and Calico and Swarm Overlay network solution in a physical 1Gbit/s network. The performance of Kubernetes network solutions was investigated for TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) protocols. The results presented in the article show that the actual average throughput of the physical medium in TCP protocol's case is 919.7 Mbit/s, and the average throughput achieved by the Calico, Flannel and Swarm Overlay plugins, respectively, are 919 Mbits, 815.9 Mbit/s, and 887.6 Mbit/s. The average throughput of Calico is very similar to the baremetal results, and the throughput of the VXLAN (Virtual Extensible Local Area Network) based Flannel and Swarm Overlay plugins are lower than the baremetal and Calico throughput.

The article discussed above compares not only the throughput of CNI plugins but also the average packet latency. The average packet latency of the CNI plugins studied in this article is very similar and it‘s about 0.1 ms higher than the baremetal latency. However, the configuration of the testbed used for the measurements, the test methodology, and the number of measurements taken in the article are not presented clearly. Also, it is worth mentioning that the authors also present UDP throughput results for mentioned CNI plugins. However, such comparison is inaccurate - due to the specifics of UDP protocol, there is no reliable way to measure channel bandwidth using UDP protocol.

The importance of comparing CNI plugins is highlighted in [2]. The authors examined the performance of the OpenStack and Kubernetes CNI plugin network in the virtual OpenStack network. The test environment consisted of a single physical server with 64GB of RAM, a Xeon (R) E5-2697 CPU, and a 10GSFP + network. There were four virtual machines running on the physical server. The first virtual machine was used as an OpenStack and Kubernetes master node, and the remaining virtual machines were used as Kubernetes worker nodes. The

978-1-7281-9779-1/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Peking University. Downloaded on August 01,2022 at 06:04:21 UTC from IEEE Xplore. Restrictions apply.

study compared the VXLAN and UDP modes of the Flannel plugin and the throughput of the OpenStack Kuryr network plugin. Throughput was investigated using the iperf network performance analysis tool, but interestingly, the throughput of the CNI plugins was measured between the first and third Kubernetes pods and whose packets were forwarded through the router on the second Kubernetes pod. Results show that the Kuryr OVS network is more efficient than the Flannel VXLAN network for 64B, 512B, 8192B, and 16384B MTU packet sizes. There is a significant throughput difference from the 512B MTU packets size and the throughput is lower by 20%, 27.27%, and 8%, respectively. The authors explain Flannels‘ poor performance because of the compute overhead caused by the plugin architecture and the frequent kernel context switching. Although the article does not emphasize the transport layer protocol which was used for the research, it can be assumed that the TCP protocol was used.

The most comprehensive study of the Kubernetes CNI plugins performance so far is presented in [3] and additional results by the same author are presented in [4]. These publications examine the performance of the 6 most commonly used CNI plugins recommended by CNCF. The study was conducted in a physical network consisting of three identical SuperMicro physical servers interconnected using SuperMicro 10Gbit switch and DAC SFP + connections. The servers ran Ubuntu 18.04 LTS OS and used Kubernetes 1.14 and Docker 18.09.2. One server was used as the Kubernetes master node, and the two other servers were used as the Kubernetes worker nodes. Bandwidth studies were performed using Kubernetes NodeSelector tags and without changing client and server worker nodes. The performance of CNI plugins has been studied at 1500B and 9000B MTU for TCP, UDP, HTTP, and SSH protocols, but only 9000B MTU results are presented in the publication. TCP and UDP throughput was measured using iperf3 tool, HTTP performance was determined using the Nginx server and curl client, the FTP protocol performance was measured using vsftpd server and the curl client, and the SSH protocol performance was examined using the OpenSSH client and server. For each protocol, at least three measurements were taken and the throughput results are presented as the average of those measurements. Plugins arranged in the following order according to throughput results: Calico, Flannel, Canal, WeaveNet, Kube-router, Cillium. The throughput of the plugins compared to the physical medium was lower respectively 0.65%, 0.89%, 0 , 91%, 1%, 1.41% and 2.26%.

Previously mentioned studies covered the Kubernetes CNI plugins which are based on the standard Linux kernel network stack. However, in specific cases the standard Linux kernel network stack is not suitable for high performance needs as ultra-low latency and high throughput. When ultra-high performance is needed, kernel bypass solutions are most commonly used to move networking functions from kernel space to user space, and solutions such as RDMA, DPDK, Netmap, Snabbswitch, PF_RING or etc. are used. RDMA and DPDK solutions usage in Kubernetes networking were examined in [5]. The research author state that the application of cloud technologies to low latency solutions would open the way for new applications of cloud technologies in the automation management of the manufacturing industry and etc. RDMA and DPDK solutions were tested using the SR-IOV Kubernetes CNI plugin developed by Intel. The Kubernetes cluster was configured using two physical servers with Intel Xeon E5-2670 v3 CPUs, 64GB RAM, and Mellanox ConnectX-4 40GbE network cards. The tests were performed using Ubuntu 16.04 OS with kernel version 4.13 and Kubernetes version 1.9. The physical servers were connected using a Mellanox 1710 40GbE switch. The authors results show that for smaller than 65536B packets RDMA and DPDK solutions have a similar delay, but with packets larger than 65536B, the delay of RDMA solution is comparatively lower than in the case of DPDK. The results of the study also show that RDMA and DPDK solutions are more efficient in terms of latency and CPU consumption than a standard Linux network stack drivers.

Another high-performance, low-latency Kubernetes network solution is proposed in a Italian-US project called Polycube [6]. Polycube is an open source software framework designed to implement high-performance kernel space networking features. The Polycube framework is based on eBPF (Extended Berkley Packet Filter), which provides the ability to build packet processing chains. The Polycube solution proposed by the authors, in contrast to the SR-IOV CNI plugin, only works in kernel space. The authors emphasize that while user space network solutions provide noticeable latency and throuoghtput improvements, reserving CPU resources deduct application resources and the entire network stack must be redesigned and moved to user space. The authors applied the Polycube framework to create a Kubernetes CNI plugin that was tested on a physical network. The Kubernetes cluster consisted of three physical servers with an Intel Xeon E3-1245v5 @ 3.50 GHz CPU, two-port Intel XL710 40GBps network cards, and the servers were connected by a point-to-point connection. Network throughtput was studied using the iperf3 tool with default settings and using the TCP protocol. Throughtput was investigated both between Kubernetes pods operating on the same worker node and pods operating on different work nodes. The study shows that the Polycube solution throughput is 15-20% higher than using usual Kubernetes CNI plugins. In the pods on different worker nodes, the highest throughput was achieved with the Polycube plugin, and Kubernetes CNI plugins were less efficient in descending order: Kube-router, Romana, WeaveNet, Cillium, Calico. Accorrding to results, it is likely that the Calico CNI plugin was tested in the default IPIP (IP-in-IP) mode.

Since this paper presents CNI plugins performance in an aggregated network environment it is worth reviewing the recent studies of Linux network interface aggregation performance.

A review of recent years publications has revealed that there is not much an in-depth research on Linux network interface aggregation, and most of the studies were performed using an older Linux network bonding drivers rather than newer teaming drivers. The performance of Linux bonding drivers has been examined in [7]. The article presents a study of the the 2.6.3 version of the bonding driver using 2.6.13 kernel version on RedHat Centos 4 OS. Even though study wasn‘t conducted recently, it deserves review because of the methodology used in it. The study was conducted using two physical servers with three 1Gbit speed network cards. The servers are directly connected with CAT6 category network cables. Throughput was measured using the iperf Linux tool and the UDP protocol was

Authorized licensed use limited to: Peking University. Downloaded on August 01,2022 at 06:04:21 UTC from IEEE Xplore. Restrictions apply.

investigated in RoundRobin bonding driver mode. Study results showed that using RoundRobin mode and UDP protocol it is possible to achieve higher total speed than in the case of individual interfaces. The authors mention that RoundRobing mode spreads packets reception order, which in the case of the TCP protocol would not increase, but reduce the total channel transmission rate due to additional packet retries.

A comprehensive study of Linux network interface aggregation is presented in [8]. The authors of the study emphasize that network interfaces aggregation is an appropriate solution to increase network throughput and reliability when there is no possibility to upgrade network infrastructure. The paper presents an investigation of all 6 modes of Linux bonding drivers using client and server architecture. For the throughput tests, the network flow was generated using the DNBD3 Linux tool, which acts as a Linux block device. The client reads the data from the block device running on the server using the Linux command dd and wrote the information to the /dev/ null device. The study examined two possible configurations for interconnecting network interfaces. In the first configuration, the client was connected to the switch via a 10 Gbit optical channel, and the server was connected to the switch via 10x1 Gbit optical channels which were aggregated to one logical channel using the Linux bonding driver. The performance of this configuration was compared to the base configuration when both client and server were connected to the switch using 1x10 Gbit channels. The results show that in the case of the basic configuration the maximum channel throughput of 9600 Mbit/s was achieved by the client using 4 dd client instances. In the case when the 0th bonding driver mode was used, the maximum channel throughput was reached at using 2 dd instances, however, maximum throughput is significantly lower - 2880 Mbit/s. In the second configuration, the server was connected to the switch using 10x1 Gbit channels, and the 61 clients were connected to the other switch using 1Gbit channels. Two mentioned switches were interconnected via a 1x10 Gbit channel. All 6 bonding driver modes were examined in this configuration. The results of the study show that only the 5th and 6th bonding driver modes increase the total bandwidth of the channel. However, the number of clients has to be large in order to achieve the total base configuration's throughput. The article authors argue that Linux network interfaces aggregation using a bonding driver is a useful solution when a large number of clients are using the same network. Also, the use of interfaces aggregation increase servers CPU load which is noticeably higher at 0th (round-robin) bonding driver mode, while in other modes a slight increase in CPU load can be observed, but it is relatively small.

The most comprehensive study of network interface aggregation is presented in [9]. The paper presents a study comparing the performance of Linux bonding and teaming drivers in a KVM (Kernel-based Virtual Machine) virtualized environment. The test environment consisted of two physical servers interconnected by an HP 1810 network switch. Each physical server had three network interfaces, two of which were the Intel 82576EB 1Gbit network interfaces and one RTL8111 network interface. Centos 7.2 OS was installed on the physical servers and a KVM virtualization stack consisting of Quemu 1.5.3 and Libvirt 1.2.8 software packages were installed. Network throughput was measured with Linux iperf3 tool. Three cases were investigated. In the scenario, one iperf server virtual machine and three client virtual machines were used, in the second and third scenarios, two iperf server virtual machines and three client virtual machines were used, but in the second scenario, there were no traffic restrictions according to QoS conditions as in the third scenario the first iperf server occupied 80% of the available bandwidth and the second server the remaining 20% . In each case it was investigated how total channel throughput was changing by increasing the number of threads assigned to virtual machines by assigning them 3, 45, and 90 threads. In the case of this paper, the most interesting is the first scenario of the study, when one virtual machine for iperf server was used and it was connected to the network using aggregated network interfaces. The results show that the total bandwidth for both bonding and teaming drivers is the same when two network interfaces are aggregated. For example, for 90-threads count, the total measured channel bandwidth when using bonding and teaming drivers are respectively, 1840 and 1874 Mbit/s, and when 3 interfaces were aggregated, 2573 and 2570 Mbit/s, respectively. It is worth noting that the results of the study are not compared with the maximum practical speed of the physical network, but it is likely that increasing the number of threads assigned to virtual machines it would bring the total bandwidth of the aggregated interfaces closer to the maximum channel bandwidth.

III. Laboratory Environment Setup

The test environment was set up in the private datacenter using 3 HP Proliant DL380 G3 servers and one HP ProCurve 2910 24G L2 switch (Fig. 1). One server was used as a Kubernetes master node and two other servers as worker nodes. Worker node servers had identical specifications and had 16GB of RAM and 8xIntel(R) Xeon(R) CPUE5606 @ 2.13 GHZ and master node had 32 GB of RAM and 8xIntel(R) Xeon(R) CPUE5620 @ 2.40 GHZ. All three servers had 6 Ethernet interfaces. 4 interfaces were provided by Broadcom NetXtreme II BCM5709 NIC and two other inter-faces by Intel 82571EB/82571GB NIC. The maximum bandwidth of all Ethernet ports was 1Gbit/s. On each server, one interface was used for the remote management and the other five interfaces were used in interfaces aggregation tests and were connected to the mentioned HP switch. Each of the five interfaces was assigned to one of the five VLANs (Virtual Local Area Network): VLAN 2, 3, 4, 5, 6. This VLAN configuration was needed in order to achieve best aggregated interfaces results. Network interfaces were aggregated using Linux teaming teamd 1.27 driver. During tests active "loadbalance" runner of teaming driver was used. Traffic among aggregated interfaces was load balanced using VLAN IDs and L4 port hashes.

Docker-CE 19.03.4, kubeadm 1.17.3 and kubelet 1.17 were used to create Kubernetes cluster. Network throughput was measured using iperf3 tool. One Kubernetes worker node ran iperf3 server pods and another worker node iperf3 client pods. Servers performance statistics were gathered using sysstat package sar tool. Each measurement took 240 seconds of which 120 seconds were used to measure throughput and other 60 seconds before and after each measurement was used to gather idle server performance statistics. All measurements were automated using Bash scripts and were started by cron on the Kubernetes master. Also, network delay was determined using

Authorized licensed use limited to: Peking University. Downloaded on August 01,2022 at 06:04:21 UTC from IEEE Xplore. Restrictions apply.

ping command and was measured for 300 seconds sending ICMP packets every 0.5 seconds.

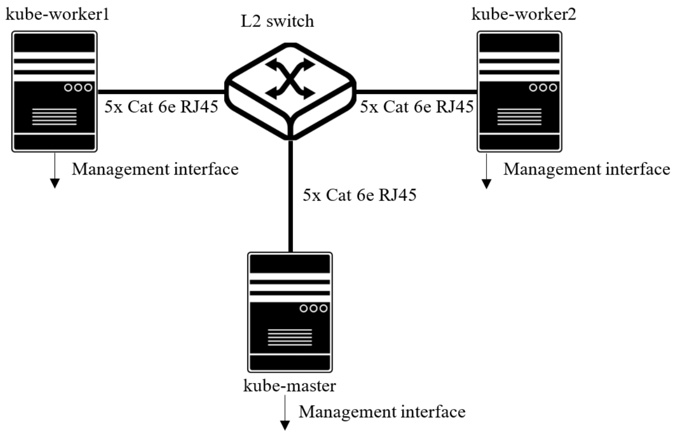

Fig. 1. General Kubernetes CNI and Linux teaming driver test environment setup.

IV. Results Analysis

During tests, 4 CNI plugins performance was measured: Flannel, Calico, Kube-router, and Weave. For Calico 3 different modes were tested: VXLAN, IPIP, and pure IP. All CNI plugins were tested using 4 different test scenarios. In A scenario the average RTT (Round Trip Time) of the ping packets were measured. In B scenario CNI plugins TCP protocol throughput was measured when servers were using one network interface. In C scenario CNI plugins performance was measured also using one network interface but for all servers NIC send/receive GSO (Generic Segmentation Offload) function was turned off. In scenario total TCP bandwidth of CNI plugins was measured aggregating 2–5 network interfaces

A. CNI Plugins Latency Results

Results are given in Fig. 2 and show CNI plugins average network delay when the different number of network interfaces were aggregated. Results show that Intel network interfaces add ~0.1 ms more delay than Broadcom interfaces. Also, it is clearly seen that all CNI plugins add a significant amount of delay compared to baremetal test results. However, pure IP CNI solutions such as Calico IP and Kube-router performs noticeably better than overlay CNI plugins. This can be explained by additional compute overhead which is added by VXLAN and IPIP headers. Also, Kube-Routers' packets RTT is less ~1–5% compared to Calico IP solution in all tested scenarios. From overlay plugins standpoint Flannel outperforms Weave and Calico VXLAN plugins despite that the same overlay technology is used.

Also, it is clearly shown that the increasing number of teamed network interfaces increases overall channel delay. This can be explained by that more interfaces are aggregated, more teaming driver active load balancing algorithm tries to re-distribute packets. This means that ping packets travel through various interfaces and path changes often which add additional delay.

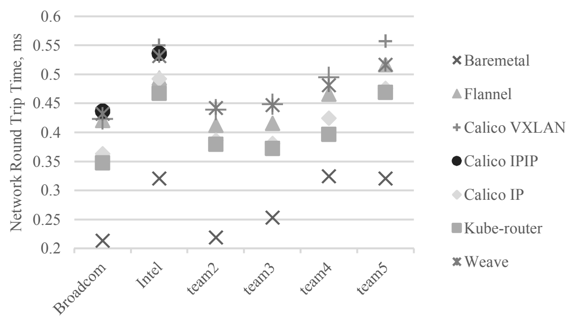

Fig. 2. CNI plugins average RTT for different number of teamed interfaces.

B. One Interface Throughput Results

In table I CNI plugins TCP protocol throughput results for various MSS (Maximum Segment Size) sizes when Intel interface is used are presented. Test results for Broadcom interfaces for 76 B MSS are ~60% and for MSS larger than 200B on average are ~2% worse compared to Intel's results. The analysis has shown that non-overlay CNI plugins Calico and Kube-router throughput is almost the same as baremetal throughput. Interestingly, Calicos' VXLAN mode performs extremely poorly when small MSS sizes are used. Flannel performs best among tested overlay plugins when small MSS are used. However, it is worth mentioning that starting from 972B MSS size all overlay plugins' throughput is almost the same. When MSS size increases towards Jumbo frame size, the difference between CNI plugins and baremetal performance is getting lower and when Jumbo frames are used the difference is so small that it can be neglected. However, Jumbo frames application to real-world workloads is quite limited as most HTTP based applications data is transmitted using 460–1448 B size packets [10].

C. NIC Offloading Results

It was also tested how NIC TCP receive and send GSO affects CNI plugins performance. Results (Fig. 3) show that performance degraded up to 80% for 460B and smaller MSS sizes. However, in this test scenario, pure IP CNI solutions performed extremely poor, and surprisingly overlay solutions performance was almost the same as when GSO was on. For all higher than 972B MSS sizes no difference was seen either GSO was on or off.

TABLE I. CNI Plugins TCP protocol Tthroughput For Various MSS Sizes When Using One Intel Interface

MSS, B	**76**	**204**	**460**	**972**	**1448**	**1996**	**4044**	**8140**	**8948**
Baremetal	457	694	836	915	941	957	978	989	990
Calico IP	458	694	836	915	939	957	978	989	990
Calico IPIP	326	649	807	898	927	948	974	987	988
Calico VXLAN	99	253	595	874	907	935	967	983	985
Flannel	159	449	767	874	909	935	967	983	985
Kube-router	458	694	836	915	942	957	978	989	990
Weave	122	346	766	874	904	935	967	983	985

Authorized licensed use limited to: Peking University. Downloaded on August 01,2022 at 06:04:21 UTC from IEEE Xplore. Restrictions apply.

TABLE II. CNI PLUGINS TCP PROTOCOL RELATIVE THROUGHPUT DEGRADATION COMPARED TO BAREMETAL RESULTS WHEN 2 INTERFACES WERE AGGREGATED

MSS, B	Flannel	Calico IP	Calico VXLAN	Kube-router	Weave
76	-53.3%	-1.9%	-63.8%	-1.6%	-64.8%
204	-47.5%	-2.3%	-60.0%	-0.3%	-61.2%
460	-7.5%	1.8%	-27.6%	1.8%	-22.8%
972	-4.4%	0.4%	-4.0%	2.2%	-4.5%
1448	-3.5%	0.5%	-3.9%	-0.6%	-3.6%
1996	-2.4%	0.4%	-2.6%	0.5%	-2.9%
4044	0.0%	1.8%	0.1%	1.8%	-0.3%
8140	2.6%	3.6%	2.7%	3.7%	1.9%
8948	2.1%	2.4%	1.5%	2.8%	1.2%

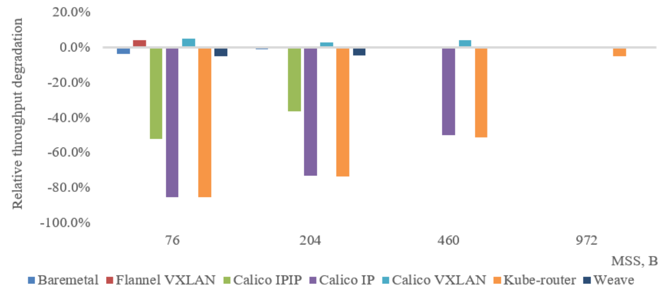

Fig. 3. CNI plugins TCP average throughput degradation when 1 network interface is used and NIC GSO is off.

TABLE III. CNI PLUGINS TCP PROTOCOL RELATIVE THROUGHPUT DEGRADATION COMPARED TO BAREMETAL RESULTS WHEN 3 INTERFACES WERE AGGREGATED

MSS,B	Flannel	Calico IP	Calico VXLAN	Kube-router	Weave
76	-53.4%	-6.0%	-64.9%	-4.2%	-65.7%
204	-47.5%	-5.2%	-60.3%	-3.1%	-61.7%
460	-8.3%	1.3%	-26.8%	1.6%	-23.8%
972	-4.2%	1.1%	-4.2%	0.9%	-4.7%
1448	-3.1%	0.6%	-3.1%	0.9%	-4.0%
1996	-2.6%	0.6%	-2.2%	0.5%	-2.9%
4044	-0.2%	1.6%	-0.2%	2.1%	-0.6%
8140	2.6%	3.7%	1.7%	3.9%	0.6%
8948	0.1%	1.0%	-0.8%	1.0%	-1.0%

D. Aggregated Interfaces Results

CNI plugins TCP protocol average total throughput for aggregated network interfaces was compared to similar baremetal results. In tables II–V are shown CNI plugins average total TCP throughput relative degradation for various MSS sizes compared to similar baremetal results. Results show that overlay plugins performs extremely poor under small MSS sizes and the same as in one interface results overlay solutions performance aligns with pure IP solutions when MSS size is 972B or more. However, when a number of aggregated interfaces increases relative degradation for smaller MSS sizes increases as well. For example, for the smallest 76B MSS size Flannel's throughput decreased ~53,3% when 2 interfaces were aggregated compared to ~60.1% when 5 interfaces were aggregated. TCP performance of pure IP CNI solutions performs noticeably good in the whole range of MSS sizes. Pure IP CNI solutions throughput is about the same to baremetal throughput starting even from 460B MSS size. Also, there is no noticeable difference between Calico IP and Kube-router solutions in terms of TCP throughput performance.

TABLE IV. CNI PLUGINS TCP PROTOCOL RELATIVE THROUGHPUT DEGRADATION COMPARED TO BAREMETAL RESULTS WHEN 4 INTERFACES WERE AGGREGATED

MSS, B	Flannel	Calico IP	Calico VXLAN	Kube-router	Weave
76	-58.6%	-6.0%	-67.4%	-5.1%	-70.3%
204	-46.3%	-5.3%	-58.6%	-6.8%	-61.4%
460	-13.9%	0.3%	-27.2%	0.8%	-23.7%
972	-4.6%	0.6%	-4.8%	0.8%	-4.6%
1448	-3.4%	0.2%	-4.0%	0.8%	-3.9%
1996	-2.6%	0.1%	-2.7%	0.2%	-3.2%
4044	-1.2%	0.7%	-0.9%	0.9%	-1.4%
8140	1.0%	2.2%	0.9%	2.8%	0.3%
8948	-0.6%	0.8%	-0.7%	1.0%	-5.7%

TABLE V. CNI PLUGINS TCP PROTOCOL RELATIVE THROUGHPUT DEGRADATION COMPARED TO BAREMETAL RESULTS WHEN 5 INTERFACES WERE AGGREGATED

MSS, B	Flannel	Calico IP	Calico VXLAN	Kube-router	Weave
76	-60.1%	-7.4%	-70.1%	-6.5%	-72.3%
204	-45.6%	-7.1%	-61.2%	-7.0%	-60.0%
460	-13.6%	-0.7%	-30.5%	-0.8%	-26.3%
972	-4.4%	0.7%	-4.2%	0.7%	-9.1%
1448	-3.4%	0.6%	-3.5%	1.0%	-4.2%
1996	-3.1%	0.0%	-3.5%	0.2%	-3.5%
4044	-1.7%	0.4%	-1.5%	0.7%	-3.5%
8140	0.5%	2.1%	0.5%	2.1%	0.1%
8948	-0.6%	0.4%	-0.8%	0.7%	-1.7%

V. CONCLUSIONS

This paper analyzed 4 CNCF recommended Kubernetes CNI plugins performance in the physical data center environment. CNI plugins performance was evaluated in 3 nodes Kubernetes cluster using 1–5 aggregated network interfaces. Results show that CNI plugins solutions add at least 0.2ms of additional latency to the channel and that pure IP solutions add 0.05ms less latency compared to overlay CNI solutions. In terms of latency Flannel overlay and Kube-router pure IP solutions performed best. Also, it was confirmed that an increasing number of teamd aggregated network interfaces adds additional delay to the channel because of packet distribution among several physical channels. All CNI plugins TCP average throughput is close to baremetal throughput for all MSS sizes larger than 972B and for

Authorized licensed use limited to: Peking University. Downloaded on August 01,2022 at 06:04:21 UTC from IEEE Xplore. Restrictions apply.

all tested cases. It shows that neither a number of aggregated interfaces, neither GSO offloading has no negative effect on average TCP throughput when larger packets are used. However, smaller than 972B MSS size packets perform extremely poor for both aggregated interfaces and when interface GSO offloading is off. Results show that an increasing number of aggregated interfaces increases CNI overlay solutions average total throughput degradation, but it does not affect pure IP solutions average TCP throughput. Also, GSO offloading affects only pure IP CNI solutions TCP throughput for smaller MSS sizes however does not affect overlay solutions TCP performance.

REFERENCES

[1] H. Zeng, B. Wang, W. Deng and W. Zhang, “Measurement and Evaluation for Docker Container Networking,” 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Nanjing, 2017, pp. 105–108.

[2] Y. Park, H. Yang and Y. Kim, “Performance Analysis of CNI (Container Networking Interface) based Container Network,” 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, 2018, pp. 248–250.

[3] A. Ducastel, “Benchmark results of Kubernetes network plugins (CNI) over 10Gbit/s network,” in itnext.io, November 2018.

[4] A. Ducastel, “Benchmark results of Kubernetes network plugins (CNI) over 10Gbit/s network (Updated: April 2019),” in itnext.io, April 2019.

[5] D. Géhberger, D. Balla, M. Maliosz and C. Simon, "Performance Evaluation of Low Latency Communication Alternatives in a Containerized Cloud Environment," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, 2018, pp. 9-16.

[6] S. Miano et al., "A Service-Agnostic Software Framework for Fast and Efficient in-Kernel Network Services," 2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Cambridge, United Kingdom, 2019, pp. 1-9.

[7] S. Aust, J. Kim, P. Davis, A. Yamaguchi and S. Obana, "Evaluation of Linux Bonding Features," 2006 International Conference on Communication Technology, Guilin, 2006, pp. 1-6.

[8] K. Meier, S. Schmelzer and D. Suchodoletz, “Evaluation of Network Bonding Performance in Client/Server Scenarios,” Hochleistungsrechnen in Baden-Wurttemberg – Ausgewahlte Aktivitaten im bwGRiD 2012, pp. 221–234, 2014

[9] K. Rattanaopas and K. Boonchuay, “A high performance network virtualization architecture with bandwidth guarantees,” 2017 International Electrical Engineering Congress (iEECON), Pattaya, 2017, pp. 1–4.

[10] Center for Applied Internet Data Analysis, “Packet size distribution comparison between Internet links in 1998 and 2008,” in caida.org.

Authorized licensed use limited to: Peking University. Downloaded on August 01,2022 at 06:04:21 UTC from IEEE Xplore. Restrictions apply.