Evaluation of Layer 3 Multipath Solutions using Container Technologies

Á. Kovács, G. Lencse
Department of Telecommunications
Széchenyi István University
Gyor, Hungary
E-mail: kovacs.akos@sze.hu, lencse@sze.hu

***Abstract*— The MPT network layer multipath communication library is capable of using multiple communication channels by creating an UDP tunnel over them. The contemporary version of MPT uses the GRE tunnel protocol. MPTCP is another multipath solution, which uses TCP subflows on kernel level to ensure multipath communication. In this paper, we are using multiple container technologies to install these multipath communication solutions. Most common Docker container and a HPC specific Singularity container interconnected with twelve 100Mbit/s links were used to evaluate the aggregation capabilities of the combination of these technologies.**

***Keywords*—Benchmarking, Container, Docker, Multipath, MPT, MPTCP, Singularity**

I. INTRODUCTION

Although containerization technology spreading nowadays especially in Cloud industry, the main goal of these technologies is quiet old. Linux's chroot environment was capable of isolating different processes without the need to emulate different hardware for them at the same time. While these container technologies mostly used to isolate network services like web servers or database servers, sometimes the main bottleneck of these services is the lack network performance. MultiPath communication could be one of the easiest way to overcome this issue. If we can use multiple interfaces for a single communications, we may achieve the multiplication of the network performance. Most of our ICT devices have more than one communication interfaces, but we only use one of them for a single communication due technical reason: one particular TCP/IP communication channel can be identified by the IP addresses and the port numbers of the network devices [1]. To create a reproducible environment, we used Docker and Singularity containers to build our testbed.

First, we used the MPT network layer multipath communication library [2], which was developed at the Faculty of Informatics, University of Debrecen, Debrecen, Hungary.

To compare the aggregation capability, we used MPTCP as second multipath communication technique. It also can utilize multiple physical devices to communicate between two compatible nodes [3].

The reminder of this paper is organized as follows, in section two, a brief introduction is given to the container technologies that we used for our measurements. In section three, the MPT Communication Library and the Multipath TCP system introduced. In section four, our test environment is described. In section five, we disclose our benchmarking measurements aimed to check the performance of iperf when it is containerized. In section six we have compared the channel capacity aggregation efficiency of MPT in three different scenarios: using Docker, Singularity and native execution. In section seven we have performed the same measurements using MPTCP. Finally, our conclusion about the experiments are given.

II. CONTAINER TECHNOLOGIES

Containerization is widely used in cloud infrastructures. It is used to create lightweight virtualization to separate services and micro services like databases and webservers. Whereas Hypervisor based virtualization allows us to run different kind of operating systems on the same host based on Linux/Unix and Windows kernel, the main idea of the containerization technology is that we sacrifice the flexibility of the virtualization and use the same kernel for each micro services. Therefore, we save resources from running different kernels and, thanks to that, the performance can be consumed by the services we use. The main difference of the virtualization and the containerization is shown in Fig. 1.

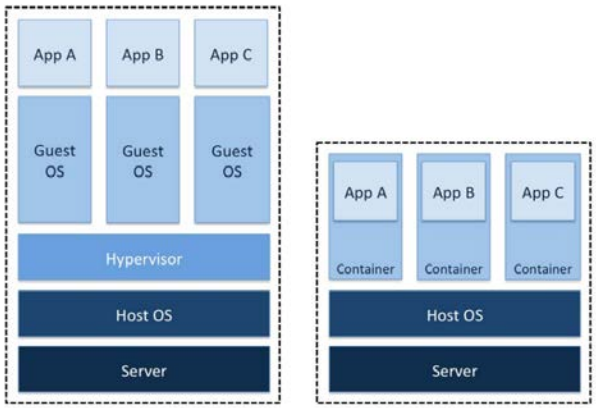

Fig. 1. KVM virtualiztaion vs. containerizaton [4]

A. *Docker*

Docker containers are widely used for flexibility and availability on different Linux distributions. The only criteria for installing Docker is the Linux Kernel version 3.10, which is a relatively old one. The most important advantage of Docker is the Docker hub, which contains wide range of containerized applications like databases, network services, and services backends as well. Docker uses a server-client architecture. The containerized applications can reach the host's resources trough the Docker daemon, which must run with root privileges. By default, Docker uses Linux-bridge to ensure network connectivity, however, to achieve the best performance, we must use the "`--network host`" directive [5], [6].

B. *Singularity*

Singularity container technology focuses on using container in HPC. It differs from Docker significantly, because it uses monolith image files for different applications, whereas Docker uses overlay based file system. Singularity container does not require root privileges, which is basic requirements for HPC systems. It can reach the basic hardware's natively, so it is an ideal decision for using special hardware's like GPUs and Inifiniband interconnects which are widely used in HPC systems. [7]

III. Multipath Applications

In this section, a brief summary is given about MultiPath technologies.

A. *MPT*

The MPT architecture uses multiple layers to communicate. Unlike other multipath solutions (e.g.: LACP, Cisco Etherchannel) which operate at layer 2, the MPT Communication library uses network layer to create a multipath communication. Thus, it can be routed so the two endpoint of the communication can be far from each other. It is based on RFC 8086 [8], which allows MPT to implement multipath communications using the GRE-in-UDP encapsulation [9].

<table>
<tr><td colspan="2">Application (Tunnel)</td></tr>
<tr><td colspan="2">TCP/UDP (Tunnel)</td></tr>
<tr><td colspan="2">IPv4 (Tunnel)</td></tr>
<tr><td colspan="2">GRE-in-UDP</td></tr>
<tr><td>UDP (Physical)</td><td>UDP (Physical)</td></tr>
<tr><td>IPv4 (Physical)</td><td>IPv4 (Physical)</td></tr>
<tr><td>interface</td><td>interface</td></tr>
</table>

Fig. 2. The architecture of MPT [2], [9]

The IP packets are transmitted through a logical tunnel interface by MPT, which encapsulates them into a GRE-in-UDP segment. MPT always uses a tunnel interface so it can map the packets coming through it. Applications that are using basic Ethernet interfaces to communicate are not needed to be modified because MPT uses a standard IP communication on the tunnel interface. MPT maps these packets and selects a real Ethernet interface to send them.

<table>
<tr><td>IPv4</td><td>UDP
Fixed Port</td><td>GRE</td><td>Tunnel
IP</td><td>Tunnel
TCP/UDP</td><td>Application
data</td></tr>
</table>

Fig. 3. The PDU structure MPT based on GRE in UDP [2]

B. *MPTCP*

Multipath TCP Linux implementation is developed by the Department of Computing Science and Engineering at Université Catholique de Louvain, Belgium. Its main goal is to improve the TCP protocol, which was designed in the 1970's. It uses a kernel module to utilize the available Ethernet NICs for a single TCP communication [10].

MPTCP does not use user space software or logical interface and it changes the default TCP implementation in the Linux kernel to a special one. MPTCP creates TCP sub-flows from a single TCP session and sends them through the available network interfaces, but it requires some configuration. To ensure the easy usage, the developers created a "network_up" script, which automatically configures the upcoming interfaces.

<table>
<tr><td colspan="2">Application</td></tr>
<tr><td colspan="2">MPTCP</td></tr>
<tr><td>Subflow (TCP)</td><td>Subflow (TCP)</td></tr>
<tr><td>IPv4 (Physical)</td><td>IPv4 (Physical)</td></tr>
</table>

Fig. 4. The architecture of MPTCP communication stack [10]

MPTCP network stack negotiates in the normal TCP/SYN exchange if the other side of the communication is capable for using multiple path for aggregate communication. After that, a new TCP sub-flow can be established. Each of these subflows has its own sequence number and congestion control like normal TCP flows [11].

C. *Comparison of MPT and MPTCP in a Nutshell*

One of the main advantages of MPT over MPTCP is that MPT is not restricted to use only TCP like MPTCP. MPT can use both TCP and UDP over the tunnel IP for communications. Therefore, MPT is more suitable for multimedia transmission. For example, MPT has been successfully applied for elimination of stalling events on YouTube video playback [12] or fast connection recovery [13].

Another important difference between MPT and MPTCP is that MPT masquerades the multipath technology under a logical interface while MPTCP uses the available interfaces with different default routes on each available NICs.

MPTCP can only utilize up to eight of the NIC but uses less resources (e.g. CPU power) than MPT. MPT has no such limits to the number of underlying paths [14].

IV. Test Environment

Two DELL Precision Workstation 490 computers were used for our tests. Their basic configuration was:

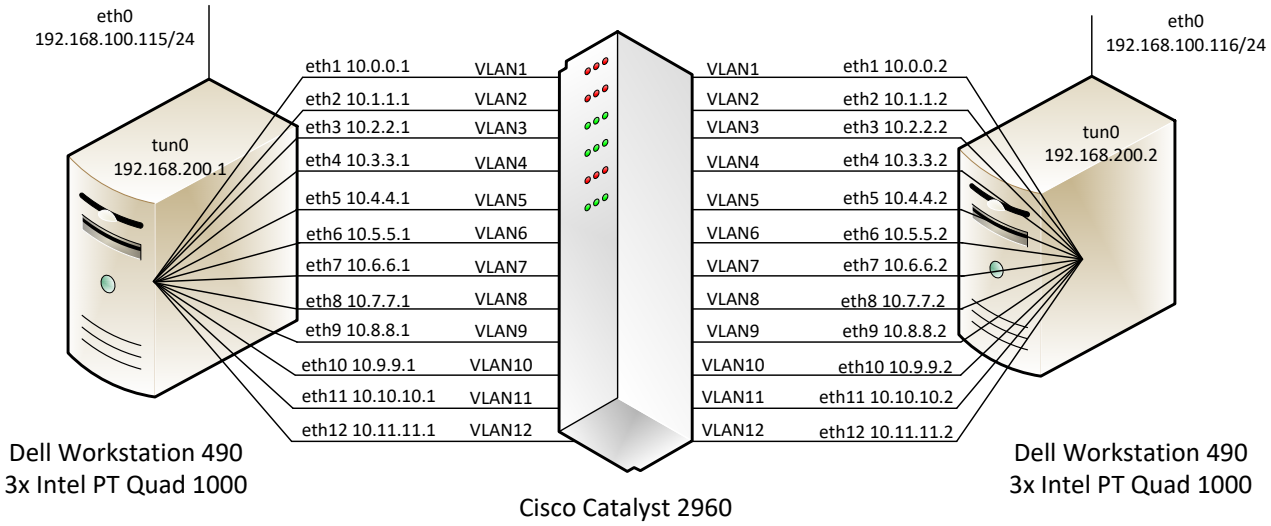

Fig. 4. The architecture of MPTCP communication stack [14]

a) DELL 0GU083 motherboard, Intel 5000X chipset
b) Two Intel Xeon 5140 2.33GHz dual core processors
c) 8x2GB 533MHz DDR2 RAM (quad channel)
d) Broadcom NetXtreme BCM5752 Gigabit Ethernet controller (PCI Express, integrated)
e) Three Intel PT quad port Gigabit Ethernet interfaces (PCI Express)
f) Debian Strech 9.7, kernel version 4.9.130-2 amd64

Each computer was able to handle 13 Ethernet interfaces, twelve for testing, and the built in one for management purposes. We used a 24 port Cisco 2960 Ethernet switch to limit each interface to 100Mbit/s because the available resources (especially CPU power) were not enough to utilize the NIC-s at 1000Mbit/s. Each pair of the interfaces was in different VLANs to eliminate the global broadcast sending. The testbed was the same as we used in [14].

V. Testing Iperf

First, we used container technology to test the industry standard iperf benchmark tool to find out if it is capable of measuring network bandwidth utilization while running inside a container. We created our own iperf container using a simple dockerfile:

```
FROM debian:9
WORKDIR /root/
RUN apt update
RUN apt install -y libssl-dev iperf net-tools
```

And we built a singularity image, using Singularity bootstrap file based on Docker Hub:

```
Bootstrap: docker
From: debian:9
%post
apt update
apt install -y libssl-dev iperf net-tools
```

To test iperf in Docker, we had to run Docker with special arguments to let the Docker use host networking, which means that the Docker daemon injects its packages among the host packages natively:

```
docker run -it --network host iperf-testing
```

Testing iperf with singularity is a bit easier while it's default behavior is to use host mode networking.

We used MPT natively for testing and running iperf in different containers and native for comparison.

To ensure that MPT and MPTCP may not use the 13th NIC of the testbed we created an iptables rule to drop all packet between the two management interfaces.

We compared the path aggregation capabilities of the created iperf containers with that of the native execution. The results are shown in Fig. 5.

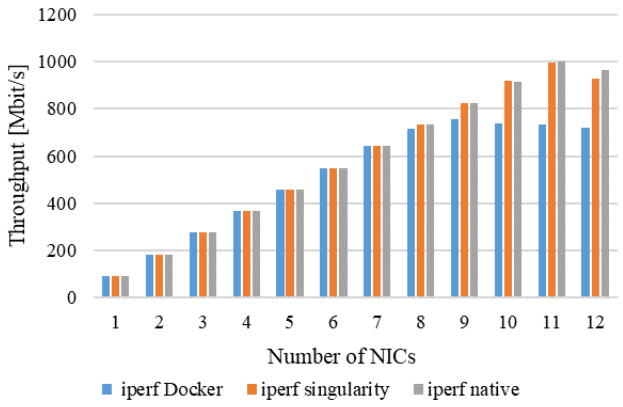

Fig. 5. Results of the iperf contaner benchmark

As we can see, if we are using singularity as container technology, the throughput can achieve almost the same performance as the native. If we are using Docker as a container solution, the results do not scale up over 9 NICs. For

the best results, we were using iperf on the host natively for the further tests while the multipath solution was containerized.

VI. Experiments using MPT

To test MPT communication library with docker, we compiled the latest source code form [2], then we created a new container with a special overlay, which contains the MPT communication library. After that, we run Docker with special arguments to let the Docker be able to create and use the tunnel interface of the MPT communication library:

```
docker run -it --network host --cap-add=NET_ADMIN \
--device /dev/net/tun:/dev/net/tun mpt
```

For examining the aggregation capability of a containerized MPT communication library, we used iperf with the following command:

```
iperf -c 10.100.0.2 -t 120 -i 2 -y C
```

This command executes iperf benchmark on the tunnel device for 120 sec, in every two seconds, it displays the result and saves it to CSV format. MPT can be run within the container like natively and it creates the `tun` device on the host computer. We used the IP addresses of these tunnel endpoints in the iperf command to ensure using MPT for the communication. The results are shown in Fig. 6.

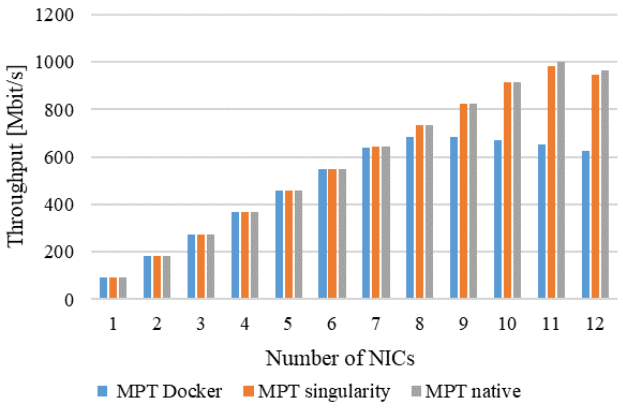

Fig. 6. MPT benchmark using container technologies

As we can see the MPT using singularity almost reached the performance of the native version of the experiment. Although the results of MPT using Docker are the same as that of the native version up to 7 NICs, they are significantly worse from 8 NIC, and the tendency is even slightly decreasing. The maximum throughput is also lower than that of the iperf tests.

VII. Experiments using MPTCP

MPTCP is a bit more complex as it uses a kernel space module to add the capability of using multiple physical NICs for multipath communication. It means that MPTCP cannot be containerized because it works as a part of the kernel. As we used MPTCP on the physical host, the containers – which use the same kernel as the host – are automatically able to use multipath communication ensured by MPTCP. We edited the `/etc/network/interfaces` file to ensure that the management interface is started last, so that it will not be used as multipath communication channel. As we were unable to containerize a kernel module, we used the before mentioned and tested iperf container to compare the aggregation capability of MPTCP. So in this case the not MPTCP, but iperf was containerized. The measurement setup was similar to that of our first experiment, but now we using MPTCP as the underlying multipath communication.

In this case we used the following iperf command:

```
singularity exec /root/iperf.simg iperf
-c 10.0.0.2 -t 120 -i 2 -y C
```

We executed this command with one of the NICs used by MPTCP because it can use only 8 NICs to communicate [14].

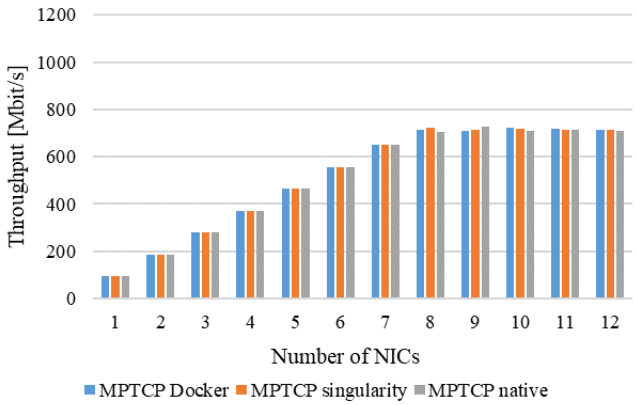

Fig. 7. MPT benchmark using container technologies

The measurement results are shown in Fig. 7. As we can see, using MPTCP with different container technologies or native execution we got almost the same results, the differences are negligible. This is because the MPTCP can only use 8 NICs, so this was limiting factor in all three cases.

VIII. Conclusion

Using container technologies to provide different network services is a widely used practice in the industry, although it has some disadvantages. To ensure the highest possible throughput, we usually prefer container technologies that use host adapters natively, but this solution violates the principle of separation of the different applications. We can use upper layer network implementations like the examined multipath technologies (MPT and MPTCP). We have shown that there is a minor performance penalty on Singularity and a major on Docker containerization technology compared to the native execution. In the future, we plan to test some more up-to-date hardware with even 1000Mbit/s throughput capability to measure the aggregation capability.

References

[1] B. Almási A. Harman, “An overview of the multipath communication technologies”, In Proceedings of the Conference on Advances in Wireless Sensor Networks 2013 (AWSN 2013), Debrecen University Press, Debrecen, Hungary, ISBN: 978-963-318-356-4, 2013, pages 7-11.

[2] B. Almási, Sz. Szilágyi “MPT Multipath Communication Library”, available: https://erlang2.inf.unideb.hu/~szilagyi/index.php/mpt-gre/

[3] C. Paasch, S. Barre, et al., Multipath TCP in the Linux Kernel, available from http://www.multipath-tcp.org. used version 0.90

[4] Ann Mary Joy, “Performance comparison between Linux containers and virtual machines” , 2015 International Conference on Advances in Computer Engineering and Applications, 2015, pp: 342 - 346

[5] Amr A. Mohallel; Julian M. Bass; Ali Dehghantaha “Experimenting with docker: Linux container and base OS attack surfaces”,2016 International Conference on Information Society (i-Society),2016,pp.: 17 - 21

[6] Flávio Ramalho; Augusto Neto, “Virtualization at the network edge: A performance comparison”, 2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016, pp.: 1 – 6.

[7] Miguel G. Xavier; Marcelo V. Neves; Fabio D. Rossi; Tiago C. Ferreto; Timoteo Lange; Cesar A. F. De Rose, “Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments”, 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing , 2013, pp.: 233 - 240

[8] L. Yong, E. Crabbe, X. Xu, T. Herbert, “GRE-in-UDP encapsulation”, IETF RFC 8086, March 2017.

[9] B. Almási, G. Lencse, Sz. Szilágyi, "Investigating the Multipath Extension of the GRE in UDP Technology" Computer Communications (Elsevier), vol. 103, no. 1, pp. 29-38, May 1, 2017.

[10] A. Ford, C. Raiciu, M. Handley, S. Barre, J. Iyengar, “Architectural Guidelines for Multipath TCP Development” RFC 6182, March 2011.

[11] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik and M. Handley, “Improving datacenter performance and robustness with multipath TCP” - SIGCOMM 2011, Toronto, Canada, August 2011.

[12] F. Fejes, S. Rácz, and G. Szabó, “Application agnostic QoE triggered multipath switching for Android devices”, In: Proc. 2017 IEEE International Conference on Communications (ICC 2017), Paris, France, May 21-25, 2017, pp. 1585–1591. DOI: 10.1109/ICC.2017.7997450

[13] F. Fejes, R. Katona, and L. Püsök, “Multipath strategies and solutions in multihomed mobile environments”, in: Proc. 7th IEEE International Conference on Cognitive InfoCommunications (CogInfoCom 2016), Wroclaw, Poland, Oct. 16-18, 2016, pp. 79–84, DOI: 10.1109/CogInfoCom.2016.7804529

[14] Á. Kovács, “Comparing the aggregation capability of the MPT communications library and multipath TCP” 2017 Proceedings of 7th IEEE Conference on Cognitive Infocommunications, Budapest, Magyarország : IEEE Hungary Section, (2017) pp. 157-162. , 6 p.