# Considerations for Benchmarking Network Performance in Containerized Infrastructures

draft-dcn-bmwg-containerized-infra-05

**KJ Sun**, Hyunsik Yang, Jangwon Lee,

Nguyen Quang Huy, Younghan Kim

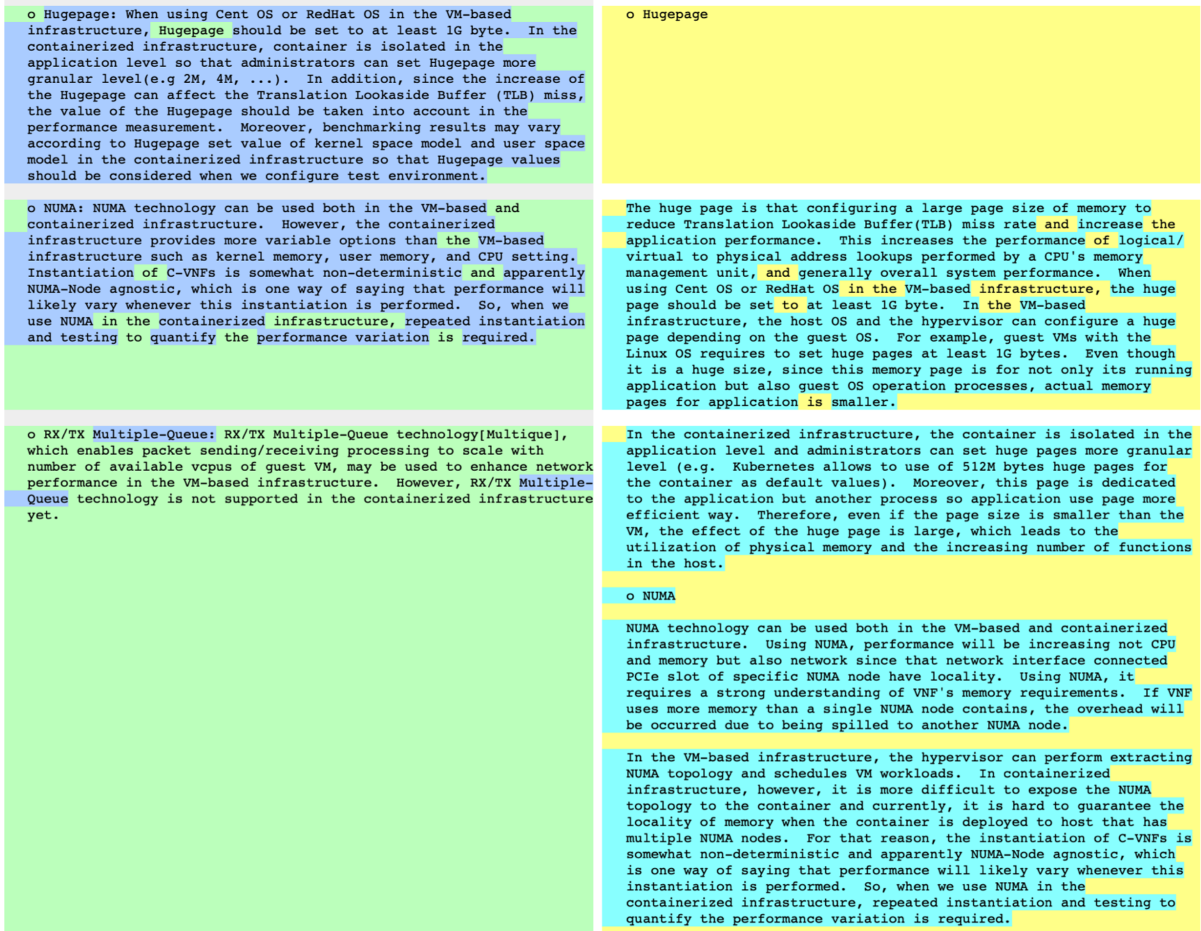
Internet Infrastructure System Technology Research Center(IISTRC)

# About draft

- Describe differences and additional considerations for benchmarking containerized infrastructure compared with VM-based infrastructure
  - Network models
    - Kernel space model
    - User space model – vswitch model, device pass-through model
  - Benchmarking scenarios
    - BMP2VMP
    - VMP2VMP
  - Resource considerations
    - NUMA
    - Huge page
    - …

# Update from -02 to -03

- Adding description in Chapter 3.3



o Hugepage: When using Cent OS or RedHat OS in the VM-based infrastructure, Hugepage should be set to at least 1G byte. In the containerized infrastructure, container is isolated in the application level so that administrators can set Hugepage more granular level(e.g 2M, 4M, ...). In addition, since the increase of the Hugepage can affect the Translation Lookaside Buffer (TLB) miss, the value of the Hugepage should be taken into account in the performance measurement. Moreover, benchmarking results may vary according to Hugepage set value of kernel space model and user space model in the containerized infrastructure so that Hugepage values should be considered when we configure test environment.

o NUMA: NUMA technology can be used both in the VM-based and containerized infrastructure. However, the containerized infrastructure provides more variable options than the VM-based infrastructure such as kernel memory, user memory, and CPU setting. Instantiation of C-VNFs is somewhat non-deterministic and apparently NUMA-Node agnostic, which is one way of saying that performance will likely vary whenever this instantiation is performed. So, when we use NUMA in the containerized infrastructure, repeated instantiation and testing to quantify the performance variation is required.

o RX/TX Multiple-Queue: RX/TX Multiple-Queue technology[Multique], which enables packet sending/receiving processing to scale with number of available vcpus of guest VM, may be used to enhance network performance in the VM-based infrastructure. However, RX/TX Multiple-Queue technology is not supported in the containerized infrastructure yet.

o Hugepage

The huge page is that configuring a large page size of memory to reduce Translation Lookaside Buffer(TLB) miss rate and increase the application performance. This increases the performance of logical/virtual to physical address lookups performed by a CPU's memory management unit, and generally overall system performance. When using Cent OS or RedHat OS in the VM-based infrastructure, the huge page should be set to at least 1G byte. In the VM-based infrastructure, the host OS and the hypervisor can configure a huge page depending on the guest OS. For example, guest VMs with the Linux OS requires to set huge pages at least 1G bytes. Even though it is a huge size, since this memory page is for not only its running application but also guest OS operation processes, actual memory pages for application is smaller.

In the containerized infrastructure, the container is isolated in the application level and administrators can set huge pages more granular level (e.g. Kubernetes allows to use of 512M bytes huge pages for the container as default values). Moreover, this page is dedicated to the application but another process so application use page more efficient way. Therefore, even if the page size is smaller than the VM, the effect of the huge page is large, which leads to the utilization of physical memory and the increasing number of functions in the host.

o NUMA

NUMA technology can be used both in the VM-based and containerized infrastructure. Using NUMA, performance will be increasing not CPU and memory but also network since that network interface connected PCIe slot of specific NUMA node have locality. Using NUMA, it requires a strong understanding of VNF's memory requirements. If VNF uses more memory than a single NUMA node contains, the overhead will be occurred due to being spilled to another NUMA node.

In the VM-based infrastructure, the hypervisor can perform extracting NUMA topology and schedules VM workloads. In containerized infrastructure, however, it is more difficult to expose the NUMA topology to the container and currently, it is hard to guarantee the locality of memory when the container is deployed to host that has multiple NUMA nodes. For that reason, the instantiation of C-VNFs is somewhat non-deterministic and apparently NUMA-Node agnostic, which is one way of saying that performance will likely vary whenever this instantiation is performed. So, when we use NUMA in the containerized infrastructure, repeated instantiation and testing to quantify the performance variation is required.

# Update from -03 to -04 (1)

- Adding Chapter 6. Benchmarking Experiences
  - Including our testing results of 106-Hackathon
    - User-space network model (v-switch)
      - DPDK / Contiv-vpp
    - Verifying CPU allocation of native Kubernetes CPU Scheduler (v1.6.1)
      - Compare with CPU pinning technology
        - CMK (CPU Manager for K8s)
        - Shared-mode / Exclusive-mode
      - NUMA-affinity
        - Network interface / Container
        - Same / Different NUMA zone
    - Traffic Generator : T-Rex – IMIX traffic

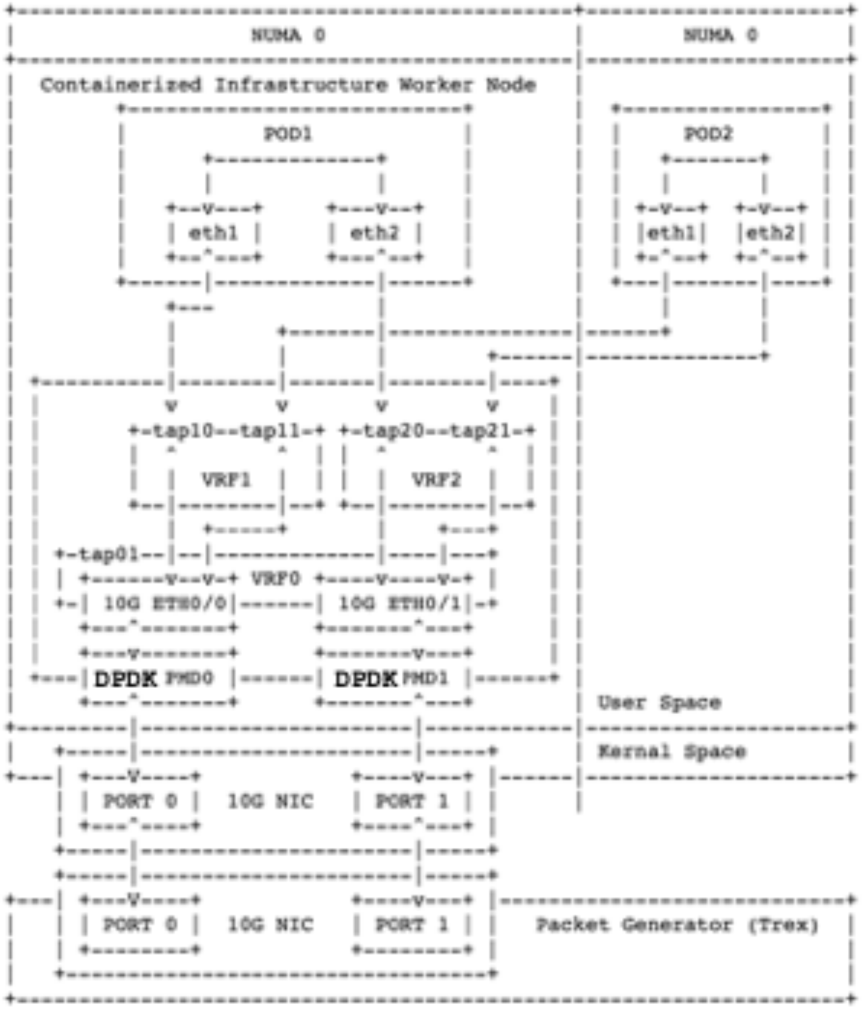
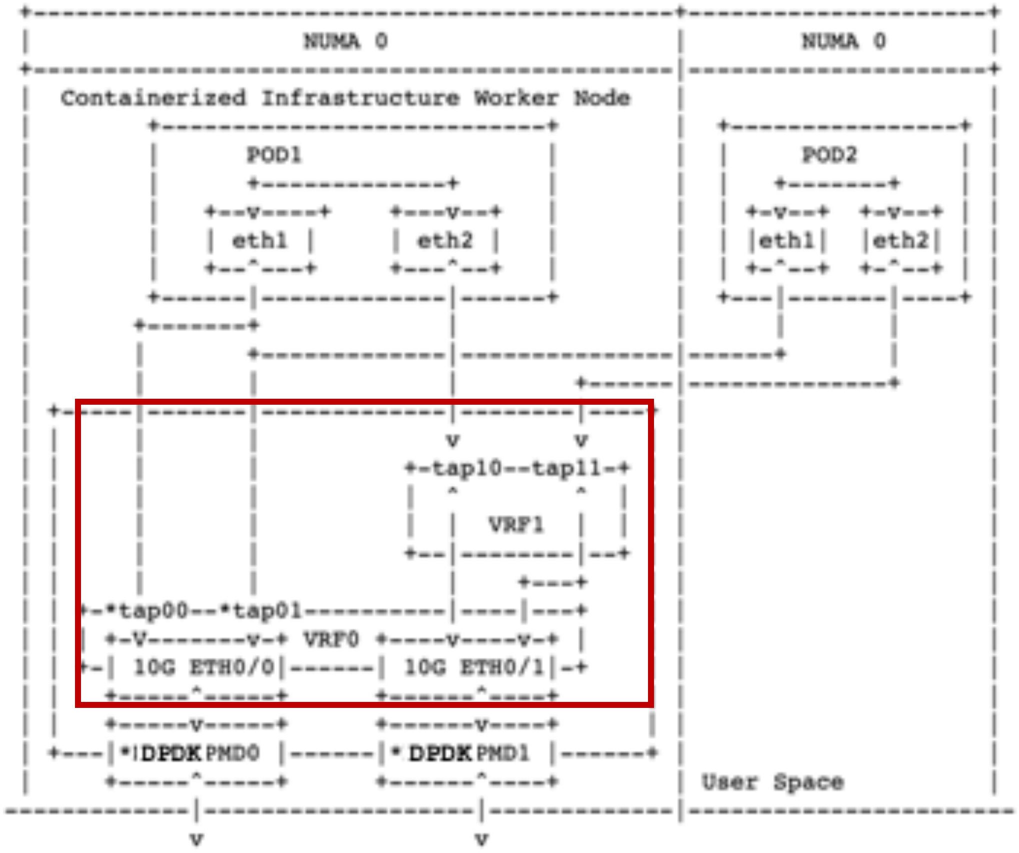


Figure 10: Test Environment-Network Architecture

# Update from -03 to -04 (2)

- Trouble-shootings
  - Routing table doesn’t work when we send packet using T-Rex
    - “IP packet forwarding rule” is processed only default Virtual Routing and Forwarding (VRF0)
    - vrf1 and vrf2 interface couldn’t route packet

  - SOLVED : assigned vrf0 and vrf1 to POD

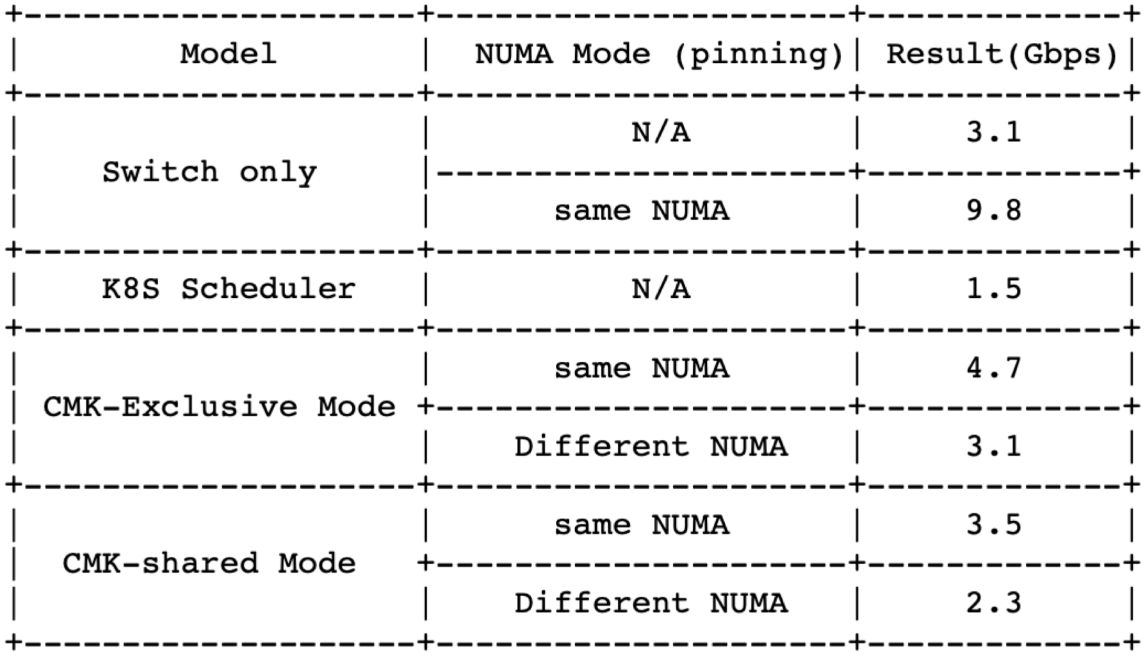


*- CPU pinning interface

Figure 11: Test Environment-Network Architecture(CPU Pinning)

# Update from -03 to -04 (3)

- Test Results
  - Performance is reduced between the vpp-switch and the POD
  - Same-NUMA affinity increased network throughput by about 50%

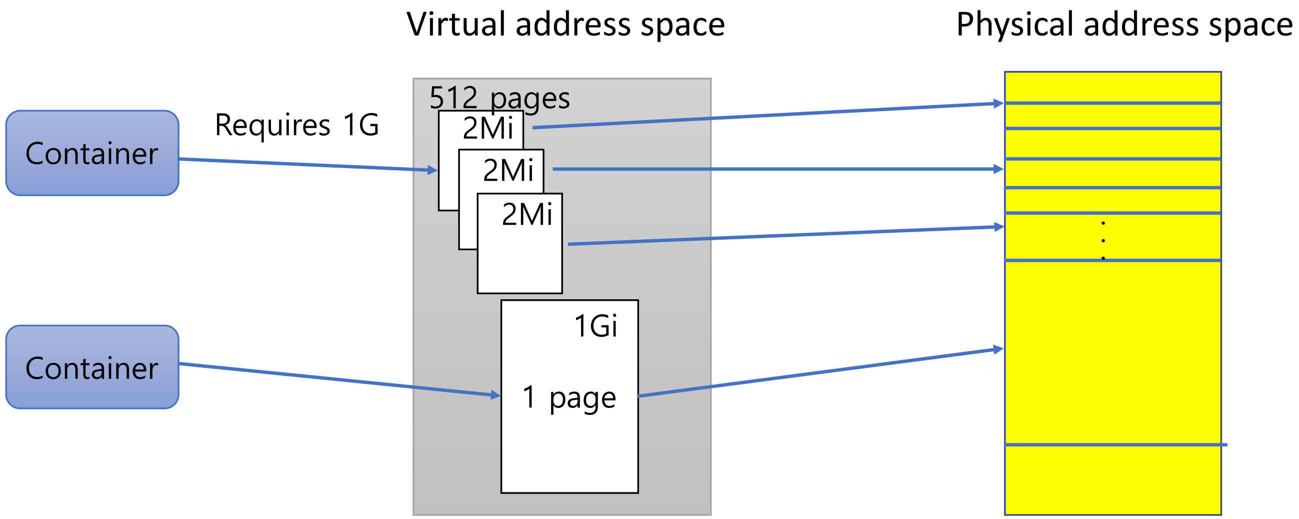


| Model | NUMA Mode (pinning) | Result(Gbps) |
|---|---|---|
| Switch only | N/A | 3.1 |
| | same NUMA | 9.8 |
| K8S Scheduler | N/A | 1.5 |
| CMK-Exclusive Mode | same NUMA | 4.7 |
| | Different NUMA | 3.1 |
| CMK-shared Mode | same NUMA | 3.5 |
| | Different NUMA | 2.3 |

Figure 12: Test Results

# Update from -04 to -05

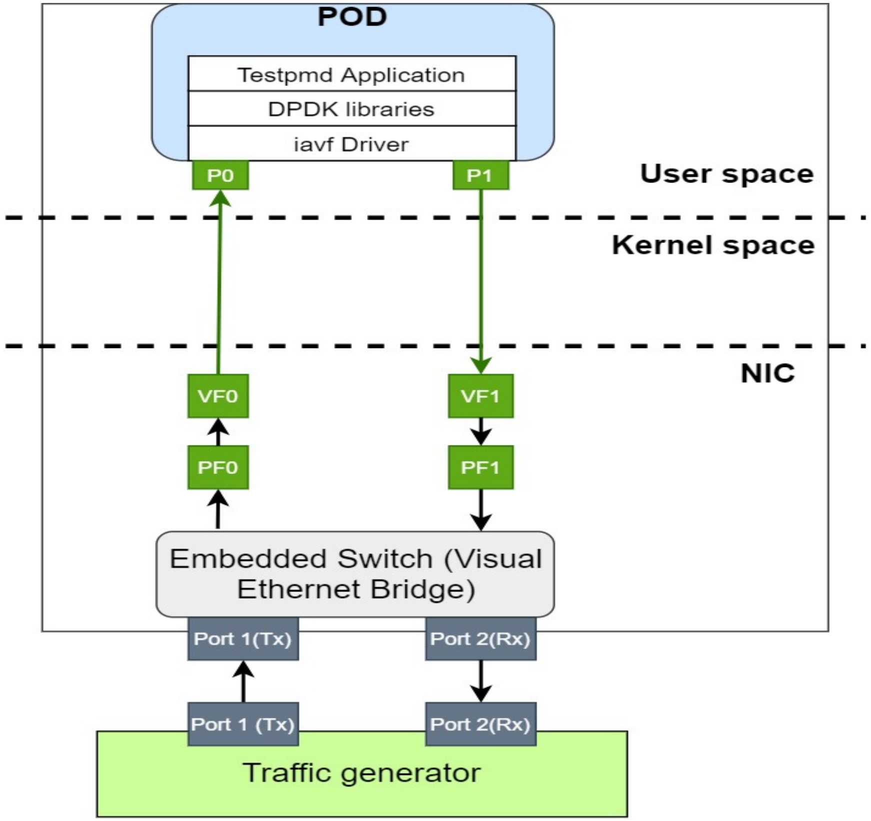
- Adding Chapter 7 – Benchmarking Experiment
  - We implemented and tested at this Hackathon 109
  - User-space network model (device pass-through)
    - SR-IOV / DPDK
  - Verifying huge pages impacts on network performance



# 109 Hackathon review (1)

- Infrastructure Setting
  - https://github.com/huyng14/bmwg-container-network

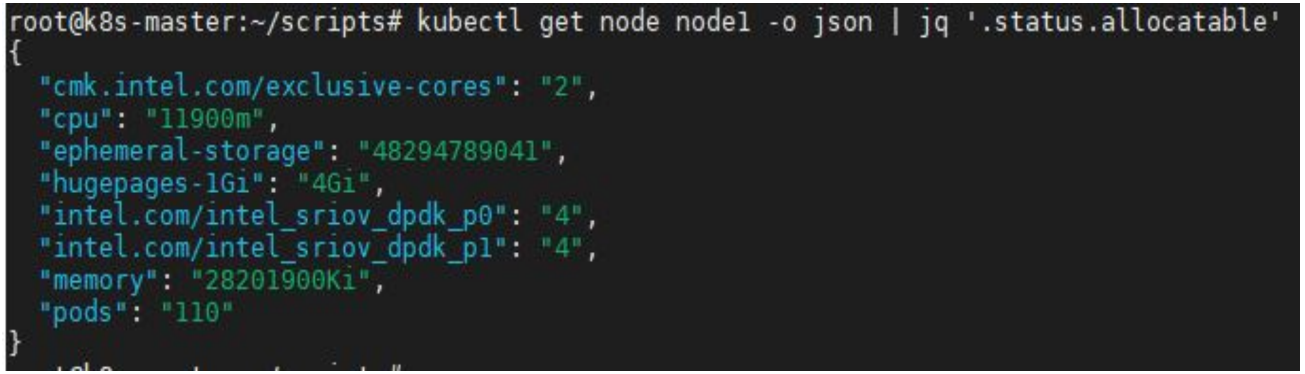
Hackathon mirror workshop at Busan, South Korea : Collaborated with IPWAVE and I2NSF team (SKKU)



- Physical HW specs are same that we used in the 106 Hackathon
- Kubernetes (1 master, 1 worker)
  - MULTUS CNI
  - CMK for CPU Pinning
  - SR-IOV plugin with DPDK

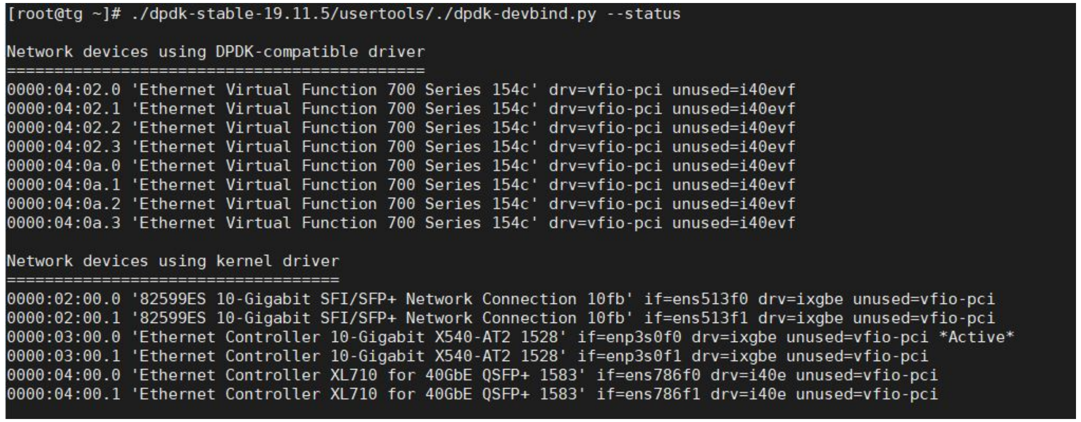
# 109 Hackathon review (2)

- Test Scenario
  - 4GB memory for each container
  - Hugepage setting
    - 2Mi * 2,048 pages / 1Gi * 4 pages
  - Traffic pattern (using T-Rex)
    - Ethernet frame – 64 / 128 / 256 / 1024 / 1518 (bytes) [RFC2544]



```
root@k8s-master:~/scripts# kubectl get node node1 -o json | jq '.status.allocatable'
{
  "cmk.intel.com/exclusive-cores": "2",
  "cpu": "11900m",
  "ephemeral-storage": "48294789041",
  "hugepages-1Gi": "4Gi",
  "intel.com/intel_sriov_dpdk_p0": "4",
  "intel.com/intel_sriov_dpdk_p1": "4",
  "memory": "28201900Ki",
  "pods": "110"
}
```

Huge page setting

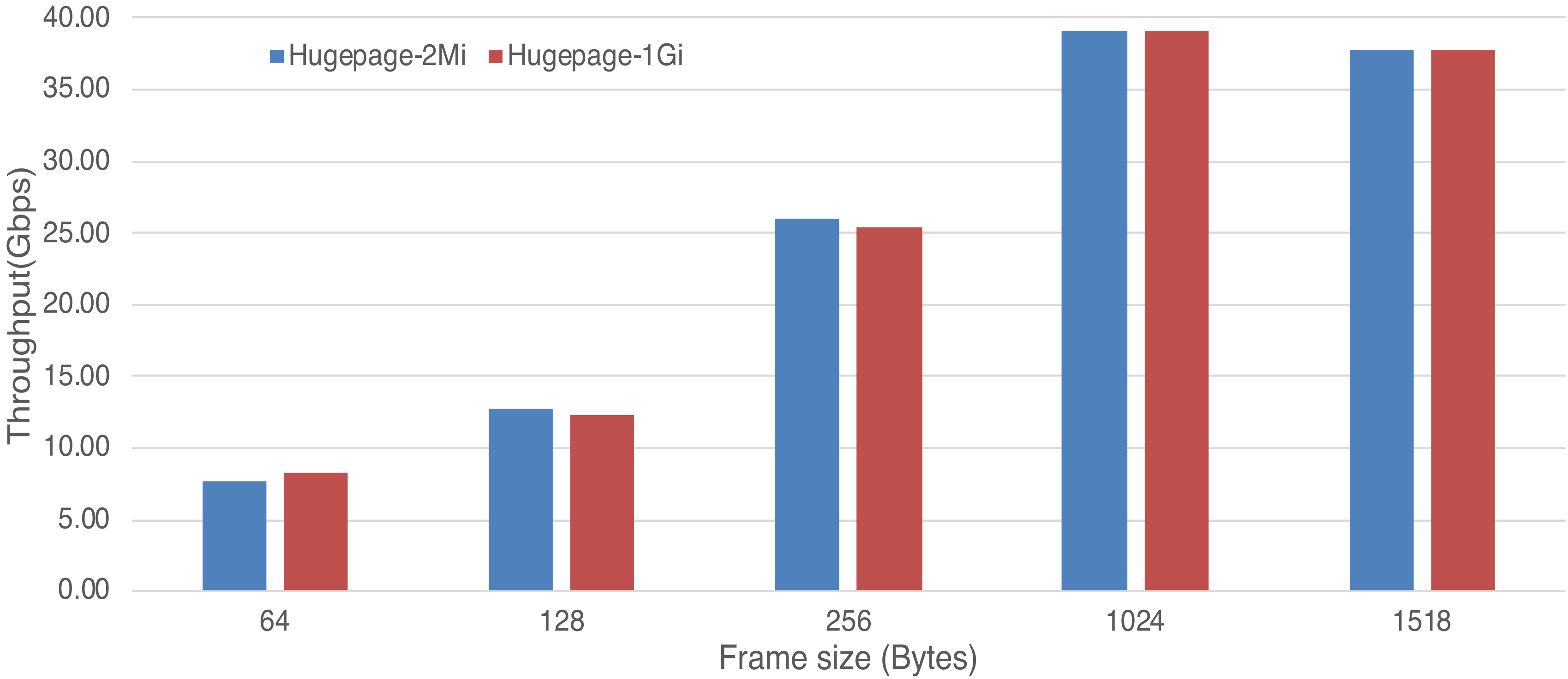


```
[root@tg ~]# ./dpdk-stable-19.11.5/usertools/./dpdk-devbind.py --status

Network devices using DPDK-compatible driver
============================================
0000:04:02.0 'Ethernet Virtual Function 700 Series 154c' drv=vfio-pci unused=i40evf
0000:04:02.1 'Ethernet Virtual Function 700 Series 154c' drv=vfio-pci unused=i40evf
0000:04:02.2 'Ethernet Virtual Function 700 Series 154c' drv=vfio-pci unused=i40evf
0000:04:02.3 'Ethernet Virtual Function 700 Series 154c' drv=vfio-pci unused=i40evf
0000:04:0a.0 'Ethernet Virtual Function 700 Series 154c' drv=vfio-pci unused=i40evf
0000:04:0a.1 'Ethernet Virtual Function 700 Series 154c' drv=vfio-pci unused=i40evf
0000:04:0a.2 'Ethernet Virtual Function 700 Series 154c' drv=vfio-pci unused=i40evf
0000:04:0a.3 'Ethernet Virtual Function 700 Series 154c' drv=vfio-pci unused=i40evf

Network devices using kernel driver
===================================
0000:02:00.0 '82599ES 10-Gigabit SFI/SFP+ Network Connection 10fb' if=ens513f0 drv=ixgbe unused=vfio-pci
0000:02:00.1 '82599ES 10-Gigabit SFI/SFP+ Network Connection 10fb' if=ens513f1 drv=ixgbe unused=vfio-pci
0000:03:00.0 'Ethernet Controller 10-Gigabit X540-AT2 1528' if=enp3s0f0 drv=ixgbe unused=vfio-pci *Active*
0000:03:00.1 'Ethernet Controller 10-Gigabit X540-AT2 1528' if=enp3s0f1 drv=ixgbe unused=vfio-pci
0000:04:00.0 'Ethernet Controller XL710 for 40GbE QSFP+ 1583' if=ens786f0 drv=i40e unused=vfio-pci
0000:04:00.1 'Ethernet Controller XL710 for 40GbE QSFP+ 1583' if=ens786f1 drv=i40e unused=vfio-pci
```

DPDK combining with SRIOV-VF

# 109 Hackathon review (3)

- Test Results
  - The huge page size does not affect the network performance
    - Ethernet frame is limited to 1518 Bytes
  - Just for networking, small size of huge page is enough



# 109 Hackathon review (4)

- Trouble-shootings & Issues
    - Out of memory error
        - Sometimes POD access to the non-allocated memory
    - Just 1 huge page size should be tested for each time
        - Different configuration of Grub, plugins and K8s should be required for each time and should be repeated
        - It takes a lot of time to change configuration, and also high risk to be error
    - Huge page impacts to application process
        - Depending on application, performance of accessing memory can impacts performance
    - Scalability
        - Huge page may impacts resource utilization / scalability of container functions
        - We will consider to figure out trade-offs between performance and resource utilization

# Next Steps

- Draft update to -06
  - Including our results and trouble-shootings of 109 Hackathon
  - Updating up-to-date networking technology
  - Also we consider to expand benchmarking scenario
    - East-west traffic benchmarking
- We also will plan to the next Hackathon
  - Finding items
    - Different network technologies, different test scenarios
- All comments, suggestions and questions are welcome