Considerations for Benchmarking Network Performance in Containerized Infrastructures

draft-dcn-bmwg-containerizedinfra-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
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
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
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%

<table>
<thead>
<tr>
<th>Model</th>
<th>NUMA Mode (pinning)</th>
<th>Result(Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch only</td>
<td>N/A</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>same NUMA</td>
<td>9.8</td>
</tr>
<tr>
<td>K8S Scheduler</td>
<td>N/A</td>
<td>1.5</td>
</tr>
<tr>
<td>CMK-Exclusive Mode</td>
<td>same NUMA</td>
<td>4.7</td>
</tr>
<tr>
<td></td>
<td>Different NUMA</td>
<td>3.1</td>
</tr>
<tr>
<td>CMK-shared Mode</td>
<td>same NUMA</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>Different NUMA</td>
<td>2.3</td>
</tr>
</tbody>
</table>

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](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'
{
  "cpu": "11990m",
  "ephemeral-storage": "48294789641",
  "hugepages-1Gi": "4Gi",
  "intel/intel_sriov_dpdk_p0": "4",
  "memory": "28281960K1",
  "pods": "110"
}
```

```
[root@k8s-master ]# /usr/local/bin/kubectl get node node1 -o json
...
"memory": "28281960K1",
"pods": "110"
```

Huge page setting

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

![Graph showing throughput vs frame size for Hugepage-2Mi and Hugepage-1Gi]
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