Advanced High Performance Networking Solutions for Red Hat® OpenShift®

Using Chelsio Virtual Functions

Executive Summary

Containers are lightweight and self-contained environments that bundle an application's components and dependencies, enabling seamless deployment across diverse infrastructures. Their compact and flexible design supports deployment across bare metal, public, private, and hybrid cloud environments, enhancing scalability and operational efficiency. Red Hat® OpenShift® enhances the container experience by providing an enterprise-grade platform that includes in-built developer tools, advanced security measures, and comprehensive enterprise support.

This paper describes how Chelsio T6 adapters provide high-performance and low-latency Ethernet networking for containers operating in a multi-host OpenShift environment. Chelsio adapters support Single Root I/O Virtualization (SR-IOV) technology, which segments a single network device into multiple Virtual Functions (VFs). This segmentation enables direct input/output (IO) access for containers. If OpenShift is deployed with the Chelsio T6 adapters, it can support up to 64 VFs through an integrated virtual switch. By integrating Chelsio T6 adapters with OpenShift, you can achieve improved network throughput, minimized latency, and enhanced scalability.

Test Results

Figure 1 shows the throughput and CPU usage results of Chelsio VFs attached to containers of different hosts. The results are collected using the **Iperf v2.2.0** tool with I/O size from 64 Bytes to 512 Kbytes and eight parallel connections.

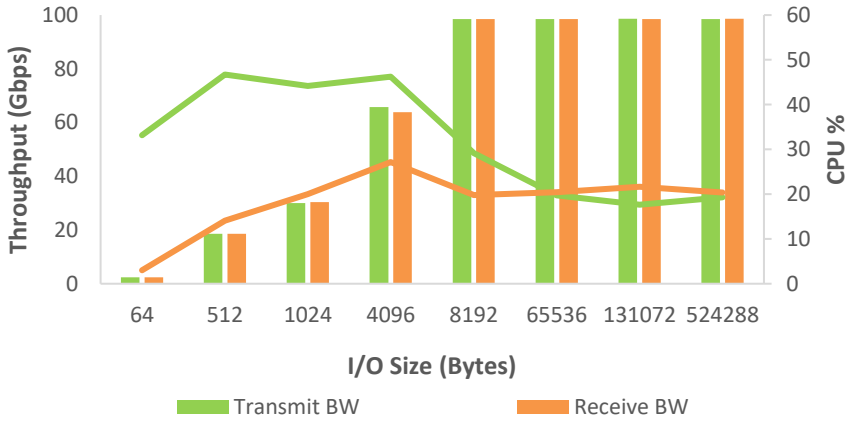

Figure 1 – VF Throughput and %CPU across different nodes vs. I/O size

Chelsio VFs consistently achieve line-rate throughput of 98 Gbps in both transmit and receive directions, optimizing network performance for containerized workloads. CPU utilization on the host is capped at just 46% even at lower I/O sizes, ensuring ample processing capacity for additional workloads.

Network Topology

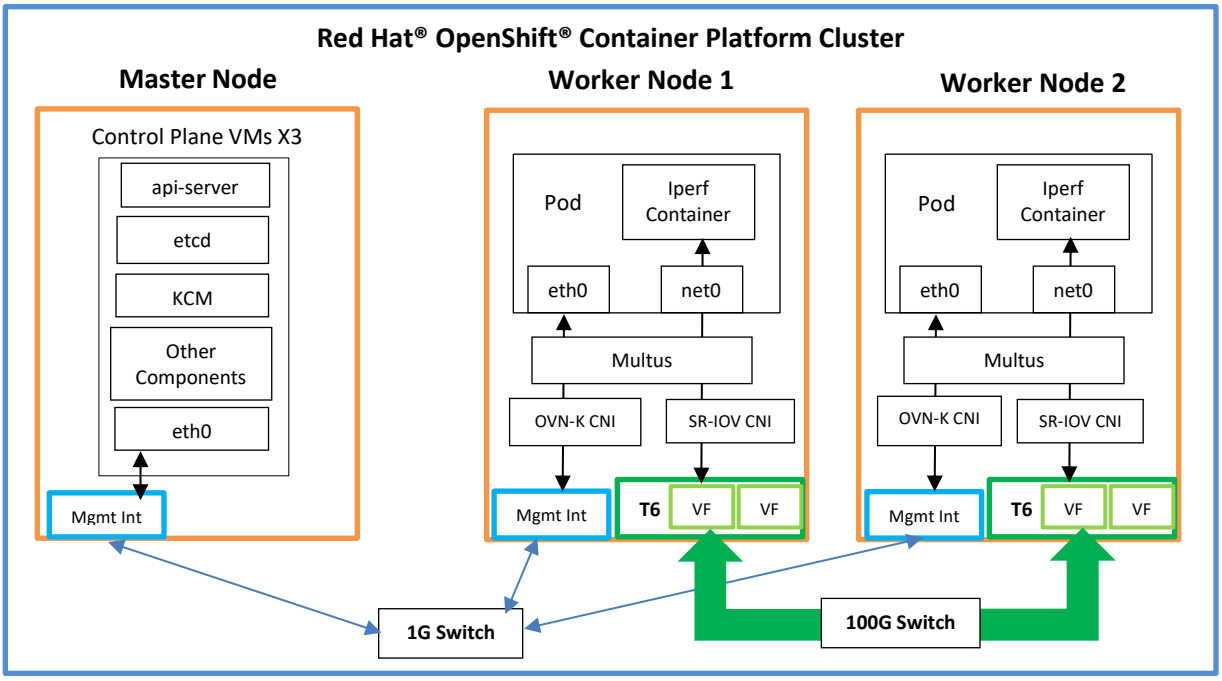

Figure 2 – Test Setup

The test environment features a five-node cluster with one master node running three control plane VMs, and two worker nodes. All nodes are connected through on-board interfaces to a 1G switch handling cluster management traffic and are configured with Red Hat® OpenShift® Container Platform v4.16.10.

Hardware Configuration

- **Control Plane VMs:** Each VM is configured with 1 Xeon(R) CPU E5-2650 v3 8-core @ 2.30GHz (HT disabled) vCPU, 16GB of Virtual Memory and RHCOS v4.16 OS (5.14.0-427.33.1.el9_4.x86_64 kernel).

- **Worker Nodes:** Each Worker Node is configured with two Intel(R) Xeon(R) CPU E5-2650 v3 10-core @ 2.30GHz (HT disabled) CPUs, 128GB of RAM, RHCOS v4.16 OS (5.14.0-427.33.1.el9_4.x86_64 kernel) and Chelsio T62100-CR Adapter.
    - Single port on each worker node is connected to a 100G switch.
    - 1 VF is brought up on each worker node and assigned to a pod.
    - Each pod has an Iperf container.
    - An MTU of 9000B is used on the ports under test.

Setup details and Instructions

Follow these steps to configure and run traffic between VFs in your Red Hat® OpenShift® cluster.

1. Install the OpenShift cluster using an Assisted Installation method by following the steps mentioned in the Red Hat blog.

2. After the installation is complete, connect to the cluster from a helper machine.

    a. Access web GUI of OpenShift cluster and generate a login token for cluster CLI access.
    b. Login to the cluster with OC Command Line utility, using the generated token.

3. Instantiate VF(s) manually on each worker node and set MTU 9000 to the VF interfaces.

```
# modprobe -v cxgb4
# ifconfig <pf4_P0> mtu 9000 up
# echo 1 > /sys/bus/pci/devices/<Chelsio pf0>/sriov_numvfs
# modprobe cxgb4vf
# ifconfig <VF int> mtu 9000
```

4. Label each nodes which are SR-IOV capable.

```
# oc label node <worker-node-name> feature.node.kubernetes.io/network-
sriov.capable="true"
```

5. Install SR-IOV operator from CLI using the below commands (For more information, refer to Red Hat Documentation).

    i) Create an openshift-sriov-network-operator namespace.

```
cat << EOF| oc create -f -
apiVersion: v1
kind: Namespace
metadata:
   name: openshift-sriov-network-operator
   annotations:
     workload.openshift.io/allowed: management
EOF
```

    ii) Create an OperatorGroup Custom Resource (CR).

```
cat << EOF| oc create -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
   name: sriov-network-operators
   namespace: openshift-sriov-network-operator
spec:
   targetNamespaces:
   - openshift-sriov-network-operator
EOF
```

    iii) Create a Subscription CR for the SR-IOV Network Operator.

```
cat << EOF| oc create -f -
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
```

```
metadata:
   name: sriov-network-operator-subscription
   namespace: openshift-sriov-network-operator
spec:
   channel: stable
   name: sriov-network-operator
   source: redhat-operators
   sourceNamespace: openshift-marketplace
EOF
```

iv) Check that the Operator is installed.

```
# oc get csv -n openshift-sriov-network-operator
NAME                         DISPLAY      VERSION   REPLACES   PHASE
sriov-network-operator.v4.16.0-202409111535   SR-IOV Network
Operator   4.16.0-202409111535              Succeeded

# oc get csv -n openshift-sriov-network-operator -o custom-
columns=Name:.metadata.name,Phase:.status.phase
```

6. Configure the SR-IOV Network Operator.

i) Create a SriovOperatorConfig custom resource (CR):

```
cat << EOF| oc create -f -
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovOperatorConfig
metadata:
   name: default
   namespace: openshift-sriov-network-operator
spec:
   disableDrain: false
   enableInjector: true
   enableOperatorWebhook: false
   logLevel: 2
EOF
```

ii) Verify below pods (sriov-network-config-daemon*) are started and running.

```
# oc get pods -A -o wide|grep -i sriov
openshift-sriov-network-operator      network-resources-injector-5gcvq      1/1     Running 0
       22s     10.129.2.54    ocpus5-cp-2           <none>         <none>
openshift-sriov-network-operator      network-resources-injector-s2gm5      1/1     Running 0
       22s     10.129.0.64      ocpus5-cp-3  <none>         <none>
openshift-sriov-network-operator      network-resources-injector-vkm9x      1/1     Running 0
       22s     10.128.0.53      ocpus5-cp-1  <none>         <none>
openshift-sriov-network-operator      sriov-network-config-daemon-b9ddq     1/1     Running 0
       22s     10.192.193.65    ocpus5-wk-2  <none>         <none>
openshift-sriov-network-operator      sriov-network-config-daemon-pbd6w     1/1     Running 0
       22s     10.192.205.195   ocpus5-wk-1  <none>         <none>
openshift-sriov-network-operator      sriov-network-operator-f8897c5d5-h4h2r       1/1
       Running 0       3m46s   10.128.0.51      ocpus5-cp-1  <none>         <none>
```

7. Create a policy to configure an SR-IOV network device.

```
# cat SriovNetworkNodePolicy.yaml

cat << EOF | oc create -f -
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetworkNodePolicy
metadata:
```

```
    name: chelsio-sriov-node
    namespace: openshift-sriov-network-operator
  spec:
    resourceName: chelsiopf0vfs
    numVfs: 2
    nodeSelector:
      feature.node.kubernetes.io/network-sriov.capable: "true"
    deviceType: netdevice
    externallyManaged: true
    nicSelector:
      deviceID: 680d
      rootDevices: ["0000:81:00.0"]
      vendor: "1425"
      priority: 99
  EOF
```

8. Verify below pods (sriov-device-plugin*) are started and running, also discovers the SR-IOV devices on each worker node.

```
# oc get pods -A |grep -i sriov
openshift-sriov-network-operator      network-resources-injector-5gcvq      1/1     Running 0
       36m
openshift-sriov-network-operator      network-resources-injector-s2gm5      1/1     Running 0
       36m
openshift-sriov-network-operator      network-resources-injector-vkm9x      1/1     Running 0
       36m
openshift-sriov-network-operator      sriov-device-plugin-5wmp5             1/1     Running 0
       6s
openshift-sriov-network-operator      sriov-device-plugin-m9bpg             1/1     Running 0
       6s
openshift-sriov-network-operator      sriov-network-config-daemon-b9ddq     1/1     Running 0
       36m
openshift-sriov-network-operator      sriov-network-config-daemon-pbd6w     1/1     Running 0
36m
openshift-sriov-network-operator      sriov-network-operator-f8897c5d5-h4h2r       1/1
       Running 0       40m

# oc logs -n openshift-sriov-network-operator sriov-device-plugin-5wmp5
.
.
.
I0927 13:28:46.572797  1      manager.go:121] Creating new ResourcePool: chelsiopf0vfs
I0927 13:28:46.572802         1      manager.go:122] DeviceType: netDevice
I0927 13:28:46.574996         1      utils.go:82] Devlink query for eswitch mode is not
supported for device 0000:81:01.0. error getting devlink device attributes for net device
0000:81:00.0 no such device
I0927 13:28:46.578617         1      utils.go:82] Devlink query for eswitch mode is not
supported for device 0000:81:01.4. error getting devlink device attributes for net device
0000:81:00.0 no such device
I0927 13:28:46.582108         1      manager.go:138] initServers(): selector index 0 will
register 2 devices
I0927 13:28:46.582127         1      factory.go:124]     device     added:     [identifier:
0000:81:01.0, vendor: 1425, device: 680d, driver: cxgb4vf]
I0927 13:28:46.582135  1      factory.go:124]  device  added:  [identifier:  0000:81:01.4,
vendor: 1425, device: 680d, driver: cxgb4vf]
I0927 13:28:46.582159         1      manager.go:156] New resource server is created for
chelsiopf0vfs ResourcePool
I0927 13:28:46.582169         1      main.go:74] Starting all servers...
I0927 13:28:46.582467         1      server.go:255] starting chelsiopf0vfs device plugin
endpoint at: openshift.io_chelsiopf0vfs.sock
I0927 13:28:46.583432         1      server.go:283] chelsiopf0vfs device plugin endpoint
started serving
I0927 13:28:46.583567  1      main.go:79] All servers started.
.
.
.
```

9. Create a policy to configure a network for the SR-IOV devices.

```
cat << EOF | oc create -f -
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetwork
metadata:
  name: chelsio-sriov-network
  namespace: openshift-sriov-network-operator
spec:
  networkNamespace: default
  resourceName: chelsiopf0vfs
  ipam: |
    {
      "type": "host-local",
      "subnet": "102.100.1.0/24",
      "rangeStart": "102.100.1.100",
      "rangeEnd": "102.100.1.200"
    }
```

10. Create two pods, each assigned to a different worker node, utilizing the Chelsio SR-IOV device within the container.

```
# oc create -f sriov-pod1.yaml
# oc create -f sriov-pod2.yaml

# cat sriov-pod1.yaml
apiVersion: v1
kind: Pod
metadata:
  name: sriovpod1
  namespace: default
  annotations:
  k8s.v1.cni.cncf.io/networks: '[
        {
                "name": "chelsio-sriov-network"
        }
]'
spec:
  nodeName: ocpus5-wk-1
  containers:
  - name: sriov-container-wk-1
    image: "localhost/centos9-iperf"
    imagePullPolicy: IfNotPresent
    command: ["sleep", "infinity"]

# cat sriov-pod2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: sriovpod2
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: '[
        {
                "name": "chelsio-sriov-network"
        }
]'
spec:
  nodeName: ocpus5-wk-2
  containers:
```

```
  - name: sriov-container-wk-2
    image: "localhost/centos9-iperf"
    imagePullPolicy: IfNotPresent
    command: ["sleep", "infinity"]
```

11. Verify the IP address assigned to VF interface from each container using below command.

```
# oc describe pod/sriovpod2 |tail
```

12. Execute the following OC commands to start traffic using the iperf tool.

    From terminal_1 command prompt, start iperf server from pod2 container:

```
# oc rsh -c sriov-container-wk-2 sriovpod2 -- iperf -s
```

    From terminal_2 command prompt, start iperf client initiated from pod1 container:

```
# oc exec -c sriov-container-wk-1 sriovpod1 -- iperf -c 102.100.1.102 -t60 -
i5 -P8 -l <IO size>
```

Conclusion

Delivering line-rate 98 Gbps throughput with just 46% host CPU utilization, Chelsio offers industry-leading TCP/IP SR-IOV networking solutions that enhance performance and scalability in Red Hat® OpenShift® environments. With a lot of CPU resources left free on the host, more number of containers can be deployed. In addition to SR-IOV, Chelsio adapters support offloading multiple network, compute, storage, and security protocols to deliver industry-leading performance and efficiency. All the traffic runs over a single network, rather than building and maintaining multiple networks, resulting in significant acquisition and operational cost savings.

Related Links

High Performance Network for Kubernetes
Windows TCP/IP SR-IOV in Virtual Environments
Windows d.VMMQ Performance