WHITE PAPER

Communications Service Providers
Characterizing VNF Performance

NUMA-Aware Hypervisor and Impact on Brocade* 5600 vRouter

Using a Brocade* 5600 vRouter as an example, this paper shows how a VNF can be used on a dual-socket COTS server, highlighting the impact of non-uniform memory access (NUMA) on the VNF's performance.

Author

Xavier Simonart

Intel

Table of Contents

Executive Summary

Many papers characterizing virtual network functions (VNF) performance use only one socket of dual-sockets commercial off-the-shelf (COTS) server. In some cases, both sockets are used independently by two VNFs. In the case of a vRouter for instance, this would mean that two independent routers would run on the dual-sockets system. All interfaces could not be connected full-mesh.

This document shows how a Brocade* 5600 vRouter can be used on a dual-socket commercial off-the-shelf (COTS) server, in cross-socket full-mesh traffic configurations. It highlights the impact of non-uniform memory access (NUMA) on the performance of VNF applications, using a Brocade 5600 vRouter. It shows the importance of a NUMA-aware QEMU and the influence of QPI.

Figure 1 shows the performance of a Brocade 5600 vRouter and the performance impact when the traffic is using the QPI link. In this setup, traffic from one interface is always routed to one and exactly one (other) interface, either on the same CPU socket, or on the other CPU socket. Other traffic profiles (e.g., traffic going from one interface to the other three interfaces on the same socket) might highlight different performances. In the rest of this paper, vRouter and Brocade 5600 refer to Brocade 5600 vRouter.

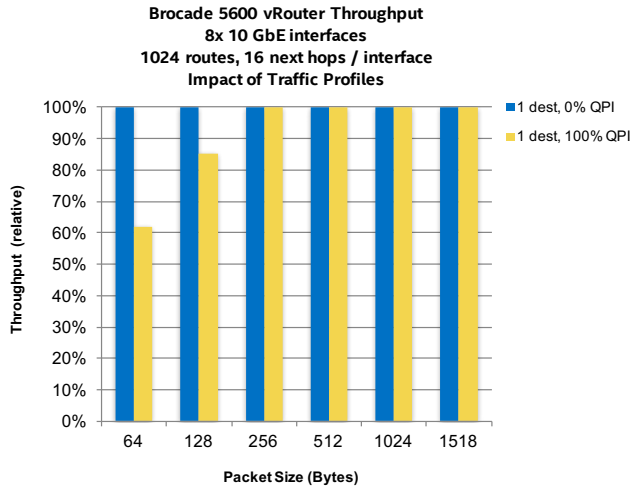


Figure 1. Impact of traffic profile on vRouter's throughput[1]

Even with traffic sent over QPI links, vRouter shows no drop in performance for packet size above 256 bytes.

[1] Intel internal analysis. See Section 3 for the system under test's configuration details, and Section 4 for the test generators' configuraton details.

1 Use-Case Details

Many papers characterizing VNF performance usually use only one socket of dual-sockets systems. In some cases, both sockets are used independently by two VNFs.

In the case of a vRouter for instance, this would mean that two independent routers would run on the dual-sockets system. If such a dual-sockets system is able to handle eight 10 GbE interfaces, the traffic could not go from any interface to any interface (Figure 2): for instance, traffic from interface 1 cannot be forwarded to interface 5.

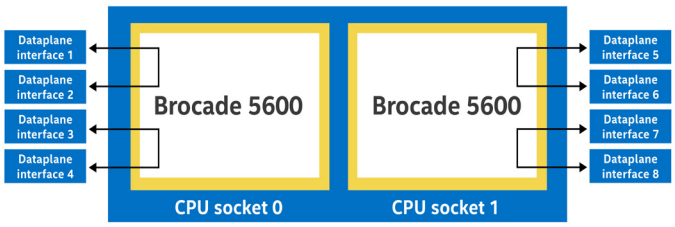


Figure 2. Two vRouter instances

In some cases, it might be required to support a full-mesh eight 10 GbE ports vRouter. Hence, it is interesting to assess the performance demonstrated by such a vRouter configuration, first (Figure 3) using the same traffic as in Figure 2 (i.e., a traffic not crossing the inter-socket link), then using a traffic crossing the inter-socket link (Figure 4).

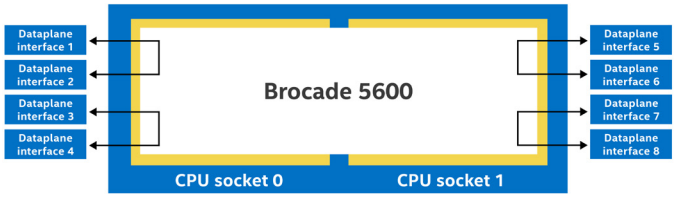


Figure 3. One vRouter instance, no inter-socket traffic

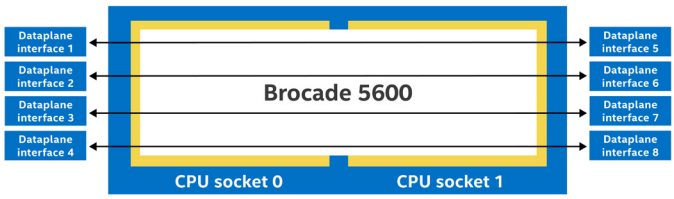


Figure 4. One vRouter instance, with inter-socket traffic

In all three cases, the traffic is setup in such a way that all traffic from one interface is always routed to one (and exactly one) other interface.

- For Figure 2 and Figure 3 this means for instance that all traffic from interface 1 is sent to interface 2, and from interface 2 to interface 1, etc.
- For Figure 4 it means that all traffic from interface 1 is sent to interface 5, from interface 5 to interface 1, etc.

Different traffic profiles (where, for instance, traffic from interface 1 might be routed to interface 2 to 4, or even 1 to 4) will highlight different performance results.

The Brocade 5600 vRouter is running in a virtual machine, using QEMU as the hypervisor and CentOS as the host operating system (see 3.1.1 for hardware and software details). PCI pass-through is being used,[2] i.e., the control of the full physical device is given to the virtual machine; there is no virtual switch involved in the fast path (see Figure 5, using two instances of the vRouter, and Figure 6, using one instance spanning both CPU sockets).

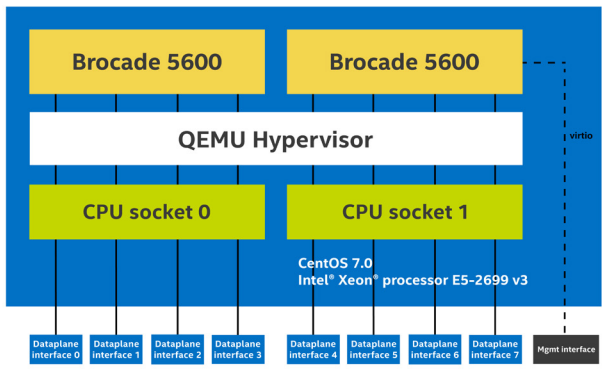


Figure 5. Two vRouter instances

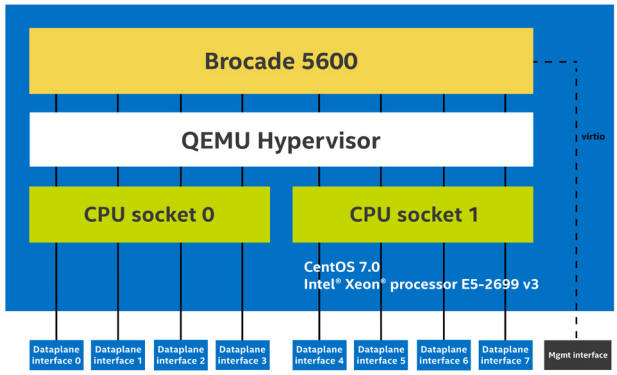


Figure 6. One vRouter instance

The vRouter is characterized under network load created by test generators: the test generators generate IP traffic towards four or eight 10 Gbps interfaces, and they measure the traffic coming from those interfaces. Those test generators can be Ixia* (or Spirent*) or COTS servers running DPDK-based applications (pktgen or prox).

For automation purposes, prox (https://01.org/intel-data-plane-performance-demonstrators/prox-overview) has been used to generate the traffic and to measure the throughput and latency from the Brocade 5600 vRouter.[3] Ixia has been used as well to confirm some key data results.

[2] PCI-Pass-through was chosen to stay focused on CPU characteristics and not be distracted by vNIC/NIC capabilities and does not reflect on what the Brocade 5600 vRouter supports

[3] Choice of test generator is simply based on engineer's preference and has no known impact on the performance numbers.

2 Test Results

2.1 One Brocade 5600 vRouter Instance (Two Sockets) – No Cross Socket Traffic

The goal of this test is to see which performance penalty is paid when one VNF uses both CPU sockets (see Figure 6) instead of two VNFs, each running on its own CPU socket (Figure 5).

Figure 7 shows the performance obtained when one Brocade 5600 vRouter instance uses interfaces from both sockets and cores from both sockets. It is compared with two instances, each running on its own socket.

Two different versions of QEMU are also compared: QEMU 1.5.3 compared to QEMU 2.4.1.

QEMU 1.5.3 is the default QEMU version included in CentOS 7.1. With this QEMU version, PCI devices passed-through to the VM cannot be associated to a NUMA node.

QEMU 2.4.1 is the latest QEMU version (at the time of writing) available from open source. This QEMU 2.4.1 has better support for NUMA, as the VM can be configured with knowledge about the NUMA nodes:

- VCPUs on NUMA node
- Huge pages on NUMA nodes
- PCI devices on NUMA node

We see in Figure 7 that the performance gain using QEMU 2.4.1 versus QEMU 1.5.3 is very important.[4] Even in the best case scenario where the traffic does not cross the QPI link, QEMU 1.5.3, not fully NUMA aware, is severely impacted by running on both CPU sockets. Even though the traffic does not cross CPU sockets, packets handling on socket 0 results in many cases of memory being used on socket 1, generating intensive QPI traffic. Using QEMU 2.4.1, there is no performance loss in using one vRouter instance of two CPU sockets instead of using two instances, each on one CPU socket. QPI traffic in this case is minimal.

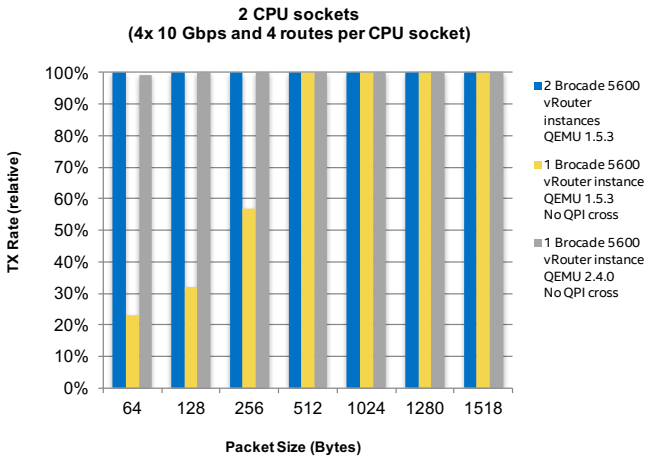


Figure 7. Impact of configuration on vRouter throughput[4]

Deploying a single vRouter instance on a dual socket server would deliver the same performance as two separated vRouter instances as long as NUMA-aware QEMU is used and as long as the QPI link is not actually utilized.

2.2 One Brocade 5600 vRouter Instance (Two Sockets) – Cross Socket Traffic

In the previous test result, the traffic was configured in such a way that it does not cross the CPU sockets (Figure 3), so that we were able to compare two instances of Brocade 5600 vRouter (where it is not possible for the traffic to cross the QPI link) and one instance of the Brocade 5600 vRouter.

In this chapter, we will check the influence of having traffic crossing the inter-socket link, taking full benefit of using only one instance of the vRouter (Figure 4).

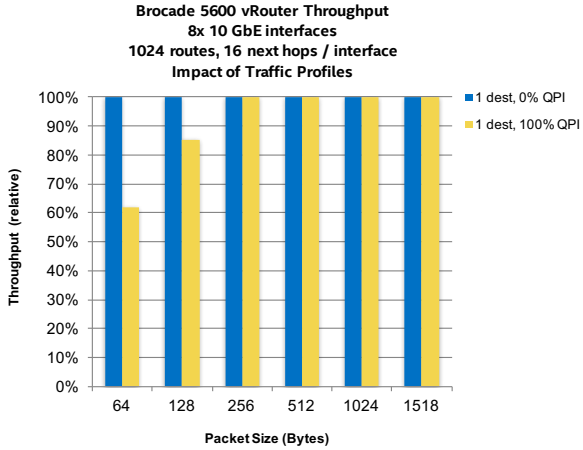


Figure 8. Impact of traffic profile on vRouter throughput[4]

We see that the performance is lower when the traffic crosses the CPU sockets. Still we can see that with any packet sizes bigger than 256 bytes, traffic line rate is reached.[4] Those results were obtained with the traffic profiles described in Figure 3 and Figure 4 (i.e., all packets from each incoming interface are always sent to only one other outgoing interface). Different traffic profiles would result in different performance results.

3 System Under Test's Configuration

3.1 Host Configuration

3.1.1 Hardware and Software Details

ITEM	DESCRIPTION	NOTES
Platform	Intel® Server Board S2600WT Family	
Form factor	2U Rack Mountable	
Processor(s)	2x Intel® Xeon® CPU E5-2699 v3	46080KB L3 Cache per CPU, 18 cores per CPU (36 logical cores due to Hyper-threading).
Memory	32 GB RAM (8x 4 GB) per socket	Quad channel 2134 DDR4
BIOS	SE5C610.86B.01.01.0009.060120151350	Hyper-threading enabled Hardware prefetching enabled COD disabled

[4] Intel internal analysis. See Section 3.1 for the system under test's configuration details.

ITEM	DESCRIPTION	NOTES
Host OS	CentOS 7.1	Kernel version: 3.10.0-229.7.2.el7.x86_64
Hypervisor	QEMU	2.4.1
vRouter	Brocade 5600 vRouter v4.0R1	
Hugepages	8x 1 GB on host or 16x 1 GB on host 5x 1 GB in VM	16x used when using 8x 10 Gbps interfaces
DPDK	used	Used in Brocade 5600 vRouter; version unknown
NICs	8x Intel® 82599 10 Gigabit Ethernet Controller	2 dual-port PCIe gen-2 cards on socket 0 and 2 on socket 1.

3.1.2 Grub.cfg

DPDK-related CPU cores must be isolated through isolcpus in grub.cfg, by making sure interrupts are handled by core 0.

```
     linux16 /vmlinuz-3.10.0-229.11.1.el7.
x86_64 root=/dev/mapper/centos-root ro
rd.lvm.lv=centos/swap vconsole.keymap=us
ipv6.disable=1 crashkernel=auto  vconsole.
font=latarcyrheb-sun16 rd.lvm.lv=centos/
root selinux=0 rhgb quiet LANG=en_US.UTF-8
intel_iommu=on iommu=pt noirqbalance
intel_pstate=disable intel_idle.max_
cstate=0 processor.max_cstate=0 default_
hugepagesz=1G hugepagesz=1G hugepages=16
transparent_hugepage=never
isolcpus=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,
51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,
68,69,70,71
```

3.1.3 QEMU

3.1.3.1 Dependencies

Following packages must be installed to be able to install and build QEMU

```
sudo yum install net-tools
sudo yum install gcc
sudo yum install bzip2
sudo yum install zlib-devel
sudo yum install glib2-devel
sudo yum install gcc-c++
sudo yum install flex bison autoconf automake
libtool
sudo yum install numactl-devel numactl
sudo yum install pciutils
sudo yum install bridge-utils
```

3.1.3.2 QEMU Build

```
tar xvjf qemu-2.4.1.tar.bz2
cd qemu-2.4.1
./configure --disable-attr --enable-kvm
--enable-vhost-net --disable-docs --disable-
vnc-png --disable-vnc-jpeg --disable-sdl
--disable-curl --disable-curses --disable-
vnc-sasl --disable-vnc-tls --enable-numa
make
make install
```

3.1.4 Scripts

The following sections list the scripts used for the characterization. There is no guarantee that those scripts will run on software or hardware different from the ones listed in this document.

Scripts expect to have the Brocade 5600 vRouter image (`vRouter.img`) in user home directory (`/home/user` in this example). When using two VMs, the image of the second VM is called `vRouter2.img`).

3.1.4.1 Scripts to Prevent Interrupts on DPDK Fast Path

Disable un-used services... (as root).

```
chkconfig abrt-ccpp off
chkconfig abrtd off
chkconfig acpid off
chkconfig atd off
chkconfig auditd off
chkconfig autofs off
chkconfig blk-availability off
chkconfig certmonger off
chkconfig cpuspeed off
chkconfig cups off
chkconfig haldaemon  off
chkconfig firewalld off
chkconfig ip6tables off
chkconfig iptables off
chkconfig irqbalance off
chkconfig kdump off
chkconfig ksmtuned off
chkconfig ksm off
chkconfig libvirt-guests off
chkconfig libvirtd off
chkconfig lvm2-monitor off
chkconfig mcelogd off
chkconfig mdmonitor off
chkconfig messagebus off
chkconfig netfs off
chkconfig nfs off
chkconfig nfslock off
chkconfig portreserve off
chkconfig postfix off
chkconfig rpcbind off
chkconfig rpcgssd off
chkconfig rpcidmapd off
chkconfig sysstat off

service abrt-ccpp stop
service abrtd stop
service acpid stop
service atd stop
service auditd stop
service autofs stop
service blk-availability stop
service certmonger stop
service cpuspeed stop
```

```
service cups stop
service haldaemon  stop
service ip6tables stop
service iptables stop
service irqbalance stop
service kdump stop
# Stop firewall - preventing vnc to qemu
service firewalld stop
service ksmtuned stop
service ksm stop
service libvirt-guests stop
service libvirtd stop
service lvm2-monitor off
service mcelogd stop
service mdmonitor stop
service messagebus stop
service netfs stop
service nfs stop
service nfslock stop
service portreserve stop
service postfix stop
service rpcbind stop
service rpcgssd stop
service rpcidmapd stop
service sysstat stop
```

```
rmmod bluetooth
rmmod rfkill
rmmod cpufreq _ stats
rmmod ip6table _ filter
rmmod ip6 _ tables
rmmod ebtable _ nat
rmmod ebtables
rmmod nf _ conntrack _ ipv4
rmmod nf _ defrag _ ipv4
rmmod xt _ state
rmmod nf _ conntrack
rmmod ipt _ REJECT
rmmod xt _ CHECKSUM
rmmod iptable _ mangle
rmmod iptable _ filter
rmmod ip _ tables
rmmod stp
rmmod llc
rmmod ipv6
rmmod dm _ mirror
rmmod dm _ region _ hash
rmmod dm _ log
rmmod dm _ mod
rmmod vhost _ net
rmmod macvtap
rmmod macvlan
rmmod vhost
rmmod tun
rmmod iTCO _ wdt
rmmod iTCO _ vendor _ support
rmmod microcode
rmmod pcspkr
rmmod lpc _ ich
rmmod mfd _ core
rmmod i2c _ algo _ bit
rmmod dca
rmmod ptp
rmmod pps _ core
```

```
rmmod mdio
rmmod sg
rmmod i2c _ i801
rmmod i2c _ core
rmmod wmi
rmmod ext4
rmmod jbd2
rmmod mbcache
rmmod sd _ mod
rmmod crc _ t10dif
rmmod crct10dif _ common
rmmod ahci
rmmod libahci
rmmod isci
rmmod libsas
rmmod scsi _ transport _ sas

echo 0 > /proc/sys/kernel/nmi _ watchdog
echo 0 > /proc/sys/kernel/numa _ balancing
echo 1 > /sys/bus/workqueue/devices/writeback/cpumask
sysctl vm/stat _ interval=1000000

killall sftp-server
killall udevd

for i in `pgrep rcu[^c]` ; do taskset -pc 0 $i ; done

./set _ irq _ affinity enp3s0f0
```

Where enp3s0f0 is the management interface and set_irq_affinity is defined by:

```
device=$1
if [ $device = "" ] ; then
  echo "Please select which interface to use"
  exit
fi

i=0
while [ 1 ]; do
  irq=`cat /proc/interrupts | grep -i $device-TxRx-$i"$"  | cut  -d:  -f1 | sed "s/ //g"`
  if [ -n  "$irq" ]; then
    printf "1 > /proc/irq/%d/smp _ affinity\n" $irq
    printf 1 > /proc/irq/$irq/smp _ affinity
    i=$(($i+1))
  else
    exit
  fi
done
```

3.1.4.2 Hugepages, vfio…

```
# Create and mount hugepages
sudo mkdir -p /mnt/huge
sudo mount -t hugetlbfs nodev /mnt/huge

# Create tap device
ip tuntap add tap0 mode tap
ip tuntap add tap1 mode tap
ifconfig tap0 up
ifconfig tap1 up

# Create bridge for mgmt. interface
brctl addbr br0
brctl addif br0 tap0
brctl addif br0 tap1

# Load and bind vfio driver
modprobe vfio-pci
echo "8086 10fb" > /sys/bus/pci/drivers/vfio-pci/new _ id
echo 0000:05:00.0 > /sys/bus/pci/devices/0000\:05\:00.0/driver/unbind
echo 0000:05:00.0 > /sys/bus/pci/drivers/vfio-pci/bind
echo "8086 10fb" > /sys/bus/pci/drivers/vfio-pci/new _ id
echo 0000:05:00.1 > /sys/bus/pci/devices/0000\:05\:00.1/driver/unbind
echo 0000:05:00.1 > /sys/bus/pci/drivers/vfio-pci/bind
echo "8086 10fb" > /sys/bus/pci/drivers/vfio-pci/new _ id
echo 0000:07:00.0 > /sys/bus/pci/devices/0000\:07\:00.0/driver/unbind
echo 0000:07:00.0 > /sys/bus/pci/drivers/vfio-pci/bind
echo "8086 10fb" > /sys/bus/pci/drivers/vfio-pci/new _ id
echo 0000:07:00.1 > /sys/bus/pci/devices/0000\:07\:00.1/driver/unbind
echo 0000:07:00.1 > /sys/bus/pci/drivers/vfio-pci/bind
echo "8086 10fb" > /sys/bus/pci/drivers/vfio-pci/new _ id
echo 0000:81:00.0 > /sys/bus/pci/devices/0000\:81\:00.0/driver/unbind
echo 0000:81:00.0 > /sys/bus/pci/drivers/vfio-pci/bind
echo "8086 10fb" > /sys/bus/pci/drivers/vfio-pci/new _ id
echo 0000:81:00.1 > /sys/bus/pci/devices/0000\:81\:00.1/driver/unbind
echo 0000:81:00.1 > /sys/bus/pci/drivers/vfio-pci/bind
echo "8086 10fb" > /sys/bus/pci/drivers/vfio-pci/new _ id
echo 0000:83:00.0 > /sys/bus/pci/devices/0000\:83\:00.0/driver/unbind
echo 0000:83:00.0 > /sys/bus/pci/drivers/vfio-pci/bind
echo "8086 10fb" > /sys/bus/pci/drivers/vfio-pci/new _ id
echo 0000:83:00.1 > /sys/bus/pci/devices/0000\:83\:00.1/driver/unbind
echo 0000:83:00.1 > /sys/bus/pci/drivers/vfio-pci/bind

service NetworkManager stop

vi /etc/sysconfig/network-scripts/ifcfg-br0
    DEVICE=br0
    #BOOTPROTO=dhcp
    BOOTPROTO=static
    IPADDR=192.168.1.142
    NETMASK=255.255.255.0
    GATEWAY=192.168.1.240
    ONBOOT=yes
    TYPE=Bridge

vi /etc/sysconfig/network-scripts/ifcfg-enp3s0f0
# Generated by dracut initrd
    NAME="enp3s0f0"
    DEVICE="enp3s0f0"
    ONBOOT=yes
    UUID="a4d56fab-015e-458a-bb40-6a08cb8cf8d9"
    TYPE=Ethernet
    BRIDGE=br0
service network restart
```

3.1.4.3 Scripts for Pinning Virtual CPUs to Physical Cores

On the host system, every QEMU thread must be pinned to a different CPU core through taskset.

`start _ vm.py` script provided by prox (in helper-scripts) can be used to launch a VM and pin virtualized cores to physical cores. The right pinning must be provided in `vm-cores.py` script. Threads handling interfaces from one CPU socket should be pinned to the same CPU socket.

3.1.4.3.1 Two Independent VMs - One VM on Each CPU Socket, Four Interfaces per VM

Configure the script to run on 18 cores in vm-cores.py (map virtual cores 0 to 17 to host logical cores 0 to 17)

```
cores = [[0],[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12],[13],[14],[15],[16],[17]]
```

Copy the `start _ vm.py` script to `start _ vm2.py`. Modify it to get core information from vm2-cores.py and configure the `vm2-cores.py` script to run on 18 cores of socket 1 (map virtual cores 0 to 17 to host logical cores 18 to 35).

```
cores = [[18],[19],[20],[21],[22],[23],[24],[25],[26],[27],[28],[29],[30],[31],[32],[33],[34],[35]]
```

3.1.4.3.2 One VM running on Both CPU Sockets, Eight Interfaces

Configure the vm-cores.py script to run on 36 cores (map virtual cores 0 to 35 to host logical cores 0 to 35):

```
cores = [[0],[1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12],[13],[14],[15],[16],[17],[18],[19],[20],[21],
[22],[23],[24],[25],[26],[27],[28],[29],[30],[31],[32],[33],[34],[35]]
```

Modify sockets=1 to sockets=2

3.1.4.4 QEMU Startup Script

See 3.1.4.3 for information on `start _ vm.py` and `start _ vm2.py` scripts used in these sections.

3.1.4.4.1 Two Independent VMs - One VM on Each CPU socket, Four Interfaces per VM

```
python start _ vm.py -name centos -enable-kvm \
        -cpu host \
        -m 8192 \
        -object memory-backend-file,prealloc=yes,mem-path=/mnt/huge,size=8192M,id=ram-
node0,host-nodes=0,policy=bind \
        -numa node,nodeid=0,cpus=0-17,memdev=ram-node0 \
        -uuid 8da5a07e-3b51-4be1-9f24-b8061c1dc271 -boot order=d   \
        -drive file=/home/user/vyatta-kvm _ 4.0R1 _ amd64.img,if=none,id=drive-ide0-0-
1,format=raw \
        -device ide-hd,bus=ide.0,unit=0,drive=drive-ide0-0-1,id=ide0-0-1 \
        -net nic,model=e1000,macaddr=52:54:00:90:f2:a2 \
        -net tap,ifname=tap0,script=no,downscript=no,vhost=on \
        -vnc 192.168.1.142:2 \
        -device vfio-pci,host=05:00.0,id=hostdev0,bus=pci.0,addr=0x3 \
        -device vfio-pci,host=05:00.1,id=hostdev1,bus=pci.0,addr=0x4 \
        -device vfio-pci,host=07:00.0,id=hostdev2,bus=pci.0,addr=0x5 \
        -device vfio-pci,host=07:00.1,id=hostdev3,bus=pci.0,addr=0x6 \
        -device cirrus-vga,id=video0,bus=pci.0,addr=0x7 \
        -daemonize

python start _ vm2.py -name centos -enable-kvm \
        -cpu host \
        -m 8192 \
        -object memory-backend-file,prealloc=yes,mem-path=/mnt/huge,size=8192M,id=ram-
node1,host-nodes=1,policy=bind \
        -numa node,nodeid=0,cpus=0-17,memdev=ram-node1 \
        -uuid 8da5a07e-3b51-4be1-9f24-b8061c1dc271 -boot order=d   \
        -drive file=/home/user/vyatta-kvm _ 4.0R1 _ amd64 _ vm2.img,if=none,id=drive-ide0-0-
1,format=raw \
        -device ide-hd,bus=ide.0,unit=0,drive=drive-ide0-0-1,id=ide0-0-1 \
        -net nic,model=e1000,macaddr=52:54:00:90:f2:a3 \
        -net tap,ifname=tap1,script=no,downscript=no,vhost=on \
        -vnc 192.168.1.142:3 \
        -device vfio-pci,host=81:00.0,id=hostdev0,bus=pci.0,addr=0x3 \
        -device vfio-pci,host=81:00.1,id=hostdev1,bus=pci.0,addr=0x4 \
        -device vfio-pci,host=83:00.0,id=hostdev2,bus=pci.0,addr=0x5 \
```

```
        -device vfio-pci,host=83:00.1,id=hostdev3,bus=pci.0,addr=0x6 \
        -device cirrus-vga,id=video0,bus=pci.0,addr=0x7 \
        -daemonize
```

3.1.4.4.2 One VM Running on Both CPU Sockets, Eight Interfaces

Modify `start _ vm.py` as indicated in 3.1.4.3.2

```
python start _ vm.py -name vyatta-2sockets -enable-kvm \
        -cpu host \
        -m 16384 \
        -object memory-backend-file,prealloc=yes,mem-path=/mnt/huge,size=8192M,id=ram-
node0,host-nodes=0,policy=bind \
        -object memory-backend-file,prealloc=yes,mem-path=/mnt/huge,size=8192M,id=ram-
node1,host-nodes=1,policy=bind \
        -numa node,nodeid=0,cpus=0-8,cpus=18-26,memdev=ram-node0 \
        -numa node,nodeid=1,cpus=9-17,cpus=27-35,memdev=ram-node1 \
        -device pxb,id=pxb0,bus=pci.0,bus _ nr=4,numa _ node=0\
        -device pxb,id=pxb1,bus=pci.0,bus _ nr=8,numa _ node=1\
        -uuid 8da5a07e-3b51-4be1-9f24-b8061c1dc271 -boot order=d   \
        -drive file=/home/user/vyatta-kvm _ 4.0R1 _ amd64 _ 2sockets.img,if=none,id=drive-ide0-0-
1,format=raw \
        -device ide-hd,bus=ide.0,unit=0,drive=drive-ide0-0-1,id=ide0-0-1 \
        -net nic,model=e1000,macaddr=52:54:00:90:f2:a2 \
        -net tap,ifname=tap0,script=no,downscript=no,vhost=on \
        -vnc 192.168.1.142:2 \
        -device vfio-pci,host=05:00.0,id=hostdev0,bus=pxb0,addr=0x1 \
        -device vfio-pci,host=05:00.1,id=hostdev1,bus=pxb0,addr=0x2 \
        -device vfio-pci,host=07:00.0,id=hostdev2,bus=pxb0,addr=0x3 \
        -device vfio-pci,host=07:00.1,id=hostdev3,bus=pxb0,addr=0x4 \
        -device vfio-pci,host=81:00.0,id=hostdev4,bus=pxb1,addr=0x1 \
        -device vfio-pci,host=81:00.1,id=hostdev5,bus=pxb1,addr=0x2 \
        -device vfio-pci,host=83:00.0,id=hostdev6,bus=pxb1,addr=0x3 \
        -device vfio-pci,host=83:00.1,id=hostdev7,bus=pxb1,addr=0x4 \
        -device cirrus-vga,id=video0,bus=pci.0,addr=0x5 \
        -daemonize
```

3.2 Brocade 5600 vRouter Configuration

3.2.1 Login and Password

Login=vyatta

Password=vyatta

3.2.2 Set Root Password

```
configure
set system login user root authentication plaintext-password 123456
commit
save
exit
```

3.2.3 Set Vyatta Management Interface IP address

```
set interfaces dataplane dp0s2 address 192.168.1.210/24
set system static-host-mapping host-name vyatta inet 192.168.1.210
```

3.2.4 Enable ssh Access + http

```
configure
set service ssh
set service ssh allow-root
set service http
commit
save
exit
```

3.2.5 Key Manipulation

Copy public key from prox management system (from which prox characterization script will be started) to `/home/vyatta/pk.pub`; then have vyatta properly load the keys. Note: copying them manually in .ssh will result in those changes being lost after reboot.

```
configure
loadkey root /home/vyatta/pk.pub
commit
save
exit
```

3.2.6 Set Dataplane IP Address

This is only one example. IP addresses and routes vary per use case.

```
set interfaces dataplane dp0s3 address 64.0.0.240/24
set interfaces dataplane dp0s4 address 65.0.0.240/24
set interfaces dataplane dp0s5 address 66.0.0.240/24
set interfaces dataplane dp0s6 address 67.0.0.240/24
```

3.2.7 Create Routes

```
set protocols static route 1.0.0.0/24 next-hop 64.0.0.1
set protocols static route 9.0.0.0/24 next-hop 65.0.0.1
set protocols static route 17.0.0.0/24 next-hop 66.0.0.1
set protocols static route 25.0.0.0/24 next-hop 67.0.0.1
```

3.2.8 Example Config File

The commands should result in a config file similar to this one. The configuration shown here uses only four interfaces and one route and next hop per interface.

```
interfaces {
        dataplane dp0s2 {
                address 192.168.1.210/24
        }
        dataplane dp0s3 {
                address 64.0.0.240/24
        }
        dataplane dp0s4 {
                address 65.0.0.240/24
        }
        dataplane dp0s5 {
                address 66.0.0.240/24
        }
        dataplane dp0s6 {
                address 67.0.0.240/24
        }
        loopback lo
}
protocols {
        static {
                route 1.0.0.0/24 {
                        next-hop 64.0.0.1
                }
                route 9.0.0.0/24 {
                        next-hop 65.0.0.1
                }
                route 17.0.0.0/24 {
                        next-hop 66.0.0.1
                }

                route 25.0.0.0/24 {
                        next-hop 67.0.0.1
                }
        }
}
```

```
service {
        https
        ssh
}
system {
        acm {
                enable
                operational-ruleset {
                        rule 9985 {
                                action allow
                                command /show/tech-support/brief/
                                group vyattaop
                        }
                        rule 9986 {
                                command /show/tech-support/brief
                                group vyattaop
                        }
                        rule 9987 {
                                command /show/tech-support
                                group vyattaop
                        }
                        rule 9988 {
                                command /show/configuration
                                group vyattaop
                        }
                        rule 9989 {
                                action allow
                                command "/clear/*"
                                group vyattaop
                        }
                        rule 9990 {
                                action allow
                                command "/show/*"
                                group vyattaop
                        }
                        rule 9991 {
                                action allow
                                command "/monitor/*"
                                group vyattaop
                        }
                        rule 9992 {
                                action allow
                                command "/ping/*"
                                group vyattaop
                        }
                        rule 9993 {
                                action allow
                                command "/reset/*"
                                group vyattaop
                        }
                        rule 9994 {
                                action allow
                                command "/release/*"
                                group vyattaop
                        }
                        rule 9995 {
                                action allow
                                command "/renew/*"
                                group vyattaop
                        }
                        rule 9996 {
                                action allow
                                command "/telnet/*"
                                group vyattaop
```

```
                        }
                        rule 9997 {
                                action allow
                                command "/traceroute/*"
                                group vyattaop
                        }
                        rule 9998 {
                                action allow
                                command "/update/*"
                                group vyattaop
                        }
                        rule 9999 {
                                command "*"
                                group vyattaop
                        }
                }
                ruleset {
                        rule 9999 {
                                action allow
                                group vyattacfg
                                operation "*"
                                path "*"
                        }
                }
        }
        config-management {
                commit-revisions 20
        }
        console {
                device ttyS0
        }
        domain-name mcplab.net
        host-name Vyatta-5600
        login {
                user root {
                        authentication {
                                encrypted-password $1$MGW0BV.5$vTG5Jtx6wPPxUGJmqalFH1
                                public-keys root@vRouter.mcplab.net {
                                        key
AAAAB3NzaC1yc2EAAAADAQABAAABAQC5Mp83HPbAKauvCejhYAINjSB/CFv/6mBFwg+DmshLORPKtfBM+zvBPdeOL1GXZ
2sXJxUIz1HYWJ0ePsnsM05DfbdVfDN8D6s5uUgIsLM4wXx0jj17wcynFE2FBHq0FWab4fnj2ZqOLY9Z4UA63TFvBTdUfAz
xP8M/sSQGsR2nWTZ2300yH9SK8v4khClAFlQRt4Qp06ltMjo6QzHAUH3z8BGRkoF0LpJ9mayAN0UZ6QcE24kZJUQMFLFrJi
DfxzUwWDSdL2U0s5HDkHXyFFlJ6x/gTFjee7hl8njlgIN7gG/uT9OWpdLI/jlUg3VgR4h8hOdsPBcGSbhMpAUo39P3
                                        type ssh-rsa
                                }
                        }
                }
                user vyatta {
                        authentication {
                                encrypted-password $1$MhxqymQQ$hpLZRkWd0hI3f5UQ0Z0ZO.
                                public-keys user@vRouter.mcplab.net {
                                        key
AAAAB3NzaC1yc2EAAAADAQABAAABAQC5Mp83HPbAKauvCejhYAINjSB/CFv/6mBFwg+DmshLORPKtfBM+zvBPdeOL1GXZ2
sXJxUIz1HYWJ0ePsnsM05DfbdVfD6s5uUgIsLM4wXx0jj17wcynFE2FBHq0FWab4fnj2ZqOLY9Z4UA63TFvBTdUfAzxP8M/sS
QGsR2nWTZ2300yH9SK8v4khClAFlQRt4Qp06ltMjo6QzHAUH3z8BGRkoF0LpJ9mayAN0UZ6QcE24kZJUQMFLFrJiDfxzUwWD
DSdL2U0s5HDkHXyFFlJ6x/gTFjee7hl8njlgIN7gG/uT9OWpdLI/jlUg3VgR4h8hOdsPBcGSbhMpAUo39P3
                                        type ssh-rsa
                                }
                        }
                        level admin
                }
        }
        static-host-mapping {
```

```
                host-name Vyatta-5600 {
                    inet 192.168.1.96
                }
        }
        syslog {
                global {
                    facility all {
                            level warning
                    }
                }
        }
}


/* Warning: Do not remove the following line. */
/* === vyatta-config-version: "config-management@1:dhcp-relay@2:pim@1:qos@2:routing@5:sflow@1:system
@13:twamp@1:vlan@1:vplane@2:vrrp@1:vrrp@2:webgui@1" === */
/* Release version: 3.5R5 */
```

3.2.9 Brocade 5600 vRouter Configuration per Use Case

The script "create_interfaces_and_routes.pl" provided by prox (in helper-scripts) can be used to show the routing tables for different numbers of routes and next hops. The routing table in the Brocade 5600 vRouter configuration files needs to be adapted accordingly. The following chapters show the Brocade 5600 vRouter configuration files for one route and one next hop per interface.

3.2.9.1 Two Independent VMs – One VM on Each CPU Socket, Four Interfaces per VM

3.2.9.1.1 VM1 Configuration

The configuration of the interfaces and routes of the first VM, with four routes and four next hops:

```
interfaces {
        dataplane dp0s2 {
                address 192.168.1.210/24
        }
        dataplane dp0s3 {
                address 64.0.0.240/24
        }
        dataplane dp0s4 {
                address 65.0.0.240/24
        }
        dataplane dp0s5 {
                address 66.0.0.240/24
        }
        dataplane dp0s6 {
                address 67.0.0.240/24
        }
        loopback lo
}
protocols {
        static {
                route 1.0.0.0/24 {
                        next-hop 64.0.0.1
                }
                route 9.0.0.0/24 {
                        next-hop 65.0.0.1
                }
                route 17.0.0.0/24 {
                        next-hop 66.0.0.1
                }
                route 25.0.0.0/24 {
                        next-hop 67.0.0.1
                }
        }
}
```

3.2.9.1.2 VM2 Configuration

The configuration of the interfaces and routes of the second VM, with four routes and four next hops.

```
interfaces {
        dataplane dp0s2 {
                address 192.168.1.211/24
        }
        dataplane dp0s3 {
                address 68.0.0.240/24
        }
        dataplane dp0s4 {
                address 69.0.0.240/24
        }
        dataplane dp0s5 {
                address 70.0.0.240/24
        }
        dataplane dp0s6 {
                address 71.0.0.240/24
        }
        loopback lo
}
protocols {
        static {
                route 33.0.0.0/24 {
                        next-hop 68.0.0.1
                }
                route 41.0.0.0/24 {
                        next-hop 69.0.0.1
                }
                route 49.0.0.0/24 {
                        next-hop 70.0.0.1
                }
                route 57.0.0.0/24 {
                        next-hop 71.0.0.1
                }
        }
}
```

3.2.9.2 One VM Running on Both CPU Sockets, Eight interfaces

3.2.9.2.1 VM Configuration

The configuration of the interfaces and routes of the VM, with eight routes and eight next hops

```
interfaces {
        dataplane dp0s2 {
                address 192.168.1.210/24
        }
        dataplane dp0s3 {
                address 64.0.0.240/24
        }
        dataplane dp0s4 {
                address 65.0.0.240/24
        }
        dataplane dp0s5 {
                address 66.0.0.240/24
        }
        dataplane dp0s6 {
                address 67.0.0.240/24
        }
        dataplane dp0s7 {
                address 68.0.0.240/24
        }
```

```
            dataplane dp0s8 {
                    address 69.0.0.240/24
            }
            dataplane dp0s9 {
                    address 70.0.0.240/24
            }
            dataplane dp0s10 {
                    address 71.0.0.240/24
            }

            loopback lo
    }
    protocols {
            static {
                    route 1.0.0.0/24 {
                            next-hop 64.0.0.1
                    }
                    route 9.0.0.0/24 {
                            next-hop 65.0.0.1
                    }
                    route 17.0.0.0/24 {
                            next-hop 66.0.0.1
                    }
                    route 25.0.0.0/24 {
                            next-hop 67.0.0.1
                    }
                    route 33.0.0.0/24 {
                            next-hop 68.0.0.1
                    }
                    route 41.0.0.0/24 {
                            next-hop 69.0.0.1
                    }
                    route 49.0.0.0/24 {
                            next-hop 70.0.0.1
                    }
                    route 57.0.0.0/24 {
                            next-hop 71.0.0.1
                    }
            }
    }
```

4 Test Generators' Configuration

4.1 Hardware and Software Details

ITEM	DESCRIPTION	NOTES
Platform	Intel® Server Board S2600WT Family	
Form factor	2U Rack Mountable	
Processor(s)	2x Intel® Xeon® CPU E5-2699 v3	46080KB L3 Cache per CPU, 18 cores per CPU (36 logical cores due to Hyper-threading).
Memory	32 GB RAM (8x 4 GB) per socket	Quad channel 2134 DDR4
BIOS	SE5C610.86B. 01. 01 0009.060120151350	Hyper-threading enabled Hardware prefetching enabled COD disabled
Host OS	CentOS 7.1	Kernel version: 3.10.0-229.7.2.el7.x86_64
Hugepages	8x 1 GB	
PROX	0.31	http://github.com/nvf-crucio/prox
DPDK	2.2.0	
NICs	8x Intel® 82599 10 Gigabit Ethernet Controller	

4.1.1 Grub.cfg

DPDK-related CPU cores must be isolated through isolcpus in grub.cfg, by making sure interrupts are handled by core 0.

```
linux16 /vmlinuz-3.10.0-229.11.1.el7.x86_64
root=/dev/mapper/centos-root ro rd.lvm.
lv=centos/swap vconsole.keymap=us ipv6.
disable=1 crashkernel=auto  vconsole.
font=latarcyrheb-sun16 rd.lvm.lv=centos/
root selinux=0 rhgb quiet LANG=en_US.UTF-8
noirqbalance intel_pstate=disable intel_
idle.max_cstate=0 processor.max_cstate=0
default_hugepagesz=1G hugepagesz=1G
hugepages=16 transparent_hugepage=never
isolcpus=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,
51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,
68,69,70,71
```

4.1.2 Scripts to Prevent Interrupts on DPDK Fast Path

Same script as in 3.1.4.1 is run to decrease interrupts.

4.2 Test Setup Details

The Brocade 5600 vRouter is characterized using test generators: the test generators generate IP traffic towards four or eight 10 Gbps interfaces and measure the traffic coming from those interfaces. Those test generators can be Ixia (or Spirent), or COTS servers running DPDK-based applications (pktgen or prox).

For automation purposes, prox (https://01.org/intel-data-plane-performance-demonstrators/prox-overview) has been used to generate the traffic and to measure the throughput and latency from the Brocade 5600 vRouter. Ixia has been used as well to confirm some key data results.

Prox contains configuration files to be used for four- and eight-interface vRouters. It also contains a set of characterization scripts.

4.3 Test Parameters

The characterization studies the influence of the following parameters on the throughput and latency:

- Packet Size

Other parameters such as number of next hops and number of routes have been fixed respectively to 16 next hops and 1024 routes per interface.

The test system measures 0 packet loss, for a fixed number of routes and next hops.

4.3.1 Traffic Profiles

For each traffic profile, traffic is received on all interfaces and sent towards some interfaces following a predefined pattern.

- In the first traffic profile (Figure 3), traffic from interface 1 is sent to interface 2 and from interface 2 towards interface 1; from interface 3 towards 4 and from 4 towards 3.
- In the second traffic profile (Figure 4), traffic is also sent towards only one outgoing interface for each incoming interface, but the outgoing interface is always on the other socket compared to the incoming interface.

A traffic profile test is obtained by configuring properly the destination IP addresses of the packets being generated. Routes and next hops configurations are the same for all traffic profiles.

4.4 Characterization Scripts

The characterization is performed in two phases. First, a configuration file listing the tests to execute is created. Then the tests are performed based on the configuration file.

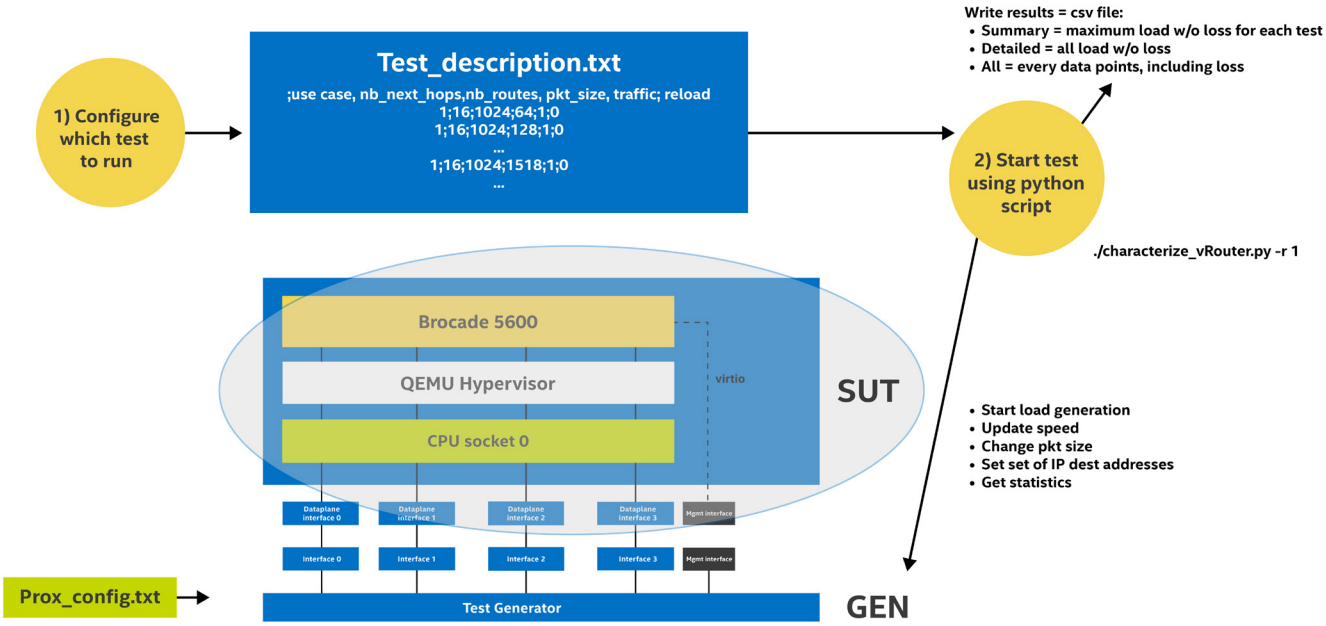


Figure 9. Characterization phases

The characterization configuration file (`test _ description.txt`) has the following format:

`Use case; next _ hops; routes;pkt _ size; traffic;reload`

The prox configuration file must be adapted to the system under test. The proper destination MAC addresses must be inserted in the packets generated. Brocade 5600 vRouter is (at least by default) not supporting promiscuous mode. Hence, packets sent with wrong MAC addresses will silently be deleted by the vRouter interfaces.

The characterization scripts have been written to support one VM. The script supports only simple characterization in the two VM case (e.g., fixed number of routes and next hops). For instance, the characterization script is unable to reload new configurations on two VMs. To run some characterization when the system under test is using two VMs, the vRouters must be manually configured so that they appear from the outside as one VM with eight cores (see 3.2.9.1). The `test _ description.txt` must be configured so that the configuration file is not reloaded (reload=0).

```
1; 16; 1024; 64; 0; 0
1; 16; 1024; 128; 0; 0
1; 16; 1024; 256; 0; 0
1; 16; 1024; 512; 0; 0
1; 16; 1024; 1024; 0; 0
1; 16; 1024; 1280; 0; 0
1; 16; 1024; 1518; 0; 0
1; 16; 1024; 64; 1; 0
1; 16; 1024; 128; 1; 0
1; 16; 1024; 256; 1; 0
1; 16; 1024; 512; 1; 0
1; 16; 1024; 1024; 1; 0
1; 16; 1024; 1280; 1; 0
1; 16; 1024; 1518; 1; 0
```

It's expected that the first four ports on the test generators are connected to the first four ports on the SUT. Failing to do so will result in bad performance.

5 Running the Characterization

When using prox 0.31 with DPDK 2.2.0, the `characterize _ vRouter` script has an issue related to the inclusion of ierrors in statistics. This requires a change in the script: the line rx+= ierrors should be commented out.

When the vRouter is properly configured, its configuration files copied in `/config/prox` and test_description.txt file created, the characterization can start. Run

```
./characterize _ vRouter.py –r 1
```

The characterization will create up to three result-related files:

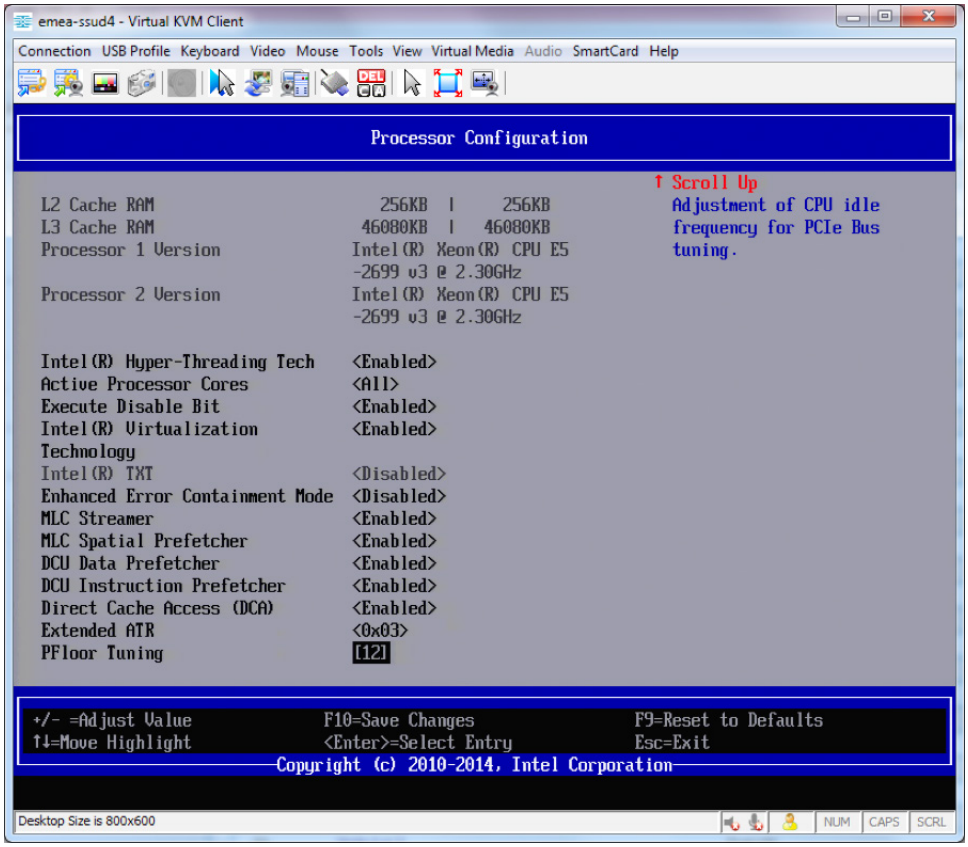
- `minimal _ results.txt` contains the results to be plotted.
- `detailed _ results.txt` contains all succeeding steps used in the binary search for 0% packet loss; it is used for debugging.
- `all _ results.txt` contains all data points. Only useful for debugging (e.g., looking at how many packets were lost and why higher throughput was not obtained).

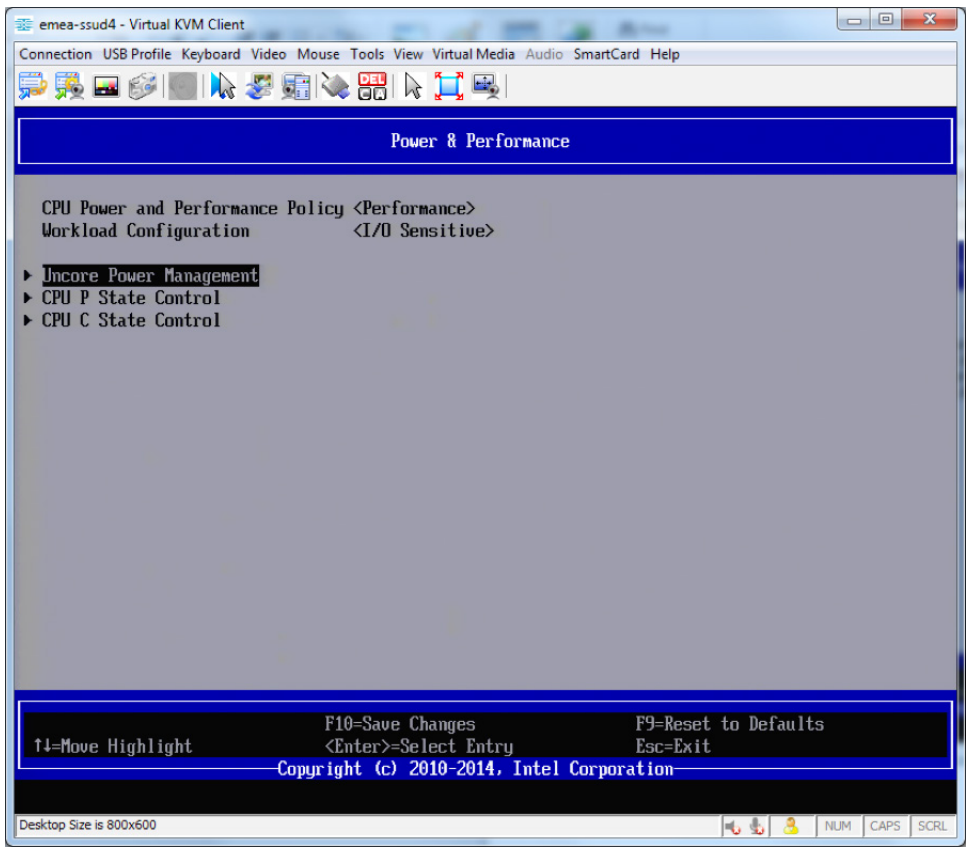
Those files contain throughput and latency results. They can be plotted using Microsoft Excel.*

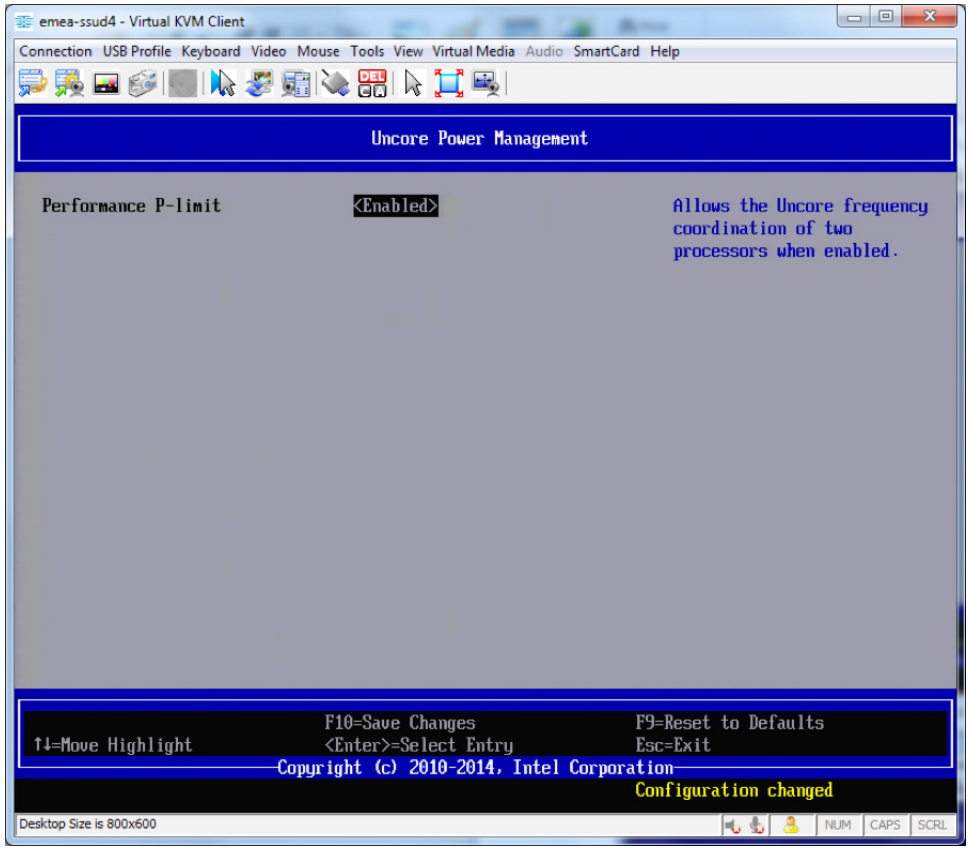
6 BIOS Settings

The Brocade 5600 vRouter system under test configuration requires some specific BIOS settings.

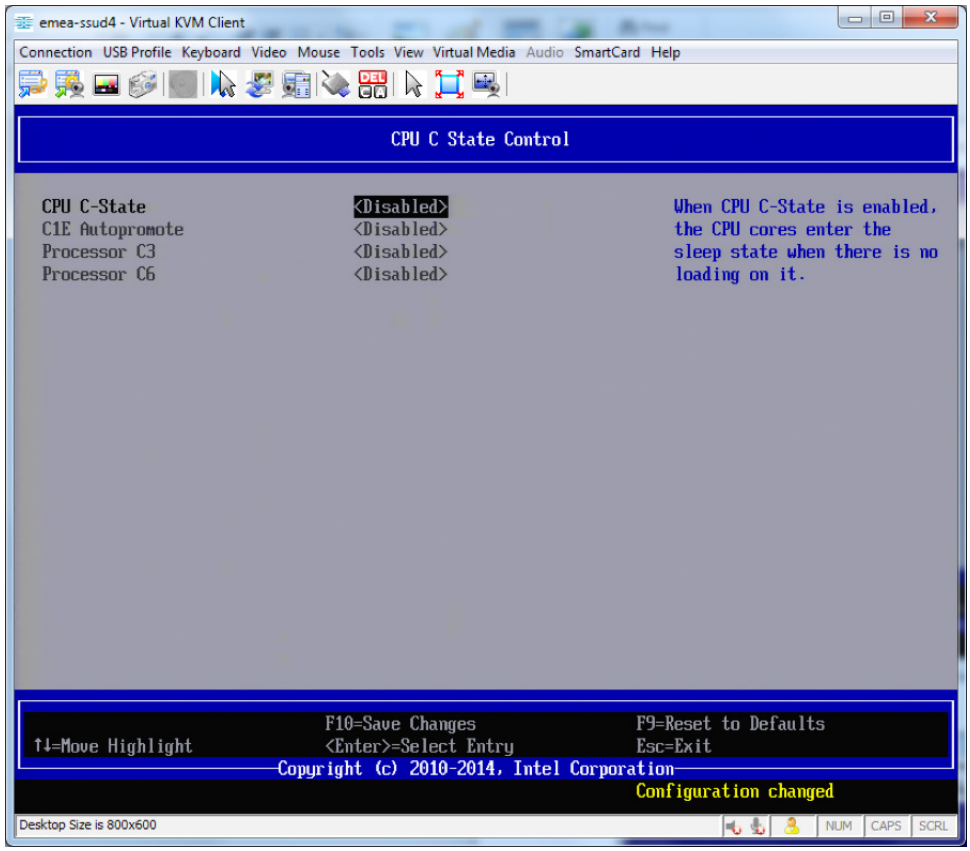
The following screens show the difference compared to “default BIOS settings”.

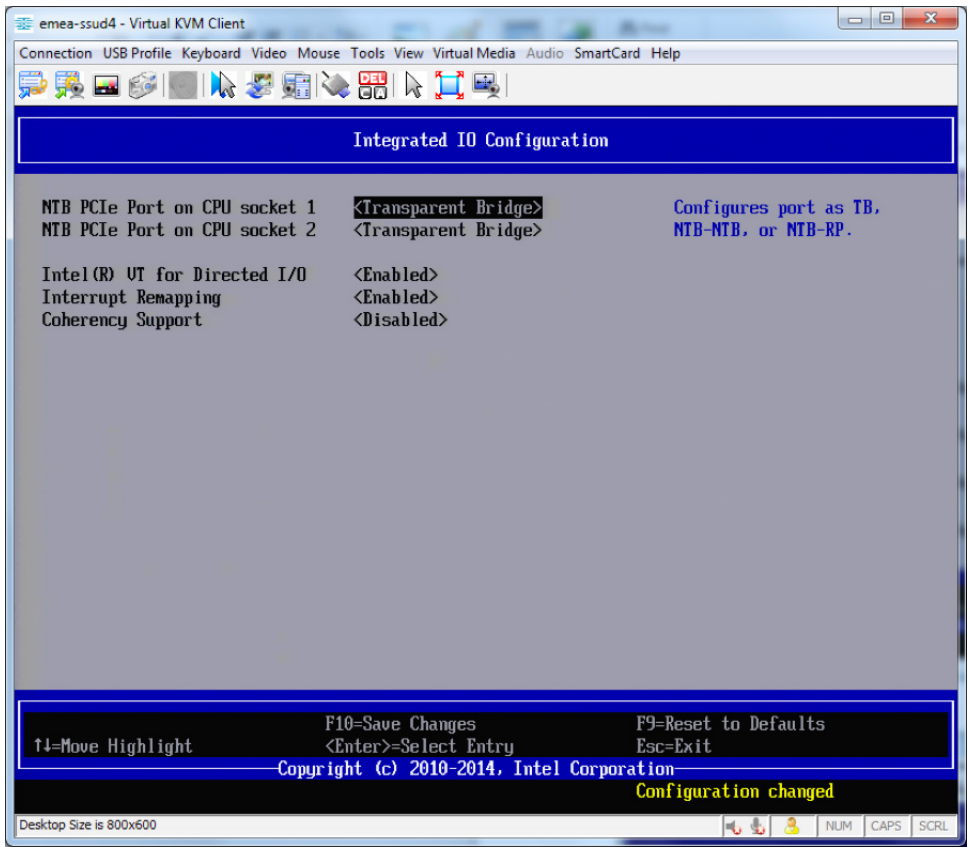





emea-ssud4 - Virtual KVM Client
Connection USB Profile Keyboard Video Mouse Tools View Virtual Media Audio SmartCard Help
Uncore Power Management
Performance P-limit
<Enabled>
Allows the Uncore frequency coordination of two processors when enabled.
F10=Save Changes
F9=Reset to Defaults
↑↓=Move Highlight
<Enter>=Select Entry
Esc=Exit
Copyright (c) 2010-2014, Intel Corporation
Configuration changed
Desktop Size is 800x600
NUM
CAPS
SCRL


emea-ssud4 - Virtual KVM Client
Connection USB Profile Keyboard Video Mouse Tools View Virtual Media Audio SmartCard Help
CPU P State Control
Enhanced Intel SpeedStep(R) Tech
<Disabled>
Enhanced Intel SpeedStep (R) Technology allows the system to dynamically adjust processor voltage and core frequency, which can result in decreased average power consumption and decreased average heat production.
Contact your OS vendor regarding OS support of this feature.
F10=Save Changes
F9=Reset to Defaults
↑↓=Move Highlight
<Enter>=Select Entry
Esc=Exit
Copyright (c) 2010-2014, Intel Corporation
Configuration changed
Desktop Size is 800x600
NUM
CAPS
SCRL


emea-ssud4 - Virtual KVM Client
Connection USB Profile Keyboard Video Mouse Tools View Virtual Media Audio SmartCard Help
CPU C State Control
CPU C-State <Disabled>
C1E Autopromote <Disabled>
Processor C3 <Disabled>
Processor C6 <Disabled>
When CPU C-State is enabled, the CPU cores enter the sleep state when there is no loading on it.
F10=Save Changes
F9=Reset to Defaults
↑↓=Move Highlight
<Enter>=Select Entry
Esc=Exit
Copyright (c) 2010-2014, Intel Corporation
Configuration changed
Desktop Size is 800x600
NUM CAPS SCRL


emea-ssud4 - Virtual KVM Client
Connection USB Profile Keyboard Video Mouse Tools View Virtual Media Audio SmartCard Help
Integrated IO Configuration
NTB PCIe Port on CPU socket 1 <Transparent Bridge>
NTB PCIe Port on CPU socket 2 <Transparent Bridge>
Configures port as TB, NTB-NTB, or NTB-RP.
Intel(R) VT for Directed I/O <Enabled>
Interrupt Remapping <Enabled>
Coherency Support <Disabled>
F10=Save Changes
F9=Reset to Defaults
↑↓=Move Highlight
<Enter>=Select Entry
Esc=Exit
Copyright (c) 2010-2014, Intel Corporation
Configuration changed
Desktop Size is 800x600
NUM CAPS SCRL

7 References

PROX https://01.org/intel-data-plane-performance-demonstrators/overview and http://github.com/nvf-crucio/prox

Brocade 5600 vRouter http://www.brocade.com/en/products-services/software-networking/network-functions-virtualization/vrouter.html

Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families: Go to http://www.intel.com/products/processor_number.

© 2017 Intel Corporation. Intel, the Intel logo, and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others. ♻ Please Recycle 0217/DO/PDF 335649-001US