# 1. Tuning parameters

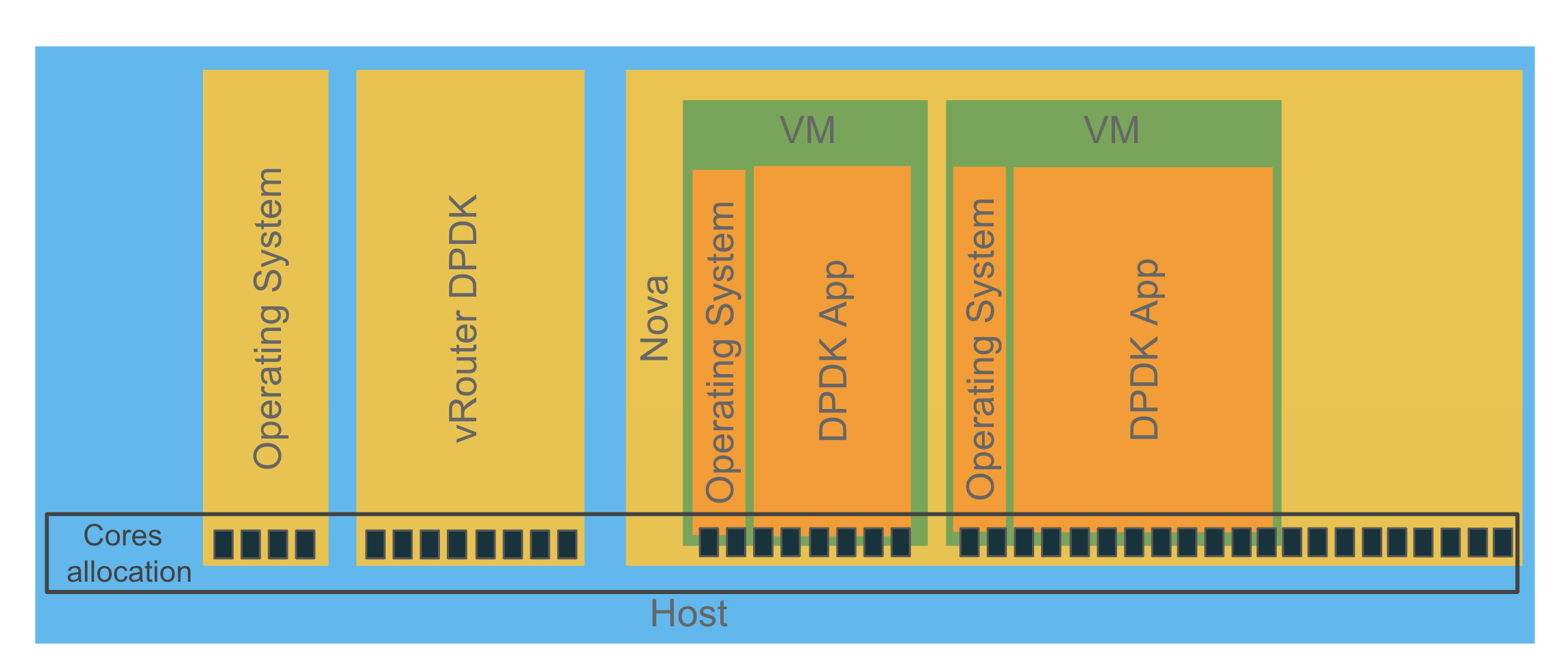
Tuning of performance of DPDK can be complex and requires an end to end solution otherwise it will not be efficient. Tuning only Operating System and vRouter still cannot provide expected performance if packet drops will occur on VNF or opposite efficient VNF the performance can be affected by wrong OS configuration.

To have full picture of tuning VNF will be included in consideration as well. in the example a VNF application will be shown as Centos VM with DPDK application Prox (traffic generator).

## 1.1 Cores allocation

The vRouter DPDK processing cores need to be pinned to given cores and Nova VMs need to pin cores for VMs.

On the picture below you can see an example of cores allocation on the host.



As presented, all available the server cores need to be planned and allocated explicitly for a given role. Remaining not allocated will be used by the Operating System to run regular processes. The same is for the VMs OS. Systemd is responsible for OS processes execution where affinity lists can narrow the list of cores to be used. The affinity list for systemd is a part of tuned profiles described later.

Pinning vRouter DPDK cores is mandatory as well VNF VM DPDK cores.

### 

### 

### 

### Host cores allocation

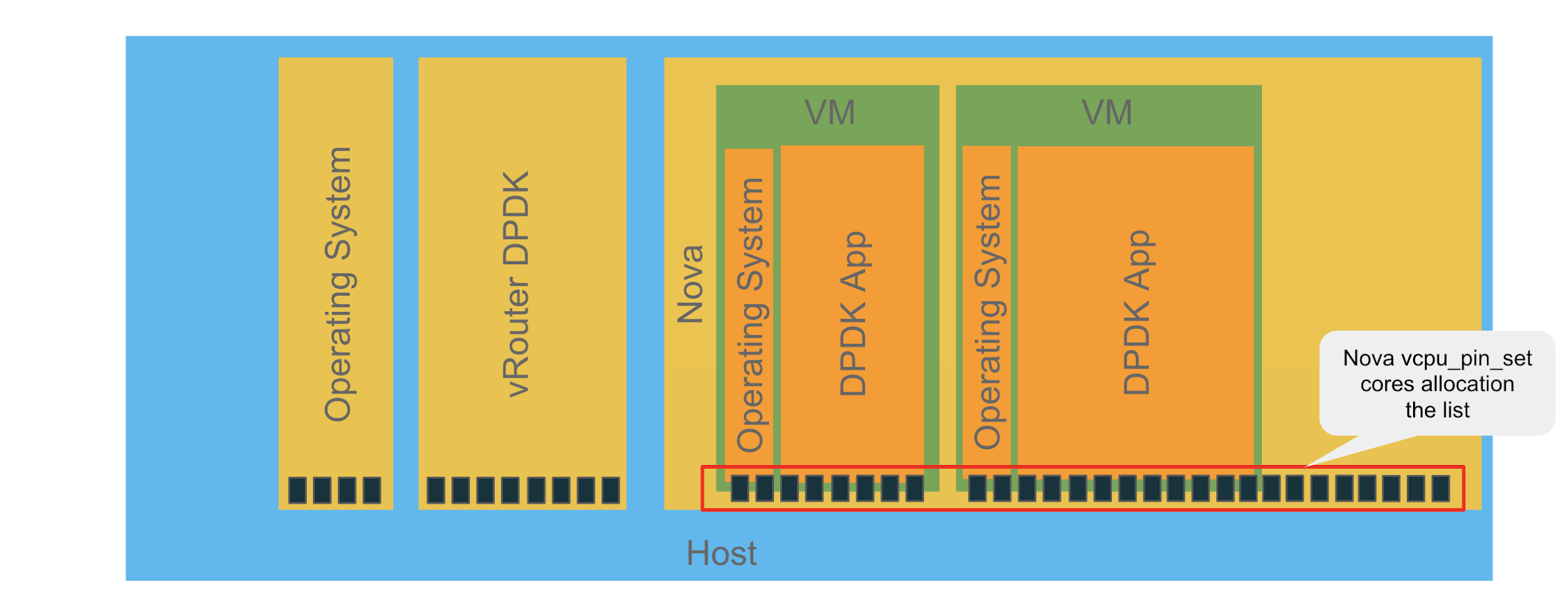
Before implementing tuning memory and CPU allocation a plan needs to be done. General CPU allocation the plan like following:

* Number of CPUS for Operating System
* Number of cores for Nova
* Number of cores for vRouter

#### 1.1.1.1 Host

Host cores (Operating System/OS) not need to be provided explicitly they are remaining cores not allocated for vRouter and Nova. Good practice is to allocate first core with his sibling from each NUMA for OS.

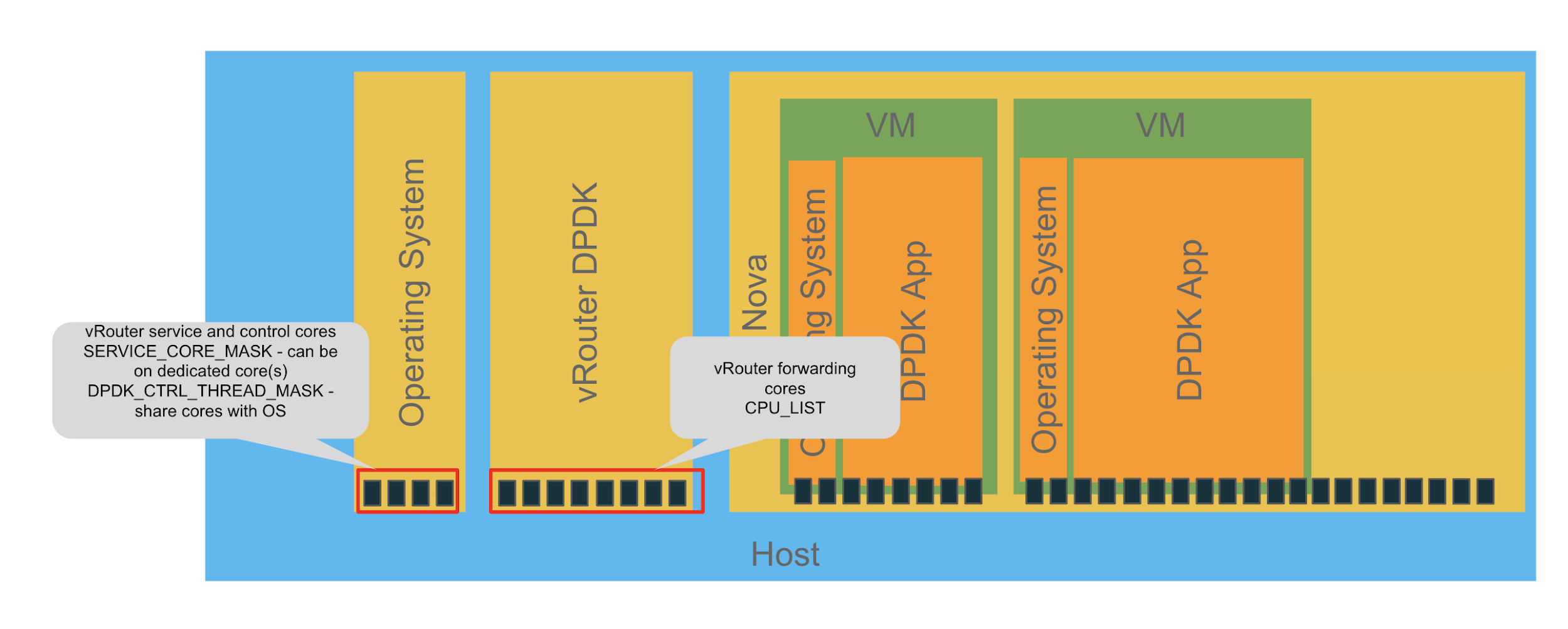
#### 1.1.1.2 Nova



Provide for Nova the list of cores that can be used for created VMs.

|  |
| --- |
| # grep ^vcpu /var/lib/config-data/puppet-generated/nova\_libvirt/etc/nova/nova.conf  vcpu\_pin\_set=**<list of cores for VMs>** |

#### 1.1.1.3 vRouter



|  |
| --- |
| # cat /etc/sysconfig/network-scripts/ifcfg-vhost0  [...]  CPU\_LIST=**<list of forwarding cores>**  SERVICE\_CORE\_MASK= **<list of cores for service threads>**  DPDK\_CTRL\_THREAD\_MASK= **<list of cores for control threads>** |

Where:

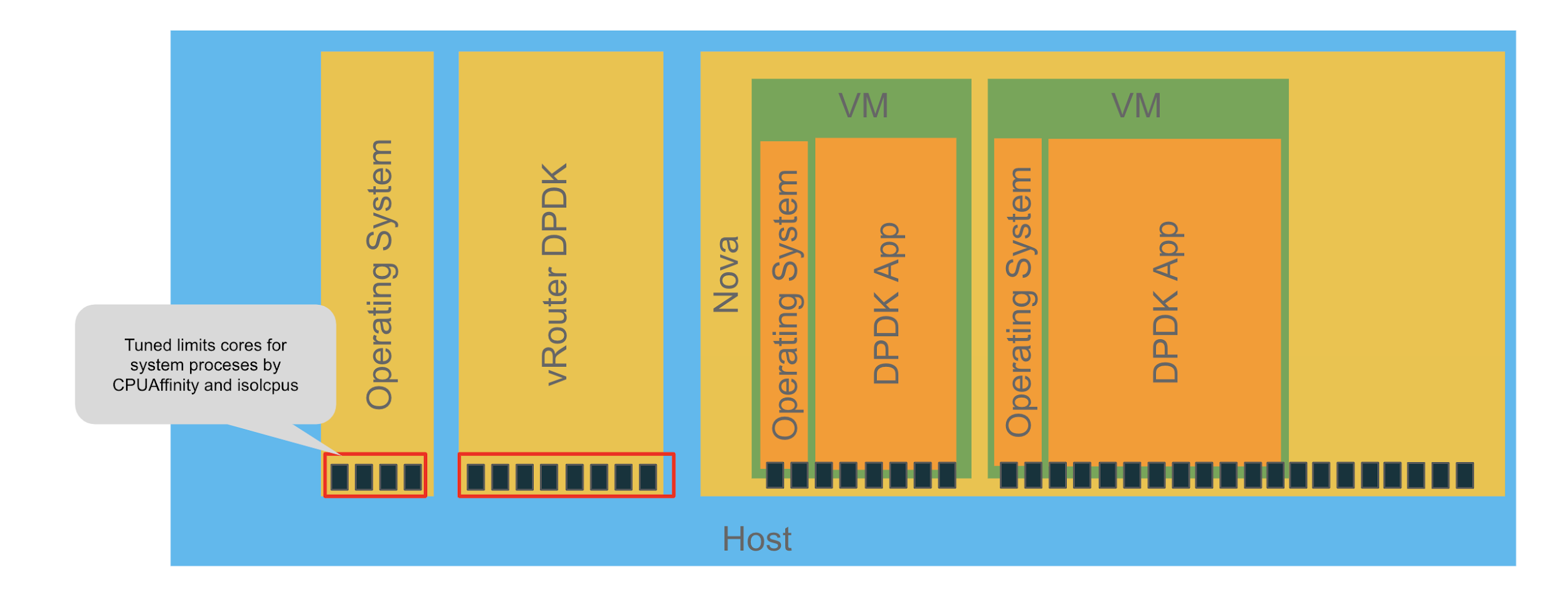
* **DPDK\_CTRL\_THREAD\_MASK** - control threads used for DPDK internal processing. eal-intr-thread, rte\_mp\_handle, rte\_mp\_async (they are generated by the DPDK library itself - DPDK setup management) DPDK control threads will be allocated onto Operating System assigned CPUs
* **SERVICE\_CORE\_MASK** - service threads used for connectivity between vrouter agent and vrouter forwarding plane (DPDK vrouter). Thread names are lcore 0 to 9 thread names are lcore 0 to 9. They each one has a specific role:
  + lcore 0: vhost0
  + lcore 1: timers
  + lcore 2: Interrupts
  + lcore 8: pkt0 (can be busy)
  + lcore 9: Netlink (can be busy)

Service threads can be allocated to Operating System cores where no intensive Flow Mode usage other ways 1 pcore with sibling should be dedicated for threads.

* **CPU\_LIST** - processing threads used for packet polling and processing (forwarding plane), thread names are lcore 10 and above

#### 1.1.1.4 Operating System (OS)

There is no need to allocate/pin OS processes, by default non pinned cores will be used. Tuned profile using systemd configuration by CPU affinity list will prevent OS threads to be run on cores reserved for Nova and vRouter.



|  |
| --- |
| # grep ^CPUAffinity /etc/systemd/system.conf  CPUAffinity= **< list of OS cores>** |

### Not listed icolspus cores will be used by operating system as described in the host section.

### 1.1.2 VNF (Guest VM) cores allocation

#### 1.1.2.1 Prox - VNF application the example

|  |
| --- |
| #cat swap.cfg  **[core 0]**  mode=master  **[core 2,3,4,5,6,7]**  name=swap  task=0  mode=swap  sub mode=l3  rx port=if0  tx port=if0  local ipv4=${local\_ip}  drop=no  #tail -f prox.log  […]  **Core 2: RX port 0 (queue 0) ==> TX port 0 (queue 0)**  **Core 3: RX port 0 (queue 1) ==> TX port 0 (queue 1)**  **Core 4: RX port 0 (queue 2) ==> TX port 0 (queue 2)**  **Core 5: RX port 0 (queue 3) ==> TX port 0 (queue 3)**  **Core 6: RX port 0 (queue 4) ==> TX port 0 (queue 4)**  **Core 7: RX port 0 (queue 5) ==> TX port 0 (queue 5)**  #ps -Tp 3193  PID SPID TTY TIME CMD  3193 3193 ? 00:04:57 prox  3193 3194 ? 00:00:00 eal-intr-thread  3193 3195 ? 00:00:00 rte\_mp\_handle  **3193 3196 ? 00:04:57 lcore-slave-2**  **3193 3197 ? 00:04:57 lcore-slave-3**  **3193 3198 ? 00:04:57 lcore-slave-4**  **3193 3199 ? 00:04:57 lcore-slave-5**  **3193 3200 ? 00:04:57 lcore-slave-6**  **3193 3201 ? 00:04:57 lcore-slave-7** |

#### 1.1.2.2 Operating System (OS)

There is no need to allocate/pin OS processes, by default non pinned cores will be used. Tuned profile using systemd affinity prevents run OS threads on cores reserved for prox.

### 1.1.3 Isolation

Once we have pinned all cores on a host the same tuning must be done inside of the VM, where we have an operating system a lot of processes and DPDK applications which need to be protected against other processes.

Definitions of isolation configuration statements:

**Kernel isolcpus** - Isolate CPUs from the kernel scheduler. This option removes a set of CPUs from the general kernel SMP balancing and scheduler algorithms. So, the only way to move a process onto or off an "isolated" CPU is via the CPU affinity syscalls. This is a kernel boot parameter specified in grub.

**Tuned isolated\_cores** - This is the recommended way to do “isolcpus” in RHEL. Uses the “tuned” daemon in its tuned-profiles. Change the “isolated\_cores” variable in /etc/tuned/cpu-partitioning-variables.conf to make systemd not to spawn any processes in these cores.

#### 1.1.3.1 Host

***Isolcpu*** and tuned configurations for the VM, where kernel isolcpus parameters define isolation from kernel scheduler perspective and need to be defined in grub.

Tuned ***isolated\_cores*** must provide the same list of “isolated” cores plus “**cpu-partitioning**” profile applied.

|  |
| --- |
| Kernel isolcpu  # cat /proc/cmdline  BOOT\_IMAGE=/boot/vmlinuz-3.10.0-1062.12.1.el7.x86\_64 root=UUID=228c59ea-82f0-4ee8-9d03-5620e5f0fafb ro console=tty0 console=ttyS0,115200n8 crashkernel=auto rhgb quiet **isolcpus=2-27,30-55** iommu=pt intel\_iommu=on default\_hugepagesz=1GB hugepagesz=1G hugepages=128 hugepagesz=2M hugepages=8192skew\_tick=1 nohz=on nohz\_full=2-27,30-55 rcu\_nocbs=2-27,30-55 tuned.non\_isolcpus=30000003 intel\_pstate=disable nosoftlockup  Tuned cpu-partitioning profile configuration  grep ^isolated\_cores /etc/tuned/cpu-partitioning-variables.conf  isolated\_cores=**2-27,30-55** |

#### 1.1.3.2 VM

***Isolcpu*** and tuned configurations for the VM, where kernel isolcpus parameters define isolation from kernel scheduler perspective and need to be defined in grub.

Tuned ***isolated\_cores*** must provide the same list of “isolated” cores plus “**realtime-virtual-guest**” profile applied.

|  |
| --- |
| Kernel isolcpu  # cat /proc/cmdline  BOOT\_IMAGE=/boot/vmlinuz-3.10.0-1062.12.1.el7.x86\_64 root=UUID=3ef2b806-efd7-4eef-aaa2-2584909365ff ro console=tty0 console=ttyS0,115200n8 crashkernel=auto console=ttyS0,115200 LANG=en\_US.UTF-8 **isolcpus=1-9** intel\_pstate=disable nosoftlockup skew\_tick=1 nohz=on nohz\_full=1-9 rcu\_nocbs=1-9  Tuned realtime-virtual-guest-variables profile configuration  grep ^isolated\_cores /etc/tuned/realtime-virtual-guest-variables.conf  isolated\_cores=1-9 |

### 1.1.4 Hugepages

Allocating huge pages should be done based on the prediction of how many VMs (their amount of memory) will be used on a hypervisor + 2GB for vRouter for default configuration. For setups with larger number of: lcores and users flows/routes, it is recommended to increase the size to 4G or 8G.

#### Host hugepages configuration

|  |
| --- |
| Kernel huge pages  # cat /proc/cmdline  BOOT\_IMAGE=/boot/vmlinuz-3.10.0-1062.12.1.el7.x86\_64 root=UUID=228c59ea-82f0-4ee8-9d03-5620e5f0fafb ro console=tty0 console=ttyS0,115200n8 crashkernel=auto rhgb quiet isolcpus=2-27,30-55 iommu=pt intel\_iommu=on **default\_hugepagesz=1GB hugepagesz=1G hugepages=128 hugepagesz=2M hugepages=8192** skew\_tick=1 nohz=on nohz\_full=2-27,30-55 rcu\_nocbs=2-27,30-55 tuned.non\_isolcpus=30000003 intel\_pstate=disable nosoftlockup |

Where:

* *Default\_hugepagesz* - defines which huge page size is a default (this size will appear in /proc/meminfo, and this size will be mounted by default when pagesize mounting option will not be used)
* *Hugepagesz* - followed by hugepages defines size and amount respectively and the pair can be repeated to configure different sizes of huge pages.

#### vRouter hugepages allocation configuration

Socket memory is pre-allocated hugepages for vRouter. In the case below, 1GB of huge-page memory is pre-allocated per NUMA.

|  |
| --- |
| #ps auxww| grep no-daemon  root 157 806 0.0 424231896 11556 ? SLl May25 136007:11 /usr/bin/contrail-vrouter-dpdk --no-daemon --vr\_flow\_entries 2500000 --yield\_option 0 --vr\_dpdk\_rx\_ring\_sz 4096 --vr\_dpdk\_tx\_ring\_sz 4096 --vr\_mempool\_sz 1048568 --service\_core\_mask (0,1,28,29) --dpdk\_ctrl\_thread\_mask (0,1,28,29) **--socket-mem 1024 1024** --vdev eth\_bond\_bond1,mode=4, xmit\_policy=l34, socket\_id=0, mac=e4:43:4b:6e:95:40, lacp\_rate=1,slave=0000:19:00.0,slave=0000:19:00.1 |

Following formula has to be used to define the socket memory size to be configured on vrouter:

* if vr\_mempool\_sz size < = 131072 then use socket\_mem = 1024
* if vr\_mempool\_sz size > 131072 then use socket\_mem = 2048

#### VM (VNF) hugepages allocation configuration

|  |
| --- |
| openstack flavor set --property hw:mem\_page\_size='large' **<flavor name>** |

### 1.1.5 VM (VNF) CPU pinning policies

By default, instance vCPU processes are not assigned to any particular host CPU, instead, they float across host CPUs like any other process. This allows for features like overcommitting of CPUs. In heavily contended systems, this provides optimal system performance at the expense of performance and latency for individual instances.

Some workloads require real-time or near real-time behavior, which is not possible with the latency introduced by the default CPU policy. For such workloads, it is beneficial to control which host CPUs are bound to an instance's vCPUs. This process is known as pinning. No instance with pinned CPUs can use the CPUs of another pinned instance, thus preventing resource contention between instances. To configure a flavor to use pinned vCPUs, a use a dedicated CPU policy. To force this, run:

|  |
| --- |
| # openstack flavor set m1.large --property hw:cpu\_policy=dedicated **<flavor name>** |

To avoid conflicts with siblings usage by others a CPU thread policy should be specified, one of two is recommended:

For workloads where sharing benefits performance, use thread siblings. To force this, run:

|  |
| --- |
| # openstack flavor set m1.large \  --property hw:cpu\_policy=dedicated \  --property hw:cpu\_thread\_policy=require **<flavor name>** |

For other workloads where performance is impacted by contention for resources, use non-thread siblings or non-SMT hosts. To force this, run:

|  |
| --- |
| # openstack flavor set m1.large \  --property hw:cpu\_policy=dedicated \  --property hw:cpu\_thread\_policy=isolate **<flavor name>** |

### 1.1.5 Multiqueue

With virtio-net multiqueue enabled network performance can be scaled across a number of vCPUs, by transferring the packets through multiple virtqueue pairs at a time.

When multiqueue is used on a Guest VM, In the ideal scenario, each multiqueue vNIC is configured with the same number of queue than the number of CPU allocated to vrouter for packet processing, vRouter can allocate queues no more than number available lcores. Current number of supported lcores by vRouter is 16.

|  |
| --- |
| openstack image set --property hw\_vif\_multiqueue\_enabled="true" **<Image name>** |

### 1.1.6 Rings buffer descriptors adjustment

Buffer descriptors are some kind of buffering mechanism in vRouter, increasing size is helping increase maximum packet per second. It needs to be set carefully because too big buffer size can increase latency.

#### 1.1.6.1 VR descriptors

By default vRouter is balancing traffic pooled from VMs queues across all cores to ensure that all cores are utilised. To help in case of not multqueue VNFs load distribution on vRouter cores or if VNF balance algorithm among queues is not ideal. For rebalance and traffic vRouter calculate RSS hash and base on this send buffer descriptors information to others. For exchanging inter-core information is using rings. A ring is a circular array of descriptors allocated by the OS in the system memory (RAM). Each descriptor contains information about a packet that has been received or that is going to be transmitted. Default values are: RX and TX = 1024.

Following parameters need to be defined:

*--vr\_dpdk\_txd\_sz* : is used to define inter coreTX Ring descriptor size.

*--vr\_dpdk\_rxd\_sz* : is used to define inter core RX Ring descriptor size.

Following formula has to be used to define the mempool size (set of mbufs) to be configured on vrouter :

**vr\_mempool\_sz** = 2 \* (vr\_dpdk\_rxd\_sz + vrdpdk\_txd\_sz) \* number\_of\_vrouter\_lcores \* number\_of\_ports\_in\_dpdk\_bond

\*Once NIC descriptors are configured as well, vr\_mempool\_sz needs to have a total of required memory size for both.

|  |
| --- |
| #etc/sysconfig/network-scripts/network-functions-vrouter-dpdk-env| grep DPDK\_COMMAND\_ADDITIONAL\_ARGS  DPDK\_COMMAND\_ADDITIONAL\_ARGS="${DPDK\_COMMAND\_ADDITIONAL\_ARGS:---vr\_flow\_entries=2000000 **--vr\_mempool\_sz 524288 --vr\_dpdk\_rx\_ring\_sz 4096 --vr\_dpdk\_tx\_ring\_sz 4096** --yield\_option 0 }" |

#### 1.1.6.2 NIC descriptors

When using Fortville, packets are by default dropped when RX descriptors are not available. Hence, by default, the only buffers available to store packets when a core is interrupted are the buffers pointed by the RX descriptors. Good practice is to increase descriptors size to 2048 or 4096. The number of TX descriptors should also be considered and depends on the application. Default values are: RX and TX = 128 and mempool size 16384 Bytes. Increasing buffer descriptors size to high value (>= 8k) can drastically increase latency!

Following parameters need to be defined:

*--dpdk\_txd\_sz* : is used to define Physical NIC TX Ring descriptor size.

*--dpdk\_rxd\_sz* : is used to define Physical NIC RX Ring descriptor size.

*--vr\_mempool\_sz* : is used to define mempool memory size.

Following formula has to be used to define the mempool size (set of mbufs) to be configured on vrouter:

***vr\_mempool\_sz*** *= 2 \* (dpdk\_rxd\_sz + dpdk\_txd\_sz) \* number\_of\_vrouter\_lcores \* number\_of\_ports\_in\_dpdk\_bond*

*\*Once VR descriptors are configured as well, vr\_mempool\_sz needs to have a total of required memory size for both.*

|  |
| --- |
| #etc/sysconfig/network-scripts/network-functions-vrouter-dpdk-env| grep DPDK\_COMMAND\_ADDITIONAL\_ARGS  DPDK\_COMMAND\_ADDITIONAL\_ARGS="${DPDK\_COMMAND\_ADDITIONAL\_ARGS:---vr\_flow\_entries=2000000 **--vr\_mempool\_sz 524288 --dpdk\_rx\_ring\_sz 4096 --dpdk\_tx\_ring\_sz 4096** --yield\_option 0 }" |

### 1.1.7 Others vRouter parameters

#### 1.1.7.1 vRouter disable yield\_option

For historical reasons the option is enabled in the vRouter, Juniper strongly recommends to disable it:

|  |
| --- |
| #etc/sysconfig/network-scripts/network-functions-vrouter-dpdk-env| grep DPDK\_COMMAND\_ADDITIONAL\_ARGS  DPDK\_COMMAND\_ADDITIONAL\_ARGS="${DPDK\_COMMAND\_ADDITIONAL\_ARGS:---vr\_flow\_entries=2000000 --vr\_mempool\_sz 524288 --dpdk\_rx\_ring\_sz 4096 --dpdk\_tx\_ring\_sz 4096 **--yield\_option 0** }" |

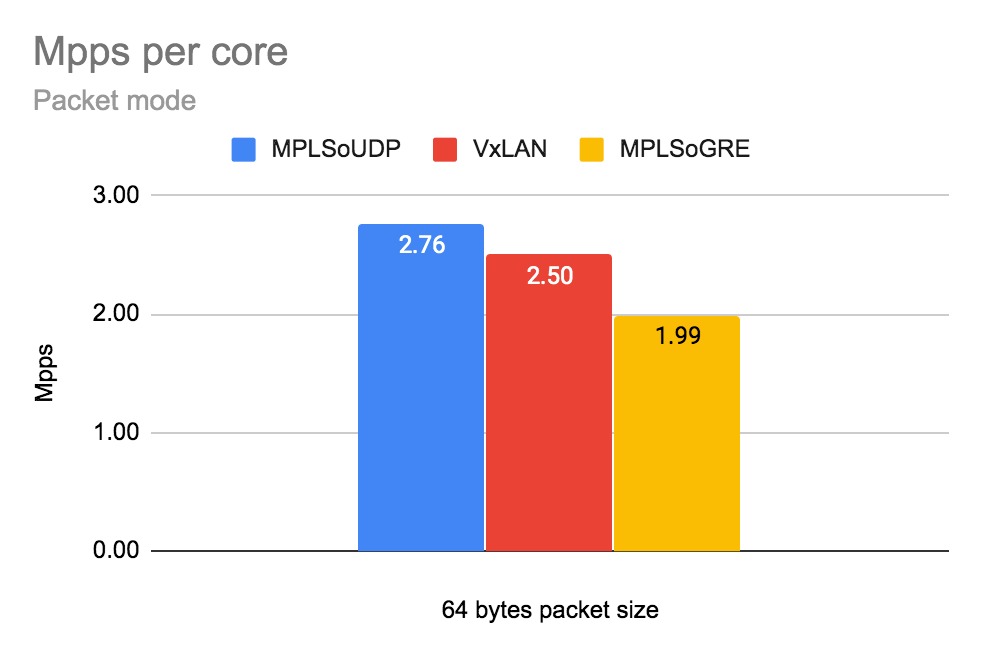
#### 1.1.7.2 LACP hashing

Enable L3/L4 hashing on vRouter and switch to utilize both links equally based on 5 tuple hash.

|  |
| --- |
| # cat /etc/sysconfig/network-scripts/ifcfg-vhost0  [...]  BOND\_POLICY=layer3+4 |

#### 1.1.7.2 Encapsulation

Use MPLSoUDP encapsulation as it provides the best performance.



# 

# End to end the example (Golden Configuration)

**Resources allocation a plan**

Available cores on the host: 56 logical cores

# lscpu

Architecture: x86\_64

CPU op-mode(s): 32-bit, 64-bit

Byte Order: Little Endian

CPU(s): 56

On-line CPU(s) list: 0-55

Thread(s) per core: 2

Core(s) per socket: 14

Socket(s): 2

NUMA node(s): 2

Vendor ID: GenuineIntel

CPU family: 6

Model: 85

Model name: Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz

Stepping: 4

CPU MHz: 2200.000

BogoMIPS: 4400.00

Virtualization: VT-x

L1d cache: 32K

L1i cache: 32K

L2 cache: 1024K

L3 cache: 19712K

NUMA node0 CPU(s): 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54

NUMA node1 CPU(s): 1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55

CPU 56 logical cores:

* Operating System: 0,28,1,29
* Nova: 10,12,14,16,18,20,22,24,26,38,40,42,44,46,48,50,52,54,3,5,7,9,11,13,15,17,19,21,23,25,27,31,33,35,37,39,41,43,45,47,49,51,53,55
* vRouter: 2,4,6,8,30,32,34,36
  + Forwarding threads: 2,4,6,8,30,32,34,3
  + Control threads: 0,28,1,29
  + Service threads: 0,28,1,29

Memory: 256G

* 220 x 1G huge pages (220G)
  + 218G Nova for VMs
  + 2G vRouter
* 8192 x 2M huge pages (16G)
* 20G Operating System

Number of NICs: 4 x 10G Intel x710

* 2 x 10G for vRouter DPDK
  + LACP: L3/L4
  + Rate: Fast
* 2 x 10G for Openstack

Buffer descriptors and memory pool size:

* Memory pool size(mempool\_sz): 262144B
* Inter core communication buffer descriptor size (TX/RX): 2048

## Configure high performance Contrail Cloud computes

site.yml

|  |
| --- |
| overcloud:  extra\_config:  NovaSchedulerDefaultFilters:  - RetryFilter  - DifferentHostFilter  - SameHostFilter  - AvailabilityZoneFilter  - ComputeFilter  - ComputeCapabilitiesFilter  - ImagePropertiesFilter  - ServerGroupAntiAffinityFilter  - ServerGroupAffinityFilter  - NUMATopologyFilter  - AggregateInstanceExtraSpecsFilter  NovaComputeExtraConfig:  nova::cpu\_allocation\_ratio: 1.0  nova::ram\_allocation\_ratio: 1.0  nova::disk\_allocation\_ratio: 1.0  #  # node 0 cpus: 0 2 4 6 8 10 12 14 16 18 20 22 24 26  # 28 30 32 34 36 38 40 42 44 46 48 50 52 54  #  # node 1 cpus: 1 3 5 7 9 11 13 15 17 19 21 23 25 27  # 29 31 33 35 37 39 41 43 45 47 49 51 53 55  #  # OS: 0,1,28,29 (4 cores, including siblings)  # vRouter: 2,4,6,8,30,32,34,36  # Nova: rest of available cores from NUMA0 and optionally NUMA1  ComputeDpdkParameters:  TunedProfileName: "cpu-partitioning"  ContrailDpdkOptions: "--vr\_flow\_entries=2500000 --vr\_mempool\_sz 262144 --vr\_dpdk\_rx\_ring\_sz 2048 --vr\_dpdk\_tx\_ring\_sz 2048 --yield\_option 0"  ContrailVrouterHugepages1GB: 220  ContrailVrouterHugepages2MB: 8192  IsolCpusList: "2-27,30-55"  KernelArgs: "isolcpus=2-27,30-55"  NovaVcpuPinSet: ['10','12','14','16','18','20','22','24','26','38','40','42','44','46','48','50','52','54']  nova::cpu\_allocation\_ratio: 1.0  nova::ram\_allocation\_ratio: 1.0  nova::disk\_allocation\_ratio: 1.0  ContrailSettings:  SERVICE\_CORE\_MASK: '0,1,28,29'  DPDK\_CTRL\_THREAD\_MASK: '0,1,28,29'  LACP\_RATE: 1 |

overcloud-nics.yml

|  |
| --- |
| # Compute Node for Swap VM (min CPU 2+2HT good to have possibility to increase during tests)  # Compute Node for the Generator (min CPU 4+4HT)  ComputeDpdk\_network\_config:  - type: interface  name: p1p1  dns\_servers:  get\_param: DnsServers  use\_dhcp: false  mtu:  get\_param: ControlPlaneNetworkMtu  addresses:  - ip\_netmask:  list\_join:  - '/'  - - get\_param: ControlPlaneIp  - get\_param: ControlPlaneSubnetCidr  routes:  -  ip\_netmask: 169.254.169.254/32  next\_hop:  get\_param: EC2MetadataIp  -  default: True  next\_hop:  get\_param: ControlPlaneDefaultRoute  use\_dhcp: false  - type: linux\_bond  name: ten2  use\_dhcp: false  bonding\_options: "mode=802.3ad xmit\_hash\_policy=layer3+4 lacp\_rate=fast updelay=1000 miimon=100"  mtu:  get\_param: InternalApi1NetworkMtu  members:  - type: interface  name: em3  primary: true  mtu:  get\_param: InternalApi1NetworkMtu  - type: interface  name: em4  mtu:  get\_param: InternalApi1NetworkMtu  - type: vlan  device: ten2  vlan\_id:  get\_param: InternalApi1NetworkVlanID  mtu:  get\_param: InternalApi1NetworkMtu  addresses:  - ip\_netmask:  get\_param: InternalApi1IpSubnet  routes:  -  ip\_netmask:  get\_param: InternalApiSupernet  next\_hop:  get\_param: InternalApi1InterfaceDefaultRoute  - type: vlan  device: ten2  vlan\_id:  get\_param: Storage1NetworkVlanID  mtu:  get\_param: Storage1NetworkMtu  addresses:  - ip\_netmask:  get\_param: Storage1IpSubnet  routes:  -  ip\_netmask:  get\_param: StorageSupernet  next\_hop:  get\_param: Storage1InterfaceDefaultRoute  - type: contrail\_vrouter\_dpdk  name: vhost0  driver: "{{ overcloud['contrail']['vrouter']['dpdk']['driver'] }}"  bond\_mode: 4  bond\_policy: layer3+4  **cpu\_list: "2,4,6,8,30,32,34,36" "**  members:  - type: interface  name: em1  - type: interface  name: em2  mtu:  get\_param: Tenant1NetworkMtu  addresses:  - ip\_netmask:  get\_param: Tenant1IpSubnet  routes:  -  ip\_netmask:  get\_param: TenantSupernet  next\_hop:  get\_param: Tenant1InterfaceDefaultRoute |

## 

## Configure high performance Contrail Networking on computes with TripleO heat templates

environments/contrail/environment-extra.yaml

|  |
| --- |
| ComputeDpdk1Hw0Parameters:  ContrailDpdkOptions: --vr\_flow\_entries=2500000 --vr\_mempool\_sz 262144 --vr\_dpdk\_rx\_ring\_sz 2048 --vr\_dpdk\_tx\_ring\_sz 2048 --yield\_option 0  ContrailSettings:  DPDK\_CTRL\_THREAD\_MASK: '0,1,28,29'  LACP\_RATE: 1  SERVICE\_CORE\_MASK: '0,1,28,29'  ContrailVrouterHugepages1GB: 220  ContrailVrouterHugepages2MB: 8192  IsolCpusList: 2-27,30-55  KernelArgs: isolcpus=2-27,30-55  NovaVcpuPinSet:  - '10'  - '12'  - '14'  - '16'  - '18'  - '20'  - '22'  - '24'  - '26'  - '38'  - '40'  - '42'  - '44'  - '46'  - '48'  - '50'  - '52'  - '54'  TunedProfileName: cpu-partitioning  nova::cpu\_allocation\_ratio: 1.0  nova::disk\_allocation\_ratio: 1.0  nova::ram\_allocation\_ratio: 1.0 |

overcloud-nics.yml

|  |
| --- |
| ComputeDpdk\_network\_config:  - type: interface  name: p1p1  dns\_servers:  get\_param: DnsServers  use\_dhcp: false  mtu:  get\_param: ControlPlaneNetworkMtu  addresses:  - ip\_netmask:  list\_join:  - '/'  - - get\_param: ControlPlaneIp  - get\_param: ControlPlaneSubnetCidr  routes:  -  ip\_netmask: 169.254.169.254/32  next\_hop:  get\_param: EC2MetadataIp  -  default: True  next\_hop:  get\_param: ControlPlaneDefaultRoute  use\_dhcp: false  - type: linux\_bond  name: ten2  use\_dhcp: false  bonding\_options: "mode=802.3ad xmit\_hash\_policy=layer3+4 lacp\_rate=fast updelay=1000 miimon=100"  mtu:  get\_param: InternalApi1NetworkMtu  members:  - type: interface  name: em3  primary: true  mtu:  get\_param: InternalApi1NetworkMtu  - type: interface  name: em4  mtu:  get\_param: InternalApi1NetworkMtu  - type: vlan  device: ten2  vlan\_id:  get\_param: InternalApi1NetworkVlanID  mtu:  get\_param: InternalApi1NetworkMtu  addresses:  - ip\_netmask:  get\_param: InternalApi1IpSubnet  routes:  -  ip\_netmask:  get\_param: InternalApiSupernet  next\_hop:  get\_param: InternalApi1InterfaceDefaultRoute  - type: vlan  device: ten2  vlan\_id:  get\_param: Storage1NetworkVlanID  mtu:  get\_param: Storage1NetworkMtu  addresses:  - ip\_netmask:  get\_param: Storage1IpSubnet  routes:  -  ip\_netmask:  get\_param: StorageSupernet  next\_hop:  get\_param: Storage1InterfaceDefaultRoute  - type: contrail\_vrouter\_dpdk  name: vhost0  driver: "{{ overcloud['contrail']['vrouter']['dpdk']['driver'] }}"  bond\_mode: 4  bond\_policy: layer3+4  **cpu\_list: "2,4,6,8,30,32,34,36"**  members:  - type: interface  name: em1  - type: interface  name: em2  mtu:  get\_param: Tenant1NetworkMtu  addresses:  - ip\_netmask:  get\_param: Tenant1IpSubnet  routes:  -  ip\_netmask:  get\_param: TenantSupernet  next\_hop:  get\_param: Tenant1InterfaceDefaultRoute |

# 