Red Hat OpenStack Platform 13

OVS-DPDK 最终用户故障排除指南

包含 OVS-DPDK 端到端故障排除步骤的指南

Last Updated: 2023-04-20

Red Hat OpenStack Platform 13 OVS-DPDK 最终用户故障排除指南

包含 OVS-DPDK 端到端故障排除步骤的指南

法律通告

摘要

OVS-DPDK 系统管理员的流程用于识别和解决与 Red Hat OpenStack Platform 13 中数据包丢失相关的常见问题。

目录

前言

本文档包含 OVS-DPDK 系统管理员的步骤，用于识别和解决与 Red Hat OpenStack Platform 13 中数据包丢失相关的问题。本指南中记录的步骤会取代之前发布的知识库文章。

第 1 章 初步检查

本指南假定您已熟悉以下文档中的规划和部署步骤：

- 规划 OVS-DPDK 部署
- 配置 OVS-DPDK 部署

第 2 章 验证 OVS-DPDK 部署

本章论述了在部署之后执行验证步骤。

2.1. 确认 OPENSTACK

使用下列命令确认 OpenStack 和 OVS-DPDK 配置。

2.1.1. 显示网络代理

确保 **Alive** 的值为 **True**，并且每个代理的 **State** 为 **UP**。如果有任何问题，请查看 **/var/log/containers/neutron** 和 **/var/log/openvswitch/ovs-vswitchd.log** 中的日志以确定问题。

```
$ openstack network agent list

+------------------+-----------------+-----------------+-------------------+-------+-------+--------------------+
| ID               | Agent Type      | Host            | Availability Zone | Alive | State | Binary             |
+------------------+-----------------+-----------------+-------------------+-------+-------+--------------------+
| 19188fa7-50f1-4a | DHCP agent      | control-0.locald | nova              | True  | UP    | neutron-dhcp-
agent |
| b1-a86c-         |                 | omain           |                   |       |       |                    |
| 986724e6e75d     |                 |                 |                   |       |       |                    |
| 6b58175c-a07e-49 | L3 agent        | control-0.locald | nova              | True  | UP    | neutron-l3-agent
|
| 56-a736-dc2a3f27 |                 | omain           |                   |       |       |                    |
| 2a34             |                 |                 |                   |       |       |                    |
| b4bc9e26-959c-   | Metadata agent  | control-0.locald | None              | True  | UP    | neutron-
metadata-  |
| 402a-ab24-b7ccad |                 | omain           |                   |       | agent |                    |
| b8119f           |                 |                 |                   |       |       |                    |
| eb7df511-5e09-46 | Open vSwitch    | control-0.locald | None              | True  | UP    | neutron-
|
| 55-a82d-         | agent           | omain           |                   |       | openvswitch-agent  |
| 8aa52537f730     |                 |                 |                   |       |       |                    |
| fc1a71f0-06af-   | Open vSwitch    | compute-0.locald | None              | True  | UP    | neutron-           |
| 43e3-b48a-       | agent           | omain           |                   |       | openvswitch-agent  |
| f0923bcec843     |                 |                 |                   |       |       |                    |
+------------------+-----------------+-----------------+-------------------+-------+-------+--------------------+
```

2.1.2. 显示 Compute Service 中的主机

确保 **Status** 的值为 **enabled**，并且每个主机的状态为 **up State**。如果有任何问题，请查看 **/var/log/containers/nova** 中的日志以确定问题。

```
$ openstack compute service list

+----+-----------------+----------------------+----------+---------+-------+----------------------------+
| ID | Binary          | Host                 | Zone     | Status  | State | Updated At                 |
+----+-----------------+----------------------+----------+---------+-------+----------------------------+
|  3 | nova-consoleauth | control-0.localdomain | internal | enabled | up    | 2019-02-
06T16:21:52.000000 |
|  4 | nova-scheduler   | control-0.localdomain | internal | enabled | up    | 2019-02-
06T16:21:51.000000 |
|  5 | nova-conductor   | control-0.localdomain | internal | enabled | up    | 2019-02-
```

```
06T16:21:50.000000 |
| 6 | nova-compute    | compute-0.localdomain | dpdk    | enabled | up   | 2019-02-
06T16:21:45.000000 |
+----+-----------------+-----------------------+---------+---------+-------+----------------------------+
```

有关确认 Red Hat OpenStack Platform 配置的更多信息，*请参阅升级 Red Hat OpenStack Platform* 指南中的 验证容器化 overcloud。

2.2. 确认计算节点 OVS 配置

要验证网络适配器和 OpenvSwitch 的配置和健康状况，请完成以下步骤：

1. 要验证计算节点上的 DPDK 网络设备，请运行以下命令。该 rpm 在 repo 中找到： **rhel-7-server-extras-rpms**。

```
$ yum install dpdk-tools
```

2. 显示由 DPDK 管理的网络设备以及用于联网的网络设备。

```
$ dpdk-devbind --status
```

使用 DPDK 驱动程序的设备在 Tripleo 计算角色模板中是 **ovs_dpdk_bond** 或 **ovs_dpdk_port** 类型：

```
Network devices using DPDK-compatible driver
============================================
0000:04:00.1 'Ethernet 10G 2P X520 Adapter 154d' drv=vfio-pci unused=
0000:05:00.0 'Ethernet 10G 2P X520 Adapter 154d' drv=vfio-pci unused=

Network devices using kernel driver
===================================
0000:02:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em1 drv=tg3 unused=vfio-pci *Active*
0000:02:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em2 drv=tg3 unused=vfio-pci
0000:03:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em3 drv=tg3 unused=vfio-pci
0000:03:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=em4 drv=tg3 unused=vfio-pci *Active*
0000:04:00.0 'Ethernet 10G 2P X520 Adapter 154d' if=p1p1 drv=ixgbe unused=vfio-pci
0000:05:00.1 'Ethernet 10G 2P X520 Adapter 154d' if=p2p2 drv=ixgbe unused=vfio-pci
```

3. 运行以下命令来确认 DPDK 已启用：

```
$ sudo  ovs-vsctl get Open_vSwitch . iface_types

[dpdk, dpdkr, dpdkvhostuser, dpdkvhostuserclient, geneve, gre, internal, lisp, patch, stt, system, tap, vxlan]
```

4. 运行以下命令。结果显示来自 DPDK 兼容驱动程序的 PCI 设备，例如 **0000:04:00.1** 和 **:05:00.0** 作为 **type: dpdk**，且没有错误。

```
$ ovs-vsctl show
```

```
Bridge "br-link0"
        Controller "tcp:127.0.0.1:6633"
            is_connected: true
        fail_mode: secure
        Port "phy-br-link0"
            Interface "phy-br-link0"
                type: patch
                options: {peer="int-br-link0"}
        Port "dpdkbond0"
            Interface "dpdk1"
                type: dpdk
                options: {dpdk-devargs="0000:04:00.1", n_rxq="2"}
            Interface "dpdk0"
                type: dpdk
                options: {dpdk-devargs="0000:05:00.0", n_rxq="2"}
        Port "br-link0"
            Interface "br-link0"
                type: internal
    ovs_version: "2.9.0"
```

以下输出显示了错误：

```
Port "dpdkbond0"
            Interface "dpdk1"
                type: dpdk
                options: {dpdk-devargs="0000:04:00.1", n_rxq="2"}
                error: "Error attaching device '0000:04:00.1' to DPDK"
```

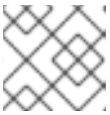
5. 要显示接口详情，请运行以下命令：

```
$ sudo ovs-vsctl list interface dpdk1 | egrep "name|mtu|options|status"
```

6. 运行以下命令。请注意，未启用 lacp。

```
$ ovs-appctl bond/show dpdkbond0

bond_mode: active-backup
bond may use recirculation:
no, Recirc-ID : -1
bond-hash-basis: 0
updelay: 0 ms
downdelay: 0 ms
lacp_status: off
lacp_fallback_ab: false
active slave mac: a0:36:9f:e5:da:82(dpdk1)

slave dpdk0: enabled
    may_enable: true

slave dpdk1: enabled
    active slave
    may_enable: true
```

7. 检查计算节点上的所有 ovs 网桥是否都为 **netdev**，表示快速数据路径（用户空间）联网



注意

不支持混合系统（内核）和 netdev（用户空间）数据路径类型。

```
$ ovs-vsctl list bridge | grep -e name -e datapath_type

datapath_type       : netdev
name                : br-int
datapath_type       : netdev
name                : "br-link0"
```

8. 运行以下命令来检查持久性 Open vSwitch 错误：

```
$ grep ERROR /var/log/openvswitch/ovs-vswitchd.log
```

2.3. 确认用于实例配置的 OVS

为确保 vhostuser DMA 正常工作，请将具有 OVS-DPDK 端口的实例配置为使用类别启用专用 CPU 和巨页。有关更多信息，请参阅： 为 OVS-DPDK 创建类别和部署实例。

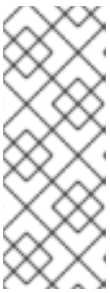
要确认实例配置，请完成以下步骤：

1. 确认实例已固定 CPU。可以使用 **virsh** 识别专用 CPU：

```
$ sudo virsh vcpupin 2
```

2. 确认用于实例的仿真程序线程没有在分配给该实例的同一 vCPU 上运行：

```
$ sudo virsh emulatorpin 2
```



注意

从 Red Hat OpenStack Platform 12 开始，您可以选择通过类别运行模拟器线程的位置。请参阅使用 Red Hat OpenStack Platform 12 配置仿真程序线程策略 。

对于较旧版本，您必须在实例开机时手动执行仿真程序线程固定。请参阅关于在带有 NFV 的虚拟环境中使用 virsh 模拟器固定的影响（以及没有 isolcpus）以及优化仿真程序线程固定。

3. 确认实例正在使用巨页，这是最佳性能所必需的。

```
$ sudo virsh numatune 1
```

4. 确认实例的接收队列是否由轮询模式驱动程序(PMD)提供服务。
端口和队列应该在 PMD 之间平等平衡。最佳情况下，端口将由与网络适配器相同的 NUMA 节点中的 CPU 服务。

```
$ sudo ovs-appctl dpif-netdev/pmd-rxq-show

pmd thread numa_id 0 core_id 2:
   isolated : false
   port: dpdk0             queue-id:  1    pmd usage:  0 %
```

```
    port: dpdk1              queue-id:  0    pmd usage:  0 %
    port: vhu94ccc316-ea     queue-id:  0    pmd usage:  0 %
pmd thread numa_id 1 core_id 3:
    isolated : false
pmd thread numa_id 0 core_id 22:
    isolated : false
    port: dpdk0              queue-id:  0    pmd usage:  0 %
    port: dpdk1              queue-id:  1    pmd usage:  0 %
    port: vhu24e6c032-db     queue-id:  0    pmd usage:  0 %
pmd thread numa_id 1 core_id 23:
    isolated : false
```

5. 显示 PMD 的统计信息。这有助于确定在 PMD 之间如何平衡队列。如需更多信息，请参阅 Open vSwitch 文档中的 PMD Threads。

注意

pmd-rxq-rebalance 选项在 OVS 2.9.0 中添加。此命令执行新的 PMD 队列分配，以便根据最新的 rxq 处理周期信息在 PMD 之间平等平衡。

pmd-stats-show 命令显示自 PMDs 运行或上次清除统计以来的完整历史记录。如果没有清除它，在设置端口前，它将融入到统计中，数据会被流处理。如果用于查看数据路径上的负载（通常是这样），那么将无用。

最好将系统置于稳定状态，清除 stats，等待几秒钟，然后显示统计信息。这可让您准确显示 datapath。

使用以下命令来显示 PMD 的统计信息：

```
$ sudo ovs-appctl dpif-netdev/pmd-stats-show

pmd thread numa_id 0 core_id 2:
    packets received: 492207
    packet recirculations: 0
    avg. datapath passes per packet: 1.00
    emc hits: 419949
    megaflow hits: 2485
    avg. subtable lookups per megaflow hit: 1.33
    miss with success upcall: 69773
    miss with failed upcall: 0
    avg. packets per output batch: 1.00
    idle cycles: 1867450752126715 (100.00%)
    processing cycles: 5274066849 (0.00%)
    avg cycles per packet: 3794046054.19 (1867456026193564/492207)
    avg processing cycles per packet: 10715.14 (5274066849/492207)
pmd thread numa_id 1 core_id 3:
    packets received: 0
    packet recirculations: 0
    avg. datapath passes per packet: 0.00
    emc hits: 0
    megaflow hits: 0
    avg. subtable lookups per megaflow hit: 0.00
    miss with success upcall: 0
    miss with failed upcall: 0
    avg. packets per output batch: 0.00
```

```
pmd thread numa_id 0 core_id 22:
  packets received: 493258
  packet recirculations: 0
  avg. datapath passes per packet: 1.00
  emc hits: 419755
  megaflow hits: 3223
  avg. subtable lookups per megaflow hit: 1.49
  miss with success upcall: 70279
  miss with failed upcall: 1
  avg. packets per output batch: 1.00
  idle cycles: 1867449561100794 (100.00%)
  processing cycles: 6465180459 (0.00%)
  avg cycles per packet: 3785961963.68 (1867456026281253/493258)
  avg processing cycles per packet: 13107.10 (6465180459/493258)
pmd thread numa_id 1 core_id 23:
  packets received: 0
  packet recirculations: 0
  avg. datapath passes per packet: 0.00
  emc hits: 0
  megaflow hits: 0
  avg. subtable lookups per megaflow hit: 0.00
  miss with success upcall: 0
  miss with failed upcall: 0
  avg. packets per output batch: 0.00
main thread:
  packets received: 16
  packet recirculations: 0
  avg. datapath passes per packet: 1.00
  emc hits: 1
  megaflow hits: 9
  avg. subtable lookups per megaflow hit: 1.00
  miss with success upcall: 6
  miss with failed upcall: 0
  avg. packets per output batch: 1.00
```

6. 重置 PMD 统计信息。**pmd-stats-show** 命令显示自上次 **pmd-stats-clear** 命令以来的 PMD 统计信息。如果没有以前的 **pmd-stats-clear**，它会包含自 PMD 开始运行时的数据。
   如果您在负载下检查系统，清除 PMD 统计信息会很有用，然后显示它们。否则，当系统未加载（流量流）前，统计也可以包含之前时间中的数据。

   使用以下命令重置 PMD 统计信息：

   ```
   $ sudo ovs-appctl dpif-netdev/pmd-stats-clear
   ```

2.4. 其他帮助命令

使用这些命令执行额外的验证检查。

- 查找由 os-net-config 配置的 OVS-DPDK 端口和物理 NIC 映射

  ```
  cat /var/lib/os-net-config/dpdk_mapping.yaml
  ```

- 使用 Nova 实例 $ID 查找实例的 DPDK 端口

```
sudo  ovs-vsctl find interface external_ids:vm-uuid="$ID" | grep ^name
```

- 使用 DPDK 端口查找实例的 Nova ID

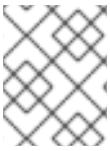
```
sudo  ovs-vsctl get interface vhu24e6c032-db external_ids:vm-uuid
```

- 在 dpdk 端口上执行 tcpdump

```
sudo ovs-tcpdump -i vhu94ccc316-ea
```

注意

OVS-tcpdump 来自位于 **rhel-7-server-openstack-13-devtools-rpms** repo 中的 openvswitch-test RPM。



注意

对于性能问题，不建议在生产环境中使用 **ovs-tcpdump**。如需更多信息，请参阅： 如何在 Red Hat OpenStack Platform 中对 vhost-user 接口使用 ovs-tcpdump？

2.5. 简单 COMPUTE 节点 CPU 分区和文件系统检查

先决条件

在部署的计算节点上运行此命令，并记下 cpu masks 如何映射到 TripleO Heat 模板值：

```
$ sudo ovs-vsctl get Open_vSwitch . other_config

{dpdk-init="true", dpdk-lcore-mask="300003", dpdk-socket-mem="3072,1024", pmd-cpu-mask="c0000c"}
```

注意以下几点：

- **DPDK-lcore-mask** 映射到 TripleO Heat Templates 中的 **OvsDpdkCoreList**。
- **DPDK-socket-mem** 映射到 TripleO Heat Templates 中的 **OvsDpdkSocketMemory**。
- TripleO Heat Templates 中的 **PMD-cpu-mask** 映射到 **OvsPmdCoreList**。
  要将这些 CPU 掩码转换为十进制值，可以将其协调回 TripleO Heat 模板和实际系统值 see： 如何将十六进制 CPU 掩码转换为位掩码并识别掩码的 CPU？

2.5.1. 检测 CPU

要检测 CPU for pid 1，请使用以下命令。这些内核不应运行 PMD 或 Nova vCPU：

```
$ taskset -c -p 1

pid 1's current affinity list: 0,1,20,21
```

2.5.2. 检测 PMD 线程

要查看 PMD 线程，请使用以下命令：输出应反映 Tripleo 参数 **OvsPmdCoreList** 的值。**OvsDpdkCoreList** 或 **HostIsolatedCoreslist** 的值不应与 Tripleo 参数的值重叠：

```
$ ps -T -o spid,comm -p $(pidof ovs-vswitchd) |grep '\<pmd' |while read spid name; do echo $name
$(taskset -p -c $spid); done

pmd44 pid 679318's current affinity list: 3
pmd45 pid 679319's current affinity list: 23
pmd46 pid 679320's current affinity list: 22
pmd47 pid 679321's current affinity list: 2
```

2.5.3. 检测 NUMA 节点

为了获得最佳性能，请确保实例的物理网络适配器、PMD 线程和固定 CPU 都在同一个 NUMA 节点上。如需更多信息，请参阅： CPU 和 NUMA 节点。

以下是检查 NUMA 分配的简单练习。

1. 检查计算节点上实例的 vhu 端口：

```
$ sudo virsh domiflist 1

Interface  Type       Source     Model       MAC
-------------------------------------------------------
vhu24e6c032-db vhostuser  -          virtio      fa:16:3e:e3:c4:c2
```

2. 检查为该端口提供服务并记录 NUMA 节点的 PMD 线程：

```
$ sudo ovs-appctl dpif-netdev/pmd-rxq-show

pmd thread numa_id 0 core_id 2:
   isolated : false
   port: vhu24e6c032-db      queue-id:  0    pmd usage:  0 %
   port: vhu94ccc316-ea      queue-id:  0    pmd usage:  0 %
```

3. 查找实例的物理固定 cpu。例如，注意此实例的端口位于 cpu 2 上，并且实例由 cpus 34 和 6 服务。

```
$ sudo virsh dumpxml 1 | grep cpuset

    <vcpupin 1 vcpu='0' cpuset='34'/>
    <emulatorpin cpuset='6'/>
```

4. 检查每个 NUMA 节点的内核。请注意，为实例提供服务(34,6)的 CPU 位于同一 NUMA 节点(0)上。

```
$ lscpu | grep ^NUMA

NUMA node(s):          2
NUMA node0 CPU(s):     0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38
NUMA node1 CPU(s):     1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39
```

另外，不受 OVS DPDK 管理的网络适配器在此有一个条目，用于指示它们所属的 NUMA 节点：

-

```
$ sudo cat /sys/class/net/<device name>/device/numa_node
```

另外，您可以通过查询 PCI 地址（即使由 OVS DPDK 管理的）来查看网络适配器的 NUMA 节点：

```
$ sudo lspci -v -s 05:00.1 | grep -i numa

Flags: bus master, fast devsel, latency 0, IRQ 203, NUMA node 0
```

这些练习演示了 PMD、实例和网络适配器都位于 NUMA 0 上，这是性能最佳。从 openvswitch 日志（位于 **/var/log/openvswitch**）中的跨 NUMA 轮询，寻找类似于以下内容的日志条目：

```
dpif_netdev|WARN|There's no available (non-isolated) pmd thread on numa node 0. Queue 0 on port 'dpdk0' will be assigned to the pmd on core 7 (numa node 1). Expect reduced performance.
```

2.5.4. 检测隔离 CPU

使用以下命令来显示隔离的 CPU。其输出应当与 TripleO 参数 **IsolCpusList** 的值相同。

```
$ cat /etc/tuned/cpu-partitioning-variables.conf | grep -v ^#

isolated_cores=2-19,22-39
```

2.5.5. 检测分配给 Nova 实例的 CPU

使用以下命令，显示专用于 Nova 实例的 CPU。这个输出应该和没有轮询模式驱动程序(PMD)CPU 的 isolcpus 的值相同：

```
$ grep ^vcpu_pin_set /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf

vcpu_pin_set=4-19,24-39
```

2.5.6. 确认 Huge Pages 配置

检查计算节点上的巨页配置。

```
[root@compute-0 ~]# cat /sys/devices/system/node/node*/meminfo  | grep -i huge
Node 0 AnonHugePages:      4096 kB
Node 0 HugePages_Total:    16
Node 0 HugePages_Free:     11
Node 0 HugePages_Surp:      0
Node 1 AnonHugePages:      8192 kB
Node 1 HugePages_Total:    16
Node 1 HugePages_Free:     15
Node 1 HugePages_Surp:      0
```

如果没有配置巨页或已耗尽，请参阅 KernelArgs。

2.6. PACKET DROPS 原因

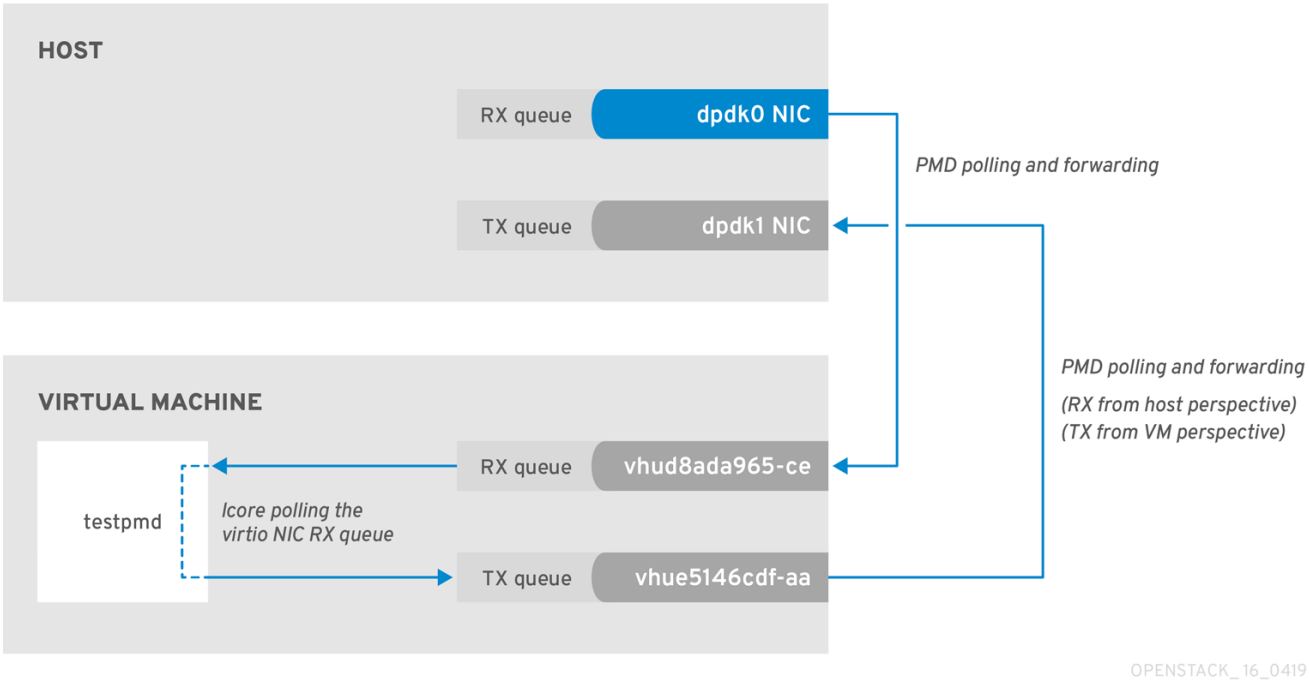
当队列已满时，数据包会被丢弃，通常是当队列没有足够快时，数据包被丢弃。瓶颈是当队列没有足够快排空时，应该排空队列的实体。在大多数情况下，使用 drop 计数器来跟踪丢弃的数据包。有时，硬件或软件设计中的一个错误可能会导致数据包跳过丢弃计数器。

Data Plan Development Kit(DPDK)包括用于转发数据包的 **testpmd** 应用。在本章中显示的情况下，testpmd 安装在虚拟机上，并轮询带有分配的逻辑内核(lcore)的端口，以将数据包从一个端口转发到另一个端口。**testpmd 常与流量生成器一起使用，在本例中为物理虚拟物理(PVP)路径中的吞吐量。**

2.6.1. OVS-DPDK Too Slow to Drain physical NIC

本例显示 PMD 线程负责轮询物理网络适配器(dpdk0)的接收(RX)队列。当 PMD 线程无法跟上数据包卷或中断时，数据包可能会被丢弃。

图 **2.1.** 轮询物理适配器 **RX** 队列



以下命令显示 dpdk0 界面中的统计信息。如果数据包被丢弃，因为 ovs-dpdk 没有快排空物理适配器，您会看到 rx_dropped 的速度会快速增长。

注意

PMD 应该为每个 NUMA 节点有一个以上物理 CPU 内核。

```
# ovs-vsctl --column statistics list interface dpdk0

statistics          : {mac_local_errors=0, mac_remote_errors=0, "rx_1024_to_1522_packets"=26,
"rx_128_to_255_packets"=243,
"rx_1523_to_max_packets"=0, "rx_1_to_64_packets"=102602, "rx_256_to_511_packets"=6100,
```

"rx_512_to_1023_packets"=27,
"rx_65_to_127_packets"=16488, rx_broadcast_packets=2751, rx_bytes=7718218, rx_crc_errors=0,
rx_dropped=0, rx_errors=0,
rx_fragmented_errors=0, rx_illegal_byte_errors=0, rx_jabber_errors=0, rx_length_errors=0,
rx_mac_short_dropped=0,
rx_mbuf_allocation_errors=0, rx_oversize_errors=0, rx_packets=125486, rx_undersized_errors=0,
"tx_1024_to_1522_packets"=63,
"tx_128_to_255_packets"=319, "tx_1523_to_max_packets"=0, "tx_1_to_64_packets"=1053,
"tx_256_to_511_packets"=50,
"tx_512_to_1023_packets"=68, "tx_65_to_127_packets"=7732, tx_broadcast_packets=12,
tx_bytes=466813, tx_dropped=0,
tx_errors=0, tx_link_down_dropped=0, tx_multicast_packets=5642, tx_packets=9285}

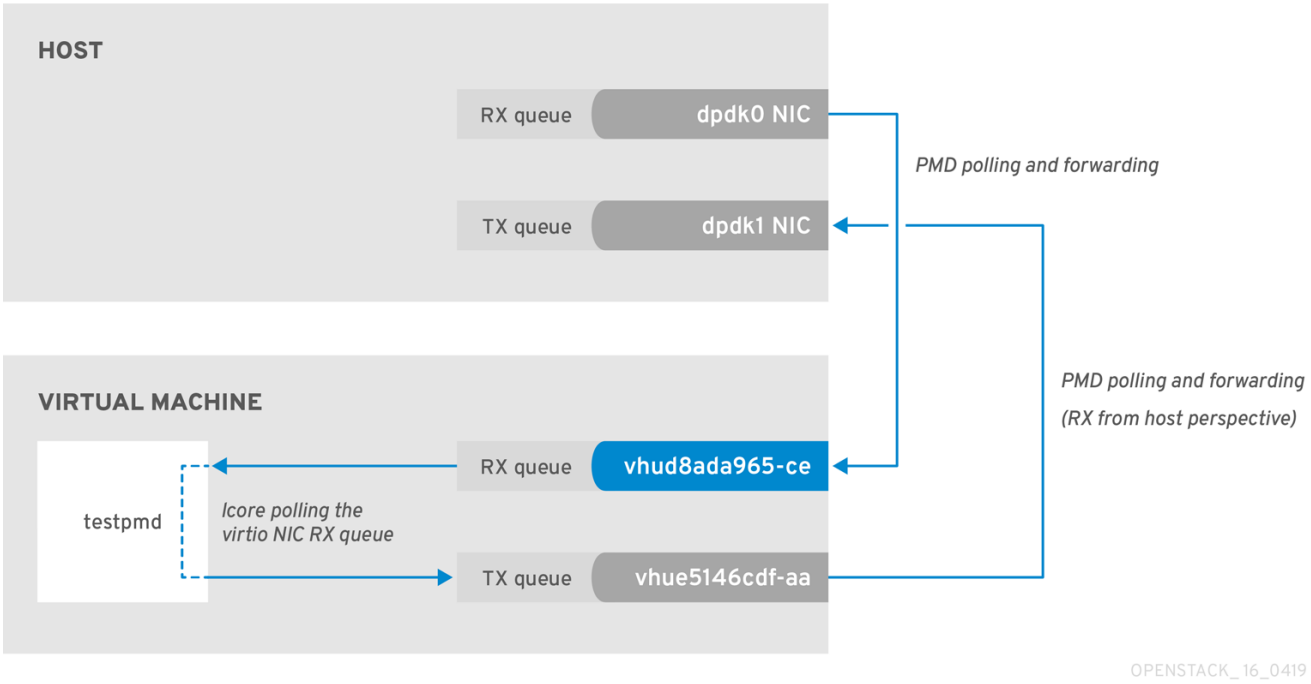
2.6.2. VM Too Slow to Drain vhost-user

此示例与 图 2.1 中的示例类似，如果 lcore 线程超过发送到实例接收(RX)队列的数据包卷，则可能会遇到数据包丢失。

如需更多信息，请参阅以下文章：

- **关于在具有 NFV 的虚拟环境中使用 virsh 模拟器固定的影响，以及没有 isolcpus 以及最佳仿真程序线程固定等影响**
- **使用 Red Hat OpenStack Director 更改连接到 OVS DPDK 的 virtio NIC 的 RX 队列大小和 TX 队列大小**

图 2.2. 轮询虚拟适配器 RX 队列



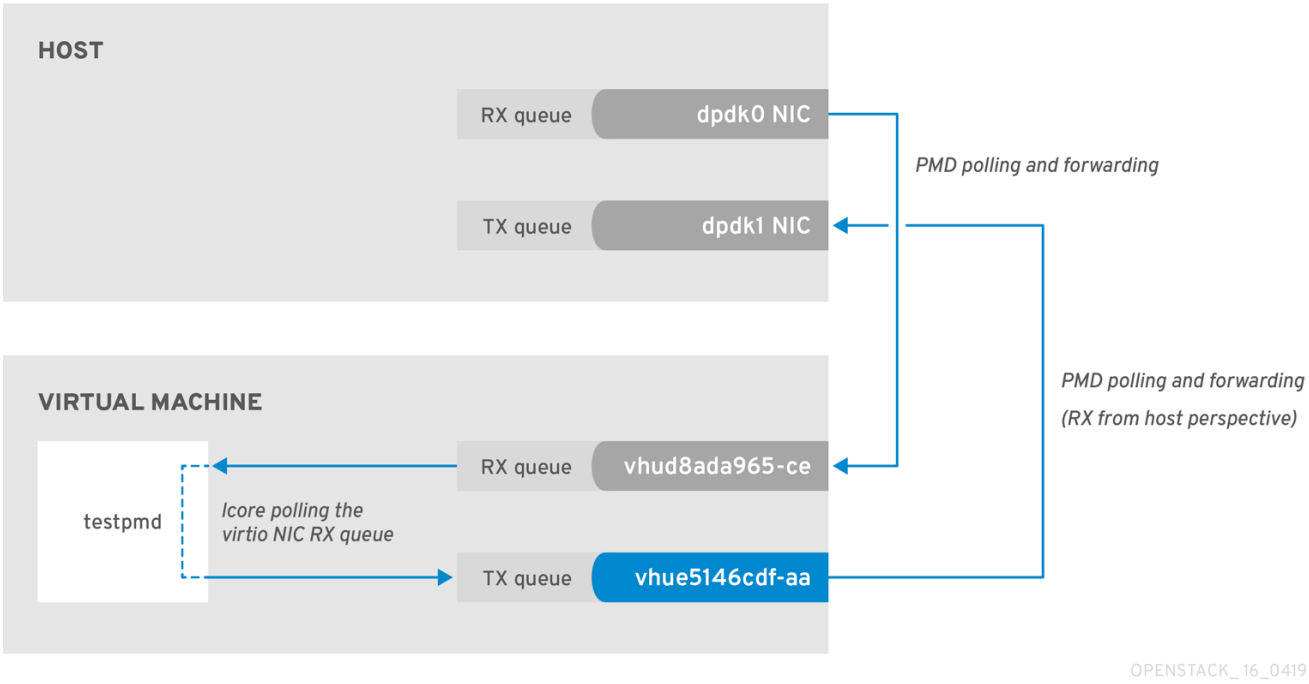
要检查主机的 tx_dropped 值是否与虚拟机的 rx_dropped 值对应，请运行以下命令：

```
ovs-vsctl --column statistics list interface vhud8ada965-ce

statistics          : {"rx_1024_to_1522_packets"=0, "rx_128_to_255_packets"=0,
"rx_1523_to_max_packets"=0,
"rx_1_to_64_packets"=0, "rx_256_to_511_packets"=0, "rx_512_to_1023_packets"=0,
"rx_65_to_127_packets"=0, rx_bytes=0,
rx_dropped=0, rx_errors=0, rx_packets=0, tx_bytes=0, tx_dropped=0, tx_packets=0}
```

2.6.3. OVS-DPDK Too Slow to Drain vhost-user

在本例中，MonyD 线程是从主机角度轮询 virtio TX 的接收队列。如果 PMD 线程被数据包卷超过，或者中断，数据包可能会丢弃。

图 2.3. 轮询虚拟适配器 TX 队列



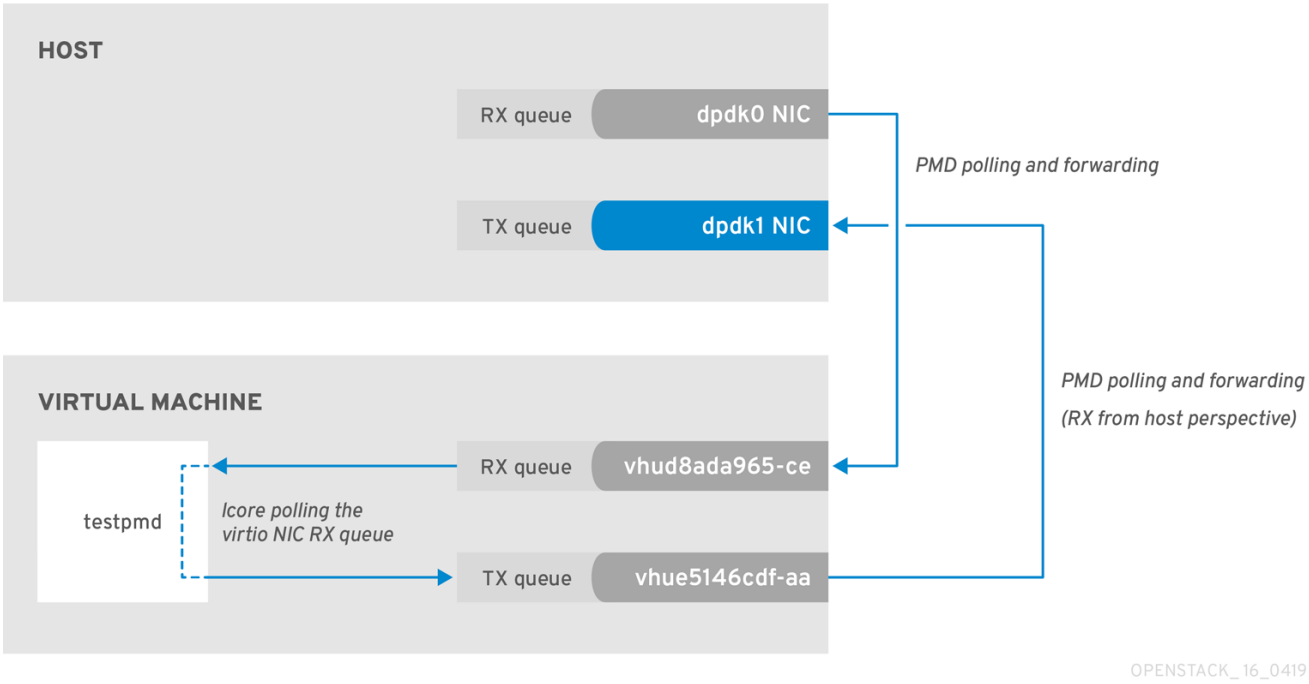
跟踪来自虚拟机的数据包的返回路径，并提供主机上的丢弃计数器(tx_dropped)和 VM(rx_dropped) 端值，运行以下命令：

```
ovs-vsctl --column statistics list interface vhue5146cdf-aa

statistics          : {"rx_1024_to_1522_packets"=0, "rx_128_to_255_packets"=0,
"rx_1523_to_max_packets"=0,
"rx_1_to_64_packets"=0, "rx_256_to_511_packets"=0, "rx_512_to_1023_packets"=0,
"rx_65_to_127_packets"=0,
rx_bytes=0, rx_dropped=0, rx_errors=0, rx_packets=0, tx_bytes=0, tx_dropped=0, tx_packets=0}
```

2.6.4. Egress 物理接口上的数据包丢失

PCIe 和 RAM 之间的慢速传输速率可能会导致物理适配器从 TX 队列丢弃数据包。虽然这不经常存在，但是了解如何识别和解决这个问题非常重要。

图 2.4. 轮询物理适配器 TX 队列



以下命令显示 dpdk1 界面中的统计信息。如果 tx_dropped 大于零且快速增长，则创建一个红帽支持问题单。

```
ovs-vsctl --column statistics list interface dpdk1

statistics          : {mac_local_errors=0, mac_remote_errors=0, "rx_1024_to_1522_packets"=26,
"rx_128_to_255_packets"=243, "rx_1523_to_max_packets"=0, "rx_1_to_64_packets"=102602,
"rx_256_to_511_packets"=6100,
"rx_512_to_1023_packets"=27, "rx_65_to_127_packets"=16488, rx_broadcast_packets=2751,
rx_bytes=7718218,
rx_crc_errors=0, rx_dropped=0, rx_errors=0, rx_fragmented_errors=0, rx_illegal_byte_errors=0,
rx_jabber_errors=0,
rx_length_errors=0, rx_mac_short_dropped=0, rx_mbuf_allocation_errors=0, rx_oversize_errors=0,
rx_packets=125486,
rx_undersized_errors=0, "tx_1024_to_1522_packets"=63, "tx_128_to_255_packets"=319,
"tx_1523_to_max_packets"=0,
"tx_1_to_64_packets"=1053, "tx_256_to_511_packets"=50, "tx_512_to_1023_packets"=68,
"tx_65_to_127_packets"=7732,
tx_broadcast_packets=12, tx_bytes=466813, tx_dropped=0, tx_errors=0, tx_link_down_dropped=0,
tx_multicast_packets=5642, tx_packets=9285}
```

如果您看到这些类型的数据包丢失，请考虑重新配置内存频道。

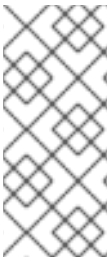
- 要计算内存频道，请参阅：网络功能虚拟化规划和聚合指南 中的内存参数。https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/{vernum}/html-

single/network_functions_virtualization_planning_and_configuration_guide/index#c_ovsdpdk-memory-params

- 要确定内存频道的数量，请参阅：如何确定 Red Hat OpenStack Platform 中的 NeutronDpdkMemoryChannels 或 OvsDpdkMemoryChannels 的内存频道数。

第 3 章 NFV 命令备忘单

本章包含许多 Red Hat OpenStack Platform 13 系统可观察性最常用的命令。



注意

默认情况下，下面的某些命令可能不可用。要为给定节点安装所需的工具，请运行以下命令：

```
sudo yum install tuna qemu-kvm-tools perf kernel-tools dmidecode
```

3.1. UNIX 套接字

使用这些命令显示进程端口和 UNIX 套接字域。

操作	命令
显示所有状态（LISTEN、Win6、CLOSE_WAIT 等等）的所有 TCP 和 UDP SOCKETS	**# lsof -ni**
显示所有状态（LISTEN、Win6、CLOSE_WAIT 等等）的所有 TCP SOCKETS	**# lsof -nit**
显示所有状态（LISTEN、Win6、CLOSE_WAIT 等等）的所有 UDP SOCKETS	**# lsof -niu**
显示所有状态（LISTEN、Win6、CLOSE_WAIT 等）且没有主机名查找 IPv4 的所有 TCP 和 UDP SOCKETS	**# lsof -ni4**
显示所有状态（LISTEN、Win6、CLOSE_WAIT 等）且没有主机名查找 IPv6 的所有 TCP 和 UDP SOCKETS	**# lsof -ni6**
显示所有相关的 SOCKETS（LISTEN、Win6、CLOSE_WAIT 等），但不为给定端口进行主机名查找	**# lsof -ni :4789**
在没有主机名查找的情况下显示 LISTEN 状态下的所有 SOCKETS	**# ss -ln**
显示在 LISTEN 状态下对 IPv4 进行主机名查找的所有 SOCKETS	**# ss -ln4**
显示所有位于 LISTEN 状态下 IPv6 的 SOCKETS	**# ss -ln6**

3.2. IP

使用这些命令显示 IP L2 和 L3 配置、驱动程序、PCI 总线和网络统计信息。

操作	命令
显示所有 L2（物理和虚拟）接口及其统计数据	**# ip -s link show**
显示所有 L3 接口及其统计数据	**# ip -s addr show**
显示默认（主）IP 路由表	**# IP route show**
显示给定路由表的路由规则	**# IP route show table external**
显示所有路由表	**# IP rule show**
显示给定目的地的路由规则	**# IP route get 1.1.1.1**
显示所有 Linux 命名空间	**# IP netns show**
登录到 Linux 命名空间	**# ip netns exec ns0 bash**
显示给定接口的详细网络接口计数器	**# tail /sys/class/net/ens6/statistics/***
显示给定绑定设备的详细绑定信息	**# cat /proc/net/bonding/bond1**
显示全局网络接口计数器视图	**# cat /proc/net/dev**
显示物理连接类型（TP、FIBER 等），为给定网络接口支持并连接链接速度模式	**# ethtool ens6**
显示给定网络接口的 Linux 驱动程序、驱动程序版本、固件和 PCIe BUS ID	**# ethtool -i ens6**
显示给定网络接口的 default、启用和禁用的硬件卸载	**# ethtool -k ens6**
显示给定网络接口的 MQ（多队列）配置	**# ethtool -l ens6**
为给定网络接口更改 RX 和 TX 的 MQ 设置	**# ethtool -L ens6 combined 8**
仅对给定网络接口的 TX 更改 MQ 设置	**# ethtool -L ens6 tx 8**
显示给定网络接口的队列大小	**# ethtool -g ens6**
更改给定网络接口的 RX 队列大小	**# ethtool -G ens6 rx 4096**

操作	命令
显示增强的网络统计数据	**# cat /proc/net/softnet_stat**
显示快速重要的网络设备信息（Interface name、MAC、NUMA、PCIe 插槽、固件、内核驱动程序）	**# biosdevname -d**
显示内核内部丢弃计数器。如需更多信息，请参阅：监控网络数据处理。	**# cat /proc/net/softnet_stat**

3.3. OVS

使用这些命令显示 Open vSwitch 相关信息。

操作	命令
OVS DPDK 人类可读的统计	请参阅 Open vSwitch DPDK 统计。
显示 OVS 基本信息（version、dpdk enabled、PMD 内核、Lcore、ODL 网桥映射、平衡、自动平衡等）	**# OVS-vsctl list Open_vSwitch**
显示 OVS 全局切换视图	**# OVS-vsctl show**
显示 OVS 所有详细的接口	**# OVS-vsctl list interface**
显示一个接口的 OVS 详情（链接速度、MAC、状态、统计等）	**# ovs-vsctl list interface dpdk0**
显示给定接口的 OVS 计数器	**# ovs-vsctl get interface dpdk0 statistics**
显示 OVS 所有详细端口	**# OVS-vsctl list port**
显示一个端口的 OVS 详情（链接速度、MAC、状态、统计等）	**# OVS-vsctl list port vhu3gf0442-00**
显示一个网桥的 OVS 详情（数据路径类型、多播侦听、停止工作等）	**# OVS-vsctl list bridge br-int**
显示 OVS 日志状态	**# ovs-appctl vlog/list**
将所有 OVS 日志更改为 debug	**# ovs-appctl vlog/set dbg**
将一个特定的 OVS 子系统更改为文件日志输出的调试模式	**# OVS-appctl vlog/set file:backtrace:dbg**

操作	命令
禁用所有 OVS 日志	**# ovs-appctl vlog/set off**
将所有 OVS 子系统更改为仅对文件日志输出进行调试	**# OVS-appctl vlog/set file:dbg**
显示所有 OVS advanced 命令	**# OVS-appctl list-commands**
显示所有 OVS 绑定	**# ovs-appctl bond/list**
显示特定 OVS 绑定的详情（状态、绑定模式、转发模式、LACP 状态、绑定成员、绑定成员状态、链接状态）	**# ovs-appctl bond/show bond1**
显示成员、绑定和合作伙伴切换的高级 LACP 信息	**# ovs-appctl lacp/show**
显示 OVS 接口计数器	**# ovs-appctl dpctl/show -s**
显示 OVS 接口计数器突出显示迭代之间的区别	**# watch -d -n1 "ovs-appctl dpctl/show -s|grep -A4 -E '(dpdk|dpdkvhostuser)'|grep -v '\-\-'"**
显示给定端口的 OVS mempool 信息	**# ovs-appctl netdev-dpdk/get-mempool-info dpdk0**
显示 PMD 性能统计	**# ovs-appctl dpif-netdev/pmd-stats-show**
以一致的方式显示 PMD 性能统计	**# OVS-appctl dpif-netdev/pmd-stats-clear && ovs-appctl dpif-grafana/pmd-stats-show**
显示 DPDK 接口统计数据人类可读的	**# OVS-vsctl get interface dpdk0 statistics|sed -e "s/,/\n/g" -e "s/[\",\},]//g" -e "s/=/ =0286 /g"**
显示端口/队列和 PMD 线程之间的 OVS 映射	**# ovs-appctl dpif-netdev/pmd-rxq-show**
触发 OVS PMD 重新平衡（基于 PMD 周期利用率）	**# ovs-appctl dpif-netdev/pmd-rxq-rebalance**
在 OVS 端口和特定 PMD 之间创建关联性（从任何平衡中禁用 PMD）	**# ovs-vsctl set interface dpdk other_config:pmd-rxq-affinity="0:2,1:4"**
（OVS 2.11+ 和 FDP18.09）根据周期设置 PMD 平衡	**# OVS-vsctl set Open_vSwitch . other_config:pmd-rxq-assign=cycles**
（OVS 2.11+ 和 FDP18.09），以轮循方式设定 PMD 平衡	**# OVS-vsctl set Open_vSwitch . other_config:pmd-rxq-assign=roundrobin**
设置 OVS-DPDK 物理端口队列的数量	**# ovs-vsctl set interface dpdk options:n_rxq=2**

操作	命令
设置 OVS-DPDK 物理端口队列大小	# ovs-vsctl set Interface dpdk0 options:n_rxq_desc=4096 # ovs-vsctl set Interface dpdk0 options:n_txq_desc=4096
显示 OVS MAC 地址表（用于 action=normal）	# OVS-appctl fdb/show br-provider
设置 OVS vSwitch MAC Address 表过期时间（默认 300s）	# OVS-vsctl set bridge br-provider other_config:mac-aging-time=900
设置 OVS vSwitch MAC Address 表大小（默认 2048s）	# OVS-vsctl set bridge br-provider other_config:mac-table-size=204800
显示 OVS 数据路径流（内核空间）	# ovs-dpctl dump-flows -m
显示 OVS 数据路径流(dpdk)	# OVS-appctl dpif/dump-flows -m br-provider
显示数据路径流端口号和端口号之间的映射	# ovs-dpctl show
显示给定网桥中的 OVS OpenFlow 规则	# OVS-ofctl dump-flows br-provider
显示 OpenFlow 流端口号和端口号之间的映射	# OVS-ofctl show br-provider
(OVS 2.11+)- 启用自动重新平衡	# ovs-vsctl set Open_vSwitch . other_config:pmd-auto-lb="true"
(OVS 2.11+)- 将自动重新平衡间隔改为不同的值（默认值 1 分钟）	# ovs-vsctl set Open_vSwitch . other_config:pmd-auto-lb-rebalance-intvl="5"
详细的 OVS 内部配置	# man ovs-vswitchd.conf.db
要下载 OVS tcpdump	# curl -O -L ovs-tcpdump.in
从 DPDK 接口执行数据包捕获	# OVS-tcpdump.py --db-sock unix:/var/run/openvswitch/db.sock -i <bond/vhu> <tcpdump 标准参数，如 -v -n -e -w <path/to/file>
(OVS 2.10+)详细的 PMD 性能统计	# ovs-appctl dpif-netdev/pmd-perf-show

3.4. IRQ

使用这些命令显示中断请求行(IRQ)软件和硬件中断。

操作	命令
显示 ksoftirqd worker 执行的每个 CPU 的 SoftIRQ 平衡	**# cat /proc/softirqs | less -S**
显示 ksoftirqd worker 每 CPU 每秒执行的 SoftIRQ 平衡	**# watch -n1 -d -t "cat /proc/softirqs"**
显示每个 CPU 的硬件和软件中断（NMI、LOC、TLB、RSE、PII、PIW）平衡	**# cat /proc/interrupts | less -S**
显示每个 CPU 的硬件和软件中断（NMI、LOC、TLB、RSE、PII、PIW）平衡	**# watch -n1 -d -t "cat /proc/interrupts"**
显示计时器中断	**# cat /proc/interrupts | grep -E "LOC|CPU" | less -S**
显示每秒中断的计时器	**# watch -n1 -d -t "cat /proc/interrupts | grep -E 'LOC|CPU'"**
显示默认 IRQ CPU 关联性	**# cat /proc/irq/default_smp_affinity**
显示给定 IRQ(CPUMask)的 IRQ 关联性	**# cat /proc/irq/89/smp_affinity**
显示给定 IRQ(DEC)的 IRQ 关联性	**# cat /proc/irq/89/smp_affinity_list**
为给定的 IRQ(CPUMask)设置 IRQ 关联性	**# echo -n 1000 > /proc/irq/89/smp_affinity**
为给定的 IRQ(DEC)设置 IRQ 关联性	**# echo -n 12 > /proc/irq/89/smp_affinity_list**
显示硬件中断 CPU 关联性	**# tuna --show_irqs**
为给定的 IRQ 设置 IRQ 关联性（DEC 支持范围，例如 0-4 表示从 0 到 4）	**# tuna --irqs=<IRQ> --cpus=<CPU> --move**
显示 IRQ CPU 使用率分发	**# mpstat -I CPU | less -S**
显示给定 CPU 的 IRQ CPU 使用率分发	**# mpstat -I CPU -P 4 | less -S**
显示 SoftIRQ CPU 使用率分布	**# mpstat -I SCPU | less -S**
显示给定 CPU 的 SoftIRQ CPU 使用率分布	**# mpstat -I SCPU -P 4 | less -S**

3.5. PROCESS

使用这些命令显示 Linux 中的进程和线程、进程调度程序和 CPU 关联性。

操作	命令
显示给定进程名称分布 CPU 用量和 CPU 关联性，包括所有进程线程	**# pidstat -p $(pidof qemu-kvm) -t**
显示给定进程名称分布 CPU 用量和 CPU 关联性，包括所有进程线程（共 10 秒）的 30 迭代	**# pidstat -p $(pidof qemu-kvm) -t 10 30**
显示给定进程名称页面错误和内存使用情况，包括所有进程线程	**# pidstat -p $(pidof qemu-kvm) -t -r**
显示给定进程名称 I/O 统计信息，包括所有进程线程	**# pidstat -p $(pidof qemu-kvm) -t -d**
显示给定进程及其 PID、所有子 PID（包括进程名称）以及 CPU 时间	**# ps -T -C qemu-kvm**
显示给定进程以及所有子 PID 的实时性能统计	**# top -H -p $(pidof qemu-kvm)**
显示所有带有进程调度程序类型、优先级、命令、CPU 关联性和上下文交换信息的系统线程	**# tuna --show_threads**
为指定 PID RealTime(FIFO)调度设置，具有最高优先级	**# tuna --threads=<PID> --priority=FIFO:99**
显示 PMD 和 CPU 线程重新调度活动	**# watch -n1 -d "grep -E 'pmd|CPU' /proc/sched_debug"**
浏览器调度程序内部操作统计	**# less /proc/sched_debug**
显示全面的进程统计和关联性视图： 1. 打开 top，然后按"zbEEH"。 2. 按"f"，并查找"P = Last Used Cpu(SMP)"。 3. 使用"箭头右"选择它。 4. 在使用"浏览"前将其移动至 CPU 使用量。 5. 使用"箭头左"取消选择它。 6. 使用"d"启用它。 7. 使用 "<" 按 CPU 号排序。	**# top**
显示所有系统进程及其 CPU 关联性	**# ps -eF**
显示所有显示睡眠和正在运行的进程的系统进程，并在睡眠后处于哪个功能	**# ps -elfL**
显示给定 PID 的 CPU 关联性	**# taskset --pid $(pidof qemu-kvm)**

操作	命令
为指定 PID 设置 CPU 关联性	# taskset --pid --cpu-list 0-9,20-29 $(pidof <Process>)

3.6. KVM

使用这些命令显示基于内核的虚拟机(KVM)相关的域统计信息。

操作	命令
显示实时 KVM 管理程序统计信息（VMExit、VMEntry、vCPU wakeup、上下文切换、计时器、Halt Pool、vIRQ）	# kvm_stat
显示深度 KVM 管理程序统计信息	# kvm_stat --once
显示给定客户机的实时 KVM 管理程序统计信息（VMExit、VMEntry、vCPU wakeup、上下文切换、计时器、Halt Pool、vIRQ）	# kvm_stat --guest=<VM name>
显示给定虚拟机的深度 KVM 管理程序统计信息	# kvm_stat --once --guest=<VM name>
显示 KVM 分析捕获统计	# perf kvm stat live
显示 KVM 分析统计	# perf kvm top
显示给定虚拟机的 vCPU 固定	# virsh vcpupin <Domain name/ID>
显示给定虚拟机的 QEMU 模拟器 Thread	# virsh emulatorpin <Domain name/ID>
显示给定虚拟机的 NUMA 固定	# virsh numatune <Domain name/ID>
显示给定虚拟机的内存统计信息	# virsh dommemstat <Domain name/ID>
显示给定虚拟机的 vCPU 统计信息	# virsh nodecpustats <Domain name/ID>
显示给定虚拟机的所有 vNIC	# virsh domiflist <Domain name/ID>
显示给定虚拟机的 vNIC 统计信息（不使用 DPDK VHU）	# virsh domifstat <Domain name/ID> <vNIC>
显示给定虚拟机的所有 vDisk	# virsh domblklist <Domain name/ID>
显示给定虚拟机的 vDisk 统计信息	# virsh domblkstat <Domain name/ID> <vDisk>

操作	命令
显示给定虚拟机的所有统计	# virsh domstats <Domain name/ID>

3.7. CPU

使用这些命令显示 CPU 使用率、进程 CPU 分发、频率和 SMI。

操作	命令
显示给定进程名称分布 CPU 用量和 CPU 关联性，包括所有进程线程	# pidstat -p $(pidof qemu-kvm) -t
显示虚拟内存、I/O 和 CPU 统计信息	# vmstat 1
显示详细的 CPU 用量聚合	# mpstat
显示详细的 CPU 使用分布	# mpstat -P ALL
显示给定 CPU 的详细 CPU 使用分布（不支持范围）	# mpstat -P 2,3,4,5
为 30 迭代显示给定 CPU 的详细 CPU 使用分布时间（10 秒）	# mpstat -P 2,3,4,5 10 30
显示给定 CPU 频率的硬件限制和频率策略	# cpupower -c 24 frequency-info
显示当前 CPU 频率信息	# cpupower -c all frequency-info|grep -E "current CPU frequency|analyzing CPU"
显示所有 CPU 的频率和 CPU % C-States 统计数据	# cpupower monitor
显示所有 CPU 的实时频率和 CPU % C-States 统计突出显示任何变化	# watch -n1 -d "cpupower monitor"
显示所有 CPU 的详细频率和 CPU % C-States 统计信息，包括 SMI（对于 RT 很有用）	# turbostat --interval 1
显示给定 CPU 的更多详细信息和 CPU % C-States 统计信息，包括 SMI（对于 RT 很有用）	# turbostat --interval 1 --cpu 4
显示 CPU 详情和 ISA 支持	# lscpu

操作	命令
具体用于 Intel CPU： 显示 CPU 使用率、CPU IPC、CPU Execution(%)、L3 和 L2 Cache Hit、Miss、Miss、Miss、Temperature、内存频道使用和 QPI/UPI 使用情况的非常低级的详细信息	git clone Processor Counter Monitor make ./pcm.x"

3.8. NUMA

使用这些命令显示 Non-Uniform Memory Access(NUMA)统计和进程分布。

操作	命令
显示硬件 NUMA 拓扑	# numactl -H
显示 NUMA 统计数据	# numastat -n
显示 meminfo，如系统范围内存用量	# numastat -m
显示给定进程名称的 NUMA 内存详情和平衡	# numastat qemu-kvm
显示给定 NUMA 节点特定统计	# /sys/devices/system/node/node<NUMA 节点数>/numastat
显示为什么 NUMA 节点和 PCI 设备的 NUMA 拓扑	# lstopo --physical
使用相关设备生成物理 NUMA 拓扑的图形（svg 格式）	# lstopo --physical --output-format svg > topology.svg

3.9. 内存

使用这些命令显示内存统计信息、巨页、DPC、物理 DIMM 和频率。

操作	命令
显示 meminfo，如系统范围内存用量	# numastat -m
显示虚拟内存、I/O 和 CPU 统计信息	# vmstat 1

操作	命令
显示全局内存信息	# cat /proc/meminfo
显示给定 NUMA 节点的 2MB 巨页总数	# /sys/devices/system/node/node<NUMA node number>/hugepages/hugepages-2048kB/nr_hugepages
显示给定 NUMA 节点的 1GB 巨页总数	# /sys/devices/system/node/node<NUMA node number>/hugepages/hugepages-1048576kB/nr_hugepages
显示给定 NUMA 节点的总可用 2MB 巨页	# /sys/devices/system/node/node<NUMA node number>/hugepages/hugepages-2048kB/free_hugepages
显示给定 NUMA 节点的总可用 1GB 巨页	# /sys/devices/system/node/node<NUMA node number>/hugepages/hugepages-1048576kB/free_hugepages
实时分配 100x 2MB 巨页到 NUMA0（可以更改 NUMA 节点）	# echo 100 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
实时分配 100x 1GB 巨页到 NUMA0（可以更改 NUMA 节点）	# echo 100 > /sys/devices/system/node/node0/hugepages/hugepages-1048576kB/nr_hugepages
显示实时 SLAB 信息	# slabtop
显示详细的 SLAB 信息	# cat /proc/slabinfo
显示总安装内存 DIMM	# dmidecode -t memory | grep viewer
显示已安装的内存 DIMM Speed	# dmidecode -t memory | grep Speed

3.10. PCI

使用这些命令显示 PCI 统计信息、PCI 详细信息和 PCI 驱动程序覆盖。

操作	命令
显示系统中详细的 PCI 设备信息	# lspci -vvvnn
显示 PCI 树视图	# lspci -vnnt

操作	命令
显示 PCI 设备 NUMA 信息	# lspci -vmm
显示给定设备的最大链接速度	# lspci -s 81:00.0 -vv | grep LnkCap
显示给定设备的 PCIe 链接速度状态	# lspci -s 81:00.0 -vv | grep LnkSta
显示 PCI 设备和内核驱动程序	# driverctl list-devices
显示 PCI 设备驱动程序覆盖（用于 DPDK 和 SR-IOV 接口）	# driverctl list-overrides
为 PCI 设备（重新引导持久）设置不同的内核驱动程序	# driverctl set-override 0000:81:00.0 vfio-pci
取消设置 PCI 设备的覆盖内核驱动程序（如果设备正在使用命令，则挂起）	# driverctl unset-override 0000:81:00.0

3.11. TUNED

使用这些命令显示 tuned 配置集、验证和日志。

操作	命令
显示 tuned 当前启用的配置集和描述	# tuned-adm profile_info
显示 tuned 可用配置集和当前启用的配置集	# tuned-adm list
启用特定的 tuned 配置集	# tuned-adm profile realtime-virtual-host
验证当前启用的配置集	# tuned-adm verify
tuned 的日志	# less /var/log/tuned/tuned.log

3.12. 分析过程

使用这些命令显示 CPU 分析、进程性能分析和 KVM 分析。

节	操作	命令
Process	特定 PID 的性能分析	**# perf record -F 99 -p PID**
Process	对具体 PID 进行性能分析（30 秒）	**# perf record -F 99 -p PID sleep 30**
Process	根据特定 PID 分析实时分析	**# perf top -F 99 -p PID**
CPU	对任何事件的特定 CPU 核心列表进行性能分析（30 秒）	**# perf record -F 99 -g -C <CPU Core(s)>10.10.10.2-sleepsleep 30s**
CPU	分析任何事件的特定 CPU 核心列表的实时信息	**# perf top -F 99 -g -C <CPU Core(s)>**
上下文切换	对具体 CPU 核心列表进行性能分析（30 秒），并只查找上下文切换	**# perf record -F 99 -g -e sched:sched_switch -C <CPU Core(s)> – sleep 30**
KVM	为给定时间分析 KVM 客户机	**# perf kvm stat record sleep 30s**
Cache	为特定 CPU 核心列表进行性能分析，以 5 秒查找缓存效率	**# perf stat -C <CPU Core(s)> -B -e cache-references,cache-misses,cycles,instructions,branches,faults,migrations sleep 5**
Report	分析 perf 分析	**# perf report**
Report	在 stdout 中报告 perf 分析	**# perf report --stdio**
Report	在 stdout 中报告 KVM 分析	**# perf kvm stat 报告**

3.13. 块 I/O

使用这些命令显示存储 I/O 分发和 I/O 分析。

操作	命令
显示所有系统设备的 I/O 详情	**# iostat**
显示所有系统设备的高级 I/O 详情	**# iostat -x**
显示所有系统设备的高级 I/O 详情，每 10 秒显示 30 次迭代	**# iostat -x 10 30**

操作	命令
为给定块设备生成高级 I/O 分析	# blktrace -d /dev/sda -w 10 && blkparse -i sda.* -d sda.bin
报告 blktrace 分析	# btt -i sda.bin

3.14. REAL TIME

使用这些命令显示 Real Time 测试相关、SMI 和延迟。

操作	命令
识别是否存在任何 SMI 是否阻止普通 RT 内核执行，以推断定义的阈值。	# hwlatdetect --duration=3600 --threshold=25
使用很多附加选项验证给定时间的最大调度延迟： --duration 指定测试运行的时间值。 --mlockall 锁定当前和将来的内存分配。 --priority 设置第一个线程的优先级。 --nanosleep 使用 clock_nanosleep 而不是 posix 间隔计时器。 --interval 在 microseconds 中设置线程的基本间隔。 --histogram 在运行后，转储直方图到 stdout 的延迟。 --histfile 将直方图的延迟转储到 &lt ;path>，而不是 stdout。 --threads 设置测试线程的数量。 --numa 标准 NUMA 测试。 --notrace 抑制追踪。	# cyclictest --duration=3600 \ --mlockall \ --priority=99 \ --nanosleep \ --interval=200 \ --histogram=5000 \ --histfile=./output \ --threads \ --numa \ --notrace

3.15. 安全性

使用这些命令验证执行和 GRUB 引导参数。

操作	命令
检查所有当前的 Speculative 执行安全状态	请参阅： 适用于 Linux 和 BSD 的 Spectre & Meltdown 漏洞/提交检查器。
GRUB 参数禁用所有 Speculative Execution 补救	**spectre_v2=off spec_store_bypass_disable=off pti=off l1tf=off kvm-intel.vmentry_l1d_flush=never**
验证 CVE-2017-5753（Spectre 变体 1）状态	**# cat /sys/devices/system/cpu/vulnerabilities/spectre_v1**
验证 IBPB 和 Retpoline（CVE-2017-5715 Spectre 变体 2 状态	**# cat /sys/devices/system/cpu/vulnerabilities/spectre_v2**
验证 KPTI(CVE-2017-5754 Meltdown)状态	**# cat /sys/devices/system/cpu/vulnerabilities/meltdown**
验证 Spectre-NG(CVE-2018-3639 Spectre Variant 4)状态	**# cat /sys/devices/system/cpu/vulnerabilities/spec_store_bypass**
验证 Foreshadow（CVE-2018-3615 Spectre Varian 5 也称为 L1TF）状态	**# cat /sys/devices/system/cpu/vulnerabilities/l1tf**
验证 Foreshadow VMEntry L1 缓存效果	**# cat /sys/module/kvm_intel/parameters/vmentry_l1d_flush**
验证 SMT 状态	**# cat /sys/devices/system/cpu/smt/control**

3.16. JUNIPER CONTRAIL VROUTER

使用这些命令显示 vRouter VIF、MPLS、Nexthost、VRF、VRF 的路由、流和转储信息。

操作	命令
vRouter Kernel space human 可读取统计	请参阅： 显示 Contrail vRouter 统计信息。
vRouter DPDK 人类可读的统计	请参阅： 显示 Contrail vRouter 统计信息。
要从 DPDK 接口执行数据包捕获（不要在 vifdump 之后使用 grep）	**# vifdump vif0/234 <tcpdump 标准参数，如 -v -nn -e -w <path/to/file>>**

操作	命令
显示所有 vRouter 接口和子接口统计和详情	**# VIF --list**
显示给定接口的 vRouter 统计和详情	**# vif --list --get 234**
显示所有接口和子接口的 vRouter packer 速率	**# VIF --list --rate**
显示给定接口的 vRouter packer 速率	**# VIF --list --rate --get 234**
显示给定接口的 vRouter 数据包丢弃统计	**# VIF --list --get 234 --get-drop-stats**
显示 vRouter 流	**# flow -l**
显示实时 vRouter 流操作	**# flow -r**
显示给定 VRF 的 vRouter 数据包统计信息（您可以从 vif --list 中找到 VRF 号）	**# vrfstats --get 0**
显示所有 VRF 的 vRouter 数据包统计信息	**# vrfstats --dump**
显示给定 VRF 的 vRouter 路由表（您可以从 vif --list 中找到 VRF 号）	**# rt --dump 0**
显示给定 VRF 的 vRouter IPv4 路由表（您可以从 vif --list 中找到 VRF 号）	**# rt --dump 0 --family inet**
显示给定 VRF 的 vRouter IPv6 路由表（您可以从 vif --list 中找到 VRF 号）	**# rt --dump 0 --family inet6**
显示给定 VRF 的 vRouter 转发表（您可以从 vif --list 中找到 VRF 号）	**# rt --dump 0 --family bridge**
在给定地址的 VRF 中显示 vRouter 路由目标	**# rt --get 0.0.0.0/0 --vrf 0 --family inet**
显示 vRouter 丢弃统计	**# dropstats**
显示给定 DPDK 内核的 vRouter 丢弃统计	**# dropstats --core 11**
显示 vRouter MPLS 标签	**# MPLS --dump**
显示给定一个 vRouter nexthop（可以通过 mpls --dump 输出找到）	**# nh --get 21**
显示所有 vRouter nexthops	**# nh --list**
显示所有 vRouter VXLAN VNID	**# VXLAN --dump**

操作	命令
显示 vRouter 代理（supervisor、xmmp connection、vrouter 代理等）状态	# contrail-status
重启 vRouter（以及所有 Contrail 本地计算节点组件）	# systemctl restart supervisor-vrouter

有关 Juniper Contrail vRouter CLI utilitlies 的更多信息，请参阅以下文档：

- Juniper Contrail 3.2 文档
- Juniper Contrail 4.0 文档
- Juniper Contrail 4.1 文档
- Juniper Contrail 5.0 文档

3.17. 容器

它们是部分常用的 Docker 和 Podman 命令。

操作	Docker RHEL7	podman RHEL8
显示所有正在运行的容器	# Docker ps	# podman ps
显示所有容器（运行、停止等）	# docker ps -a	# podman ps -a
显示所有容器（运行、停止等），而不截断输出	# docker ps -a --no-trunc	# podman ps -a --no-trunc
显示所有容器（运行、停止等）json 输出	# docker ps --format '{{ json .}}' | jq -C '.' s|# podman ps -a --format json	jq -C '.'
显示给定容器的容器进程树	# Docker top <container ID>	# podman pod top <container ID>

操作	Docker RHEL7	podman RHEL8
显示实时容器资源使用率（CPU、内存、I/O、Net） - TOP	# Docker stats	# podman stats
显示给定容器的实时资源利用率（CPU、内存、I/O、Net） - TOP	# Docker stats <container ID>	# podman stats <container ID>
登录到给定的正在运行的容器	# docker exec -it <container ID> /bin/bash	# podman exec -it <container ID> /bin/bash
以 root 用户身份登录给定正在运行的容器	# docker exec -u root -it <container ID> /bin/bash	# podman exec -u root -it <container ID> /bin/bash
显示给定容器中的端口映射	# Docker 端口 <container ID>	# podman port <container ID>
显示名称、ID 和标签的所有本地存储的镜像	# Docker image ls # docker images"	# podman image ls # podman images"
显示给定镜像的历史记录	# Docker history <image id>	# podman history <image id>
显示给定容器的低级别配置	# Docker inspect <container ID>	# podman inspect <container ID>
显示给定容器的所有卷	# docker inspect -f "{{ .Mounts }}" <container ID>	# podman inspect -f "{{ .Mounts }}" <container ID>
重启具有相同模式的所有容器	# docker ps -q --filter "name=swift" | xargs -n1 docker restart	# podman ps -q --filter "name=swift" | xargs -n1 docker restart

有关 docker 或 podman 的更多信息，请参阅以下文档：

- Docker 命令参考
- podman 命令参考

3.18. OPENSTACK

使用这些 OpenStack 命令显示虚拟机计算节点。

操作	命令
显示根据计算节点排序的计算节点上所有虚拟机的列表	$ Nova list --fields name,OS-EXT-SRV-ATTR:host --sort host
显示按照虚拟机名称排序的计算节点上所有虚拟机的列表	$ Nova list --fields name,OS-EXT-SRV-ATTR:host

第 4 章 实例 TAP 接口的 TX 队列中的高数据包丢失

使用本节对 TX 队列中的数据包丢失进行故障排除，而不是 OVS-DPDK。

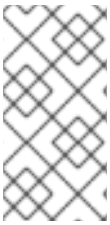
4.1. 症状

在使用仅主机联网的虚拟功能(VNF)测试过程中，可以在实例的 tap 接口的 TX 队列中观察高数据包丢失。测试设置会将一个节点上的一个虚拟机的数据包发送到同一节点上的另一虚拟机。数据包丢失。

以下示例显示了 tap 的 TX 队列中丢弃的大量数据包。

```
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500034259301 132047795 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
               0        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5481296464 81741449 0       11155280 0       0
    TX errors: aborted  fifo   window heartbeat transns
          0        0       0       0       0
```

4.2. 诊断



注意

本节检查数据包丢弃（内核路径）接口。有关用户数据路径中的 vhost 用户界面中的数据包丢弃，请参考 https://access.redhat.com/solutions/3381011

TX 丢弃是因为实例 vCPU 和虚拟机监控程序上其他进程之间的干扰而造成的。tap 接口的 TX 队列是一个缓冲区，可以在实例无法获取数据包时存储简短的数据包。如果实例的 CPU 阻止运行（或冻结）长时间，会出现这种情况。

TUN/TAP 设备是一个虚拟设备，一个末尾是一个内核网络接口，另一个端是用户空间文件描述符。

TUN/TAP 接口可在两种模式下运行：

- TAP 模式使用 L2 标头将 L2 以太网帧馈送到设备中，并期望从用户空间接收相同的数据。这个模式用于虚拟机。

- tun 模式使用 L3 标头将 L3 IP 数据包馈到该设备中，并期望从用户空间接收相同的数据包。这个模式主要用于 VPN 客户端。

在 KVM 网络中，用户空间文件描述符归 qemu-kvm 进程所有。发送到 tap 的帧（从 hypervisor 的视角到TX），最终在 qemu-kvm 内作为 L2 帧，然后将这些帧作为收到到虚拟网络接口的网络数据包（从虚拟机的视角来看）将那些帧发送到虚拟机中的虚拟网络设备。

使用 TUN/TAP 的重要概念是管理程序中的传输方向是虚拟机的接收方向。相反方向也是如此；从虚拟机传输虚拟机监控程序的接收是相同的。

virtio-net 设备上没有数据包的"ring buffer"。这意味着，如果 TUN/TAP 设备的 TX 队列被填满，因为虚拟机未接收（足够快或根本），那么新数据包都没有什么位置，并且虚拟机监控程序看到利用的 TX 丢失。

如果您注意到 TUN/TAP 上的 TX 丢失，增加 tap txqueuelen 以避免，类似增加 RX 环缓冲，以停止物理 NIC 丢失。

但是，假设虚拟机在接收中只是"slow"和"bursty"。如果虚拟机没有足够快，或者根本没有收到接收，则调整 TX 队列长度不会帮助。您必须发现为什么虚拟机没有运行或接收。

4.2.1. 临时解决方案

为了减少少量的延迟和其他缺点的成本，可以增加 TX 队列。

要临时增加 txqueuelen，请使用以下命令：

```
/sbin/ip link set tap<uuid> txqueuelen <new queue length>
```

4.2.2. 诊断步骤

使用以下脚本查看 CPU 时间与管理程序影响。

```
[root@ibm-x3550m4-9 ~]# cat generate-tx-drops.sh
#!/bin/bash

trap 'cleanup' INT

cleanup() {
  echo "Cleanup ..."
  if [ "x$HPING_PID" != "x" ]; then
    echo "Killing hping3 with PID $HPING_PID"
    kill $HPING_PID
  fi
  if [ "x$DD_PID" != "x" ]; then
    echo "Killing dd with PID $DD_PID"
    kill $DD_PID
  fi
  exit 0
}

VM_IP=10.0.0.20
VM_TAP=tapc18eb09e-01
VM_INSTANCE_ID=instance-00000012
LAST_CPU=$( lscpu | awk '/^CPU\(s\):/ { print $NF - 1 }' )
# this is a 12 core system, we are sending everything to CPU 11,
# so the taskset mask is 800 so set dd affinity only for last CPU
TASKSET_MASK=800

# pinning vCPU to last pCPU
echo "virsh vcpupin $VM_INSTANCE_ID 0 $LAST_CPU"
virsh vcpupin $VM_INSTANCE_ID 0 $LAST_CPU

# make sure that: nova secgroup-add-rule default udp 1 65535 0.0.0.0/0
# make sure that: nova secgroup-add-rule default tcp 1 65535 0.0.0.0/0
# make sure that: nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
# --fast, --faster or --flood can also be used
echo "hping3 -u -p 5000 $VM_IP --faster > /dev/null "
hping3 -u -p 5000 $VM_IP --faster > /dev/null &
HPING_PID=$!

echo "hping is running, but dd not yet:"
for i in { 1 .. 3 }; do
  date
  echo "ip -s -s link ls dev $VM_TAP"
  ip -s -s link ls dev $VM_TAP
  sleep 5
done

echo "Starting dd and pinning it to the same pCPU as the instance"
echo "dd if=/dev/zero of=/dev/null"
dd if=/dev/zero of=/dev/null &
DD_PID=$!
echo "taskset -p $TASKSET_MASK $DD_PID"
taskset -p $TASKSET_MASK $DD_PID

for i in { 1 .. 3 }; do
  date
  echo "ip -s -s link ls dev $VM_TAP"
```

```
  ip -s -s link ls dev $VM_TAP
  sleep 5
done

cleanup
```

登录实例，再启动 **dd if=/dev/zero of=/dev/null**，以在它只在 **vCPU** 上生成额外负载。请注意，这是用于演示目的。您可以在虚拟机内重复相同的测试，且无需加载。仅在虚拟机监控程序上的另一个进程从实例的 **vCPU** 传输时间时进行 **TX** 丢弃。

以下示例显示了测试前的一个实例：

```
%Cpu(s): 22.3 us, 77.7 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1884108 total,  1445636 free,    90536 used,   347936 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  1618720 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
30172 root      20   0  107936    620    528 R 99.9  0.0   0:05.89 dd
```

运行以下脚本，并观察 **TX** 队列中丢弃的软件包。仅当 **dd** 进程从实例的 **CPU** 占用大量处理时间时，才会发生这些情况。

```
[root@ibm-x3550m4-9 ~]# ./generate-tx-drops.sh
virsh vcpupin instance-00000012 0 11

hping3 -u -p 5000 10.0.0.20 --faster > /dev/null
hping is running, but dd not yet:
Tue Nov 29 12:28:22 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500034259301 132047795 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
               0        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5481296464 81741449 0       11155280 0       0
    TX errors: aborted  fifo   window heartbeat transns
               0        0       0       0       0
Tue Nov 29 12:28:27 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500055729011 132445382 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
               0        0       0       0       0
```

```
    TX: bytes  packets  errors  dropped carrier collsns
    5502766282 82139038 0       11155280 0       0
    TX errors: aborted  fifo   window heartbeat transns
             0       0       0       0       0
Tue Nov 29 12:28:32 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500077122125 132841551 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
             0       0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5524159396 82535207 0       11155280 0       0
    TX errors: aborted  fifo   window heartbeat transns
             0       0       0       0       0
Tue Nov 29 12:28:37 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500098181033 133231531 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
             0       0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5545218358 82925188 0       11155280 0       0
    TX errors: aborted  fifo   window heartbeat transns
             0       0       0       0       0
Tue Nov 29 12:28:42 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500119152685 133619793 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
             0       0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5566184804 83313451 0       11155280 0       0
    TX errors: aborted  fifo   window heartbeat transns
             0       0       0       0       0
Starting dd and pinning it to the same pCPU as the instance
dd if=/dev/zero of=/dev/null
taskset -p 800 8763
pid 8763's current affinity mask: fff
pid 8763's new affinity mask: 800
Tue Nov 29 12:28:47 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500140267091 134010698 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
```

```
        0       0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5587300452 83704477 0       11155280 0       0
    TX errors: aborted  fifo   window heartbeat transns
        0       0       0       0       0
Tue Nov 29 12:28:52 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500159822749 134372711 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
        0       0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5606853168 84066563 0       11188074 0       0
    TX errors: aborted  fifo   window heartbeat transns
        0       0       0       0       0
Tue Nov 29 12:28:57 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500179161241 134730729 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
        0       0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5626179144 84424451 0       11223096 0       0
    TX errors: aborted  fifo   window heartbeat transns
        0       0       0       0       0
Tue Nov 29 12:29:02 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500198344463 135085948 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
        0       0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5645365410 84779752 0       11260740 0       0
    TX errors: aborted  fifo   window heartbeat transns
        0       0       0       0       0
Tue Nov 29 12:29:07 EST 2016
ip -s -s link ls dev tapc18eb09e-01
69: tapc18eb09e-01: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master qbrc18eb09e-01 state UNKNOWN mode DEFAULT qlen 1000
    link/ether fe:16:3e:a5:17:c0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    5500217014275 135431570 0       0       0       0
    RX errors: length   crc     frame   fifo    missed
        0       0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    5664031398 85125418 0       11302179 0       0
    TX errors: aborted  fifo   window heartbeat transns
```

```
          0       0       0       0       0
Cleanup ...
Killing hping3 with PID 8722
Killing dd with PID 8763
[root@ibm-x3550m4-9 ~]#
--- 10.0.0.20 hping statistic ---
3919615 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

以下示例显示了测试过程中 dd 对虚拟机监控程序的影响。st 标签标识虚拟机监控程序中被攻击的时间百分比。

```
%Cpu(s):  7.0 us, 27.5 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi, 20.2 si, 45.4 st
KiB Mem :  1884108 total,  1445484 free,    90676 used,   347948 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  1618568 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
30172 root      20   0  107936    620    528 R 54.3  0.0   1:00.50 dd
```

请注意，ssh 可在实例上进行第二一半测试时变得缓慢，包括如果测试用时过长，可能会超时。

4.3. 解决方案

增加 TX 队列有助于缓解这些小冻结，对 CPU 固定完全隔离，内核参数中的 isolcpus 是最佳解决方案。更加信息，请参阅在 OpenStack 中配置 CPU 固定与 OpenStack 中的 NUMA 固定。

第 5 章 使用 OPEN VSWITCH DPDK 的 INSTANCE VHU 接口上 TX DROPS

使用此流程对实例 vhost-user(VHU)接口传输的丢弃进行故障排除。

5.1. 症状

数据包通过 virtio 传输从 vswitch 传输到客户机，而不通过内核或 qemu 进程。这可以通过使用 VHU 接口交换数据包来完成。

VHU 主要由 DPDK librte_vhost 实施，其也提供要发送或接收数据包批量的功能。VHU 的后端是由 qemu 提供的 virtio 环形，用于与虚拟机交换数据包。virtio ring 具有特殊的格式，它由描述符和缓冲区组成。

TX/RX(transmit/receive)统计用于 OpenvSwitch(OVS)。这意味着传输统计与接收虚拟机的统计数据直接相关。

如果虚拟机没有足够快地处理数据包，OVS TX 队列溢出，并丢弃数据包。

5.1.1. Packet Drops 的说明

饱和 virtio 环会导致 TX 丢弃 vhost-user 设备。virtio ring 位于客户机的内存中，它与 vhost-user 推送数据包并消耗它们的队列一样。如果虚拟机无法足够快地使用数据包，virtio 环会从缓冲和 vhost-user 丢弃数据包中运行。

使用 Perf 和 Ftrace 工具对数据包丢弃进行故障排除。

- 使用 Perf 计算调度程序切换的数量，这可能会显示 qemu 线程是否被抢占。
- 使用 Ftrace 显示抢占的原因以及其所需的时间。

抢占原因包括：

- 时间中断（内核循环）：

这些添加至少两个上下文切换的成本。计时器中断也可以运行 read-copy update(RCU)回调，这些回调可能会花费无法预计的时间。

- CPU 电源管理和超线程

您可以在以下软件包中找到这些工具：

- PERF： 在 rhel-7-server-rpms/7Server/x86_64 中 perf rpm。如需更多信息，请参阅关于 Perf
- Fhandler： trace-cmd info rhel-7-server-rpms/7Server/x86_64.如需更多信息，请参阅关于 Ftrace

5.1.2. 其他丢弃的说明

在 OVS 2.9 之前，vHost 用户端口是在 dpdkvhostuser 模式中创建的。在这种模式中，OVS 充当 vhost 服务器，QEMU 充当客户端。当实例停机或重启时，OVS 网桥上的 vhost 用户端口仍然活跃，丢弃目的地为虚拟机的数据包。这会增加 tx_drop_counter ：

在以下示例中，虚拟机被 nova stop <UUID> 停止：

```
[root@overcloud-compute-0 network-scripts]# ovs-vsctl list interface vhubd172106-73 | grep _state
admin_state         : up
link_state          : down
```

当内核端口使用 ip link 设置 dev <br 内部端口名称> down 和 用户空间中丢弃 帧时，这个情况类似。

当虚拟机启动时，它会连接到同一 vhu socket，并将开始清空 virtio 环缓冲。TX 不再中断，正常的网络流量恢复。

5.1.3. 为 DPDK 增加 TX 和 RX 队列长度

您可以使用以下 OpenStack director 模板修改来更改 DPDK 的 TX 和 RX 队列长度：

```
NovaComputeExtraConfig:
      nova::compute::libvirt::rx_queue_size: '"1024"'
      nova::compute::libvirt::tx_queue_size: '"1024"'
```

以下示例显示了验证检查：

```
[root@overcloud-compute-1 ~]# ovs-vsctl get interface vhu9a9b0feb-2e status
{features="0x0000000150208182", mode=client, num_of_vrings="2", numa="0",
socket="/var/lib/vhost_sockets/vhu9a9b0feb-2e", status=connected, "vring_0_size"="1024",
"vring_1_size"="1024"}

[root@overcloud-compute-1 ~]# virsh dumpxml instance-00000017 | grep rx
<driver rx_queue_size='1024' tx_queue_size='1024'/>
<driver rx_queue_size='1024' tx_queue_size='1024'/>
```

由于内核限制，您无法将队列大小增加到 1024 以上。

注意

如果您计划通过 DPDK 对 neutron 网络使用 PXE 引导，则必须验证 PXE 版本是否支持 1024 字节。

5.2. 诊断

当客户机无法接收数据包时，可以看到 TX 丢弃到 vhost 用户端口。TCP 旨在恢复数据包丢失，这在普通网络条件中发生。NFVI 具有严格的要求，但对数据包丢弃的容错较少。

使用 DPDK 加速的 OVS，因为内核数据路径太慢。另外，部署启用了 DPDK 的客户机（可匹配主机的数据包处理速度）非常重要。

5.3. 解决方案

确保分配给虚拟机的 vCPU 只处理客户机的任务。

- 检查集群是否使用 heat 以下模板参数进行了部署：
  - IsolcpusList: 从调度中删除 CPU

- **NovaVcpuPinSet ：为固定分配 CPU**
- **NovaComputeCpuSharedSet: 为仿真程序线程固定分配 CPU**

例如：

```
  parameter_defaults:
    ComputeOvsDpdkParameters:
      KernelArgs: "default_hugepagesz=1GB hugepagesz=1G hugepages=32 iommu=pt
intel_iommu=on isolcpus=2-19,22-39"
      IsolCpusList: "2-19,22-39"
      NovaVcpuPinSet: ['4-19,24-39']
      NovaReservedHostMemory: 4096
      OvsDpdkSocketMemory: "3072,1024"
      OvsDpdkMemoryChannels: "4"
      OvsDpdkCoreList: "0,20,1,21"
      OvsPmdCoreList: "2,22,3,23"
      NovaComputeCpuSharedSet: [0,20,1,21]
```

- **确保虚拟机已部署了利用固定 CPU 和仿真器池集的类别。**

例如：

```
openstack flavor create --ram <size_mb> --disk <size_gb> -\
-vcpus <vcpus> --property dpdk=true \
--property hw:mem_page_size=1G \
--property hw:cpu_policy=dedicated \
--property hw:emulator_threads_policy=share <flavor>
```

- **确保这些设置按预期运行。如需更多信息，请参阅 简单 Compute 节点 CPU 分区和文件系统检查。**

如果您将完全专用 CPU 资源分配给实例，并仍然观察网络数据包丢失，请确保实例经过正确调整并启用了 DPDK。

第 6 章 使用 DPDK 在 OPEN VSWITCH 中解释 PMD-STATS-SHOW 命令的输出

使用本节来解释使用 DPDK 的 Open vSwitch(OVS)中的 pmd-stats-show 命令(ovs-appctl dpif-netdev/pmd-stats-show)的输出。

6.1. 症状

ovs-appctl dpif-netdev/pmd-stats-show 命令提供了一个不准确测量。这是因为从 PMD 开始图表到统计。

6.2. 诊断

要获得有用的输出，请将系统置于稳定状态并重置要测量的统计信息：

```
# put system into steady state
ovs-appctl dpif-netdev/pmd-stats-clear
# wait <x> seconds
sleep <x>
ovs-appctl dpif-netdev/pmd-stats-show
```

以下是输出结果的示例：

```
[root@overcloud-compute-0 ~]# ovs-appctl dpif-netdev/pmd-stats-clear && sleep 10 && ovs-appctl
dpif-netdev/pmd-stats-show |
egrep 'core_id (2|22):' -A9
pmd thread numa_id 0 core_id 22:
    emc hits:17461158
    megaflow hits:0
    avg. subtable lookups per hit:0.00
    miss:0
    lost:0
    polling cycles:4948219259 (25.81%)
    processing cycles:14220835107 (74.19%)
    avg cycles per packet: 1097.81 (19169054366/17461158)
    avg processing cycles per packet: 814.43 (14220835107/17461158)
--
pmd thread numa_id 0 core_id 2:
    emc hits:14874381
    megaflow hits:0
    avg. subtable lookups per hit:0.00
    miss:0
    lost:0
    polling cycles:5460724802 (29.10%)
```

```
    processing cycles:13305794333 (70.90%)
    avg cycles per packet: 1261.67 (18766519135/14874381)
    avg processing cycles per packet: 894.54 (13305794333/14874381)
```

请注意，core_id 2 主要忙碌，花费 70% 的时间处理以及时间轮询时间的 30%。

```
polling cycles:5460724802 (29.10%)
processing cycles:13305794333 (70.90%)
```

在本例中，错误表示 DPDK 数据路径中未分类的数据包（'emc' 或 'dp' 类器）。在正常情况下，它们将发送到 a proto 层。在个别情况下，由于流重新验证锁定，或者 a proto 层返回错误，数据包会被丢弃。在这种情况下，丢失 的值也会递增，以指示丢失。

```
emc hits:14874381
megaflow hits:0
avg. subtable lookups per hit:0.00
miss:0
lost:0
```

有关更多信息，请参阅 OVS-DPDK 数据路径分类器。

6.3. 解决方案

本节演示了使用 ovs-appctl 命令查看流量流的步骤。

6.3.1. idle PMD

以下示例显示了一个系统，其中 core_ids 为 dpdk0 固定为 dpdk0，它只会管理通过 dpdk0 的流量流：

```
[root@overcloud-compute-0 ~]# ovs-appctl dpif-netdev/pmd-stats-clear && sleep 10 && ovs-appctl
dpif-netdev/pmd-stats-show |
egrep 'core_id (2|22):' -A9
pmd thread numa_id 0 core_id 22:
    emc hits:0
    megaflow hits:0
    avg. subtable lookups per hit:0.00
    miss:0
    lost:0
    polling cycles:12613298746 (100.00%)
    processing cycles:0 (0.00%)
--
pmd thread numa_id 0 core_id 2:
    emc hits:5
```

```
    megaflow hits:0
    avg. subtable lookups per hit:0.00
    miss:0
    lost:0
    polling cycles:12480023709 (100.00%)
    processing cycles:14354 (0.00%)
    avg cycles per packet: 2496007612.60 (12480038063/5)
    avg processing cycles per packet: 2870.80 (14354/5)
```

6.3.2. 带有数据包丢弃的 load 测试中的 PMD

以下示例显示了一个系统，其中 core_ids 为 dpdk0 固定为 dpdk0，其负载测试流通过 dpdk0，从而导致了大量 RX 丢弃：

```
[root@overcloud-compute-0 ~]# ovs-appctl dpif-netdev/pmd-stats-clear && sleep 10 && ovs-appctl
dpif-netdev/pmd-stats-show |
egrep 'core_id (2|4|22|24):' -A9
pmd thread numa_id 0 core_id 22:
    emc hits:35497952
    megaflow hits:0
    avg. subtable lookups per hit:0.00
    miss:0
    lost:0
    polling cycles:1446658819 (6.61%)
    processing cycles:20453874401 (93.39%)
    avg cycles per packet: 616.95 (21900533220/35497952)
    avg processing cycles per packet: 576.20 (20453874401/35497952)
--
pmd thread numa_id 0 core_id 2:
    emc hits:30183582
    megaflow hits:0
    avg. subtable lookups per hit:0.00
    miss:2
    lost:0
    polling cycles:1497174615 (6.85%)
    processing cycles:20354613261 (93.15%)
    avg cycles per packet: 723.96 (21851787876/30183584)
    avg processing cycles per packet: 674.36 (20354613261/30183584)
```

数据包被丢弃时，您可以看到处理周期与轮询周期的高比率（超过 90% 的处理周期）：

```
polling cycles:1497174615 (6.85%)
processing cycles:20354613261 (93.15%)
```

检查每个数据包的平均周期(CPP)和每个数据包的平均处理周期(PCPP)。对于完全加载的 PMD，您可以预计 PCPP/CPP 比率为 1，因为没有空闲循环。

```
avg cycles per packet: 723.96 (21851787876/30183584)
avg processing cycles per packet: 674.36 (20354613261/30183584)
```

6.3.3. pmD 在 loadtest 下具有 50% 的 mpps 容量

以下示例显示了一个系统，其中 core_ids 为 dpdk0 固定于 dpdk0 的 PMD，其负载测试流过 dpdk0，发送 6.4 Mpps（大约大约 12.85 Mpps）：

```
[root@overcloud-compute-0 ~]# ovs-appctl dpif-netdev/pmd-stats-clear && sleep 10 && ovs-appctl
dpif-netdev/pmd-stats-show |
egrep 'core_id (2|4|22|24):' -A9
pmd thread numa_id 0 core_id 22:
    emc hits:17461158
    megaflow hits:0
    avg. subtable lookups per hit:0.00
    miss:0
    lost:0
    polling cycles:4948219259 (25.81%)
    processing cycles:14220835107 (74.19%)
    avg cycles per packet: 1097.81 (19169054366/17461158)
    avg processing cycles per packet: 814.43 (14220835107/17461158)
--
pmd thread numa_id 0 core_id 2:
    emc hits:14874381
    megaflow hits:0
    avg. subtable lookups per hit:0.00
    miss:0
    lost:0
    polling cycles:5460724802 (29.10%)
    processing cycles:13305794333 (70.90%)
    avg cycles per packet: 1261.67 (18766519135/14874381)
    avg processing cycles per packet: 894.54 (13305794333/14874381)
```

当 pps 为接口的最大值为一半时，您可以看到处理周期与轮询周期的比率较低（大约 70% 的处理周期）：

```
polling cycles:5460724802 (29.10%)
processing cycles:13305794333 (70.90%)
```

6.3.4. hit vs miss vs lost

以下示例显示了有关主题的 man page：

```
an ovs-vswitchd
(...)
  DPIF-NETDEV COMMANDS
     These commands are used to expose  internal  information  (mostly  statistics)
```

about the `dpif-netdev` userspace datapath. If there is only one datapath (as is often the case, unless dpctl/ commands are used), the dp argument can be omitted.

dpif-netdev/pmd-stats-show [dp]
Shows performance statistics for each pmd thread of the datapath dp. The special thread ``main'' sums up the statistics of every non pmd thread. The sum of ``emc hits'', ``masked hits'' and ``miss'' is the number of packets received by the datapath. Cycles are counted using the TSC or similar facilities when available on the platform. To reset these counters use dpif-netdev/pmd-stats-clear. The duration of one cycle depends on the measuring infrastructure.
(...)

Raw

man ovs-dpctl
(...)
dump-dps
Prints the name of each configured datapath on a separate line.

[-s | --statistics] show [dp...]
Prints a summary of configured datapaths, including their datapath numbers and a list of ports connected to each datapath. (The local port is identified as port 0.) If -s or --statistics is specified, then packet and byte counters are also printed for each port.

The datapath numbers consists of flow stats and mega flow mask stats.

The "lookups" row displays three stats related to flow lookup triggered by processing incoming packets in the datapath. "hit" displays number of packets matches existing flows. "missed" displays the number of packets not matching any existing flow and require user space processing.
"lost" displays number of packets destined for user space process but subsequently dropped before reaching userspace. The sum of "hit" and "miss" equals to the total number of packets datapath processed.
(...)

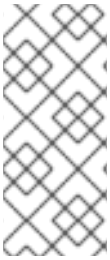
Raw

man ovs-vswitchd
(...)
dpctl/show [-s | --statistics] [dp...]
Prints a summary of configured datapaths, including their datapath numbers and a list of ports connected to each datapath. (The local port is identified as port 0.) If -s or --statistics is specified, then packet and byte counters are also printed for each port.

The datapath numbers consists of flow stats and mega flow mask stats.

The "lookups" row displays three stats related to flow lookup triggered by processing incoming packets in the datapath. "hit" displays number of packets matches existing flows. "missed" displays the number of packets not matching any existing flow and require user space processing. "lost" displays number of packets destined for user space process but subsequently dropped before reaching

```
userspace. The sum of "hit" and "miss" equals to the total number of packets
        datapath processed.
(...)
```



注意

其中一些文档指的是内核数据路径，因此当它指出 用户空间处理时，意味着数据包不会被归类为内核 sw 缓存中（等同于 emc 和 dpcls）并将其发送到用户空间中的 proto 层。

第 7 章 在 NOVA 中附加和分离 SR-IOV 端口

使用以下部分附加和分离 SR-IOV 端口。

7.1. 症状

您无法在 Red Hat OpenStack Platform 10 及之后的版本中的 nova 中附加或分离 SR-IOV 端口。Nova 日志报告 VIF 类型 hw_veb 尚未转换。

7.2. 诊断

您不能将 SR-IOV 端口附加到已创建的实例。SR-IOV 端口需要在实例创建时附加。

7.3. 解决方案

以下示例显示了在实例启动后尝试附加接口：

```
RHEL_INSTANCE_COUNT=1
NETID=$(neutron net-list | grep provider1 | awk '{print $2}')
for i in `seq 1 $RHEL_INSTANCE_COUNT`;do
#  nova floating-ip-create provider1
  portid1=`neutron port-create sriov1 --name sriov1 --binding:vnic-type direct  | awk '$2 == "id" {print
$(NF-1)}'`
  portid2=`neutron port-create sriov2 --name sriov2 --binding:vnic-type direct  | awk '$2 == "id" {print
$(NF-1)}'`
  openstack server create --flavor m1.small  --image rhel --nic net-id=$NETID --key-name id_rsa
sriov_vm${i}
  serverid=`openstack server list | grep sriov_vm${i} | awk '{print $2}'`
  status="NONE"
  while [ "$status" != "ACTIVE" ]; do
    echo "Server $serverid not active ($status)" ; sleep 5 ;
    status=`openstack server show $serverid | grep -i status | awk '{print $4}'`
  done
  nova interface-attach --port-id $portid1 $serverid
  nova interface-attach --port-id $portid2 $serverid
done
```

这失败并显示以下错误：

```
ERROR (ClientException): Unexpected API Error. Please report this at
http://bugs.launchpad.net/nova/ and attach the Nova API log if possible.
<type 'exceptions.KeyError'> (HTTP 500) (Request-ID: req-36b544f4-91a6-442e-a30d-
6148220d1449)
```

-

正确的方法是直接生成带有 SR-IOV 端口的实例：

```
RHEL_INSTANCE_COUNT=1
NETID=$(neutron net-list | grep provider1 | awk '{print $2}')
for i in `seq 1 $RHEL_INSTANCE_COUNT`;do
#  nova floating-ip-create provider1
  portid1=`neutron port-create sriov1 --name sriov1 --binding:vnic-type direct  | awk '$2 == "id" {print $(NF-1)}'`
  portid2=`neutron port-create sriov2 --name sriov2 --binding:vnic-type direct  | awk '$2 == "id" {print $(NF-1)}'`
  openstack server create --flavor m1.small  --image rhel --nic net-id=$NETID --nic port-id=$portid1 --nic port-id=$portid2 --key-name id_rsa   sriov_vm${i}
done
```

第 8 章 使用 OPEN VSWITCH DPDK 配置和测试 LACP 绑定

注意

取决于您使用的 Red Hat OpenStack Platform(RHOSP)版本，使用 LACP 的 OVS 绑定可能不被支持。检查产品文档，以验证是否支持使用 LACP 的 OVS 绑定。

要使用 Open vSwitch DPDK 配置和测试 LACP 绑定，请完成以下任务：

1. 为 LACP 配置交换机端口。
2. 为 LACP 配置 Linux 内核绑定作为基准。
3. 为 LACP 配置 OVS DPDK 绑定。

注意

本主题论述了使用 Dell S4048-ON 交换机的切换配置。RHEL 和 OVS 的配置保持不变，不同的交换机厂商的操作系统将使用不同的语法来配置 LACP。

8.1. 为 LACP 配置交换端口

1. 将交换机接口重置为默认设置：

```
S4048-ON-sw#config t
S4048-ON-sw(conf)#default int te1/2
S4048-ON-sw(conf)#default int te1/7
```

2. 配置端口通道和其他端口设置：

```
S4048-ON-sw(conf)#int range te1/2,te1/7
S4048-ON-sw(conf-if-range-te-1/2,te-1/7)#port-channel-protocol lacp
S4048-ON-sw(conf-if-range-te-1/2,te-1/7-lacp)#
S4048-ON-sw(conf-if-range-te-1/2,te-1/7-lacp)#port-channel 1 mode active
S4048-ON-sw(conf-if-range-te-1/2,te-1/7-lacp)#end
S4048-ON-sw#config t
S4048-ON-sw(conf)#int range te1/2,te1/7
```

```
S4048-ON-sw(conf-if-range-te-1/2,te-1/7)# no ip address
S4048-ON-sw(conf-if-range-te-1/2,te-1/7)# mtu 9216
S4048-ON-sw(conf-if-range-te-1/2,te-1/7)# flowcontrol rx on tx off
S4048-ON-sw(conf-if-range-te-1/2,te-1/7)# no shutdown
S4048-ON-sw(conf-if-range-te-1/2,te-1/7)#end
S4048-ON-sw#show run int te1/2
!
interface TenGigabitEthernet 1/2
 no ip address
 mtu 9216
 flowcontrol rx on tx off
!
 port-channel-protocol LACP
  port-channel 1 mode active
 no shutdown
```

3. **配置 VLAN：**

```
S4048-ON-sw#config t
S4048-ON-sw(conf)#int range vlan901-909
S4048-ON-sw(conf-if-range-vl-901-909)#tagged Port-channel 1
S4048-ON-sw(conf-if-range-vl-901-909)#end
S4048-ON-sw#
```

4. **验证 VLAN 标记：**

```
S4048-ON-sw#show vlan id 902

Codes: * - Default VLAN, G - GVRP VLANs, R - Remote Port Mirroring VLANs, P - Primary,
C - Community, I - Isolated
       O - Openflow, Vx - Vxlan
Q: U - Untagged, T - Tagged
   x - Dot1x untagged, X - Dot1x tagged
   o - OpenFlow untagged, O - OpenFlow tagged
   G - GVRP tagged, M - Vlan-stack
   i - Internal untagged, I - Internal tagged, v - VLT untagged, V - VLT tagged

    NUM    Status    Description                Q Ports
    902    Active    Tenant                     T Po1()
                                                T Te 1/1,1/3-1/6,1/8-1/20
```

5. **验证 LACP 配置：**

```
S4048-ON-sw#show lacp 1
Port-channel 1 admin up, oper down, mode lacp
LACP Fast Switch-Over Disabled
Actor   System ID:  Priority 32768, Address 1418.7789.9a8a
Partner System ID:  Priority 0, Address 0000.0000.0000
Actor Admin Key 1, Oper Key 1, Partner Oper Key 1, VLT Peer Oper Key 1
```

```
LACP LAG 1 is an individual link

LACP LAG 1 is a normal LAG

A - Active LACP, B - Passive LACP, C - Short Timeout, D - Long Timeout
E - Aggregatable Link, F - Individual Link, G - IN_SYNC, H - OUT_OF_SYNC
I - Collection enabled, J - Collection disabled, K - Distribution enabled
L - Distribution disabled, M - Partner Defaulted, N - Partner Non-defaulted,
O - Receiver is in expired state, P - Receiver is not in expired state

Port Te 1/2 is disabled, LACP is disabled and mode is lacp
Port State: Not in Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEHJLMP Key 1 Priority 32768
  Partner is not present

Port Te 1/7 is enabled, LACP is enabled and mode is lacp
Port State: Not in Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEHJLMP Key 1 Priority 32768
  Partner is not present
```

8.2. 为 LACP 配置 LINUX 内核绑定作为基线

将 Linux 内核绑定配置为基线，然后验证主机是否可以使用交换机组成 LACP 绑定。

1. **将所有接口移动到内核空间并使用内核空间绑定进行测试。在本例中，p1p1 映射到总线地址 0000:04:00.0 和 p1p2 映射到总线地址 0000:04:00.1。**

```
[root@baremetal ~]# driverctl unset-override 0000:04:00.0
[root@baremetal ~]# driverctl unset-override 0000:04:00.1
```

2. **加载绑定驱动程序，配置绑定接口(bond10)和 enslave 接口 p1p1 和 p1p2 ：**

```
[root@baremetal ~]# modprobe bonding miimon=100 mode=4 lacp_rate=1
[root@baremetal ~]# ip link add name bond10 type bond
[root@baremetal ~]# ifenslave bond10 p1p1 p1p2
Illegal operation; the specified master interface 'bond10' is not up.
[root@baremetal ~]# ip link set dev bond10 up
[root@baremetal ~]# ifenslave bond10 p1p1 p1p2
```

3. **从 RHEL 验证 LACP：**

```
[root@baremetal ~]# cat /proc/net/bonding/bond10
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)
```

```
Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
802.3ad info
LACP rate: fast
Min links: 0
Aggregator selection policy (ad_select): stable
System priority: 65535
System MAC address: a0:36:9f:e3:dd:c8
Active Aggregator Info:
        Aggregator ID: 1
        Number of ports: 2
        Actor Key: 13
        Partner Key: 1
        Partner Mac Address: 14:18:77:89:9a:8a

Slave Interface: p1p1
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: a0:36:9f:e3:dd:c8
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: monitoring
Partner Churn State: monitoring
Actor Churned Count: 0
Partner Churned Count: 0
details actor lacp pdu:
    system priority: 65535
    system mac address: a0:36:9f:e3:dd:c8
    port key: 13
    port priority: 255
        port number: 1
        port state: 63
    details partner lacp pdu:
        system priority: 32768
        system mac address: 14:18:77:89:9a:8a
        oper key: 1
        port priority: 32768
        port number: 203
        port state: 63

    Slave Interface: p1p2
    MII Status: up
    Speed: 10000 Mbps
    Duplex: full
    Link Failure Count: 0
    Permanent HW addr: a0:36:9f:e3:dd:ca
    Slave queue ID: 0
    Aggregator ID: 1
    Actor Churn State: monitoring
    Partner Churn State: monitoring
```

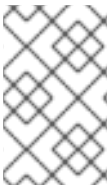
```
Actor Churned Count: 0
Partner Churned Count: 0
details actor lacp pdu:
    system priority: 65535
    system mac address: a0:36:9f:e3:dd:c8
    port key: 13
    port priority: 255
    port number: 2
    port state: 63
details partner lacp pdu:
    system priority: 32768
    system mac address: 14:18:77:89:9a:8a
    oper key: 1
    port priority: 32768
    port number: 208
    port state: 63
```

4. **从交换机验证 LACP：**

```
S4048-ON-sw#show lacp 1
Port-channel 1 admin up, oper up, mode lacp
LACP Fast Switch-Over Disabled
Actor   System ID:  Priority 32768, Address 1418.7789.9a8a
Partner System ID:  Priority 65535, Address a036.9fe3.ddc8
Actor Admin Key 1, Oper Key 1, Partner Oper Key 13, VLT Peer Oper Key 1
LACP LAG 1 is an aggregatable link
LACP LAG 1 is a normal LAG

A - Active LACP, B - Passive LACP, C - Short Timeout, D - Long Timeout
E - Aggregatable Link, F - Individual Link, G - IN_SYNC, H - OUT_OF_SYNC
I - Collection enabled, J - Collection disabled, K - Distribution enabled
L - Distribution disabled, M - Partner Defaulted, N - Partner Non-defaulted,
O - Receiver is in expired state, P - Receiver is not in expired state

Port Te 1/2 is enabled, LACP is enabled and mode is lacp
Port State: Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEGIKNP Key 1 Priority 32768
  Partner Admin: State BDFHJLMP Key 0 Priority 0
          Oper: State ACEGIKNP Key 13 Priority 255

Port Te 1/7 is enabled, LACP is enabled and mode is lacp
Port State: Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEGIKNP Key 1 Priority 32768
  Partner Admin: State BDFHJLMP Key 0 Priority 0
          Oper: State ACEGIKNP Key 13 Priority 255
S4048-ON-sw#
```

5. **删除绑定配置：**

```
[root@baremetal ~]# ip link del dev bond10
[root@baremetal ~]#
```



注意

有关更改绑定模式的详情，请参考：如何在不重启系统的情况下更改绑定模式？

8.3. 为 LACP 配置 OVS DPDK 绑定

下一目标是在 OVS DPDK 中配置 LACP 绑定。

8.3.1. 准备 Open vSwitch

1. 确定在 RHEL 中配置巨页和其他值：

```
[root@baremetal bonding]# cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-3.10.0-693.17.1.el7.x86_64 root=UUID=fa414390-f78d-49d4-
a164-54615a32977b ro console=tty0
console=ttyS0,115200n8 crashkernel=auto rhgb quiet default_hugepagesz=1GB
hugepagesz=1G hugepages=32 iommu=pt intel_iommu=on
isolcpus=2,4,6,8,10,12,14,16,18,22,24,26,28,30,32,34,36,38,3,5,7,9,11,13,15,17,19,23,25,27,2
9,31,33,35,37,39 skew_tick=1
nohz=on
nohz_full=2,4,6,8,10,12,14,16,18,22,24,26,28,30,32,34,36,38,3,5,7,9,11,13,15,17,19,23,25,27,
29,31,33,35,37,39
rcu_nocbs=2,4,6,8,10,12,14,16,18,22,24,26,28,30,32,34,36,38,3,5,7,9,11,13,15,17,19,23,25,2
7,29,31,33,35,37,39
tuned.non_isolcpus=00300003 intel_pstate=disable nosoftlockup
```

2. 为 DPDK 配置 OVS：

```
[root@baremetal bonding]# ovs-vsctl list Open_vSwitch | grep other
other_config        : {}
[root@baremetal bonding]# ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-
mask=0x17c0017c
[root@baremetal bonding]# ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-
mask=0x00000001
[root@baremetal bonding]# ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-
init="true"
```

3. 将接口切换到用户空间：

```
[root@baremetal bonding]# ethtool -i p1p1 | grep bus
```

```
bus-info: 0000:04:00.0
[root@baremetal bonding]# ethtool -i p1p2 | grep bus
bus-info: 0000:04:00.1
[root@baremetal bonding]# driverctl set-override 0000:04:00.0 vfio-pci
[root@baremetal bonding]# driverctl set-override 0000:04:00.1 vfio-pci
```

4. **重启 Open vSwitch、journalctl -u ovs-vswitchd -f & amp; 在后台运行：**

```
[root@baremetal bonding]# systemctl restart openvswitch
Apr 19 13:02:49 baremetal systemd[1]: Stopping Open vSwitch Forwarding Unit...
Apr 19 13:02:49 baremetal systemd[1]: Stopping Open vSwitch Forwarding Unit...
Apr 19 13:02:49 baremetal ovs-ctl[91399]: Exiting ovs-vswitchd (91202) [  OK  ]
Apr 19 13:02:49 baremetal ovs-ctl[91399]: Exiting ovs-vswitchd (91202) [  OK  ]
Apr 19 13:02:49 baremetal systemd[1]: Starting Open vSwitch Forwarding Unit...
Apr 19 13:02:49 baremetal systemd[1]: Starting Open vSwitch Forwarding Unit...
Apr 19 13:02:49 baremetal ovs-ctl[91483]: Starting ovs-vswitchd EAL: Detected 40 lcore(s)
Apr 19 13:02:49 baremetal ovs-ctl[91483]: Starting ovs-vswitchd EAL: Detected 40 lcore(s)
Apr 19 13:02:49 baremetal ovs-ctl[91483]: EAL: Probing VFIO support...
Apr 19 13:02:49 baremetal ovs-vswitchd[91509]: EAL: Probing VFIO support...
Apr 19 13:02:49 baremetal ovs-ctl[91483]: EAL: VFIO support initialized
Apr 19 13:02:49 baremetal ovs-vswitchd[91509]: EAL: VFIO support initialized
Apr 19 13:02:49 baremetal ovs-ctl[91483]: EAL: Probing VFIO support...
Apr 19 13:02:49 baremetal ovs-vswitchd[91509]: EAL: Probing VFIO support...
Apr 19 13:02:49 baremetal ovs-ctl[91483]: EAL: VFIO support initialized
Apr 19 13:02:49 baremetal ovs-vswitchd[91509]: EAL: VFIO support initialized
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL: PCI device 0000:04:00.0 on NUMA socket 0
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL: PCI device 0000:04:00.0 on NUMA socket 0
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL: PCI device 0000:04:00.0 on NUMA
socket 0
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL:   using IOMMU type 1 (Type 1)
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL:   using IOMMU type 1 (Type 1)
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL: PCI device 0000:04:00.0 on NUMA
socket 0
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL:   using IOMMU type 1 (Type 1)
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL:   using IOMMU type 1 (Type 1)
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL: Ignore mapping IO port bar(2) addr: 3021
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL: Ignore mapping IO port bar(2) addr: 3021
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL: Ignore mapping IO port bar(2) addr:
3021
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL: Ignore mapping IO port bar(2) addr:
3021
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL: PCI device 0000:04:00.1 on NUMA
socket 0
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL: PCI device 0000:04:00.1 on NUMA
socket 0
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL: PCI device 0000:04:00.1 on NUMA socket 0
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL: PCI device 0000:04:00.1 on NUMA socket 0
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL:   probe driver: 8086:154d net_ixgbe
```

```
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL: Ignore mapping IO port bar(2) addr: 3001
Apr 19 13:02:59 baremetal ovs-ctl[91483]: EAL: Ignore mapping IO port bar(2) addr: 3001
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL: Ignore mapping IO port bar(2) addr:
3001
Apr 19 13:02:59 baremetal ovs-vswitchd[91509]: EAL: Ignore mapping IO port bar(2) addr:
3001
Apr 19 13:03:00 baremetal ovs-ctl[91483]: EAL: PCI device 0000:05:00.0 on NUMA socket 0
Apr 19 13:03:00 baremetal ovs-ctl[91483]: EAL: PCI device 0000:05:00.0 on NUMA socket 0
Apr 19 13:03:00 baremetal ovs-vswitchd[91509]: EAL: PCI device 0000:05:00.0 on NUMA
socket 0
Apr 19 13:03:00 baremetal ovs-vswitchd[91509]: EAL: PCI device 0000:05:00.0 on NUMA
socket 0
Apr 19 13:03:00 baremetal ovs-ctl[91483]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:03:00 baremetal ovs-ctl[91483]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:03:00 baremetal ovs-ctl[91483]: EAL: PCI device 0000:05:00.1 on NUMA socket 0
Apr 19 13:03:00 baremetal ovs-ctl[91483]: EAL: PCI device 0000:05:00.1 on NUMA socket 0
Apr 19 13:03:00 baremetal ovs-ctl[91483]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:03:00 baremetal ovs-ctl[91483]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:03:00 baremetal ovs-vswitchd[91509]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:03:00 baremetal ovs-vswitchd[91509]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:03:00 baremetal ovs-vswitchd[91509]: EAL: PCI device 0000:05:00.1 on NUMA
socket 0
Apr 19 13:03:00 baremetal ovs-vswitchd[91509]: EAL: PCI device 0000:05:00.1 on NUMA
socket 0
Apr 19 13:03:00 baremetal ovs-vswitchd[91509]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:03:00 baremetal ovs-vswitchd[91509]: EAL:   probe driver: 8086:154d net_ixgbe
Apr 19 13:03:00 baremetal ovs-ctl[91483]: [  OK  ]
Apr 19 13:03:00 baremetal ovs-ctl[91483]: [  OK  ]
Apr 19 13:03:00 baremetal ovs-ctl[91483]: Enabling remote OVSDB managers [  OK  ]
Apr 19 13:03:00 baremetal ovs-ctl[91483]: Enabling remote OVSDB managers [  OK  ]
Apr 19 13:03:00 baremetal systemd[1]: Started Open vSwitch Forwarding Unit.
Apr 19 13:03:00 baremetal systemd[1]: Started Open vSwitch Forwarding Unit.
[root@baremetal bonding]#
```

8.3.2. 配置 LACP Bond

1. **添加绑定：**

```
[root@baremetal bonding]# ovs-vsctl add-br ovsbr0  -- set bridge ovsbr0
datapath_type=netdev
[root@baremetal bonding]# ovs-vsctl add-bond ovsbr0 dpdkbond dpdk0 dpdk1
bond_mode=balance-tcp     lacp=active -- set
interface dpdk0 type=dpdk     -- set Interface dpdk1 type=dpdk
```

2. **从 Open vSwitch 验证：**

```
[root@baremetal bonding]# ovs-appctl lacp/show dpdkbond
---- dpdkbond ----
        status: active negotiated
        sys_id: a0:36:9f:e3:dd:c8
```

```
        sys_priority: 65534
        aggregation key: 1
        lacp_time: slow

slave: dpdk0: current attached
        port_id: 2
        port_priority: 65535
        may_enable: true

        actor sys_id: a0:36:9f:e3:dd:c8
        actor sys_priority: 65534
        actor port_id: 2
        actor port_priority: 65535
        actor key: 1
        actor state: activity aggregation synchronized collecting distributing

        partner sys_id: 14:18:77:89:9a:8a
        partner sys_priority: 32768
        partner port_id: 203
        partner port_priority: 32768
        partner key: 1
        partner state: activity timeout aggregation synchronized collecting distributing

slave: dpdk1: current attached
        port_id: 1
        port_priority: 65535
        may_enable: true

        actor sys_id: a0:36:9f:e3:dd:c8
        actor sys_priority: 65534
        actor port_id: 1
        actor port_priority: 65535
        actor key: 1
        actor state: activity aggregation synchronized collecting distributing

        partner sys_id: 14:18:77:89:9a:8a
        partner sys_priority: 32768
        partner port_id: 208
        partner port_priority: 32768
        partner key: 1
        partner state: activity timeout aggregation synchronized collecting distributing

[root@baremetal bonding]# ovs-appctl bond/show dpdkbond
---- dpdkbond ----
bond_mode: balance-tcp
bond may use recirculation: yes, Recirc-ID : 1
bond-hash-basis: 0
updelay: 0 ms
downdelay: 0 ms
next rebalance: 6817 ms
lacp_status: negotiated
active slave mac: a0:36:9f:e3:dd:c8(dpdk0)

slave dpdk0: enabled
        active slave
        may_enable: true
```

```
slave dpdk1: enabled
        may_enable: true
```

3. **从交换机验证：**

```
S4048-ON-sw#show lacp 1
Port-channel 1 admin up, oper up, mode lacp
LACP Fast Switch-Over Disabled
Actor   System ID:  Priority 32768, Address 1418.7789.9a8a
Partner System ID:  Priority 65534, Address a036.9fe3.ddc8
Actor Admin Key 1, Oper Key 1, Partner Oper Key 1, VLT Peer Oper Key 1
LACP LAG 1 is an aggregatable link
LACP LAG 1 is a normal LAG

A - Active LACP, B - Passive LACP, C - Short Timeout, D - Long Timeout
E - Aggregatable Link, F - Individual Link, G - IN_SYNC, H - OUT_OF_SYNC
I - Collection enabled, J - Collection disabled, K - Distribution enabled
L - Distribution disabled, M - Partner Defaulted, N - Partner Non-defaulted,
O - Receiver is in expired state, P - Receiver is not in expired state

Port Te 1/2 is enabled, LACP is enabled and mode is lacp
Port State: Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEGIKNP Key 1 Priority 32768
  Partner Admin: State BDFHJLMP Key 0 Priority 0
          Oper: State ADEGIKNP Key 1 Priority 65535

Port Te 1/7 is enabled, LACP is enabled and mode is lacp
Port State: Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEGIKNP Key 1 Priority 32768
  Partner Admin: State BDFHJLMP Key 0 Priority 0
          Oper: State ADEGIKNP Key 1 Priority 65535
S4048-ON-sw#
```

8.3.3. 从 OVS 启用/禁用端口

您可以使用 ovs-ofctl mod-port <bridge> <port> [up|down]启用或禁用端口

1. **关闭端口：**

```
[root@baremetal bonding]#  ovs-ofctl mod-port ovsbr0 dpdk1 down
```

2. **验证关闭：**

```
[root@baremetal bonding]# ovs-appctl lacp/show dpdkbond
---- dpdkbond ----
        status: active negotiated
        sys_id: a0:36:9f:e3:dd:c8
        sys_priority: 65534
        aggregation key: 1
        lacp_time: slow

slave: dpdk0: current attached
        port_id: 2
        port_priority: 65535
        may_enable: true

        actor sys_id: a0:36:9f:e3:dd:c8
        actor sys_priority: 65534
        actor port_id: 2
        actor port_priority: 65535
        actor key: 1
        actor state: activity aggregation synchronized collecting distributing

        partner sys_id: 14:18:77:89:9a:8a
        partner sys_priority: 32768
        partner port_id: 203
        partner port_priority: 32768
        partner key: 1
        partner state: activity timeout aggregation synchronized collecting distributing

slave: dpdk1: defaulted detached
        port_id: 1
        port_priority: 65535
        may_enable: false

        actor sys_id: a0:36:9f:e3:dd:c8
        actor sys_priority: 65534
        actor port_id: 1
        actor port_priority: 65535
        actor key: 1
        actor state: activity aggregation defaulted

        partner sys_id: 00:00:00:00:00:00
        partner sys_priority: 0
        partner port_id: 0
        partner port_priority: 0
        partner key: 0
        partner state:
[root@baremetal bonding]# ovs-appctl bond/show dpdkbond
---- dpdkbond ----
bond_mode: balance-tcp
bond may use recirculation: yes, Recirc-ID : 1
bond-hash-basis: 0
updelay: 0 ms
downdelay: 0 ms
next rebalance: 3315 ms
lacp_status: negotiated
active slave mac: a0:36:9f:e3:dd:c8(dpdk0)
```

```
slave dpdk0: enabled
        active slave
        may_enable: true

slave dpdk1: disabled
        may_enable: false
```

3. **在交换机中验证：**

```
S4048-ON-sw#show lacp 1
Port-channel 1 admin up, oper up, mode lacp
LACP Fast Switch-Over Disabled
Actor   System ID:  Priority 32768, Address 1418.7789.9a8a
Partner System ID:  Priority 65534, Address a036.9fe3.ddc8
Actor Admin Key 1, Oper Key 1, Partner Oper Key 1, VLT Peer Oper Key 1
LACP LAG 1 is an aggregatable link
LACP LAG 1 is a normal LAG

A - Active LACP, B - Passive LACP, C - Short Timeout, D - Long Timeout
E - Aggregatable Link, F - Individual Link, G - IN_SYNC, H - OUT_OF_SYNC
I - Collection enabled, J - Collection disabled, K - Distribution enabled
L - Distribution disabled, M - Partner Defaulted, N - Partner Non-defaulted,
O - Receiver is in expired state, P - Receiver is not in expired state

Port Te 1/2 is enabled, LACP is enabled and mode is lacp
Port State: Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEGIKNP Key 1 Priority 32768
  Partner Admin: State BDFHJLMP Key 0 Priority 0
          Oper: State ADEGIKNP Key 1 Priority 65535

Port Te 1/7 is disabled, LACP is disabled and mode is lacp
Port State: Not in Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEHJLNP Key 1 Priority 32768
  Partner is not present
```

4. **重新启用该端口：**

```
[root@baremetal bonding]#  ovs-ofctl mod-port ovsbr0 dpdk1 up
```

5. **从 RHEL 验证：**

```
[root@baremetal bonding]# ovs-appctl bond/show dpdkbond
---- dpdkbond ----
bond_mode: balance-tcp
bond may use recirculation: yes, Recirc-ID : 1
bond-hash-basis: 0
updelay: 0 ms
```

```
downdelay: 0 ms
next rebalance: 7846 ms
lacp_status: negotiated
active slave mac: a0:36:9f:e3:dd:c8(dpdk0)

slave dpdk0: enabled
        active slave
        may_enable: true

slave dpdk1: enabled
        may_enable: true

[root@baremetal bonding]# ovs-appctl lacp/show dpdkbond
---- dpdkbond ----
        status: active negotiated
        sys_id: a0:36:9f:e3:dd:c8
        sys_priority: 65534
        aggregation key: 1
        lacp_time: slow

slave: dpdk0: current attached
        port_id: 2
        port_priority: 65535
        may_enable: true

        actor sys_id: a0:36:9f:e3:dd:c8
        actor sys_priority: 65534
        actor port_id: 2
        actor port_priority: 65535
        actor key: 1
        actor state: activity aggregation synchronized collecting distributing

        partner sys_id: 14:18:77:89:9a:8a
        partner sys_priority: 32768
        partner port_id: 203
        partner port_priority: 32768
        partner key: 1
        partner state: activity timeout aggregation synchronized collecting distributing

slave: dpdk1: current attached
        port_id: 1
        port_priority: 65535
        may_enable: true

        actor sys_id: a0:36:9f:e3:dd:c8
        actor sys_priority: 65534
        actor port_id: 1
        actor port_priority: 65535
        actor key: 1
        actor state: activity aggregation synchronized collecting distributing

        partner sys_id: 14:18:77:89:9a:8a
        partner sys_priority: 32768
        partner port_id: 208
```

```
        partner port_priority: 32768
        partner key: 1
        partner state: activity timeout aggregation synchronized collecting distributing
```

6. **从交换机验证：**

```
S4048-ON-sw#show lacp 1
Port-channel 1 admin up, oper up, mode lacp
LACP Fast Switch-Over Disabled
Actor   System ID:  Priority 32768, Address 1418.7789.9a8a
Partner System ID:  Priority 65534, Address a036.9fe3.ddc8
Actor Admin Key 1, Oper Key 1, Partner Oper Key 1, VLT Peer Oper Key 1
LACP LAG 1 is an aggregatable link
LACP LAG 1 is a normal LAG

A - Active LACP, B - Passive LACP, C - Short Timeout, D - Long Timeout
E - Aggregatable Link, F - Individual Link, G - IN_SYNC, H - OUT_OF_SYNC
I - Collection enabled, J - Collection disabled, K - Distribution enabled
L - Distribution disabled, M - Partner Defaulted, N - Partner Non-defaulted,
O - Receiver is in expired state, P - Receiver is not in expired state

Port Te 1/2 is enabled, LACP is enabled and mode is lacp
Port State: Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEGIKNP Key 1 Priority 32768
  Partner Admin: State BDFHJLMP Key 0 Priority 0
          Oper: State ADEGIKNP Key 1 Priority 65535

Port Te 1/7 is enabled, LACP is enabled and mode is lacp
Port State: Bundle
  Actor   Admin: State ACEHJLMP Key 1 Priority 32768
          Oper: State ACEGIKNP Key 1 Priority 32768
  Partner Admin: State BDFHJLMP Key 0 Priority 0
          Oper: State ADEGIKNP Key 1 Priority 65535
```

第 9 章 使用 OVS DPDK 部署不同的绑定模式

在 Red Hat OpenStack Platform 中使用 OVS-DPDK 部署不同的绑定模式。

9.1. 解决方案

对 compute.yaml 环境文件进行以下更改。请注意，本示例还将 MTU 值设为 2000。

```
(...)
          -
            type: ovs_user_bridge
            name: br-link
            mtu: 2000
            use_dhcp: false
            members:
              -
                type: ovs_dpdk_bond
                name: dpdkbond0
                ovs_options: "bond_mode=balance-slb"
                mtu: 2000
                ovs_extra:
                  - set interface dpdk0 mtu_request=$MTU
                  - set interface dpdk1 mtu_request=$MTU
                members:
                  -
                    type: ovs_dpdk_port
                    name: dpdk0
                    members:
                      -
                        type: interface
                        name: p1p2
                  -
                    type: ovs_dpdk_port
                    name: dpdk1
                    members:
                      -
                        type: interface
                        name: p1p1
(...)
```

使用上述模板更改部署或重新部署 overcloud。完成后，在 overcloud 节点上执行以下步骤。

验证 os-net-config 配置：

```
cat /etc/os-net-config/config.json | python -m json.tool
(...)
```

```
        {
            "members": [
                {
                    "members": [
                        {
                            "members": [
                                {
                                    "name": "p1p2",
                                    "type": "interface"
                                }
                            ],
                            "name": "dpdk0",
                            "type": "ovs_dpdk_port"
                        },
                        {
                            "members": [
                                {
                                    "name": "p1p1",
                                    "type": "interface"
                                }
                            ],
                            "name": "dpdk1",
                            "type": "ovs_dpdk_port"
                        }
                    ],
                    "mtu": 2000,
                    "name": "dpdkbond0",
                    "ovs_extra": [
                        "set interface dpdk0 mtu_request=$MTU",
                        "set interface dpdk1 mtu_request=$MTU"
                    ],
                    "ovs_options": "bond_mode=balance-slb",
                    "type": "ovs_dpdk_bond"
                }
            ],
            "mtu": 2000,
            "name": "br-link",
            "type": "ovs_user_bridge",
            "use_dhcp": false
        },
(...)
```

验证绑定：

```
[root@overcloud-compute-0 ~]# ovs-appctl bond/show dpdkbond0
---- dpdkbond0 ----
bond_mode: balance-slb
bond may use recirculation: no, Recirc-ID : -1
bond-hash-basis: 0
updelay: 0 ms
downdelay: 0 ms
next rebalance: 9221 ms
lacp_status: off
active slave mac: a0:36:9f:e5:da:82(dpdk1)
```

```
slave dpdk0: enabled
    may_enable: true

slave dpdk1: enabled
    active slave
    may_enable: true
```

第 10 章 在 OVS-VSCTL 显示消息中接收 COULD NOT OPEN NETWORK DEVICE DPDK0（不包括这样的设备）

10.1. 症状

您收到 ovs-vsctl 显示消息中的 Could not open network device dpdk0（不包括这样的设备）。

10.2. 诊断

红帽支持在 DPDK 支持的硬件 中列出的轮询模式驱动程序(PMD)的子集。红帽在 2017 年 8 月被禁用的 PMD。

上游 PMD 可能会存在安全性或性能问题。因此， M PMD 需要经过大量测试才能通过红帽的资格测试。

您可以查看 /usr/share/doc/openvswitch-<version>/README.DPDK-PMDS 中所有启用的 PMD 的列表。此列表可能包含红帽不支持 PMD。不支持在 README.DPDK-PMDS 中列出的轮询模式驱动程序。

10.3. 解决方案

以下示例显示了 openvswitch-2.6.1 支持的 PMD：

```
[root@overcloud-compute-0 ~]# cat /usr/share/doc/openvswitch-2.6.1/README.DPDK-PMDS
DPDK drivers included in this package:

E1000
ENIC
I40E
IXGBE
RING
VIRTIO

For more information about the drivers, see
http://dpdk.org/doc/guides-16.11/nics/index.html
```

本例显示了 openvswitch-2.9.0 支持的 PMD：

```
[root@undercloud-r430 ~]# cat /usr/share/doc/openvswitch-2.9.0/README.DPDK-PMDS
DPDK drivers included in this package:
```

BNXT
E1000
ENIC
FAILSAFE
I40E
IXGBE
MLX4
MLX4_GLUE
MLX5
MLX5_GLUE
NFP
RING
SOFTNIC
VIRTIO

For more information about the drivers, see
http://dpdk.org/doc/guides-17.11/nics/index.html

第 11 章 可用主机内存页面不足，可使用 OPEN VSWITCH DPDK 分配客户机 RAM

11.1. 症状

您可以将实例部署到具有足够巨页和其他资源的计算节点上，您会看到如下输出，例如：

```
[stack@undercloud-4 ~]$ nova show 1b72e7a1-c298-4c92-8d2c-0a9fe886e9bc
(...)
| fault                        | {"message": "Exceeded maximum number of retries. Exceeded max
scheduling attempts 3
for instance 1b72e7a1-c298-4c92-8d2c-0a9fe886e9bc. Last exception: internal error: process exited
while connecting to monitor:
2017-11-23T19:53:20.311446Z qemu-kvm: -chardev pty,id=cha", "code": 500, "details": "  File
\"/usr/lib/python2.7/site-packages
/nova/conductor/manager.py\", line 492, in build_instances |
|                              |     filter_properties, instances[0].uuid)
|
|                              |   File \"/usr/lib/python2.7/site-packages/nova/scheduler/utils.py\", line 184, in
populate_retry
|
|                              |     raise exception.MaxRetriesExceeded(reason=msg)
|
|                              | ", "created": "2017-11-23T19:53:22Z"}
(...)
```

和计算节点上的 /var/log/containers/nova/nova-compute.log 提供以下 ERROR 消息：

```
2017-11-23 19:53:21.021 153615 ERROR nova.compute.manager [instance: 1b72e7a1-c298-4c92-
8d2c-0a9fe886e9bc]
2017-11-23T19:53:20.477183Z qemu-kvm: -object memory-backend-file,id=ram-
node0,prealloc=yes,mem-path=/dev/hugepages/libvirt
/qemu/7-instance-00000006,share=yes,size=536870912,host-nodes=0,policy=bind:
os_mem_prealloc: Insufficient free host memory
pages available to allocate guest RAM
```

另外，libvirt 会创建以下日志文件：

```
[root@overcloud-compute-1 qemu]# cat instance-00000006.log
2017-11-23 19:53:02.145+0000: starting up libvirt version: 3.2.0, package: 14.el7_4.3 (Red Hat, Inc.
<http://bugzilla.redhat.com/bugzilla>, 2017-08-22-08:54:01, x86-039.build.eng.bos.redhat.com),
qemu version: 2.9.0(qemu-
kvm-rhev-2.9.0-10.el7), hostname: overcloud-compute-1.localdomain
LC_ALL=C PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin QEMU_AUDIO_DRV=none
/usr/libexec/qemu-kvm -name
guest=instance-00000006,debug-threads=on -S -object
secret,id=masterKey0,format=raw,file=/var/lib/libvirt/qemu/domain-
5-instance-00000006/master-key.aes -machine pc-i440fx-rhel7.4.0,accel=kvm,usb=off,dump-guest-
```

```
core=off -cpu
SandyBridge,vme=on,hypervisor=on,arat=on,tsc_adjust=on,xsaveopt=on -m 512 -realtime mlock=off
-smp
1,sockets=1,cores=1,threads=1 -object memory-backend-file,id=ram-node0,prealloc=yes,mem-
path=/dev/hugepages/libvirt/qemu/5
-instance-00000006,share=yes,size=536870912,host-nodes=0,policy=bind -numa
node,nodeid=0,cpus=0,memdev=ram-node0 -uuid
1b72e7a1-c298-4c92-8d2c-0a9fe886e9bc -smbios 'type=1,manufacturer=Red
Hat,product=OpenStack
Compute,version=14.0.8-5.el7ost,serial=4f88fcca-0cd3-4e19-8dc4-4436a54daff8,uuid=1b72e7a1-
c298-4c92-8d2c
-0a9fe886e9bc,family=Virtual Machine' -no-user-config -nodefaults -chardev
socket,id=charmonitor,path=/var/lib/libvirt
/qemu/domain-5-instance-00000006/monitor.sock,server,nowait -mon
chardev=charmonitor,id=monitor,mode=control -rtc
base=utc,driftfix=slew -global kvm-pit.lost_tick_policy=delay -no-hpet -no-shutdown -boot strict=on -
device piix3
-usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2 -drive file=/var/lib/nova/instances/1b72e7a1-c298-4c92-
8d2c-0a9fe886e9bc
/disk,format=qcow2,if=none,id=drive-virtio-disk0,cache=none -device virtio-blk-
pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-
virtio-disk0,id=virtio-disk0,bootindex=1 -chardev
socket,id=charnet0,path=/var/run/openvswitch/vhu9758ef15-d2 -netdev vhost-
user,chardev=charnet0,id=hostnet0 -device virtio-net-
pci,netdev=hostnet0,id=net0,mac=fa:16:3e:d6:89:65,bus=pci.0,addr=0x3
-add-fd set=0,fd=29 -chardev file,id=charserial0,path=/dev/fdset/0,append=on -device isa-
serial,chardev=charserial0,id=serial0
-chardev pty,id=charserial1 -device isa-serial,chardev=charserial1,id=serial1 -device usb-
tablet,id=input0,bus=usb.0,port=1
-vnc 172.16.2.8:2 -k en-us -device cirrus-vga,id=video0,bus=pci.0,addr=0x2 -device virtio-balloon-
pci,id=balloon0,bus=pci.0,addr=0x5 -msg timestamp=on
2017-11-23T19:53:03.217386Z qemu-kvm: -chardev pty,id=charserial1: char device redirected to
/dev/pts/3 (label charserial1)
2017-11-23T19:53:03.359799Z qemu-kvm: -object memory-backend-file,id=ram-
node0,prealloc=yes,mem-path=/dev/hugepages/libvirt
/qemu/5-instance-00000006,share=yes,size=536870912,host-nodes=0,policy=bind:
os_mem_prealloc: Insufficient free host memory
pages available to allocate guest RAM

2017-11-23 19:53:03.630+0000: shutting down, reason=failed
2017-11-23 19:53:10.052+0000: starting up libvirt version: 3.2.0, package: 14.el7_4.3 (Red Hat, Inc.
<http://bugzilla.redhat.com/bugzilla>, 2017-08-22-08:54:01, x86-039.build.eng.bos.redhat.com),
qemu version: 2.9.0(qemu-
kvm-rhev-2.9.0-10.el7), hostname: overcloud-compute-1.localdomain
LC_ALL=C PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin QEMU_AUDIO_DRV=none
/usr/libexec/qemu-kvm -name
guest=instance-00000006,debug-threads=on -S -object
secret,id=masterKey0,format=raw,file=/var/lib/libvirt/qemu/domain-
6-instance-00000006/master-key.aes -machine pc-i440fx-rhel7.4.0,accel=kvm,usb=off,dump-guest-
core=off -cpu
SandyBridge,vme=on,hypervisor=on,arat=on,tsc_adjust=on,xsaveopt=on -m 512 -realtime mlock=off
-smp
1,sockets=1,cores=1,threads=1 -object memory-backend-file,id=ram-node0,prealloc=yes,mem-
path=/dev/hugepages/libvirt/qemu/6-
instance-00000006,share=yes,size=536870912,host-nodes=0,policy=bind -numa
node,nodeid=0,cpus=0,memdev=ram-node0 -uuid
```

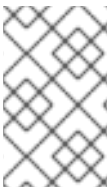
```
1b72e7a1-c298-4c92-8d2c-0a9fe886e9bc -smbios 'type=1,manufacturer=Red
Hat,product=OpenStack
Compute,version=14.0.8-5.el7ost,serial=4f88fcca-0cd3-4e19-8dc4-4436a54daff8,uuid=1b72e7a1-
c298-4c92-8d2c-
0a9fe886e9bc,family=Virtual Machine' -no-user-config -nodefaults -chardev
socket,id=charmonitor,path=/var/lib/libvirt
/qemu/domain-6-instance-00000006/monitor.sock,server,nowait -mon
chardev=charmonitor,id=monitor,mode=control -rtc
base=utc,driftfix=slew -global kvm-pit.lost_tick_policy=delay -no-hpet -no-shutdown -boot strict=on -
device piix3
-usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2 -drive file=/var/lib/nova/instances/1b72e7a1-c298-4c92-
8d2c-0a9fe886e9bc
/disk,format=qcow2,if=none,id=drive-virtio-disk0,cache=none -device virtio-blk-
pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-
virtio-disk0,id=virtio-disk0,bootindex=1 -chardev
socket,id=charnet0,path=/var/run/openvswitch/vhu9758ef15-d2 -netdev vhost-
user,chardev=charnet0,id=hostnet0 -device virtio-net-
pci,netdev=hostnet0,id=net0,mac=fa:16:3e:d6:89:65,bus=pci.0,addr=0x3
-add-fd set=0,fd=29 -chardev file,id=charserial0,path=/dev/fdset/0,append=on -device isa-
serial,chardev=charserial0,id=serial0
-chardev pty,id=charserial1 -device isa-serial,chardev=charserial1,id=serial1 -device usb-
tablet,id=input0,bus=usb.0,port=1
-vnc 172.16.2.8:2 -k en-us -device cirrus-vga,id=video0,bus=pci.0,addr=0x2 -device virtio-balloon-
pci,id=balloon0,bus=pci.0,addr=0x5 -msg timestamp=on
2017-11-23T19:53:11.466399Z qemu-kvm: -chardev pty,id=charserial1: char device redirected to
/dev/pts/3 (label charserial1)
2017-11-23T19:53:11.729226Z qemu-kvm: -object memory-backend-file,id=ram-
node0,prealloc=yes,mem-path=/dev/hugepages/libvirt
/qemu/6-instance-00000006,share=yes,size=536870912,host-nodes=0,policy=bind:
os_mem_prealloc: Insufficient free host memory
pages available to allocate guest RAM

2017-11-23 19:53:12.159+0000: shutting down, reason=failed
2017-11-23 19:53:19.370+0000: starting up libvirt version: 3.2.0, package: 14.el7_4.3 (Red Hat, Inc.
<http://bugzilla.redhat.com/bugzilla>, 2017-08-22-08:54:01, x86-039.build.eng.bos.redhat.com),
qemu version: 2.9.0(qemu-
kvm-rhev-2.9.0-10.el7), hostname: overcloud-compute-1.localdomain
LC_ALL=C PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin QEMU_AUDIO_DRV=none
/usr/libexec/qemu-kvm -name
guest=instance-00000006,debug-threads=on -S -object
secret,id=masterKey0,format=raw,file=/var/lib/libvirt/qemu/domain-
7-instance-00000006/master-key.aes -machine pc-i440fx-rhel7.4.0,accel=kvm,usb=off,dump-guest-
core=off -cpu
SandyBridge,vme=on,hypervisor=on,arat=on,tsc_adjust=on,xsaveopt=on -m 512 -realtime mlock=off
-smp
1,sockets=1,cores=1,threads=1 -object memory-backend-file,id=ram-node0,prealloc=yes,mem-
path=/dev/hugepages/libvirt/qemu/7-
instance-00000006,share=yes,size=536870912,host-nodes=0,policy=bind -numa
node,nodeid=0,cpus=0,memdev=ram-node0 -uuid
1b72e7a1-c298-4c92-8d2c-0a9fe886e9bc -smbios 'type=1,manufacturer=Red
Hat,product=OpenStack
Compute,version=14.0.8-5.el7ost,serial=4f88fcca-0cd3-4e19-8dc4-4436a54daff8,uuid=1b72e7a1-
c298-4c92-8d2c
-0a9fe886e9bc,family=Virtual Machine' -no-user-config -nodefaults -chardev
socket,id=charmonitor,path=/var/lib/libvirt
/qemu/domain-7-instance-00000006/monitor.sock,server,nowait -mon
```

```
chardev=charmonitor,id=monitor,mode=control -rtc
base=utc,driftfix=slew -global kvm-pit.lost_tick_policy=delay -no-hpet -no-shutdown -boot strict=on -
device piix3-
usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2 -drive file=/var/lib/nova/instances/1b72e7a1-c298-4c92-
8d2c-0a9fe886e9bc
/disk,format=qcow2,if=none,id=drive-virtio-disk0,cache=none -device virtio-blk-
pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-
virtio-disk0,id=virtio-disk0,bootindex=1 -chardev
socket,id=charnet0,path=/var/run/openvswitch/vhu9758ef15-d2 -netdev vhost-
user,chardev=charnet0,id=hostnet0 -device virtio-net-
pci,netdev=hostnet0,id=net0,mac=fa:16:3e:d6:89:65,bus=pci.0,addr=0x3
-add-fd set=0,fd=29 -chardev file,id=charserial0,path=/dev/fdset/0,append=on -device isa-
serial,chardev=charserial0,id=serial0
-chardev pty,id=charserial1 -device isa-serial,chardev=charserial1,id=serial1 -device usb-
tablet,id=input0,bus=usb.0,port=1
-vnc 172.16.2.8:2 -k en-us -device cirrus-vga,id=video0,bus=pci.0,addr=0x2 -device virtio-balloon-
pci,id=balloon0,bus=pci.0,addr=0x5 -msg timestamp=on
2017-11-23T19:53:20.311446Z qemu-kvm: -chardev pty,id=charserial1: char device redirected to
/dev/pts/3 (label charserial1)
2017-11-23T19:53:20.477183Z qemu-kvm: -object memory-backend-file,id=ram-
node0,prealloc=yes,mem-path=/dev/hugepages/libvirt
/qemu/7-instance-00000006,share=yes,size=536870912,host-nodes=0,policy=bind:
os_mem_prealloc: Insufficient free host memory
pages available to allocate guest RAM

2017-11-23 19:53:20.724+0000: shutting down, reason=failed
```

11.2. 诊断

如果没有额外的设置，nova 不知道其他进程使用了一定数量的巨页内存。默认情况下，nova 假定所有巨页内存都可用于实例。如果假定这个 NUMA 节点仍有 pCPU 和空闲的巨页内存，Nova 将首先填满 NUMA 节点 0。这可能是因为以下原因造成的：

- **请求的 pCPU 仍适用于 NUMA 0**
- **所有现有实例的合并内存加上实例的内存仍然被生成至 NUMA 节点 0**
- **OVS 等另一个进程在 NUMA 节点 0 上保存一定数量的巨页内存。**



注意

确保分配与巨页倍数相等的类别 RAM 量，以避免 [Errno 12] 无法分配内存 错误。

11.2.1. 诊断步骤

1. **检查 meminfo。以下显示了每个 NUMA 节点有 2MB 巨页和 512 个可用巨页的虚拟机监控程序：**

```
[root@overcloud-compute-1 ~]# cat /sys/devices/system/node/node*/meminfo  | grep -i huge
Node 0 AnonHugePages:      2048 kB
Node 0 HugePages_Total:  1024
Node 0 HugePages_Free:    512
Node 0 HugePages_Surp:      0
Node 1 AnonHugePages:      2048 kB
Node 1 HugePages_Total:  1024
Node 1 HugePages_Free:    512
Node 1 HugePages_Surp:      0
```

2. **检查 NUMA 架构：**

```
[root@overcloud-compute-1 nova]# lscpu  | grep -i NUMA
NUMA node(s):          2
NUMA node0 CPU(s):     0-3
NUMA node1 CPU(s):     4-7
```

3. **检查 OVS 保留的巨页。在以下输出中，OVS 会为每个 NUMA 节点保留 512MB 的巨页：**

```
[root@overcloud-compute-1 virt]# ovs-vsctl list Open_vSwitch | grep mem
other_config        : {dpdk-init="true", dpdk-lcore-mask="3", dpdk-socket-mem="512,512",
pmd-cpu-mask="1e"}
```

4. **部署具有以下类别的实例（1 个 vCPU 和 512 MB 或内存）：**

```
[stack@undercloud-4 ~]$ nova flavor-show m1.tiny
+----------------------------+-------------------------------------------------------------+
| Property                   | Value                                                       |
+----------------------------+-------------------------------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                                       |
| OS-FLV-EXT-DATA:ephemeral  | 0                                                           |
| disk                       | 8                                                           |
| extra_specs                | {"hw:cpu_policy": "dedicated", "hw:mem_page_size": "large"} |
| id                         | 49debbdb-c12e-4435-97ef-f575990b352f                        |
| name                       | m1.tiny                                                     |
| os-flavor-access:is_public | True                                                        |
| ram                        | 512                                                         |
| rxtx_factor                | 1.0                                                         |
| swap                       |                                                             |
| vcpus                      | 1                                                           |
+----------------------------+-------------------------------------------------------------+
```

新实例将引导并将使用来自 NUMA 1 的内存：

```
[stack@undercloud-4 ~]$ nova list | grep d98772d1-119e-48fa-b1d9-8a68411cba0b
| d98772d1-119e-48fa-b1d9-8a68411cba0b | cirros-test0 | ACTIVE | -          | Running     |
provider1=2000:10::f816:3eff:fe8d:a6ef, 10.0.0.102 |
```

```
[root@overcloud-compute-1 nova]# cat /sys/devices/system/node/node*/meminfo  | grep -i
huge
Node 0 AnonHugePages:      2048 kB
Node 0 HugePages_Total:  1024
Node 0 HugePages_Free:      0
Node 0 HugePages_Surp:      0
Node 1 AnonHugePages:      2048 kB
Node 1 HugePages_Total:  1024
Node 1 HugePages_Free:    256
Node 1 HugePages_Surp:      0
```

```
nova boot --nic net-id=$NETID --image cirros --flavor m1.tiny --key-name id_rsa cirros-test0
```

这个实例无法引导：

```
[stack@undercloud-4 ~]$ nova list
+--------------------------------------+--------------+--------+------------+-------------+------------------
----------------------+
| ID                                   | Name         | Status | Task State | Power State | Networks
|
+--------------------------------------+--------------+--------+------------+-------------+------------------
----------------------+
| 1b72e7a1-c298-4c92-8d2c-0a9fe886e9bc | cirros-test0 | ERROR  | -          | NOSTATE     |
|
| a44c43ca-49ad-43c5-b8a1-543ed8ab80ad | cirros-test0 | ACTIVE | -          | Running     |
provider1=2000:10::f816:3eff:fe0f:565b, 10.0.0.105 |
| e21ba401-6161-45e6-8a04-6c45cef4aa3e | cirros-test0 | ACTIVE | -          | Running     |
provider1=2000:10::f816:3eff:fe69:18bd, 10.0.0.111 |
+--------------------------------------+--------------+--------+------------+-------------+------------------
----------------------+
```

5. **从计算节点，检查 NUMA 节点 0 上的空闲巨页是否已耗尽。然而，NUMA 节点 1 有足够的空间：**

```
[root@overcloud-compute-1 qemu]# cat /sys/devices/system/node/node*/meminfo  | grep -i
huge
Node 0 AnonHugePages:      2048 kB
Node 0 HugePages_Total:  1024
Node 0 HugePages_Free:      0
Node 0 HugePages_Surp:      0
Node 1 AnonHugePages:      2048 kB
```

```
Node 1 HugePages_Total:  1024
Node 1 HugePages_Free:    512
Node 1 HugePages_Surp:      0
```

6. **/var/log/containers/nova/nova-compute.log 中的信息显示实例 CPU 固定到 NUMA 节点 0：**

```
<name>instance-00000006</name>
  <uuid>1b72e7a1-c298-4c92-8d2c-0a9fe886e9bc</uuid>
  <metadata>
    <nova:instance xmlns:nova="http://openstack.org/xmlns/libvirt/nova/1.0">
      <nova:package version="14.0.8-5.el7ost"/>
      <nova:name>cirros-test0</nova:name>
      <nova:creationTime>2017-11-23 19:53:00</nova:creationTime>
      <nova:flavor name="m1.tiny">
        <nova:memory>512</nova:memory>
        <nova:disk>8</nova:disk>
        <nova:swap>0</nova:swap>
        <nova:ephemeral>0</nova:ephemeral>
        <nova:vcpus>1</nova:vcpus>
      </nova:flavor>
      <nova:owner>
        <nova:user uuid="5d1785ee87294a6fad5e2bdddd91cc20">admin</nova:user>
        <nova:project uuid="8c307c08d2234b339c504bfdd896c13e">admin</nova:project>
      </nova:owner>
      <nova:root type="image" uuid="6350211f-5a11-4e02-a21a-cb1c0d543214"/>
    </nova:instance>
  </metadata>
  <memory unit='KiB'>524288</memory>
  <currentMemory unit='KiB'>524288</currentMemory>
  <memoryBacking>
    <hugepages>
      <page size='2048' unit='KiB' nodeset='0'/>
    </hugepages>
  </memoryBacking>
  <vcpu placement='static'>1</vcpu>
  <cputune>
    <shares>1024</shares>
    <vcpupin vcpu='0' cpuset='2'/>
    <emulatorpin cpuset='2'/>
  </cputune>
  <numatune>
    <memory mode='strict' nodeset='0'/>
    <memnode cellid='0' mode='strict' nodeset='0'/>
  </numatune>
```

在 numatune 部分中，nodeset="0" 表示内存将被声明从 NUMA 0。

11.3. 解决方案

管理员可以输入实例没有使用的巨页内存量到 nova。

```
[root@overcloud-compute-1 virt]# grep reserved_huge /var/lib/config-data/puppet-
generated/nova_libvirt/etc/nova/nova.conf -B1
[DEFAULT]
reserved_huge_pages=node:0,size:2048,count:512
reserved_huge_pages=node:1,size:2048,count:512
```

size 参数是巨页大小（以 KiB 为单位）。count 参数是每个 NUMA 节点使用的 OVS 巨页数量。例如，对于 Open vSwitch 使用的 4096 个套接字内存，请使用以下值：

```
[DEFAULT]
reserved_huge_pages=node:0,size:1GB,count:4
reserved_huge_pages=node:1,size:1GB,count:4
```

如需了解如何 使用 OpenStack director 实现这一点的详细信息，请参阅如何在 Red Hat OpenStack Platform 10 中设置 reserved_huge_pages。

```
reserved_huge_pages = None

(Unknown) Number of huge/large memory pages to reserved per NUMA host cell.

Possible values:

    A list of valid key=value which reflect NUMA node ID, page size (Default unit is KiB) and number of
pages to be reserved.

    reserved_huge_pages = node:0,size:2048,count:64 reserved_huge_pages =
node:1,size:1GB,count:1

    In this example we are reserving on NUMA node 0 64 pages of 2MiB and on NUMA node 1 1 page
of 1GiB.
```

在 /etc/nova/nova.conf 中启用了 debug 后，您应该在重启 openstack-nova-compute 后在日志中看到以下信息：

```
[root@overcloud-compute-1 virt]# docker restart nova_compute
(...)
[root@overcloud-compute-1 virt]# grep reserved_huge_pages /var/log/containers/nova/nova-
compute.log | tail -n1
2017-12-19 17:56:40.727 26691 DEBUG oslo_service.service [req-e681e97d-7d99-4ba8-bee7-
5f7a3f655b21 - - - - -]
reserved_huge_pages            = [{'node': '0', 'count': '512', 'size': '2048'}, {'node': '1', 'count': '512', 'size':
'2048'}] log_opt_values /usr/lib/python2.7/site-packages/oslo_config/cfg.py:2622
[root@overcloud-compute-1 virt]#
```

第 12 章 使用 PERF 和 COLLECT 对 OVS DPDK PMD CPU 使用情况进行故障排除并发送故障排除数据

1. **先决条件使用本节中的步骤来安装故障排除工具。**

2. **在计算节点上安装 perf ：**

```
yum install perf -y
```

3. **安装 Open vSwitch 调试 RPM:**

```
subscription-manager repos --enable=rhel-7-server-openstack-13-debug-rpms
```

4. **安装 sysstat（需要 pidstat 命令）：**

```
yum install sysstat -y
```

12.1. 诊断

使用本节中的步骤对数据进行故障排除并收集数据。

12.1.1. pmD Threads

1. **确定 PMD 线程的位置：**

```
IFS=$'\n' ; for l in $(ps -T -p `pidof ovs-vswitchd` | grep pmd);do PID=`echo $l | awk '{print
$2}'`; PMD=`echo $l | awk
'{print $NF}'` ; PCPU=`taskset -c -p $PID | awk '{print $NF}'` ; echo "$PMD with PID $PID in
on pCPU $PCPU"; done
```

例如：

```
[root@overcloud-compute-1 ~]# IFS=$'\n' ; for l in $(ps -T -p `pidof ovs-vswitchd` | grep
pmd);do PID=`echo $l | awk
'{print $2}'`; PMD=`echo $l | awk '{print $NF}'` ; PCPU=`taskset -c -p $PID | awk '{print $NF}'`
; echo "$PMD with PID $PID in on pCPU
$PCPU"; done
pmd545 with PID 412314 in on pCPU 2
pmd555 with PID 412315 in on pCPU 4
```

```
pmd550 with PID 412316 in on pCPU 6
pmd551 with PID 412317 in on pCPU 8
pmd553 with PID 412318 in on pCPU 22
pmd554 with PID 412319 in on pCPU 24
pmd549 with PID 412320 in on pCPU 26
pmd556 with PID 412321 in on pCPU 28
pmd546 with PID 412322 in on pCPU 3
pmd548 with PID 412323 in on pCPU 5
pmd547 with PID 412324 in on pCPU 23
pmd552 with PID 412325 in on pCPU 25
```

2. **在重现问题时，请运行 perf 记录和 perf 报告并保存输出。**

- **创建 script gather_perf_data_a.sh ：**

```
cat<<'EOF'>>gather_perf_data_a.sh
#!/bin/bash -x
IFS=$'\n' ;
dir_name=/tmp/perf_record_a
mkdir ${dir_name}
rm -f ${dir_name}/*

for l in $(ps -T -p `pidof ovs-vswitchd` | grep pmd);do PID=`echo $l | awk '{print $2}'`;
PMD=`echo $l | awk '{print $NF}'` ; PCPU=`taskset -c -p $PID | awk '{print $NF}'` ; echo
"$PMD with PID $PID in on pCPU $PCPU"; done > ${dir_name}/pmds.txt

for l in $(ps -T -p `pidof ovs-vswitchd` | grep pmd);do
  PID=`echo $l | awk '{print $2}'`;
  PMD=`echo $l | awk '{print $NF}'` ;
  PCPU=`taskset -c -p $PID | awk '{print $NF}'` ;
  echo "$PMD with PID $PID in on pCPU $PCPU";
  date
  perf record -C $PCPU -g -o perf_record_-g_$PCPU sleep 60 &
done

sleep 80

for l in $(ps -T -p `pidof ovs-vswitchd` | grep pmd);do
  PID=`echo $l | awk '{print $2}'`;
  PMD=`echo $l | awk '{print $NF}'` ;
  PCPU=`taskset -c -p $PID | awk '{print $NF}'` ;
  echo "$PMD with PID $PID in on pCPU $PCPU";
  date
  perf record -C $PCPU -o perf_record_$PCPU sleep 60 &
done

sleep 80

for f in perf_record_-g_*;do
  perf report -g -i $f | cat > ${dir_name}/perf_report_$f.txt ;
  rm -f $f
done
```

```
for f in perf_record_*;do
  perf report -i $f | cat > ${dir_name}/perf_report_$f.txt ;
  rm -f $f
done

archive_name="${dir_name}_`hostname`_`date '+%F_%H%m%S'`.tar.gz"
tar -czf $archive_name ${dir_name}
echo "Archived all data in archive ${archive_name}"
EOF
```

- 运行脚本：

```
chmod +x gather_perf_data_a.sh
./gather_perf_data_a.sh
```

可使用 perf report -i ${archive_name} 读取报告。如果出现这种情况，在红帽支持中打开，请将生成的 tar 归档附加到问题单中。

12.1.2. 附加数据

1. 创建 script gather_perf_data_b.sh 以收集其他数据：

```
cat<<'EOF'>>gather_perf_data_b.sh
#!/bin/bash -x
dir_name=/tmp/perf_record_b
mkdir ${dir_name}
rm -f ${dir_name}/*

date > ${dir_name}/pidstat1.txt
pidstat -u -t -p `pidof ovs-vswitchd`,`pidof ovsdb-server` 5 12 >> ${dir_name}/pidstat1.txt &
perf record -p `pidof ovs-vswitchd` -g --call-graph dwarf sleep 60

sleep 20

date > ${dir_name}/pidstat2.txt
pidstat -u -t -p `pidof ovs-vswitchd`,`pidof ovsdb-server` 1 60 >> ${dir_name}/pidstat2.txt

mv perf.data perf.data_openvswitch

perf script -F tid -i perf.data_openvswitch | sort -u | grep -o '[0-9]*' | xargs -n1 -I{} perf report -i
perf.data_openvswitch --no-children --percentage relative --stdio --tid {} -g none >
${dir_name}/perf_reports.txt
perf script -F tid -i perf.data_openvswitch | sort -u | grep -o '[0-9]*' | xargs -n1 -I{} perf report -i
perf.data_openvswitch --no-children --percentage relative --stdio --tid {}  >
${dir_name}/perf_reports_callgraph.txt

rm -f perf.data_openvswitch
```

```
archive_name="${dir_name}_`hostname`_`date '+%F_%H%m%S'`.tar.gz"
tar -czf $archive_name ${dir_name}
echo "Archived all data in archive ${archive_name}"
EOF
```

2. 执行脚本：

```
chmod +x gather_perf_data_b.sh
./gather_perf_data_b.sh
```

注意

确保有足够的磁盘空间。'perf.data' 文件可占用几个 Gigabytes 磁盘空间。

如果是红帽支持问题单，请将生成的 tar 归档附加到该问题单中。

12.1.3. Open vSwitch 日志

1. 提供所有 Open vSwitch(OVS)日志。确定 /var 有足够的磁盘空间。使用 df -h 确定 /var 和 du -sh /var/log/openvswitch 上的可用磁盘空间，以确定 OVS 日志的总大小。

```
tar -cvzf /var/openvswitch_`hostname`_`date +"%F_%H%M%S"`.tar.gz /var/log/openvswitch
```

2. 将生成的文件（例如 /var/openvswitch_overcloud-compute-0_2018-02-27_153713.tar.gz ）附加到用于分析的支持问题单中。

3. 生成并提供 sosreport。确定 /var 有足够的磁盘空间。使用 df -h 确定 /var 上的可用磁盘空间。

```
sosreport --batch --all-logs
```

第 13 章 在带有 NFV 的虚拟环境中使用 VIRSH 模拟器

使用此流程确定在 Red Hat OpenStack Platform 中使用 virsh 模拟器与 NFV 的影响。

13.1. 症状

您体验 Red Hat OpenStack Platform {vernum} NFV 环境中的数据包丢失，并且尚未配置仿真程序线程固定。

13.2. 解决方案

使用这个部分来调查和配置仿真程序线程固定。

13.2.1. qemu-kvm Emulator Threads

仿真程序线程处理虚拟机硬件模拟的中断请求和非阻塞进程。未运行 vCPU 的线程是 qemu-kvm 仿真程序线程。请参见以下示例。

```
[root@overcloud-compute-0 ~]# ps -Tp `pgrep -f instance-00000009`
   PID    SPID TTY          TIME CMD
364936  364936 ?        00:00:02 qemu-kvm
364936  364946 ?        00:00:00 qemu-kvm
364936  364952 ?        00:00:52 CPU 0/KVM
364936  364953 ?        00:00:26 CPU 1/KVM
364936  364954 ?        00:00:30 CPU 2/KVM
364936  364956 ?        00:00:00 vnc_worker
```

由于 Linux CFS（完全公平调度程序），模拟程序线程通常会定期从一个 pCPU 移到另一个（在 libvirt 的仿真器中定义）中。

在 NFV 环境中，如果您使用 isolcpus 参数配置仿真程序线程时，您可能会遇到问题，因为这种内核配置会禁用这些 CPU 上的 CFS 调度。如果您不使用 isolcpus 参数，可以在仿真程序线程中断正在处理数据包的 CPU 时遇到数据包丢失。

仿真程序线程示例包括：

- qemu-kvm 线程

- vnc_worker 线程
- vhost-<qemu-kvm PID> 内核线程（使用 virtio-net（虚拟机监控程序上的内核网络）

13.2.2. 默认行为线程线程处理

默认情况下，nova 将配置仿真程序线程固定设置，该设置跨越分配给所有 vCPU 的 pCPU。如果您不使用 isolcpus 参数，则可以在任何 pCPU 上调度模拟器线程，并定期从一个 pCPU 移到另一个 CPU。

```
virsh dumpxml instance-0000001d
(...)
  <vcpu placement='static'>4</vcpu>
  <cputune>
    <shares>4096</shares>
    <vcpupin vcpu='0' cpuset='34'/>
    <vcpupin vcpu='1' cpuset='14'/>
    <vcpupin vcpu='2' cpuset='10'/>
    <vcpupin vcpu='3' cpuset='30'/>
    <emulatorpin cpuset='10,14,30,34'/>
  </cputune>
(...)
```

```
[root@overcloud-compute-0 ~]# virsh dumpxml instance-00000009
(...)
       <nova:vcpus>3</nova:vcpus>
  <vcpu placement='static'>3</vcpu>
    <vcpupin vcpu='0' cpuset='1'/>
    <vcpupin vcpu='1' cpuset='2'/>
    <vcpupin vcpu='2' cpuset='3'/>
(...)
<emulatorpin cpuset='1-3'/>
(...)
```

因此，这些 CPU 都可以由 qemu 的仿真程序线程来抢占，从而降低数据包丢失的风险。

有关仿真器线程固定新功能的当前进度，请参阅 Bug 1468004 和 OpenStack Change 510897

在编写本文时，草案指定了以下线程策略：

```
Valid THREAD-POLICY values are:

 - ``share``: (default) The emulator threads float across the pCPUs
   associated to the guest. To place a workload's emulator threads on
```

```
    a set of isolated physical CPUs, set ``share``` and
    ``[compute]/cpu_shared_set`` configuration option to the set of
    host CPUs that should be used for best-effort CPU resources.

  - ``isolate``: The emulator threads are isolated on a single pCPU.
```

13.2.3. 关于在 Emulator Thread scheduling 上 isolcpus 的影响

使用 isolcpus 时，C CFS 调度程序被禁用，所有仿真程序线程都将在第一个可用、最低索引的 pCPU 上运行。因此，如果没有干预或进一步配置，实例的一个 vCPU 会为仿真器线程的资源争用造成高风险。

更多信息请参阅 Kernel.org Bugzilla - Bug 116701。

使用以下算法来确定仿真程序线程使用哪个 vCPU：

```
PCPU=MIN([EMULATORPINSET])
VCPU=REVERSE_CPUSET(PCPU)

REVERSE_CPUSET :=  SELECT pcpu from `virsh dumpxml <instance name> | grep
"cpuset=$PCPU"`
```

例如，在这个实例中，所有仿真程序线程和子项从默认模拟器固定集继承了关联 1-3：

```
[root@overcloud-compute-0 ~]# taskset -a -c -p `pgrep -f instance-00000009`
pid 364936's current affinity list: 1-3
pid 364946's current affinity list: 1-3
pid 364952's current affinity list: 1
pid 364953's current affinity list: 2
pid 364954's current affinity list: 3
pid 364956's current affinity list: 1-3
[root@overcloud-compute-0 ~]# ps -Tp `pgrep -f instance-00000009`
    PID    SPID TTY          TIME CMD
 364936  364936 ?        00:00:02 qemu-kvm
 364936  364946 ?        00:00:00 qemu-kvm
 364936  364952 ?        00:00:51 CPU 0/KVM
 364936  364953 ?        00:00:26 CPU 1/KVM
 364936  364954 ?        00:00:30 CPU 2/KVM
 364936  364956 ?        00:00:00 vnc_worker
[root@overcloud-compute-0 ~]# pgrep -f vhost- | xargs -I {} taskset -a -c -p {}
pid 364948's current affinity list: 1-3
pid 364949's current affinity list: 1-3
pid 364950's current affinity list: 1-3
[root@overcloud-compute-0 ~]#
```

与 isolcpus 相结合，所有仿真程序线程和 vhost-* 线程在 pCPU1 上执行，且永远不会重新调度：

```
cat /proc/sched_debug | sed '/^cpu#/,/^runnable/{//!d}' | grep vhost -C3
(...)
cpu#1, 2099.998 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
     watchdog/1    11        -2.995579    410285     0         0.000000      5025.887998         0.000000 0 /
     migration/1    12         0.000000        79     0         0.000000         3.375060         0.000000 0 /
     ksoftirqd/1    13   5172444.259776        54   120         0.000000         0.570500         0.000000 0 /
     kworker/1:0    14   5188475.472257       370   120         0.000000        14.707114         0.000000 0 /
    kworker/1:0H    15      8360.049510        10   100         0.000000         0.150151         0.000000 0 /
     kworker/1:1  2707   5045807.055876     16370   120         0.000000       793.611916         0.000000
0 /
    kworker/1:1H  2763   5187682.987749     11755   100         0.000000       191.949725
0.000000 0 /
       qemu-kvm 364936      3419.522791     50276   120         0.000000      2476.880384
0.000000 0 /machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/emulator
       qemu-kvm 364946      1270.815296       102   120         0.000000        23.204111         0.000000
0 /machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/emulator
       CPU 0/KVM 364952     52703.660314     53709   120         0.000000     52715.105472
0.000000 0 /machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/vcpu0
      vnc_worker 364956       123.609634         1   120         0.000000         0.016849         0.000000 0
/machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/emulator
    vhost-364936 364948      3410.527677      1039   120         0.000000        84.254772         0.000000
0 /machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/emulator
    vhost-364936 364949      3407.341502        55   120         0.000000         2.894394         0.000000 0
/machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/emulator
    vhost-364936 364950      3410.395220       174   120         0.000000        10.969077         0.000000
0 /machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/emulator

cpu#2, 2099.998 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
     watchdog/2    16        -5.995418    410285     0         0.000000      5197.571153         0.000000 0 /
     migration/2    17         0.000000        79     0         0.000000         3.384688         0.000000 0 /
     ksoftirqd/2    18        -7.031102         3   120         0.000000         0.019079         0.000000 0 /
     kworker/2:0    19         0.119413        39   120         0.000000         0.588589         0.000000 0 /
    kworker/2:0H    20        -1.047613         8   100         0.000000         0.086272         0.000000 0 /
     kworker/2:1  2734   1475469.236026     11322   120         0.000000       241.388582         0.000000
0 /
       CPU 1/KVM 364953     27258.370583     33294   120         0.000000     27269.017017
0.000000 0 /machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/vcpu1

cpu#3, 2099.998 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
     watchdog/3    21        -5.996592    410285     0         0.000000      4970.777439         0.000000 0 /
     migration/3    22         0.000000        79     0         0.000000         3.886799         0.000000 0 /
     ksoftirqd/3    23        -7.035295         3   120         0.000000         0.014677         0.000000 0 /
     kworker/3:0    24        17.758583        38   120         0.000000         0.637152         0.000000 0 /
    kworker/3:0H    25        -1.047727         8   100         0.000000         0.077141         0.000000 0 /
     kworker/3:1 362530    154177.523420        83   120         0.000000         6.544285         0.000000 0
```

```
/
    CPU 2/KVM 364954    32456.061889    25966  120        0.000000    32466.719084
0.000000 0 /machine.slice/machine-qemu\x2d6\x2dinstance\x2d00000009.scope/vcpu2
```

13.2.4. 模拟线程的最佳位置

本节提供了将仿真程序线程放在以下网络中的描述：

- 实例的 DPDK 网络和 Open vSwitch 中的 netdev 数据路径
- 实例中 DPDK 网络，位于 Open vSwitch 中的系统数据路径和虚拟机监控程序上的内核空间网络
- 实例中内核和 Open vSwitch 中的内核网络

13.2.4.1. 使用 DPDK 网络使用 DPDK 网络的最佳部署线程的放置于 Open vSwitch 中的实例和 netdev 数据路径

如果 DPDK 在实例内运行，则为完全在用户空间中进行数据包处理。不要将 PMD 调度到 vCPU0 上运行，因为这应该留给操作系统和中断处理。由于实例中的 PMD CPU 运行活跃循环并且需要 100% 的 CPU，所以不应被抢占。如果其中一个 vCPU 被抢占，则可能会发生数据包丢失。因此，模拟仿真程序需要进行配置，从而使它不会与处理编号为 1 及以上虚拟 CPU 的物理 CPU 重叠。

在实例中使用 DPDK 网络，仿真程序线程的最佳位置是处理 vCPU 0 的 pCPU，也可以是不处理任何虚拟 CPU 的专用物理 CPU。

如果虚拟机监控程序和 DPDK 在实例上使用 OVS-DPDK，请将仿真器线程放在 vCPU 0 上。

13.2.4.2. 使用 DPDK 网络在 Open vSwitch 中的实例和系统数据路径中优化调度线程的放置

如果虚拟机监控程序上使用内核空间网络，则在内核内执行对管理程序的数据包处理。

在实例中使用 DPDK 网络，仿真程序线程的最佳位置是处理 vCPU 0 的 pCPU，也可以是不处理任何虚拟 CPU 的专用物理 CPU。

请注意，在这种情况下，vNIC 队列的数据包处理在 hypervisor 的 vhost-<qemu-kvm PID> 内核

线程中执行。在高流量下，这些内核线程可生成大量 CPU 负载。需要根据情况确定仿真程序线程的最佳位置。

```
[root@overcloud-compute-0 ~]# ps aux | grep vhost-
root     364948  0.0  0.0      0     0 ?        S    20:32   0:00 [vhost-364936]
root     364949  0.0  0.0      0     0 ?        S    20:32   0:00 [vhost-364936]
root     364950  0.0  0.0      0     0 ?        S    20:32   0:00 [vhost-364936]
```

13.2.4.3. 在实例内使用 Kernel Networking 和 Open vSwitch 中的 netdev 数据路径的最佳放置 (Mulator Threads)

在实例中使用内核联网，有两个选项：

- 优化中断分布，例如：实例的 softirqs。在这种情况下，您不必为仿真程序线程分配额外的 pCPU，并可将仿真程序线程分配给不处理任何网络中断的 pCPU。
- 在同一 NUMA 节点上使用一个专用的 pCPU 用于仿真程序线程。

由于第一个选项的复杂性，建议使用第二个选项。

13.3. 诊断

13.3.1. 演示环境

演示环境运行一个实例： instance-0000001d。其关联的 qemu-kvm 线程具有以下 PID：

```
[root@overcloud-compute-0 ~]# pidof qemu-kvm
73517
```

13.3.2. Emulatorpin 的工作原理

默认情况下，Red Hat OpenStack Platform 部署使用以下设置：

```
virsh dumpxml instance-0000001d
(...)
  <vcpu placement='static'>4</vcpu>
  <cputune>
    <shares>4096</shares>
    <vcpupin vcpu='0' cpuset='34'/>
```

```
    <vcpupin vcpu='1' cpuset='14'/>
    <vcpupin vcpu='2' cpuset='10'/>
    <vcpupin vcpu='3' cpuset='30'/>
    <emulatorpin cpuset='10,14,30,34'/>
  </cputune>
(...)
```

这会导致仿真程序线程的无法预计分配，如 qemu-kvm、vnc_worker 等：

```
[root@overcloud-compute-0 ~]# ps -T -p 73517
   PID    SPID TTY          TIME CMD
 73517   73517 ?        00:00:00 qemu-kvm
 73517   73527 ?        00:00:00 qemu-kvm
 73517   73535 ?        00:00:06 CPU 0/KVM
 73517   73536 ?        00:00:02 CPU 1/KVM
 73517   73537 ?        00:00:03 CPU 2/KVM
 73517   73538 ?        00:00:02 CPU 3/KVM
 73517   73540 ?        00:00:00 vnc_worker
[root@overcloud-compute-0 ~]# taskset -apc 73517
pid 73517's current affinity list: 10,14,30,34
pid 73527's current affinity list: 10,14,30,34
pid 73535's current affinity list: 34
pid 73536's current affinity list: 14
pid 73537's current affinity list: 10
pid 73538's current affinity list: 30
pid 73540's current affinity list: 10,14,30,34
```

```
[root@overcloud-compute-0 ~]# virsh vcpupin instance-0000001d | awk '$NF~/[0-9]+/ {print $NF}' |
sort -n | while read CPU; do sed '/cpu#/,/runnable task/{//!d}' /proc/sched_debug | sed -n
"/^cpu#${CPU},/,/^$/p" ; done
cpu#10, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
    migration/10    64         0.000000       107     0         0.000000        90.232791         0.000000 0 /
    ksoftirqd/10    65       -13.045337         3   120         0.000000         0.004679         0.000000 0 /
    kworker/10:0    66       -12.892617        40   120         0.000000         0.157359         0.000000 0 /
   kworker/10:0H    67        -9.320550         8   100         0.000000         0.015065         0.000000 0 /
    kworker/10:1 17996      9695.675528        23   120         0.000000         0.222805         0.000000 0 /
       qemu-kvm 73517      1994.534332     27105   120         0.000000       886.203254         0.000000
0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
       qemu-kvm 73527       722.347466        84   120         0.000000        18.236155         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
      CPU 2/KVM 73537      3356.749162     18051   120         0.000000      3370.045619
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu2
     vnc_worker 73540       354.007735         1   120         0.000000         0.047002         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
         worker 74584      1970.499537         5   120         0.000000         0.130143         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
         worker 74585      1970.492700         4   120         0.000000         0.071887         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
         worker 74586      1982.467246         3   120         0.000000         0.033604         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
```

```
        worker 74587      1994.520768         1   120         0.000000         0.076039         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
        worker 74588      2006.500153         1   120         0.000000         0.004878         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator

cpu#14, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
-------------------------------------------------------------------------------------------------------------
    migration/14    88         0.000000       107     0         0.000000        90.107596         0.000000 0 /
    ksoftirqd/14    89       -13.045376         3   120         0.000000         0.004782         0.000000 0 /
    kworker/14:0    90       -12.921990        40   120         0.000000         0.128166         0.000000 0 /
   kworker/14:0H    91        -9.321186         8   100         0.000000         0.016870         0.000000 0 /
    kworker/14:1 17999      6247.571171         5   120         0.000000         0.028576         0.000000 0 /
       CPU 1/KVM 73536      2274.381281      6679   120         0.000000      2287.691654         0.000000
0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu1

cpu#30, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
-------------------------------------------------------------------------------------------------------------
    migration/30   180         0.000000       107     0         0.000000        89.206960         0.000000 0 /
    ksoftirqd/30   181       -13.045892         3   120         0.000000         0.003828         0.000000 0 /
    kworker/30:0   182       -12.929272        40   120         0.000000         0.120754         0.000000 0 /
   kworker/30:0H   183        -9.321056         8   100         0.000000         0.018042         0.000000 0 /
    kworker/30:1 18012      6234.935501         5   120         0.000000         0.026505         0.000000 0 /
       CPU 3/KVM 73538      2474.183301     12595   120         0.000000      2487.479666
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu3

cpu#34, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
-------------------------------------------------------------------------------------------------------------
    migration/34   204         0.000000       107     0         0.000000        89.067908         0.000000 0 /
    ksoftirqd/34   205       -13.046824         3   120         0.000000         0.002884         0.000000 0 /
    kworker/34:0   206       -12.922407        40   120         0.000000         0.127423         0.000000 0 /
   kworker/34:0H   207        -9.320822         8   100         0.000000         0.017381         0.000000 0 /
    kworker/34:1 18016     10788.797590         7   120         0.000000         0.042631         0.000000 0 /
       CPU 0/KVM 73535      5969.227225     14233   120         0.000000      5983.425363
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu0
```

可使用 virsh 模拟器线程移动模拟程序线程：

```
virsh emulatorpin instance-0000001d 34
```

请注意，所有非 CPU 线程更改的关联性。

```
[root@overcloud-compute-0 ~]# ps -T -p 73517
   PID    SPID TTY          TIME CMD
 73517   73517 ?        00:00:00 qemu-kvm
 73517   73527 ?        00:00:00 qemu-kvm
```

```
  73517   73535 ?        00:00:06 CPU 0/KVM
  73517   73536 ?        00:00:02 CPU 1/KVM
  73517   73537 ?        00:00:03 CPU 2/KVM
  73517   73538 ?        00:00:02 CPU 3/KVM
  73517   73540 ?        00:00:00 vnc_worker
[root@overcloud-compute-0 ~]# taskset -apc 73517
pid 73517's current affinity list: 34
pid 73527's current affinity list: 34
pid 73535's current affinity list: 34
pid 73536's current affinity list: 14
pid 73537's current affinity list: 10
pid 73538's current affinity list: 30
pid 73540's current affinity list: 34
```

请注意在 /proc/sched_debug 中历史数据中的切换数。在以下示例中，PID 73517 已移至 cpu#34。其他仿真程序 worker 没有从最后的输出运行，因此仍然会在 cpu#10 上显示：

```
[root@overcloud-compute-0 ~]# virsh vcpupin instance-0000001d | awk '$NF~/[0-9]+/ {print $NF}' |
sort -n | while read CPU; do sed '/cpu#/,/runnable task/{//!d}' /proc/sched_debug | sed -n
"/^cpu#${CPU},/,/^$/p" ; done
cpu#10, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
    migration/10    64         0.000000       107     0         0.000000        90.232791         0.000000 0 /
    ksoftirqd/10    65       -13.045337         3   120         0.000000         0.004679         0.000000 0 /
    kworker/10:0    66       -12.892617        40   120         0.000000         0.157359         0.000000 0 /
   kworker/10:0H    67        -9.320550         8   100         0.000000         0.015065         0.000000 0 /
    kworker/10:1 17996      9747.429082        26   120         0.000000         0.255547         0.000000 0 /
       qemu-kvm 73527       722.347466        84   120         0.000000        18.236155         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
       CPU 2/KVM 73537      3424.520709     21610   120         0.000000      3437.817166
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu2
     vnc_worker 73540       354.007735         1   120         0.000000         0.047002         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator

cpu#14, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
    migration/14    88         0.000000       107     0         0.000000        90.107596         0.000000 0 /
    ksoftirqd/14    89       -13.045376         3   120         0.000000         0.004782         0.000000 0 /
    kworker/14:0    90       -12.921990        40   120         0.000000         0.128166         0.000000 0 /
   kworker/14:0H    91        -9.321186         8   100         0.000000         0.016870         0.000000 0 /
    kworker/14:1 17999      6247.571171         5   120         0.000000         0.028576         0.000000 0 /
       CPU 1/KVM 73536      2283.094453      7028   120         0.000000      2296.404826         0.000000
0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu1

cpu#30, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
    migration/30   180         0.000000       107     0         0.000000        89.206960         0.000000 0 /
    ksoftirqd/30   181       -13.045892         3   120         0.000000         0.003828         0.000000 0 /
```

```
    kworker/30:0   182       -12.929272        40   120         0.000000         0.120754         0.000000 0 /
   kworker/30:0H   183        -9.321056         8   100         0.000000         0.018042         0.000000 0 /
    kworker/30:1 18012      6234.935501         5   120         0.000000         0.026505         0.000000 0 /
       CPU 3/KVM 73538      2521.828931     14047   120         0.000000      2535.125296
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu3

cpu#34, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
    migration/34   204         0.000000       107     0         0.000000        89.067908         0.000000 0 /
    ksoftirqd/34   205       -13.046824         3   120         0.000000         0.002884         0.000000 0 /
    kworker/34:0   206       -12.922407        40   120         0.000000         0.127423         0.000000 0 /
   kworker/34:0H   207        -9.320822         8   100         0.000000         0.017381         0.000000 0 /
    kworker/34:1 18016     10788.797590         7   120         0.000000         0.042631         0.000000 0 /
       qemu-kvm 73517         2.613794     27706   120         0.000000       941.839262         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
       CPU 0/KVM 73535      5994.533905     15169   120         0.000000      6008.732043
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu0
```

请注意，线程 73517 移到 cpu#34。现在，如果您与 VNC 会话交互，您可以看到 /proc/sched_debug 显示了 cpu#34 上的 vnc_worker 线程。

```
[root@overcloud-compute-0 ~]# virsh vcpupin instance-0000001d | awk '$NF~/[0-9]+/ {print $NF}' |
sort -n | while read CPU; do sed '/cpu#/,/runnable task/{//!d}' /proc/sched_debug | sed -n
"/^cpu#${CPU},/,/^$/p" ; done
cpu#10, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
    migration/10    64         0.000000       107     0         0.000000        90.232791         0.000000 0 /
    ksoftirqd/10    65       -13.045337         3   120         0.000000         0.004679         0.000000 0 /
    kworker/10:0    66       -12.892617        40   120         0.000000         0.157359         0.000000 0 /
   kworker/10:0H    67        -9.320550         8   100         0.000000         0.015065         0.000000 0 /
    kworker/10:1 17996      9963.300958        27   120         0.000000         0.273007         0.000000 0 /
       qemu-kvm 73527       722.347466        84   120         0.000000        18.236155         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
       CPU 2/KVM 73537      3563.793234     26162   120         0.000000      3577.089691
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu2

cpu#14, 2197.477 MHz
runnable tasks:
            task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
----------------------------------------------------------------------------------------------------------
    migration/14    88         0.000000       107     0         0.000000        90.107596         0.000000 0 /
    ksoftirqd/14    89       -13.045376         3   120         0.000000         0.004782         0.000000 0 /
    kworker/14:0    90       -12.921990        40   120         0.000000         0.128166         0.000000 0 /
   kworker/14:0H    91        -9.321186         8   100         0.000000         0.016870         0.000000 0 /
    kworker/14:1 17999      6247.571171         5   120         0.000000         0.028576         0.000000 0 /
       CPU 1/KVM 73536      2367.789075      9648   120         0.000000      2381.099448         0.000000
0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu1

cpu#30, 2197.477 MHz
runnable tasks:
```

```
          task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
-------------------------------------------------------------------------------------------------------------
    migration/30   180         0.000000       107     0         0.000000        89.206960         0.000000 0 /
    ksoftirqd/30   181       -13.045892         3   120         0.000000         0.003828         0.000000 0 /
    kworker/30:0   182       -12.929272        40   120         0.000000         0.120754         0.000000 0 /
   kworker/30:0H   183        -9.321056         8   100         0.000000         0.018042         0.000000 0 /
    kworker/30:1 18012      6234.935501         5   120         0.000000         0.026505         0.000000 0 /
       CPU 3/KVM 73538      2789.628278     24788   120         0.000000      2802.924643
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu3

cpu#34, 2197.477 MHz
runnable tasks:
          task   PID         tree-key  switches  prio     wait-time             sum-exec        sum-sleep
-------------------------------------------------------------------------------------------------------------
    migration/34   204         0.000000       107     0         0.000000        89.067908         0.000000 0 /
    ksoftirqd/34   205       -13.046824         3   120         0.000000         0.002884         0.000000 0 /
    kworker/34:0   206       -12.922407        40   120         0.000000         0.127423         0.000000 0 /
   kworker/34:0H   207        -9.320822         8   100         0.000000         0.017381         0.000000 0 /
    kworker/34:1 18016     11315.391422        25   120         0.000000         0.196078         0.000000 0 /
        qemu-kvm 73517       471.930276     30975   120         0.000000      1295.543576         0.000000
0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
       CPU 0/KVM 73535      6160.062172     19201   120         0.000000      6174.260310
0.000000 0 /machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/vcpu0
      vnc_worker 73540       459.653524        38   120         0.000000         7.535037         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
          worker 78703       449.098251         2   120         0.000000         0.120313         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
          worker 78704       449.131175         3   120         0.000000         0.066961         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
          worker 78705       461.100994         4   120         0.000000         0.022897         0.000000 0
/machine.slice/machine-qemu\x2d1\x2dinstance\x2d0000001d.scope/emulator
```