PERFORMANCE OPTIMIZATION IN RED HAT OPENSTACK PLATFORM

LUNCH & LEARN

devs openstack summit users

Razique Mahroua

Red Hat Training - Services Content Architect

redhat

ABOUT ME

- Course author of the Red Hat OpenStack Administration courses (CL110, CL210, CL310).
- Services content architect for Red Hat Training since 2014.
- Worked on the majority of the cloud portfolio (Red Hat OpenStack Platform, OpenShift Container Platform, CloudForms, RHV, etc.)
- Published some whitepapers for IBM and Amazon


rmahroua@redhat.com

GitHub https://github.com/rmahroua


www.linkedin.com/in/rmahroua

AGENDA

1. DVR Architecture
2. Network Components
3. Routing Flows
4. Instances (CPU pinning, host aggregates, hugepages, and filters)

1: DVR Architecture

NETWORK NODES PROVIDE

- IP forwarding
  - Inter-subnet
  - Floating IP
  - Default SNAT
- Metadata agent
  - Access to the Nova metadata agent

DVR: AN INTRODUCTION

ISSUES

- Scalability
- Single point of failure
- Performance bottleneck

DVR: AN INTRODUCTION

DISTRIBUTED ROUTING IN NEUTRON

- Compute nodes provide IP forwarding for local VMs & metadata agents for VMs (Inter-subnet) and floating IPs

BENEFITS

- Bypasses the network nodes for better performance
- Scales with the number of compute nodes
- Limited domain failure
- Limitation: *the default SNAT function is still centralized*

DVR: AN INTRODUCTION

Neutron Distributed Virtual Routing is a routing model that places L3 routers directly onto compute nodes

- **Enables both inter-project instance traffic and external traffic** to be directly routed without traversing the controller nodes.

- **Implements a floating IP namespace on each compute node** where VMs are running, providing source network address translation (SNAT) behavior for private VMs.

DVR: AN INTRODUCTION

- Introduced as a technology preview feature in Red Hat OpenStack Platform 6, **DVR is fully supported with Red Hat OpenStack Platform 10**.
- Is an optional feature, typical installations default to legacy, centralized routing.

2: Nodes Components

CONTROLLER NODE COMPONENTS

- **Open vSwitch Agent** manages virtual switches, connectivity among them, and interaction over virtual ports with other networking components (namespaces, Linux bridges, and physical interfaces).
- **DHCP Agent** manages DHCP network namespaces (provides IP allocations for instances).
  - The DHCP agent starts dnsmasq processes to manage IP address allocation.

CONTROLLER NODE COMPONENTS

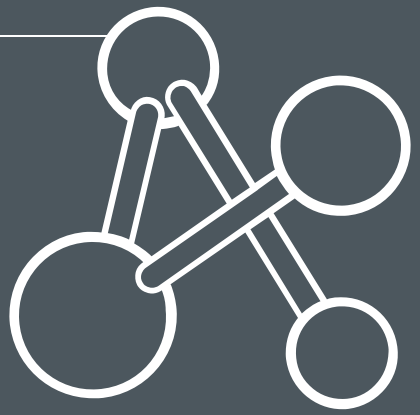
- **L3 Agent** manages the **qrouter** and SNAT namespaces.
  - qrouter NS routes north-south and east-west network, performs DNAT and SNAT, routes metadata traffic between instances and the metadata agent.
  - SNAT namespaces perform SNAT for north-south network traffic for instances with fixed IP address and project networks on distributed routers.

- **Metadata Agent** processes metadata operations for instances using project networks on legacy routers.

COMPUTE NODE COMPONENTS

- **Open vSwitch Agent**
- **DHCP Agent**
- **L3 Agent**
- **Metadata agent**
- **Linux Bridge**: the Neutron service uses a Linux bridge to manage security groups for instances.

3: Routing Flows & DVR



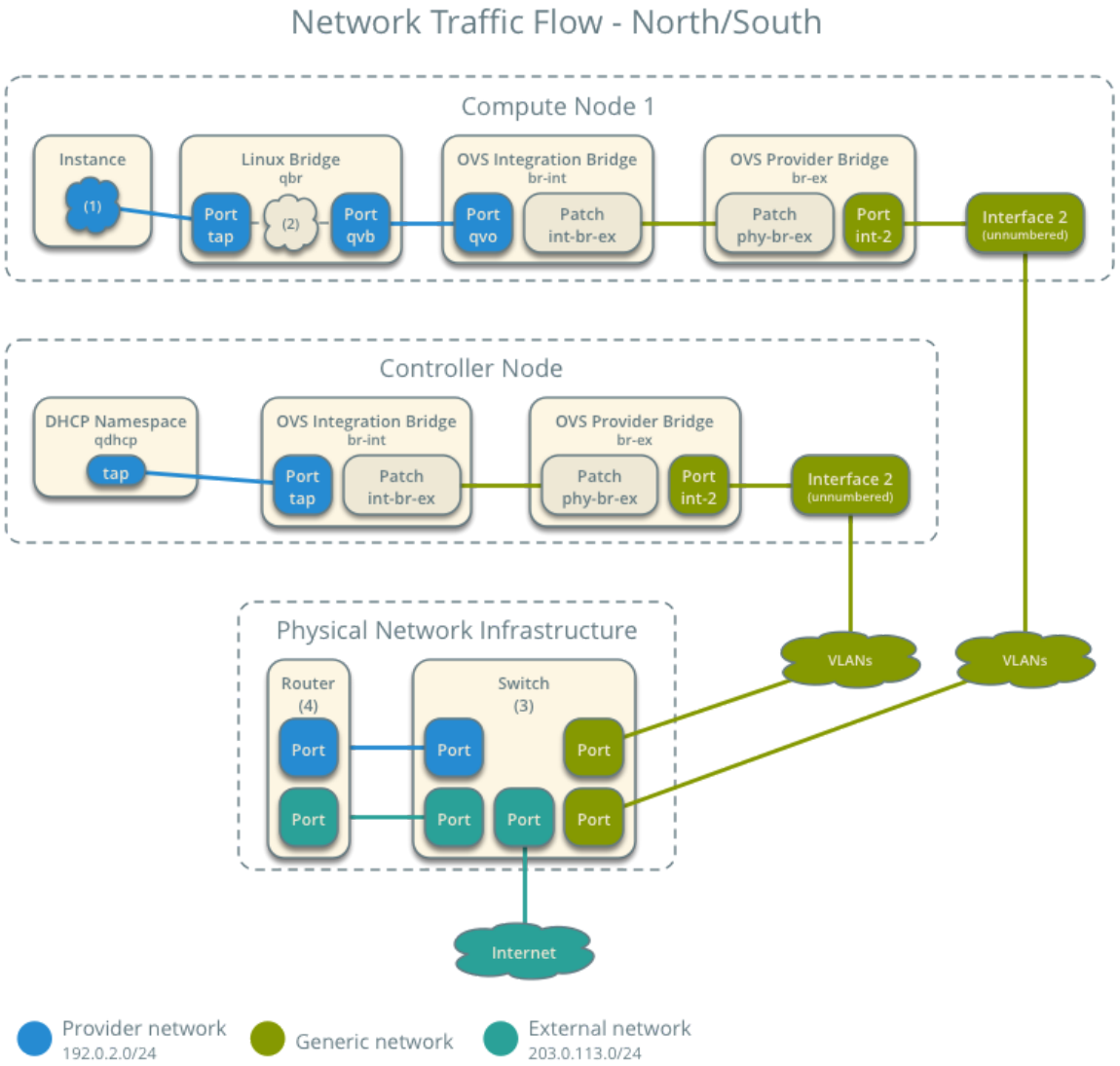
ROUTING TRAFFIC FLOWS

- Routing services in OpenStack can be categorized into: distributed (**DVR**) and centralized (**legacy**), and either traffic between VMs within an environment (*east-west*) or between a VM and systems external to the OpenStack installation (*north-south*).

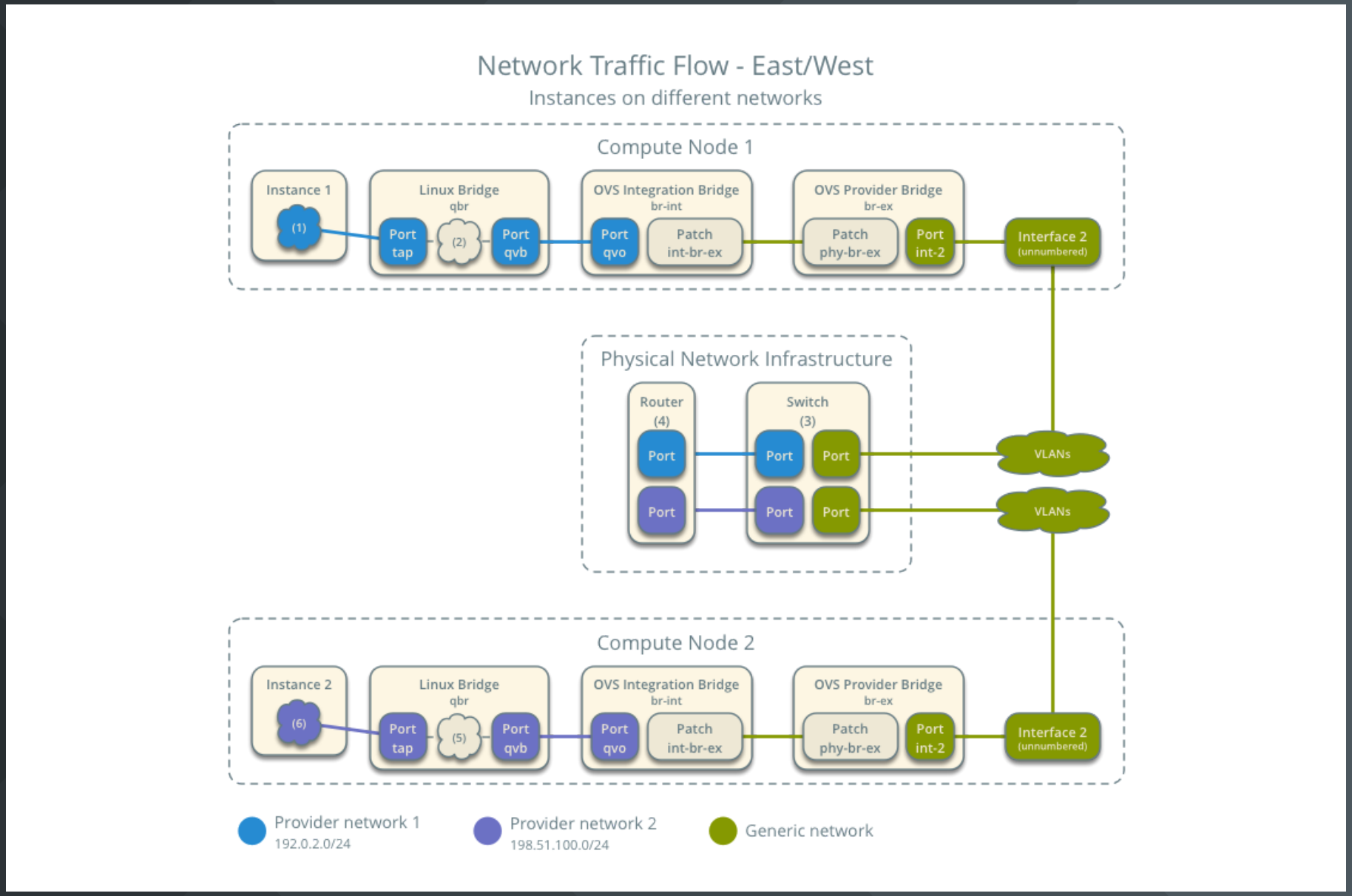
- East-west non-DVR (legacy) traffic is routed between different subnets, IPv4 or IPv6, in the same project or between subnets of different projects (requires legacy routers).

- *Such traffic remains within OpenStack nodes and does not traverse external networks.*

ROUTING TRAFFIC FLOWS

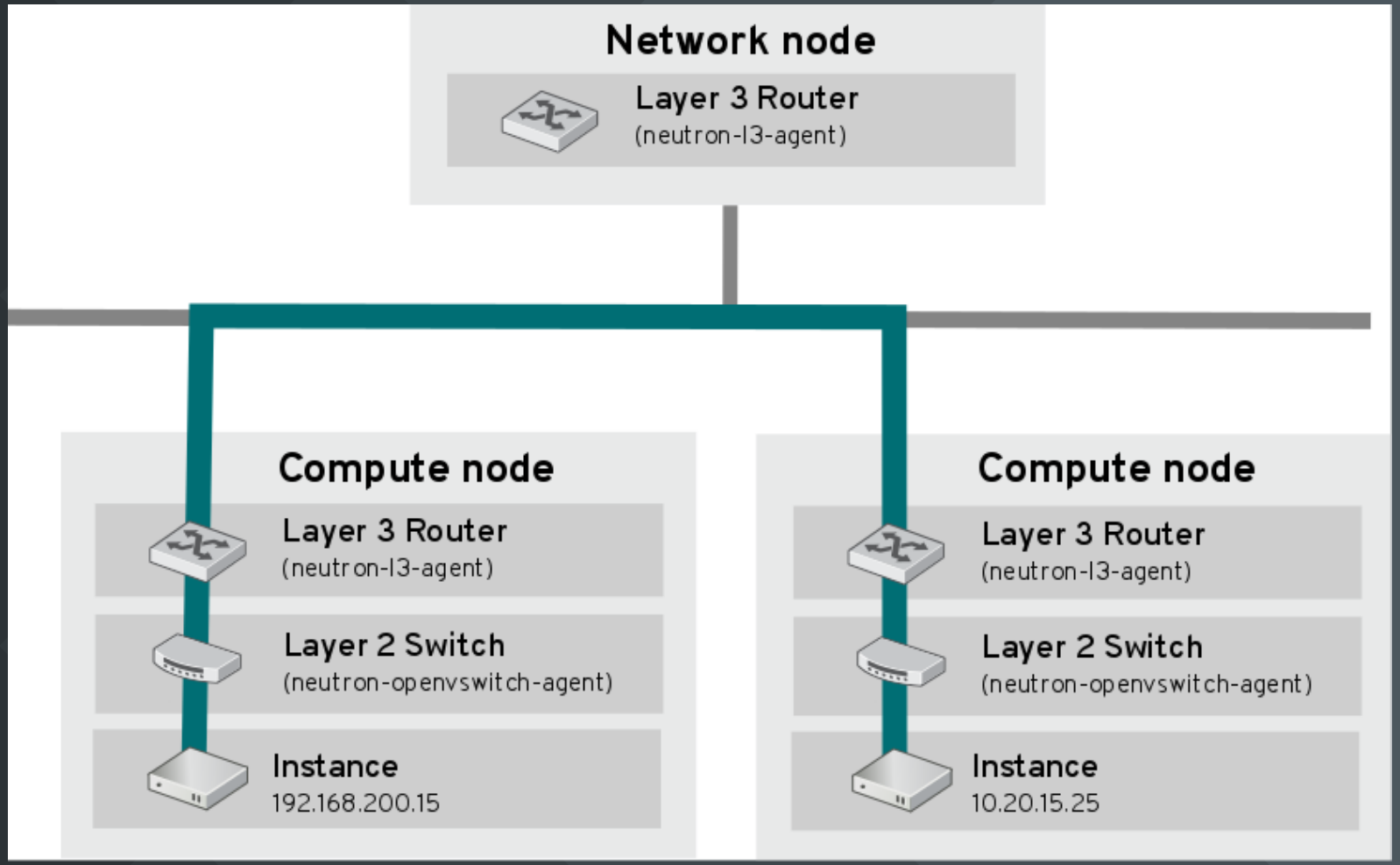


ROUTING TRAFFIC FLOWS



ROUTING TRAFFIC FLOWS

In the diagram below, the instances on separate subnets are able to communicate, without routing through the network node:



ROUTING TRAFFIC FLOWS

- East-west DVR traffic is routed directly from a compute node in a distributed design (bypasses controller nodes).

- North-south non-DVR (legacy) traffic with floating IPs is **a one-to-one association between the floating IP and an instance's fixed address**, implemented by IPv4 NAT tables on a legacy networking service router.
  - Instances communicate with external resources using floating IPs reserved from the provider network.
  - Instances configured with IPv6 use routable *Global Unicast Addresses* (**GUAs**), precluding the need for address overlap management, and are routed without needing NAT.

ROUTING TRAFFIC FLOWS

- North-south DVR traffic with floating IPs is distributed and routed directly from the compute nodes (requires external provider network connectivity on every compute node).

- North-south non-DVR (legacy) traffic without floating IPs is handled by a *port address translation* (**PAT**) service, enabling instances to initiate bidirectional traffic to external systems.

ROUTING TRAFFIC FLOWS

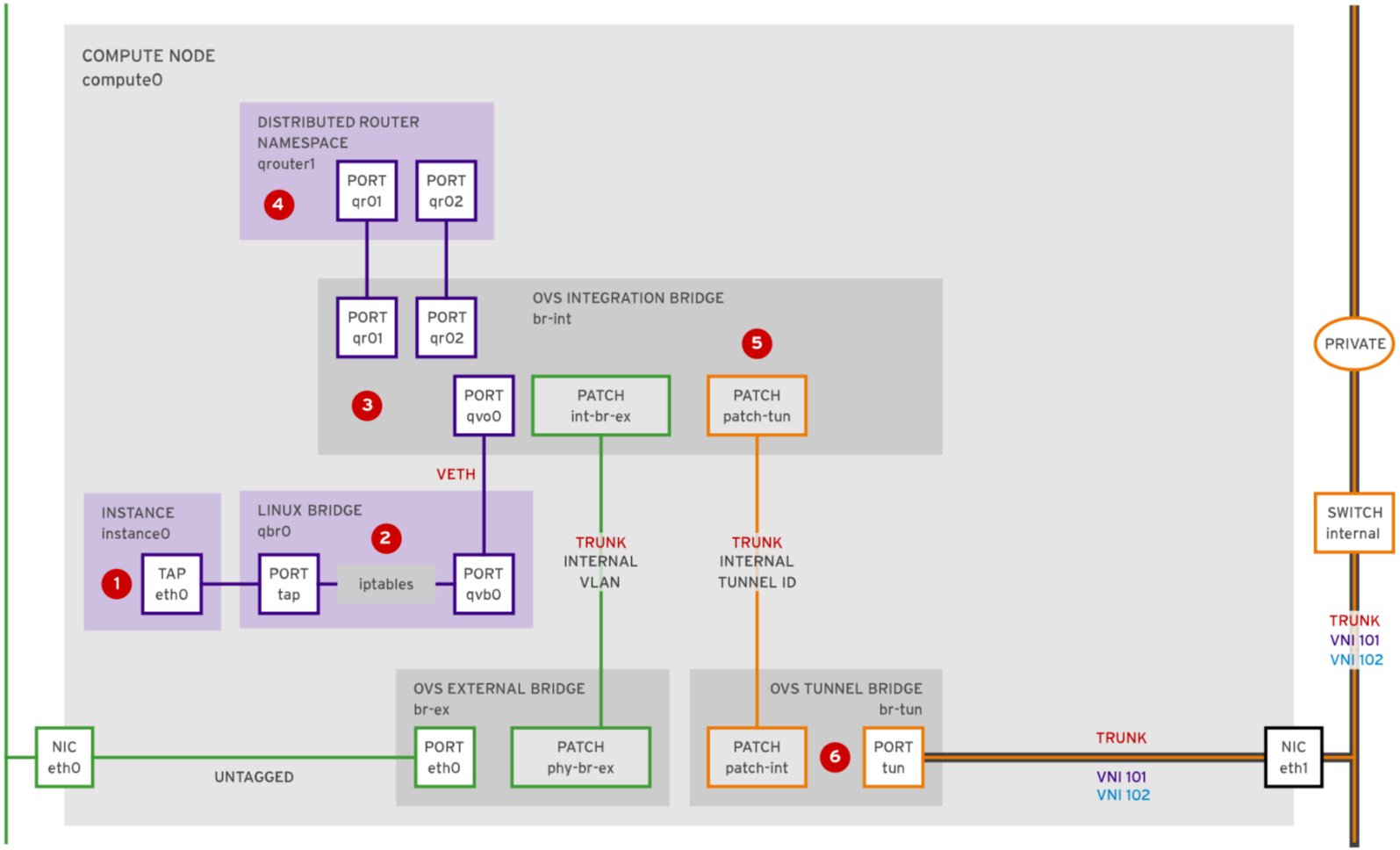
- Externally initiated traffic must traverse the networking service (controller) nodes for proper firewall filtering and route addressing to locate instances on their compute host.
  - SNAT applies to IPv4 traffic only.

- North-south DVR traffic without floating IPs (DVR) is not distributed, and requires a dedicated Neutron Service (controller) node.

EAST-WEST PACKET FLOW

- Compute nodes route east-west network traffic between project networks on an outbound router, bypassing controller node processing, using the instances' fixed and floating IP addresses.

EAST-WEST PACKET FLOW

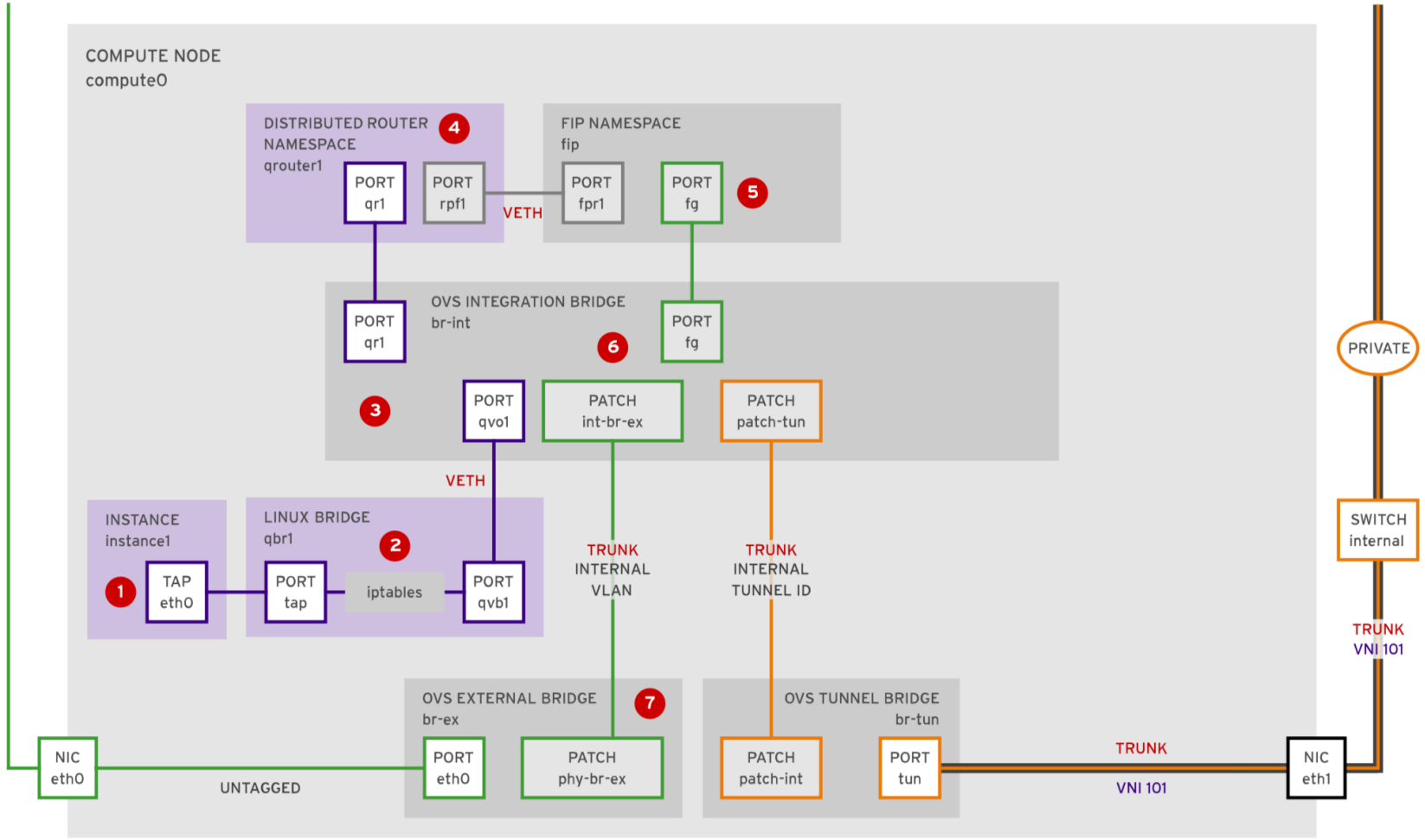


East-west flow in two subnets across two hosts - outbound

NORTH-SOUTH PACKET FLOW

- Outbound from any VM on any compute host with FIP processing.
- In this scenario, outbound packets from an instance are routed and processed for network address translation, then exit the external interface.

NORTH-SOUTH PACKET FLOW



North-south flow on one subnet with FIP handling - outbound

CONFIGURING DVR

DVR can be configured using an YAML file (environment file). The **neutron-ovs-dvr.yaml** file is used to configure the required parameters for DVR.

REQUIREMENTS

Both compute and controller nodes must have an interface connected to the physical network for external network traffic.

A bridge is required on compute and controller nodes, with an interface for external network traffic.

The configured bridge must be allowed in the Neutron configuration for the bridge to be used.

CONFIGURING DVR

There are two files which contain a value that must match. The **environments/neutron-ovs-dvr.yaml** file contains a value for **OS::TripleO::Compute::Net::SoftwareConfig**, which must match the value for the same variable contained in the environment file used to deploy the overcloud.

Without this configuration, the appropriate external network bridge for the compute node's L3 agent is not created.

CONFIGURING DVR

Configure a neutron port for the compute node on the external network by modifying **OS::TripleO::Compute::Ports::ExternalPort**

Include the the path to **external.yaml**.

- **OS::TripleO::Compute::Ports::ExternalPort: ../network/ports/external.yaml**

Include **neutron-ovs-dvr.yaml** as an environment file when deploying the overcloud.

Ensure that L3 HA is disabled.

MANAGING DVR ROUTERS

CONSIDERATIONS

- The only backends supported for DVR are the **ML2 core plug-in** and the **Open vSwitch mechanism driver**.

- *IPv6 traffic is not distributed.* If you use IPv6, avoid DVR at this time (all IPv6 routed traffic traverses the centralized controller node).

- DVR is not supported with L3 HA. Routers are still scheduled from the controller nodes, but in the event of an L3 agent failure, all routers hosted by that agent also fail.
  - Use of the **allow_automatic_l3agent_failover** flag so that routers are rescheduled to a different node should a network node fail.

MANAGING DVR ROUTERS

To list the routers

```
$ openstack router show my-dvr-router
...output omitted...
| distributed | True |
...output omitted...
| id | c7ab6809-1fee-40e9-8891-eb3816bdeb6f |
```

To view the ports for the router

```
$ openstack port list --router my-dvr-router -c 'ID' -c 'Fixed IP Addresses' -f json
[{
...
  "ID": "0db330bc-5eed-4788-a6cf-3fa87073274a"
},{
...
  "ID": "75e6b190-3532-471e-be3e-2620be7263ca" },{
...
  "ID": "e2ea1c33-398f-4528-878f-d3d5fbe3c6e2"}]
```

MANAGING DVR ROUTERS

To retrieve the IP of the router

```
$ openstack port show 0db330bc-5eed-4788-a6cf-3fa87073274a
+-----------------------+-----------------------------------------+
| Field                 | Value                                   |
+-----------------------+-----------------------------------------+
...output omitted...
| device_owner          | network:router_gateway                  |
| extra_dhcp_opts       |                                         |
| fixed_ips             | ip_address='172.25.250.190', subnet_id  |
|                       | ='a879632b-b2e4-4824-99fc-              |
|                       | cae727451ba8'                           |
|                       | ip_address='2001:db8::190',             |
|                       | subnet_id='f5982da9-b7dd-               |
|                       | 495b-8465-6d486e1cb717'                 |
| id                    | c7ab6809-1fee-40e9-8891-eb3816bdeb6f    |
| mac_address           | fa:16:3e:0d:7d:a9                       |
...output omitted...
+-----------------------+-----------------------------------------+
```

MANAGING DVR ROUTERS

To view a router from a compute node

```
[root@compute0 ~]# sudo ip netns
qrouter-c7ab6809-1fee-40e9-8891-eb3816bdeb6f
```

To list the addresses

```
[root@compute0 ~]# sudo ip -n qrouter-c7ab6809-1fee-40e9-8891-eb3816bdeb6f addr show
...
15: qr-e2eb2c44-29: BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1446 qdisc noqueue state UNKNOWN qlen 1000
link/ether fa:16:3e:86:38:44 brd ff:ff:ff:ff:ff:ff
inet 192.168.2.1/24 brd 192.168.2.255 scope global qr-e2ea1c33-39 valid_lft forever preferred_lft forever
inet6 fe80::f816:3eff:fe86:3844/64 scope link
valid_lft forever preferred_lft forever
```

MANAGING DVR ROUTERS

The **br-int** OVS integration bridge forwards packets to the project network interface (**qr-*XXX***) in the **qrouter-*c7ab6809-XXX*** distributed router namespace.

```
[root@compute0 ~]# ovs-vsctl list-ports br-int int-br-ex
patch-tun
qr-e2ea1c33-39
qvocb7f8c98-1c
```

To list the route tables that exist in the **qrouter** namespace from the compute node.

```
[root@compute0 ~]# ip -n qrouter-c7ab6809-1fee-40e9-8891-eb3816bdeb6f \
rule list 0: from all lookup local
32766: from all lookup main
32767: from all lookup default
3232236033: from 192.168.2.1/24 lookup 3232236033
```

MANAGING DVR ROUTERS

To view the main route table that forwards the packets for east-west communication between instances (the main route table's output is the same as that of **ip route show**). The default route table is empty.

```
[root@compute0 ~]# ip -n qrouter-c7ab6809-1fee-40e9-8891-eb3816bdeb6f \
route show table main
192.168.2.0/24 dev qr-e2ea1c33-39 proto kernel scope link src 192.168.2.1
```

The **3232236033** route table forwards the packets to the SNAT namespace on controller0.

3232236033 is the route table name listed earlier

192.168.1.9 is the SNAT interface IP (router interface in the subnet).

```
[root@compute0 ~]# ip -n qrouter-c7ab6809-1fee-40e9-8891-eb3816bdeb6f \
route show table 3232236033 default via 192.168.2.9 dev qr-e2ea1c33-39
```

MANAGING DVR ROUTERS

To list the SNAT network namespace and the interfaces in the namespace from the controller. The **snat-c7ab6809-XXX** namespace is the distributed router SNAT namespace. The ID associated with the SNAT namespace name is equivalent to the project DVR router ID, **my-dvr-router**.

```
[root@controller0 ~]# ip netns list
...output omitted...
snat-c7ab6809-1fee-40e9-8891-eb3816bdeb6f
```

To list the interfaces in the **snat-c7ab6809-XXX** namespace.

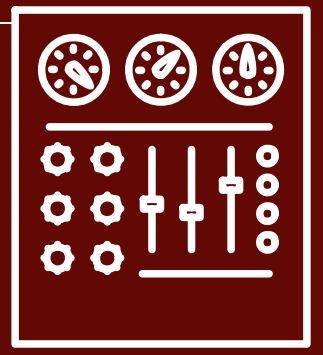
```
[root@controller0 ~]# ip -n snat-c7ab6809-1fee-40e9-8891-eb3816bdeb6f address
...output omitted...
44: sg-75e6b190-35: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1446 qdisc noqueue
inet 192.168.2.9/24 brd 192.168.2.255 scope global sg-75e6b190-35
...
47: qg-0db330bc-5e: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1496 qdisc noqueue state UNKNOWN qlen 1000
...
inet 172.25.250.190/24 brd 172.25.250.255 scope global qg-0db330bc-5e
```

MANAGING DVR ROUTERS

To view the IP tables rule that translates the source IP address of the instance to the IP address assigned to the external gateway interface.

```
[root@controller0 ~]# ip netns exec snat-c7ab6809-1fee-40e9-8891-eb3816bdeb6f \
iptables -t nat --line-numbers -nL neutron-l3-agent-snat
Chain neutron-l3-agent-snat (1 references)
num target  prot opt source     destination
1   SNAT    all  --  0.0.0.0/0  0.0.0.0/0   to:172.25.250.190
2   SNAT    all  --  0.0.0.0/0  0.0.0.0/0   mark match ! 0x2/0xffff ctstate \
 DNAT to:172.25.250.190
```

4: Tuning Cloud Apps



PERFORMANCE OPTIMIZATION

- CPU pinning
- Host aggregates
- Hugepages
- Filters

CPU PINNING

CPU pinning refers to reserving physical cores for specific virtual guest instances.

The concept is also referred to as *CPU isolation* or *processor affinity*.

The configuration is in two parts:

1. Ensuring that virtual guests can only run on dedicated cores.
2. Ensuring that common host processes do not run on those cores.

The term pinning refers to the 1-to-1 mapping of a physical core to a guest vCPU.

CPU PINNING

In OpenStack, CPU pinning establishes a mapping between a virtual CPU and a physical core.

With CPU pinning, the instance's virtual CPU processes always run on the same physical core and NUMA node.

This improves performance (ensures that memory access to memory is always local in the NUMA topology).

NUMA TOPOLOGY

On i386 and x86-64 systems, all memory is equally accessible and shared by all CPUs. Access times are the same to any memory address, for any CPU performing a memory operation.

CPUs are connected by a shared front-side bus to main memory to coordinate access requests.

This memory architecture is called *Uniform Memory Access* (UMA).

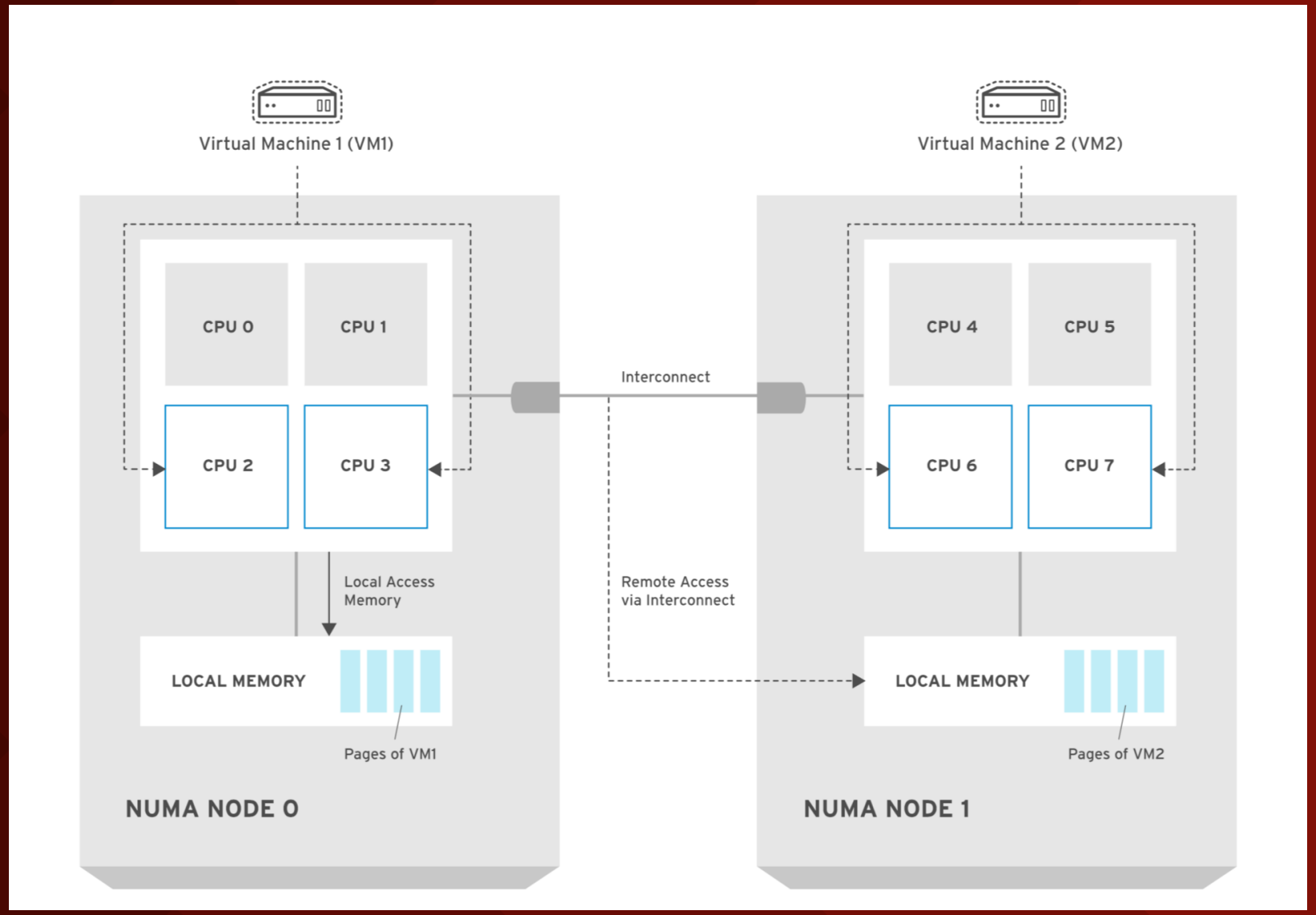
On larger systems, system memory is configured in banks connected directly to particular CPUs or sockets.

NUMA TOPOLOGY

The local CPU is responsible for controlling that bank, in addition to having exclusive access to that memory.

In a Non-Uniform Memory Access (NUMA) system, memory access is faster for the local CPU. Memory addresses not found in the local bank generates a memory miss, requiring the data to be copied from the remote bank, which is slower.

NUMA TOPOLOGY



CONFIGURING CPU PINNING

1: View the NUMA topology for the compute node.

2: On compute node, set **vcpu_pin_set** in the **/etc/nova/nova.conf** to a range of physical CPU cores to be reserved for virtual machine processes.

The OpenStack Compute service will ensure that virtual machine instances are pinned to these physical CPU cores.

In the following example, the compute host reserves two cores in each NUMA node. The 2nd and 3rd core from NUMA node0 and the 6th and 7th core from NUMA node1.

```
vcpu_pin_set=2-3,6-7
```

CONFIGURING CPU PINNING

3: On the compute node, set **reserved_host_memory_mb** to reserve RAM for common host processes.

In the following example, the value is set to 512 MB.

Restart the **openstack-nova-compute service**.

```
reserved_host_memory_mb=512
```

CONFIGURING CPU PINNING

4: Host processes should not run on the CPU cores reserved for virtual machine processes. Ensure CPU isolation by adding the **isolcpus** argument to kernel boot arguments.

Use the same range of CPU cores reserved for virtual machine processes in **/etc/nova/nova.conf** as the **isolcpus** parameter.

Edit **/etc/default/grub** to add the isolcpus parameter and rebuild the grub.cfg file using **grub2-mkconfig**.

```
[root@compute ~]# cat /etc/default/grub
GRUB_TIMEOUT=1
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="...isolcpus=2-3,6-7"
GRUB_DISABLE_RECOVERY="true"
[root@compute ~]# grub2-mkconfig > /boot/grub2/grub.cfg
[root@compute ~]# reboot
```

HOST AGGREGATES

Host aggregates is a method for grouping hypervisor hosts based on configurable metadata.

For example, hosts may be grouped based on hardware features, capabilities or performance characteristics (for example CPU pinning).

Host aggregates are not visible by users, but are used automatically for instance scheduling. A compute node can be included in multiple host aggregates.

HOST AGGREGATES

To specify the required features requested for an instance deployment, an administrator builds a flavor with extra specifications to match to available host's aggregate metadata.

At deployment, the compute scheduler matches the request declared in the flavor by scheduling the provisioning on a compute host in an adequate host aggregate.

HOST AGGREGATES

1: Create the **performance** host aggregate for hosts that will receive

```
+-------------------+----------------------------+
| Field             | Value                      |
+-------------------+----------------------------+
| availability_zone | None                       |
| created_at        | 2018-02-22T10:22:25.878358 |
| deleted           | False                      |
| deleted_at        | None                       |
| id                | 1                          |
| name              | performance                |
| updated_at        | None                       |
+-------------------+----------------------------+
```

2: Set the properties on the performance aggregate, used to match the extra specification

```
[user@demo ~(admin)]# openstack aggregate set --property pinned=true performance
```

HOST AGGREGATES

3: Set the properties on the a default aggregate to not use CPU pinning.

```
[user@demo ~(admin)]# openstack aggregate set --property pinned=false default
```

4: Create a new flavor for instances requiring NUMA topology and CPU

```
[user@demo ~(admin)]# openstack flavor create --ram 2048 \
--disk 10 --vcpus 2 demo-flavor
+----------------------------+--------------------------------------+
| Field                      | Value                                |
+----------------------------+--------------------------------------+
| OS-FLV-DISABLED:disabled   | False                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                    |
| disk                       | 10                                   |
| id                         | e2af23bc-286b-4382-8b76-0894e3a070d8 |
| name                       | demo-flavor                          |
| os-flavor-access:is_public | True                                 |
...
```

HOST AGGREGATES

5: Set the flavor to use extra specifications and values.

```
[user@demo ~(admin)]#  openstack flavor set \
--property aggregate_instance_extra_specs:pinned=true \
--property hw:cpu_policy=dedicated \
--property hw:cpu_thread_policy=prefer \
demo-flavor
```

Set **hw:cpu_policy** to **dedicated** to force instance vCPUs to be pinned to a set of host physical CPUs.

The default for **hw:cpu_policy** is **shared** for vCPUs to float to any host physical CPUs. The **hw:cpu_thread_policy** option is only valid if **hw:cpu_policy** is set to **dedicated**.

Setting **hw:cpu_thread** to **prefer** will have thread siblings pinned together.

6: Add the compute node to host to the performance and default aggregates.

```
[user@demo ~(admin)]#  openstack aggregate add host performance compute0.overcloud.example.com
[user@demo ~(admin)]#  openstack aggregate add host default compute0.overcloud.example.com
```

HUGEPAGES

Modern computer systems use paging to securely and flexibly manage system memory. Memory on a computer system is organized into fixed-size chunks called **pages**.

Processes do not address physical memory directly. Each process has a virtual address space. In a Linux system, physical memory is mapped to virtual addresses for process use.

The process of looking up a page mapping on a hierarchical page table can be expensive. Therefore, when a mapping from a virtual address to a physical address is looked up in the page table, it is cached in dedicated hardware called the *Translation Look-aside Buffer*, or TLB.

HUGEPAGES

The number of TLB entries for a processor is fixed, but with a larger page size, the referenced TLB space for a processor becomes correspondingly larger.

Having fewer TLB entries that point to more memory means that a TLB hit is more likely to occur.

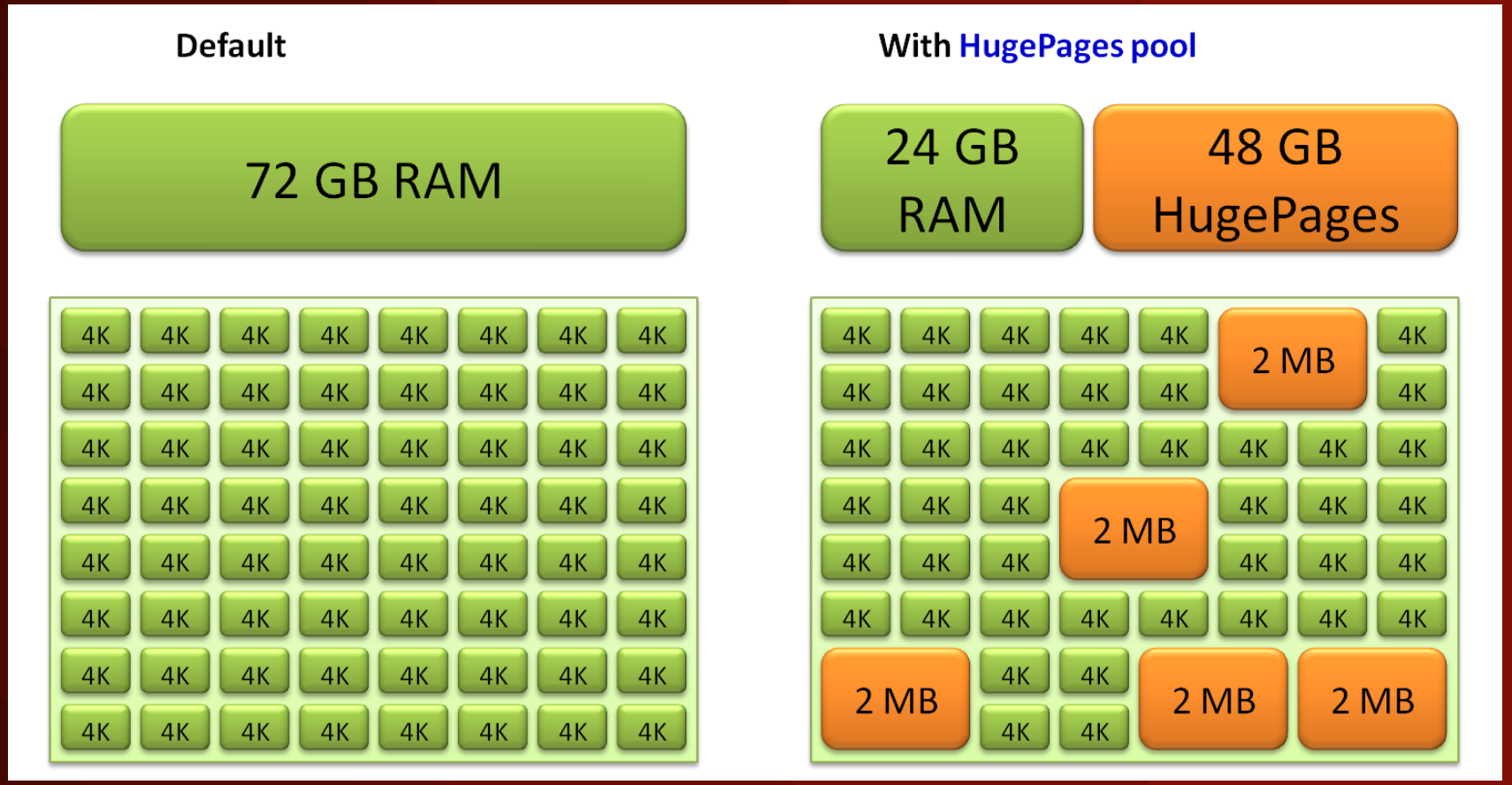
HUGEPAGES

To assist performance for processes accessing a large (but contiguous) amount of memory frequently (databases), the processor architectures support a feature called *hugepages*.

This allows memory allocation with a larger fixed page size (2 MiB to 1 GiB) to more efficiently use the TLB.

Drawbacks: Hugepages force large blocks of contiguous memory to be reserved and allocated to a process as a single unbreakable chunk.

HUGEPAGES



Source: lilinuxkernel.com

HUGEPAGES

Configure the **default_hugepagesz**, **hugepagesz**, and **hugepages** kernel arguments with appropriate values and reboot the node.

```
[root@compute0 ~]# cat /etc/default/grub
...
GRUB_CMDLINE_LINUX="...default_hugepagesz=2M hugepagesz=2M hugepages=2048"
```

HUGEPAGES

OpenStack instances do not use hugepages by default, unless explicitly requested. They can be a big performance improvement for large critical use cases such as NFV applications.

Hugepages are requested explicitly through flavor *extra specifications* or *metadata* attached to an image.

To use hugepages set the **hw:mem_page** flavor extra specification:

```
[user@demo ~(demo)]$ openstack flavor set default --property hw:mem_page_size=2048
```

HUGEPAGES

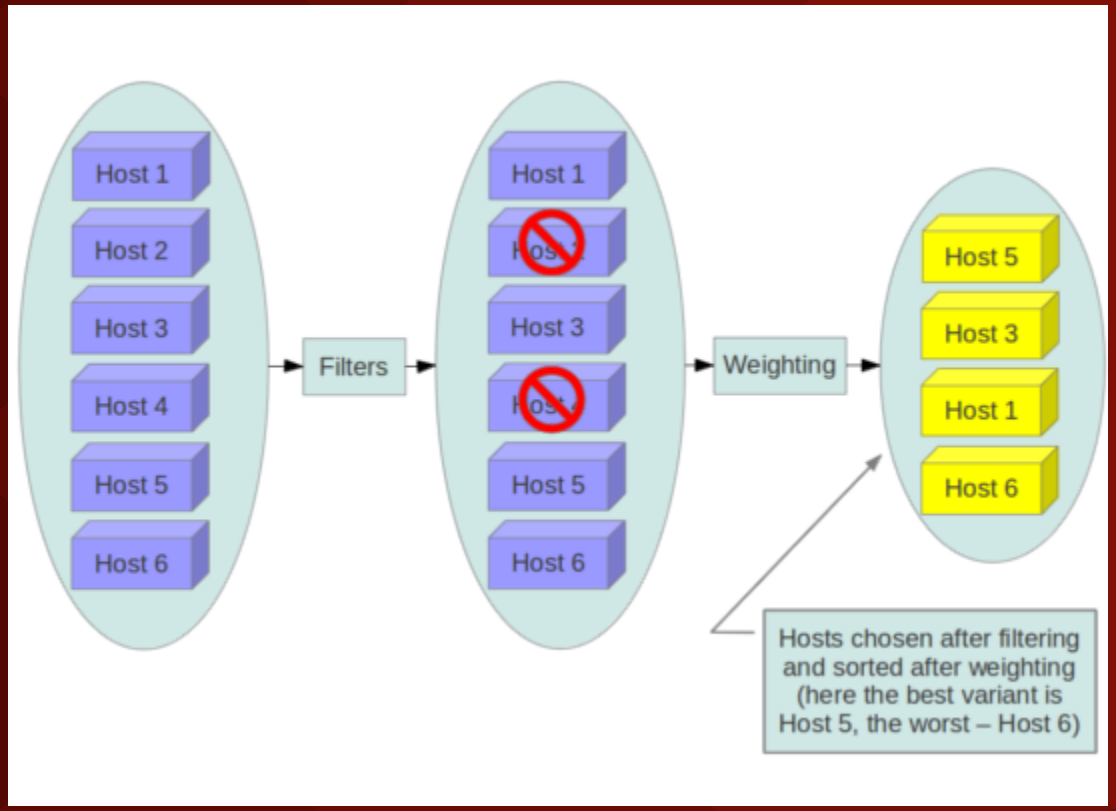
Use the **hw:mem_page_size** property to request a supported hugepage size. With no unit suffix, defaults to Kilobytes.

- Setting the **hw:mem_page_size** property to **large** requests to only use larger page sizes (either an architecture-dependent 2MB or 1GB).
- Similarly, the default of **small** specifies to only use the small page sizes of 4Kb.
- Setting **hw:mem_page_size** to **any** allows the compute driver implementation to decide the hugepage size to use.

FILTERS

The Filter Scheduler for Nova supports filtering and weighting to make informed decisions on where a new instance should be created.

This Scheduler is only supported by compute nodes.



Source: docs.openstack.org

FILTERS

Compute is configured with the following default scheduler options in the **/etc/nova/nova.conf** file:

```
scheduler_driver=nova.scheduler.multi.MultiScheduler
scheduler_driver_task_period=60
compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
scheduler_available_filters=nova.scheduler.filters.all_filters
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,\
ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,\
ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter
```

FILTERS

This configuration considers hosts that meet the following criteria:

- Have not been attempted for scheduling purposes (**RetryFilter**).
- Are in the requested availability zone (**AvailabilityZoneFilter**).
- Have sufficient RAM available (**RamFilter**).
- Are capable of servicing the request (**ComputeFilter**).
- Satisfy the extra specs associated with the instance type (**ComputeCapabilitiesFilter**).
- Satisfy any architecture, hypervisor type, or virtual machine mode properties specified on the instance's image properties. (**ImagePropertiesFilter**).

FILTERS

MORE THAN 35 FILTERS AVAILABLE!

- AggregateCoreFilter
- AggregateDiskFilter
- AggregateRamFilter
- AggregateTypeAffinityFilter
- AllHostsFilter
- AvailabilityZoneFilter
- ComputeCapabilitiesFilter
- ComputeFilter
- DiskFilter
- ExactCoreFilter
- ExactDiskFilter
- GroupAffinityFilter
- ImagePropertiesFilter
- IsolatedHostsFilter
- IoOpsFilter
- NUMATopologyFilter
- NumInstancesFilter
- RetryFilter
- etc.

NUMA TOPOLOGY FILTERS

AggregateInstanceExtraSpecsFilter

- Matches properties defined in extra specification for an instance type against properties on a host aggregate defined by the administrator. Works with specifications that are scoped with **aggregate_instance_extra_specs**.

NUMATopologyFilter

- Filters hosts based on the NUMA topology specified for the instance through the use of flavor extra specifications, in combination with image properties.

WEIGHT FILTERS

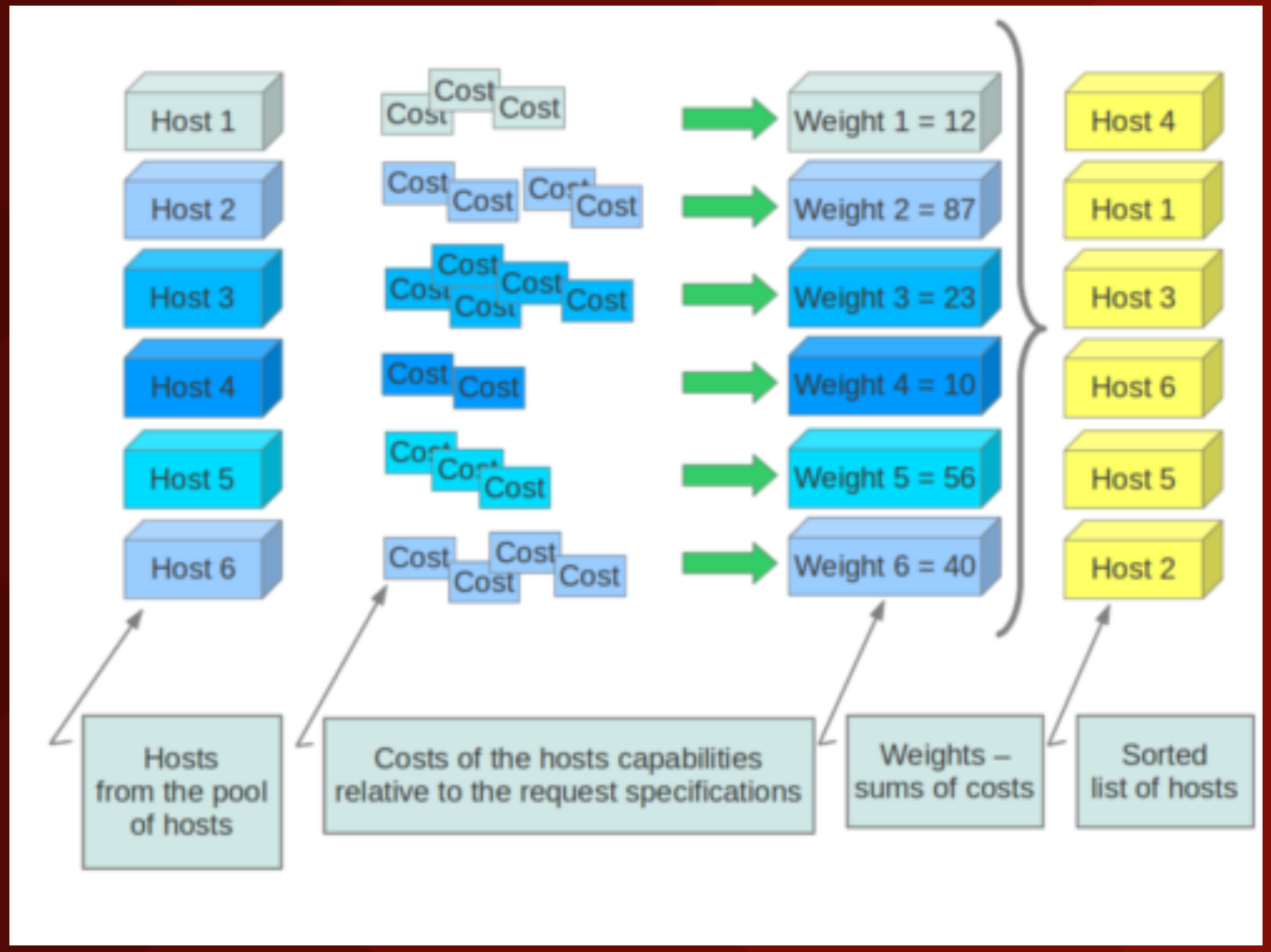
When provisioning instances, the Filter Scheduler filters and weighs each host in the list of acceptable hosts.

Each time the scheduler selects a host, it virtually consumes resources on it, and subsequent selections are adjusted accordingly.

This process is useful when the user asks for the same large amount of instances, because weight is computed for each requested instance.

All weights are normalized before being summed up; the host with the largest weight is given the highest priority.

WEIGHT FILTERS



Source: docs.openstack.org

HOST WEIGHTING OPTIONS

Section	Option	Description
[DEFAULT]	ram_weight_multiplier	By default, the scheduler spreads instances across all hosts evenly.
[DEFAULT]	scheduler_host_subset_size	New instances are scheduled on a host that is chosen randomly from a subset of the **N** best hosts. This property defines the subset size from which a host is chosen. A value of 1 chooses the first host returned by the weighing functions. A value less than 1 is ignored, and 1 is used instead.
[DEFAULT]	scheduler_weight_classes	Defaults to **nova.scheduler.weights.all_weighers**, which selects the only available weigher, the **RamWeigher**. Hosts are then weighed and sorted with the largest weight winning.
[metrics]	weight_multiplier	Multiplier for weighing metrics.

```
[DEFAULT]
scheduler_host_subset_size=1
scheduler_weight_classes=nova.scheduler.weights.all_weighers
ram_weight_multiplier=1.0
[metrics]
weight_multiplier=1.0
weight_setting=name1=1.0, name2=-1.0
required=false
weight_of_unavailable=-10000.0
```

WRAPPING UP!

VARIOUS WAYS TO IMPROVE PERFORMANCE IN AN OPENSTACK ENVIRONMENT

- **Network**, which includes traffic, HA, and distribution: DVR, but also NFV datapaths, QoS, or DSCP (not discussed here)

- **Instances**, with CPU pinning (*defines CPU allocation*), host aggregates (*provides a mechanism to allow administrators to assign key-value pairs to groups of machines*), hugepages, and compute filters for advanced placement.

openstack

THANK YOU!

THIS PRESENTATION IS PART OF THE RED HAT TRAINING CLOUD CURRICULUM

FIND MORE INFORMATION, VIEW SCHEDULE OF CLASSES AND ENROLL TODAY

HTTP://WWW.REDHAT.COM/TRAINING