Hewlett Packard
Enterprise

Helion OpenStack Carrier Grade 4.0

SYSTEM ENGINEERING GUIDELINES

4.0

Acknowledgements

Java® and Oracle® are registered trademarks of Oracle and/or its affiliates

http://www.hpe.com/info/storagewarranty

Helion OpenStack Carrier Grade 4.0

System Engineering Guidelines

Contents

1

System Engineering Guidelines

Helion OpenStack Carrier Grade 4.0 (HCG 4.0) is a high-performance, high-availability Network Functions Virtualization (NFV) cloud solution capable of meeting demanding customer requirements for scalability, performance, capacity, and availability. It is not a one-size-fits-all solution but a highly flexible framework that provides a range of deployment models, types and quantity of hardware used, and tunable parameters to enable an optimized deployment for varying customer requirements.

HCG 4.0 brings together the flexibility and scalability of the IT cloud and the high-availability and performance demanded by the Telecommunications industry into a unique carrier grade, industry-leading solution to deliver a price-performance ratio well above alternative solutions. HCG 4.0 is aligned with the ETSI-NFV architecture.

This document provides engineering guidelines, rules, and system parameters to assist cloud architects, installers, and administrators in deploying and scaling the HCG 4.0 to meet application-specific requirements. Failure to follow the recommendations, guidelines and engineering rules will result in unknown performance trade-offs likely compromising reliability and performance.

2

Deployment Options

Helion OpenStack Carrier Grade 4.0 supports four deployment models to provide a range of options. The deployment models include:

- HCG 4.0 Standard Solutions
  - HCG 4.0 Standard Configuration with Dedicated Storage
  - HCG 4.0 Standard Configuration with Controller Storage
  - HCG 4.0 in Multi-Region Environment
- HCG 4.0 CPE

Each model supports the same functionality with different ranges of scalability and performance. The models are differentiated by where the controller, storage, and compute functions reside, with the largest configuration (Standard Configuration with Dedicated Storage) distributing these functions on different hosts and the smallest configuration (HCG 4.0 CPE) combining all three functions onto a single redundant pair of hosts. The Standard Configuration with Controller Storage resides between these two and combines controller and storage functionality onto a redundant pair of hosts supporting one or more compute nodes. A HCG 4.0 Multi-Region deployment is also supported, which provides a mechanism for scaling beyond the limits of a single HCG 4.0 cloud or region in some deployment models.

The following sections describe these configurations in more detail and provide guidelines on the appropriate use for each model.

The following table provides a guideline for deployment model scaling.

Table 1 **Deployment Model Scaling**

	Controller Nodes	Storage Nodes	Compute Nodes	VMs
Standard Configuration with Dedicated Storage	2	2–8	100	1000 (10 per node, max tested / supported in this release, dependent on sufficient vCPU, memory, and storage resources)
Standard Configuration with Controller Storage	2	0	10 (typical due to controller storage performance but not fixed)	100 (10 per node, dependent on sufficient vCPU, memory, and storage resources, and limited by controller disk capacity, throughput, and boot times)
HCG 4.0 CPE	2 (with compute function)	0	0	20 (10 per node, dependent on sufficient vCPU, memory, and storage resources)

NOTE: The baseline used to create this table includes a heterogeneous mix of VMs and features, as follows:

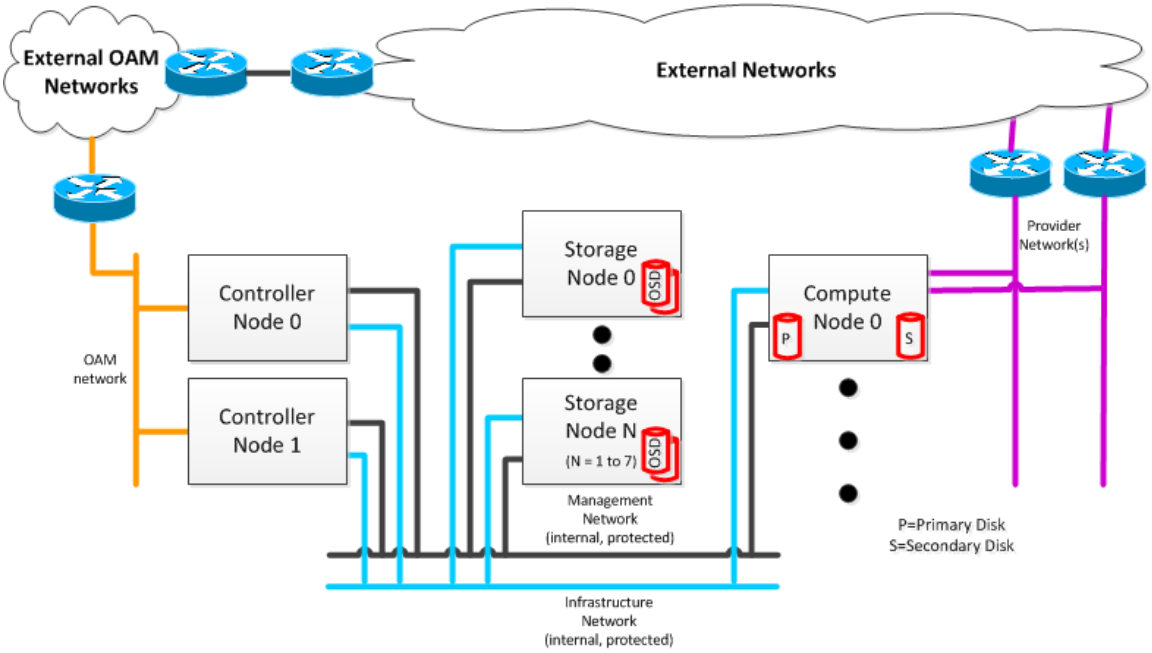
- 1–4 cores
- 1–4 GB memory
- 600 MB and 3.6 GB images
- 1/2/4/10/20/40 GB volumes
- single and multi-NUMA nodes
- local storage and remote storage
- launch via nova and heat
- dedicated and shared CPU policies
- VM scaling
- CPU scaling
- 1–3 network connections per VM

Standard Configuration with Dedicated Storage

Deployment of HCG 4.0 with Dedicated Storage Nodes provides the highest capacity (single region) and performance with scalability up to 100 compute nodes and 1,000 VMs, assuming sufficient CPU, memory and storage resources and a VM distribution similar to that described in Deployment Options on page 3. The differentiating physical feature of this model is that the controller, storage, and compute functionalities are deployed on separate physical hosts allowing controller nodes, storage nodes, and compute nodes to scale independently from each other.

The following figure shows the nodes and logical networks supported in this configuration.

Figure 1: Dedicated Storage Configuration

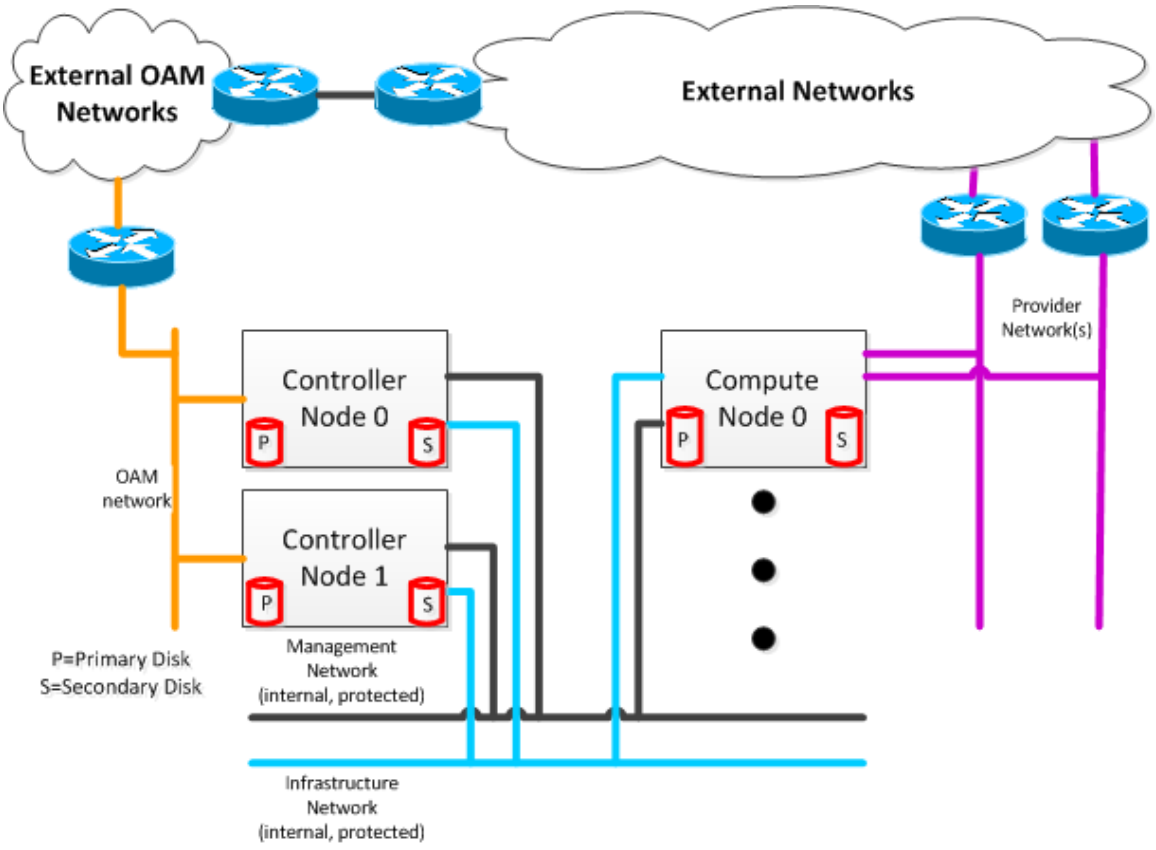


The controller nodes provide the control function for the system and two controller nodes are required to provide active/standby redundancy. The controller nodes can be sized in terms of the server and peripherals, for example, CPU cores/speed, memory, storage, and network interfaces depending on requirements. Two to eight storage nodes, deployed in pairs, provide the storage function for Glance images, Cinder volumes, remote Nova ephemeral disks and Swift containers. Redundancy and scalability is provided through the number of Ceph object storage device (OSD) pairs, with more OSDs providing more capacity and better storage performance. OSD size and speed, optional SSD Ceph journals, optional SSD Ceph cache tiering, CPU cores and speeds, memory, disk controllers, and networking also impact the scalability and performance of the storage function. This model supports up to 100 compute nodes, each of which can be sized independently in terms of CPU cores/speed, memory, local storage, and interfaces. The actual number of VMs supported may be limited by availability of compute node resources and has been tested to 1000 VMs in a 100 compute node configuration.

Standard Configuration with Controller Storage

HCG 4.0 supports a small-scale deployment option using Logical Volume Manager/Internet Small Computer System Interface (LVM/iSCSI) as a back end for Cinder volumes on the controller nodes instead of using dedicated storage nodes. This configuration uses dedicated physical disks in the controller node synchronized with the other controller node. The primary disk is used by the platform and for Glance image storage. The secondary disk is used for Cinder volumes. Because the primary and secondary disks are single devices, this configuration does not scale to the capacity of the Standard Configuration with Dedicated Storage, nor does it provide the same performance and latencies for activities such as VM creation and deletion. The number of compute nodes and VMs supported is limited by the Cinder/Glance storage backend capacity, storage throughput, and VM launch time requirements. Figure 2: Standard Configuration with Controller Storage on page 6 shows a typical deployment.

Figure 2: Standard Configuration with Controller Storage



HCG 4.0 in a Multi-Region Environment

HCG 4.0 can be deployed in a multi-region configuration as the Primary Region or a Secondary Region, or both.

A *region* is a discrete OpenStack environment with dedicated API endpoints that typically shares at least the Identity/Keystone service with other regions. Region configurations provide for the scalability of HCG 4.0 clouds and inter-working with non-HCG 4.0 clouds.

HCG 4.0 offers flexible deployment in a regions configuration. Deployment options include:

- HCG 4.0 as Primary and Secondary Region
- Third-party OpenStack Cloud as Primary Region and HCG 4.0 as Secondary Region

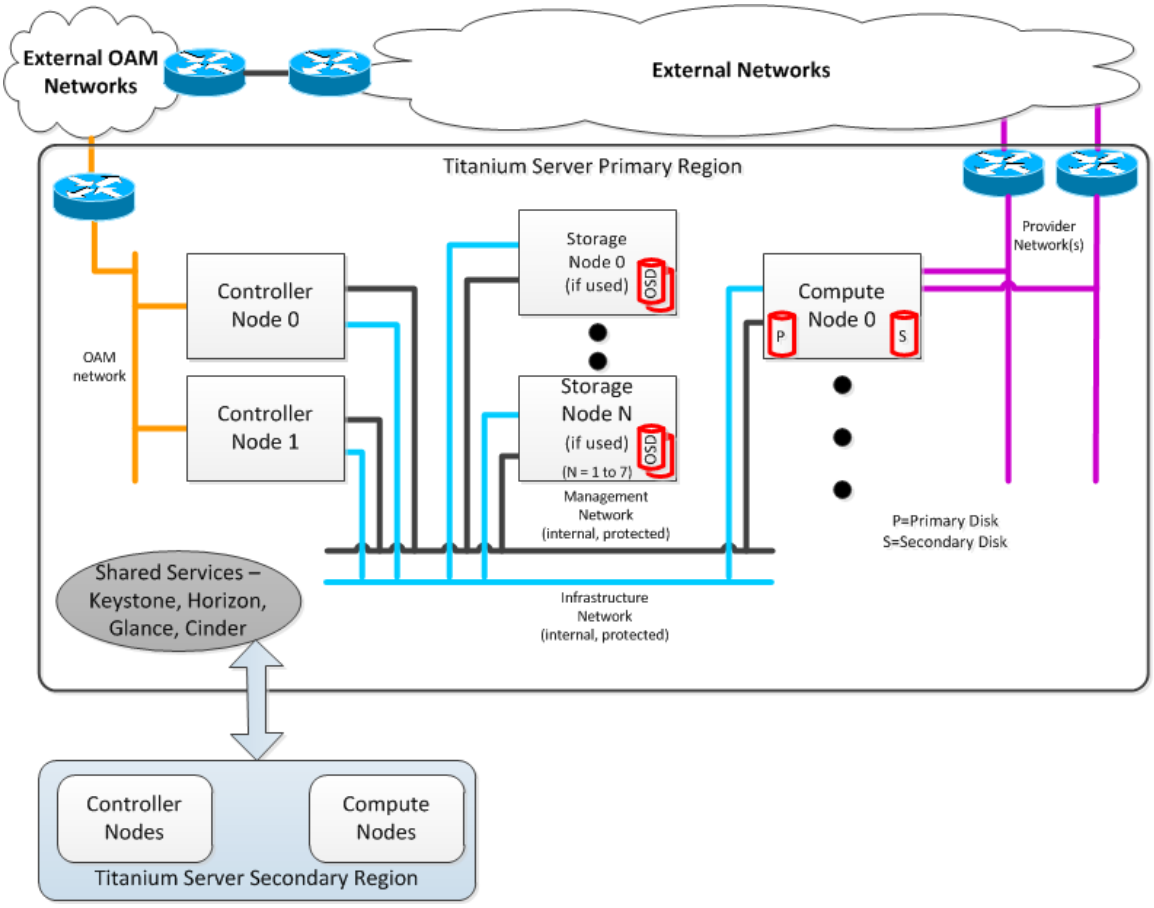
For information about other supported multi-region configurations, including both additional shared services and multi-region configurations with non-HCG 4.0 OpenStack clouds, refer to *HCG 4.0 for Regions*. For multi-region dimensioning and system engineering, consult HCG 4.0 personnel for your use cases.

HCG 4.0 as Primary and Secondary Region

This configuration provides for scalability. It enables you to scale the number of overall compute nodes to more than that supported by a single HCG 4.0 region, with some level of integration (through shared services) across all regions, and with separate Nova and Neutron domains for each region.

As shown in Figure 3: Scalability Use Case with HCG 4.0 Primary and Secondary Region on page 8, selected controller services are shared between regions. Shared central services provide multi-region-wide management capabilities.

Figure 3: Scalability Use Case with HCG 4.0 Primary and Secondary Region

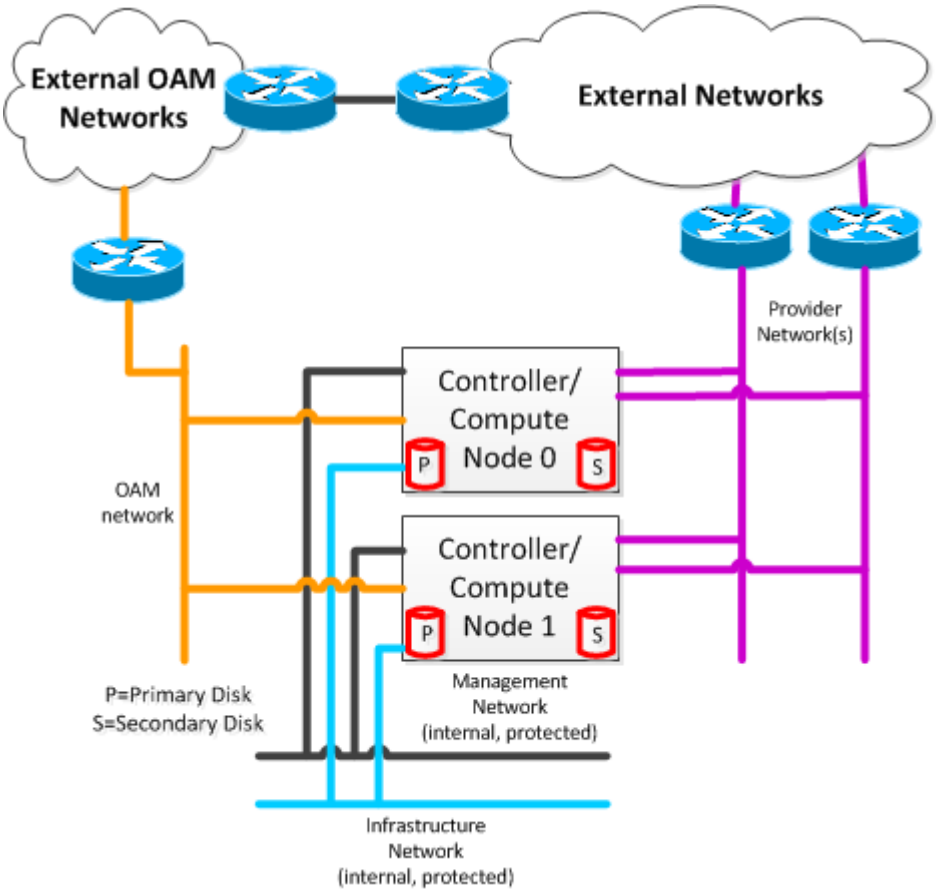


HCG 4.0 CPE

HCG 4.0 CPE (Customer Premises Equipment) provides a scaled-down HCG 4.0 deployment option that combines controller, storage, and compute functionality on a redundant pair of hosts.

In this configuration, the controller functionality is active/standby across the nodes and the compute functionality is active/active, meaning VMs can be deployed on both hosts. LVM/iSCSI is used as the Cinder volume backend. Processor bandwidth and memory is reserved for the controller function.

Figure 4: HCG 4.0 CPE Configuration



Supported Networks and Interface Configurations

The same logical networks are supported in all configurations. Actual network requirements and recommendations vary depending on the configuration.

The following networks are supported in HCG 4.0:

- External Operations and Maintenance (OAM) network (controller nodes only)
- Internal Management network (all nodes)
- Optional pxeboot Network (all nodes)
- Infrastructure Network
- Optional Board Management Network
- Provider Network(s) (all compute nodes)

Logical interfaces can be dedicated to single or link aggregated interfaces for maximum performance and isolation or combined in various combinations with other logical interfaces onto a common physical interface or a link aggregated interface and separated into isolated L2 LAN segments in the top-of-rack switch.

3

Controller Dimensioning

This section describes dimensioning of controller nodes. For additional details and hard limits of the controller node hardware, including the list of supported NICs supported on the controller nodes, refer to *HCG 4.0 Planning: HCG 4.0 Hardware Requirements*.

Controller CPU Requirements

Minimally a dual socket Intel Xeon® E5 26xx (Sandy Bridge) machine with 8 cores/socket is required for the controller node.

BIOS/UEFI Settings for Controller Nodes

For a controller node, the following settings must be configured in the platform BIOS or UEFI:

Table 2 **Controller Node BIOS/UEFI Settings**

Parameter	Setting	Comments
Mode	BIOS or UEFI mode	UEFI Secure Boot and UEFI IPv6 PXE Boot are not supported in this release
Hyperthreading	Enabled or Disabled	
Power profile	Max Performance	
Minimum Processor Idle Power	No C States	
Boot Order	Hard Disk, PXE, USB	Boot from hard disk after installation

Controller Memory Requirements

At least 64 GB RAM is mandatory for the controller node.

Controller Networking Requirements

The management and infrastructure interfaces are logical interfaces that can be mapped to physical interfaces or shared physical interfaces using VLANs for separation. The following management/infrastructure interface combinations are supported. Link aggregation of all interfaces is recommended for high availability.

- 1 GbE management network and 10 GbE infrastructure network
- 10 GbE management network and no infrastructure network (recommended configuration)
- 10 GbE network with management and infrastructure VLANs

If the controller node supports a board management via an iLO management interface, the Board Management Controller (BMC) module's Ethernet port can be connected to a dedicated private network. For controller node communication with the BMC modules, a VLAN on the internal management logical interface is configured on the Board Management Network. It is recommended, but not required, to use the OAM network for controller access to the board management network.

Primary Disk Requirements for Controller Nodes

The controller node primary disk is divided into a number of partitions, some of which are fixed size and some of which are configurable. The following table shows the partitioning of the disk and the following sections describe the dimensioning of the disk.

Table 3 **Controller Node Primary Disk Partitions and Sizes**

Partition	Size	Comments
rootfs	19.5 GB	Fixed
boot	500 MB	Fixed
scratch	7.5 GB	Fixed
log	7.5 GB	Fixed
platform	2 GB	Fixed
rabbitmq	2 GB	Fixed
cgcs	10 GB	Fixed for Dedicated Storage configuration
	configurable	For Controller Storage/HCG 4.0 CPE configurations – See Glance Dimensioning for Controller Node Storage on page 14
database	configurable	See Database Dimensioning on page 14
backup	configurable	See Backup Dimensioning on page 15
image conversion	configurable	See Image Conversion Dimensioning for Controller Node Storage on page 14

Disk Hardware Configuration for System Storage

For the controller node, a minimum 500 GB solid-state drive (SSD) is mandatory for capacity and performance. However, a bottom-up calculation of the disk size requirement should be used to ensure 500 GB is sufficient.

Total Primary Disk requirement = 39 GB (fixed platform) + D GB (database) + X GB (images) + Y GB (Image Conversion) + B GB (backup)

Where:

D = Database (see Database Dimensioning section)
X = Images (see Glance Backend Dimensioning section)
Y = Conversion (see Dimensioning Section)
B = Backup (see Backup Dimensioning section)

Database Dimensioning

The database dimensioning default is 40 GB (20 GB * 2 for upgrades). The database size is primarily driven by the number of telemetry service meters and these are related to the number of compute nodes and VMs. The following formula should be used to determine if a larger database partition is required. If the number of meters is known, 200 KB per meter can be used to calculate a database size.

D = Database Storage = 2 * (5 GB base + VMs *25 MB + ComputeNodes * 10 MB)

Glance Dimensioning for Controller Node Storage

The **/opt/cgcs** partition is used by various controller applications as data storage that is replicated between the controller nodes. For the Standard Configuration with Controller Storage and HCG 4.0 CPE configuration the partition is also used for used for Glance image storage and is therefore a configurable size. For the Standard Configuration with Dedicated Storage the size is fixed at 10 GB. To determine the logical disk capacity requirement for images, calculate the total expected image requirement plus some growth margin.

X = Image Storage Requirement = N * AvgImageSize + Margin

Where:

N = number of images
AvgImageSize = Average image size
Margin = User-defined margin (recommend at least 25%)

Image Conversion Dimensioning for Controller Node Storage

The image conversion partition is used by Cinder for temporary storage for image conversions.

The default value of the volume is 20 GB. It should be increased if any of the following conditions are met.

- Standard Configuration with Controller Storage and HCG 4.0 CPE
  - Creation of multiple Cinder volumes from qcow2 images in parallel.
- Standard Configuration with Dedicated Storage
  - Creation of multiple Cinder volumes from qcow2 images in parallel when the images have not been cached as raw images at image create time using the --cache-raw option. This also applies if volumes are created immediately after image creation with the --cache-raw option before the images are cached.
  - Creation of very large images from qcow2 image files with the --cache-raw option. Note that if creating multiple images in parallel the caching is performed serially so only the largest image matters.

NOTE: Use of the --cache-raw option is highly recommended when creating qcow2 images in the Standard Configuration with Dedicated Storage. This will speed up volume creation significantly.

To determine the logical disk capacity requirement for image conversion, calculate the total expected image conversion space required plus some growth margin.

Standard Configuration with Dedicated Storage:

Y = Image Conversion Requirement = 2 * SimultaneousVolumeCreations * AvgImageDiskSize + 2 * MaxImageDiskSize + Margin

Standard Configuration with Controller Storage:

Y = Image Conversion Requirement = SimultaneousVolumeCreations * AvgImageDiskSize + Margin

Where:

MaxImageDiskSize = Size of largest qcow2 image on disk (per "qemu-img info <file>") that will be cached as a raw image.
AvgImageDiskSize = Average image size on disk being created (per "qemu-img info <file>"). Recommend using MaxImageDiskSize
Margin = User-defined margin (recommend at least 25%)
SimultaneousVolumeCreations = Maximum simultaneous volume creations using uncached qcow2 images or before the images are cached.

NOTE: SimultaneousVolumeCreations is limited by the number of Cinder workers. Cinder workers is set to the number of cores on the controller node or 20, whichever is less.

Backup Dimensioning

Backup storage must be dimensioned to allow for backing up the database and images. Use the following equation to calculate the minimum backup storage allocation:

B = Backup Storage = D GB (database) + X GB (images) + 20 GB (overhead)

Where:

D = Database (see Database Dimensioning section)
X = Images (see Glance Backend Dimensioning section)

LVM Storage Requirements for Controller Nodes

The following requirements only apply to configurations that use LVM/iSCSI backend for Cinder volumes (Standard Configuration with Controller Storage or HCG 4.0 CPE configurations). For this configuration a secondary disk on the controller node is mandatory.

Disk Hardware Configuration for LVM Storage

The minimum hardware disk configuration for a configuration using LVM for image/volume storage is a 500 GB serial-attached SCSI (SAS) disk at 10K revolutions per minute (RPM) but SSD

is highly recommended. Depending on the number of VM volumes expected, which may be greater than the number of VMs deployed, additional storage may be required. To determine the disk capacity requirement for LVM, multiply the number of instances for each type of VM by the volume size for the type.

Secondary Disk Size Requirement = N * AvgVolSize + Margin

Where:

N = maximum number of volumes to be supported
AvgVolSize = Average volume size on disk
Margin = User-defined margin (recommend at least 25%)

NOTE: Thin and thick provisioning are supported for volume storage. With thin provisioning, the volume size on disk is based on how much disk space the VM currently uses, not the advertised volume size. HCG 4.0 does not support overcommit of storage, so for capacity dimensioning, thin and thick provisioning are equivalent.

Disk Throughput Dimensioning for LVM Storage

Two aspects of disk throughput should be considered when dimensioning the secondary storage: VM creation from volumes times, and guest disk throughput.

The main factor in creation times for volumes and booting the VMs from the volumes is disk throughput of the primary and secondary disks, since images are stored and converted on the primary disk and copied to volumes on the secondary disk. The storage device throughput, disk controller throughput, image size, and volume size all contribute to the overall Volume+ VM creation times. Platform activity such as database on the primary disk and logging also affects disk throughput and Volume + VM creation times.

Since the primary and secondary disks are single devices, this configuration does not scale as well as a dedicated storage configuration and as the following charts show, the volume creation + VM boot times can be long when multiple volumes are created at the same time.

The following data was obtained through benchmarking and measuring simultaneous volume plus VM creation times. The data provides guidance in dimensioning for throughput and for making decisions to use a dedicated storage configuration for better performance. For more guidance in choosing a Dedicated Storage or Controller Storage deployment, see Storage Node Dimensioning on page 41.

Wildcat Pass Reference Configuration:

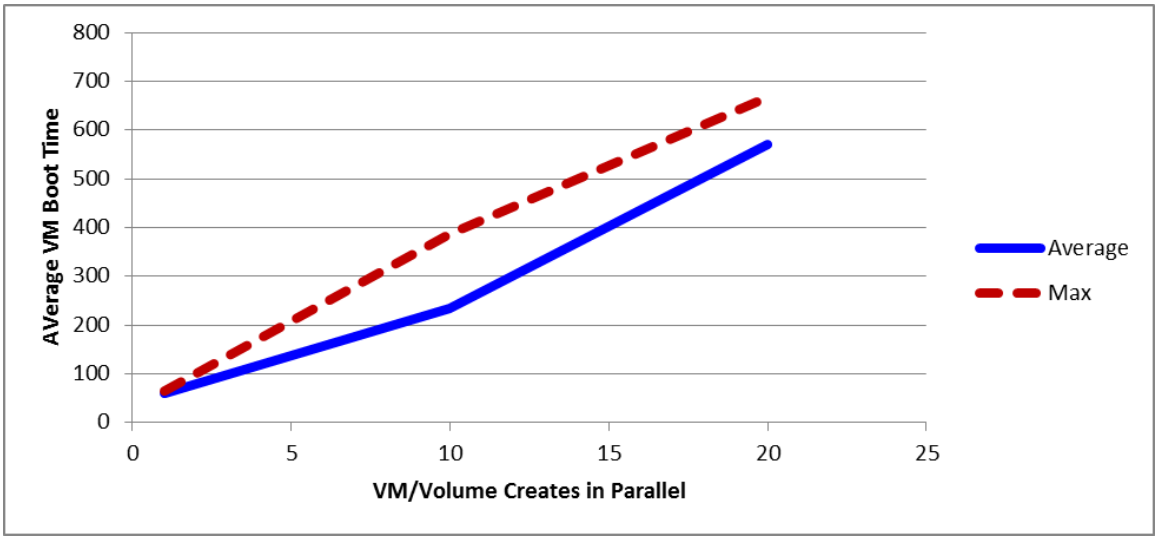
- Intel® Server Board S2600WTT (Wildcat Pass)
- Intel® Broadwell Processor - Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
- Intel® C612 Chipset
- 480 GB Intel® SSDSC2BB24 SSDs for primary and secondary disk
- Cinder configured for thin provisioning

The following chart shows the times to create a volume and boot an image for a number of tests with Centos 7 qcow2 images resulting in 8 GB raw image size and using thin provisioning for Cinder on the reference configuration. Booting of a single VM with a Cinder volume required about 1 minute. As more volumes and VMs are created in parallel, the average Volume + M create time increases linearly.

If many VMs are being created, faster disk controllers and storage media may be needed or the deployment model should be changed from a Controller Storage configuration to a Dedicated Storage configuration. Based on this data, it is highly recommended that no more than five VMs with volumes be created simultaneously when using SSD disks, fewer if using SAS disks.

NOTE: The choice of thin or thick provisioning can affect VM/volume creation times. For thin provisioning, volume creation and deletion can be significantly faster. Thick provisioning requires disk copying and zeroizing the entire advertised disk size and may take an order of magnitude longer for volumes to create. Thin provisioning copies and zeroizes only the currently used disk space for the volume and the configured volume size has no effect on volume creation or deletion times.

Figure 5: Volume Plus VM Create Times with Controller Storage



To avoid possible Cinder operation failures, VMs using a Cinder LVM backend must operate within supported I/O levels . The HCG 4.0 overload monitor raises a Major alarm if congestion is detected, and a Critical alarm if acceptable I/O limits are exceeded.

Customers can avoid Cinder I/O congestion by using disk I/O quotas to limit the amount of I/O their VMs can consume. For more information, see the following references:

- http://ceph.com/planet/openstack-ceph-rbd-and-qos/
- http://docs.openstack.org/admin-guide/compute-flavors.html#compute-flavors (disk tuning section)

DRBD Synchronization

HCG 4.0 uses DRBD to synchronize storage between controller nodes.

Steady-state synchronization occurs in real time. Initial synchronization or resynchronization is restricted to avoid overloading the network and storage devices with synchronization traffic. For large disks, this requires a significant amount of time during which the controller is in a degraded state.

The following partitions are synchronized between the controllers:

- Postgres
- RabbitMQ
- Platform
- Images
- Cinder (Controller Storage and CPE Configurations Only)

The reference configurations below were used to benchmark the default DRDB performance and provide guidance on DRBD synchronization times. Both configurations use the default DRBD synchronization settings and use SSD for the primary storage. For Cinder, one configuration uses a 10K RPM SAS disk and the other uses SSD.

Configuration 1:

- Intel® Broadwell Processor - Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
- Primary Disk - INTEL SSDSC2BB48 480 GB SSD
- Cinder Disk – Seagate ST1200MM0018 1200 GB 10K RPM SAS
- Storage Controllers
  - Intel Corporation C610/X99 series chipset SATA Controller [AHCI mode] (rev 05)
  - Intel Corporation C610/X99 series chipset 6-Port SATA Controller [AHCI mode] (rev 05)

Configuration 2:

- Intel® Broadwell Processor - Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
- Primary Disk - INTEL SSDSC2BB48 480 GB SSD
- Cinder Disk – Seagate ST1200MM0018 1200 GB 10K RPM SAS
- Storage Controllers
  - Intel Corporation C610/X99 series chipset SATA Controller [AHCI mode] (rev 05)
  - Intel Corporation C610/X99 series chipset 6-Port SATA Controller [AHCI mode] (rev 05)

Table 4 **Configuration 1 (SAS) Reults**

	postgres (SSD)	rabbitmq (SSD)	platform (SSD)	images (SSD)	Cinder (SAS)
Partition Size (GB)	50	2	2	330	1000
Sync Time (minutes)	05:01	< 00:30	< 00:30	25:10	50:05
Sync Rate (MB/sec)	170	N/A	N/A	224	155

Table 5 **Configuration 2 (SSD)**

	postgres (SSD)	rabbitmq (SSD)	platform (SSD)	images (SSD)	Cinder (SSD)
Partition Size (GB)	50	2	2	330	446
Sync Time (minutes)	04:56	< 00:30	< 00:30	26:04	36:11
Sync Rate (MB/sec)	173	N/A	N/A	216	210

The DRBD synchronization has a tunable sync rate algorithm that can be used to account for replication link speed and achieve an expected level of parallelism and networking utilization that may be different from the defaults. The default settings are designed such that DRBD will not impact storage and disk utilization to the point that it significantly impacts service. HCG 4.0 provides some ability to tune the maximum sync rate based on the system configuration and performance. It is assumed the setting will be low enough that the DRBD link (usually the infrastructure network) is not saturated.

- Reducing parallel (e.g. from 2 to 1) has the effect of increasing max sync rate by 2x.
- Increasing utilization (e.g. from 40 to 80) has the effect of increasing max sync rate by 2x.

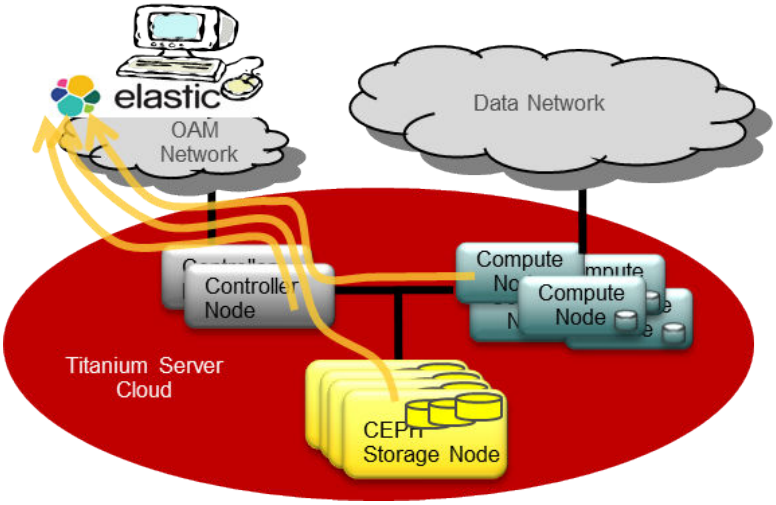
NOTE: If the system is limited by disk performance (for example, if disk utilization is near 100%) the DRBD tunable parameters have no bearing on the effective sync rate. In addition, if there is other I/O activity in the system, DRBD self-regulates and the effective sync rate is reduced.

Remote Logging Requirements

HCG 4.0 supports continuously streaming detailed system logs from all hosts to a remote log server for centralized review and analysis.

When centralized logging is enabled, logs written to the **/var/log** directory on each host are also sent to a remote log server. Each host sends logs through the active controller to a remote log server over the OAM network using either TCP or UDP. For added security, a TLS connection can be configured.

The following figure shows how Remote Logging works.



The following tables show typical impacts of logging to the network and remote log server for dimensioning purposes.

Table 6 **Typical Log Rates, OAM Bandwidth, and Remote Storage Impacts of Logging**

Host Type	# of logs/day	OAM Bandwidth	Remote Log Server Storage/Day
Controller	250,000	5.80 Kbps	62.5 MBytes
Storage	30,000	0.70 Kbps	77.5 MBytes
Compute	300,000	6.95 Kbps	75.0 MBytes

Table 7 **2x Controller, 2x Storage, 20x Compute Logging Impacts**

# logs/day	6,560,000 logs/day
OAM Bandwidth	152 Kbps
Remote Log Server Storage/day	1.64 GBytes/day

4

Compute Node Dimensioning

This section describes dimensioning of compute nodes. For additional details and hard limits of the compute node hardware, including the list of supported NICs that are supported on the compute nodes, refer to *HCG 4.0 Planning: HCG 4.0 Hardware Requirements*.

Compute Node CPU Requirements

Minimally a dual socket Intel Xeon® E5 26xx (Sandy Bridge) machine with 8 cores/socket is required for the Compute Node.

BIOS/UEFI Settings for Compute Nodes

For compute nodes, the following settings must be configured in the platform BIOS or UEFI.

Table 8 **Compute Node BIOS/UEFI Settings.**

Parameter	Setting	Comments
Mode	BIOS or UEFI mode	UEFI Secure Boot and UEFI IPv6 PXE Boot are not supported in this release
Hyperthreading	Enabled or Disabled	
Intel® Virtualization Technologies (VT-x and VT-d)	Enabled	
Power profile	Max Performance	
Minimum Processor Idle Power	C States off	
Intel® Turbo Boost Technology	On	
Boot Order	Hard Disk, PXE, USB	PXE for initial install, Hard Disk post-install

Compute Node Memory Dimensioning

At least 32 GB RAM evenly distributed across the NUMA nodes is mandatory for the compute node. Default allocation of this memory is shown below although it is reconfigurable if needed. It is highly recommended that all memory channels be populated or published performance numbers will not be met.

Memory Dimensioning: Platform

A total of 10 GB is reserved for the platform: 6 GB for platform usage plus 2 GB per NUMA node for kvm overhead.

Accelerated vSwitch (AVS)

One GB huge page is allocated per NUMA node. The reserved 1GB huge pages for AVS are not configurable.

Memory Dimensioning: VMs

By default all memory not reserved for the platform or AVS is allocated to 2 MB huge pages for use by VMs. If required, the memory can be configured into pools of 2MB and 1 GB pages per NUMA node for use by the VMs.

CPU Dimensioning

CPU allocations on the compute nodes are configurable.

Hyperthreading is supported on the compute nodes; however, maximum throughput and determinism can only be achieved if hyperthreading is disabled. Image flavor options can be used to prevent other VMs from using the sibling threads; this mitigates the impact of hyperthreading, but does not eliminate it. For more information on hyperthreading support, see Hyperthreading on page 54.

CPU Dimensioning: Platform

One physical core (two logical cores if hyperthreading is enabled) on socket 0 is dedicated to the platform on the compute node by default. The number of cores is configurable.

AVS

Two physical cores are reserved for AVS by default. If hyperthreading is enabled, four logical cores are reserved. This allocation is configurable based on throughput requirements. A maximum of eight physical cores can be allocated to AVS.

Shared CPU

One physical core per socket (two logical cores if hyperthreading is enabled) can be configured as a shared CPU for use by low load vCPUs for VMs configured with a **Shared vCPU** flavor extra spec.

For guidelines on how to use this feature, see VM Scheduling and Placement on page 51.

CPU Dimensioning: VMs

All CPU cores not reserved for the platform, AVS, or as shared CPU are available for scheduling of VM vCPUs. For guidelines on how to use this feature, see VM Scheduling and Placement on page 51.

Compute Node Networking Requirements

The management and infrastructure interfaces are logical interfaces that can be mapped to physical interfaces or to shared physical interfaces using VLANs for separation. The following management/infrastructure interface combinations are supported. Link aggregation of all interfaces is recommended for high availability.

- 1 GbE management network and 10 GbE infrastructure network
- 10 GbE management network and no infrastructure network (recommended configuration)
- 10 GbE network with management and infrastructure VLANs

If the compute node supports a board management via an iLO management interface, the Board Management Controller (BMC) module's Ethernet port can be connected to a dedicated private network. For controller node communication with the BMC modules, a VLAN on the internal management logical interface is configured on the Board Management Network. It is recommended, but not required, to use the OAM network for controller access to the board management network.

Primary Disk Requirements for Compute Nodes

The primary disk on the compute node is used by the platform. One or more secondary disks are required for local VM storage. The primary disk cannot be used for local VM storage.

Disk Hardware Configuration

The primary disk on the compute node is used for boot, rootfs, logging, and scratch storage. The minimum requirements for the primary disk is a 120 GB 10K RPM SAS disk but SSD is highly recommended. The primary disk is divided into partitions according to the following table.

Table 9 **Compute Node Primary Disk Partitions and Sizes**

Partition	Size	Comments
rootfs	19.5 GB	Fixed
boot	500 MB	Fixed
scratch	3.75 GB	Fixed
log	3.75 GB	Fixed
platform	2 GB	Fixed

Local Storage Requirements for Compute Nodes

One or more secondary disk(s) are mandatory for local backing storage of images and volumes on the compute node. VMs launched from remote Cinder volumes do not use local storage.

Disk Hardware Configuration for Local Storage

For local storage on the compute node, at least one secondary disk with 500 GB capacity is mandatory and it must be at least 10K RPM SAS. SSD storage is highly recommended. Additional disks may be required to support more storage or higher guest storage bandwidth.

Backing Considerations for Nova Ephemeral Block Storage

There are three options for Nova ephemeral block storage backends.

Nova ephemeral storage can be implemented using:

- Copy on Write (CoW) image on the compute node
- LVM on the compute node
- Remote storage using Ceph on storage nodes

This choice is available for each individual compute node, so more than one backing type can be used within the same cloud. Flavor extra specs can be used to indicate the type of local storage backing required by the VM.

The primary difference between LVM and CoW image as a backing store is that for LVM, raw volumes are fully allocated out of free nova local storage space up front, while for image the volumes are stored as sparse files in the instances partition and utilize local storage space only as required. Determining which backing store to use should take into account the following:

- LVM backing is recommended for I/O intensive VMs because the VM writes are direct to disk rather than to sparse files.
- When cold-migrating VMs with LVM backing storage, the volume is rebuilt on the new compute node and user data is lost. With image backing storage, the local storage is block-migrated and user data is preserved. Use image backing storage if data must be preserved across a migration.
- VM launch times and associated local volume create times are much faster with CoW-image backed storage.
- Live block migration is supported with CoW-image backed storage. Use CoW-image backed storage if live migration is required.
- Remote ephemeral storage must be used if live migration of ephemeral, swap, and boot from image root disks is required.

Disk Capacity for Local Storage

The disk capacity configuration for local storage depends on whether LVM or CoW-image backing store is used. In both cases, one or more secondary disks can be assigned to the **nova-local** volume group. Capacity can be scaled using larger storage devices or more storage devices.

CoW-Image Backed Local Storage Capacity

When using CoW-image backing storage, the entire nova-local volume group is allocated to the Instances Logical Volume (LV) mapped to /etc/nova/instances. VMs volumes are stored as sparse files in the Instances LV.

Local Storage Requirement = N * AvgImageSize + Overhead + Margin

Where:

N = Total number of VMs expected on compute node
AvgImageSize = average size of all sparse VM volumes on expected on compute node (qcow2 files)
Overhead =Overhead for bookkeeping, console, and libvirt (recommend 10–20 GB)
Margin = User-defined margin (recommend at least 25%)

LVM-backed Local Storage Capacity

When using an LVM backing store, the **nova-local** volume group is broken into the Instance LV and free nova-local space. The Instances LV is used for bookkeeping, libvirt, console data, and caching of images. Individual VM volumes are expanded to raw volumes that are allocated out of the free space in the nova-local volume group.

Local Storage Requirement = Instance Backing Storage Requirement + Local Volume Storage Requirement

Where

Instance Backing Requirement – see below
Local Volume Storage Requirement – see below

The Instance Backing Storage Requirement can be calculated as follows:

Instance Backing Storage Requirement = N * AverageCachedImageSize + Overhead + Margin

Where:

N = Max number of cached image size (note – this accumulates over time)
AvgCachedImageSize = Average volume size on disk in GB (note that sparse files are used)
Overhead =Overhead for bookkeeping, console, and libvirt (recommend 10-20 GB)
Margin = User-defined margin (recommend at least 25%)

The local volume Storage Requirement can be calculated as follows:

Local Volume Storage Requirement = N * AverageVmVolumeSize + Margin

Where:

N = Max number of VMs expected to run on the Compute Node

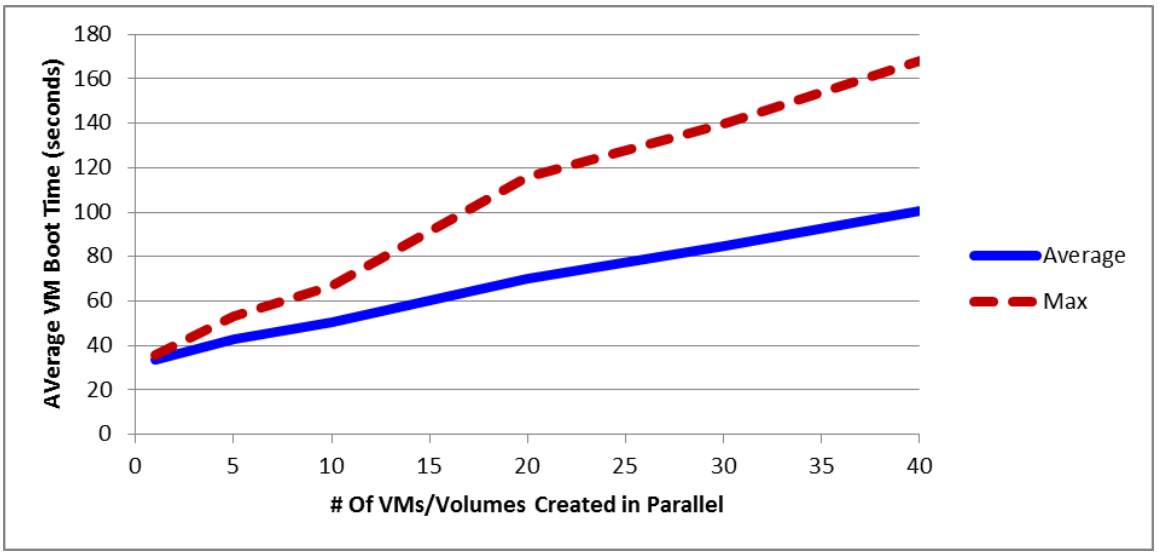
AverageVmVolumeSize = Average volume size of raw volumes in GB
Margin = User-defined margin (recommend at least 25%)

Disk Throughput Dimensioning for Local Storage

The primary impact of disk throughput is on VM boot times (including local volume creation) and guest disk throughput.

The following figure provides VM boot time data from local storage for two compute nodes with 10K RPM SAS and booting a CentOS 7 image. The compute node was configured to boot from the CoW image. The results are the same when the image is expanded to 10 GB, 20GB, or 40 GB. As shown, boot times for local storage VMs increase linearly with the number of VMs. The time can be impacted by the image size and by how many compute nodes the load is spread across (there were two nodes in this configuration). SSD disks are highly recommended for best boot times and if multiple VMs are being launched simultaneously. Use of LVM storage backing instead of image storage backing will significantly increase the boot times (and delete times) for large volumes, for example 10 GB or larger.

Figure 6: VM Create Times for Local Storage – SAS Disks on compute node.



Guest disk reads/writes when using local storage should be considered in deploying the compute node storage. Use of local storage for VMs is recommended for VMs that have high rates of disk storage access and do not require remote storage because the storage access is local to the compute node. LVM backing storage may also provide better performance than image backing storage, although image backing storage improves over time for long-lived VMs. SSD storage for the secondary disk provides significantly better throughput than SAS disks. Storage QoS is recommended to better guarantee predictable performance.

Maximum Number of VMs per Compute Node

The maximum number of VMs per compute node is not a fixed number but based on resources, VM sizes, resource locations (e.g. NUMA nodes), and policies.

HCG 4.0 will not schedule VMs if the resources and other conditions associated with the VM cannot be met. Factors that determine the number of VMs supported on a given compute node at a specific time are:

- The number of cores and memory on the compute node as well as local VM storage if VMs are booted locally.
- The sizes of individual VMs to be scheduled on the compute node (cores, memory, and local storage).
- Availability of resources such as NUMA memory and PCI-Passthrough or SR-IOV devices and policies such as **strict** or **prefer** for scheduling.
- Use of Shared CPU policy for VMs or Shared vCPU cores on VMs

For details of VM scheduling, see VM Scheduling and Placement on page 51.

It is highly recommended that excess capacity in the form of additional Compute Nodes be deployed to accommodate the loss of one or more compute nodes and successful migration/ evacuation of all VMs to the remaining compute nodes. For the HCG 4.0 CPE configuration, 100% excess capacity is required for full redundancy in case one controller node with compute function fails.

5

Networking Dimensioning

This section describes the dimensioning of the data network of the compute nodes. The data network is the backing network for the overlay tenant networks and therefore has a direct impact on the networking performance of the guest.

Please refer to the compute node dimensioning section for other factors that may impact the overall guest network performance.

vSwitch Dimensioning

The Accelerated vSwitch (AVS) provides the virtual switching interconnect between the physical network termination of the compute node and the virtual network interfaces (vNICs) of the guest VMs.

There are several factors that contribute to the overall network performance:

- The number of physical CPUs assigned to an AVS instance
- The number of physical NICs assigned to an AVS instance
- The number of virtual NICs scheduled on an AVS instance
- The NUMA node placement of CPUs assigned to AVS relative to physical NICs and guest VM
- The average packet size for the traffic being switched
- The vif-model selected for the virtual NICs of the guest VM (for example, virtio or avp)

The number of physical CPUs assigned to AVS can be configured against a compute node through the CPU assignments for that host. To ensure sufficient cores are available to host the guest VMs, configure only the minimum number of AVS cores required to meet the target application throughput.

The physical NIC assignments are performed by configuring specific interfaces to have a data network type. A physical NIC that has been assigned to AVS is detached from the kernel device driver and assigned to AVS for accelerated packet switching.

The number of virtual NICs being serviced by a single AVS instance can affect performance when there is contention for the CPU resources of a single compute host. If there are special service level requirements for traffic throughput of a particular guest VM, it is recommended to configure QoS policies of the tenant network for that VM to ensure the guest VM with the higher service level requirements will get the necessary resources. The QoS policies provide the capability to assign a weight to a tenant network relative to other tenant networks, so that when CPU contention occurs, the tenant network with the higher weight receives a higher percentage of CPU resources for packet processing.

Finally, AVS performance is highly impacted by the average packet size of the traffic being sent and received by the guest VM. If deploying an application that requires the use of smaller packet sizes, then it may be necessary to increase the number of assigned AVS cores to reach the target throughput capacity.

Supported data NICs

For a list of supported NIC devices that can be configured as accelerated data interfaces attached to AVS, refer to *HCG 4.0 Planning: HCG 4.0 Hardware Requirements*.

Throughput Dimensioning

The throughput capabilities of AVS are highly dependent on the CPU assignments, NUMA placement and the guest application traffic profile.

The performance of AVS is also highly dependent on the packet size of the traffic being exchanged with the guest and the physical network. Smaller packet sizes result in higher overhead, therefore the overall data rate is lower than that of larger packet sizes that have less overhead. The packet data must also be copied into the guest VM memory from AVS and vice versa. Therefore to obtain higher traffic rates, it may be necessary to increase the number of CPUs assigned to both AVS and the guest, depending on which is the limiting factor in the packet processing. Typically the guest VM has higher overhead on a per-packet basis due to data copy overhead and on the actual payload processing of the packet, compared to AVS which typically does not inspect the payload of the traffic beyond the layer 2 headers. The only exception to this

is when security groups are being leveraged to provide ACL services. In this configuration, AVS will inspect up to the layer 4 headers of IP traffic.

Related Links

CPU Requirements and Placement on page 31
By default, two physical CPUs on NUMA Node 0 are assigned to AVS. This is suitable for most applications, but can be changed to meet the throughput requirements of the application.

Guest Interface Model Considerations on page 31
There are several VIF models that can be assigned to virtual NICs of the guest. The VIF model depends on the networking requirements of the application and the support for the different drivers available.

NUMA considerations on page 32

CPU Requirements and Placement

By default, two physical CPUs on NUMA Node 0 are assigned to AVS. This is suitable for most applications, but can be changed to meet the throughput requirements of the application.

The number of physical CPU cores assigned to AVS can be decreased to one for lower throughput applications to free up the number of available cores for the VMs, or increased up to eight to maximize the available throughput to the application. In addition to the total number of cores assigned to AVS, the specific NUMA node for that allocation can also be specified. It is recommended to align the AVS core allocation to the same NUMA node as the assigned physical NIC PCI NUMA node termination. The AVS core assignments should be engineered to have the least amount of cores required to reach the target throughput of the application to ensure there are sufficient CPU cores available to the guest VMs for application processing.

Guest Interface Model Considerations

There are several VIF models that can be assigned to virtual NICs of the guest. The VIF model depends on the networking requirements of the application and the support for the different drivers available.

virtio - Paravirtualized I/O Network Driver (default)

The virtio interface model can be leveraged by guests that have the virtio-net driver available. In order to further enhance the performance of virtio based devices, the network traffic for these devices is handled by AVS directly, bypassing QEMU/KVM using a feature of QEMU called vhost-user. Vhost-user significantly reduces the overhead of exchanging network traffic with the guest, and is therefore suitable for most network based applications while still providing compatibility to standard device drivers. To ensure vhost-user can be leveraged, the guest must be backed by huge pages (4K pages cannot be used). In addition, vhost-user can only be leveraged if QEMU itself can leverage KVM to utilize hardware virtualization features. In the absence of these conditions, virtio network traffic must be exchanged via QEMU which greatly reduces the network performance of this interface model.

avp - Accelerated Virtual Port

To maximize networking performance, it is recommended to use the AVP kernel or AVP DPDK PMD drivers to leverage the AVP VIF model. The custom drivers are available on the following github repo: https://github.com/Wind-River/titanium-server. The AVP operates in poll mode only; therefore it is highly suited for DPDK based applications. In the AVP kernel driver, the

same polling operation is performed but is scheduled by the Linux kernel as a kernel thread before handing off to the application in the guest. This permits high throughput at higher traffic rates, but may have an impact on latency when running at lower traffic rates if packet latency is a concern for the guest application.

Emulated Network Devices

For guests that are unable to leverage accelerated drivers, several emulated devices (for example, **e1000**) are available to provide the necessary network attachment. Since the devices are emulated, and therefore suffer from poor performance, they should only be used when no other driver is available.

NUMA considerations

AVS will minimize the impacts of NUMA locality of VMs and physical NICs by optimizing the memory allocation scheme to match the NUMA topology. This optimization of memory locality does not need to be configured; however consideration should be made for VM placement relative to the AVS core allocations. For high throughput applications, it is therefore recommended to use NUMA node pinning via flavor extra specs to match the AVS NUMA node core assignment to avoid crossing the CPU interconnect link (QPI) for all network traffic. In other words, it is not recommended to configure AVS on NUMA node 0 and restrict networking VMs to NUMA node 1.

AVS can also be configured in a split configuration where AVS cores are assigned to multiple NUMA nodes. AVS will automatically isolate traffic to a particular NUMA node if it detects that the physical NICs assigned as data interfaces are also split in the same NUMA topology. This permits traffic from a guest VM to be localized to the current NUMA node, without the need to cross the interconnect link between the CPU. However, care must be taken that the underlying physical NIC configuration also respects the same localization of traffic to avoid traffic destined to the localized VM from being received on a different NUMA node.

Maximum number of supported ports per AVS instance

AVS supports a maximum of 254 physical and virtual ports, with a restriction of 4 physical ports per compute node. A physical port is a physical NIC assigned as a data interface to the compute node. A virtual port can be a vNIC that has been assigned to a guest or a host port that is used to terminate tenant networks for hosting a virtual router or DHCP server. Therefore the number of virtual ports used by an AVS instance is dependent on the number of networks and VMs scheduled on a particular compute node.

Maximum number of supported vNICs per guest

A single VM can be assigned a maximum of eight virtual NICs. This is independent of the VIF model, and therefore applies to emulated devices (e.g. e1000), para-virtualized devices (virtio) and accelerated devices (avp).

Maximum entries in MAC address table

A single AVS instance supports a maximum of 1024 MAC address table entries per network segment. In addition, AVS supports a maximum of 256 network segments. AVS will only populate active entries as it learns endpoints, therefore it is possible for a network segment to contain greater than 1024 connected devices as long as the single compute node is not actively communicating with all end-points simultaneously. MAC entries are aged out after a fixed interval of 120 seconds of inactivity.

Maximum security rules per node

Security groups are enforced by AVS on a per-virtual port (vNIC) basis for both ingress and egress traffic. For a given port, 128 rules including both ingress and egress rules can be configured. In addition, each AVS instance supports a maximum of 16384 rules in total. Increasing the number of security group rules for a given guest has a direct impact on performance; therefore it is recommended to use wildcard matching and port ranges whenever possible. In addition, security group rules are compressed; therefore rules that have overlapping match criteria (for example, address ranges), can be optimized to reduce the performance penalty for a larger number of rule sets.

vSwitch Dimensioning: Throughput Benchmarks

The following diagram shows the test setup for the AVS throughput benchmarks for a single guest VM. The test setup is an end-to-end test with bidirectional traffic through the VM.

The following test configurations were used to measure the different VIF models using two different guest NIC drivers performing Layer 2 switching. For details of the different VIF models, see Guest Interface Model Considerations on page 31. In this test, vhost-user is automatically enabled as the virtio backend.

Test Configurations:

virtio-kernel

virtio vif model with kernel layer2 bridging in the guest

avp-kernel

avp vif model with kernel layer2 bridging in the guest

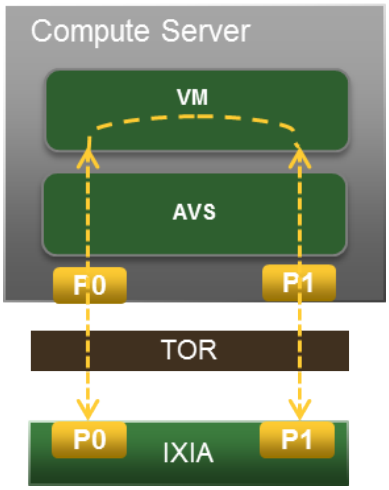
virtio-dpdk

virtio vif model with DPDK based layer2 switching in the guest

avp-dpdk

avp vif model with DPDK based layer2 switching in the guest

Figure 7: Guest Throughput Benchmark Test Configuration



vSwitch Dimensioning: Test Details

The following description and chart captures the data throughput for differing packet sizes using the AVS configuration configured against a single NUMA node.

- System
  - Manufacturer: Intel Corporation
  - Product Name: S2600WT2R
  - Version: H21573-366
- BIOS
  - Vendor: Intel Corporation
  - Version: SE5C610.86B.01.01.0016.033120161139
  - Release Date: 03/31/2016
  - Settings
    - Processor Configuration
      - : Hyper-Threading = Disabled
    - Power & Performance
      - Policy = Performance
      - Workload = Balanced
    - P-States
      - SpeedStep = Enabled
      - Turbo Boost = Enabled
      - Energy Efficient Turbo = Disabled

    - C-States
        - CPU C-State = Disabled
    - Acoustic and Performance
        - Fan Profile = Performance
- CPU
    - Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz
- RAM
    - DDR4 @ 2133 MHz - 4 channels
- Physical ports
    - 2 Intel Corporation 82599ES 10-Gigabit Devices
    - 1 physical port per NIC per PCI slot
    - PCI Express 3 8-lane slots
- VM
    - 2 vCPUs performing L2 switching
    - VM and AVS collocated on NUMA socket 0
    - VM backed by 2M huge pages in host
    - Two vSwitch cores measured
- IXIA Test
    - RFC2544 Bi-directional traffic @ 0.01% loss tolerance

Figure 8: AVS throughput - % 20Gbe

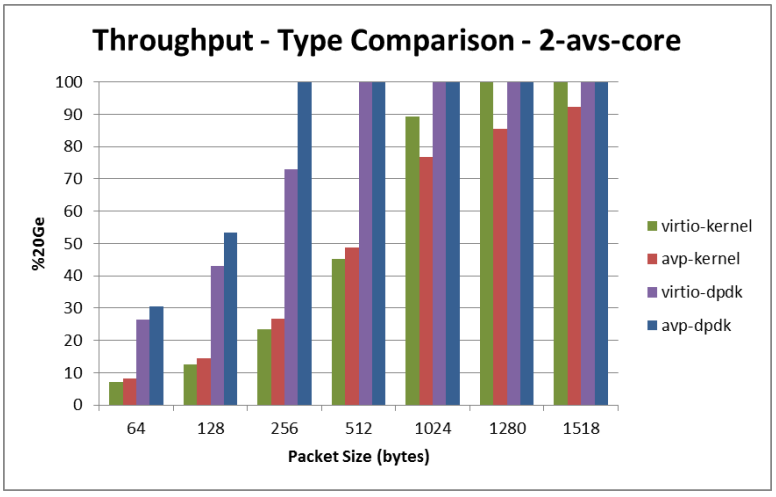
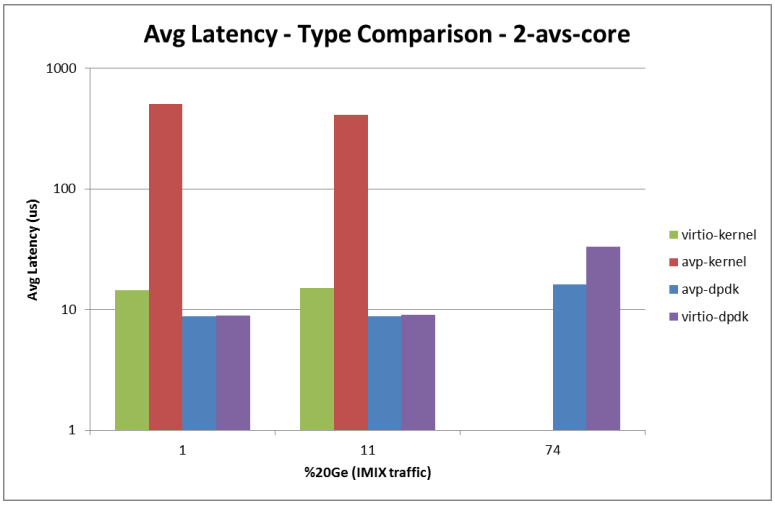


Figure 9: AVS average cut-through latency - IMIX fixed rates



SR-IOV Dimensioning

This section describes the dimensioning of SR-IOV PCI passthrough devices to provide direct I/O to a guest VM, bypassing the virtual switch. SR-IOV allows a device such as a network adapter to separate access to its resources among various PCIe hardware functions (VFs), allowing for multiple guests to share the same underlying NIC without multiplexing from a virtual switch (that is, AVS). The limitations of this configuration is that advanced features of AVS are not applied to traffic that has direct communication with the guest VM, such as QoS and ACLs. In addition, the SR-IOV port is not directly integrated with other tenant network such as virtual routers and DHCP services.

Supported SR-IOV NICs

For a list of supported NIC devices that can be configured for SR-IOV PCI passthrough, refer to *HCG 4.0 Planning: HCG 4.0 Hardware Requirements*.

Number of SR-IOV devices supported per node

The maximum number of VFs per node is dependent on the number of physical NICs and the maximum number of VFs that can be supported by a single NIC. The number of SR-IOV VFs is configured against the interface and will be validated against the maximum supported by the underlying port capabilities. The supported maximum for a given port is reported via the system host port list.

Number of SR-IOV devices supported per guest

The maximum number of SR-IOV VFs that can be mapped to a single guest is eight.

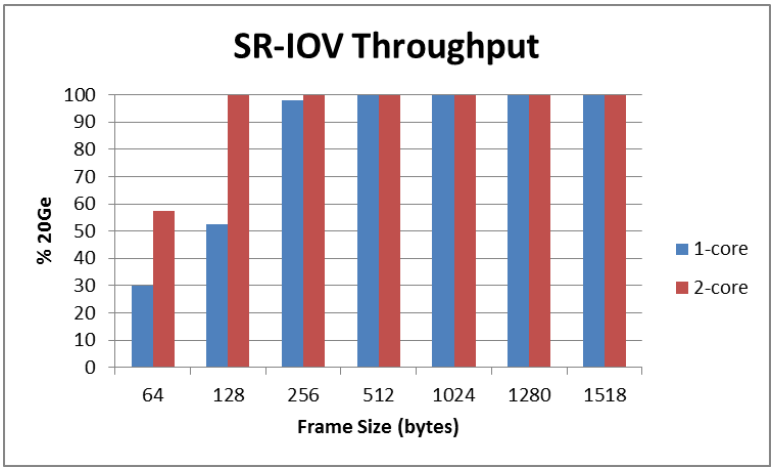
SR-IOV Dimensioning: Throughput Benchmarks

The following description and chart captures the data throughput for differing packet sizes using SR-IOV passthrough to the guest.

SR-IOV Dimensioning: Test Details

- CPU
  - Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
- RAM
  - DDR4-2133
- Physical ports
  - 2–82599 ports on separate PCI buses
  - 1 VF per physical port exposed to guest
- VM:
  - DPDK application running in the guest performing L3 routing
  - VM collocated on same socket as NICs
  - VM backed by 1G huge pages in the host
  - One and two forwarding cores measured
- IXIA Test
  - RFC2544 Bi-directional traffic @ 0.01% loss tolerance

Figure 10: SRIOV Throughput - % 20Ge



PCI-PT Dimensioning

This section describes the dimensioning of PCI passthrough (PCI-PT) devices to provide direct I/O to a guest VM, bypassing the virtual switch. PCI-PT allows guests to have exclusive access to PCI devices on the host and is mapped up directly into the guest; therefore the full capabilities of a NIC are available for the guest VM. The limitations of this configuration is that advanced features of AVS are not applied to traffic that has direct communication with the guest VM, such as QoS and ACLs. In addition, the PCI-PT port is not directly integrated with other tenant network such as virtual routers and DHCP services.

Supported PCI Passthrough NICs

For a list of supported NIC devices that can be configured for PCI passthrough, refer to *HCG 4.0 Planning: HCG 4.0 Hardware Requirements*.

Number of PCI-PT devices supported per node

The maximum number of PFs per node is dependent on the limitations of the host node itself. Since an entire NIC device must be available the guest, the limitations are based on the maximum number of NICs that can be installed into a single compute node.

Number of PCI-PT devices supported per guest

The maximum number of PCI-passthrough NICs that can be mapped to a single guest is eight.

PCI-PT Dimensioning: Throughput Benchmarks

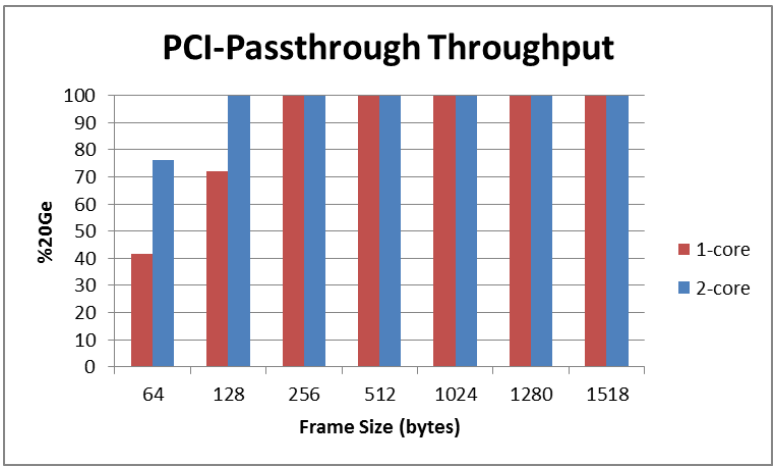
The following chart captures the data throughput for differing packet sizes using SR-IOV passthrough to the guest.

PCI PT Dimensioning: Test Details

- CPU
  - Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
- RAM
  - DDR4-2133
- Physical ports
  - 2- 82599 ports on separate PCI buses

- VM
  - DPDK application running in the guest performing L2 switching
  - VM collocated on same socket as NICs
  - VM backed by 1G huge pages in the host
  - One and two forwarding cores measured
- IXIA Test
  - RFC2544 Bi-directional traffic @ 0.01% loss tolerance

Figure 11: PCI-passthrough throughput - % 20Ge



6

Storage Node Dimensioning

This section describes dimensioning of storage nodes. Storage nodes provide dedicated storage for Cinder (volumes), Glance (images), Swift (objects), and Nova ephemeral storage in the Standard Configuration with Dedicated Storage. Storage nodes are deployed as a pair using a cluster of disks across the two nodes for scalability and redundancy.

For additional details and hard limits of the storage node hardware, including the list of supported NICs that are supported on storage nodes, refer to *HCG 4.0 Planning: HCG 4.0 Hardware Requirements*.

Storage Node CPU Requirements

Minimally a dual socket Intel Xeon® E5 26xx (Sandy Bridge) machine with eight cores per socket is required for the storage node.

BIOS/UEFI Settings for Storage Nodes

For storage nodes, the following settings must be configured in the platform BIOS or UEFI.

Table 10 **Storage BIOS/UEFI Settings.**

Parameter	Setting	Comments
Mode	BIOS or UEFI mode	UEFI Secure Boot and UEFI IPv6 PXE Boot are not supported in this release
Hyperthreading	Enabled or Disabled	
Power profile	Max Performance	
Minimum Processor Idle Power	No C States	
Boot Order	Hard Disk, PXE, USB	PXE for initial install, Hard Disk post-install

Storage Node Memory Requirements

A minimum of 64 GB RAM is mandatory for the storage node. This provides dedicated memory for the platform and storage functions as well as non-dedicated memory for caching and buffers. Additional memory may help performance by providing additional caching and buffers.

Storage Node Networking Requirements

The management and infrastructure interfaces are logical interfaces that can be mapped to physical interfaces or shared physical interfaces using VLANs for separation. The following management/infrastructure interface combinations are supported. Link aggregation of all interfaces is recommended for high availability.

- 1 GbE management network and 10 GbE infrastructure network
- 10 GbE management network and no infrastructure network (recommended configuration)
- 10 GbE network with management and infrastructure VLANs

If the storage node supports a board management via an iLO management interface, the Board Management Controller (BMC) module's Ethernet port can be connected to a dedicated private network. For controller node communication with the BMC modules, a VLAN on the internal management logical interface is configured on the board management network. It is recommended, but not required, to use the OAM network for controller access to the board management network.

Primary Disk Requirements for Storage Nodes

The primary disk on the storage node is used for boot, rootfs, logging, and scratch storage. A minimum primary disk size of 120 GB is mandatory with at least 10K RPM (SAS) or SSD drives.

The storage node primary disk is divided into a number of partitions. The following table shows the partitioning of the disk. The partition sizes are not configurable.

Table 11 **Storage Node Primary Disk Partitions and Sizes**

Partition	Size	Comments
rootfs	19.5 GB	Fixed
boot	500 MB	Fixed
scratch	7.5 GB	Fixed
log	7.5 GB	Fixed
platform	2 GB	Fixed

OSD Engineering for Storage Nodes

Object storage devices (OSDs) are the base unit of storage in the Ceph cluster deployed across the two storage nodes, with each OSD consisting of a single physical disk, and an equal number of OSDs on each storage node to provide redundancy. The quantity of OSDs, the disk specifications (size, speed, technology), and the disk controller specifications all contribute to the performance and capacity of Cinder and Glance storage. This section provides some data and guidelines for engineering the OSD cluster.

Hardware Requirements for OSDs

For a Ceph cluster OSD, a minimum specification of 300 GB 10K RPM SAS disks is mandatory. Faster SAS disks, such as 15K RPM SAS or SSDs are recommended. Two to eight OSDs per storage node (4 to 16 OSDs total) are currently supported.

SSD OSDs are highly recommended for more disk throughput, especially for VM disk operations with many VMs writing in parallel or if VMs have high disk throughput. An aggregate of 2000 IOPS and 200 MB/sec throughput (as measured by an IOPS tool executing random reads) for all VMs is a recommended point where using SSDs instead of SAS disks should be considered.

CPU/Memory Requirements for OSDs

The storage node must be dimensioned with at least 1.5 cores per OSD plus two cores for platform usage. A minimum of 64 GB memory is mandatory.

Scaling Rules/Recommendations for Storage Nodes

Three key performance parameters to be considered when engineering the OSD cluster are the required capacity, I/O operations per second (IOPS), and data throughput. The deployment choices that contribute to these parameters are disk size, disk speed, disk controller performance, and the number of disks. Processor speed, cores, and memory may contribute but to a lesser degree.

In general, scaling the cluster by adding more OSDs is preferable to scaling through the use of larger or faster devices. This is because adding OSDs not only increases capacity but also performance by providing more devices to spread reads and writes across. The following sections provide guidance on selecting the number of OSDs for a specific hardware configuration.

Storage Capacity for Storage Nodes

The total storage capacity is the sum of the Glance and Cinder storage required:

Storage Capacity Requirement = Glance Storage Requirement + Cinder Storage Requirement + Nova Ephemeral Storage Requirement + Switch Storage Requirement

Where:

Glance Storage Requirement is total image storage required (see Glance Dimensioning for Storage Nodes on page 45)
Cinder Storage Requirement is total volume storage required (see Cinder Dimensioning for Storage Nodes on page 45)
Nova Ephemeral Storage is the total volume storage required for ephemeral volumes that will use the Storage Node (See Ephemeral Dimensioning for Storage Nodes on page 46)
Swift Storage Requirement is the object pool storage size (see Swift Dimensioning for Storage Nodes on page 47)

Glance Dimensioning for Storage Nodes

To determine the logical disk capacity requirement for images, calculate the total storage required for images plus some margin to handle growth.

Glance Storage Requirement = N * AvgImageSize + N * AvgCachedImageSize + Margin

Where:

N = Total number of images
AvgImageSize = average size of all images on disk (e.g. from “qemu-img info <file>”)
AvgCachedImageSize = average size of all images on disk if --cache-raw is used
Margin = User-defined margin (recommend at least 25%)

NOTE: AvgCachedImageSize only applies to the Standard Configuration with Dedicated Storage and for images when the --cache-raw option is used. Use of --cache-raw is highly recommended for volume creation performance of qcow2 images.

Cinder Dimensioning for Storage Nodes

To determine the logical disk capacity requirement for Cinder volumes calculate the total storage for volumes required to be supported plus some margin to handle growth.

Cinder Storage Requirement = N * AvgVolSize + Margin

Where:

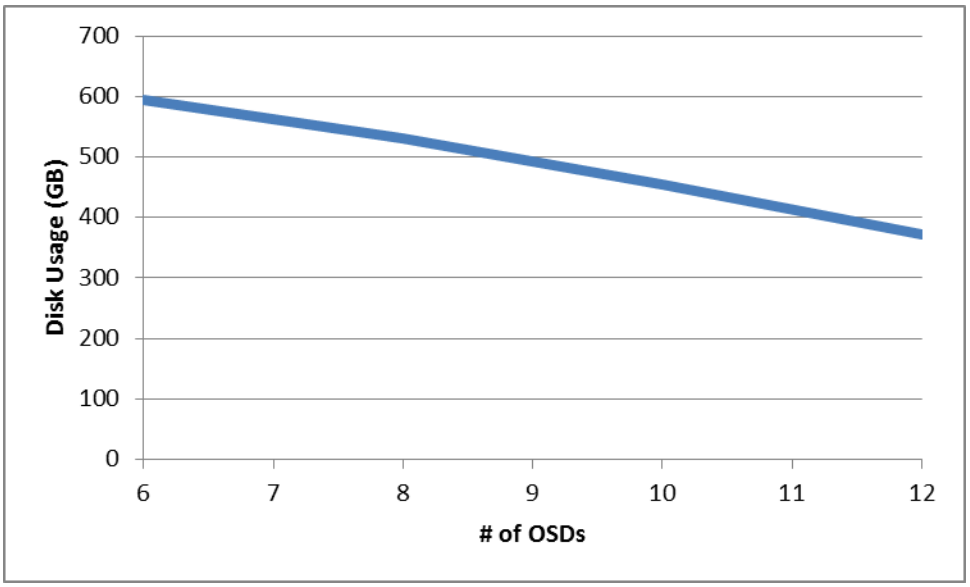
N = Total number of volumes required
AvgVolSize = Average volume size on disk in GB (note that sparse files are used)
Margin = User-defined margin (recommend at least 25%)

NOTE: Volumes are stored as sparse files; therefore the volume size on disk is based on how much disk space the VM uses, not the advertised volume size. The average volume size on disk should reflect the maximum average amount in the volume that the VM is expected to use over time. Furthermore, scheduling of volumes is based on advertised volume sizes so if creating many volumes at the same time, consider increasing the margin.

NOTE: The number of volumes is not necessarily the same as the number of VMs since VMs can have more than one volume and volumes can be created at different times than the VMs.

Storage utilization is affected by the number of OSDs so this must also be taken into account. For a reference deployment, 380 volumes with a distribution of 1, 2, 4, 10, 20, and 40 GB volumes averaging 12 GB were created. For 6 OSDs the actual disk usage per Storage Node was 371 GB, or about 1 GB per volume since sparse files are used to store the volumes. As the number of OSDs decreases, the actual storage utilization increased to 600 GB for the same volumes with 6 OSDs, a 38% increase in disk usage.

Figure 12: Disk Usage versus number of OSDs for 15K RPM SAS Disks.



Ephemeral Dimensioning for Storage Nodes

To determine the logical disk capacity requirement for ephemeral volumes for virtual machines specifically configured to use ephemeral volumes on the storage nodes, calculate the total storage for ephemeral volumes required to be supported plus some margin to handle growth.

This does not include VMs and ephemeral volumes that use the default ephemeral storage on the compute nodes.

Ephemeral Storage Requirement = N * AvgVolSize + Margin

Where:

N = Total number of volumes required
AvgVolSize = Average volume size on disk in GB (note that sparse files are used)
Margin = User-defined margin (recommend at least 25%)

Swift Dimensioning for Storage Nodes

Swift storage is not based on VMs but on the total number and size of the objects to be stored.

To determine the disk capacity requirement for the Swift object pool, calculate the total storage for Swift objects required to be supported plus some margin to handle growth.

Swift Storage Requirement = N * AvgObjectSize + Margin

Where:

N = Total number of volumes required
AvgObjectSize = Average size of swift objects on disk in GB
Margin = User-defined margin (recommend at least 25%, possibly more for object storage)

Storage Throughput for Storage Nodes

There are two dimensions of storage performance that should be considered: overall disk throughput, and throughput available for guest disk read/write. Overall throughput contributes to volume creation/VM boot times. Guest disk throughput is throughput available to the VMs for disk operations and is expressed in terms of I/O operations per second (IOPS) and data throughput.

Overall Throughput

Raw throughput of the Ceph cluster contributes to large operations such as volume creation. Throughput can be increased by increasing the number of OSDs (parallelism) and the raw speed of the individual OSDs.

The key performance parameters impacted by throughput is volume create times and to a lesser degree VM boot times. The volume creation/boot times in turn are affected by the volume sizes and the number of simultaneous volume creation/boot operations and background disk operations of VMs already running.

VM Guest Disk Throughput

To determine the guest throughput requirements, evaluate the disk I/O requirements for the VMs. The key parameters to evaluate are IOPS and throughput. The number of OSDs, disk speed and type, and disk controllers should be chosen to comfortably meet the disk requirement. If this is not done, high disk utilization will degrade VM disk performance through increased latencies as well as increase volume creation/deletion and boot times.

Guest IOPS Requirement = N * AvgIopsPerVM + Margin
Guest Throughput Requirement = N * AvgThroughputPerVM + Margin

Where:

N = number of VMs
AvgIopsPerVM = Average Logical Disk I/O throughput in IOPS for a VM
AvgThroughputPerVM = Average Logical Disk I/O throughput in MB/second for a VM
Margin = User-defined margin (for example, additional 25%)

SAS-Based Dimensioning for Storage Nodes

This section provides benchmarking information from a reference deployment of storage nodes with 300 GB SAS disks that can be used to determine an optimum number of OSDs based on throughput and capacity requirements. This is a reference only and results will vary depending on selection of server, storage media type and speed (SAS, SSD), disk controllers, and so on, and the choice of Storage Node platform should be done based on system-level behaviors rather than individual components.

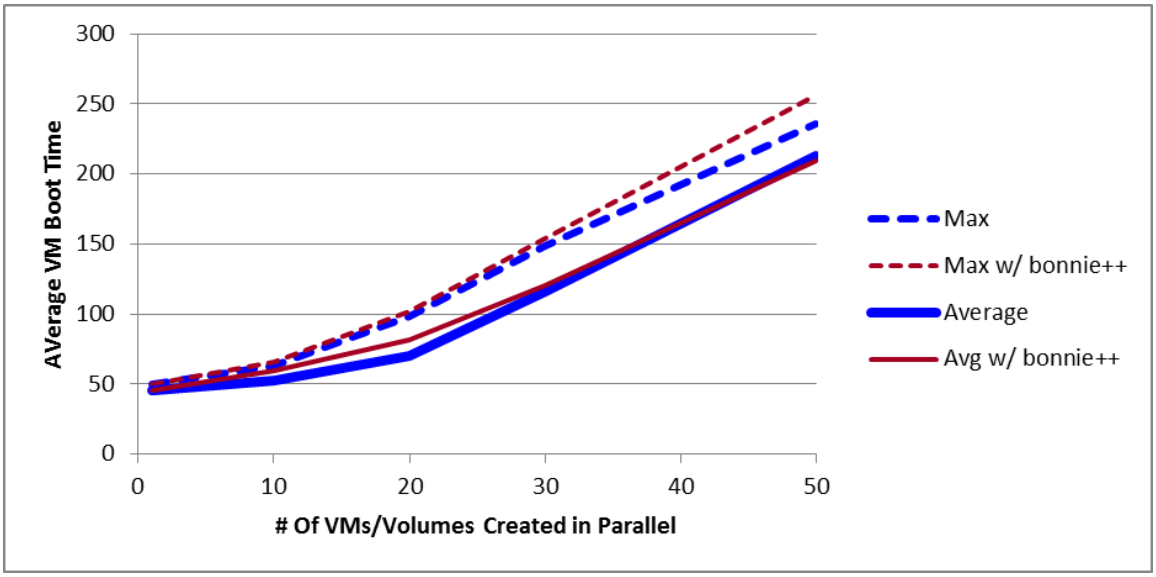
Storage Node hardware configuration:

- Dual Ivy Bridge processor - Intel Xeon® CPU E5-2690 v2 @ 3.00GHz
- Intel® Server Board S2600IP4 (Iron Pass)
- Intel® C606 Chipset
- Intel® 82599 NIC on Infrastructure Interface
- Ceph OSDs – 10 OSDs
  - Disk Model Number: HP EH0300FBQDD (SAS)
  - 300GB Capacity
  - 2.5” disk, 15K RPM
  - Sector size (logical/physical): 512B/512B
- VM image sizes: 1GB and 3.6 GB
- Ceph journaling enabled
  - INTEL SSDSC2BB24 240 GB SSD
- Cinder volume sizes: Distribution of 1, 2, 4, 10, 20, and 40GB
- Placement Group = 512

The following figure shows data for creation of VMs and Cinder volumes on SAS disks. A portion of the creation time is volume creation and volume creation is sensitive to disk throughput. Peak throughputs of 700 KB/s were observed during the benchmarking.

Volume plus VM create times were measured while running **bonnie++** on existing VMs as a background storage load in parallel with volume plus VM creation. The effect of running 40 and 80 VMs running **bonnie++** in parallel with the VM plus volume creates is shown in Figure 13: VM Boot Times with Volume Creation for 15K RPM SAS Disks. on page 49. As shown, there is some impact of the background load but it is not extensive.

Figure 13: VM Boot Times with Volume Creation for 15K RPM SAS Disks.



Ceph Journaling and Cache Tiering

Journal Functions

Each OSD on a storage node has an associated Ceph transaction journal that tracks changes to be committed to disk for data storage and replication, and if required, for data recovery. By default, it is collocated on the OSD, which typically uses slower but less expensive hard disk drive-backed storage. For faster commits and improved reliability, the journals can be assigned to a dedicated SSD on the storage node. HCG 4.0 strongly recommends configuring the journal function on a dedicated SSD if hard disks are used as OSDs. If SSDs are used, the journal can be collocated with the OSD with no performance impact.

Cache Tiering

Ceph cache tiering uses dedicated Ceph-caching storage hosts equipped with more expensive SSDs that provide high-speed Ceph-caching storage using SSDs in front of a set of Ceph-backing storage hosts using slower hard disk drive-based storage. Data is migrated to the cache tier when it is needed, maintained there for read-write operations, and removed when it is no longer required. Modified objects are removed to the storage tier (flushed), and unmodified objects are discarded (evicted). These operations are managed automatically by the Ceph objecter, which determines which tier to use for an object, and the cache tiering agent, which handles data migration between the tiers.

Cache tiering is beneficial to performance for systems where the same few data objects are accessed frequently and can have a detrimental performance for some workloads, especially those that access a large number of objects. Cache tiering must also be configured with a sufficiently large cache tier to avoid cache thrashing. HCG 4.0 recommends configuring cache

tiering only if the storage workload is known to benefit from cache tiering; otherwise performance may be degraded.

NOTE: To ensure that cache tiering is appropriate for the requirements, review the public Ceph documentation for caveats surrounding the use of this feature at http://docs.ceph.com/docs/master/rados/operations/cache-tiering/?highlight=tier#aword-of-caution.

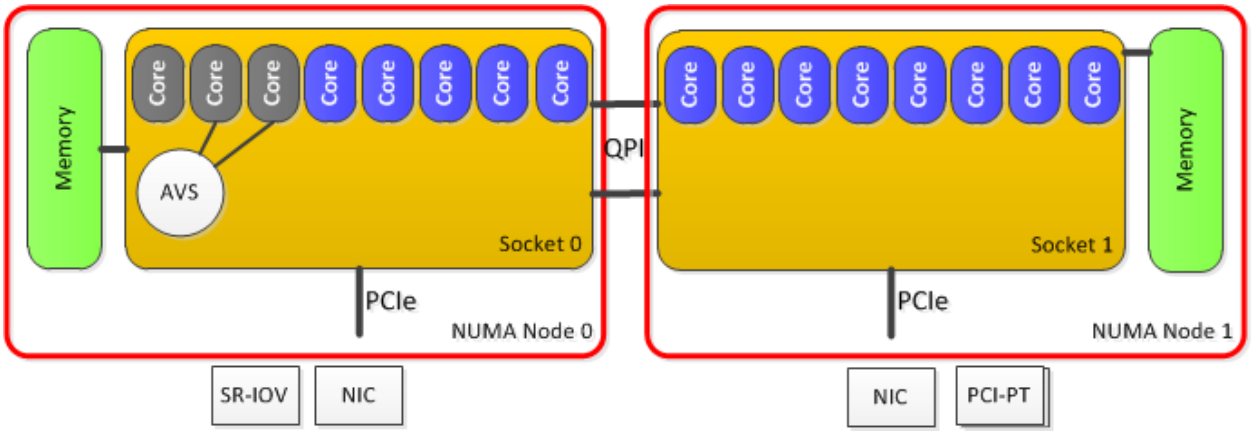
7

VM Scheduling and Placement

HCG 4.0 provides many features for controlling the scheduling and placement of VMs to provide multiple tiers of performance depending on requirements and to optimize resources where performance is not as critical.

An important part of understanding the capabilities is the understanding the Intel processor architecture. The following figure shows an example 8-core dual socket deployment with default configuration of the cores (core 0 reserved for the platform and AVS running only on socket 0 and using two cores). Hyperthreading is not shown. PCIe busses on each socket connect to devices such as NICs.

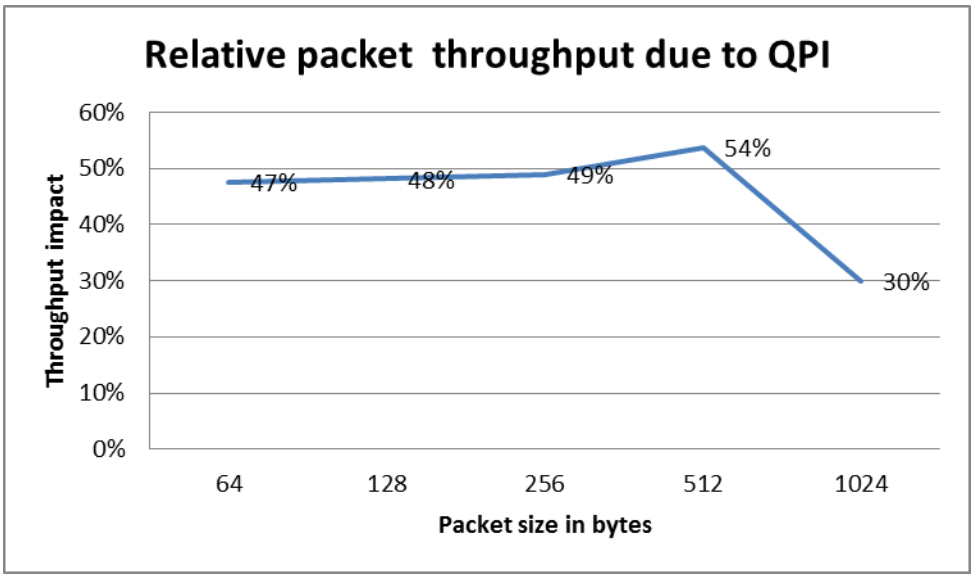
Figure 14: Dual Socket NUMA Example



The memory layout and socket interconnect using Quick Path Interconnect (QPI) play a critical role in performance and therefore VM placement decisions. All memory is accessible by applications (including VMs) on either socket but throughput and access times are dependent on whether the memory accessed is local to the socket or to memory attached to the other socket. The term used to describe this is Non-

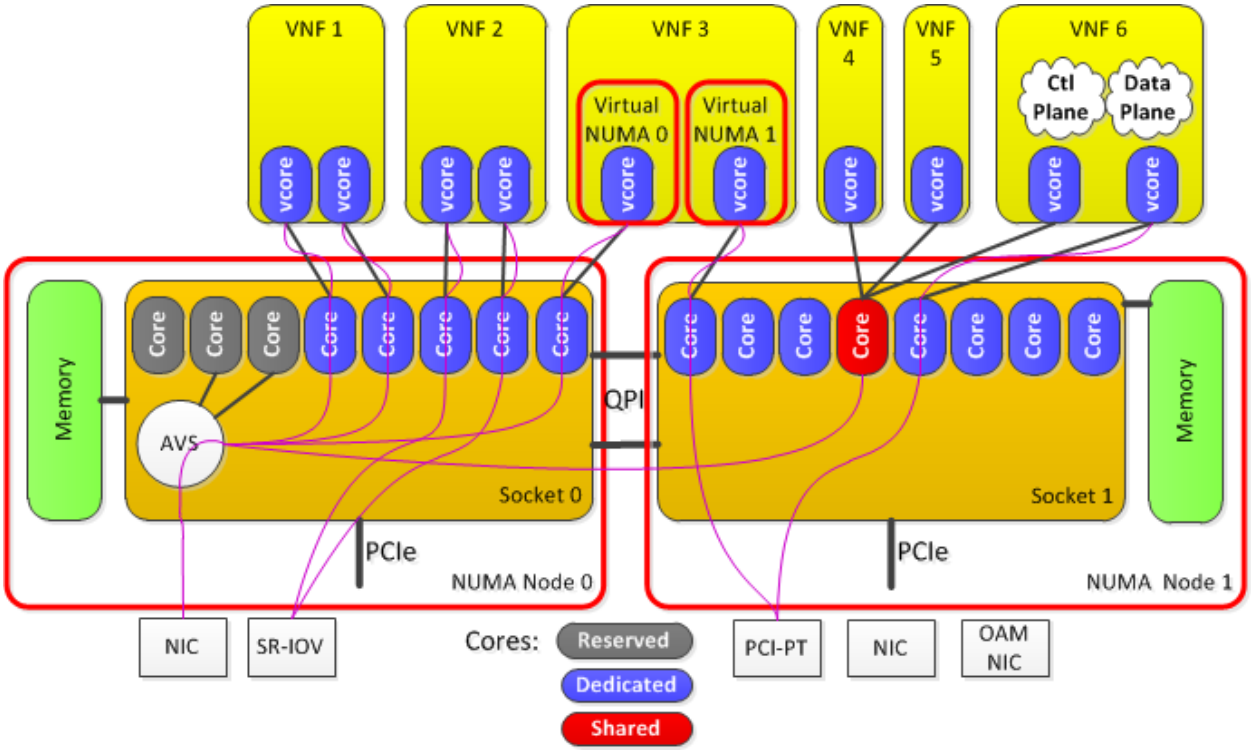
Uniform Memory Access (NUMA) because memory access times are different depending on the location of the memory being accessed. A NUMA node is defined as a socket and its directly connected memory. For maximum performance, all memory accesses associated with a VM must be within the same NUMA node, including AVS, NICs, and PCIe devices. The penalty for accessing memory in another NUMA node through the QPI bus is demonstrated may be 50% or more as shown in the following example chart of relative networking throughput for inter-NUMA node versus intra-NUMA node access.

Figure 15: Throughput Impact with Intra-NUMA Node Access



The following reference figure with VMs will be used as examples for the deployment discussion in the following subsections. For details on implementing the features described, refer to *Helion OpenStack Carrier Grade 4.0 Cloud Administration: Options for VM Customization*.

Figure 16: NUMA Architecture with VMs



VMs and NUMA Nodes

By default, a VM contains a single virtual NUMA node that is mapped to an available physical NUMA node on the compute node and the VM will only use memory from the backing physical NUMA node. HCG 4.0 also supports VMs with multiple virtual NUMA nodes to address VM scenarios where multiple NUMA nodes are needed.

- By default the virtual cores are distributed evenly amongst the virtual NUMA nodes. This can be overridden if needed.
- By default memory is distributed evenly across the virtual NUMA nodes. This can be overridden by assigning memory to each virtual zone using extra specs or image properties.
- By default the memory is allocated from the designated host physical NUMA node. This can be applied using a **strict** setting so the memory must come from the designated zone, or a **prefer** setting to allow the memory to come from another physical NUMA nodes to be used if necessary.
- By default the virtual NUMA nodes are mapped to available physical zones. This may be overridden so that each virtual NUMA node can be mapped to specific physical NUMA nodes. For example, in Figure 16: NUMA Architecture with VMs on page 53, VNF3 can map one core virtual NUMA node to physical NUMA node 0 and the other to physical NUMA node 1 to directly access resources in each physical NUMA node.

Dedicated or Shared CPU Policy for VMs

Two CPU policies are supported for VMs: Dedicated and Shared. These modes are controlled using extra specifications in the flavor definition.

The Shared CPU Policy is designed to maximize resource utilization for applications that do not have strict performance requirements, such as management applications. The shared VM virtual cores can be scheduled to run on any available physical core on the host up to an overcommit ratio of 16:1 virtual CPUs to a single core (this is not configurable). Available physical cores are any cores that are not reserved by the platform or dedicated to other VMs. VNF4 and VNF5 in the example figure are VMs with shared CPU policy.

The dedicated VM policy maps the VM virtual cores to cores on the compute node. This policy is mandatory for VMs that require carrier-grade performance and determinism and for VMs that use CPU scaling. Dedicating cores to VMs prevents resources from being overcommitted and prevents VM latency and throughput of the guest kernel from being not affected by scheduling of other VMs. Dedicated cores must be used for performance-critical VMs or performance targets will not be met. VNF1 and VNF2 in the example figure are examples of dedicated VMs.

Some VMs have both a high-performance data plane and a low performance control or management plane. A good example of this is a DPDK-based application. For mixed-mode VMs such as this the dedicated CPU policy can be supplemented to designate one virtual core as shared physical CPU. This optimizes resources by allowing the control plane virtual CPU to share its physical CPU with other VM control planes while maintaining the data plane cores as dedicated cores, thus providing the maximum performance for the data plane. To use this mode, a Shared CPU must be designated on the host (the core connected to VNF 4, VNF 5, and VNF 6 in Figure 16: NUMA Architecture with VMs on page 53) and a flavor extra spec must define which virtual CPU core to use the shared host CPU. VNF 6 in the same figure is an example of a dedicated VM with a shared virtual CPU.

Hyperthreading

Hyperthreading is an Intel processor technology that presents a single physical core as two logical cores, each with its own architectural state but with shared execution engine, caches, and bus interface.

The resource sharing allows a logical processor to process instructions when the other logical processor on the core is stalled waiting for data from memory. While hyperthreading presents a single physical core as two logical cores, it critical to be aware that it is a single processing pipeline whose performance gain is very dependent on the OS and applications. In some cases hyperthreading can actually cause a decrease in performance.

Hyperthreading should be used with an understanding of the applications that will share the physical cores and the requirements of the VMs deployed on it. Some general guidelines are:

- Each logical core depends on the other logical core stalls to acquire execution cycles so hyperthreading performance and latency are inherently not deterministic. Performance-critical VMs must not share a physical core with other VMs if hyperthreading is enabled.
- Hyperthreading works best when applications have frequent cache misses and external memory accesses because that provides the execution time for the other thread. If there are not a lot of processor stalls creating gaps for the other thread to execute in, the performance gain is less and can even result in no gain.
- Hyperthreading works worst when two logical cores start competing for the same cache lines and cause cache thrashing. VMs with high rates of cached memory access should not share physical core with any other VM.
- Low performance VMs and management VMs can share a physical core with another VM.

HCG 4.0 supports enabling or disabling hyperthreading. If hyperthreading is used, HCG 4.0 provides a CPU thread policy extra spec to optimize performance or optimize resources when using a dedicated CPU policy. Two options for the policy are provided:

- Isolate – Enforces the used of only one hyperthread logical core in each physical core. The sibling logical core will be unused by any other VM.
- Require- Requires that both siblings logical cores on the same physical core be assigned to the VM so that no other VMs will be scheduled to use the thread on that physical core. This feature is useful if the VM itself is optimized to leverage hyperthreading to improve performance.

NOTE: The CPU thread policies mitigate the effects of using hyperthreading. For maximum throughput and determinism, hyperthreading should be disabled.

Resource Placement

For VMs requiring maximum determinism and throughput, the VM must be placed in the same NUMA node as all of its resources, including VM memory, AVS, NICs, and any other resource such as SR-IOV or PCI-Passthrough devices. VNF 1 and VNF 2 in the example figure are examples of VMs deployed for maximum throughput with AVS or SR-IOV.

A VM such as VNF 6 in Figure 16: NUMA Architecture with VMs on page 53 will not have the same performance as VNF 1 and VNF 2 because access to the AVS is occurring across NUMA nodes. There are multiple ways to maximize performance for VNF 6 in this case:

- Use vswitch_numa_affinity flavor extra specs to force VNF 6 to be scheduled on NUMA nodes where AVS is running. This is the recommended option. The affinity may be **strict** or **prefer**:
  - **Strict** affinity guarantees scheduling on the same NUMA node as an AVS vSwitch or the VM will not be scheduled.
  - **Prefer** affinity uses best effort so it will only schedule the VM on a non-AVS NUMA node if no NUMA nodes with AVS switches have available cores. Note that **prefer** mode does

not provide the same performance or determinism guarantees as **strict** mode if it schedules the VM on an non-AVS NUMA node but it may be good enough for some applications.

- Configure AVS to run on both NUMA nodes 0 and 1. This reduces the number of cores available to run VMs on NUMA node 1 but only provides maximum performance if AVS connects to NICs on both sockets.
- Pin the VM to NUMA node 0 using flavor extra specs or image properties. This forces the scheduler to schedule the VM on NUMA node 0. However, this requires prior knowledge of which cores AVS is running on and will not run VMs on NUMA node 1 of any Compute Node even if AVS is running in that zone so it is not recommended.

If accessing PCIe devices directly from a VM using PCI-Passthrough or SR-IOV, maximum performance can only be achieved by pinning the VM cores to the same NUMA node as the PCIe device. For example, VNF1 and VNF2 will have optimum SR-IOV performance if deployed on NUMA node 0 and VNF6 will have maximum PCI-Passthrough performance if deployed in NUMA node 1. Options for controlling access to PCIe devices are:

- Use pci_numa_affinity flavor extra specs to force VNF6 to be scheduled on NUMA nodes where the PCI device is running. This is the recommended option because it does not require prior knowledge of which socket a PCI device resides on. The affinity may be **strict** or **prefer**:
  - **Strict** affinity guarantees scheduling on the same NUMA node as a PCIe Device or the VM will not be scheduled.
  - **Prefer** affinity uses best effort so it will only schedule the VM on a NUMA node if no NUMA nodes with that PCIe device are available. Note that prefer mode does not provide the same performance or determinism guarantees as strict, but may be good enough for some applications.
- Pin the VM to the NUMA node 0 with the PCI device using flavor extra specs or image properties. This will force the scheduler to schedule the VM on NUMA node 0. However, this requires knowledge of which cores the applicable PCIe devices run on and does not work well unless all nodes have that type of PCIe node attached to the same socket.

Huge Pages

Memory not reserved for platform use on the compute node is made available as backing memory for VM virtual memory allocation requests. By default, the VM memory is partitioned as 2 MB hugepages but this can be changed on a per-NUMA Node basis into pools of 4 KB, 2 MB, and 1 GB huge pages. Larger pages can reduce page management overhead and improve system performance for systems with large amounts of virtual memory and many running instances.

Memory page sizes for VMs can be specified using flavor extra specs or image metadata. With this option, small (4KB), large (largest available), any (same as large), 2 MB, or 1 GB page sizes for a VM can be specified. Explicit page size request definition is recommended because use of “large” or “any” may cause migration issues if the page size configuration on the destination node is different.

8

System Limits

This section captures key system limits. The overall limits are described in Table 12 on page 57, with a more verbose explanation for some limits in the following sections.

Table 12 **System Limits Summary**

Category	Capacity	Tested	Theoretical Limit	Comment
Compute Node	Max VMs per compute node	20	NA	Resource, configuration, and VM-dependent
Hardware	Max compute nodes in Standard Configuration with Dedicated Storage	100	NA	100 is maximum supported in this release
Hardware	Max controller nodes	2	2	Two required for redundancy
Hardware	Max storage nodes	2	8	Standard Configuration with Dedicated Storage only
Networking	Max (tenant) networks per system	450	NA	
Networking	Max provider networks per system	42	NA	
Networking	Max virtual routers and DHCP servers per system	20	NA	Assumes provider access partitioned to avoid 250 per compute node limit

Category	Capacity	Tested	Theoretical Limit	Comment
Networking	Max subnets per system	850	NA	Assumes provider access partitioned to avoid 250 per compute node limit
Networking	Max subnets sharing a compute node	250	NA	Limited by the maximum number of vSwitch ports per node
vSwitch	Max virtual routers/DHCP servers sharing a compute node	250	NA	Limited by the maximum number of vSwitch ports per node
vSwitch	Max entries in MAC address Table in node	2700	262144	1024 per network segment
vSwitch	Max ports per node	254	254	Includes virtual and physical ports
vSwitch	Max AVS physical ports per node	4		
vSwitch	Max ports per VM	8	8	KVM limit
vSwitch	Max security rules per node	16384	32512	128 rules per port
Storage Node	Max OSDs	16		8 OSDs per Storage Node
Storage Node	Min OSDs	4		2 OSDs per Storage Node
Virtual Machines	Max Number of Virtual Machines	1000	NA	Resource, configuration, and VM-dependent 1000 is maximum supported in this release
Heat Template	Max Heat resources	1000	NA	Max resources in a single Heat stack
Heat Template	Max VMs using local storage	80	NA	Over 10 compute nodes
Heat Template	Max VMs using Cinder volumes (dedicated storage)	40	NA	Backed by storage node with a single raw cache image used
Heat Template	Max VMs using Cinder volumes (controller storage)	10	NA	Backed by the controller

Maximum Number of Compute Nodes

Up to 100 Compute Nodes have been validated in a Standard Configuration with Dedicated Storage. The Standard Configuration with Controller Storage limit is driven not by compute node complement, but by storage capacity, storage performance, and volume creation/boot times. In practice, 10 computes may be the practical limit for the Standard Configuration with Controller Storage.

Maximum Number of VMs

Up to 1000 VMs have been tested and supported but the actual number supported may be less and is highly dependent on a large number of factors, including compute node resources, VM sizes, configuration parameters, and flavor settings and policies.

A heterogeneous mix of VMs and features were used for testing 1000 VMs. The mix uses VM with 1–4 cores, 1–4 GB memory, 600 MB and 3.6 GB images, 1/2/4/10/20/40 GB volumes, single and multi-NUMA nodes, local storage and remote storage, launch via nova and heat, dedicated and shared vCPUs, VM scaling, CPU scaling, and 1–3 network connections per VM.

Maximum Number of Tenant Networks

Up to 450 tenant networks are supported in the Standard Configurations.

Maximum Number of Subnets

Up to 850 subnets are supported across the system. The networks must be partitioned such that no more than 250 subnets are supported by any given compute node. This limit also applies to DHCP servers and virtual routers.

The system will redistribute DHCP servers and virtual routers across the available compute nodes, assuming they meet the provider network requirements for the given network attachments of the subnet.

The scheduling of DHCP servers and virtual routers will ensure they are evenly distributed amongst the available compute nodes when they are configured and when new hosts become

available. The redistribution is performed periodically; therefore the servers and routers will be rescheduled if an imbalance is detected.

The distribution of DHCP servers and virtual routers helps to mitigate the impact of compute host failures by reducing the scope of the rescheduling of the failed services, in addition to allowing for higher throughput capabilities for North-South traffic due to the distributed nature of the routing function across all compute hosts.