



Reference Architecture:

Red Hat Hyperconverged Infrastructure for Virtualization

Version 1.0

Feng Xia, Miro Halas

May 10, 2019

DRAFT COPY

Lenovo Open Cloud Red Hat Hyperconverged Infrastructure for Virtualization^[1], aka. RHHI-v, is a multi-server virtualization environment built for scalability and high performance based on Red Hat Hyperconverged Infrastructure (RHHI) version 1.5. This reference environment provides a comprehensive example demonstrating server configuration, network cabling, switch configuration, and RHHI deployment.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 11 |
| 2 | Business problem and business value | 14 |
| 2.1 | Business problem | 14 |
| 2.2 | Business value | 15 |
| 3 | Requirements | 16 |
| 3.1 | Functional requirements (use cases) | 16 |
| 3.2 | Non-functional requirements | 18 |
| 3.2.1 | Fault tolerance | 18 |
| 3.2.2 | High availability | 19 |
| 3.2.3 | Maintenance | 19 |
| 3.2.4 | Ease of installation | 20 |
| 3.2.5 | High performance | 20 |
| 4 | Architectural Overview | 22 |
| 5 | Component model | 25 |
| 6 | Operational model | 30 |
| 6.1 | Hardware | 30 |
| 6.1.1 | Lenovo SR650 Servers | 30 |
| 6.1.2 | Lenovo 930-x RAID controller | 32 |
| 6.1.3 | Lenovo Network Switches | 32 |
| 6.1.3.1 | Lenovo RackSwitch G8272 (10Gb) | 33 |
| 6.1.3.2 | Lenovo RackSwitch G8052 (1Gb) | 33 |
| 6.1.3.3 | Lenovo ThinkSystem NE2572 (25Gb) | 34 |
| 6.2 | Software | 35 |
| 6.2.1 | Red Hat Virtualization (RHV) | 36 |
| 6.2.2 | Red Hat Gluster Storage (RHGS) | 38 |

| | | |
|----------|---|-----------|
| 7 | Deployment considerations | 40 |
| 7.1 | Server Configurations | 40 |
| 7.1.1 | Disk configurations | 40 |
| 7.1.2 | RAID configurations | 41 |
| 7.1.3 | Firmware | 42 |
| 7.2 | Network Configurations | 43 |
| 7.2.1 | Design Conventions | 43 |
| 7.2.2 | Network Overview | 45 |
| 7.2.3 | Topology | 46 |
| 7.2.4 | Logical networks | 47 |
| 7.2.5 | VLANs | 49 |
| 7.2.6 | Subnets | 50 |
| 7.2.7 | Map VLAN to Server NIC | 51 |
| 7.2.7.1 | Server NIC connections | 51 |
| 7.2.7.2 | Server NIC bonding interfaces | 52 |
| 7.2.7.3 | Server NIC bridge interfaces | 53 |
| 7.2.8 | Map Server NIC to Switch | 53 |
| 7.2.9 | Define Cable Schema | 54 |
| 7.2.10 | Configure Switch Ports | 55 |
| 7.3 | Storage | 57 |
| 7.3.1 | Gluster | 57 |
| 7.3.1.1 | Bricks | 57 |
| 7.3.1.2 | Volumes | 59 |
| 7.3.1.3 | Volume type — Replicated | 60 |
| 7.3.2 | NFS | 60 |
| 7.3.3 | RHV storage domains | 61 |
| 7.3.4 | LVM cache | 62 |
| 8 | Deployment | 65 |
| 8.1 | Prerequisite | 65 |
| 8.2 | Bootstrap appliance | 65 |
| 8.2.1 | Bootstrap services | 65 |
| 8.2.2 | Bootstrap networks | 66 |

| | | |
|-----------|--|-----------|
| 9 | Appendix: Bill of Material | 69 |
| 9.1 | Switches | 69 |
| 9.1.1 | 1Gb switch | 69 |
| 9.1.2 | 10Gb switch | 69 |
| 9.1.3 | 25Gb switch | 70 |
| 9.2 | Servers | 71 |
| 9.2.1 | SAS SSDs | 72 |
| 9.2.2 | 25Gb NIC | 73 |
| 9.3 | Software | 73 |
| 9.4 | Other recommendations | 74 |
| 9.4.1 | TPM 2.0 and Secure Boot | 74 |
| 9.4.2 | ThinkSystem XClarity Controller Standard to Enterprise Level | 74 |
| 9.4.3 | Best recipes | 75 |
| 10 | Appendix: Implementation Questionnaire | 77 |
| 10.1 | Switches | 77 |
| 10.1.1 | Switch port sizing worksheet | 77 |
| 10.2 | Storage sizing worksheet | 78 |
| 10.3 | Network mapping | 79 |
| 10.3.1 | Strategy 1 | 79 |
| 10.3.2 | Strategy 2 | 80 |
| 10.3.3 | Strategy 3 | 80 |
| 10.3.4 | Strategy 4 | 81 |
| 10.3.5 | Strategy 5 | 81 |
| 10.3.6 | Network mapping worksheet | 82 |
| 11 | Contributors | 84 |
| 12 | Appendix: Apply switch configurations using CLI | 85 |
| 12.1 | Login to switch | 86 |
| 12.2 | Enter config mode | 86 |
| 12.3 | Apply access mode configurations | 87 |
| 12.4 | Apply trunk mode without LACP | 87 |
| 12.5 | Setup inter-switch-links (ISL) | 87 |
| 12.5.1 | Connect the 2 switches with the ISL cable line | 88 |
| 12.5.2 | G8052 ISL configuration | 88 |

| | |
|--|------------|
| 12.5.3 For G8272 ISL configuration | 89 |
| 12.6 Apply trunk + port-channel + vlag for server ports redundancy | 90 |
| 13 Appendix: Configure RHEL bonding interfaces | 92 |
| 14 Appendix: Map RHEL interface to RHV network | 96 |
| 15 Appendix: LOC RHHI-v storage performance test | 100 |
| 15.1 Performance index | 102 |
| 15.2 Type of tests | 102 |
| 15.3 Test Setup | 104 |
| 15.3.1 Gitlab CE | 105 |
| 15.3.2 PTS | 105 |
| 15.3.3 Jenkins | 106 |
| 15.3.4 ELK stack | 106 |
| 15.3.4.1 Elasticsearch | 106 |
| 15.3.4.2 Logstash | 107 |
| 15.3.4.3 Kibana | 107 |
| 15.4 Theoretical values | 107 |
| 15.5 Sequential Read Test | 108 |
| 15.5.1 Checkpoints: LVM(1), GlusterFS(4), VM(5) | 109 |
| 15.6 Sequential Write | 110 |
| 15.6.1 Checkpoints: LVM(1), GlusterFS(4), VM(5) | 111 |
| 15.7 Sequential Mixed | 113 |
| 15.7.1 Checkpoints: LVM(1), GlusterFS(4), VM(5) | 113 |
| 15.8 Random Read | 114 |
| 15.8.1 Checkpoints: LVM(1), GlusterFS(4), VM(5) | 115 |
| 15.9 Random Write | 116 |
| 15.9.1 Checkpoints: LVM(1), GlusterFS(4), VM(5) | 117 |
| 15.10 Random Mixed R/W | 118 |
| 15.10.1 Checkpoints: LVM(1), GlusterFS(4), VM(5) | 119 |
| 15.11 Conclusions | 120 |
| 16 References | 122 |

List of Tables

| | | |
|------|--|----|
| 7.1 | Red Hat Hyperconverged Infrastructure for Virtualization Server disk configurations | 40 |
| 7.2 | Red Hat Hyperconverged Infrastructure for Virtualization Server Disk Configurations w/ support of file-based storage | 41 |
| 7.3 | Red Hat Hyperconverged Infrastructure for Virtualization Server RAID Configurations | 41 |
| 7.4 | Red Hat Hyperconverged Infrastructure for Virtualization Server Firmware | 42 |
| 7.5 | Example Red Hat Hyperconverged Infrastructure for Virtualization logical network VLANs | 50 |
| 7.6 | Red Hat Hyperconverged Infrastructure for Virtualization Logical Network Subnets | 50 |
| 7.7 | Red Hat Hyperconverged Infrastructure for Virtualization VLAN to Server's NIC Mapping | 51 |
| 7.8 | Red Hat Hyperconverged Infrastructure for Virtualization VLAN to Server Bonding Interfaces Mapping | 52 |
| 7.9 | Red Hat Hyperconverged Infrastructure for Virtualization VLAN to Server Bonding Interfaces Mapping | 53 |
| 7.10 | Red Hat Hyperconverged Infrastructure for Virtualization Server to Switch Connections | 54 |
| 7.11 | Red Hat Hyperconverged Infrastructure for Virtualization Server-Switch Cable Schema of a 3-server Configuration | 55 |
| 7.12 | Example of G8052 #1 Port Configurations | 56 |
| 7.13 | Example of G8052 #2 Port Configurations | 56 |
| 7.14 | Example of G8272 #1 & #2 Port Configurations | 56 |
| 7.15 | Red Hat Hyperconverged Infrastructure for Virtualization Gluster bricks and mounting point | 58 |
| 7.16 | Red Hat Hyperconverged Infrastructure for Virtualization storage to RHV storage domain mapping | 62 |
| 8.1 | Bootstrap Appliance Core Services | 66 |
| 8.2 | Bootstrap Appliance Add-on Services | 66 |

| | |
|---|-----|
| 9.1 Red Hat Hyperconverged Infrastructure for Virtualization RackSwitch G8052 1Gb bill of materials, rear to front | 69 |
| 9.2 Red Hat Hyperconverged Infrastructure for Virtualization RackSwitch G8272 10Gb Bill of Material | 70 |
| 9.3 Red Hat Hyperconverged Infrastructure for Virtualization RackSwitch NE2572 25Gb Bill of Material | 70 |
| 9.4 SR650 Bill of Materials based on SATA SSDS | 71 |
| 9.5 SR650 Bill of Materials based on SAS SSDs | 72 |
| 9.6 SR650 Bill of Materials based on 25G NIC | 73 |
| 9.7 LOC 3-node RHHI-v software BOM, 3 year premium service support . . | 73 |
| 9.8 Red Hat Hyperconverged Infrastructure for Virtualization SR650 server best recipe | 75 |
| 10.1 Implementation Worksheet — Switches | 77 |
| 10.2 Switch port sizing worksheet | 78 |
| 10.3 Storage sizing worksheet (replicated gluster) | 78 |
| 10.4 IP address sizing worksheet, all separated. | 79 |
| 10.5 IP address sizing worksheet, campus internal == BMC, ovirt separated . . | 80 |
| 10.6 IP address sizing worksheet, campus internal == ovirt, BMC separated . . | 80 |
| 10.7 IP address sizing worksheet, ovirt == BMC, Campus internal separated . . | 81 |
| 10.8 IP address sizing worksheet, a flat network. | 82 |
| 10.9 LOC RHHI network mapping worksheet | 82 |
| 12.1 Example of LOC RHHI-v Server-Switch Cable Schema | 85 |
| 15.2 LOC RHHI-v Sequential Read parameters | 108 |
| 15.3 LOC RHHI-v Sequential Write parameters | 111 |
| 15.4 LOC RHHI-v Sequential Mixed parameters | 113 |
| 15.5 LOC RHHI-v Random Read parameters | 114 |
| 15.6 LOC RHHI-v Random Write parameters | 117 |
| 15.7 LOC RHHI-v Random Mixed parameters | 118 |

List of Figures

| | | |
|-----|---|----|
| 3.1 | Functional requirements | 17 |
| 4.1 | Lenovo Open Cloud RHHI-v Architecture of a 3-server Configuration . | 23 |
| 5.1 | RHHI-v component model | 26 |
| 6.1 | Lenovo ThinkSystem SR650 | 30 |
| 6.2 | ThinkSystem RAID 930-8i RAID Controller | 32 |
| 6.3 | Lenovo RackSwitch G8272 | 33 |
| 6.4 | Lenovo RackSwitch G8052 | 33 |
| 6.5 | Lenovo RackSwitch NE2572 | 34 |
| 6.6 | Red Hat Hyperconverged Infrastructure for Virtualization architecture (source: [10]) | 36 |
| 6.7 | Red Hat Virtualization Self-Hosted Engine Architecture (source: [13]) . | 37 |
| 6.8 | Red Hat Gluster Storage architecture (source: [15]) | 39 |
| 7.1 | Server-Switch network convention | 44 |
| 7.2 | Lenovo Open Cloud RHHI-v network topology | 46 |
| 7.3 | Lenovo Open Cloud RHHI-v logical networks | 47 |
| 7.4 | Red Hat Hyperconverged Infrastructure for Virtualization Server Map VLAN to NIC | 52 |
| 7.5 | Lenovo Open Cloud Platform Server to Switch | 54 |
| 7.6 | Red Hat Hyperconverged Infrastructure for Virtualization Host Logical Volumes and Gluster Bricks | 58 |
| 7.7 | 3-server Red Hat Hyperconverged Infrastructure for Virtualization Gluster cluster overview | 59 |
| 7.8 | Red Hat Hyperconverged Infrastructure for Virtualization Host Logical Volumes and NFS | 61 |
| 7.9 | Red Hat Hyperconverged Infrastructure for Virtualization node disk configurations w/ LVM cache enabled | 63 |
| 8.1 | LOC Bootstrap Appliance Networks | 67 |

| | |
|--|-----|
| 13.1 Example of LOC RHHI-v server bonding interface and their switch connections | 93 |
| 14.1 View RHV Network List | 96 |
| 14.2 Create New RHV Network | 97 |
| 14.3 Setup RHV Network VLAN | 97 |
| 14.4 Link RHV Network to a bonding interface | 98 |
| 15.1 LOC Storage Performance Test Data Path | 101 |
| 15.2 LOC storage performance test setup | 104 |
| 15.3 Sequential Read Throughput, checkpoints 1,4,5 | 109 |
| 15.4 Sequential Read IOPS, checkpoints 1,4,5 | 110 |
| 15.5 Sequential Write Throughput, checkpoints 1,4,5 | 111 |
| 15.6 Sequential Write IOPS, checkpoints 1,4,5 | 112 |
| 15.7 Random Read Throughput | 115 |
| 15.8 Random Read IOPS | 116 |
| 15.9 Random Write Throughput | 117 |
| 15.10 Random Write IOPS | 118 |
| 15.11 Random Mixed Throughput | 119 |
| 15.12 Random Mixed IOPS | 120 |

1

Introduction

Technology evolves fast. In the age of cloud computing, user and workload owner expects a feeling of unlimited resources available on-demand instantly, fast, cheap, and reliable. To build and to maintain such an infrastructure is taken for granted as if servers, switches, even cables, have fixed recipes that with a swing of magic wand, they are wired, configured, provisioned, deployed, and readily consumable.

It is not quite as easy, but not too far from reality either. Technology of server virtualization has been the talk of the day for the last two decades. Making a physical server shareable among workloads through a hypervisor that hides complexity of hardware and gives its tenant a sense of a complete machine dedicated to its own use is still the backbone of many software development and deployment. If you go to AWS today and register an EC2 instance, your expectation is that this machine can be treated just like a real server with the configurations you have requested — CPU, memory, and so on, except it is virtualized, meaning that its resources are shared, but you don't know, and you don't care.

Regardless which software your company is working on, the inevitable question you will have to answer at some point is, “where to put it?” This is both a question for development, and for production. Development stack is often contained within a single host environment — it is not uncommon to see an entire stack inside a single virtual machine, sometimes even run as a production instance by some entrepreneurs. But once your design grows, you start seeing the word “scalability”, “high availability”, and that's when a single host environment will start to break. To have more “host environments”, you are facing two choices: physical servers, or virtual.

Here we are using the word “virtual” referring to any virtualized environment in general. This includes cloud such as AWS, OpenStack, and containers. Anything but phys-

ical is thus, virtual. But virtual is the outcome, not the starting point. As infrastructure architect, or an infrastructure administrator, this is the result you are delivering to your customer, and you are responsible to build it, to run it, and to keep it up to date.

Whether the seemingly unlimited resources are presented as virtual machine, container, or some other forms, these are software in its core, and they need to run on a server, have access to storage, to network, and to be coordinated and orchestrated by its managerial software. Therefore, if we take a broad view of all these software, there are essentially two groups of them: management, and workloads.

Management These are software responsible to coordinate and orchestrate workloads that can run distributively across network over multiple infrastructure. It has the meta model of its workload, has the capability of collect and monitor its subordinates, and to create and delete workloads, and to have an UI or API that communicates to the world outside itself.

Workloads These are software worker bees that makes up the actual application. It responds to customer input, performs business logic, and reads and writes data as the logic determines.

So as a starting point, before any of the [workload](#) software exist, you have to find a place for your [management](#) software, and that's where the Lenovo Open Cloud (LOC) RHHI-v fits in — it describes a comprehensive approach from ground up that provides RHEL 7.5 on baremetal and a virtual-machine ready platform supported by RHHI 1.5 that gives users a foothold for the [management](#) piece, and scale it out to include more hardware and software as your resources grow.

In this RA we describe the system architecture for the Lenovo Open Cloud (LOC) RHHI-v based on Lenovo ThinkSystem servers, Lenovo network switches, and RHHI 1.5. It provides details of the hardware requirements and configuration. It also describes the network architecture and switch configurations. The hardware bill of materials is provided for all required components to build the Open Cloud cluster. An example deployment is used to show how to prepare, provision, and deploy Red Hat Hyperconverged Infrastructure for Virtualization on Lenovo ThinkSystem servers and Lenovo network switches.

The target audience for this Reference Architecture (RA) is system administrators of

virtualized environment, or system architects who is considering to deploy workloads into virtual machines. Some experience with Lenovo server and switch configurations are highly desirable. Experience with Red Hat implementation of virtualization and Red Hat Enterprise Linux (RHEL) version 7.5 is helpful, but it is not required.

2

Business problem and business value

2.1 Business problem

Many software continue to be deployed to either physical server (aka. baremetal), and virtual machine. Container based deployment is gaining ground, especially in software development, and for application whose architecture fits for containerization. However, virtual machine still hold advantage in areas such as security, backwards compatibility, human resource who has knowledge of the technology, technology maturity, support, and operational experience.

Within the virtualization sphere, kernel-based virtual machine (aka. KVM) is a popular hypervisor. It is open source and production ready. But this is only half the battle. Infrastructure must also provide adequate storage backend, networking, and management of all these aspects.

Building such a solution from the ground up is not an easy task. Software is an experience product, meaning that one has to see it in action and get hands on experience in order to relate to its function, strength, and limitation. How to quickly deploy an instance in lab for evaluation without yet having the full knowledge of all the components from hardware all the way up some web UI of hundreds of options an admin can turn on and off is a constant struggle.

On the other hand, another parity is the development environment vs. production environment. A development infrastructure, if deviated from the production one, can seriously risk the stability once the application moves off its nest to the real world. How to build and maintain an identical stack that fit for both development who wants agile, small foot print, light weight, constantly being rebuilt (and constantly being

broken by testing, by bugs, and so on), and for production that stability, scalability, security are as important as function.

Then, there is a whole world of hardware pieces to match — server, CPU, disk, disk controller, network card, switches, down to cables, server height, weight, power consumption, facing orientation, air flow, and an army of software configurations at firmware level, at operating system level, at application level, and so on.

2.2 Business value

This document will describe in details how Lenovo has used Red Hat Hyperconverged Infrastructure for Virtualization^[1], an oVirt based hyperconverged infrastructure solution, on top of selected Lenovo servers and switches, to build such a virtualization environment. We will leave out some topics such as floor weight, rack foot print, but will focus on the configuration of hardware and software, and choice of settings based on our experience and test results.

There isn't one size fit for all. By looking into the technology and describing how they work with hardware and network, this document provides a broader scope of knowledge than the Red Hat Hyperconverged Infrastructure for Virtualization 1.5^[2]. Once one has grasped the information in this document, one will be well positioned to consider and discuss options in both the hardware and the software with a clear understanding of where they fit together and why so.

3

Requirements

3.1 Functional requirements (use cases)

The primary use case of Red Hat Hyperconverged Infrastructure for Virtualization is to deploy software application in virtual machine. One thus gains the benefit of virtualization such as sharing common physical resource, resilience to hardware failure through backup, snapshot, export and import, and live migration. Further, virtual machine can usually be converted into a format portable to other virtualization environment, eg. VMware and KVM.

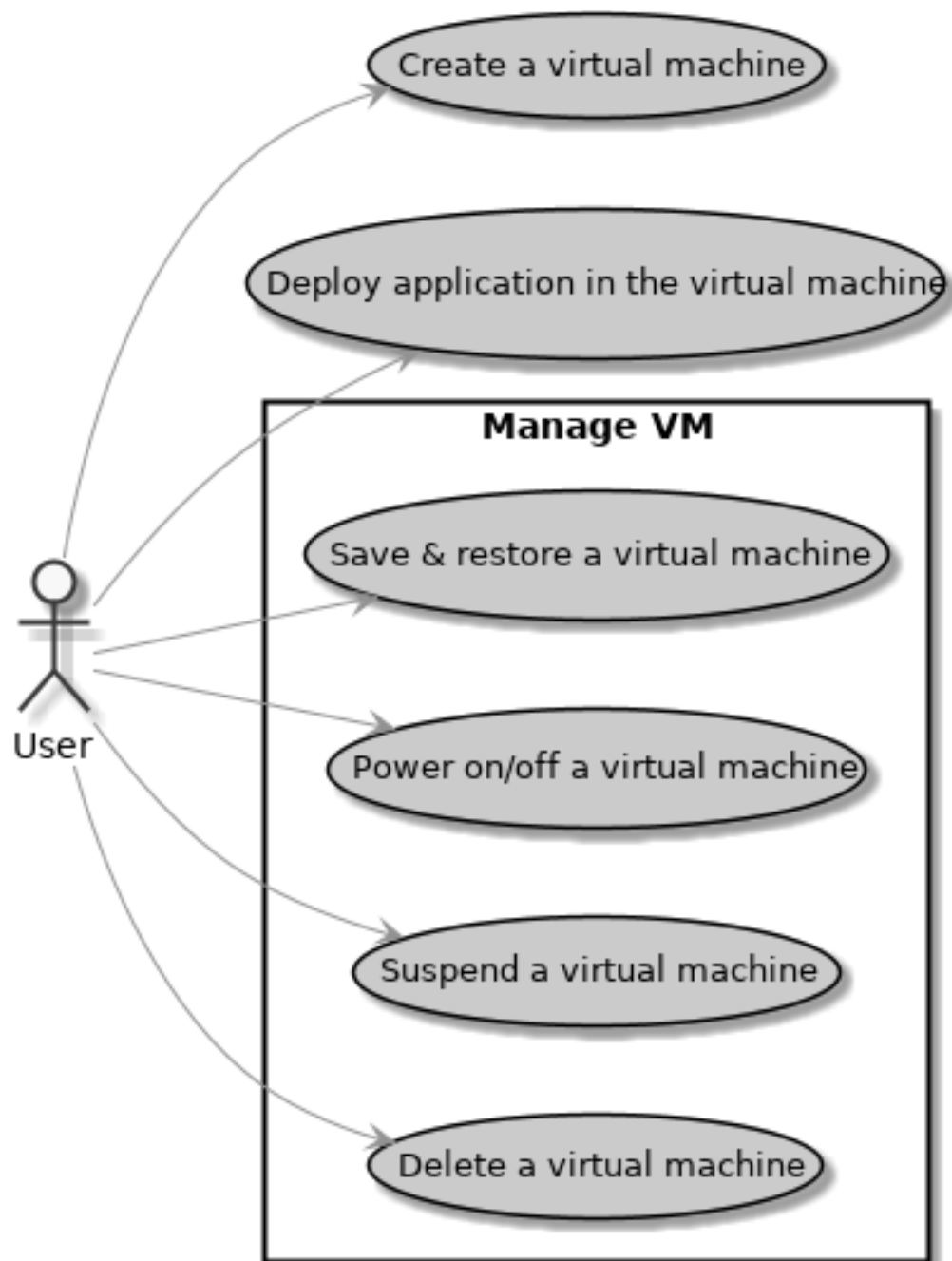


Figure 3.1: Functional requirements

1. As a cloud user, I want to deploy application in virtual machine(s), so that the deployment is portable to VMware and KVM.

2. As a cloud user, I want to create VM from disk image, so that I can reuse an existing VM image.
3. As a cloud user, I want to create VM from a configuration template that defines VM attributes, such as CPU, memory, disk size, base image, and so on, so that I will get an identical VM if using the same values.

There are many dials in a virtual machine that one can play with. The common ones are easy — CPU, memory, disk, and network. In addition, Red Hat Hyperconverged Infrastructure for Virtualization is set to meet the [affinity](#) requirement, that is to pin a virtual machine to a physical node.

4. As a cloud user, I want to pin VM to a specific physical node, so that I can use a HA model on my application by mandating VMs over different physical nodes so to prevent single point of server failure.
5. As a cloud user, I want to pin VM to CPU threads, IO threads, emulator threads, or NUMA nodes, so that I can meet performance requirement of my workload.
6. As a cloud user, I want to take snapshot of VM to preserve its execution states so that I can restore the states and resume the execution from that.

3.2 Non-functional requirements

In addition to the use cases above, a private cloud needs fault tolerance, high availability, maintenance, ease of installation, and high performance.

3.2.1 Fault tolerance

Fault tolerance is covering cases to maintain the integrity of the infrastructure when some hardware resource become unavailable. The most important consideration is the safety of data — management data, workload execution states, and user data.

1. As a cloud admin, I want the data saved in the storage resilient in the event of single disk failure or single server failure.

2. As a cloud admin, I want to create backup of VM snapshots, so that snapshots can be restored from backup in the event of snapshot corruption. This, however, does not dictate whether backups are themselves known-good.
3. As a cloud admin, I want the infrastructure to meet all the functional requirements when a single node is broken. This includes malfunction of any sub-components inside the node, eg. NIC, RAID controller.
4. As a cloud admin, I want to have redundancy in network connections, so that losing one connection in the pair will not disrupt the networks on that line logically.

3.2.2 High availability

This reference architecture looks over two common failure modes — server failure, and network connection loss, so to alleviate disruption of service:

1. As a cloud admin, I want cluster management application cut over to a working node in the cluster automatically when its underline host breaks its function.
2. As a cloud admin, if a network has more than one connection for redundancy purpose, I want the network traffic to cut over to a working connection automatically if the one it is on is broken.

However, some part of the Red Hat Hyperconverged Infrastructure for Virtualization is highly sensitive to networking disruption, which is intrinsic to its underline technology.

3.2.3 Maintenance

Infrastructure requires regular maintenance. The key consideration is to strategize a maintenance by understanding its impact and its goal. Some will inevitably disrupt the service and seemingly break high availability. But in the long run, it brings a healthy system, sound integrity, and long-running operation without which HA is not possible:

1. As a cloud admin, I want to know which virtual machines are running on which physical node, so that I can evaluate the impact of shutdown of that node in the event of server maintenance or repair.
2. As a cloud admin, I want to know which physical node and VMs are running on a network, so that I can evaluate the impact of shutdown of that network in the event of network maintenance or repair.
3. As a cloud admin, I want to migrate VM to another physical node manually in the event of server shutdown for maintenance or repair¹.

3.2.4 Ease of installation

One of this document is to make it easy for data center to build such an infrastructure for their own use or for their customer.

1. As a cloud admin, I want to build the infrastructure on baremetal servers with a configuration file as input, so that the end state of the infrastructure is fully descriptive.
2. As a cloud admin, I want to rebuild the infrastructure to the same state each time if the configuration inputs remain the same, so that the construction process is deterministic.
3. As a cloud admin, I want to automate the entire installation, so that building one takes single input — the configuration file, and one CLI or a button click.

3.2.5 High performance

In general, one can expect from this architecture that:

1. As a cloud admin, I want to create >1,000 VMs that has 2 CPU core, 8G memory, and 100 GB disk space, so that I can support these many workloads of the size.

¹A pin-ed VM, however, is not migratable

2. As a cloud admin, I want to have >200 MB/s write throughput and >500 MB/s read throughput seen by a virtual machine showing <50% CPU idleness.

4

Architectural Overview

Red Hat Hyperconverged Infrastructure for virtualization (RHHI-v) meets the requirements described in the previous chapter. RHHI-v consists of an integrated **Red Hat Virtualization** cluster and **Red Hat Gluster** cluster on a set of Lenovo servers and switches. Figure below shows an overview using three servers.

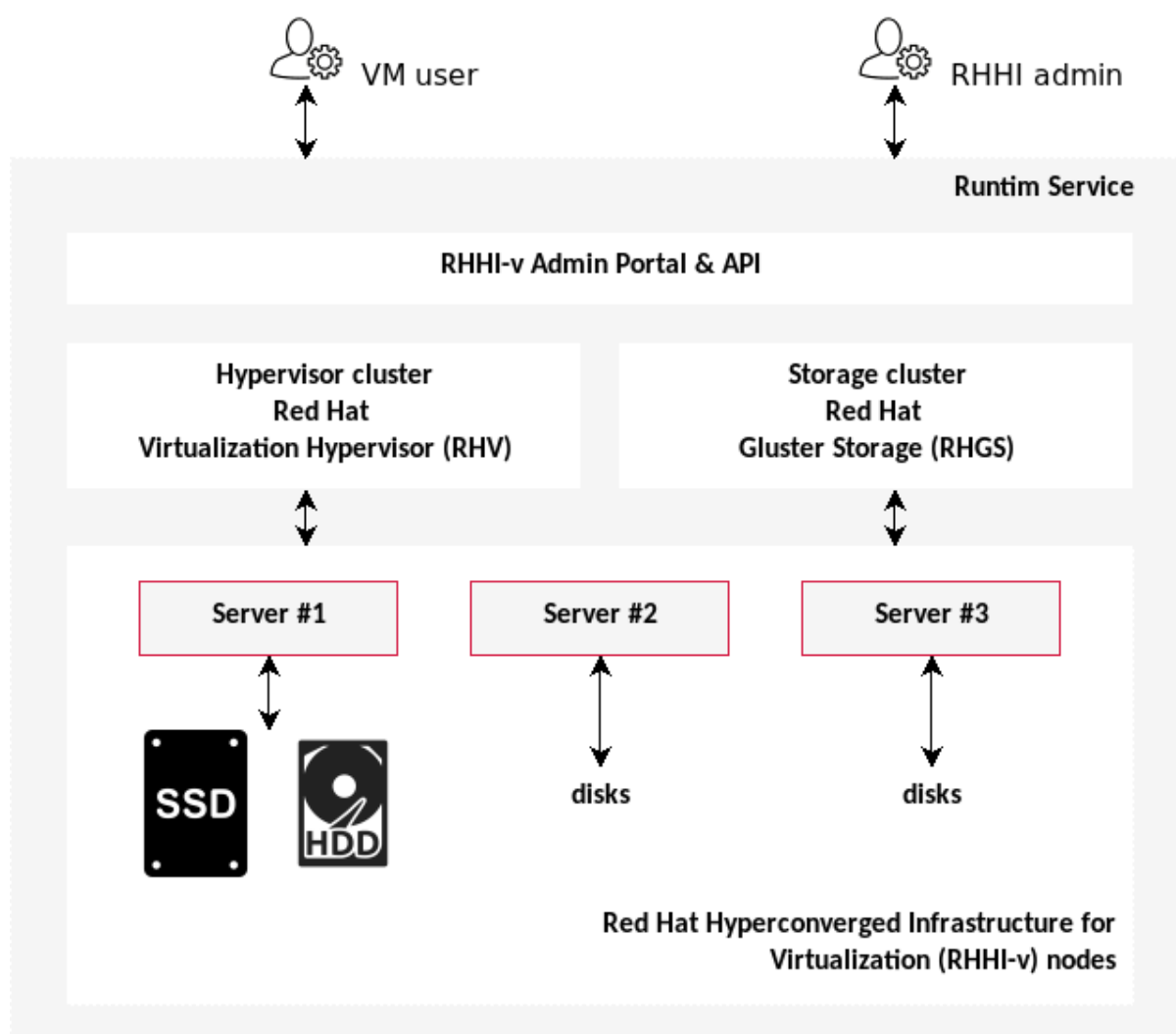


Figure 4.1: Lenovo Open Cloud RHHI-v Architecture of a 3-server Configuration

Red Hat Hyperconverged Infrastructure for Virtualization offers four configurations (see [BOM](#)):

1. 3-server configuration
2. 6-server configuration
3. 9-server configuration
4. 12-server configuration

Primary storage is a Gluster cluster that uses internal drives in servers to form distributed storage. By default data blocks are replicated to all servers in the cluster.

The server's compute resources are virtualized through a hypervisor, and workloads are deployed in virtual machines (VMs) that can then be located on any of the hosts in the cluster. The cluster's management is also in redundant VMs and will automatically use another host when there is a problem, thus providing an uninterrupted management capability. There is also a management CLI application on all cluster hosts that provides administration capabilities.

Initial setup can choose a minimal 3-server configuration. Once the software is deployed, more servers can be added to extend the cluster to provide additional resources for workloads.

5

Component model

The Red Hat Hyperconverged Infrastructure for virtualization (RHHI-v) consists of four components: physical networks, storage, hypervisor, and interfaces.

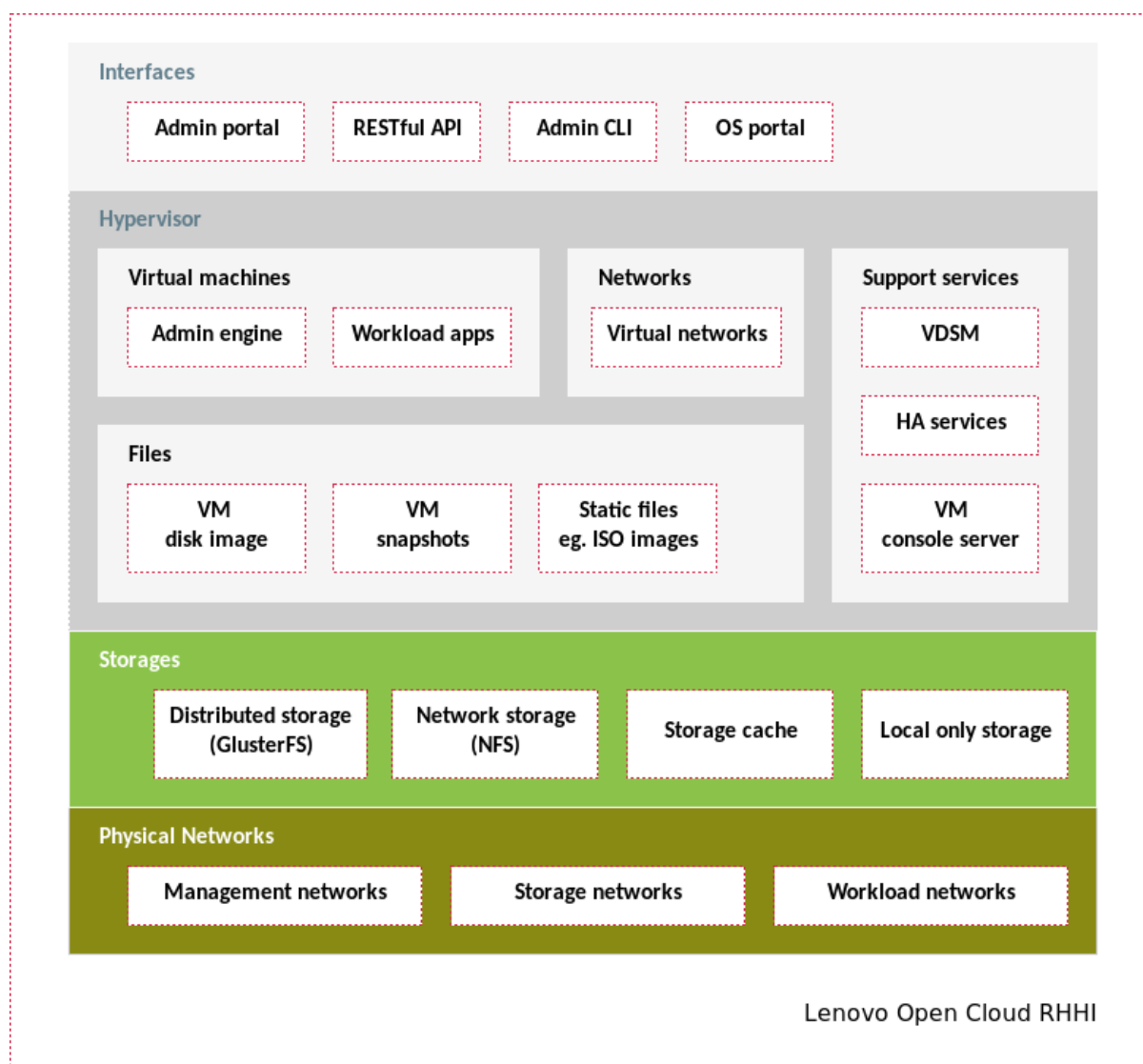


Figure 5.1: RHHI-v component model

Physical networks They represent the actual physical network connections between switches and servers. Based on their primary purpose, it can be further broken down into:

1. **Management networks:** These carry administrative functions of the cluster such as the admin portal, remote access to host, and API.
2. **Storage networks:** These are the data traffic to retrieve from and write to storage backend. In use cases in which storage performance is the key, this will be

the determining factor in design to support high data throughput.

3. **Workload networks:** Networks to support applications (aka. workloads) deployed on the infrastructure.

Storage RHHI-v provides four types of storage:

1. **Distributed storage:** This is a network storage. The key difference between this and other network storage such as NFS, is that it has built-in capability to distribute data read & write to multiple storage endpoints, and to create data replication. This can not only improve throughput performance, but is the key design factor to support fault tolerance and HA.

However, redundancy comes with penalty — the storage capacity will not always linearly grow by adding more disks to the cluster. See section **Gluster volume type** for details.

2. **Network storage:** Unlike distributed storage above, this storage does not offer data replication or data distribution. This fit for application who has already considered these functions in its design without depending on the storage itself. Benefit of such simplification is that storage capacity will grow linearly with the number of disks.
3. **Storage cache:** Along the storage data path, one would find many data caches — RAID cache, OS kernel cache, file system cache, database cache, and so on. We use this term to describe include all these caches that will hold data, even temporarily. They are usually working behind the scene and are beyond a user's access. However, their existence and their configurations have much effect on the infrastructure's performance and stability.
4. **Local only storage:** These are un-shared storages by either a host or an application.

Hypervisor is the virtualization layer of the infrastructure, in which hardware resources will be represented as virtual resources.

1. **Virtual machines:** These are virtualized compute resources. There are two types of virtual machine on the RHHI-v infrastructure — workloads, and management.

1. **Workloads:** Any software deployed on the infrastructure is a workload.
2. **Management:** VMs are for administrating the RHHI-v infrastructure's.
2. **Networks:** Physical networks are virtualized so a server' physical NICs and cable connections are shareable by virtual machines.

Virtual networks are not associated with any particular physical server. Thus regardless where the virtual machine is running, these networks can be available to it. In order for this to work, however, all servers in a cluster must have identical physical networks. See section [Network Configurations](#) for details.

3. **Files:** There are three important types of files in RHHI-v:
 1. **VM disk image:** A virtual machine consists of one or many disk images. They function more or less like the physical disk in a server. But since they are file based, there can be features VM disk can do that are impossible for a physical disk.
 2. **VM snapshot:** A key motivation of the RHHI-v is to virtualize a workload, thus allowing user to take snapshot — freeze the states of a virtual machine in time, and recover and resume from a snapshot.
 3. **Static files:** As an infrastructure, RHHI-v can be the source of truth of static files such as ISO images..
4. **Support services:** There are many support services for the infrastructure. Three key services are:
 1. **VDSM:** Daemon running on each host to support the admin application.
 2. **HA services:** responsible to keep the admin application running on a healthy host so that administrative capability of the infrastructure has high availability.
 3. **VM console server:** gives remote console access to the virtual machines. Before VM's network is created, this is the only mean to gain access to the VM for administrative tasks.

Interfaces These are entry points through which the infrastructure admin or user can access the infrastructure's function.

1. **Admin interfaces:** There are three interfaces for the infrastructure admin — a web portal, a RESTful API, and a CLI. They cover both the hypervisor and the gluster storage.
2. **OS portal:** RHEL 7 has a built-in portal, the [cockpit](#), that makes it easy to administer a RHEL-based host.
3. **OOB portal/CLI:** Interface to manage hardware when they are not yet provisioned. This includes admin portal and CLI of switches.

6

Operational model

6.1 Hardware

This section describes the hardware infrastructure aspects of the LOC RHHI-v reference architecture.

6.1.1 Lenovo SR650 Servers



Figure 6.1: Lenovo ThinkSystem SR650

The Lenovo ThinkSystem SR650 server is an enterprise class 2U two-socket versatile server that incorporates outstanding reliability, availability, and serviceability (RAS), security, and high efficiency for business-critical applications and cloud deployments. Unique Lenovo AnyBay technology provides the flexibility to mix-and-match SAS/SATA HDDs/SSDs and NVMe SSDs in the same drive bays. Four direct-connect NVMe ports on the motherboard provide ultra-fast read/writes with NVMe drives and reduce costs by eliminating PCIe switch adapters. Plus, storage can be tiered for greater application performance, to provide the most cost-effective solution.

Combined with the Intel® Xeon® Scalable processors product family, the Lenovo ThinkSystem SR650 server offers a high density combination of workloads and performance. Its flexible, pay-as-you-grow design and great expansion capabilities solidify dependability for any kind of virtualized workload, with minimal downtime. Additionally, it supports two 300W high-performance GPUs and ML2 NIC adapters with shared management.

The Lenovo ThinkSystem SR650 server provides internal storage density of up to 100 TB (with up to 26 x 2.5-inch drives) in a 2U form factor with its impressive array of workload-optimized storage configurations. The ThinkSystem SR650 offers easy management and saves floor space and power consumption for the most demanding storage virtualization use cases by consolidating the storage and server into one system. The Lenovo ThinkSystem SR650 server supports up to twenty-four 2.5-inch or fourteen 3.5-inch hot-swappable SAS/SATA HDDs or SSDs together with up to eight on-board NVMe PCIe ports that allow direct connections to the U.2 NVMe PCIe SSDs. The ThinkSystem SR650 server also supports up to two NVIDIA GRID cards for AI or media processing acceleration.

The SR650 server supports up to two processors, each with up to 28-core or 56 threads with hyper-threading enabled, up to 38.5 MB of last level cache (LLC), up to 2666 MHz memory speeds and up to 3 TB of memory capacity. The SR650 also support up to 6 x PCIe slots. Its on-board Ethernet solution provides 2/4 standard embedded Gigabit Ethernet ports and 2/4 optional embedded 10 Gigabit Ethernet ports without occupying PCIe slots. All these advanced features make the server ideal to run data and bandwidth intensive VNF workload and storage functions of NFVI platform.

For more information, see product guide [3].

6.1.2 Lenovo 930-x RAID controller

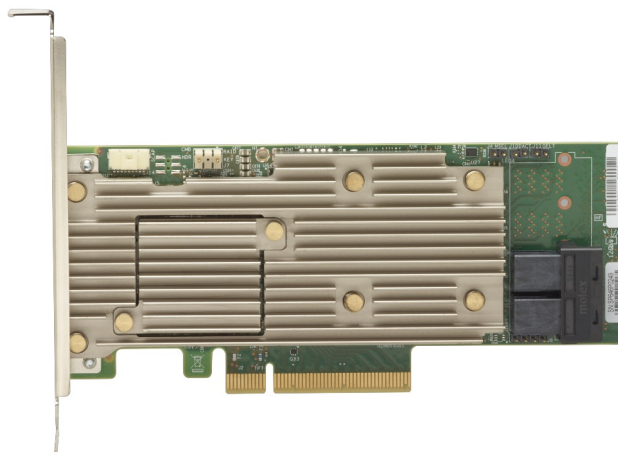


Figure 6.2: ThinkSystem RAID 930-8i RAID Controller

RAID controller is a critical component in Red Hat Hyperconverged Infrastructure for Virtualization architecture. Server disks must be RAID-ed to achieve both high availability and good performance. In addition, having the correct RAID firmware and feature enabled are crucial to make [Gluster](#) cluster work. In the following section we will show how to validate your server meeting this requirement.

For more information, see product guide[4].

6.1.3 Lenovo Network Switches

The following sections describe the Top-of-Rack (ToR) switches used in this reference architecture. The Networking Operating System (see [switch firmware BOM](#)) software features of these Lenovo switches deliver seamless, standards-based integration into upstream switches.

Two 10Gb switches and two 1Gb switches are recommended in this architecture. For high throughput workloads, we recommend Lenovo 25Gb switches such as NE2572, in which case server should also equip 25G NIC (see [BOM](#) for details) instead of 10G NICs. The 25G switch are compatible with 10G NICs, but not the other way around.

6.1.3.1 Lenovo RackSwitch G8272 (10Gb)



Figure 6.3: Lenovo RackSwitch G8272

The Lenovo RackSwitch G8272 uses 10Gb SFP+ and 40Gb QSFP+ Ethernet technology and is specifically designed for the data center. It is an enterprise class Layer 2 and Layer 3 full featured switch that delivers line-rate, high-bandwidth, low latency switching, filtering, and traffic queuing without delaying data. Large data center-grade buffers help keep traffic moving, while the hot-swap redundant power supplies and fans (along with numerous high-availability features) help provide high availability for business sensitive traffic.

The RackSwitch G8272 (as shown in fig. 6.3) is ideal for latency sensitive applications, such as high-performance computing clusters, financial applications and NFV deployments. In addition to 10 Gb Ethernet (GbE) and 40 GbE connections, the G8272 can use 1 GbE connections (need the Photoelectric conversion module).

For more information, see product guide[5].

6.1.3.2 Lenovo RackSwitch G8052 (1Gb)



Figure 6.4: Lenovo RackSwitch G8052

The Lenovo RackSwitch™ G8052 (as shown in fig. 6.4) is a top-of-rack data center switch that delivers unmatched line-rate Layer 2/3 performance at an attractive price.

It has 48x 10/100/1000BASE-T RJ-45 ports and four 10 Gigabit Ethernet SFP+ ports (it also supports 1 GbE SFP transceivers), and includes hot-swap redundant power supplies and fans as standard, which minimizes your configuration requirements. Unlike most rack equipment that cools from side-to-side, the G8052 has rear-to-front or front-to-rear airflow that matches server airflow.

For more information, see product guide[6].

6.1.3.3 Lenovo ThinkSystem NE2572 (25Gb)



Figure 6.5: Lenovo RackSwitch NE2572

The NE2572 is optimized for enterprise data centers with features, such as:

- 🔊 Software Defined Data Center interfaces ensure smooth interoperability with popular enterprise management applications.

- Open Architecture: NE2572 supports automation and control while
- Lenovo Network Orchestrator automates vlan changes to improve. integrating with industry-standard application and APIs, such as REST, OpenStack, OpenContrail, and Ansible.
- Resiliency: NE2572 offers outstanding fault tolerance with one of the quickest high-availability failovers in the industry so running applications are less likely to require a restart.

For more information, see product guide[7].

6.2 Software

Based on Red Hat Hyperconverged Infrastructure for Virtualization 1.5[2]. RHHI-v can be viewed as being comprised of two primary components: Red Hat Virtualization (RHV)[8], and Red Hat Gluster (RHGS)[9].

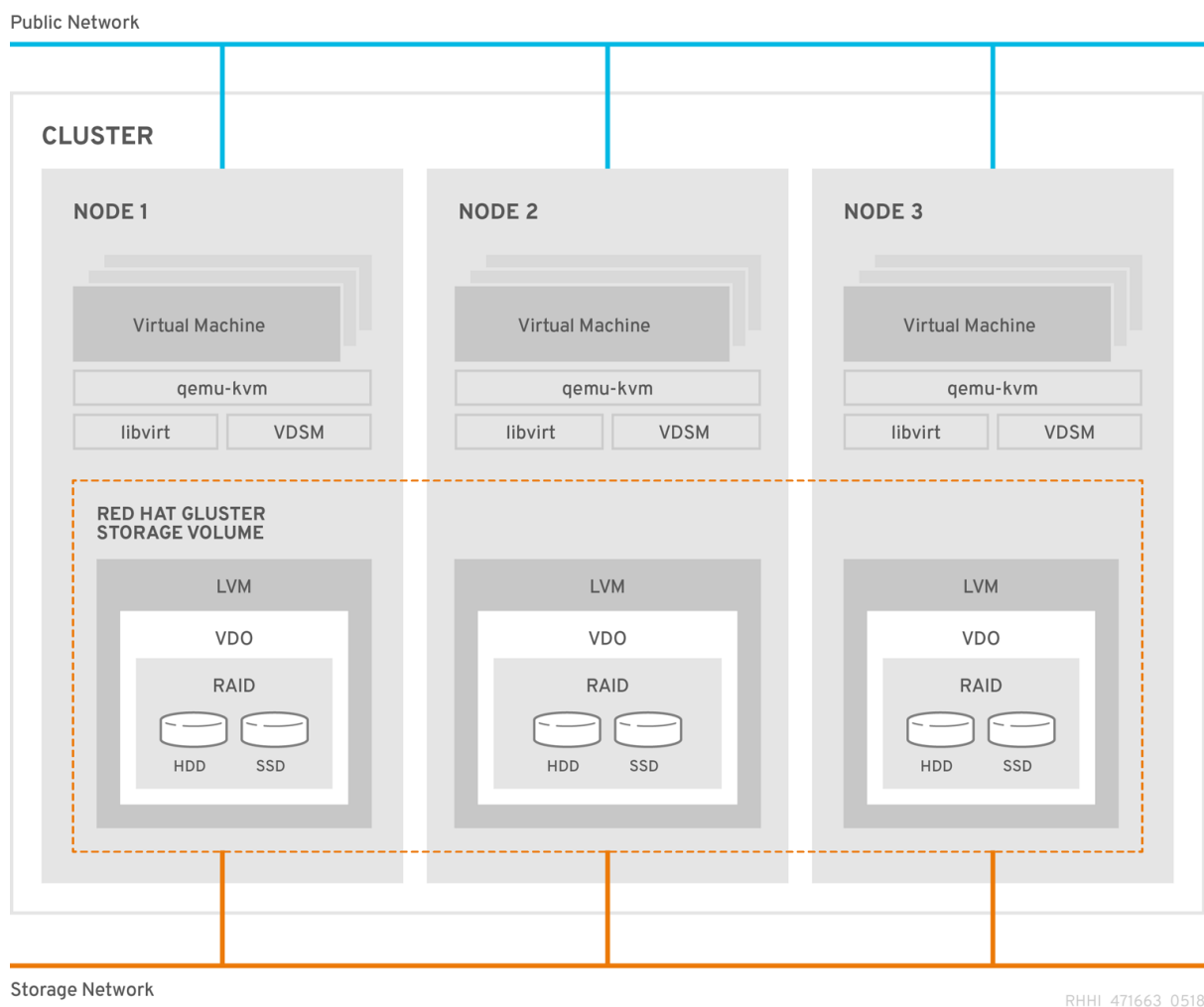


Figure 6.6: Red Hat Hyperconverged Infrastructure for Virtualization architecture (source: [10])

6.2.1 Red Hat Virtualization (RHV)

Red Hat Virtualization is an enterprise-grade virtualization platform built on Red Hat Enterprise Linux. Virtualization allows users to easily provision new virtual servers and workstations, and provides more efficient use of physical server resources. With Red Hat Virtualization, you can manage your entire virtual infrastructure - including hosts, virtual machines, networks, storage, and users - from

a centralized graphical user interface or RESTful API. (source: [11])

RHV has three key components:

1. **Red Hat Virtualization Manager:** A service that provides a graphical user interface and a RESTful API to manage the resources in the environment.
2. **Hosts:** Red Hat Enterprise Linux hosts (RHEL-based hypervisors) and Red Hat Virtualization Hosts (image-based hypervisors) are the two supported types of host. Hosts use Kernel-based Virtual Machine (KVM) technology and provide resources used to run virtual machines.
3. **Shared Storage:** A storage service is used to store the data associated with virtual machines.

There are two deployment modes: standalone or self-hosted engine. In this reference architecture, we are using the [self-hosted engine](#) method (fig. 6.7):

The Red Hat Virtualization Manager runs as a virtual machine on self-hosted engine nodes (specialized hosts) in the same environment it manages. A self-hosted engine environment requires one less physical server, but requires more administrative overhead to deploy and manage. The Manager is highly available without external HA management. (source: [12])

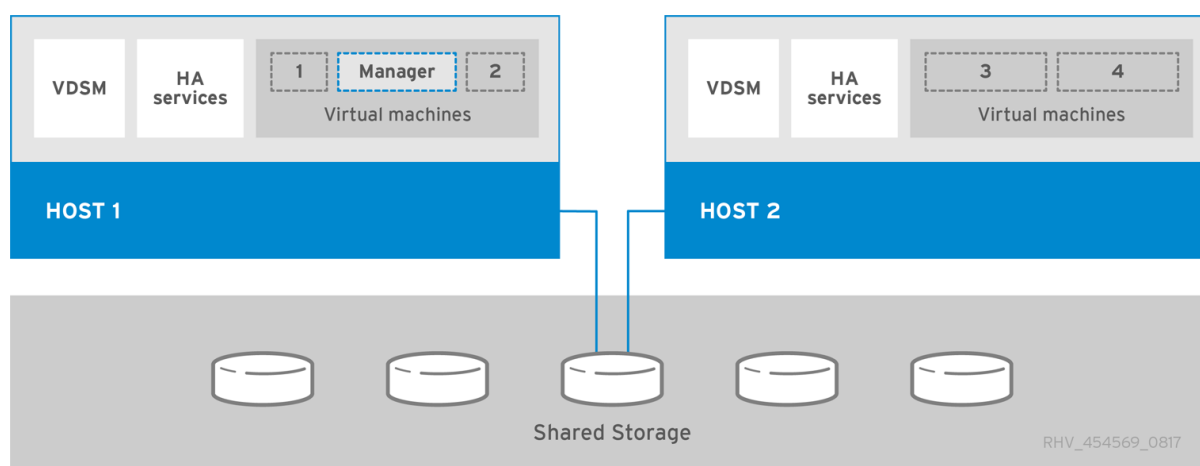


Figure 6.7: Red Hat Virtualization Self-Hosted Engine Architecture (source: [13])

The [hosted-engine](#) VM can be started on any server. If the server on which it is currently residing goes down, eg. for maintenance or due to hardware failure, the VM

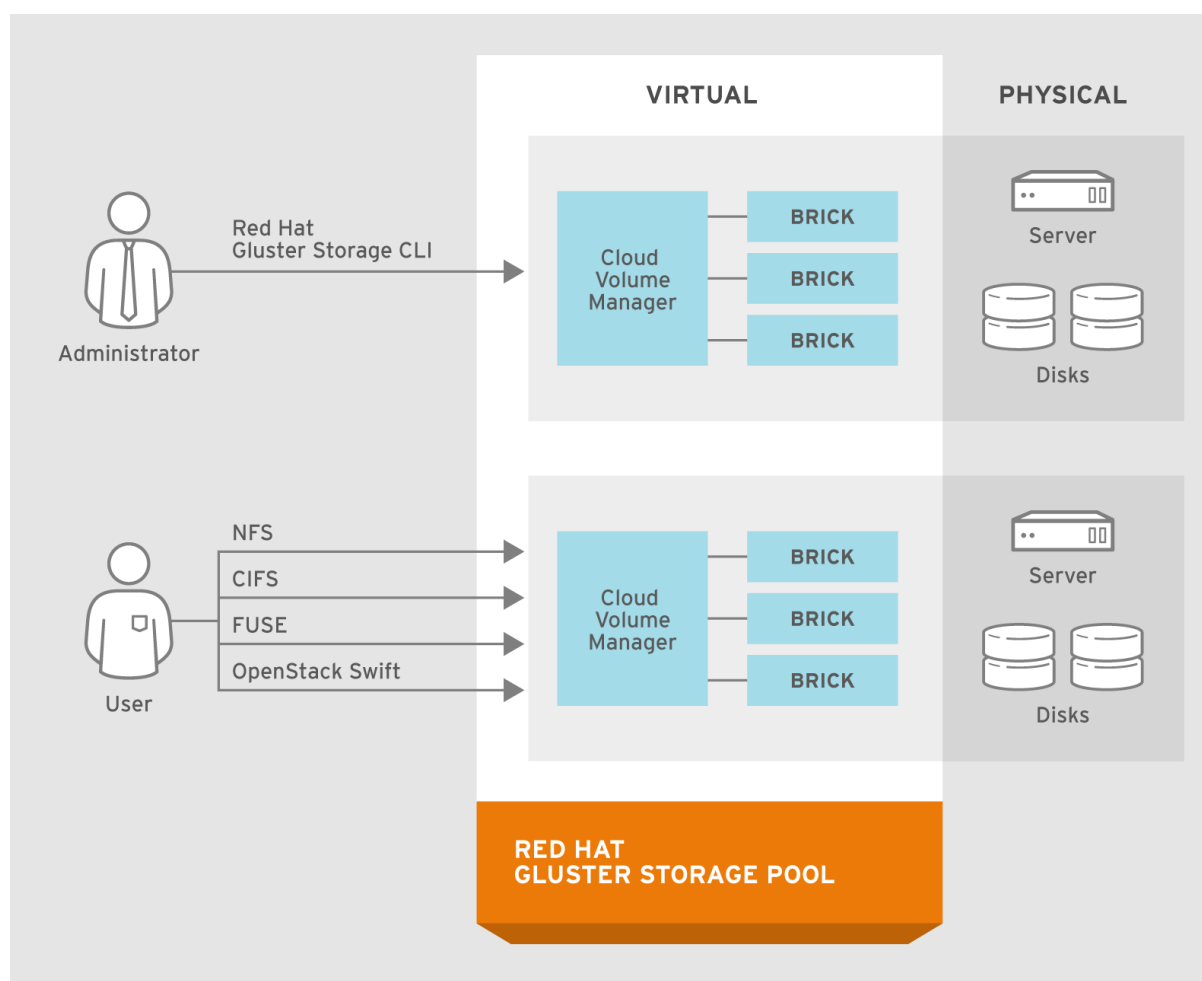
will automatically migrate to a different server¹, thus providing a seamless user experience.

6.2.2 Red Hat Gluster Storage (RHGS)

Red Hat Gluster Storage is a software-defined, scale-out storage that provides flexible and affordable unstructured data storage for the enterprise. Red Hat Gluster Storage 3.4 provides new opportunities to unify data storage and infrastructure, increase performance, and improve availability and manageability in order to meet a broader set of an organization's storage challenges and requirements.

GlusterFS, a key building block of Red Hat Gluster Storage, is based on a stackable user space design and can deliver exceptional performance for diverse workloads. GlusterFS aggregates various storage servers over network interconnects into one large parallel network file system. (source: [14])

¹RHV can run on minimal two-servers, but its deployment requires at least three servers.



#153460_GLUSTER_1.0_334434_0415

Figure 6.8: Red Hat Gluster Storage architecture (source: [15])

The key feature of RHGS is that unlike other distributed storage methods in which tracking the location of data is based on a metadata server, RHGS locates files mathematically using an elastic hashing algorithm. This not only removes single-point-of-failure of metadata server, but also makes the system linearly scalable.

In this reference architecture, [Gluster](#) is the primary storage backend. Depending on the workload requirement, Red Hat Hyperconverged Infrastructure for Virtualization can also provide other storage solutions such as NFS for OpenStack Swift object store.

7

Deployment considerations

7.1 Server Configurations

7.1.1 Disk configurations

Disk configuration is important to achieve high performance. SR650 server is flexible in this regards. It supports various types of disks. In general SSD will yield better performance than HDD. In this example, we recommend the following in each SR650 server:

Table 7.1: Red Hat Hyperconverged Infrastructure for Virtualization Server disk configurations

| Type | Position | Number | Size | RAID | Purpose |
|----------|-----------------|--------|-------|-------|------------------|
| SATA SSD | Front backplane | 2 | 480GB | RAID1 | Operating system |
| SATA SSD | Front backplane | 2 | 960GB | RAID1 | LVM cache |
| SAS HDD | Front backplane | 20 | 2.4TB | RAID6 | GlusterFS |

By default gluster volumes will be in **replicated** mode, which renders total storage capacity in a 3-server HCI Gluster: $20 * 2.4\text{TB} = 48\text{TB}$. LVM cache can be turned on and off in deployment (see **deployment template**). To determine whether to enable it, see **LVM cache** for details.

In addition, Red Hat Hyperconverged Infrastructure for Virtualization supports alter-

native network storage such as NFS, for example, used for SWIFT object store. Disk configuration will thus require two additional disks for the purpose:

Table 7.2: Red Hat Hyperconverged Infrastructure for Virtualization Server Disk Configurations w/ support of file-based storage

| Type | Position | Number | Size | RAID | Purpose |
|----------|-----------------|--------|-------|-------|------------------|
| SATA SSD | Front backplane | 2 | 480GB | RAID1 | Operating system |
| SATA SSD | Front backplane | 2 | 960GB | RAID1 | LVM cache |
| SAS HDD | Front backplane | 18 | 2.4TB | RAID6 | GlusterFS |
| SAS HDD | Front backplane | 2 | 2.4TB | RAID1 | Affinitized VMs |

7.1.2 RAID configurations

RAID affects not only performance, but provides data redundancy. Table below lists the key attributes of Red Hat Hyperconverged Infrastructure for Virtualization:

Table 7.3: Red Hat Hyperconverged Infrastructure for Virtualization Server RAID Configurations

| Config | Value |
|---------------------------|---------------------|
| Stripe size | 256 KB |
| Read policy | Adaptive Read Ahead |
| Write policy | Always Write Back |
| I/O policy | Direct I/O |
| Access policy | Read Write |
| Disk cache policy | Unchanged |
| Background initialization | Disabled |

Regarding the [write policy](#), in some older RAID controllers, there was a need to attach an add-on cache card and had option of having an on-board battery in order to use the [Write With BBU](#) setting for the write policy. On the Lenovo's 930-x RAID controller, however, this has been simplified:

MegaRAID flash cache protection uses NAND flash memory, which is powered by a [CacheVault Power Module](#) supercapacitor, to protect data that is stored in the controller cache. This module eliminates the need for a lithium-ion battery, which is commonly used to protect DRAM cache memory on PCI RAID controllers. To avoid the possibility of data loss or corruption during a power or server failure, flash cache protection technology transfers the contents of the DRAM cache to NAND flash using power from the offload power module. After the power is restored to the RAID controller, the content of the NAND flash is transferred back to the DRAM, which is flushed to disk. (source: [\[4\]](#))

Therefore, setting the write policy to [Always Write Back](#) is recommended.

7.1.3 Firmware

All RHHI-v servers should meet the following firmware requirements:

Table 7.4: Red Hat Hyperconverged Infrastructure for Virtualization Server
Firmware

| Firmware | Version | Build |
|----------|---------|---------|
| IMM | 4.40 | TCOO36C |
| UEFI | 2.81 | TCE138G |

7.2 Network Configurations

The most critical piece of a successful Red Hat Hyperconverged Infrastructure for Virtualization deployment is its networking. In this section we will go over the details of a networking design that follows the basic RHHI-v's requirement, but also taking advantage of the Lenovo hardware we have introduced so that you gain in both robustness and flexibility.

7.2.1 Design Conventions

We recognize there are endless possibility to design a network, and each data center is different. In this architecture we have followed these conventions:

1. Data center switches follow a [spine-leaf](#) topology.
2. Switch to switch connections are always paired (inter-switch-links, aka. ISL) for redundancy.
3. Except BMC connection, server to switch connections are always paired (fig. [7.1](#)). In addition, each pair should connect to separate NICs on the server side, and to separate switches.

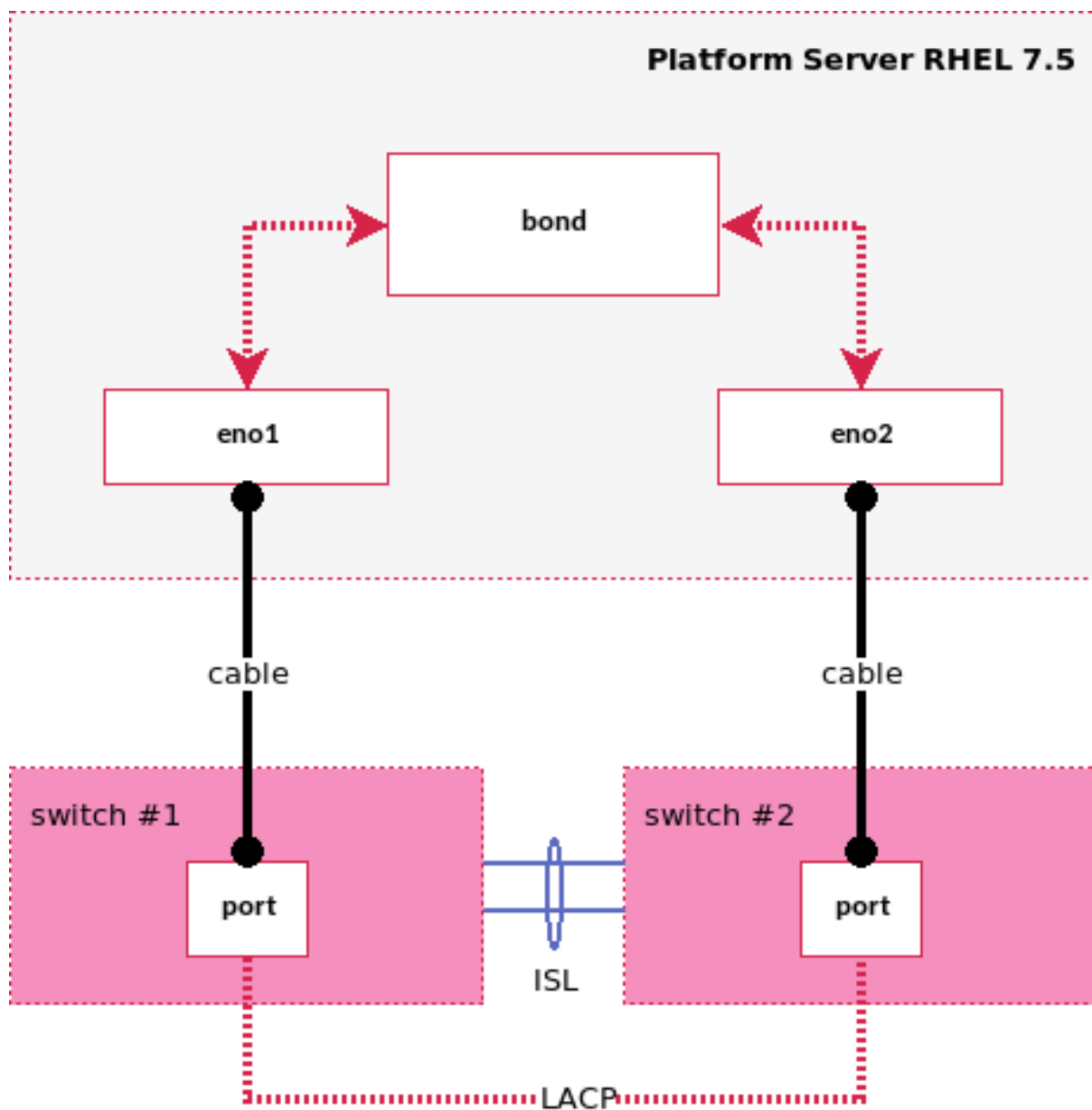


Figure 7.1: Server-Switch network convention

This requires matching configuration on the switch side, and on the server using **active-active** [bonding\[16\]](#) or **active-backup**. Both we will demonstrate how to create in the following sections.

4. Separate management traffic from data traffic whenever possible.

5. Improve network isolation by assigning private IP space to internal only traffic whenever possible.

7.2.2 Network Overview

Red Hat Hyperconverged Infrastructure for Virtualization networks can be categorized into three groups: **management network**, **storage network**, and **workloads network**. But this is a much simplified view. In practice, there is also traffic to manage baremetal servers, to access BMC, and workloads may have their specific pattern of consuming and congesting a shared NIC.

Considering each environment is rather different, we recommend these steps to understand design in this reference architecture. This same approach can then be applied to a specific workload in order to build a well-planned network.

1. **Define logical networks by their function:** this defines the purpose of this network, thus clarifying its characteristics such as load, latency, space, etc..
2. **Assign VLAN to logical networks:** each logical network is assigned a unique VLAN. We recommend to establish a consistent VLAN scheme, like a naming convention, that all VLANs in your RHHI-v environment should follow.
3. **Map network/VLANs to server's network interface:** this defines how server side interfaces will be configured to support these networks.
4. **Map server's networking interface to switch:** this defines the connection between server and switch, thus the switch side configurations including port mode, vLAG, native vlan, and untagged vlans.
5. **Cabling schema:** this defines switch port assignments, which server port is to be connected to which switch and which port.
6. **Configure switches:** this shows how to apply switch side configurations to Lenovo G8052 and Lenovo G8272 switches.
7. **Configure server network interfaces:** this shows configuration files used to create network interfaces on the server with RHEL 7.5.
8. **Map workload network to VLAN and RHV network:** this defines networks that are needed to support workloads running on top of LOC.

7.2.3 Topology

Overall topology of Red Hat Hyperconverged Infrastructure for Virtualization networking is shown in fig. 7.2.

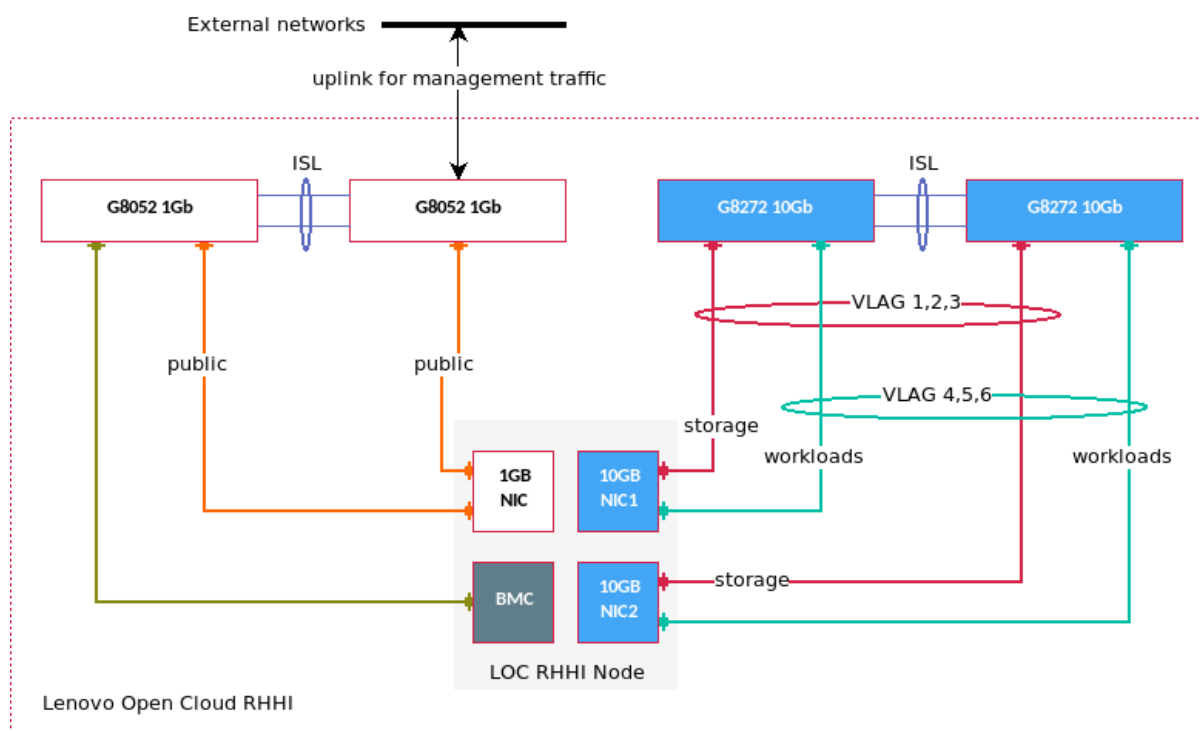


Figure 7.2: Lenovo Open Cloud RHHI-v network topology

1. One of LOC's 1Gb switch is connected to uplink to allow management access from external network.
2. LOC switches are paired via inter-switch-link (ISL).
3. Except BMC connection, all server to switch links are in pairs on switch side.
4. 10G connections are teamed in vLAG; 1G connections are not teamed.¹

¹If there is only one 10G switch available, ports can be teamed using [port-channel](#) instead of [vLAG](#).

7.2.4 Logical networks

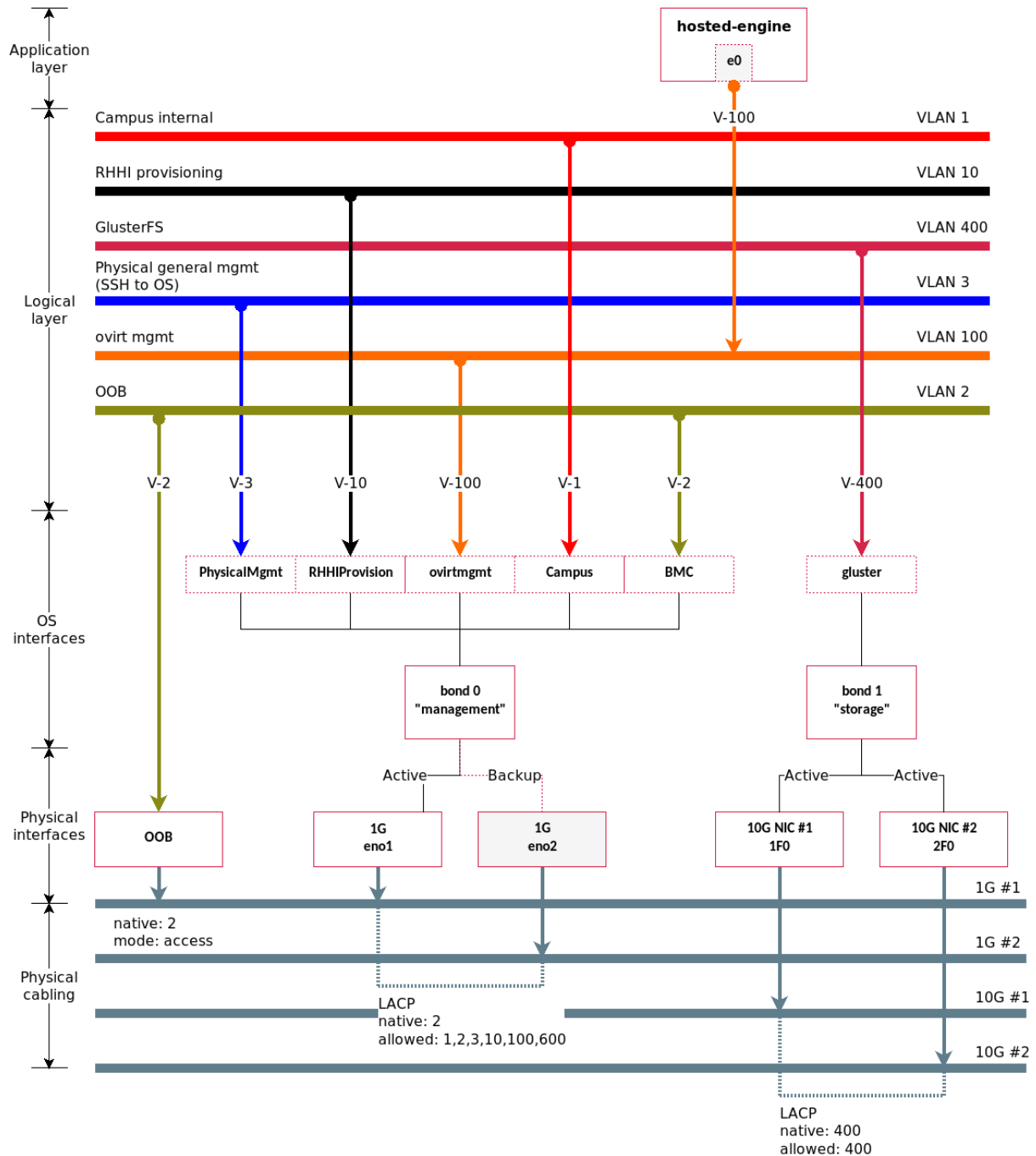


Figure 7.3: Lenovo Open Cloud RHHI-v logical networks

Red Hat Hyperconverged Infrastructure for Virtualization networks are designed to offer performance and high availability. These are extensions of the minimal RHHI-v networking requirement shown in fig. 6.6. It is possible to merge these to conform with the minimal setup, or to reuse existing ones for its purpose (see [Implementation Questionnaire](#) for details). By highlighting their functions individually, they serve to distinguish operation and data that are actually included in a RHHI-v setup, and such distinction assists admin to understand their impact and implication, and help to make decisions as to what is the best way to manage and operate these functional aspects.

Campus internal aka. public network. A common case is a web application that is accessible by general audience remotely within a data center or even from the Internet.

In a typical RHHI-v deployment, [ovirt](#) management network serves the admin UI. Therefore it seems redundant to have another network for such purpose. However, following our design philosophy of making the deployment as much self-contained, we are to treat [ovirt](#) as an internal network, while having this [Campus](#) network representing public access. Therefore, RHHI-v admin gains more control when accessing the RHHI-v admin port and ovirt API should be allowed by general public. An implementation is through a proxy (or jump host) which we will demonstrate in section [xxx].

BMC aka. [Out-of-Band](#) (OOB) management network. It connects to a dedicated management port on physical servers. In practice this is often replaced by [Campus](#) so admin can remotely dial into a BMC controller and activate commands such as rebooting servers to UEFI. However, this is risking security for convenience. We highly recommend using a dedicated [BMC](#) network for OOB so they can be independently managed and secured.

ovirt management is the network linking RHHI-v management console to RHHI-v clusters. This is the default network serving RHHI-v admin portal and ovirt API.

As mentioned above, a standard deployment will use [ovirt](#) network for public access. However, we recommend treating it as an internal while establishing a [Campus](#) network for public consumption. To expose admin port and API for tasks such as automation and integration w/ other applications who run on

[Campus](#), one can utilize proxy, jump host, or routing to achieve the effect while maintaining control over this network.

GlusterFS is a private network for [Gluster](#) data storage. The Gluster cluster is highly sensitive to network interruption.

In this design we dedicate two 10G ports on each server for this network. Further, we are to take measures in server configs and switch configs (shown in section [Map Server NIC to Switch](#)) to improve both data throughput and redundancy so to satisfy high availability.

Physical server management is to support [In-Band](#) managerial tasks, eg. [ssh](#) to a server. This is to recognize the need that remote administration of a physical server is often limited to operators who has that privilege. In practice, setting this to the same as [Campus](#) is not uncommon.

RHHI provisioning is to support data traffic of installing the OS on a physical server. This is highly depending on the technology of server provisioning you are choosing.

The most common method is PXE. PXE is a broadcast protocol, thus isolating it to this network avoids cross-talk.

However, Lenovo's xClarity server management software can provision a baremetal by accessing BMC and mount remote disk images directly, thus can work without this network.

7.2.5 VLANs

VLANs are assigned to Red Hat Hyperconverged Infrastructure for Virtualization logical networks, which later will determine [switch port configurations](#). These values are examples we will use for the rest of this document to illustrate networking considerations. Before deploying a Red Hat Hyperconverged Infrastructure for Virtualization instance, user should use the [Implementation Questionnaire](#) to map these VLANs to their network environment.

Table 7.5: Example Red Hat Hyperconverged Infrastructure for Virtualization logical network VLANs

| Logical Network | VLAN |
|----------------------------|------|
| Campus internal | 1 |
| BMC | 2 |
| Physical server management | 3 |
| RHHI provisioning | 10 |
| OVIRT management | 100 |
| GlusterFS | 400 |

7.2.6 Subnets

Last, when reserving IP space for each network, we use 192.168.x.x IP addresses defined in RFC 1918[17] to keep Red Hat Hyperconverged Infrastructure for Virtualization network self-contained. As suggested earlier, operator can use proxy, routing, and jump host to grant access to broader audience if needs arise.

We will use the following naming convention for LOC subnets:

```
1 192.168.[vlan index].x
```

If vlan index is greater than 255, we remove the last digit of the vlan number, eg. 400 to 40, thus making its subnet pattern 192.168.40.x.

Table 7.6: Red Hat Hyperconverged Infrastructure for Virtualization Logical Network Subnets

| Logical Network | Subnet | Addresses | Mask | Gateway |
|-----------------|-------------|-----------|------|-------------|
| Campus internal | 192.168.1.x | 254 | /24 | 192.168.1.1 |
| BMC | 192.168.2.x | 254 | /24 | 192.168.2.1 |

| Logical Network | Subnet | Addresses | Mask | Gateway |
|----------------------------|---------------|-----------|------|---------------|
| Physical server management | 192.168.3.x | 254 | /24 | 192.168.3.1 |
| RHHI provisioning | 192.168.10.x | 3/6/9 | /24 | 192.168.10.1 |
| OVIRT management | 192.168.100.x | 3/6/9 | /29 | 192.168.100.1 |
| GlusterFS | 192.168.40.x | 3/6/9 | /29 | 192.168.40.1 |

7.2.7 Map VLAN to Server NIC

Mapping VLAN to server NIC makes VLANs available to RHV.

7.2.7.1 Server NIC connections

Following design convention, pair cables connect two NIC ports on two different switches. Later on [switch configurations](#), 1G connections will have identical configurations but do not use LACP; 10G connections are teamed in [vLAG](#).

Table 7.7: Red Hat Hyperconverged Infrastructure for Virtualization VLAN to Server's NIC Mapping

| Logical Network | VLAN | BMC | 2 x 1G | 2 x 10G | Switch Port LACP |
|----------------------------|------|-----|--------|---------|------------------|
| Campus internal | 1 | | x | | No |
| BMC ² | 2 | x | | | No |
| Physical server management | 3 | | x | | No |
| RHHI provisioning | 10 | | x | | No |
| OVIRT management | 100 | | x | | No |
| GlusterFS | 400 | | | x | Yes |

7.2.7.2 Server NIC bonding interfaces

1G connections form a [active-backup](#) bonding interface ([bond 0](#) for management), and 10G connections form [active-active](#) bonding interfaces ([bond 1](#) for gluster storage).

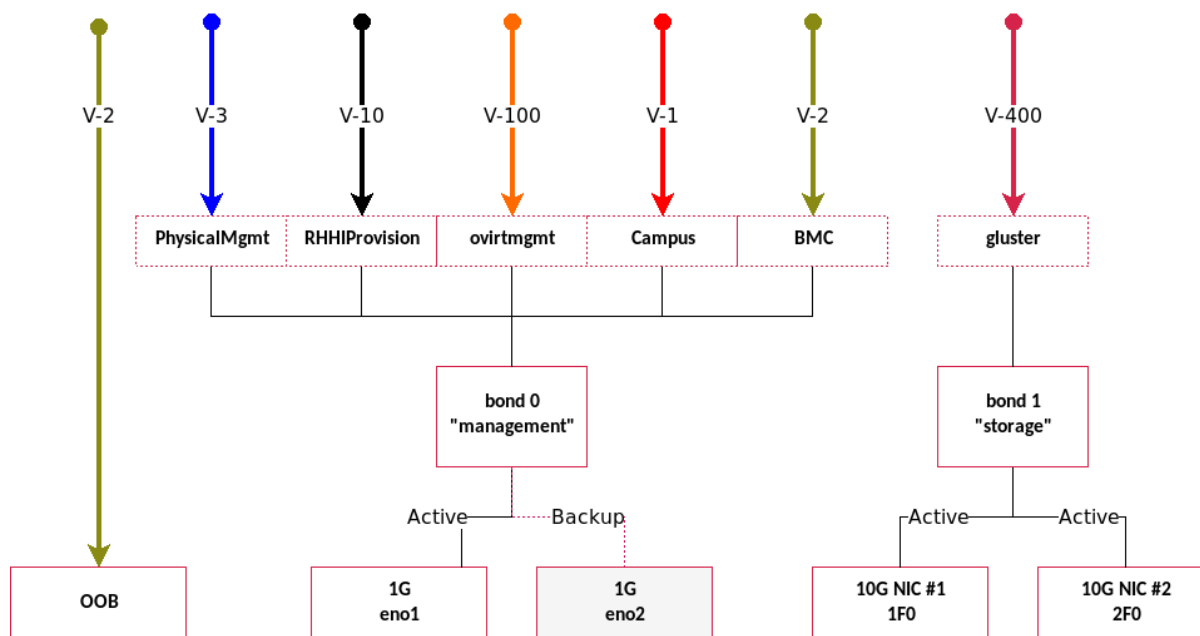


Figure 7.4: Red Hat Hyperconverged Infrastructure for Virtualization Server Map VLAN to NIC

Table 7.8: Red Hat Hyperconverged Infrastructure for Virtualization VLAN to Server Bonding Interfaces Mapping

| Logical Network | VLAN | Bonding Interfaces |
|----------------------------|------|---------------------|
| Campus internal | 1 | bond 0 (management) |
| BMC | 2 | bond 0 (management) |
| Physical server management | 3 | bond 0 (management) |
| RHHI provisioning | 10 | bond 0 (management) |

²BMC network is also exposed as a RHV network [BMC](#) which is through 1G connections so VMs can have access to BMC controllers on server. However, here BMC network is shown connected only to the BMC port on server.

| Logical Network | VLAN | Bonding Interfaces |
|------------------|------|---------------------|
| OVIRT management | 100 | bond 0 (management) |
| GlusterFS | 400 | bond 1 (storage) |

7.2.7.3 Server NIC bridge interfaces

Virtual bridge interfaces are created on top of a bonding interface to define a one-to-one mapping between VLAN and a RHV network.

Table 7.9: Red Hat Hyperconverged Infrastructure for Virtualization VLAN to Server Bonding Interfaces Mapping

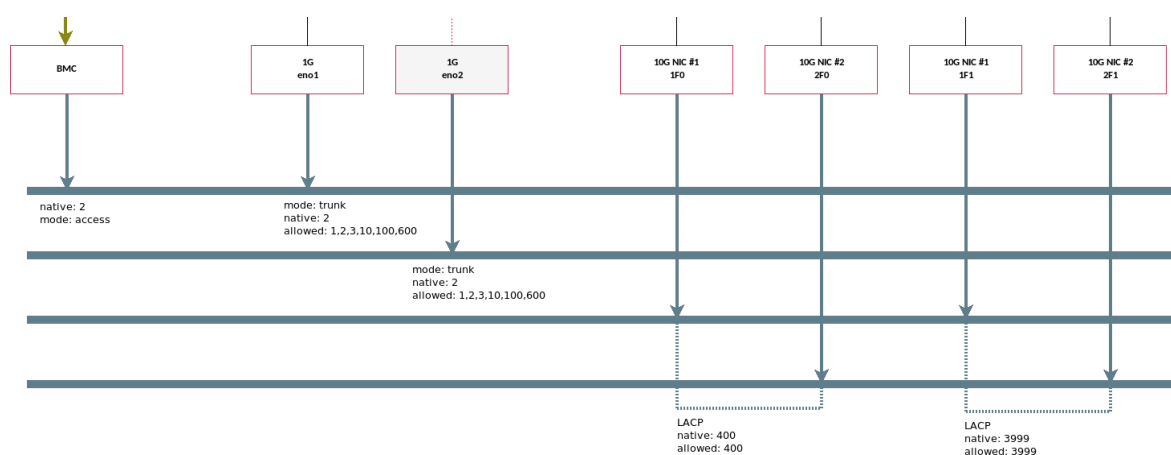
| Logical Network | VLAN | Bonding Interface | Bridge Interface (RHV network) |
|----------------------------|------|-------------------|--------------------------------|
| Campus internal | 1 | bond 0 | CampusInternal |
| BMC | 2 | bond 0 | BMC |
| Physical server management | 3 | bond 0 | PhysicalMgmt |
| RHHI provisioning | 10 | bond 0 | RHHIProvision |
| OVIRT management | 100 | bond 0 | ovirtmgmt |
| GlusterFS | 400 | bond 1 | gluster |

7.2.8 Map Server NIC to Switch

Red Hat Hyperconverged Infrastructure for Virtualization uses two G8052 (1Gb) switches and two G8272 (10Gb) switches in this architecture so that each logical network is running on a cable pair that is connected to two different switches for redundancy.

Table 7.10: Red Hat Hyperconverged Infrastructure for Virtualization Server to Switch Connections

| Server Port | Switch | Port Redundancy | Mode | Native VLAN | Allowed VLANs |
|-------------|--------|-----------------|--------|-------------------|---------------|
| BMC | 1Gb | none | access | – | 2 |
| eno1, eno2 | 1Gb | pair w/o LACP | trunk | 2 | 1,2,3,10,100 |
| 1F0, 2F0 | 10Gb | pair w/ LACP | trunk | 3999 | 400,3999 |
| 1F1, 2F1 | 10Gb | pair w/ LACP | trunk | 3999 ³ | 3999 |

**Figure 7.5:** Lenovo Open Cloud Platform Server to Switch

7.2.9 Define Cable Schema

Both Lenovo G8052 and G8272 switches have 52 ports, in which 48 ports are used for server connections, and the other four ports are for inter-switch connections and switch uplinks. Here we present an example cable schema following the network

³VLAN 3999 is a special vlan that represents a *not-in-use* network in this document. The two 10Gb ports of 1F1 & 2F1 will be used to support LOC RHHI workloads such as a Red Hat Openstack. Therefore, its switch configuration will be determined by the requirement of workload. For examples of networking setup of LOC and Red Hat OSP 10 & 13 workloads, refer to the “Lenovo Open Cloud Reference Architecture”[18].

designs described in previous section. We will also use this schema to demonstrate switch port configurations.

Consider that Red Hat Hyperconverged Infrastructure for Virtualization can support up to 12 servers, we have reserved port 1-16 on one G8052 switch for BMC connections.

Table 7.11: Red Hat Hyperconverged Infrastructure for Virtualization Server-Switch Cable Schema of a 3-server Configuration

| Port | G8052 #1 | G8052 #2 | G8272 #1 | G8272 #2 |
|------|---------------|---------------|--------------|--------------|
| 1 | server 1 BMC | | server 1 1F0 | server 1 2F0 |
| 2 | server 2 BMC | | server 2 1F0 | server 2 2F0 |
| 3 | server 3 BMC | | server 3 1F0 | server 3 2F0 |
| 4 | | | server 1 1F1 | server 1 2F1 |
| 5 | | | server 2 1F1 | server 2 2F1 |
| 6 | | | server 3 1F1 | server 3 2F1 |
| 17 | server 1 eno1 | server 1 eno2 | | |
| 18 | server 2 eno1 | server 2 eno2 | | |
| 19 | server 3 eno1 | server 3 eno2 | | |

7.2.10 Configure Switch Ports

Lenovo networking switches are highly configurable. We have developed tools for switch administrators including CLI and Ansible module. See [Switch Port Configuration Methods](#) for more information.

Following the cable schema in previous section, switch port configs are now ready to be applied:

Table 7.12: Example of G8052 #1 Port Configurations

| Port | LACP | Mode | Native VLAN | Tagged VLANs |
|------|------|--------|-------------|--------------|
| 1 | No | Access | – | 2 |
| 2 | No | Access | – | 2 |
| 3 | No | Access | – | 2 |
| 17 | Yes | Trunk | 2 | 1,2,3,10,100 |
| 18 | Yes | Trunk | 2 | 1,2,3,10,100 |
| 19 | Yes | Trunk | 2 | 1,2,3,10,100 |

Table 7.13: Example of G8052 #2 Port Configurations

| Port | LACP | Mode | Native VLAN | Tagged VLANs |
|------|------|-------|-------------|--------------|
| 17 | Yes | Trunk | 2 | 1,2,3,10,100 |
| 18 | Yes | Trunk | 2 | 1,2,3,10,100 |
| 19 | Yes | Trunk | 2 | 1,2,3,10,100 |

Table 7.14: Example of G8272 #1 & #2 Port Configurations

| Port | LACP | Mode | Native VLAN | Tagged VLANs |
|------|------|-------|--------------------|--------------------|
| 1 | Yes | Trunk | 3999 | 400,3999 |
| 2 | Yes | Trunk | 3999 | 400,3999 |
| 3 | Yes | Trunk | 3999 | 400,3999 |
| 4 | Yes | Trunk | workload dependent | workload dependent |
| 5 | Yes | Trunk | workload dependent | workload dependent |
| 6 | Yes | Trunk | workload dependent | workload dependent |

7.3 Storage

7.3.1 Gluster

The primary storage in Red Hat Hyperconverged Infrastructure for Virtualization is the gluster cluster. Gluster is a user space storage solution. GlusterFS, though called a “file system”, is not a file system in a strict sense. Instead, it is an abstraction of a common interface to backing file stores which can be many kinds — [xfs](#), [ext4](#), [nfs](#), [samba](#), just to name a few. In LOC RHHI-v, all underline file systems is a XFS.

All Red Hat Hyperconverged Infrastructure for Virtualization servers in the gluster trusted pool has identical configuration. Therefore, we will use a 3-server RHHI-v configuration for illustration.

7.3.1.1 Bricks

The glusterFS basic unit of storage, represented by an export directory on a server in the trusted storage pool. A brick is expressed by combining a server with an export directory in the following format: [SERVER:EXPORT](#)

There are three bricks in each server, and each brick is corresponding to a logical volume created on the host.

1. [engine](#) brick: space for hosted-engine VM storage only.
2. [vmstore](#) brick: space for all RHV virtual machine disk images.
3. [data](#) brick: space for static file such as an operating system ISO image that is then used to create virtual machines.

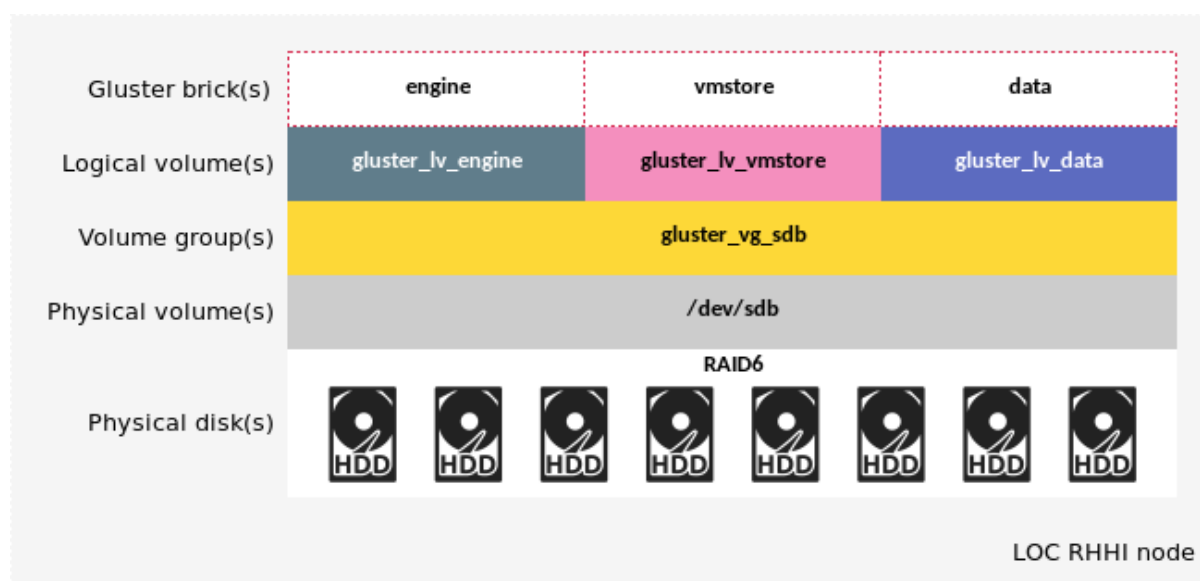


Figure 7.6: Red Hat Hyperconverged Infrastructure for Virtualization Host Logical Volumes and Gluster Bricks

A brick is then “exported” by given a mounting point on the system:

Table 7.15: Red Hat Hyperconverged Infrastructure for Virtualization Gluster bricks and mounting point

| Index | LV Name | File System | Mounting Point | LV Type |
|-------|--------------------|-------------|-------------------------|---------|
| 1 | gluster_lv_engine | XFS | /gluster_bricks/engine | thick |
| 2 | gluster_lv_vmstore | XFS | /gluster_bricks/vmstore | thin |
| 3 | gluster_lv_data | XFS | /gluster_bricks/data | thin |

7.3.1.2 Volumes

Brick is a local storage to a host. The power of the GlusterFS is to build a cluster of storage over bricks on multiple servers. A group of bricks is called gluster volume⁴. A 3-server Red Hat Hyperconverged Infrastructure for Virtualization configuration will have a gluster cluster that includes three servers in its trusted pool, and the cluster will have three volumes — engine, vmstore, and data.

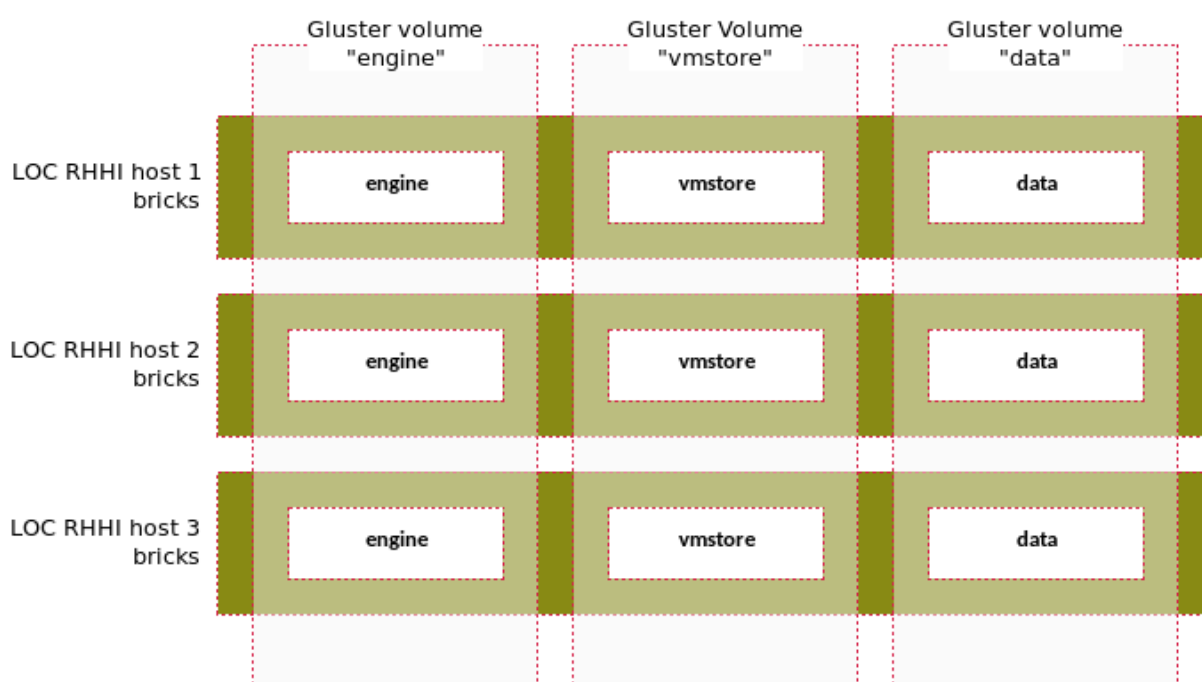


Figure 7.7: 3-server Red Hat Hyperconverged Infrastructure for Virtualization
Gluster cluster overview

3. **engine**: collection of all **engine** bricks.
4. **vmstore**: collection of all **vmstore** bricks.
5. **data**: collection of all **data** bricks.

⁴Gluster volume is different from logical volume. It is a logical grouping of gluster bricks, which in turn is a grouping of logical volumes.

7.3.1.3 Volume type — Replicated

For a Red Hat Hyperconverged Infrastructure for Virtualization deployment, a key decision is the volume type for the gluster. Default volume type is **replicated** mode, which will replicate files across all bricks in the volume. Refer to Gluster doc[19] for a full list of supported mode.

Relicated mode benefits workload that seeks high availability and high reliability. The downside is that the cluster will **not** grow linearly by adding more storage servers to the trusted pool. The size of the storage is limited to the size of the smallest brick in the volume.

Therefore, in a default Red Hat Hyperconverged Infrastructure for Virtualization configuration, the mean to increase the gluster capacity is limited to:

1. Purchase disk of larger capacity than what is suggested in the **BOM**.
2. Use RAID10 instead of RAID6 on gluster disks.

All these, however, are only applicable to a single server. If growing storage is a priority, one should replace **replicated** mode with **distributed** or **distributed replicated** mode.

Note that there is **no** arbiter volume in the default setup.

7.3.2 NFS

NFS is a network file system supported by the Red Hat Hyperconverged Infrastructure for Virtualization.

First of all, NFS can be the format of backing store that forms a gluster volume. Therefore, a gluster can very well be NFS based.

Here we are discussing NFS without gluster abstraction, in which case the key difference is the loss of benefit determined by those gluster modes[19] such as data replication. On the flip side NFS removes gluster's overhead, thus provides better WRITE performance in general.

NFS in Red Hat Hyperconverged Infrastructure for Virtualization are based on minimal two disks in each server:

1. Disks are configured in **RAID1**.
2. Linux logical volume `vm-data` is created.
3. Assign mounting point `/mnt/vm`.
4. Export share to NFS in `/etc/exports` as: `/mnt/vm *(rw)`

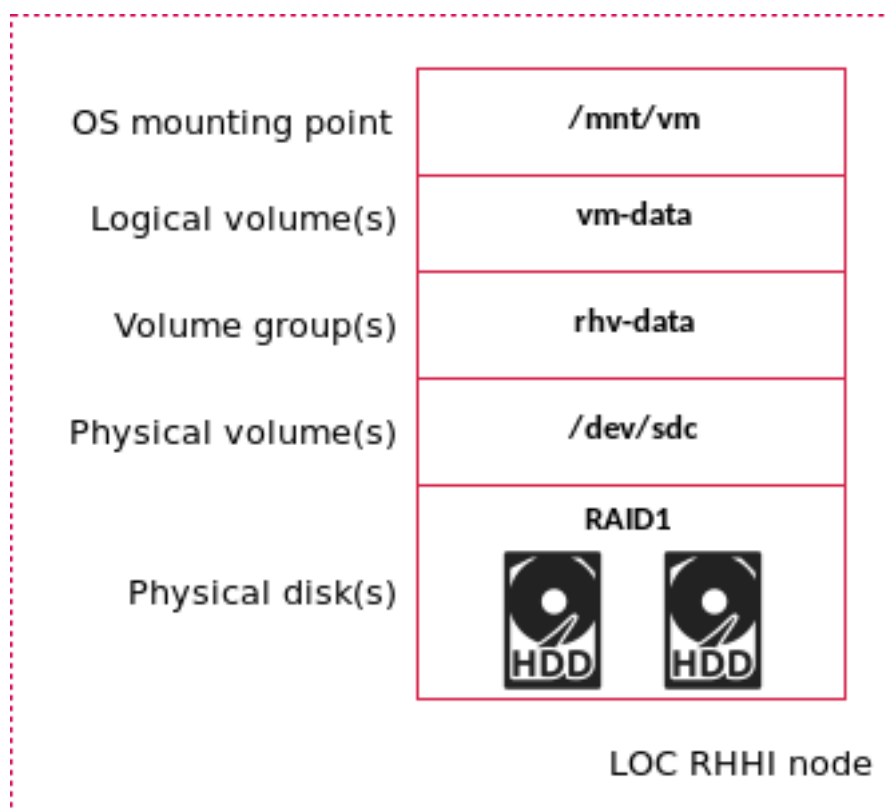


Figure 7.8: Red Hat Hyperconverged Infrastructure for Virtualization Host Logical Volumes and NFS

7.3.3 RHV storage domains

All storages are made available in RHV as storage domains.

A storage domain is a collection of images that have a common storage interface. A storage domain contains complete images of templates and virtual machines (including snapshots), or ISO files. A storage domain can be made of either block devices (SAN - iSCSI or FCP) or a file system (NAS - NFS or other POSIX compliant

file systems).

There are two [domains types][20] to consider⁵:

1. **Data** domain: A data domain holds the virtual hard disks and OVF files of all the virtual machines and templates in a data center. In addition, snapshots of the virtual machines are also stored in the data domain.

There must be at least one data domain in each Red Hat Hyperconverged Infrastructure for Virtualization deployment.

2. **ISO** domain: ISO domains store ISO files (or logical CDs) used to install and boot operating systems and applications for the virtual machines. An ISO domain removes the data center's need for physical media. An ISO domain can be shared across different data centers. ISO domains can only be NFS-based.

Only one ISO domain can be added to a data center.

Table 7.16: Red Hat Hyperconverged Infrastructure for Virtualization storage to RHV storage domain mapping

| RHHI-v Storage | RHV Domain Name | Domain Type |
|--|-----------------|-------------|
| gluster volume engine | hosted_storage | Data |
| gluster volume vmstore | vmstore | Data |
| gluster volume data | iso | ISO |
| NFS | host<1/2/3>nfs | Data |

7.3.4 LVM cache

An LVM Cache logical volume (LV) can be used to improve the performance of a block device by attaching to it a smaller and much faster device to act as a data acceleration layer. When a cache is attached to an LV, the Linux kernel subsys-

⁵[Export](#) domain is deprecated and replaced by [Data](#) domain.

tems attempt to keep “hot” data copies in the fast cache layer at the block level. Additionally, as space in the cache allows, writes are made initially to the cache layer. The results can be better Input/Output (I/O) performance improvements for many workloads.

In general, enabling cache will improve I/O performance. [LVM cache for Red Hat Gluster storage][21] has more details on how to determine whether the lvm cache should be used.

In Red Hat Hyperconverged Infrastructure for Virtualization, enabling this feature takes:

1. In **disk configuration** and **BOM**, two 800GB SSD disks are set in RAID1 for this purpose.

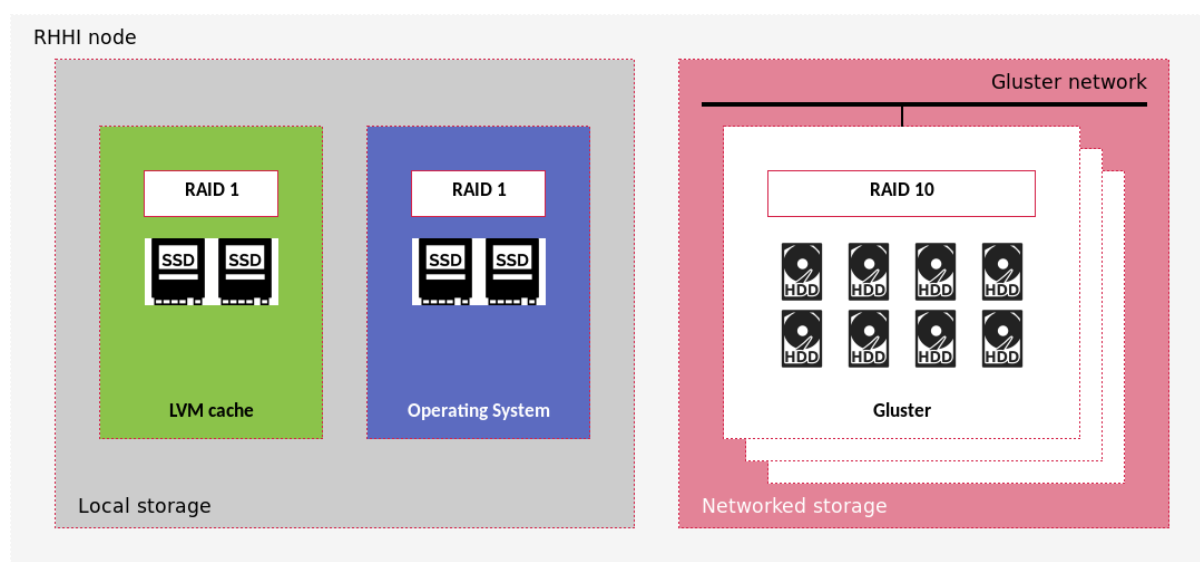


Figure 7.9: Red Hat Hyperconverged Infrastructure for Virtualization node disk configurations w/ LVM cache enabled

2. In **deployment template**:
3. Set `storage/lvm/size` to the size of cache in GB. It should be an integer value ≤ 800 .
4. Set `storage/lvm/mounting_point`, eg. `sdb`.

Leaving the value of `storage/lvm` blank is to skip setting up the cache on all hosts in

deployment even though disks are in place.

8

Deployment

Traditionally deploying RHHI-v on servers have two options: using an ISO image, or using RHEL and Red Hat [yum](#) package manager like installing other Red Hat software. Lenovo Open Cloud has developed a [bootstrap](#) appliance to make RHHI-v deployment fully automated.

8.1 Prerequisite

Before starting auto deployment process, the following conditions should be met:

1. Servers are cabled as in [server cable schema](#).
2. Switch ports are configured as in [configured](#).

8.2 Bootstrap appliance

8.2.1 Bootstrap services

Bootstrap appliance can be either a physical server or a virtual appliance. It includes a set of core services to facilitate deployment:

Table 8.1: Bootstrap Appliance Core Services

| Index | Service | Description |
|-------|--------------------|--|
| 1 | Software repo | Source of software installation |
| 2 | Automation | Deployment orchestration |
| 3 | OS deployment | Out-of-band (OOB) management of server |
| 4 | Configuration repo | Source of deployment templates & instance configurations |
| 5 | Validation | Deployment validation |
| 6 | OS Image | Source of OS images and VM images |
| 7 | Launchpad | Automation runtime worker environment |

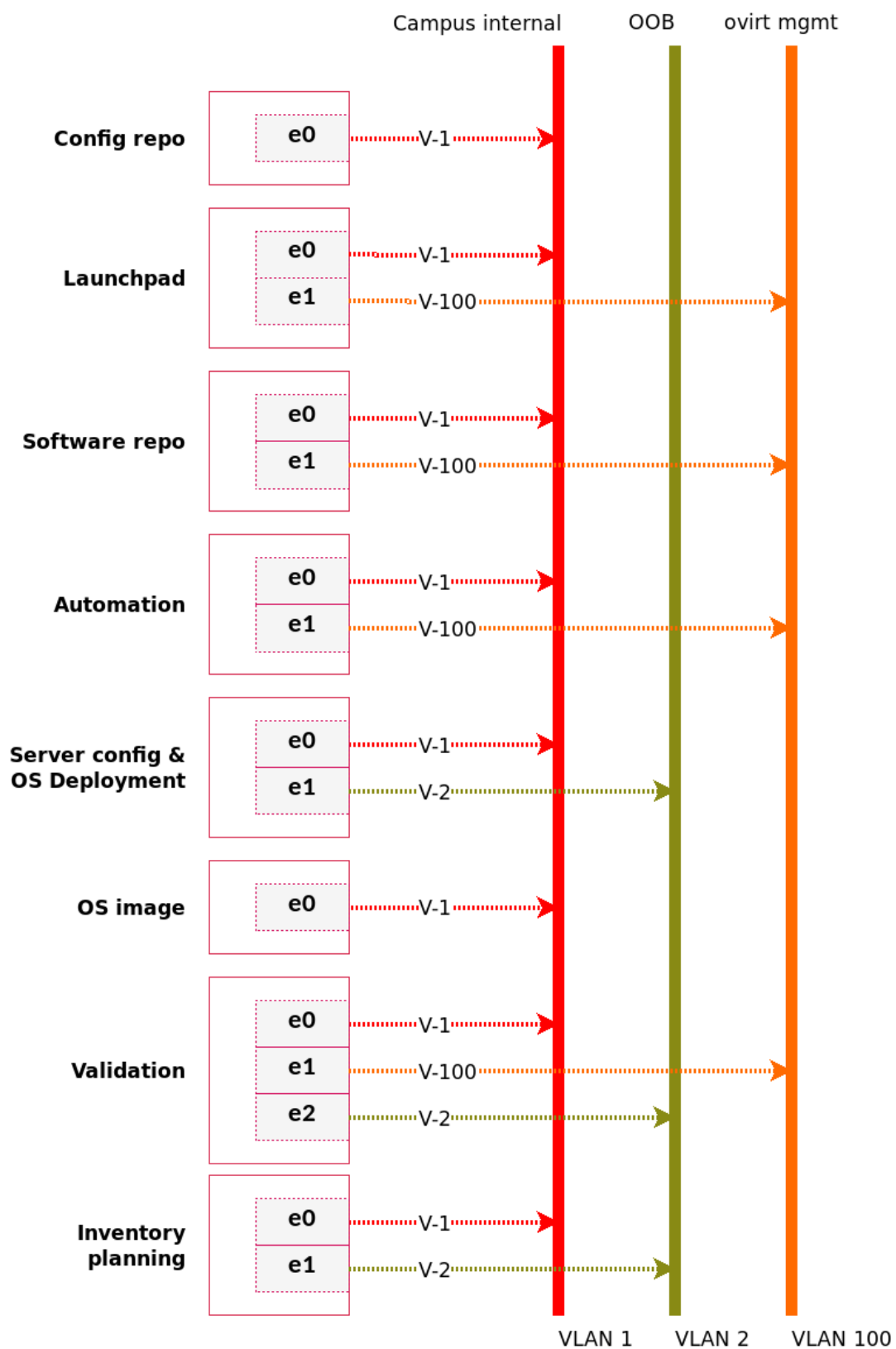
In addition, there are add-on services that can be enabled to extend automation such as discovering server on a monitored network:

Table 8.2: Bootstrap Appliance Add-on Services

| Index | Service | Description |
|-------|----------------------|--|
| 1 | Discovery | Hardware detection and configuration |
| 2 | Inventory | Infrastructure planning and role-based hardware commission |
| 3 | Single pane of glass | Unified administration interface |

8.2.2 Bootstrap networks

Bootstrap appliance must have proper networking connection to LOC RHHI-v's servers. As shown in fig. 8.1, three networks are essential. They are also part of the overall **LOC RHHI logical networks**:

**Figure 8.1:** LOC Bootstrap Appliance Networks

- ☞ **Campus internal**: general purpose connection between bootstrap appliance services and Red Hat Hyperconverged Infrastructure for Virtualization node.
- ☞ **BMC**: for OOB server management and OS provisioning.
- ☞ **ovirt mgmt**: for RHV administration during buildup.

Noticeably three RHHI-v networks are not present in fig. 8.1: **GlusterFS** (VLAN 400), **Physical server management** (VLAN 3), and **RHHI provisioning** (VLAN 10). These omissions are intentional because they are not used by bootstrap appliance. Their configurations are defined in **Red Hat Hyperconverged Infrastructure for Virtualization deployment template**, and will be setup accordingly.

9

Appendix: Bill of Material

9.1 Switches

9.1.1 1Gb switch

Table 9.1: Red Hat Hyperconverged Infrastructure for Virtualization RackSwitch G8052 1Gb bill of materials, rear to front

| Part Number | Product Description | Qty |
|-------------|--|----------|
| 7159HC1 | Switch-G8052 : Lenovo RackSwitch G8052 (Rear to Front) | 1 |
| ASY2 | Lenovo RackSwitch G8052 (Rear to Front) | 1 |
| 3793 | 3m Yellow Cat5e Cable | up to 37 |
| A1PJ | 3m Passive DAC SFP+ Cable | 1 |
| 6311 | 2.8m, 10A/100-250V, C13 to C14 Jumper Cord | 2 |
| 00WW816 | Essential Service - 3Yr 24x7 4Hr Response | 1 |

For front-to-rear configuration, replace [ASY2](#) with [ASY1](#).

9.1.2 10Gb switch

Table 9.2: Red Hat Hyperconverged Infrastructure for Virtualization RackSwitch G8272 10Gb Bill of Material

| Part Number | Product Description | Qty |
|-------------|--|----------|
| 7159HCW | Switch-G8272 : Lenovo RackSwitch G2752 (Rear to Front) | 1 |
| ASRD | Lenovo RackSwitch G8272 (Rear to Front) | 1 |
| A1PJ | 3m Passive DAC SFP+ Cable | up to 24 |
| A1DP | 1m QSFP to QSFP+ Cable | 1 |
| 3793 | 3m Yellow Cat5e Cable | 1 |
| 6311 | 2.8m, 10A/100-250V, C13 to C14 Jumper Cord | 2 |
| 00WW781 | Foundation Service - 3Yr Next Business Day Response | 1 |

For front-to-rear configuration, replace [ASRD](#) with [ASRE](#).

9.1.3 25Gb switch

Table 9.3: Red Hat Hyperconverged Infrastructure for Virtualization RackSwitch NE2572 25Gb Bill of Material

| Part Number | Product Description | Qty |
|-------------|---|----------|
| 7159HE3 | Switch-25G : Lenovo ThinkSystem NE2572 RackSwitch (Rear to Front) | 1 |
| AV19 | Lenovo ThinkSystem NE2572 RackSwitch (Rear to Front) | 1 |
| AV1F | Lenovo 3m 25G SFP28 Active Optical Cable | up to 24 |
| AV1L | Lenovo 3m 100G QSFP28 Active Optical Cable | 1 |
| 3793 | 3m Yellow Cat5e Cable | 1 |

| Part Number | Product Description | Qty |
|-------------|---|-----|
| 6204 | 2.8m, 10A/100-250V, C13 to IEC 320-C20 Rack Power Cable | 2 |
| 5WS7A0688 | Premier with Essential - 3Yr 24x7 4Hr Response | 1 |

For front-to-rear configuration, replace [AV19](#) with [AV1A](#).

9.2 Servers

Default configuration uses SATA SSD and 10G NICs. For SAS SSDs, see the [SAS SSDs](#) section:

Table 9.4: SR650 Bill of Materials based on SATA SSDS

| Part Number | Product Description | Qty |
|-------------|--|-----|
| 7X06CTO1W | Red Hat Hyperconverged Infrastructure for Virtualization: ThinkSystem SR650 - 3yr Warranty | 1 |
| AUVV | ThinkSystem SR650 2.5" Chassis with 8, 16 or 24 bays | 1 |
| AWEC | Intel Xeon Silver 4114 10C 85W 2.2GHz Processor | 2 |
| AUND | ThinkSystem 32GB TruDDR4 2666 MHz (2Rx4 1.2V) RDIMM | 12 |
| AUR5 | ThinkSystem SR650 2.5" AnyBay 8-Bay Backplane | 3 |
| AUNK | ThinkSystem RAID 930-16i 4GB Flash PCIe 12Gb Adapter | 1 |
| B49M | ThinkSystem 2.5" Intel S4610 480GB Mainstream SATA 6Gb Hot Swap SSD | 2 |
| B49N | ThinkSystem 2.5" Intel S4610 960GB Mainstream SATA 6Gb Hot Swap SSD | 2 |

| Part Number | Product Description | Qty |
|-------------|--|----------|
| B0YS | ThinkSystem 2.5" 2.4TB 10K SAS 12Gb Hot Swap 512e HDD | up to 20 |
| AUR7 | ThinkSystem 2U x8/X8/x8ML2 PCIE FH Riser 1 | 1 |
| AURC | ThinkSystem SR550/SR590/SR650 (x16/x8)/(x16/x16) PCIe FH Riser 2 Kit | 1 |
| AUKK | ThinkSystem 1Gb 4-port RJ45 LOM | 1 |
| AUKX | ThinkSystem Intel X710-DA2 PCIe 10Gb 2-Port SFP+ Ethernet Adapter | 2 |
| AVWF | ThinkSystem 1100W (-48V DC) Platinum Hot-Swap Power Supply | 2 |
| 6400 | 2.8m, 13A/100-250V, C13 to C14 Jumper Cord | 2 |
| AUPW | ThinkSystem XClarity Controller Standard to Enterprise Upgrade | 1 |
| AXCH | ThinkSystem Toolless Slide Rail Kit with 2U CMA | 1 |
| A51P | 2m Passive DAC SFP+ Cable | 4 |

9.2.1 SAS SSDs

To use SAS SSDs, replace B49M with B16Z, and B49N with B170:

Table 9.5: SR650 Bill of Materials based on SAS SSDs

| Part Number | Product Description | Qty |
|-------------|---|-----|
| B16Z | ThinkSystem 3.5" HUSMM32 400GB Performance SAS 12Gb Hot Swap SSD 7N47A00997 | 2 |

| Part Number | Product Description | Qty |
|-------------|--|-----|
| B170 | ThinkSystem 3.5" HUSMM32 800GB Performance SAS 12Gb Hot Swap SSD | 2 |

9.2.2 25Gb NIC

To use 25Gb NIC, replace AUKX with B0WY:

Table 9.6: SR650 Bill of Materials based on 25G NIC

| Part Number | Product Description | Qty |
|-------------|--|-----|
| B0WY | Intel XXV710-DA2 10/25GbE SFP28 2-Port PCIe Ethernet Adapter | 2 |

9.3 Software

Based on a 3-node Red Hat Hyperconverged Infrastructure for Virtualization deployment:

Table 9.7: LOC 3-node RHHI-v software BOM, 3 year premium service support

| SKU | Product Description | Qty |
|-----------|---|-----|
| RS00139F3 | Red Hat Hyperconverged Infrastructure for Virtualization (RHHI-v) | 1 |
| RH00031F3 | Smart Management | 3 |
| MCT3475F3 | Red Hat Insights | 1 |
| 00MT202 | Lenovo xClarity Enterprise | 3 |

9.4 Other recommendations

We also recommend the features and upgrades in this section to maximize the security and manageability of the Red Hat Hyperconverged Infrastructure for Virtualization solution built using the configurations discussed in this document.

9.4.1 TPM 2.0 and Secure Boot

Trusted Platform Module (TPM) is an international standard for a secure cryptoprocessor, a dedicated microcontroller designed to secure hardware through integrated cryptographic keys. TPM technology is designed to provide hardware-based, security-related functions and is used extensively by Microsoft in Windows Server 2016 technologies including BitLocker, Device Guard, Credential Guard, UEFI Secure Boot, and others. **There is no additional cost to enable TPM 2.0 on Lenovo ThinkSystem servers.**

For the SR650, order Feature Code [AUK7](#).

9.4.2 ThinkSystem XClarity Controller Standard to Enterprise Level

The Lenovo XClarity™ Controller (aka. XCC) is the next generation management controller that replaces the baseboard management controller (BMC) for Lenovo ThinkSystem servers. Although the XCC Standard Level includes many important manageability features, we recommend upgrading to the XCC Enterprise Level of functionality. This enhanced set of features includes Virtual Console (out of band browser-based remote control), Virtual Media mounting, and other remote management capabilities.

For the SR650, order Feature Codes [B173](#).

9.4.3 Best recipes

Lenovo best recipe describes the exact Lenovo firmware versions that have been tested successfully. This includes firmware of all components in a server. Together with the **BOM**, they define precisely what a LOC RHHI host should have.

Below is an example of a best recipe for a LOC RHHI three-server configuration based on Lenovo System x3650 M5 rack servers with suggested **disk configurations** but without additional SSDs for LVM cache:

Table 9.8: Red Hat Hyperconverged Infrastructure for Virtualization SR650 server best recipe

| Slot | Updatable Unit | Installed Version |
|------|---|-------------------|
| 0 | Drive(ST1000NM0045) | LK86 |
| 1 | Drive(ST1000NM0045) | LK86 |
| 2 | Drive(ST1000NM0023) | LC5H |
| 3 | Drive(ST1000NM0045) | LK86 |
| 4 | Drive(ST1000NM0045) | LK86 |
| 4 | Intel X710 2x10GbE SFP+ Adapter (Etrack ID) | 80002A0A |
| 4 | Intel X710 2x10GbE SFP+ Adapter (Combined Option ROM Image) | 1.1313.0 |
| 5 | Drive(ST1000NM0045) | LK86 |
| 6 | Drive(ST1000NM0045) | LK86 |
| 7 | Drive(ST1000NM0045) | LK86 |
| 9 | ServeRAID M5210 (10140454) | 24.16.0-0104 |
| 9 | ServeRAID M5210 (10140454) | 24.16.0-0104 |
| N/A | IMM2 Firmware | TCOO46F-5.11 |
| N/A | IMM2 Backup Firmware | TCOO26H-3.70 |
| N/A | UEFI Firmware/BIOS | TCE138D-2.80 |

| Slot | Updatable Unit | Installed Version |
|------|---------------------------|-------------------|
| N/A | UEFI Backup Firmware/BIOS | TCE128I-2.31 |
| N/A | DSA Diagnostic Software | DSALB2Q-10.3 |

Please consult with the Lenovo sales and support for the best recipe of your hardware and configuration.

10

Appendix: Implementation Questionnaire

Implementation questionnaire are tools to map Lenovo Open Cloud infrastructure to your environment.

10.1 Switches

Table 10.1: Implementation Worksheet — Switches

| Switches | Type | Model | Firmware | Qty | |
|---------------|------|-------|---------------|-----|-------------------------|
| access switch | 1Gb | G8052 | 8.4.12 ENOS | 2 | suggested: your env: |
| access switch | 10Gb | G8272 | 10.9.3.0 CNOS | 2 | suggested your env: |

10.1.1 Switch port sizing worksheet

This worksheet is to assist network admin to determine switch port sizing based on RHHI-v configuration. Dedicating the Lenovo switches in the **BOM** section will have enough ports to support the largest configuration (12-node deployment).

However, in many cases deployment is sharing switches with other infrastructure.

Thus, the admin can use this worksheet to look up how many ports are needed for a configuration.

Table 10.2: Switch port sizing worksheet

| RHHI-v Config Index | RHHI-v Servers | G8052 #1 | G8052 #2 | G8272 #1 | G8272 #2 |
|---------------------|----------------|----------|----------|----------|----------|
| 1 | 3 | 6 | 3 | 3 | 3 |
| 2 | 6 | 12 | 6 | 6 | 6 |
| 3 | 9 | 18 | 9 | 9 | 9 |
| 4 | 12 | 24 | 12 | 12 | 12 |

10.2 Storage sizing worksheet

Storage sizing is based on replicated gluster volumes. For additional gluster capacity, [distributed](#) or [distributed replicated](#) mode should be used instead.

Table 10.3: Storage sizing worksheet ([replicated](#) gluster)

| RHHI-v Config Index | RHHI-v Servers | Gluster (TB) | NFS (TB) |
|---------------------|----------------|--------------|----------|
| 1 | 3 | 12 | 4 |
| 2 | 6 | 12 | 8 |
| 3 | 9 | 12 | 12 |
| 4 | 12 | 12 | 16 |

10.3 Network mapping

In order for Lenovo Open Cloud functions to work as designed, it is important to map your network environment to meet requirements presented by suggested values.

The setup described in the **logical networks** section is an ideal networking case. In reality networking environment can hardly map directly into these schema. Further, each of the management services have an admin portal and an API, thus a deployment requires a set of IP addresses on each network. How many of them are now tied with networking choices.

Here we have identified five deployment strategies of the LOC networks to assist your decision. The key is how to map these **three core management networks**:

1. campus internal
2. BMC
3. ovirt management

10.3.1 Strategy 1

Keep all three networks separated is the recommended solution.

Table 10.4: IP address sizing worksheet, all separated.

| RHHI Config Index | RHHI Servers | Campus internal | BMC | ovirt |
|-------------------|--------------|-----------------|-----|-------|
| 1 | 3 | 9 | 8 | 7 |
| 2 | 6 | 9 | 11 | 10 |
| 3 | 9 | 9 | 14 | 13 |
| 4 | 12 | 9 | 17 | 16 |

10.3.2 Strategy 2

A common strategy benefits OOB management of servers by combining `campus` \hookrightarrow `internal` with `BMC`. If you have a dedicated OOB management network, `BMC` can also be combined with that, in which case this becomes the **Case 1** above because these three networks are still separated.

In any case, we highly recommend to avoid mixing `BMC` with other networks, because OOB is the last line of defense when the deployment is broken in some way.

Table 10.5: IP address sizing worksheet, `campus internal == BMC`, `ovirt` separated

| RHHI Config Index | RHHI Servers | Campus internal== BMC | ovirt |
|-------------------|--------------|-----------------------|-------|
| 1 | 3 | 13 | 8 |
| 2 | 6 | 16 | 11 |
| 3 | 9 | 19 | 14 |
| 4 | 12 | 22 | 17 |

10.3.3 Strategy 3

This strategy improves accessibility to the Runtime service management portal and API. It does not affect other Platform management services because they have `Campus internal` for UI and API already.

However, it makes management traffic between the Runtime service and other Platform management services, which are program calls to APIs, share the same network with other management tasks, which are mostly manual.

Table 10.6: IP address sizing worksheet, `campus internal == ovirt`, `BMC` separated

| RHHI Config Index | RHHI Servers | Campus internal== ovirt | BMC |
|-------------------|--------------|-------------------------|-----|
| 1 | 3 | 13 | 8 |

| RHHI Config Index | RHHI Servers | Campus internal== ovirt | BMC |
|-------------------|--------------|-------------------------|-----|
| 2 | 6 | 16 | 11 |
| 3 | 9 | 19 | 14 |
| 4 | 12 | 22 | 17 |

10.3.4 Strategy 4

This strategy treats `ovirt` and `BMC` internal, and emphasizes `Campus internal` being the converge point for instance access.

Table 10.7: IP address sizing worksheet, `ovirt == BMC`, `Campus internal` separated

| RHHI Config Index | RHHI Servers | ovirt == BMC | Campus internal |
|-------------------|--------------|--------------|-----------------|
| 1 | 3 | 10 | 9 |
| 2 | 6 | 13 | 9 |
| 3 | 9 | 16 | 9 |
| 4 | 12 | 19 | 9 |

10.3.5 Strategy 5

This strategy is the simplest solution because all three networks are combined into one flat network. It benefits accessibility, thus is often seen in a development setting.

However, it removes intended isolation by the design, making the deployment prone to error and hard to debug because all traffics are now mixed.

Table 10.8: IP address sizing worksheet, a flat network.

| RHHI Config Index | RHHI Servers | Campus internal == ovirt == BMC |
|-------------------|--------------|---------------------------------|
| 1 | 3 | 14 |
| 2 | 6 | 17 |
| 3 | 9 | 20 |
| 4 | 12 | 23 |

10.3.6 Network mapping worksheet

Once a deployment strategy has been selected, the worksheet below can be filled out to further clarify network requirements:

Table 10.9: LOC RHHI network mapping worksheet

| Network | | VLAN | Subnet | Mask | Gateway |
|----------------------------|------------|------|---------------|------|---------------|
| Campus internal | suggested: | 1 | 192.168.1.x | /24 | 192.168.1.1 |
| | your env: | | | | |
| BMC | suggested: | 2 | 192.168.2.x | /24 | 192.168.2.1 |
| | your env: | | | | |
| Physical server management | suggested: | 3 | 192.168.3.x | /24 | 192.168.3.1 |
| | your env: | | | | |
| RHHI provisioning | suggested: | 10 | 192.168.10.x | /28 | 192.168.10.1 |
| | your env: | | | | |
| OVIRT management | suggested: | 100 | 192.168.100.x | /28 | 192.168.100.1 |
| | your env: | | | | |
| GlusterFS | suggested: | 400 | 192.168.40.x | /28 | 192.168.40.1 |

| Network | VLAN | Subnet | Mask | Gateway |
|---------|------|--------|------|---------|
|---------|------|--------|------|---------|

your env:

11

Contributors

A special thank you to the following Lenovo colleagues for their contributions to this document:

- ☞ Jay Bryant, Senior Engineer — Data Center Group
- ☞ Weiwei Chen, Engineer — Data Center Group
- ☞ Chengzhen Chu, Engineer — Data Center Group
- ☞ Patrick Jennings, Engineer — Data Center Group
- ☞ Guannan Sun, Engineer — Data Center Group
- ☞ Ricky Stambach, Engineer — Data Center Group
- ☞ Zelin Wang, Engineer — Data Center Group

12

Appendix: Apply switch configurations using CLI

In this section we take the two port 17 on G8052 switches shown below as example to demonstrate switch side configurations using switch CLI. The same procedure and CLI commands are applicable to G8272 switch, too.

Table 12.1: Example of LOC RHHI-v Server-Switch Cable Schema

| Port | G8052 #1 | G8052 #2 |
|------|---------------|---------------|
| 17 | server 1 eno1 | server 1 eno2 |

CLI is conventional method that admin takes to manually examine and configure a switch. In general, the workflow follows these steps:

1. Login in to the switch as `admin` user.
2. Enter `config` mode.
3. Select the port to configure.
4. Apply settings.
5. `exit` config mode. This then allows you to select another port without leaving the `config` mode.
6. Save the settings to switch after all ports have been configured. This is important because the switch will lose all the temporary settings after reboot.

12.1 Login to switch

Lenovo G8052 switch allows telnet access by default. It can also support SSH. To login, you need the switch IP, for example, 192.168.43.51, user name, and password. This user account should have been set up to have proper privilege to change intended configurations.

```
1 $ telnet 192.168.43.51
2 Trying 192.168.43.51...
3 Connected to 192.168.43.51.
4 Escape character is '^]'.
5
6 Lenovo RackSwitch G8052.
7
8
9 Enter login username: xxx    <-- admin user name
10 Enter login password: xxx   <-- admin password
11
12 G8052>
```

Note that prompt `G8052>` shown above is depending on the switch name, in this example, `G8052`. This value is configurable. Thus your terminal prompt may look different.

12.2 Enter config mode

The rest of port configurations will be applied in **config mode**.

```
1 G8052> enable <-- enable admin mode
2 G8052> configure terminal <-- to enter config mode
```

12.3 Apply access mode configurations

Example:

```
1 G8052> interface port 1 <-- select port
2 G8052> switchport mode access <-- set switch port mode
3 G8052> switchport access vlan 2 <-- set switch port native vlan
```

1. Select the port to config: replace 1/1 with 1/2 for port 2, and 1/3 for port 3.
2. Set port mode: `access`
3. Set default vlan: 2

12.4 Apply trunk mode without LACP

Example:

```
1 G8052> interface port 17 <-- select port 17
2 G8052> switchport mode trunk <-- set port mode to "trunk"
3 G8052> switchport trunk allowed vlan 1-3,10,100 <-- set tagged vlans
4 G8052> switchport trunk native vlan 2 <-- set native vlan
```

Note that you must set `native` vlan after `allowed` vlan, because the native vlan must also be included as an *allowed* vlan. Otherwise, the switch will use the default vlan 1 as the default native vlan.

12.5 Setup inter-switch-links (ISL)

Usually we use the port 49~52 in G8052 (G8272: 49~54) for ISL, because port 49~52 have 10G speed, port 1~48 only have 1G speed. Using the G8052 & G8272 switch for below example description. NE2572 ISL config is similar with G8272.

Note Two switches firmware version must be the same in order to use the ISL for the redundancy.

12.5.1 Connect the 2 switches with the ISL cable line

We need connect them by one or two ISL lines to make the redundancy.

- ☞ For G8052, use 10G Passive DAC SFP+ Cable (see BOM) as the ISL line for ports 49-52.
- ☞ For G8272, use the 40G Passive DAC QSFP+ Cable (see BOM) as the ISL line for ports 49-54.
- ☞ For NE8272, use the 100G Passive DAC SFP+ Cable (see BOM) as the ISL line for ports 49-54.

12.5.2 G8052 ISL configuration

Using port 49 (commonly aliased XGE1) as the ISL ports for example.

1. Setup G8052 #1's XGE1 port configurations:

```
1 G8052> interface port XGE1 <-- config port 49
2 G8052> description "ISL" <-- define this is the ISL port
3 G8052> switchport mode trunk
4 G8052> switchport trunk allowed vlan 1-3,10,100,3999
5 G8052> switchport trunk native vlan 3999
6 G8052> exit
```

2. Enable LACP on XGE1:

```
1 G8052> interface port XGE1 <-- config port 49
2 G8052> lacp mode active
3 G8052> lacp key 999 <-- create LACP key
```

3. Setup vLAG using the LACP key created above:

```
1 G8052> vlag enable
2 G8052> vlag tier-id 200
3 G8052> vlag hlthchk peer-ip x.x.x.x <-- IP of G8052 #2
4 G8052> vlag isl adminkey 999 <-- LACP key created above
```


4. On G8052 #2 switch, repeat steps 1-3, especially use the same `tier-id` and ISL `adminkey`, at the line `vlag hlthchk peer-ip` use the IP of G8052 #1 switch.

12.5.3 For G8272 ISL configuration

Using port 52 (commonly aliased `XGE4`) as the ISL ports for example.

1. Setup G8272 #1's XGE4 port configurations:

```
1 G8272> interface Ethernet 1/52
2 G8272> description ISL
3 G8272> switchport mode trunk
4 G8272> switchport trunk allowed vlan 3999, <other workload
  ↪ vlags>
5 G8272> switchport trunk native vlan 3999
6 G8272> exit
```

2. Setup a port-channel:

```
1 G8272> interface port-channel 52 <-- port-channel index
2 G8272> lacp suspend-individual
3 G8272> switchport mode trunk
4 G8272> switchport trunk allowed vlan 3999, <other workload
  ↪ vlags>
5 G8272> switchport trunk native vlan 3999
6 G8272> exit
```

3. Add XGE4 to the port-channel:

```
1 G8272> interface Ethernet 1/52
2 G8272> channel-group 52 mode active <-- same port-channel index
```

4. Setup ISL on the port-channel:

```
1 G8272> vlag tier-id 201
2 G8272> vlag isl port-channel 52
```

```
3 G8272> vlag hlthchk peer-ip x.x.x.x vrf management <-- mgmt IP  
  ↪ of G8272 #2  
4 G8272> vlag startup-delay 1  
5 G8272> vlag enable
```

5. On G8272 #2 switch, repeat steps 1-4, especially use the same `tier-id` and “port-channel” index, at the line `vlag hlthchk peer-ip` use the IP of G8272 #1 switch.

12.6 Apply trunk + port-channel + vlag for server ports redundancy

This ties two ports from two different switches together. Example shows steps for setting up G8272 switch.

1. On G8272 #1 switch, set vlans for the port:

```
1 G8272> interface Ethernet 1/1  
2 G8272> switchport mode trunk  
3 G8272> switchport trunk allowed vlan 400,3999  
4 G8272> switchport trunk native vlan 3999  
5 G8272> exit
```

2. Setup a port-channel:

```
1 G8272> interface port-channel 1 <-- port-channel index  
2 G8272> switchport mode trunk  
3 G8272> switchport trunk allowed vlan 400,3999  
4 G8272> switchport trunk native vlan 3999  
5 G8272> exit
```

3. Add port 1 to the port-channel 1:

```
1 G8272> interface Ethernet 1/1  
2 G8272> channel-group 1 mode active <-- same port-channel index
```

4. Setup vlag instance on the port-channel:

```
1 G8272> vlag instance 1 port-channel 1
2 G8272> vlag instance 1 enable
```

5. On G8272 #2 switch, repeat steps 1-4, especially use the same port and “port-channel” index.

13

Appendix: Configure RHEL bonding interfaces

LOC deployment program will set up server side bonding interfaces automatically. User can set up additional interface and map it to a RHV network.

We will use `bond 0` for example to demonstrate creation of a bonding interface on a LOC RHVI-v server running RHEL 7.5 (fig. 13.1). We will highlight options and values that are important for Lenovo Open Cloud. For general information of bonding interface, please refer to Red Hat Enterprise Linux 7 Networking Guide[22] and Linux kernel bonding interface[23].

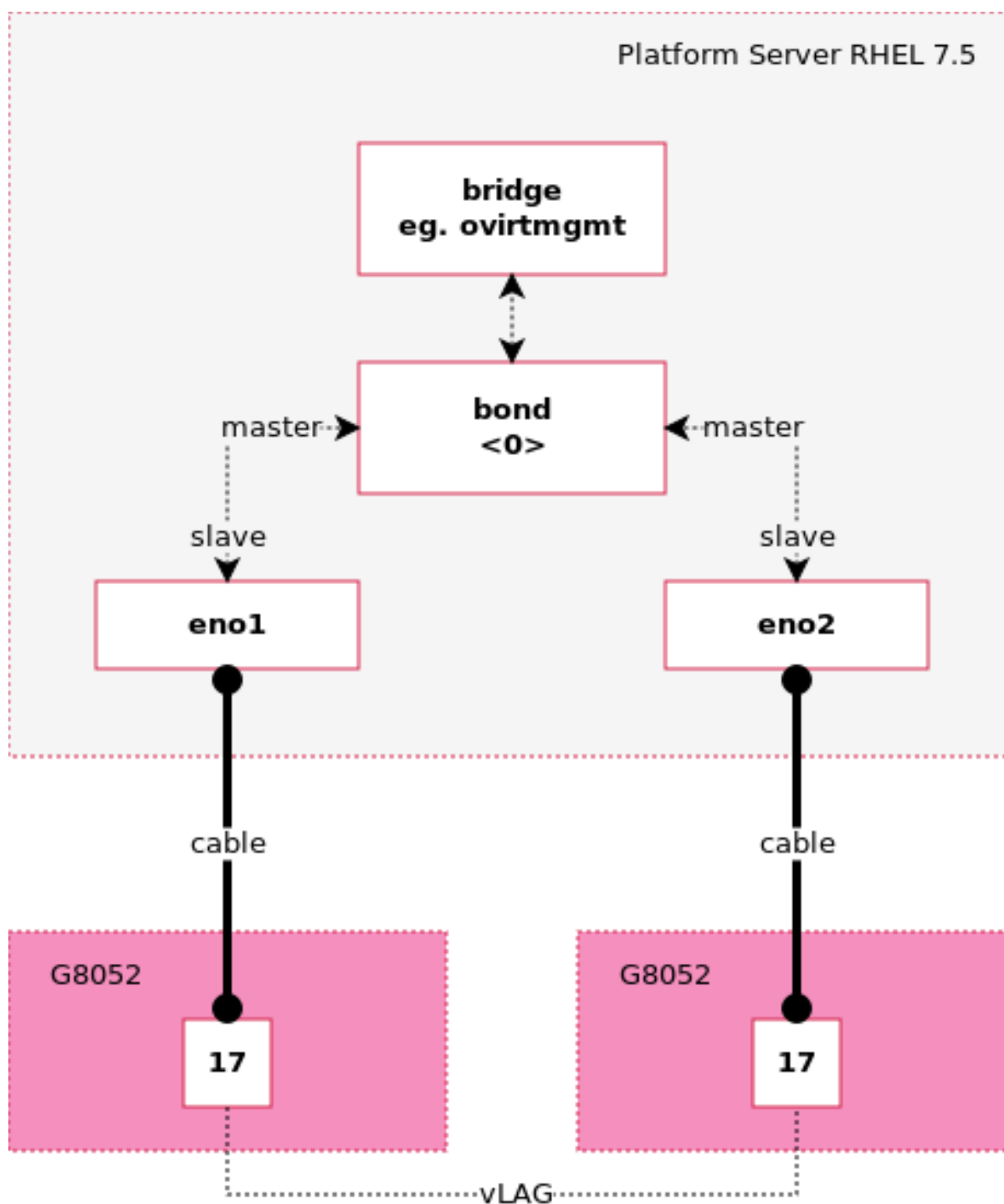


Figure 13.1: Example of LOC RHHI-v server bonding interface and their switch connections

Login a LOC RHHL-v servers:

1. Go to `/etc/sysconfig/network-scripts/`.
2. Create networking config file `ifcfg-bond0`. Replace `IPADDR`, `NETMASK`, `GATEWAY`, and `DNS1` values with your values:

```
1  `` `shell
2  DEVICE=bond0
3  BONDING_OPTS='mode=1 miimon=100'
4  ONBOOT=yes
5  BOOTPROTO=none
6  DEFROUTE=no
7  IPADDR=10.240.41.231
8  NETMASK=255.255.252.0
9  GATEWAY=10.240.40.1
10 DNS1=10.240.0.10
11  `` `
```

Note:

1. `DEVICE`: name of the bonding interface, in this example, `bond0`.
 2. `BONDING_OPTS`:
 1. `mode=1`: set to **active-backup**. Use `mode=4` for active-active interface. Refer to Red Hat Enterprise Linux 7 USING CHANNEL BONDING[24] for more information of bonding modes and their implications.
 2. `miimon=100`: enable MII monitoring.
 3. `DEFROUTE`: must be `no`.
3. Create `ifcfg-en01`:

```
1  `` `shell
2  DEVICE=en01
3  MASTER=bond0
4  SLAVE=yes
5  ONBOOT=yes
6  DEFROUTE=no
7  `` `
```

Note:

1. Here we setup a master-slave between the physical NIC interface `eno1` (as slave) and bonding interface `bond0` (as master).
4. Similar to `eno1` bonding, create `ifcfg-eno2`:

```
1  ````shell
2  DEVICE=eno2
3  MASTER=bond0
4  SLAVE=yes
5  ONBOOT=yes
6  DEFROUTE=no
7  ````
```

5. Restart the network service to take effect: `systemctl network restart`.
6. Use `ip a` to verify that three interfaces are up and running &mdash: `eno1`, `eno2`, and `bond0`.

14

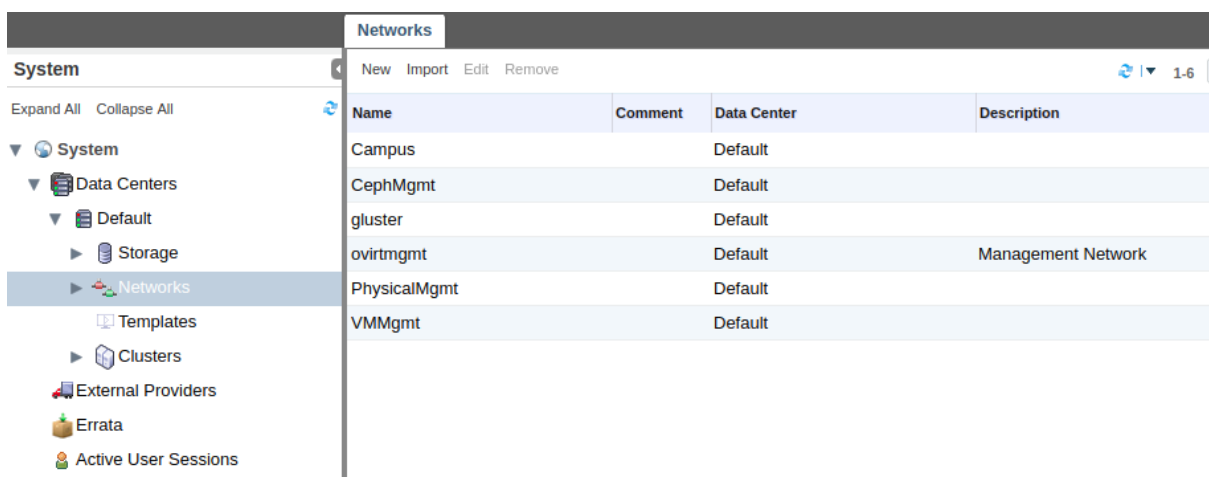
Appendix: Map RHEL interface to RHV network

Each RHV network is mapped to a virtual bridge on host. Virtual bridge is a flexible way to create multiple interfaces that can be tied to a physical or another virtual interface. In this reference architecture, we build virtual bridges on top of the bonding interface to serve two purposes:

1. Isolate networks by their function.
2. Ease of maintenance since users can add and remove user-defined networks as virtual bridges freely through LOC Automation service, or through the RHV Administrator Portal.

In this example, we will setup bridge interfaces using the RHV Administrator Portal:

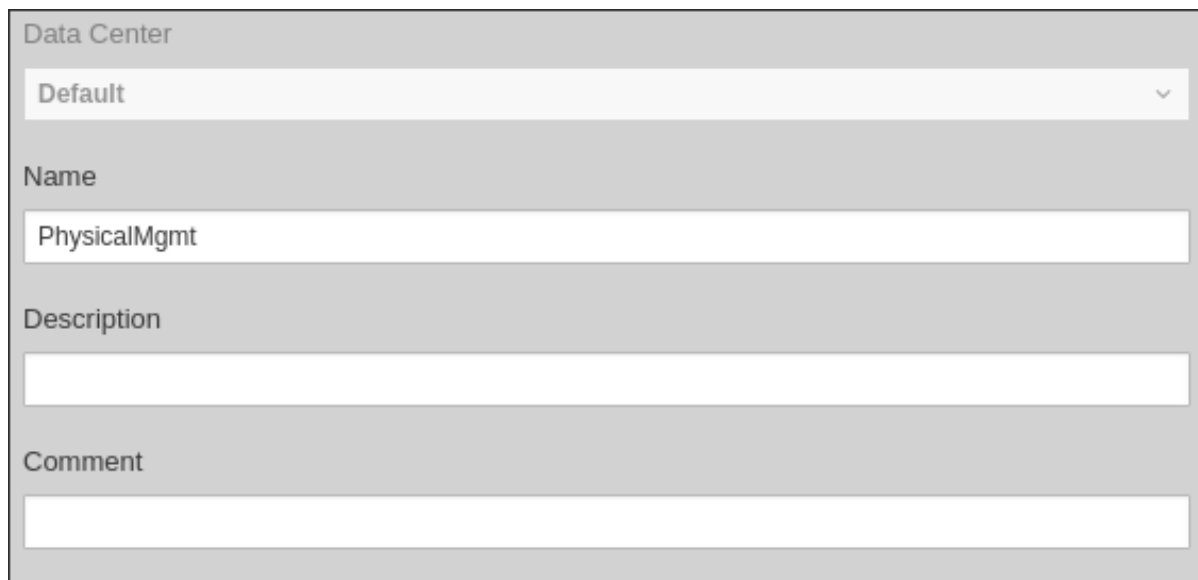
1. Login Admin portal and select [System](#) > [Network](#) (fig. 14.1).



| Name | Comment | Data Center | Description |
|--------------|---------|-------------|--------------------|
| Campus | | Default | |
| CephMgmt | | Default | |
| gluster | | Default | |
| ovirtmgmt | | Default | Management Network |
| PhysicalMgmt | | Default | |
| VMMgmt | | Default | |

Figure 14.1: View RHV Network List

2. Click [New](#) to create a new network, or [Edit](#) to modify an existing one. This will in turn create a virtual bridge on the host. In this example, [PhysicalMgmt](#) (fig. 14.2):



Data Center

Default

Name

PhysicalMgmt

Description

Comment

Figure 14.2: Create New RHV Network

3. Set VLAN. Only one VLAN can be associated with one bridge interface (fig. 14.3).



☒ Enable VLAN tagging

3

☒ VM network 

Figure 14.3: Setup RHV Network VLAN

4. Link the bridge to a bonding interface. This needs to be repeated for each RHHI-v hosts.
 1. Click [Compute](#), then [Hosts](#), and select a RHHI-v host.
 2. Click [Network Interfaces](#) subtab and then click [Setup Host Networks](#).

3. Drag and drop the newly created network to the correct interface (fig. 14.4).

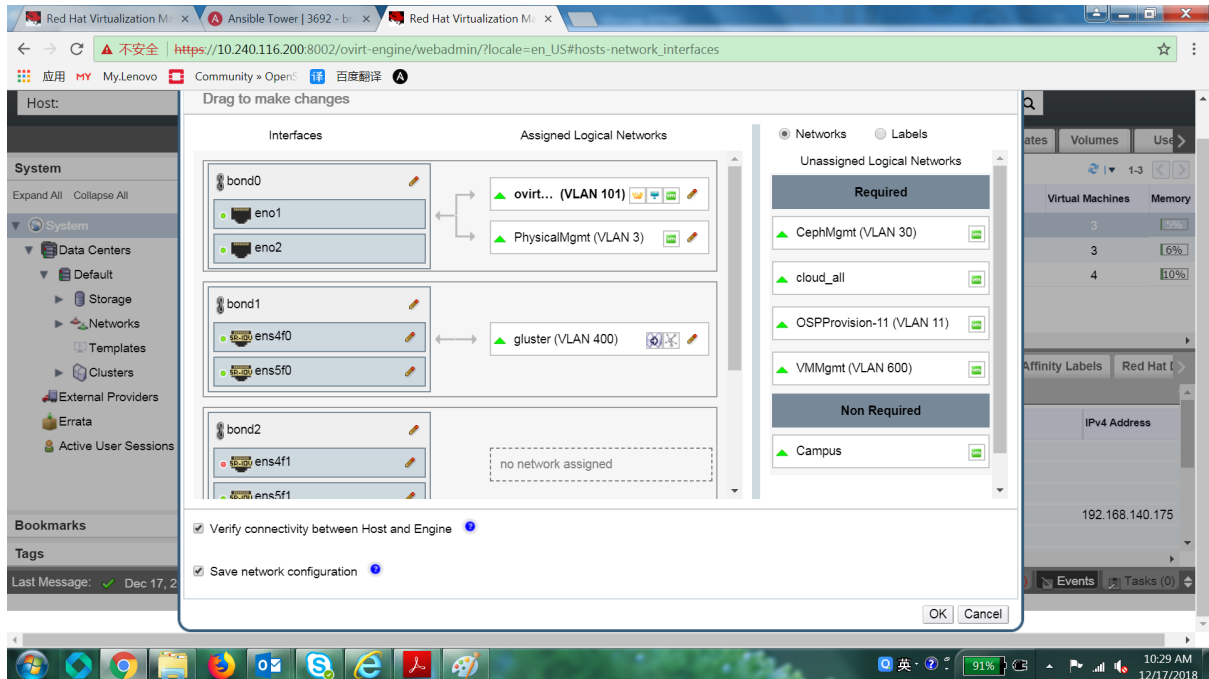


Figure 14.4: Link RHV Network to a bonding interface

To view the resulting bridge interface configurations created by the Admin Portal:

1. Login into any LOC RHVI-v server.
2. Go to `/etc/sysconfig/network-scripts/`.
3. Check `ifcfg -PhysicalMgmt`:

```

1  `` `shell
2  DEVICE=PhysicalMgmt
3  TYPE=Bridge
4  DELAY=0
5  STP=off
6  ONBOOT=yes
7  MTU=1500
8  DEFROUTE=no
9  NM_CONTROLLED=no

```

```
10 IPV6INIT=no
11 ```
```

Note:

1. **DEVICE**: bridge interface name, in this case, `PhysicalMgmt`.
 2. **TYPE**: must be `Bridge`.
4. Check `ifcfg -bond0.3` that links bridge `PhysicalMgmt` to `bond0`, and supports VLAN 3:

```
1  ```shell
2  DEVICE=bond0.3
3  VLAN=yes
4  BRIDGE=PhysicalMgmt
5  ONBOOT=yes
6  MTU=1500
7  DEFROUTE=no
8  NM_CONTROLLED=no
9  IPV6INIT=no
10 ```
```

Note:

1. **DEVICE**: to define a VLAN on an interface, both the filename and **DEVICE** should be set in format of `<interface name>.<vlan number>`.
2. **BRIDGE**: name of the bridge, in this case `PhysicalMgmt`, that this bonding interface is connected to.

15

Appendix: LOC RHHI-v storage performance test

Storage performance tests are to provide a reference of how well RHHI-v stack can retrieve data (read) and persist data (write).

As we have seen, RHHI-v is a virtualization application with layers of software stack. In each layer the software can handle read and write differently — some use cache so data lives in memory for a quick retrieval; others can force to flush to infrastructure and read back for verification. Database developers can rely on the atomic nature of a DB, while application developers might choose to flush data to file at mercy of his/her preference. It is impossible to enumerate all the possibilities, just that it is possible, but improbable to lay flat all the code and libraries in between, and examine every possible data path when a read/write is commanded from a layer up, or even several layers up.

In this RA we have developed a simplified view of the RHHI-v stack (see fig. 15.1). One fact for sure is that data, regardless of how it is being buffered, will eventually land on a hard disk in the system. This is true for networked storage also. Therefore, starting from disks, we attempted to define a path that data must go through if a READ is called, and vice versa, when a WRITE is commanded. It is not an exhaustive view that is applicable to all data. But it covers the concept in a high level and provided us a framework to design tests that we can then gauge performance of each end point as well as debug for performance degradation when a form of indirection, such as virtualization, is introduced.

Primarily we have divided the stack into seven layers (from bottom up), thus forming seven checkpoints for the test.

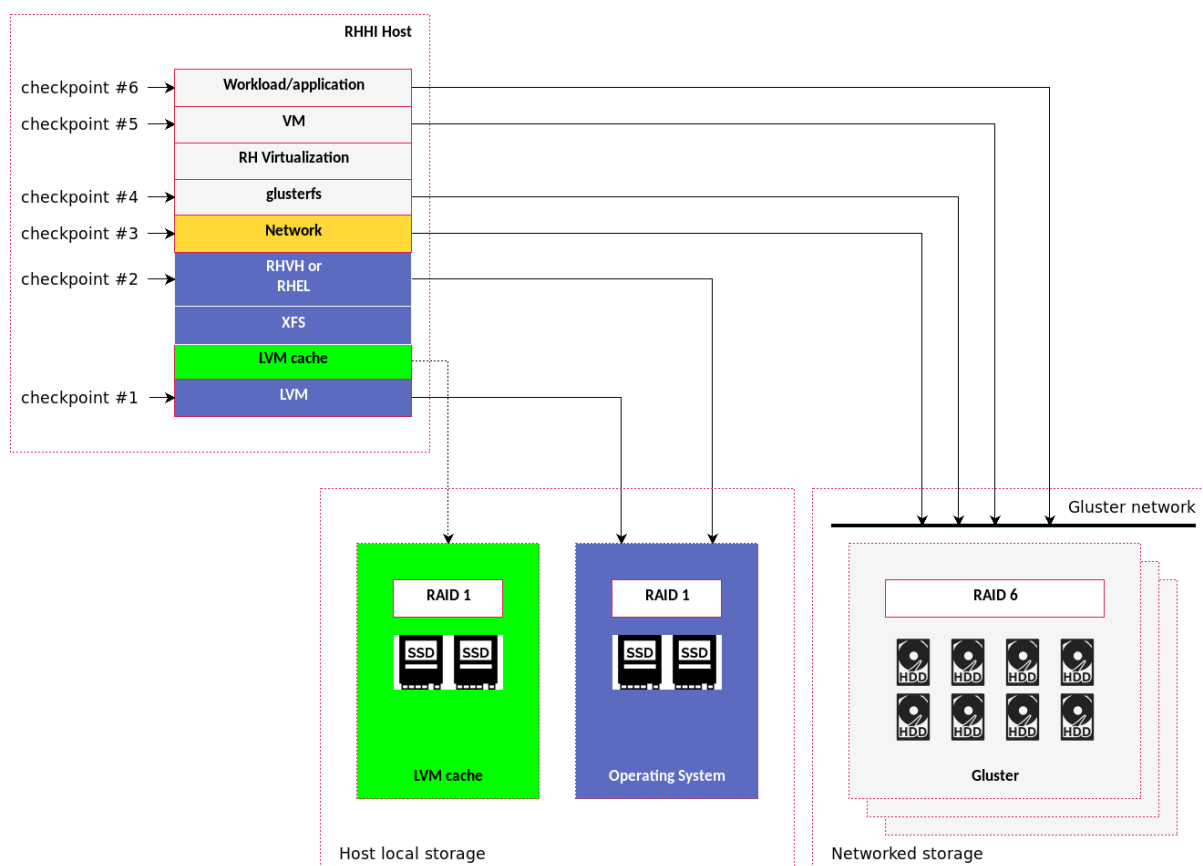


Figure 15.1: LOC Storage Performance Test Data Path

Checkpoint 0 Hard disks (SSD & HDD). This is the Physical layer.

Checkpoint 1 LVM cache. This is a virtual layer. LVM cache is optional in a deployment.

Checkpoint 2 Operating system. This is a virtual layer. All read and writes involve OS to coordinate and also some level of caching.

Checkpoint 3 Networking. This is a physical layer. Physical. This is relevant for networked storage, such as gluster, that networking bandwidth can be a limiting factor.

Checkpoint 4 Gluster file system. Virtual. RHHI-v is primarily based on the Gluster file system.

Checkpoint 5 Virtual machines. Virtual. Virtualization implementation has its own characteristics of read and write.

Checkpoint 6 Workload. Virtual. Application running inside VM. They can have a wide range of impacts upon data read and write depending on its stack, imple-

mentation, even code quality.

15.1 Performance index

By industry standard we are to seek two indexes from a test: **I/O operations per second** ↪ (aka. **IOPS**, **Transactions per second**, **TPS**) and **Mega bytes/per second** (aka. **MB/s**).

MB/s is a throughput measurement. Storage throughput is the amount of data flowing through a point in data path within a unit of measure, in this case, one second. Using an analogy of a person walking, **MB/s** is the total distance one covers within, say, one hour.

IOPS is the number of transactions a storage does in a second. In our analogy this is how many steps one takes per hour. But it doesn't say how big a step is, and that is specified by **IO size**.

In theory, the following equation holds:

$$1 \text{ MB/s} = \text{IOPs} * \text{IO block size}$$

In a person's walk, the larger each step is, the harder it usually becomes, thus we would expect a drop in the number of steps s/he can take in the same one hour period — a decreasing **IOPS** number. But since they are not in inverse linearly, the multiplication of the two — **MB/s** (the total distance s/he covers by varying step size) — may go up or down.

In general, the higher the throughput, the better the performance. In our test we will enumerate **IO size** (aka. **block size**) from **4K bytes** up to **16M bytes** with a **4K bytes** increment, and collected **IOPS** and **MB/s** values for 3,999 test points.

15.2 Type of tests

Fundamentally there are only two ways to interact w/ any storage — read, and write. Mixing these two basic operations, we get six storage access patterns, thus six types of tests:

1. **sequential read:** is a disk access pattern whereby contiguous blocks of data are read from adjacent locations on the surface of a device.

This is a common pattern for how large files are stored and thus retrieved. This is especially true for spinning disks where seeking a new sector is expensive compared to accessing an adjacent block.

2. **sequential write:** is the same pattern as [sequential read](#), but data is flowing to storage and saved to adjacent locations.

A variation of this pattern is the [sustained sequential write](#) which is the average of sequential write measurement over a 60 second period.

3. **sequential mixed read and write:** There is hardly anything that only writes or only reads. In reality it is a mix of the two. If 100% represents the total read and write, a divide, such as 70-30, is used so that the pattern has 70% sequential reads and 30% sequential writes.

4. **random read:** is a disk access pattern whereby blocks of data are read in a non-contiguous manner.

For electromechanical storage devices, its [seek time](#) is the determining factor. For SSD devices, since there isn't a spinning disk anymore, it is determined by the storage device's internal controller and memory interface speeds.

5. **random write:** the opposite of [random read](#) pattern.

6. **random mixed read and write:** similar to [sequential mixed read and write](#) but reads and writes are now in a non-contiguous manner.

15.3 Test Setup

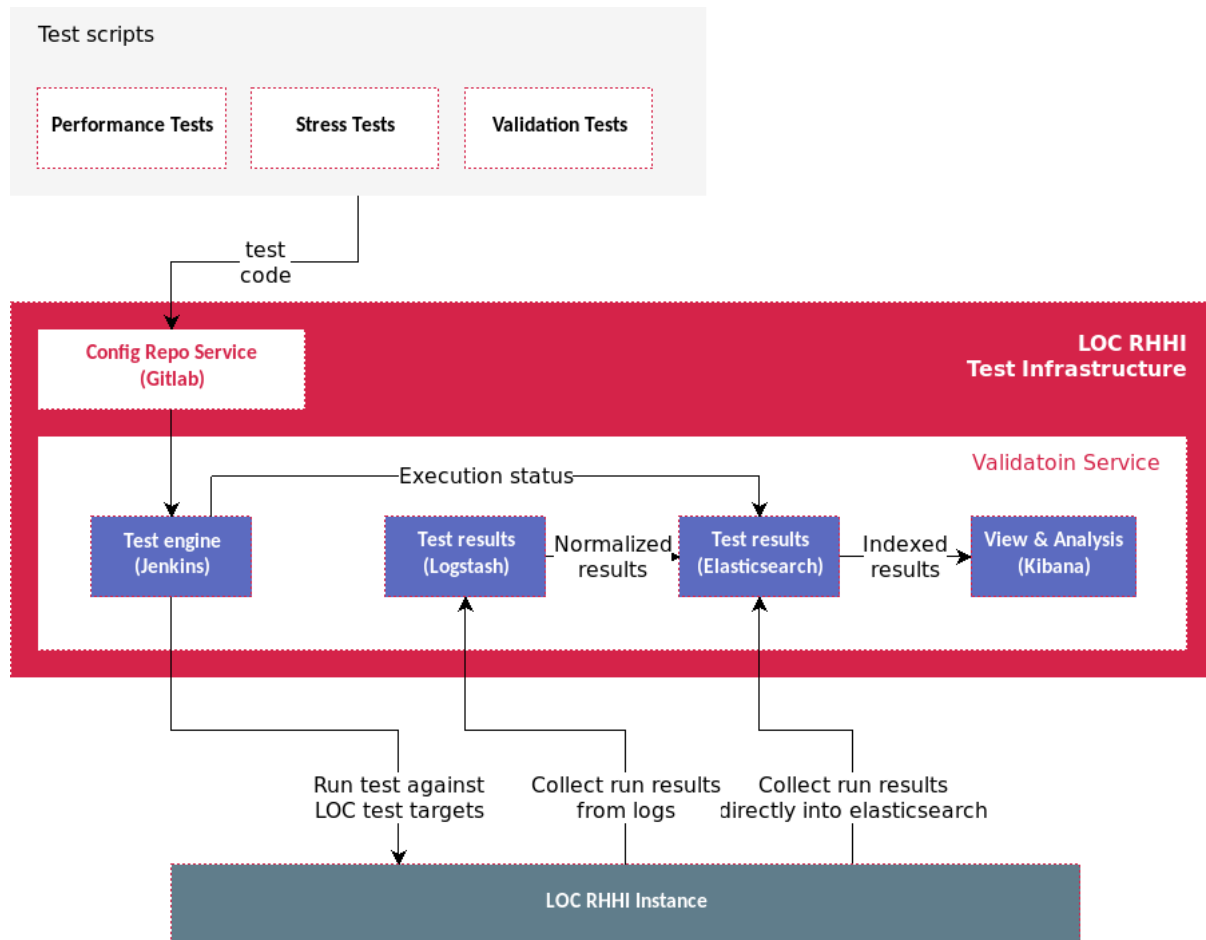


Figure 15.2: LOC storage performance test setup

All six types of test can be viewed in four steps:

1. write test code,
2. execute test run
3. collect test results, and
4. analyze results

Around these steps, we have adopted the following tools to build a test facility that is powerful, robust, and flexible.

15.3.1 Gitlab CE

We believe test codes are code and should be viewed and handled as software development. One can use any version control tool for this purpose. Among them we choose [Gitlab CE](#) so to have access to all [git](#) commands, support of code review, merge request, issue tracking, and integration with CI/CD.

15.3.2 PTS

There are many test tools for storage testing: [IOmeter](#), [IOzone](#), [FIO](#), [sysbench](#), [Geekbench](#), [byte-unixbench](#), [Phoronix Test Suite \(aka. PTS\)](#), [ioping](#), [Google PerfKitBenchmarker](#), and others.

In this setup we will use PTS. PTS is a comprehensive testing and benchmarking platform available that provides an extensible framework for which new tests can be easily added. It groups many of the popular test tools into a [suite](#).

A test suite can be a collection of test profiles with a given set of test options and/or it can also be a collection of other test suites to run, in a fully extensible manner. New test suites can be created by running `phoronix-test-suite build-suite` or from the Phoromatic web interface.

Under the [Disk Test Suite \(pts/disk\)](#), one will find:

```
1 pts/compress - gzip
2 pts/sqlite
3 pts/apache
4 pts/pgbench
5 pts/compilebench
6 pts/iozone
7 pts/dbench
8 pts/fs - mark
9 pts/fio
10 pts/tiobench
11 pts/postmark
12 pts/aio - stress
13 pts/unpack - linux
```

Our tests are mainly [pts/fio](#) tests.

15.3.3 Jenkins

[Jenkins](#) offers a simple way to set up a continuous integration and continuous delivery environment for almost any combination of languages and source code repositories. While Jenkins doesn't eliminate the need to create scripts for individual steps, it does give you a faster and more robust way to integrate your entire chain of build, test, and deployment tools than you can easily build yourself.

We use Jenkins as test orchestrator. By integrating it with [gitlab CE](#), it can build a complex test pipeline from individual test cases managed by git repo.

15.3.4 ELK stack

ELK stack is a name for three individual open-source software programs: [Elasticsearch](#), [Logstash](#), and [Kibana](#).

15.3.4.1 Elasticsearch

Elasticsearch is a search engine based on the Apache's Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

In our setup, test results are piped to Elasticsearch in JSON format via its API. Data then is parsed, indexed, and ready to be queried. The strength of Elasticsearch lies in its sophisticated data reference schema and powerful SQL-like query language. Once data has been managed by Elasticsearch, one can easily compose complex query for analysis and visualization.

15.3.4.2 Logstash

Logstash is an open source data collection engine with real-time pipelining capabilities.

We use Logstash to collect LOC's system logs and service logs during test. Data are then saved in Elasticsearch.

15.3.4.3 Kibana

Kibana is an open source data visualization plugin for Elasticsearch. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data. It provides tight integration with Elasticsearch, a popular analytics and search engine, which makes Kibana the default choice for visualizing data stored in Elasticsearch.

In the following sections, we will demonstrate how we have used Kibana to analyze test results, and have included sample Elasticsearch queries.

15.4 Theoretical values

Before looking at test results, let's first examine the disk configurations of our data path. fig. 15.1 and [data configuration](#) both indicate that there are three storage configurations in LOC RHHI-v:

| Checkpoint | Type | Number | Size | RAID | Purpose |
|------------|--------------------|--------|-------|-------|------------------|
| 1 | Intel S4610 SSD | 2 | 128GB | RAID1 | LVM cache |
| 2 | Intel S4610 SSD | 2 | 800GB | RAID1 | Operating system |
| 4 | Lenovo 10k SAS HDD | 8 | 2TB | RAID6 | GlusterFS |

Intel S4610 SSD[25] disks has a published sequential read rate of up to 560 MB/s (128KB blocks) and sequential write rate of up to 340 MB/s (128KB blocks). Config-

ured in RAID1 for LVM cache and Operating system, we have a theoretical throughput of sequential read rate of up to 1120 MB/s (128KB blocks) that is double the single disk's performance, but sequential write remains up to 340 MB/s (128KB blocks).

Similarly, Lenovo 10K SAS HDD[26] disks have a published sequential read rate of up to 200 MB/s. Since we are using a RAID 6 configuration, read and write accesses will benefit, performance wise, due to the striping of data across the array. Sequential accesses will offer superior throughput. While random accesses will help distribute operations across the disks but will be hindered by the relatively high completion time needed of operations compared with SSDs.

15.5 Sequential Read Test

There are four variables we will tweak in our runs:

Table 15.2: LOC RHHI-v Sequential Read parameters

| Block size | File size | Warmup (in seconds) | Test run (in seconds) |
|-------------------------------|-----------|---------------------|-----------------------|
| start:4KB, end:16MB, step:4KB | 2 GB | 15 | 60 |

15.5.1 Checkpoints: LVM(1), GlusterFS(4), VM(5)

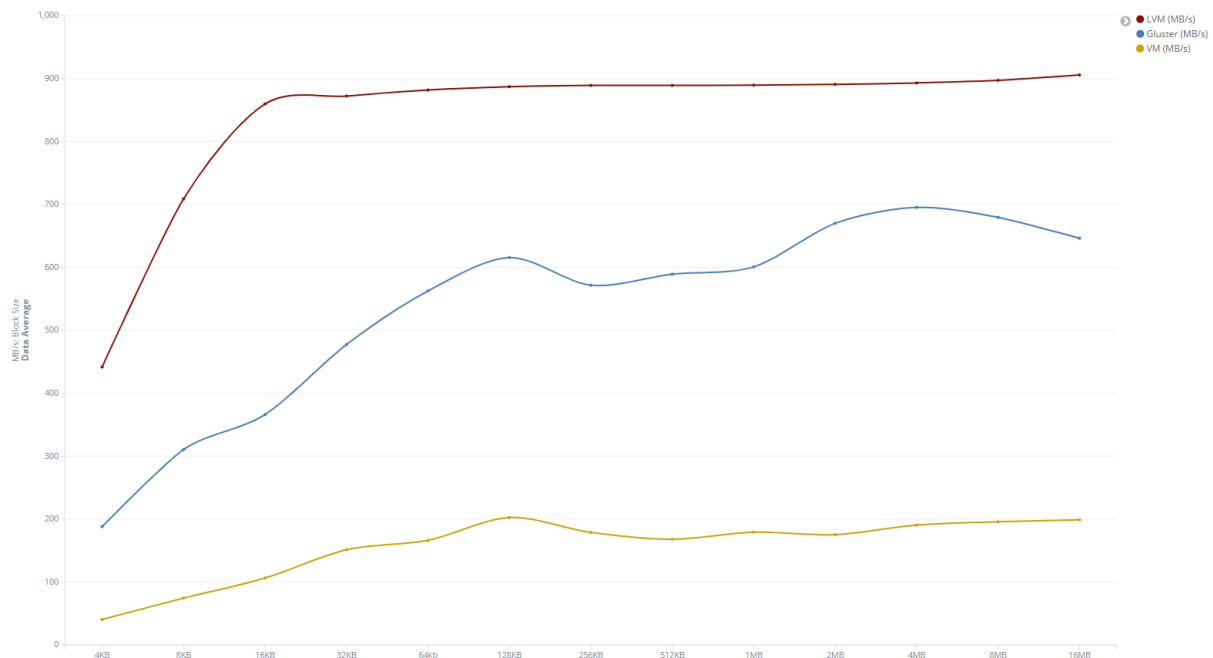


Figure 15.3: Sequential Read Throughput, checkpoints 1,4,5

We see a large increase in throughput between the 4K and 16MB block size access types across all of our checkpoints as expected.

The LVM throughput is most impressive due to the low completion latency of the SSDs. From 4KB to 16KB, the SSD RAID 1 setup climbs to its peak of 900MB/s throughput and holds steady. This value is very close to the hardware's theoretical throughput.

The gluster checkpoint shows impressive throughput results given that conventional hard drives were used. Due to the sequential nature of this specific test case, the hard drives are able to maximize the performance with a 695MB/s run at 4MB block size.

Finally, we see that the VM I/O performance is significantly lower than what is available from the hardware on top of Gluster.

There will always be a penalty due to the virtualization required for the instance to

run on. Looking at the latency of these results, we are seeing up to five times the time required for the same operation to complete on the VM as the Gluster volume itself at low (4K) block sizes. This gap closes as the block size is increased to only 3 times at 16MB.

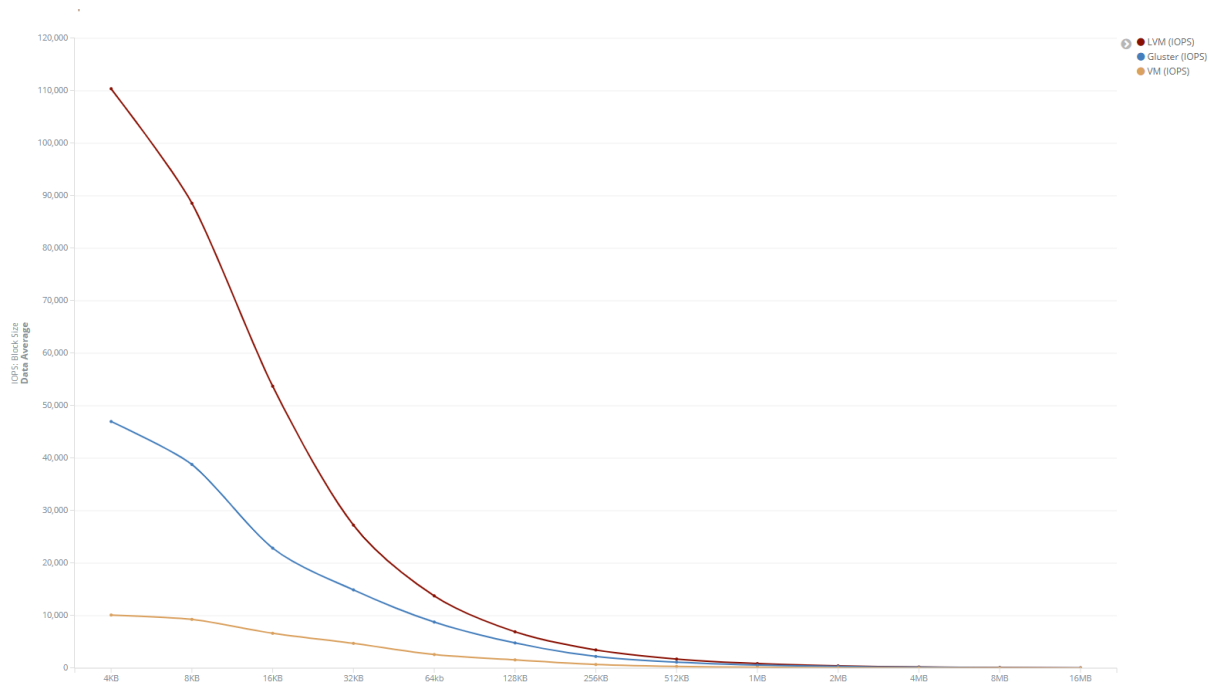


Figure 15.4: Sequential Read IOPS, checkpoints 1,4,5

For IOPS, as expected the performance is inversely proportional to the throughput comparing the same block size. Again, the LVM checkpoint offers the best I/O operations per second, followed by the GlusterFS, and finally the virtual machine.

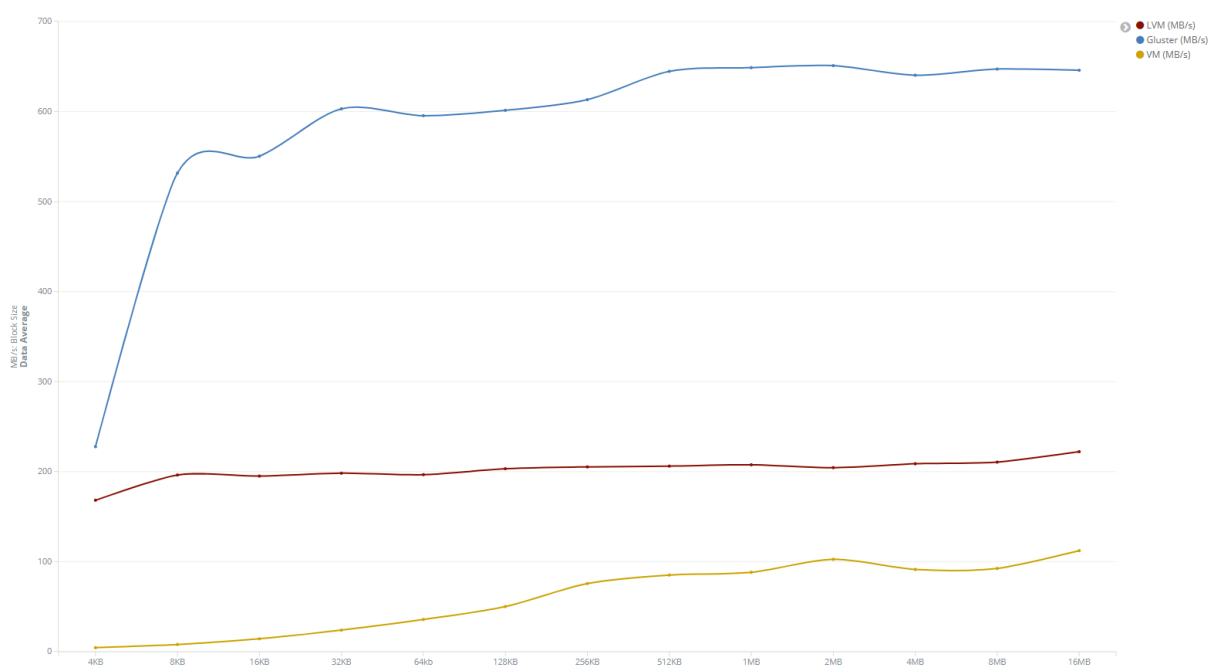
15.6 Sequential Write

There are four variables we will tweak in our runs:

Table 15.3: LOC RHHI-v Sequential Write parameters

| Block size | File size | Warmup (in seconds) | Test run (in seconds) |
|-------------------------------|-----------|---------------------|-----------------------|
| start:4KB, end:16MB, step:4KB | 2 GB | 15 | 60 |

15.6.1 Checkpoints: LVM(1), GlusterFS(4), VM(5)

**Figure 15.5:** Sequential Write Throughput, checkpoints 1,4,5

For sequential write testing, we see throughput stay consistent for our SSD at around 200 MB/s across our block size accesses. This is close to the maximum throughput available for the SSD in RAID 1 configuration.

The Gluster volume offered superior write performance due to the distributed nature of the sequential writes across the RAID array.

In this RHHI-v setup, our Gluster is configured to replicate each write to all three

Gluster hosts. Since a write will always take place local to one host, the write must be replicated to the other two hosts to maintain quorum and atomicity. This leads us to the following formula:

$$1 \text{ Outbound Network Bandwidth} = \text{Block Size} * \text{Replicas}$$

During the execution of this test on the Gluster, we noticed network throughput egress nearly 10 Gbps due to the replication. This information will be important to consider when requiring a specific total write throughput.

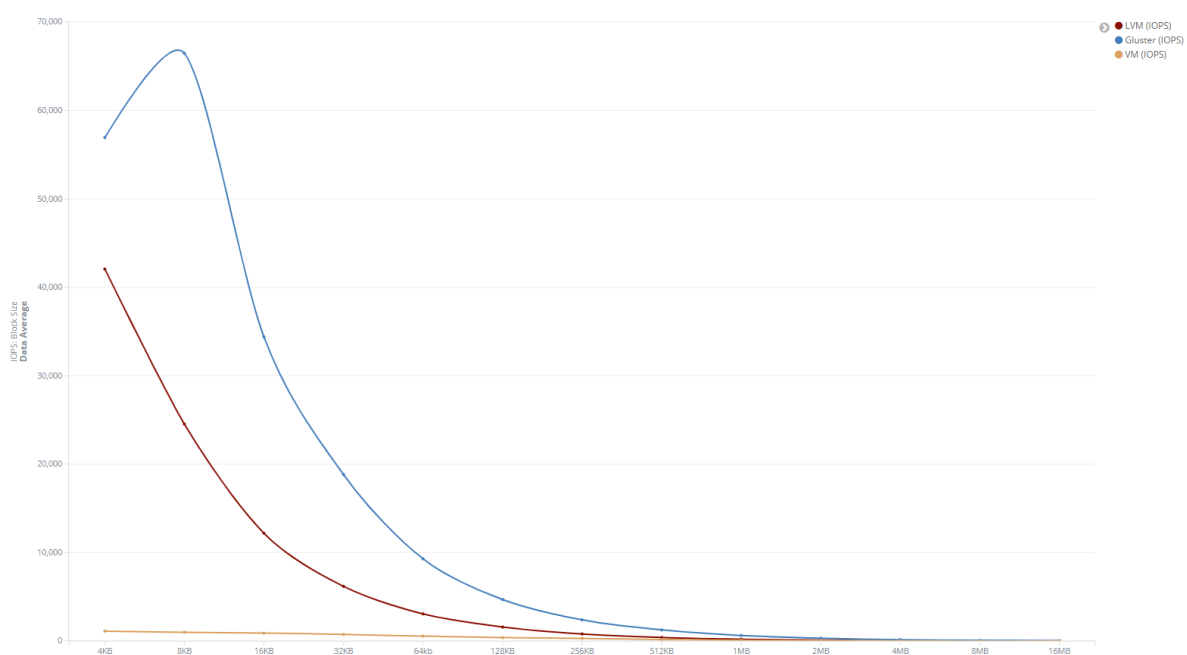


Figure 15.6: Sequential Write IOPS, checkpoints 1,4,5

For IOPS, a gap between the GlusterFS and VM checkpoints is larger using smaller block sizes. The VM checkpoint is only able to offer 4% of the performance at 4KB block size compared with GlusterFS. We see this gap close exponentially after 16KB until 1MB block size where the performance is only 50% between the two.

15.7 Sequential Mixed

100% READ or WRITE are rare. In reality all applications are a mix of reads and writes. Typical PC operations are 50% of each. In our tests, we choose 50%.

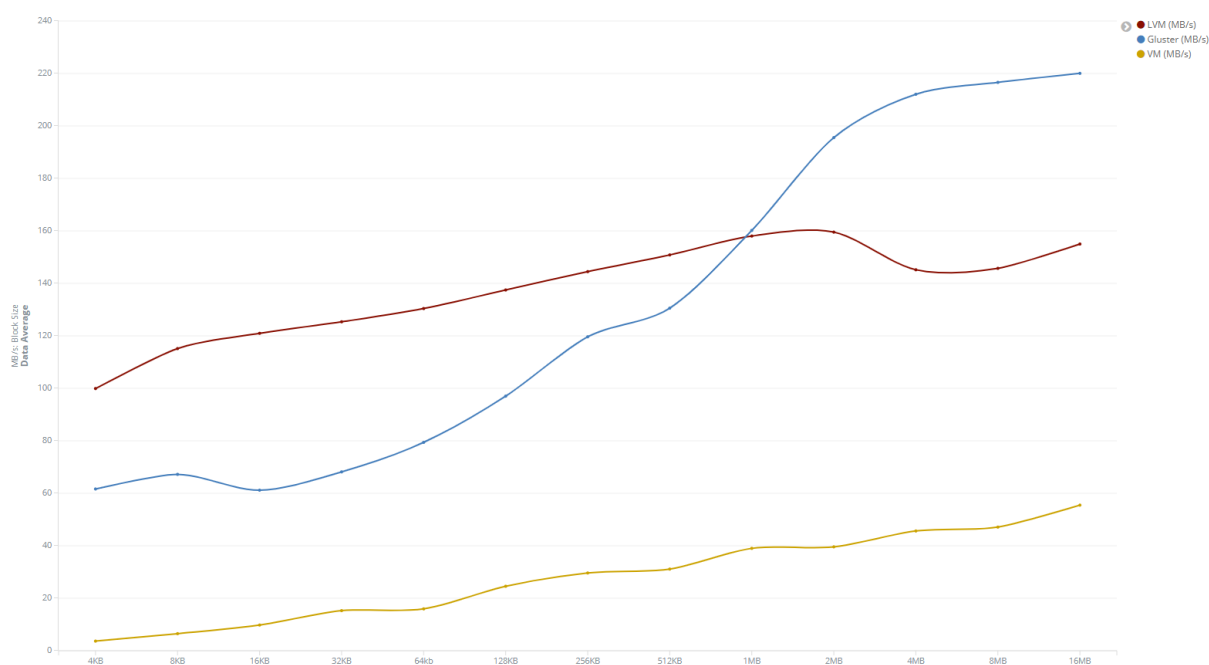
There are five variables we will tweak in our runs:

Table 15.4: LOC RHHI-v Sequential Mixed parameters

| Block size | File size | Warmup (in seconds) | Test run (in seconds) | % or READ |
|-------------------------------|-----------|---------------------|-----------------------|-----------|
| start:4KB, end:16MB, step:4KB | 2 GB | 15 | 60 | 50 |

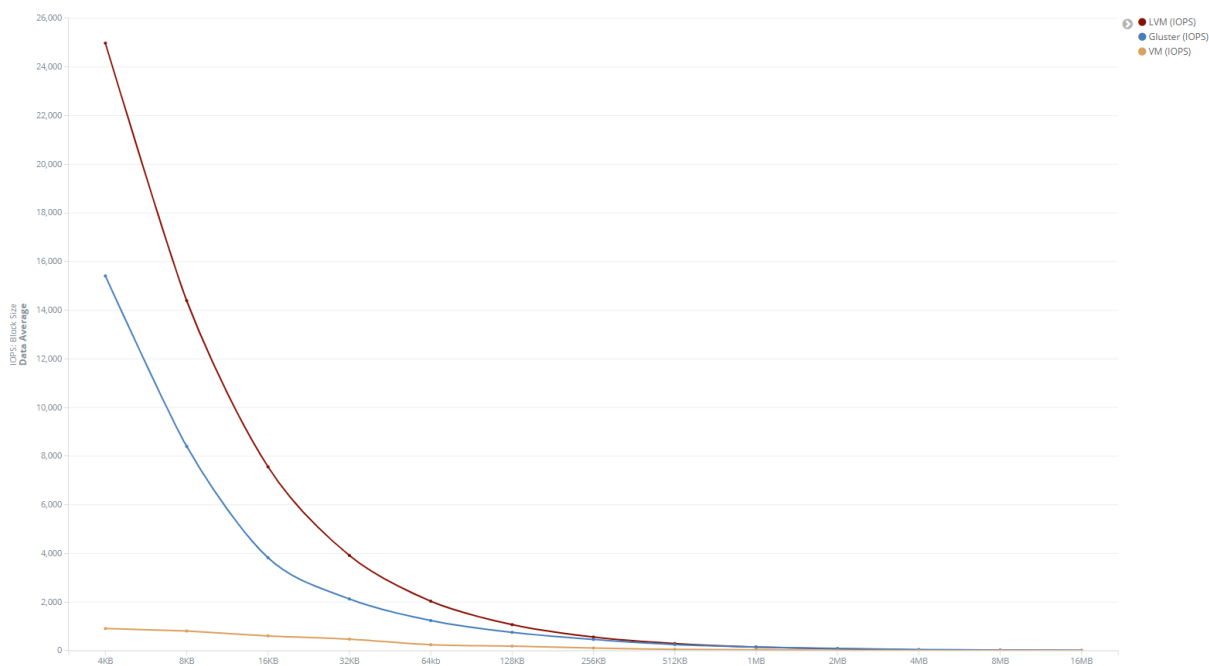
15.7.1 Checkpoints: LVM(1), GlusterFS(4), VM(5)

As can be seen, sequential mixed access workloads experience worse performance compared with full read or write access. The mixed accesses can cause caches to invalidate more aggressively due to increased variation in access methods.



For throughput, we experience a linear improvement except for a similar variation

at 16KB as seen in previous test cases.



For IOPS, the sequential mixed access test case showed inversely proportional results to the throughput.

15.8 Random Read

Random Read is a disk access pattern whereby small blocks of data are read from random locations on the surface of the device being tested.

There are four variables we will tweak in our runs:

Table 15.5: LOC RHHI-v Random Read parameters

| Block size | File size | Warmup (in seconds) | Test run (in seconds) |
|-------------------------------|-----------|---------------------|-----------------------|
| start:4KB, end:16MB, step:4KB | 2 GB | 15 | 60 |

15.8.1 Checkpoints: LVM(1), GlusterFS(4), VM(5)

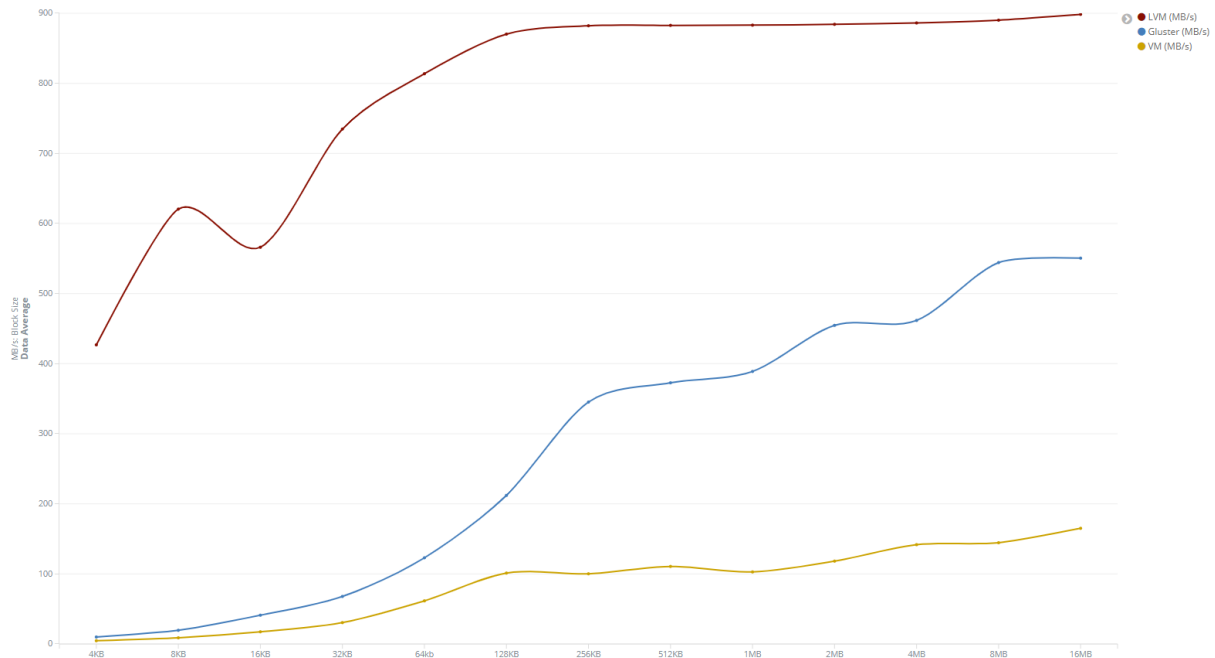


Figure 15.7: Random Read Throughput

For random read, we experience a large decrease in performance compared with sequential read for the Gluster and VM. Random accesses are less performant than sequential due to the hindrance of spatial caching as well as the inability for conventional spinning hard disks to order accesses. This leads to longer latency between operations. Often SSD caching can minimize the impact of random access operations with much less latency.

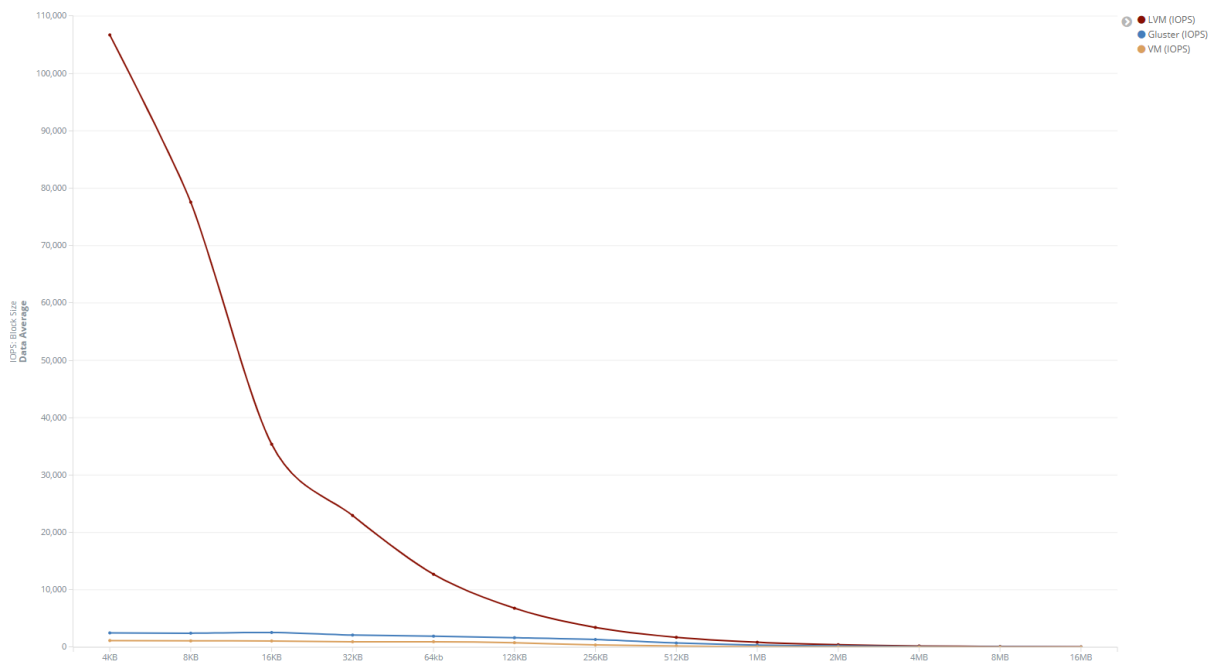


Figure 15.8: Random Read IOPS

Again, we see the SSD backed LVM checkpoint is able to offer similar performance in terms of IOPS comparing sequential and random accesses. There is only a 4% decrease in IOPS between the two on the RAID 1 SSD at a 4K block size.

The conventional hard drives in RAID 6 struggle to keep up with the performance results of the sequential workload. 95% of the IOPS performance is lost between random and sequential accesses.

15.9 Random Write

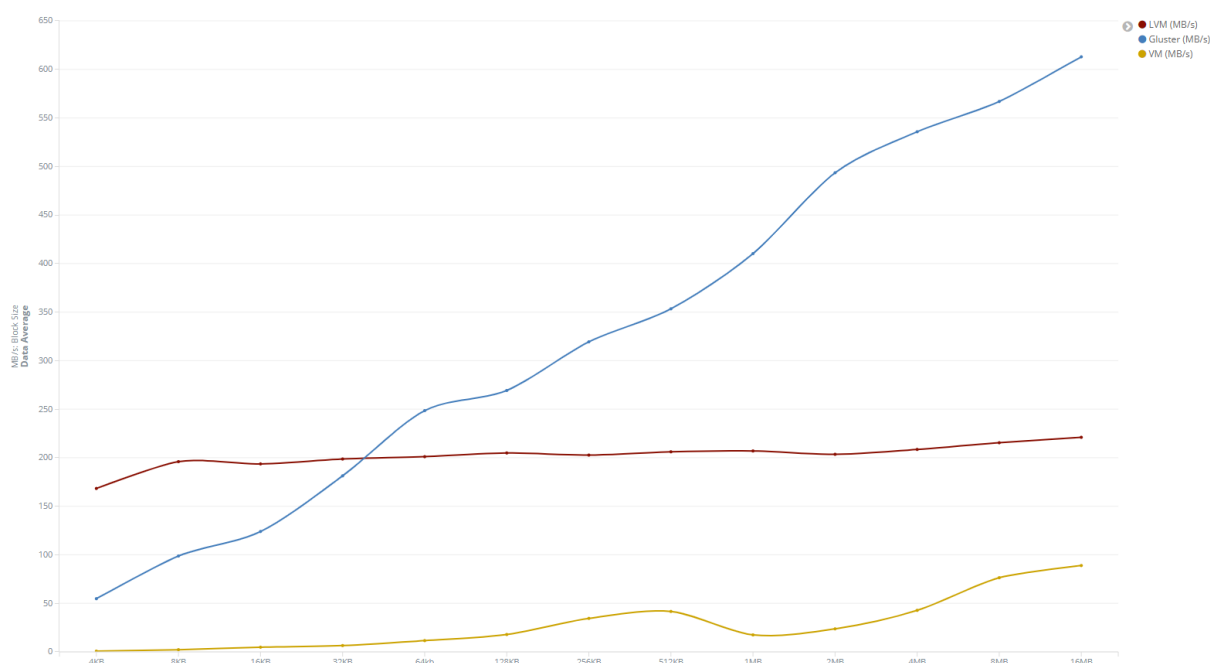
Random write is a disk access pattern whereby small blocks of data are written to random locations on the surface of a storage device.

There are four variables we will tweak in our runs:

Table 15.6: LOC RHHI-v Random Write parameters

| Block size | File size | Warmup (in seconds) | Test run (in seconds) |
|-------------------------------|-----------|---------------------|-----------------------|
| start:4KB, end:16MB, step:4KB | 2 GB | 15 | 60 |

15.9.1 Checkpoints: LVM(1), GlusterFS(4), VM(5)

**Figure 15.9:** Random Write Throughput

Again, the GlusterFS checkpoints are limited by the latency added by the physical rotation of the conventional hard drive mechanics. As the hard drive arm spins around, it cannot efficiently load data when the accesses are scattered across the platter surface.

As block size increases, we do see less of a performance loss as more sequential data can be loaded at a time from the storage.

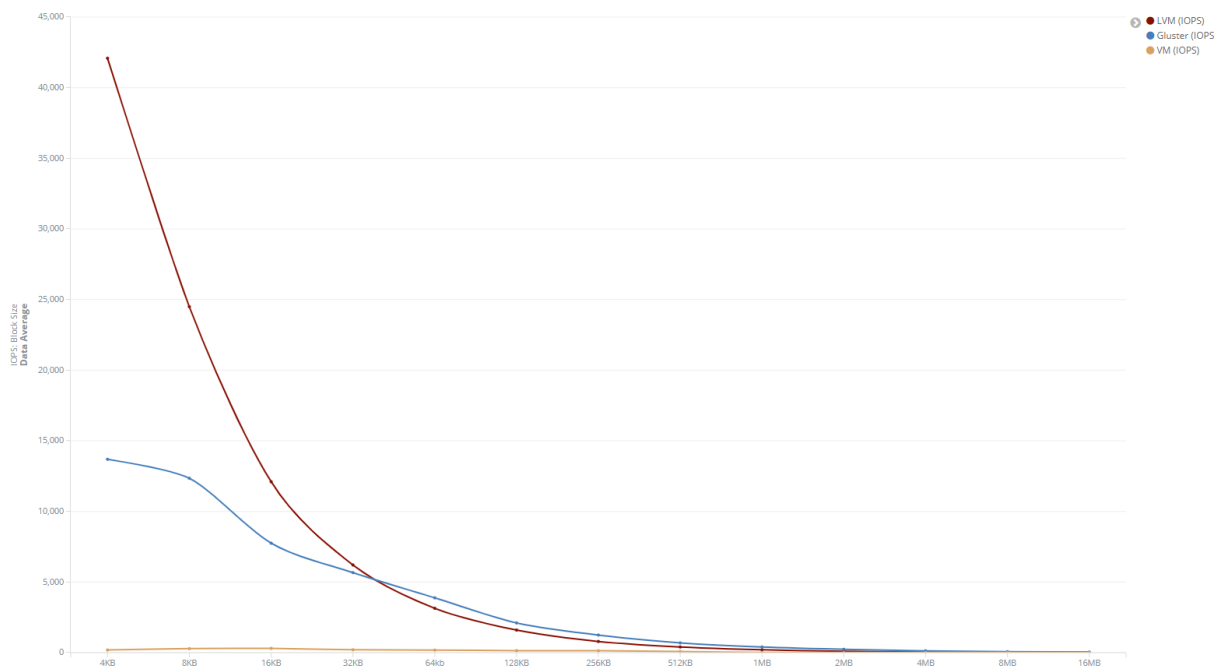


Figure 15.10: Random Write IOPS

The IOPS shows a similar graph. The SSDs are able to offer consistent performance when compared with sequential write accesses. The results for sequential write and random write accesses for the RAID 1 SSDs, we found to be nearly identical.

15.10 Random Mixed R/W

In real world applications a significant portion of workloads are mixed, meaning that there are both read and write IOs.

Table 15.7: LOC RHHI-v Random Mixed parameters

| Block size | File size | Warmup (in seconds) | Test run (in seconds) | % or READ |
|-------------------------------|-----------|---------------------|-----------------------|-----------|
| start:4KB, end:16MB, step:4KB | 2 GB | 15 | 60 | 50 |

15.10.1 Checkpoints: LVM(1), GlusterFS(4), VM(5)

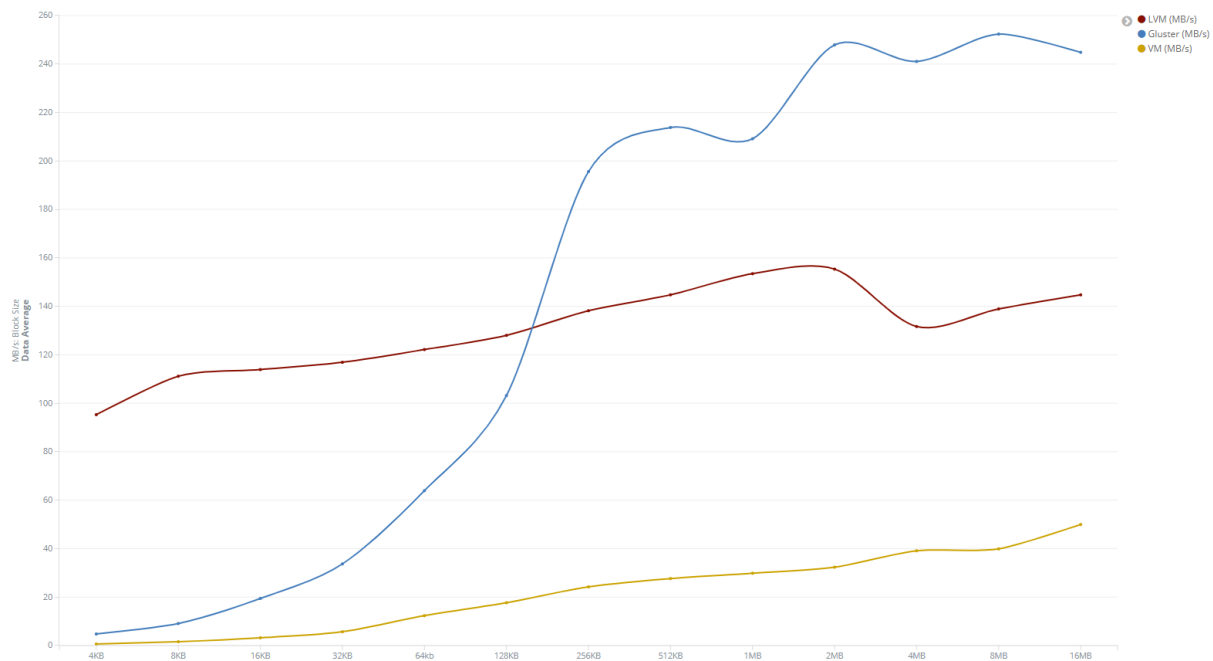


Figure 15.11: Random Mixed Throughput

This access type represents the worst performing of the six. We see very low throughput available from the VM. Latency is high due to the in-ability to efficiently perform read and writes.

Again, we notice that the SSD layers do not show any loss in performance due to the random access. The conventional hard drive checkpoints do at block sizes below 256KB.

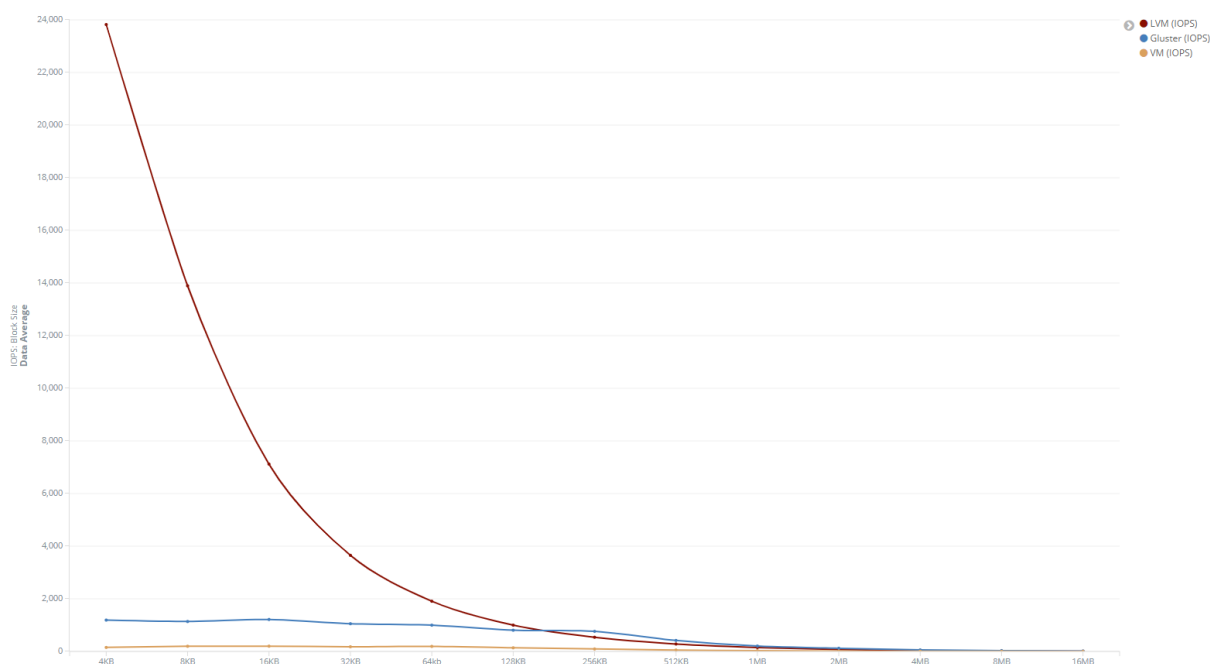


Figure 15.12: Random Mixed IOPS

Finally, the random mixed test cases show the inverse relation compared with the results of the throughput. Again, we see consistency of performance with the SSDs across the types of accesses.

15.11 Conclusions

There is not “the best performance” in an absolute sense. Characteristics of the workload ultimately determines the priority of one performance over another. Along the data path many dials can then be experimented in order to find that sweet spot, whatever it may be:

1. at block level: SSD vs. HDD, or even NVMe
2. at RAID level: different RAID levels, strip size, write/read policy.
3. at logical volumes: enable LVM cache, block alignment w/ underline storage.
4. at GlusterFS: volume type (religated vs. not-replicated), and many [tuning options][27].

Not to mention the importance of the proper networking requirement to satisfy distributed storage solution including GlusterFS and NFS. We recommend the **networking design convention** as the best practice in this regard so data traffic is both secure and efficient. Lenovo hardware and LOC RHHI-v design have been carefully tuned to create a sound baseline from which user can further explore possibilities. These test results, therefore, should not be taken as the end, but the start of a workload-oriented, user-focused tuning process.

By following the test methodologies demonstrated in this section, and using the testing infrastructure shown here, one can establish his own test cases and seek for his “optimal” combination through iteration with little overhead.

16

References

- [1] Red Hat, “Red hat hyperconverged infrastructure.” [Online]. Available: <https://access.redhat.com/products/red-hat-hyperconverged-infrastructure>
- [2] Red Hat, “Product documentation for red hat hyperconverged infrastructure for virtualization 1.5.” [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_hyperconverged_infrastructure_for_virtualization/1.5/
- [3] Lenovo, “Lenovo thinksystem sr650 server (xeon sp gen 1): Product guide.” [Online]. Available: <https://lenovopress.com/lp0644-thinksystem-sr650-server-xeon-sp-gen1>
- [4] Lenovo, “Lenovo thinksystem raid 930 series internal raid adapters: Product guide.” [Online]. Available: <https://lenovopress.com/lp0652-thinksystem-raid-930-series-internal-raid-adapters>
- [5] Lenovo, “Lenovo rackswitch g8272: Product guide.” [Online]. Available: <https://lenovopress.com/tips1267-lenovo-rackswitch-g8272>
- [6] Lenovo, “Lenovo rackswitch g8052: Product guide.” [Online]. Available: <https://lenovopress.com/tips1270-lenovo-rackswitch-g8052>
- [7] Lenovo, “Lenovo thinksystem ne2572 rackswitch: Product guide.” [Online]. Available: <https://lenovopress.com/lp0608-lenovo-thinksystem-ne2572-rackswitch>
- [8] Red Hat, “RED hat virtualization 4.2: PRODUCT guide.” [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.2/html/product_guide/index
- [9] Red Hat, “Product documentation for red hat gluster storage 3.4.” [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_gluster_stora

ge/3.4/

- [10] Red Hat, Available: https://access.redhat.com/webassets/avalon/d/Red_Hat_Hyperconverged_Infrastructure_for_Virtualization-1.5-Deploying_Red_Hat_Hyperconverged_Infrastructure_for_Virtualization-en-US/images/4a9666b1f646ebbee25951430518a999/rhhi-architecture.png
- [11] Red Hat, "RED hat virtualization 4.1 product guide: CHAPTER 1. INTRODUCTION to red hat virtualization." [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.1/html/product_guide/introduction
- [12] Red Hat, "RED hat virtualization 4.2 planning and prerequisites guide: CHAPTER 1. RED hat virtualization architecture." [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.2/html/planning_and_prerequisites_guide/rhv_architecture
- [13] Red Hat, Available: https://access.redhat.com/webassets/avalon/d/Red_Hat_Virtualization-4.2-Product_Guide-en-US/images/894aaa576d2f26123a3b3149a5e18159/RHV_SHE_ARCHITECTURE1.png
- [14] Red Hat, "RED hat gluster storage 3.4 installation guide: CHAPTER 1. PLANNING red hat gluster storage installation." [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.4/html/installation_guide/chap-planning_red_hat_storage_installation
- [15] Red Hat, Available: https://access.redhat.com/webassets/avalon/d/Red_Hat_SStorage-3.1-Administration_Guide-en-US/images/667b8206666b18dbed70d300c3e2710c/RH_Gluster_Storage_diagrams_334434_0415_JCS_5.png
- [16] Red Hat, "RED hat enterprise linux 6 deployment guide: 11.2.4. CHANNEL bonding interfaces." [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/s2-networkscripts-interfaces-chan
- [17] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address allocation for private internets." Feb-1996 [Online]. Available: <https://tools.ietf.org/html/rfc1918>
- [18] F. Xia, L. Gu, J. Bryant, and M. Halas, "Lenovo open cloud reference architecture,"

May 2019 [Online]. Available: <https://lenovopress.com/lp1149>

[19] “Setting up glusterfs volumes.” [Online]. Available: <https://docs.gluster.org/en/latest/Administrator%20Guide/Setting%20Up%20Volumes/>

[20] “Chapter 8: Storage.” [Online]. Available: <https://www.ovirt.org/documentation/admin-guide/chap-Storage.html>

[21] Red Hat, “RED hat gluster storage 3.4 administration guide: 21.8. LVM cache for red hat gluster storage.” [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.4/html/administration_guide/sect-lvm_cache

[22] Red Hat, “RED hat enterprise linux 7 networking guide: CHAPTER 7. CONFIGURE network bonding.” [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/ch-configure_network_bonding#sec-Bond-Understanding_the_Default_Behavior_of_Master_and_Slave_Interfaces

[23] T. L. F. Wiki, “Bonding.” [Online]. Available: <https://wiki.linuxfoundation.org/networking/bonding>

[24] Red Hat, “RED hat enterprise linux 7 networking guide: 7.6. USING channel bonding.” [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/sec-using_channel_bonding

[25] Lenovo, “ThinkSystem intel s4610 mainstream sata 6Gb ssds: Product guide.” [Online]. Available: <https://lenovopress.com/lp0925-intel-s4610-mainstream-sata-6gb-ssd>

[26] Lenovo, “Lenovo thinksystem hdd summary: Reference information.” [Online]. Available: <https://lenovopress.com/lp1059-lenovo-thinksystem-hdd-summary>

[27] “Managing glusterfs volumes.” [Online]. Available: <https://docs.gluster.org/en/v3/Administrator%20Guide/Managing%20Volumes/#tuning-options>