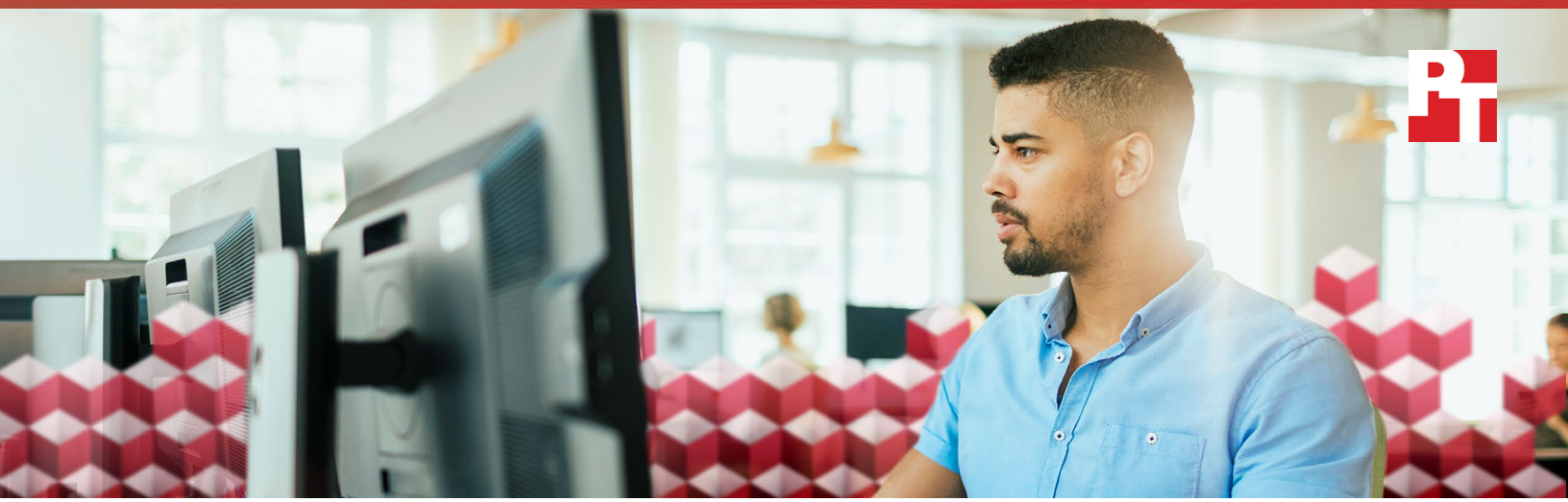


Performance of persistent apps on Container-Native Storage for Red Hat OpenShift Container Platform

Introduction

Companies are constantly seeking ways to improve on application and services delivery while using flexible technologies at the service, application, and storage layers. Open-source software-defined storage (SDS) and container technologies are evolving to enable this. Meanwhile, companies are searching for ways to use new, advanced solid-state storage hardware while also leveraging their investments in both licensing and expertise on other enterprise technologies, such as VMware® vSphere®.

The Red Hat OpenShift Container Platform is a container application platform that brings containers and Kubernetes® to the enterprise. Container-Native Storage (CNS) for Red Hat® OpenShift Container Platform is software-defined storage built upon Red Hat Gluster Storage. It provides a platform-agnostic storage layer for persistent applications running on OpenShift. With this integrated solution, Red Hat aims to provide a flexible and scalable solution for companies looking to adopt a container-based platform for apps with persistent data needs. By virtualizing CNS and OpenShift on VMware vSphere, companies can avoid having to abandon existing virtualization technologies. In this paper, we discuss results of benchmarking several web application workloads with this Container-Native Storage solution. We also demonstrate performance and sizing using two Western Digital® storage offerings: the Ultrastar® SS200 solid-state drive (SSD) and Ultrastar He[10] hard disk drive (HDD).

Executive summary of testing

Test matrix

We tested the Container-Native Storage for Red Hat OpenShift Container Platform solution in de-coupled mode running on Red Hat® Enterprise Linux® (RHEL) Atomic Host VMs in a VMware cluster for two hardware configurations that differ in their persistent-storage media:

- Ultrastar SS200 solid-state drives
- Ultrastar He^{10} hard drives

On each configuration, we scaled many application instances running in OpenShift containers, and executed two test workloads, which stressed different system components of the containers:

- IO-intensive workload against a MySQL database
- CPU-intensive workload against a Linux, Apache, MySQL, and PHP (LAMP) stack application

For the IO-intensive workload, we tested two CNS-node vCPU allocations.

These choices yielded six permutations of test conditions. For each, we first tested with a single app instance and then scaled to many app instances, stopping when app performance declined or app response time became unacceptably long.

Acronyms we use in this report

CNS	Container-Native Storage for Red Hat OpenShift Container Platform
CRS	container-ready storage
DS2	DVD Store 2
DS3	DVD Store 3
HDD	hard disk drive
LAMP	Linux, Apache, MySQL, and PHP/Python/Perl
OCI	Open Container Initiative
OLTP	online transaction processing
OPM	orders per minute
SDLC	software development lifecycle
SDS	software-defined storage
SSD	solid-state drive
VM	virtual machine

Findings

- CNS operating in de-coupled mode successfully provided storage to the container-based apps in both the IO-intensive and CPU-intensive scenarios.
- Understanding your application's performance profile is critical to the design of your CNS configuration. For example, IO-intensive apps backed by CNS may need additional CPU resources allocated to the CNS nodes. CPU-intensive apps may perform evenly regardless of whether the backing storage uses hard drives or solid-state drives.
- CNS provided scalable storage to our container-based apps in several hardware and workload configurations, up to 128 app instances.
- On the IO-intensive, database-only workload, the Ultrastar SS200 solid-state drive configuration achieved maximum performance greater than five times that of the He^{10} hard drive configuration, showing that CNS on the solid-state drive was better suited to handle the IO-intensive workload.
- On the CPU-intensive, full-web app LAMP stack workload, the two configurations achieved similar maximum performance levels, with the solid-state drive configuration outperforming the hard drive configuration by less than 5 percent.
- The modest improvement the solid-state storage delivered over the hard drive configuration on the CPU-intensive workload illustrates the importance of understanding your workload when deciding among hardware options.
- Depending on a business's needs, either storage option can be a good choice. Because the total cost of the solutions tested (including hardware and software) differs by only 1.24 percent, choosing the solid-state drive configuration can offer added value.

We worked with Red Hat and Western Digital Data Propulsion Lab® Technologists to configure the test bed and design the testing approach. For details on the roles the partners played and who executed each part of the study, see Appendix B.

Our two-phase approach

Our testing consisted of two phases. The first used an IO-intensive app scenario and the second used a CPU-intensive LAMP stack app scenario. For our test workload, we selected DVD Store, an online transaction processing (OLTP) database workload generator. DVD Store, which is part of the VMware VMmark® benchmark suite, is available in two versions, DVD Store 2 (DS2) and DVD Store 3 (DS3). Both versions model real-world, human-interactive applications. The versions have different database schemas and queries, resulting in a CPU-intensive workload profile for DS3 and an IO-intensive workload profile for DS2. We measured performance in the IO-intensive scenario using only the database tier in DS2, and in the CPU-intensive scenario using the full LAMP stack for DS3. We provide more detail about the two versions in the Workload discussion section.

In each scenario, we compared how well the solution executed each workload profile when we used two different types of storage media, Ultrastar SS200 solid-state drives and Ultrastar He[10] hard drives.

To summarize, we tested the following:

- Phase 1: IO-intensive, database-only workload
  - Phase 1a: 4 vCPUs per CNS node
  - Phase 1b: 16 vCPUs per CNS node
- Phase 2: CPU-intensive, full-web app LAMP stack workload (16 vCPUs per CNS node)

The following performance and sizing questions guided our test design:

- In both the IO-intensive scenario and the CPU-intensive scenario, how did total workload performance scale as the number of app and workload instances increased?
- Where did bottlenecks occur? How did cluster saturation manifest itself when additional instances caused response times to increase (e.g., CPU pressure, RAM pressure, NIC pressure, disk IO saturation, etc.)?
- How did performance and scalability change when additional resources eliminated the bottleneck?
- How many database app instances (scenario 1) or LAMP app instances (scenario 2) could we run on OpenShift Container Ready Storage backed by three nodes, with either 12 Ultrastar He[10] hard drives per node or five Ultrastar SS200 solid-state drives per node?

Western Digital drive technologies

SSD technology

We used HGST Ultrastar SS200 solid-state drives. According to Western Digital, these enterprise-grade 12Gb/s SAS drives have a maximum read throughput up to 1,800MB/s and a capacity of 7.68 TB in a 2.5-inch small form factor. Western Digital positions the drives as a high-endurance option well suited to mixed-use and read-intensive workloads.

To learn more, see the Ultrastar SS2200 data sheet at https://www.hgst.com/sites/default/files/resources/Ultrastar-SS200-datasheet.pdf

HDD technology

We used HGST Ultrastar He[10] hard disk drives, with 10 TB of capacity in a 3.5-inch form factor. According to Western Digital, these drives offer greater capacity, power efficiency, and reliability than 8TB air drives do, and are appropriate for enterprise and data center applications that demand high capacity density. They offer both SATA 6Gb/s and SAS 12Gb/s options. In our testing, we used SATA drives.

To learn more, see the Ultrastar He[10] data sheet at https://www.hgst.com/sites/default/files/resources/Ultrastar-He10-DS.pdf

Container-Native Storage for Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform is an offering based on Kubernetes that provides automation, orchestration, and self-service for application developers. OpenShift integrates several open-source technologies into an enterprise-class solution that can deploy and manage many applications. These technologies include the following:

- Linux® containers, Docker, and Kubernetes for cluster management and automation
- Open vSwitch® software-defined networking for container communication within the cluster
- Red Hat CloudForms, which gives infrastructure and operations teams a unified management experience across containers and the underlying bare-metal, virtual, or cloud infrastructure

The platform also offers a range of storage options for persistent application data and an internal container registry to store and manage container images for deployment. For example, application and data center administrators can create reproducible workflows that are designed to create efficiencies in the software development lifecycle (SDLC) and accelerate application delivery.

Scaling through automation

By automating many processes, OpenShift makes it practical to implement containers in a large-scale environment. OpenShift software-defined storage and networking capabilities support many types of cloud-native and stateful apps and make it possible to connect these applications.

OpenShift containers

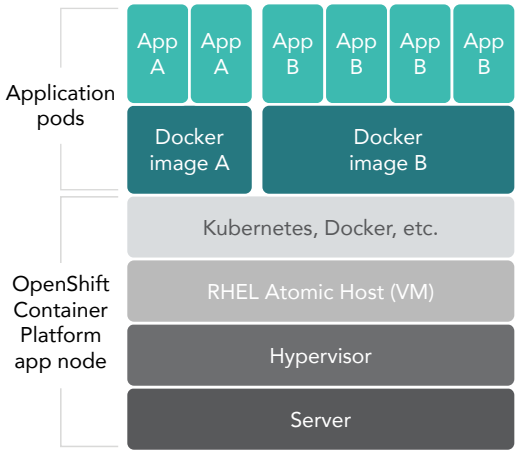
Containers bundle software applications with all the OS resources they need to run (e.g., application dependencies on specific versions of system libraries). This ensures consistent app behavior across operating environments. The OpenShift Container Platform uses an Open Container Initiative (OCI)-compatible runtime for container execution.

Docker

Docker is an open-source containerization technology that adds flexibility to the creation and deployment of Linux containers. Docker lets you create, deploy, and copy containers.

Kubernetes

Kubernetes is a container orchestration solution. OpenShift, the Red Hat container application platform offering built on top of Kubernetes, can extend and enhance Kubernetes functionality by providing networking, routing, scaling, health checking, persistent storage connection, and other features that enable enterprise-scale implementation of containers.



Container-Native Storage for Red Hat OpenShift Container Platform

Container-Native Storage for Red Hat OpenShift Container Platform is a software-defined storage solution for container environments built on Red Hat Gluster Storage that pools and presents direct-attached storage for use as a distributed file system. CNS features include data redundancy, tiering, read-only and read-write snapshots, and—as we show in this paper—strong integration with the Red Hat OpenShift Container Platform. CNS can run on physical machines, virtual machines, containers, or in a cloud environment. CNS can serve as a persistent backing store for applications—such as database environments, as we show in this paper—and can run co-located with applications co-residing on the same nodes. It can also provide storage for unstructured data such as VM images, backup images, and video.

CNS operating in de-coupled mode

CNS can run in either full CNS mode or de-coupled mode. The primary difference between full CNS and de-coupled modes is where the storage services are running. In full CNS mode, the storage services run co-located with applications on the same OpenShift-managed nodes. In de-coupled mode, the storage services run on separate nodes (virtual or bare metal). Our testing used de-coupled mode for all configurations. The OpenShift app developer experience is the same with CNS configured in either mode (full CNS or de-coupled). In either mode, persistent storage is dynamically provisioned by OpenShift when a persistent volume claim is submitted.

Test bed configuration

Western Digital assembled hardware to create a common configuration that would support the goal of investigating Container-Native Storage for Red Hat OpenShift Container Platform performance, scaling, and sizing and showcase the capacity of the two Ultrastar storage offerings.

Hardware details

The hardware configuration we used for our testing consisted of nine physical servers. Three had Ultrastar He[10] hard drives and six had Ultrastar SS200 solid-state drives. All servers had a single additional SSD used for the OS (OS SSD). The placement of virtual machines varied based on which configuration we tested, per the diagrams on the next two pages.

In each configuration, for testing and pricing, we deployed the VMs for OpenShift components and test clients only on the OS SSD, and not on either the five SSDs or the 12 HDDs. We used these latter drives only for CNS pooled storage. The CNS server specifications were as follows:

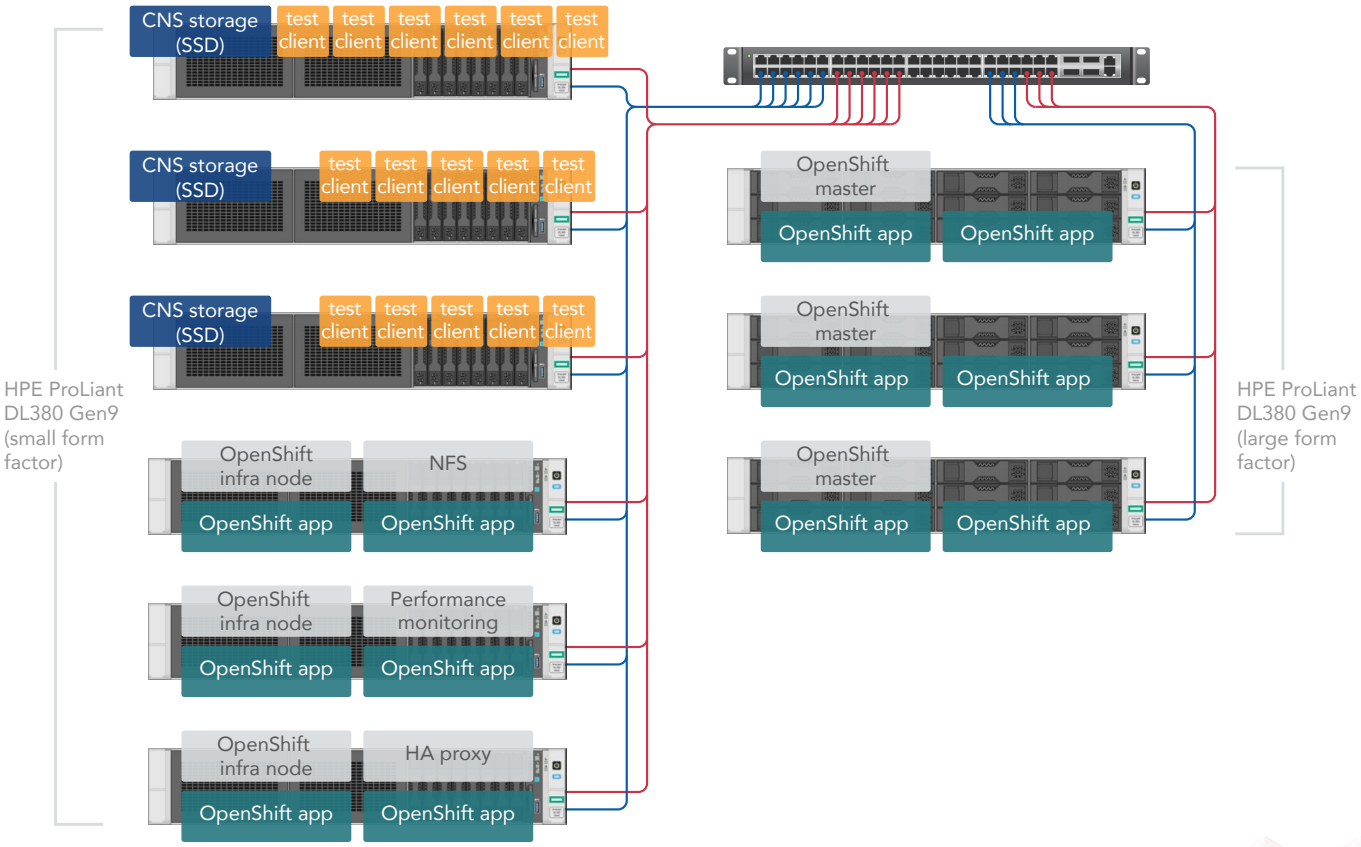
- Three HPE™ ProLiant DL380 Gen9 small form factor (SFF), each with
  - Two Intel® Xeon® E5-2697A v4 processors
  - 256 GB of RAM
  - Five Ultrastar SS200 solid-state drives
  - HPE SmartHBA H240ar
- Three HPE ProLiant DL380 Gen9 large form factor (LFF), each with
  - Two Intel Xeon E5-2697 v4 processors
  - 256 GB of RAM
  - 12 Ultrastar He[10] hard drives
  - HPE SmartArray P440ar
  - HPE SmartHBA H240ar
  - One RAID5 volume containing the 12 disks

Virtual machine details and topology

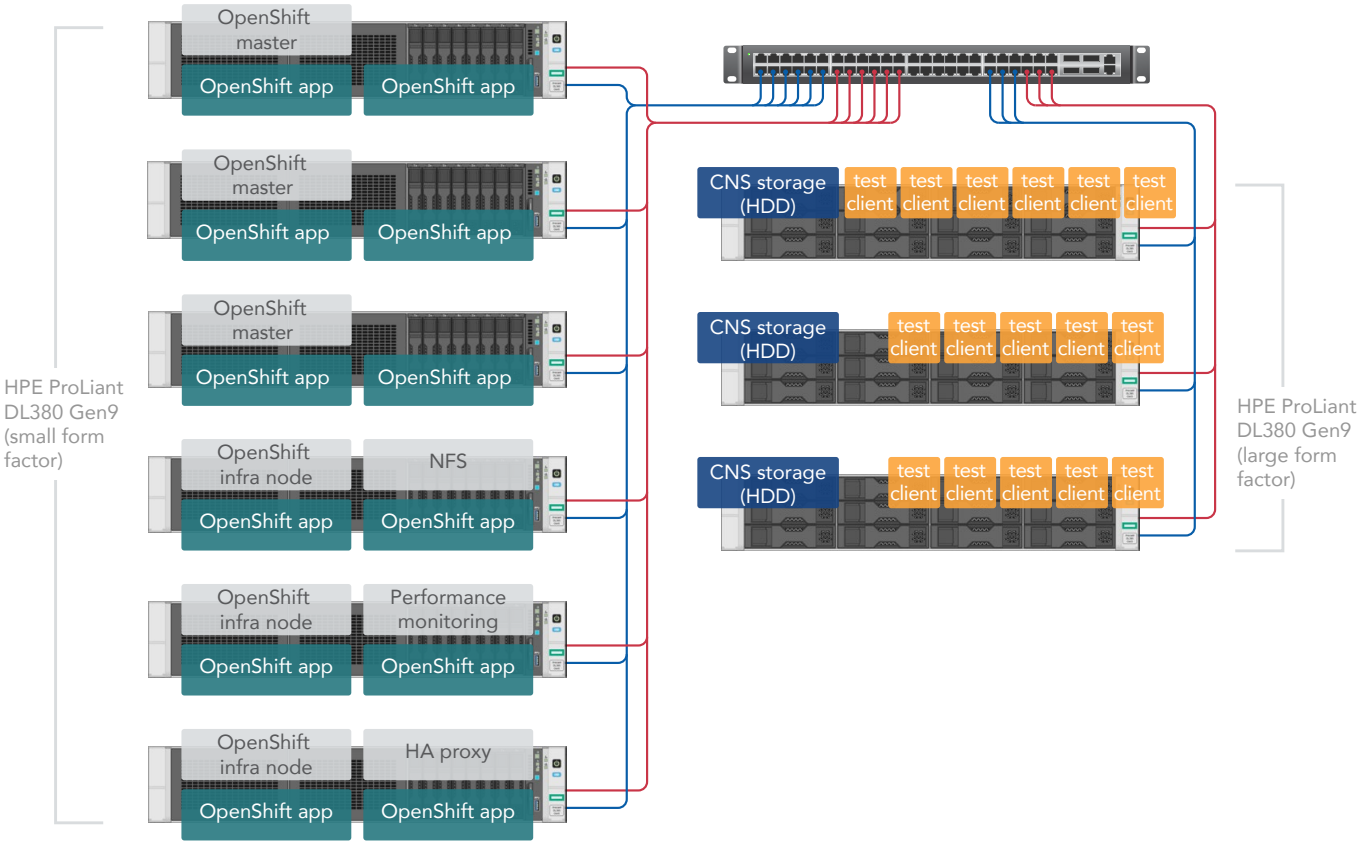
Through the phases of the testing, the servers ran 24 VMs (also referred to as nodes) comprising the OpenShift and CNS infrastructure, and up to 24 VMs for test-client nodes, as follows:

Virtual machine nodes	Role
12 OpenShift app nodes	Run OpenShift apps for both test cases. In Phase 1, we ran apps, each composed of one container running one MySQL database. In Phase 2, we ran apps, each composed of two paired containers, a PHP web service, and a MySQL database.
3 CNS storage nodes	Run CNS in de-coupled mode to present persistent storage for OpenShift apps. We provisioned six CNS VMs, three for the SSD-backed storage and three for the HDD-backed storage. For each test configuration, we used only the three appropriate VMs and shut down the three unneeded ones.
9 OpenShift infra nodes	OpenShift infrastructure nodes and other supporting services
1 to 24 test-client nodes	Generate the workload targeting the test apps in each phase

This diagram shows one of the Ultrastar SS200 solid-state drive test configurations we used for both DVD Store 2 and DVD Store 3. We used the 16 test clients, shown below, to generate workloads for 128 apps.



This diagram shows one of the Ultrastar He[10] hard drive test configurations we used for both DVD Store 2 and DVD Store 3. We used the 16 test clients, shown below, to generate workloads for 128 apps.



Note that the DVD Store clients and CNS nodes are co-resident on their physical servers, as the above diagrams show. This proximity did not sway our results because the end-to-end data flow of the benchmark utility requests was as follows:

- The client utility initiated the application request.
- The client sent the request to the applications running on the OpenShift app nodes, running on remote physical servers.
  - In the IO-intensive database-only DS2 scenario, the client sent the request directly to the MySQL database running on the OpenShift app node.
  - In the full-web app DS3 scenario, the client sent the request to the Apache web server running as part of the LAMP stack, which communicated with its MySQL server.
- In both scenarios, the networked backing store for MySQL data resided on remote CNS nodes, so the IO requests went back the other direction.
- The IO returned to the application.
- The application signaled to the client that the application request was complete.

VM node configurations

Note: As we discuss on pages 11 and 12, we used two different VM configurations for the CNS nodes in the IO-intensive workload testing. We sized VMs were sized to avoid oversubscription of underlying physical hardware.

Role		# of vCPUs	RAM (GB)	Virtualized NICs
CNS de-coupled nodes	Configuration 1 (IO-intensive only)	4	8	2x 10GbE
	Configuration 2	16	32	2x 10GbE
OpenShift app nodes		12	96	2x 10GbE
OpenShift infrastructure nodes		2	8	2x 10GbE
OpenShift masters		8	32	1x 10GbE
OpenShift HA proxy		4	4	1x 10GbE
OpenShift NFS		4	32	1x 10GbE
Performance monitoring server		2	8	1x 10GbE
DVD Store clients		2	2	1x 10GbE

Software details

- VMware vSphere 6.5
- VMware vCenter Server™ 6.5
- Red Hat Enterprise Linux 7.3
- Container-Native Storage 3.6 for Red Hat OpenShift Container Platform (based on Red Hat Gluster Storage 3.3)
- Red Hat OpenShift Container Platform 3.6
  - OpenShift 3.6 is deployed in the system with the following subcomponents:
    - Features: Basic-Auth GSSAPI Kerberos SPNEGO
    - openshift v3.6.173.0.5
    - kubernetes v1.6.1+5115d708d7

Workload discussion

DVD Store

This workload generator, two versions of which are available online at https://github.com/dvdstore, tests online transaction processing systems and has been a part of benchmark suites, notably VMware VMmark.

DVD Store simulates an ecommerce store and reports results primarily in terms of number of orders per minute (OPM) and response time. We used DVD Store 2 for the IO-intensive scenario and DVD Store 3 for the full-web app scenario. Due to the different underlying structure of each version's database, and the procedures they execute, we have found they simulate significantly different systems. For this reason, the results from the two scenarios should not be compared to one another.

DVD Store 2

The DVD Store version 2 workload generator involves simulated customers creating accounts, logging in, searching for items, and placing orders. It is available for MySQL, Microsoft® SQL Server®, Oracle, and PostgreSQL. In this study, we used MySQL 5.7 as it is shipped from the Red Hat Container Registry. While it is possible to have DVD Store 2 run with a web front end, we did not do so in our testing.

The DS2 client, which runs on separate virtual machines, simulates a specified number of users or customers. After invoking the workload, each user initiates a connection to the database and attempts to log into the application. If the customer does not exist, DS2 creates a new customer; by default, this occurs 20 percent of the time. If the customer does exist, the process continues. The customer searches for items a specified number of times, places an order with multiple lines, and the order process completes. The client then waits for a specified period of think time.

DVD Store 3

DVD Store version 3 builds on the logic and operations of DVD Store 2, adding features that customers typically expect in more modern applications. In DS3, customers can add product reviews and rate the helpfulness of other users' reviews on a scale of 1 to 10. To facilitate these additions, this version includes seven new stored procedures. The product review logic and helpfulness ratings logic take place just prior to the order being placed. Additionally, during the new customer creation step of the DS3 workload, logic occurs that enables the selection of membership tiers, such as bronze, silver, or gold.

The addition of these features shifts the workload profile from an IO focus toward a more CPU-intensive focus. Because of this shift, we determined that DVD Store 3 was the more appropriate choice for the full-web app scenario and used the full LAMP stack with the web front end. In our testing, the front-end application resided in one containerized app while the back-end MySQL store resided in another containerized app.

About response time

In both phases of our testing, our goal was to determine the maximum performance, in terms of OPM, that each configuration could achieve without an excessively long response time, as reported by the application. While opinions vary on what constitutes an acceptable threshold, faster is always better. In our analysis, we used an upper cutoff of one second (1,000 milliseconds), though our solutions delivered sub-200ms response times under many conditions.

Test phases and results

Phase 1: Testing storage performance under IO-intensive conditions with DVD Store 2

Overview

In IO-intensive environments, one criterion for success is the IO subsystem being able to keep up with a very large amount of concurrent app requests. We designed the first phase of our testing with this IO-intensive scenario in mind. We tested both the Ultrastar SS200 solid-state drive and the Ultrastar He[10] hard drive configurations to demonstrate the impact and scaling of a heavy IO-intensive workload. To achieve our goal of determining the maximum performance within acceptable response-time limits, we tested scaling with the following quantities of concurrent application instances: 1, 2, 8, 32, 96, and 128. We stopped at 128 app instances because we found performance began to decline or response time reached unacceptable levels at this point.

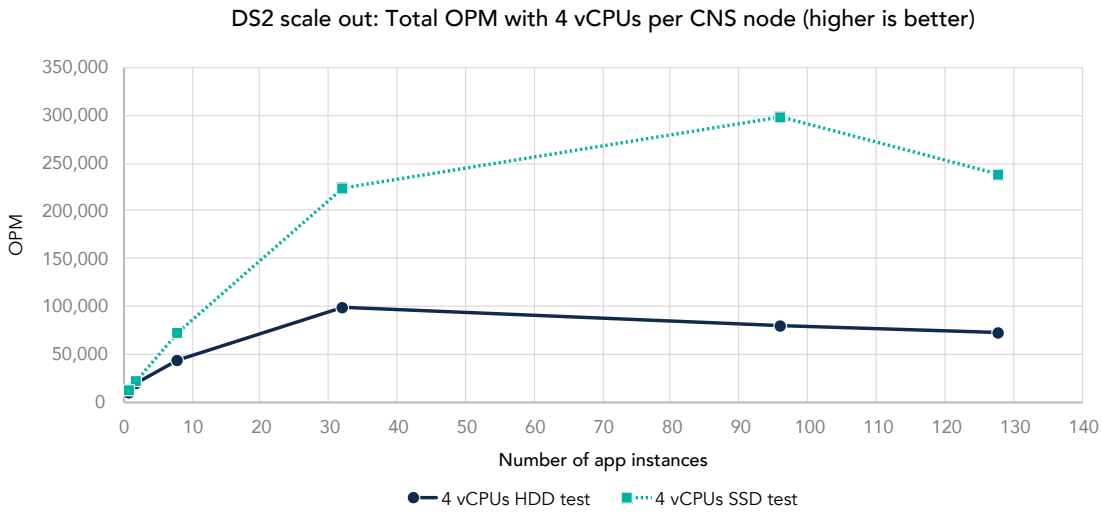
Both versions of DVD Store let you adjust the number of users and think time to have the tool simulate a heavier or lighter load. In this phase of testing, we used DVD Store 2 with the following configuration parameters to create a heavy, IO-intensive load:

- Database size: 20 GB
- Think time: 10 ms
- User count (per instance): 8
- Test mode: Database only

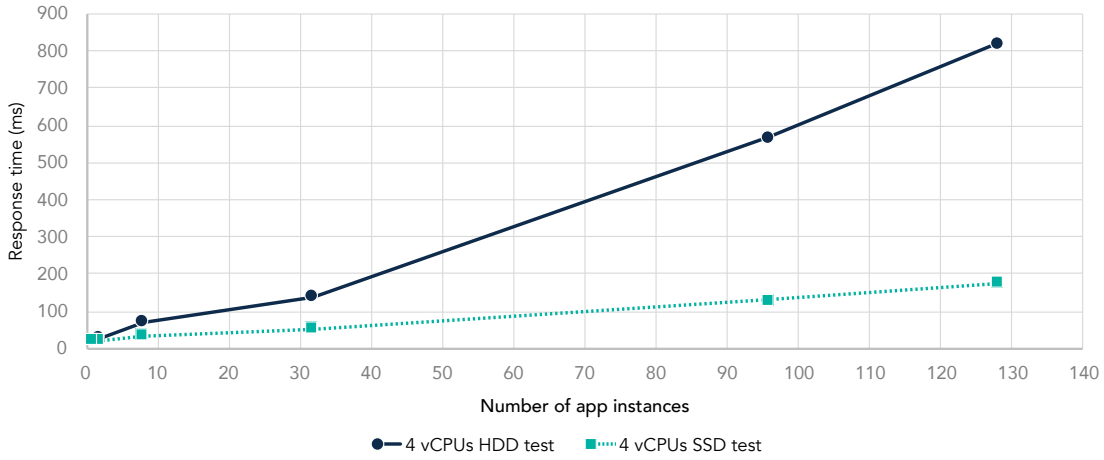
Test results

Phase 1a: Four vCPUs per CNS node

During these first test runs, we assigned our CNS nodes four vCPUs each. We started our testing with a single app instance, and then scaled up to use 2, 8, 32, 96, and 128 app instances. The following figures show the effect of this scaling on orders per minute and response time as reported by DVD Store 2 for the IO-intensive scenario. (The total OPM is the sum of OPM from each DVD Store client, and the response time is the average response time from each DVD Store client.)



DS2 scale out: Average response time with 4 vCPUs per CNS node (lower is better)



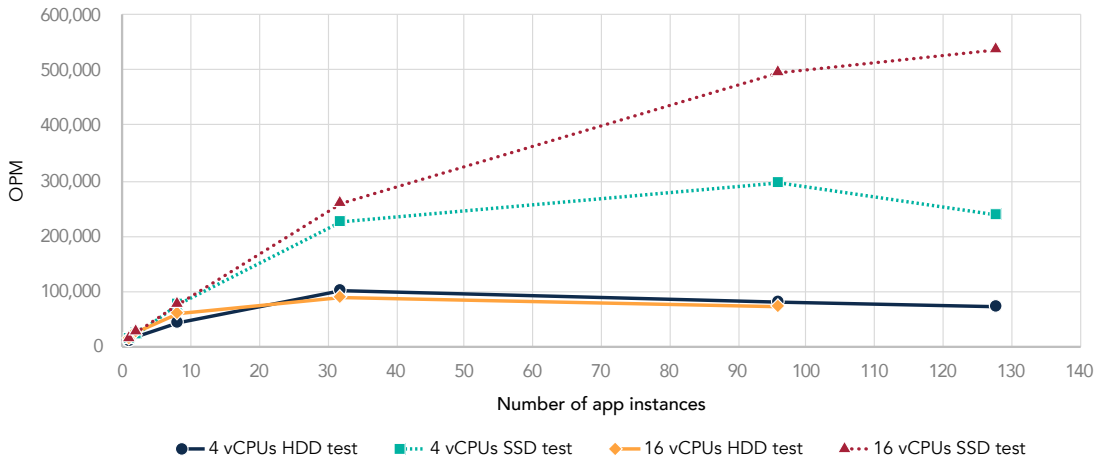
As the charts above show, when the number of app instances increased from 32 to 96, performance in terms of OPM ceased to scale optimally and response time increased dramatically. To understand why this happened, we performed resource utilization analysis, which revealed that the limiting resource was CPU cycles on the CNS nodes.

We present resource utilization data in Appendix D. As those data show, overall system throughput (orders per minute) was indeed constrained by the number of vCPUs assigned to the CNS node.

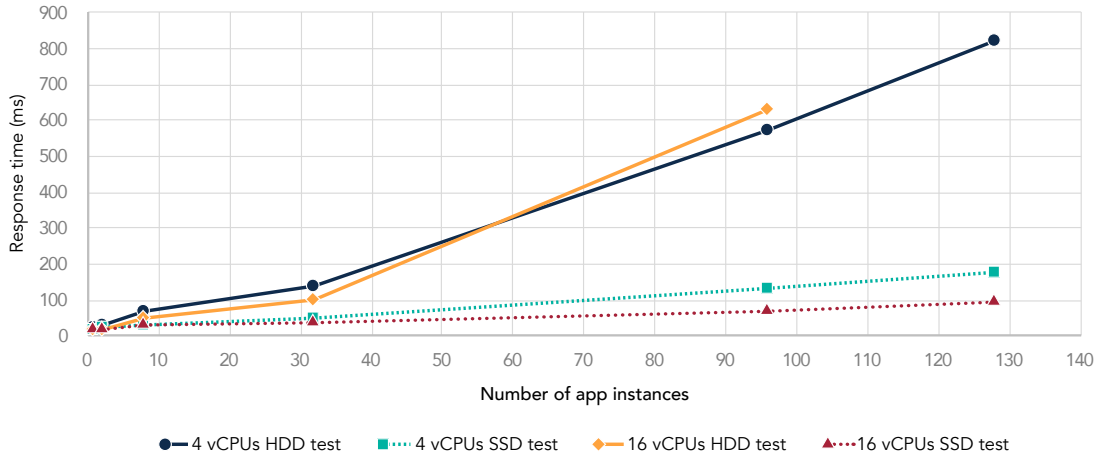
Phase 1b: I6 vCPUs per CNS node

Keeping in mind our goal of identifying bottlenecks, we wanted to confirm that the number of vCPUs allocated to each CNS node was in fact constraining throughput. To do so, we increased this number from four to 16, and repeated the testing above. The following figures show the effect of increasing the vCPU allocation to the CNS nodes on OPM and response time as reported by DVD Store 2. (The total OPM is the sum of OPM from each DVD Store client, and the response time is the average response time from each DVD Store client.)

DS2 scale out: Total OPM with 4 vCPUs and 16 vCPUs per CNS node (higher is better)



DS2 scale out: Average response time with 4 vCPUs and 16 vCPUs per CNS node (lower is better)



Analysis

As the OPM figure above shows, the additional vCPUs had little effect on Ultrastar He[10] hard drive performance at lower app instance quantities. With 128 app instances, however, response time on this configuration increased so much that the workload timed out before returning a result.

In contrast, the additional vCPUs allowed the performance of the Ultrastar SS200 solid-state drives to increase as we scaled from 96 to 128 app instances. This shows that our hypothesis—that the number of vCPUs allocated to each CNS node was constraining throughput—was correct. It also shows that even when vCPU is adequate, having a slower disk subsystem will cause latencies to rise and throughput to decrease, as the 16 vCPU HDD test results show.

The response time graph above shows that the additional vCPUs reduced the already low response time with the Ultrastar SS200 solid-state drive configuration. Being able to deliver a sub-200ms threshold for more app instances, and by extension more end users, can have a direct impact on a company's bottom line.

Phase 2: Testing performance in a full-web app environment with the DVD Store 3 workload

Overview

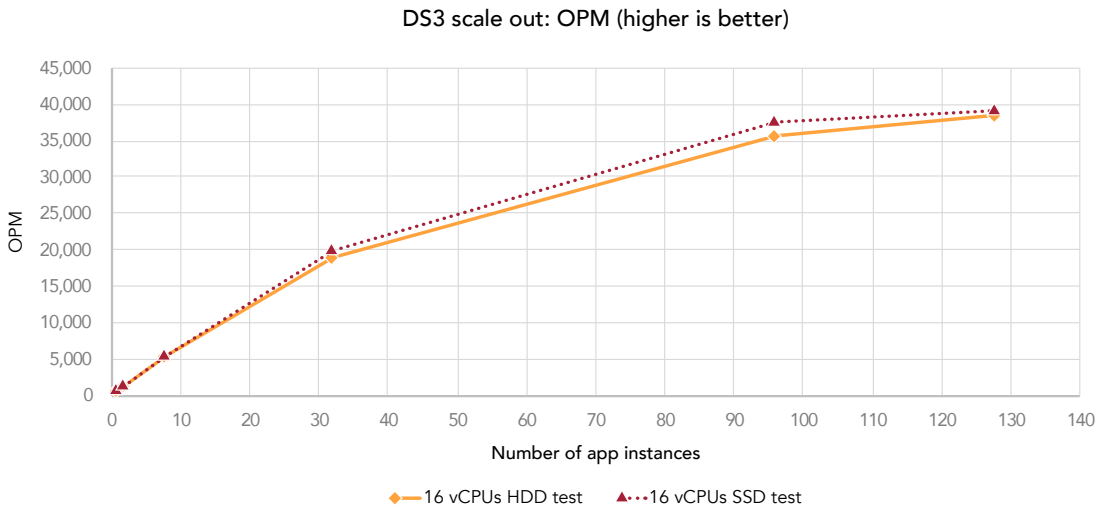
Many modern applications put a heavier emphasis on CPU processing than on IO. To see how our solution handled this kind of workload, we tested both the Ultrastar SS200 solid-state drive and the Ultrastar He[10] hard drive configurations to demonstrate the impact and scaling of a CPU-intensive, full-web app OLTP workload. We used the vCPU allocation of 16 per CNS node that we used in phase 1b. We tested scaling at six app instance counts: 1, 2, 8, 32, 96, and 128.

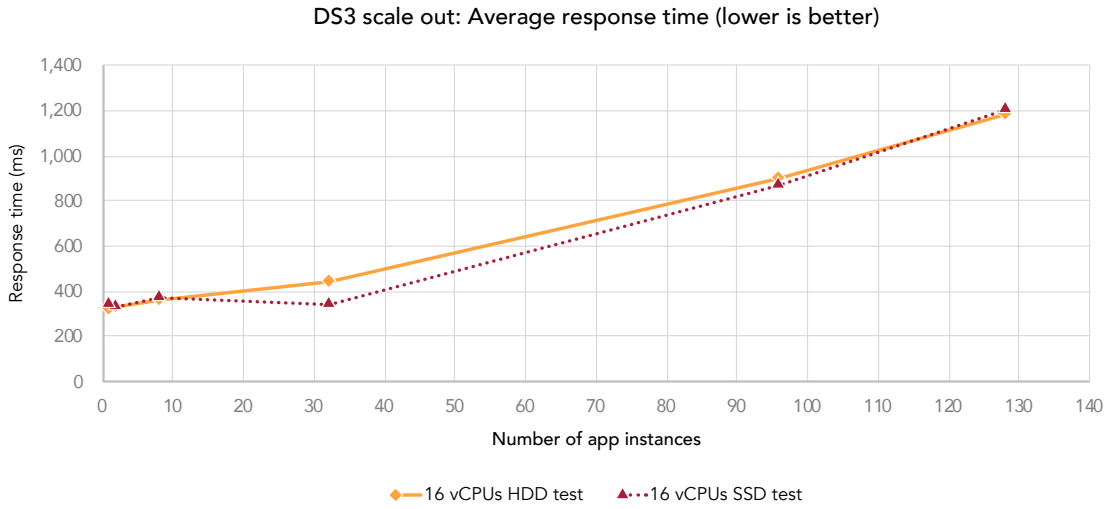
To create the CPU-intensive workload, we used the following parameters:

- Database size: 5 GB
- Think time: 300 ms
- User count (per instance): 6
- Test mode: HTTP API

Test results

The following figures show the OPM and response time as reported by DVD Store 3 as we scaled from 1 to 128 app instances on the two configurations. (The total OPM is the sum of OPM from each DVD Store client, and the response time is the average response time from each DVD Store client.)





Analysis

While running the CPU-intensive DVD Store 3 workload, the Ultrastar He[10] hard drive configuration and the Ultrastar SS200 solid-state drive configuration performed comparably, with maximum OPM counts of 35,786 and 37,424 respectively. From analyzing utilization data, we concluded that the OpenShift-based app itself was constrained by CPU.

This difference of less than 5 percent was in marked contrast to the dramatic performance differential we saw between the two configurations on the IO-intensive workload. These results highlight the importance of understanding your application behavior so that you can select the appropriate hardware.

Price/performance analysis

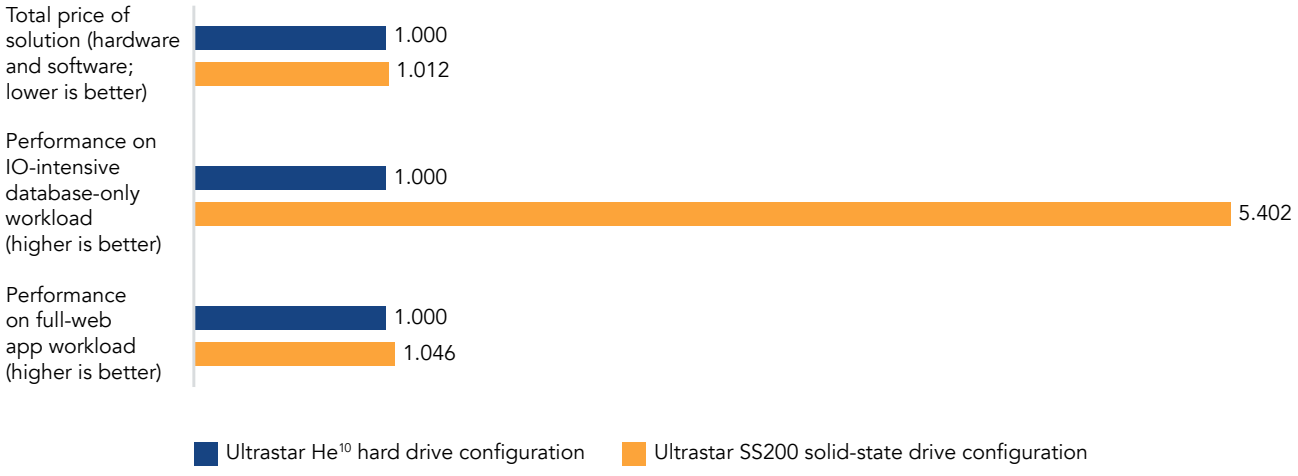
Whenever companies select an IT solution, they must consider the value in performance they will reap from different levels of investment. In previous sections of this document, we presented findings from our proof-of-concept study of a Container-Native Storage implementation using Red Hat OpenShift Container Platform and Red Hat Container-Native Storage on VMware vSphere. We looked at two hardware configurations using Western Digital media for CNS storage. Depending on a company's needs, either of these storage options—the Ultrastar SS200 solid-state drive or the Ultrastar He[10] hard drive—can be an appropriate choice.

In this section, we consider the total pricing of the two configurations we tested—including hardware and software—along with the performance results from our study to analyze the price/performance they deliver. Note that the pricing for each configuration includes the list price of the nine HPE ProLiant DL380 Gen9 servers in each solution including three-year support, the Ultrastar drives used for CNS storage, and licensing for the VMware and Red Hat software we ran on the solution.

For ease of comparison, we present the total price for each solution and the maximum total performance each achieved in normalized values. For actual pricing, and a discussion of our methods for determining it, see Appendix E. For actual performance results, see Appendix D. (Note that we used the maximum performance each configuration achieved while delivering a response time of less than one second.)

As the figure below shows, in the IO-intensive scenario, although the Ultrastar SS200 solid-state drive configuration is only 1.24 percent more expensive than the Ultrastar He[10] hard drive configuration, it delivered more than five times the performance on this workload.

Normalized comparison of pricing and maximum performance



In the CPU-intensive scenario, the Ultrastar SS200 solid-state drive configuration delivered 1.05 times the performance of the Ultrastar He[10] hard drive configuration—an improvement of 4.6 percent—while again costing only 1.24 percent more. For companies whose applications follow a similar workload profile in terms of resource usage, this can make the Ultrastar solid-state drive-based solution a wise investment. The hard drive configuration does provide more than double the total storage capacity over the SSD configuration.

Another way to look at the relationship between pricing and performance is to divide the total cost of the solution by the maximum number of OPM it achieved. The charts below show a normalized comparison of cost per OPM with the two solutions on the two workloads.

Normalized cost to perform one OPM on IO-intensive workload

Normalized cost to perform one OPM on full-web app workload

As the figures above show, those running workloads similar in nature to those we used in the IO-intensive phase of testing could see an 81.3 percent better price-per-performance by choosing the SSD configuration over the HDD configuration. Those whose workloads follow the more balanced profile we tested in the DS3 phase could see a 3.2 percent better price-per-performance by making that same choice.

Conclusion

For companies in need of a comprehensive strategy for containers and software-defined storage, Red Hat Container Ready Storage paired with Red Hat OpenShift Container Platform offer a solution that allows them to leverage their investment in VMware vSphere. In our proof-of-concept study, we explored the scaling capabilities of a CNS implementation using two types of Western Digital storage media, Ultrastar He[10] hard drives and the new Ultrastar SS200 solid-state drives. We tested the solutions under a variety of conditions, using both IO-intensive and CPU-intensive workloads, multiple vCPU allocation counts, and a range of quantities of app instances. In this document, we have presented some of the many resulting data points, including price/performance metrics, which have the potential to assist IT professionals implementing CNS to meet the unique needs of their businesses.

In September 2017, Western Digital's Data Propulsion Lab, who hosted the hardware, finalized most of the hardware we used for this testing. They added RAID controllers to the HDD servers on November 29, 2017. On November 11, 2017, Red Hat finalized the CNS software configurations we tested. Updates for current and recently released hardware and software appear often, so unavoidably these configurations may not represent the latest versions available when this report appears. We concluded hands-on testing on December 8, 2017.

Appendix A: System configuration information

Server configuration information	3x HPE ProLiant DL380 Gen9 (small form factor)	3x HPE ProLiant DL380 Gen9 (large form factor)	3x HPE ProLiant DL380 Gen9 (small form factor, OpenShift only)
BIOS name and version	P89 v2.40 (02/17/2017)	P89 v2.40 (02/17/2017)	P89 v2.40 (02/17/2017)
Non-default BIOS settings	Disabled C States, Enabled Hyper Threading, Set to UEFI	Disabled C States, Enabled Hyper Threading, Set to UEFI	Disabled C States, Enabled Hyper Threading, Set to UEFI
Operating system name and version/release number	ESXi 6.5 4564106-HPE-650.9.6.0.28	ESXi 6.5 4564106-HPE-650.9.6.0.28	ESXi 6.5 4564106-HPE-650.9.6.0.28
Power management policy	Max Performance Profiles, Max Cooling	Max Performance Profiles, Max Cooling	Max Performance Profiles, Max Cooling
Processor			
Number of processors	2	2	2
Vendor and model	Intel Xeon E5-2697A v4	Intel Xeon E5-2697 v4	Intel Xeon E5-2697A v4
Core count (per processor)	16	18	16
Core frequency (GHz)	2.60	2.30	2.60
Memory module(s)			
Total memory in system (GB)	256	256	256
Number of memory modules	8	8	8
Size (GB)	32	32	32
Type	DDR4	DDR4	DDR4
Speed (MHz)	2,400	2,400	2,400
Storage controller 1			
Vendor and model	HPE SmartHBA H240ar	HPE SmartHBA H240ar	HPE SmartHBA H240ar
Storage controller 2			
Vendor and model		HPE SmartArray P440ar	
Cache size (GB)		2	
Firmware version		6.06	

Server configuration information	3x HPE ProLiant DL380 Gen9 (small form factor)	3x HPE ProLiant DL380 Gen9 (large form factor)	3x HPE ProLiant DL380 Gen9 (small form factor, OpenShift only)
OS drives			
Number of drives	1	1	1
Drive vendor and model	SanDisk® Lightning Ascend™ Gen II	SanDisk Lightning Ascend Gen II	SanDisk Lightning Ascend Gen II
Drive size (GB)	800	800	800
Drive information (speed, interface, type)	12 Gb/s, SAS, SSD	12 Gb/s, SAS, SSD	12 Gb/s, SAS, SSD
Controller	HPE SmartHBA H240ar	HPE SmartHBA H240ar	HPE SmartHBA H240ar
Backing drives for CNS			
Number of drives	5	12	
Drive vendor and model	HGST Ultrastar SS200 SSD	HGST Ultrastar He[10]	
Drive size (TB)	1.92	10	
Drive information (speed, interface, type)	12 Gb/s, SAS, SSD	6 Gb/s, SATA, HDD	
Controller	HPE SmartHBA H240ar	HPE SmartHBA H240ar	
RAID configuration		12 disks, RAID 5	
Network adapter			
Vendor and model	Connectx3 HP LOM 764285-B21	Connectx3 HP LOM 764285-B21	Connectx3 HP LOM 764285-B21
Number and type of ports	2	2	2
Power supplies			
Vendor	HPE	HPE	HPE
Number of power supplies	2	2	2
Wattage of each (W)	800	800	800

The small form factor servers and the large form factor servers had slightly different processor models due to availability. Virtual configurations were sized as equally as possible.

Appendix B: How we tested

We worked with Red Hat and Western Digital engineers to configure the test bed and design the testing approach. Hardware resided onsite at a Western Digital data center. Western Digital engineers performed hardware configuration, maintenance, and VMware vSphere installation and configuration. Western Digital and Red Hat engineers installed virtual machines running Red Hat Enterprise Linux. Red Hat engineers performed configuration of the Red Hat OpenShift Container Platform and the Container Ready Storage configuration. PT engineers worked with Red Hat engineers on configuration changes through the course of the project and performed the scenario test runs.

Phase 1: IO-intensive tests

Preparing for testing

Note: We performed steps 1 through 8 on one of the OpenShift master nodes and steps 9 and 10 on all DVD Store client nodes.

1. We created one 20GB DS2 dataset, using the DS2 tools, and compressed the CSV tables. We deleted an unneeded copy of the prod.csv file in the orders directory.
2. We created two shared persistent volumes via storage claims. We used one volume for the SSD tests and the second for HDD tests. They held the DS2 scripts and data that we would load into each application. We executed this command, using the template `create-aux-storage.yml` (see Appendix C), setting the claim size to 20GB, and the storage class to crs-ssd and then crs-hdd.

   ```
   oc create -f create-aux-storage.yml
   ```

From each volume, we performed steps 3 through 6.

3. From each persistent-volume's metadata, we identified the name of the Gluster volume, and temporarily mounted it on one OpenShift app node.
4. We modified the DS2 database load SQL files to reference data tables in the directory GB20.
5. We created the directory GB20 and copied the DS2 data directories from step 1 to it.
6. We copied the script `mysqlds2_create_all.zip.sh` (see Appendix C) to this volume and unmounted it.
7. We created the app instances needed for this test run by running the script **`deploy-ds2-apps.sh`**, which references the template `ds2-only-mysql-on-openshift3.yml` (see Appendix C for both files). We modified the script to set the number of app instances in the loop.
8. After checking the logs from the MySQL pods to confirm that the app instances were ready, we ran the script **`reset-ds2.sh`** (see Appendix C) to load the DS2 data. Note: You can see the progress of the data loading by checking the MySQL status on each pod via remote command.
9. We opened one PowerShell window on each DVD Store client VM required for this run. (We needed one DVD Store client VM for every eight app instances.)
10. On each DVD Store client node, we created static routes to the exposed IP addresses of the eight clients that would receive requests from this node. The MySQL database uses an IP address on an internal OpenShift network that is exposed on the OpenShift master nodes. These nodes act as gateways as they are on the same network as the DVD Store clients. We specified these IP addresses in the app-creation template so that App1, App2, App3, …, received IP addresses of 172.29.0, 172.29.0,2, 172.29.0.3, … The static routes are manually balanced between the three master nodes (namely, IP addresses 172.0.109.202-204). For example,

    On DS2 client 1, we created these routes:

    ```
    route add 172.29.0.1 mask 255.255.255.255 172.0.10.202
    route add 172.29.0.2 mask 255.255.255.255 172.0.10.203
    route add 172.29.0.3 mask 255.255.255.255 172.0.10.204
    route add 172.29.0.4 mask 255.255.255.255 172.0.10.202
    route add 172.29.0.5 mask 255.255.255.255 172.0.10.203
    route add 172.29.0.6 mask 255.255.255.255 172.0.10.204
    route add 172.29.0.7 mask 255.255.255.255 172.0.10.202
    route add 172.29.0.8 mask 255.255.255.255 172.0.10.203
    ```

    …

    On DS2 client 4, we created these routes:

    ```
    route add 172.29.0.1 mask 255.255.255.255 172.0.10.202
    route add 172.29.0.2 mask 255.255.255.255 172.0.10.203
    route add 172.29.0.3 mask 255.255.255.255 172.0.10.204
    route add 172.29.0.4 mask 255.255.255.255 172.0.10.202
    route add 172.29.0.5 mask 255.255.255.255 172.0.10.203
    route add 172.29.0.6 mask 255.255.255.255 172.0.10.204
    route add 172.29.0.7 mask 255.255.255.255 172.0.10.202
    route add 172.29.0.8 mask 255.255.255.255 172.0.10.203
    ```

Executing the test

Note: The following step performs a single test run. We conducted three runs.

1. To start the test, we ran the following command, with the targets set to the IP addresses for the app instances used by that client. For example, on DVD Store client 1, using four app instances, we ran:

```
.\ds2mysqldriver.exe --db_size=20GB --run_time=5 --warmup_time=2 --think_time=0.01 `
--n_threads=8 --target='172.29.0.1;172.29.0.2;172.29.0.3;172.29.0.4'
```

Phase 2: CPU-intensive tests

Preparing for testing

Note: We performed steps 1 through 8 on one of the OpenShift master nodes and step 9 on all DVD Store client nodes.

1. We created one 5GB DS3 dataset, using the DS3 tools, and compressed the CSV tables. We deleted an unneeded copy of the prod.csv file in the orders directory.
2. We created two shared persistent volumes via storage claims. We used one volume for the SSD tests and the second for HDD tests. They held the DS3 scripts and data that would be loaded into each application. We executed this command, using the template `create-aux-storage.yml` (see Appendix C), setting the claim size to 11 GB, and the storage class to crs-ssd and then crs-hdd.

```
oc create -f create-aux-storage.yml
```

From each volume, we performed steps 3 to 6.

3. From each persistent-volume's metadata, we identified the name of the Gluster volume and temporarily mounted it on one OpenShift app node.
4. We modified the DS3 database load SQL files to reference data tables in the directory GB5.
5. We created the directory GB5 and copied the DS3 data directories from step 1 to it.
6. We copied the script `mysqlds3_create_all.zip.sh` (see Appendix C) to this volume and unmounted it.
7. We created the app instances needed for this test run by running the script `deploy-ds3-apps.sh`, which references the template `ds3-web-mysql-on-openshift3.yml` (see Appendix C for both files). We modified the script to set the number of app instances in the loop.
8. After checking the logs from the MySQL pods to confirm that the app instances were ready, we ran the script `reset-ds3.sh` (see Appendix C) to load the DS3 data. Note: You can see the progress of the data loading by checking the MySQL status on each pod via remote command.
9. We opened one PowerShell window on each DVD Store client VM needed for this run. (We needed one DVD Store client VM for every eight app instances.)

Executing the test

Note: The following step performs a single test run. We conducted three runs.

1. To start the test, we simultaneously ran the following command, with the targets set to the URLs assigned automatically by OpenShift for the apps used by that client. For example, on DVD Store client 1, using four apps, we ran

```
.\ds3webdriver.exe --db_size=5GB --run_time=10 --warmup_time=5
--think_time=0.3 --n_threads=6 --virt_dir=\'
--target= target='ds3-mysql-web-1-bbb.apps.dpl.local;ds3-mysql-web-2-bbb.apps.dpl.local;ds3-mysqlweb-
3-bbb.apps.dpl.local;ds3-mysql-web-4-bbb.apps.dpl.local' 2>&1 >> ds3_test-01_001.txt
```

Appendix C: Scripts

mysqlds2_create_all.zip.sh

```
#!/bin/bash
# directories for fifos and networked data
zip=/tmp/zip
dat=/tmp/data

mysql -u root < ${dat}/grants.sql

export MYSQL_PWD=web

# create DB and schema
mysql -u web < ${dat}/mysqlds2_create_db.sql
mysql -u web < ${dat}/mysqlds2_create_ind.sql
mysql -u web < ${dat}/mysqlds2_create_sp.sql
# create and prime fifos
mkdir -p ${zip}/GB20/{cust,orders,prod}

for file in ${dat}/GB20/*/*.csv.gz ; do
  pipe=${file%.gz}
  pipe=${zip}/${pipe#$dat}
  mkfifo --mode=0660 ${pipe}
  gzip -d < ${file} > ${pipe} &
done

printf "Starting data load at "; date

cd ${zip}

mysql -u web < ${dat}/mysqlds2_load_cust.sql
mysql -u web < ${dat}/mysqlds2_load_orders.sql
mysql -u web < ${dat}/mysqlds2_load_orderlines.sql
mysql -u web < ${dat}/mysqlds2_load_cust_hist.sql
mysql -u web < ${dat}/mysqlds2_load_prod.sql
mysql -u web < ${dat}/mysqlds2_load_inv.sql

wait
printf "Ending data load at "; date

# clean up fifios
rm -f ${zip}/GB20/*/*.csv 2>&1 > /dev/null
rmdir -p ${zip}/GB20/* 2>&1 > /dev/null
```

grants.sql

```
GRANT ALL ON *.* TO 'web'@'%' IDENTIFIED by 'web';
GRANT ALL ON *.* TO 'web'@'localhost' IDENTIFIED by 'web';
```

mysqlds3_create_all.zip.sh

```
#!/bin/bash
# directories for fifos and netwworked data
zip=/tmp/zip
dat=/tmp/data

# create DB and schema
mysql -u root < ${dat}/mysqlds3_create_db.sql
mysql -u root -e 'SET GLOBAL sql_mode = "NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION" '

export MYSQL_PWD=toomanysecrets

mysql -u ds3 DS3 < ${dat}/mysqlds3_create_ind.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_create_sp.sql

# create and prime fifos
mkdir -p ${zip}/GB5/{cust,membership,orders,prod,reviews}

for file in ${dat}/GB5/*/*.csv.gz ; do
  pipe=${file%.gz}
  pipe=${zip}/${pipe#$dat}
  mkfifo --mode=0660 ${pipe}
  gzip -d < ${file} > ${pipe} &
done

printf "Starting data load at "; date
cd ${zip}

mysql -u ds3 DS3 < ${dat}/mysqlds3_load_cust.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_load_orders.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_load_orderlines.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_load_cust_hist.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_load_prod.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_load_inv.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_load_member.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_load_reviews.sql
mysql -u ds3 DS3 < ${dat}/mysqlds3_load_review_helpfulness.sql

wait
printf "Ending data load at "; date

# clean up fifios
rm -f ${zip}/GB5/*/*.csv 2>&1 > /dev/null
rmdir -p ${zip}/GB5/* 2>&1 > /dev/null
```

deploy-ds2-apps.sh

```
#!/bin/bash
oc delete template ds2-only-mysql-on-openshift3 --ignore-not-found=true

for num in {1..2} ; do
  oc process -f ds2-only-mysql-on-openshift3.yml \
     -p APP_NAME=ds2-mysql-${num} \
     -p MYSQL_VOLUME_CAPACITY=40Gi \
     -p MYSQL_USER=web \
     -p MYSQL_PASSWORD=web \
     -p MYSQL_MEMORY_LIMIT=5Gi \
     -p EXTERNAL_IP=172.29.0.${num} \
     -p STORAGE_CLASS=crs-ssd \
     -p SAMPLE_DATA_PVC=sample-data \
     --labels app=ds2-mysql-${num} |\
    oc create -f -
    sleep 1
done

exit
```

deploy-ds3-apps.sh

```
#!/bin/bash
oc delete template ds3-web-mysql-on-openshift3 --ignore-not-found=true

for num in {1..128};  do
  oc delete cm ds3-mysql-web-${num}-php-config >& /dev/null

  oc process -f ds3-web-mysql-on-openshift3.yml \
     -p APP_NAME=ds3-mysql-web-${num} \
     -p MYSQL_VOLUME_CAPACITY=15Gi \
     -p MYSQL_MEMORY_LIMIT=5Gi \
     -p STORAGE_CLASS=crs-hdd \
     -p SAMPLE_DATA_PVC=sample-data-small \
     --labels app=ds3-mysql-web-${num} |\
    oc create -f -
    sleep 1
done

exit
```

reset-ds2.sh

```
#!/bin/bash

job_limit () {
   # Test for single positive integer input
   if (( $# == 1 )) && [[ $1 =~ ^[1-9][0-9]*$ ]] ; then
      # Check number of running jobs
      joblist=($(jobs -rp))
      while (( ${#joblist[*]} >= $1 )) ; do
         # Wait for any job to finish
         command='wait '${joblist[0]}
         for job in ${joblist[@]:1} ; do
            command+=' || wait '$job
         done
         eval $command
         joblist=($(jobs -rp))
      done
   fi
}

for i in $(oc get pod | awk '/-database-/ {print $1}') ; do
  oc rsh -T $i bash /tmp/data/mysqlds2_create_all.zip.sh < /dev/null &
  job_limit 8
done
```

reset-ds3.sh

```
#!/bin/bash

job_limit () {
   # Test for single positive integer input
   if (( $# == 1 )) && [[ $1 =~ ^[1-9][0-9]*$ ]] ; then
      # Check number of running jobs
      joblist=($(jobs -rp))
      while (( ${#joblist[*]} >= $1 )) ; do
         # Wait for any job to finish
         command='wait '${joblist[0]}
         for job in ${joblist[@]:1} ; do
            command+=' || wait '$job
         done
         eval $command
         joblist=($(jobs -rp))
      done
   fi
}

for i in $(oc get pod | awk '/-database-/ {print $1}') ; do
  oc rsh -T $i bash /tmp/data/mysqlds3_create_all.zip.sh < /dev/null &
  job_limit 8
done
```

create-aux-storage.yml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
# name: sample-data-small
 name: sample-data
spec:
# storageClassName: crs-ssd
 storageClassName: crs-hdd
 accessModes:
  - ReadWriteMany
 resources:
   requests:
#     storage: 11Gi
     storage: 20Gi
```

ds2-web-mysql-on-openshift3.yml

```
apiVersion: v1
kind: Template
metadata:
  name: ds2-only-mysql-on-openshift3
  annotations:
    description: "DS2 MySQL on OpenShift 3"
    tags: "instant-app"
objects:
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
    name:  ${APP_NAME}-database
  spec:
    replicas: 1
    selector:
      deploymentconfig: ${APP_NAME}-database
    template:
      metadata:
        labels:
          deploymentconfig: ${APP_NAME}-database
      spec:
```

```
        containers:
        - name: mysql
          env:
            - name: MYSQL_USER
              value: "ds3"
            - name: MYSQL_PASSWORD
              value: "${MYSQL_PASSWORD}"
            - name: MYSQL_ROOT_PASSWORD
              value: "${MYSQL_PASSWORD}"
            - name: MYSQL_DATABASE
              value: "DS3"
          image: ' '
          resources:
            limits:
              memory: ${MYSQL_MEMORY_LIMIT}
          volumeMounts:
          - mountPath: /var/lib/mysql/data
            name: ${APP_NAME}-database-storage
          - mountPath: /tmp/data
            name: ${APP_NAME}-sample-data
        volumes:
        - name: ${APP_NAME}-database-storage
          persistentVolumeClaim:
            claimName: ${APP_NAME}-database-storage
        - name: ${APP_NAME}-sample-data
          persistentVolumeClaim:
            claimName: ${SAMPLE_DATA_PVC}
    triggers:
    - imageChangeParams:
        automatic: true
        containerNames:
        - mysql
        from:
          kind: ImageStreamTag
          name: mysql:5.7
          namespace: openshift
      type: ImageChange
    - type: ConfigChange
- apiVersion: v1
  kind: Service
  metadata:
    name: ${APP_NAME}-database
  spec:
    ports:
      - name: mysql
        port: 3306
        protocol: TCP
        targetPort: 3306
    loadBalancerIP: ${EXTERNAL_IP}
    type: LoadBalancer
    selector:
      deploymentconfig: ${APP_NAME}-database
- apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name: ${APP_NAME}-database-storage
  spec:
    storageClassName: ${STORAGE_CLASS}
    accessModes:
      - "ReadWriteOnce"
    resources:
      requests:
        storage: ${MYSQL_VOLUME_CAPACITY}
parameters:
- name: MYSQL_USER
  value: ds3
- name: MYSQL_PASSWORD
```

```
    value: toomanysecrets
  - description: The name assigned to all of the frontend objects defined in this template.
    displayName: Name
    name: APP_NAME
    required: true
    value: ds2-mysql
  - description: The exposed hostname that will route to the DS2 app.
    displayName: Application Hostname
    name: APPLICATION_DOMAIN
  - description: The size of the volume for MySQL data files.
    displayName: MySQL Volume Capacity
    name: MYSQL_VOLUME_CAPACITY
    required: true
    value: 10Gi
  - description: Maximum amount of memory the MySQL container can use.
    displayName: MySQL Memory Limit
    name: MYSQL_MEMORY_LIMIT
    required: true
    value: 1Gi
  - description: Maximum amount of memory the DS2 container can use.
    displayName: DS2 App Memory Limit
    name: DS3_MEMORY_LIMIT
    required: true
    value: 512Mi
  - description: The StorageClass to use to request the volume
    displayName: StorageClass for MySQL backend
    name: STORAGE_CLASS
    required: true
    value: glusterfs-storage
  - description: Existing PersistentVolumeClaim to mount to MySQL on /tmp/sampledata
    displayName: Sample Data PVC
    name: SAMPLE_DATA_PVC
    required: true
    value: sample-data
  - description: The external IP the MySQL service should use (always port 3306).
    displayName: External IP
    name: EXTERNAL_IP
    required: true
```

ds3-only-mysql-on-openshift3.yml

```
apiVersion: v1
kind: Template
metadata:
  name: ds3-web-mysql-on-openshift3
  annotations:
    description: "DS3 Web interface and MySQL on OpenShift 3"
    tags: "instant-app"
objects:
- apiVersion: v1
  kind: ImageStream
  metadata:
    name: ${APP_NAME}
- apiVersion: v1
  kind: BuildConfig
  metadata:
    name: ${APP_NAME}
  spec:
    triggers:
      - type: ImageChange
      - type: ConfigChange
    source:
      type: Git
      git:
        uri: https://github.com/dmesser/ds3-on-openshift.git
        ref: no-data-files
      contextDir: "ds3/mysqlds3/web/php5"
```

```
    strategy:
      type: Source
      sourceStrategy:
        from:
          kind: ImageStreamTag
          name: "php:5.6"
          namespace: openshift
    output:
      to:
        kind: ImageStreamTag
        name: ${APP_NAME}:latest
- apiVersion: v1
  kind: ConfigMap
  metadata:
    name: ${APP_NAME}-php-config
  data:
    ds3.ini: |-
      zend_extension=opcache.so
      opcache.enable=1
      opcache.memory_consumption=128
      opcache.interned_strings_buffer=8
      opcache.max_accelerated_files=4000
      opcache.revalidate_freq=2
      opcache.fast_shutdown=1
      opcache.blacklist_filename=/etc/opt/rh/rh-php56/php.d/opcache*.blacklist
      opcache.max_file_size=1M
      zend_extension=xdebug.so
      extension=bcmath.so
      extension=bz2.so
      extension=calendar.so
      extension=ctype.so
      extension=curl.so
      extension=dom.so
      extension=exif.so
      extension=fileinfo.so
      extension=ftp.so
      extension=gd.so
      extension=gettext.so
      extension=gmp.so
      extension=iconv.so
      extension=intl.so
      extension=ldap.so
      extension=mbstring.so
      extension=mysqlnd.so
      extension=pdo.so
      extension=pgsql.so
      extension=phar.so
      extension=posix.so
      extension=shmop.so
      extension=simplexml.so
      extension=soap.so
      extension=sockets.so
      extension=sqlite3.so
      extension=sysvmsg.so
      extension=sysvsem.so
      extension=sysvshm.so
      extension=tokenizer.so
      extension=xml.so
      extension=xmlwriter.so
      extension=xsl.so
      extension=zip.so
      extension=mysql.so
      extension=mysqli.so
      extension=pdo_mysql.so
      extension=pdo_pgsql.so
      extension=pdo_sqlite.so
      extension=wddx.so
```

```
      extension=xmlreader.so
      extension = json.so
      extension=memcache.so
      error_reporting = E_ALL ^ (E_DEPRECATED)
      display_errors = On
      mysql.default_host = ${APP_NAME}-database
      mysql.default_user = ds3
      mysql.default_password = ${MYSQL_PASSWORD}
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
    name: ${APP_NAME}
  spec:
    replicas: 1
    selector:
      deploymentconfig: ${APP_NAME}
    template:
      metadata:
        labels:
          deploymentconfig: ${APP_NAME}
        name: ${APP_NAME}
      spec:
        containers:
        - name: ${APP_NAME}
          image: ' '
          ports:
            - containerPort: 8080
              protocol: TCP
          imagePullPolicy: Always
          volumeMounts:
            - mountPath: /etc/opt/rh/rh-php56/php.d/
              name: volume-php-config
        volumes:
          - configMap:
              name: ${APP_NAME}-php-config
              defaultMode: 420
            name: volume-php-config
        resources:
          limits:
            memory: ${DS3_MEMORY_LIMIT}
    triggers:
    - imageChangeParams:
        automatic: true
        containerNames:
        - ${APP_NAME}
        from:
          kind: ImageStreamTag
          name: ${APP_NAME}:latest
      type: ImageChange
    - type: ConfigChange
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      description: Exposes and load balances the application pods
      service.alpha.openshift.io/dependencies: '[{"name": "${APP_NAME}-database",
        "kind": "Service"}]'
    name:  ${APP_NAME}
  spec:
    ports:
      - name: 8080-tcp
        port: 8080
        targetPort: 8080
        protocol: TCP
    selector:
      deploymentconfig: ${APP_NAME}
- apiVersion: v1
```

```
  kind: Route
  metadata:
    annotations:
      template.openshift.io/expose-uri: http://{.spec.host}{.spec.path}
    name: ${APP_NAME}
  spec:
    host: ${APPLICATION_DOMAIN}
    port:
      targetPort: 8080-tcp
    to:
      kind: Service
      name: ${APP_NAME}
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
    name:  ${APP_NAME}-database
  spec:
    replicas: 1
    selector:
      deploymentconfig: ${APP_NAME}-database
    template:
      metadata:
        labels:
          deploymentconfig: ${APP_NAME}-database
      spec:
        containers:
        - name: mysql
          env:
            - name: MYSQL_USER
              value: "ds3"
            - name: MYSQL_PASSWORD
              value: "${MYSQL_PASSWORD}"
            - name: MYSQL_ROOT_PASSWORD
              value: "${MYSQL_PASSWORD}"
            - name: MYSQL_DATABASE
              value: "DS3"
          image: ' '
          resources:
            limits:
              memory: ${MYSQL_MEMORY_LIMIT}
          volumeMounts:
          - mountPath: /var/lib/mysql/data
            name: ${APP_NAME}-database-storage
          - mountPath: /tmp/data
            name: ${APP_NAME}-sample-data
        volumes:
        - name: ${APP_NAME}-database-storage
          persistentVolumeClaim:
            claimName: ${APP_NAME}-database-storage
        - name: ${APP_NAME}-sample-data
          persistentVolumeClaim:
            claimName: ${SAMPLE_DATA_PVC}
    triggers:
    - imageChangeParams:
        automatic: true
        containerNames:
        - mysql
        from:
          kind: ImageStreamTag
          name: mysql:5.7
          namespace: openshift
      type: ImageChange
    - type: ConfigChange
- apiVersion: v1
  kind: Service
  metadata:
    name: ${APP_NAME}-database
```

```
  spec:
    ports:
      - name: mysql
        port: 3306
        protocol: TCP
        targetPort: 3306
    selector:
      deploymentconfig: ${APP_NAME}-database
- apiVersion: v1
  kind: PersistentVolumeClaim
  metadata:
    name: ${APP_NAME}-database-storage
  spec:
    storageClassName: ${STORAGE_CLASS}
    accessModes:
      - "ReadWriteOnce"
    resources:
      requests:
        storage: ${MYSQL_VOLUME_CAPACITY}
parameters:
- name: MYSQL_PASSWORD
  value: toomanysecrets
- description: The name assigned to all of the frontend objects defined in this template.
  displayName: Name
  name: APP_NAME
  required: true
  value: ds3-mysql-web
- description: The exposed hostname that will route to the DS3 app.
  displayName: Application Hostname
  name: APPLICATION_DOMAIN
- description: The size of the volume for MySQL data files.
  displayName: MySQL Volume Capacity
  name: MYSQL_VOLUME_CAPACITY
  required: true
  value: 10Gi
- description: Maximum amount of memory the MySQL container can use.
  displayName: MySQL Memory Limit
  name: MYSQL_MEMORY_LIMIT
  required: true
  value: 1Gi
- description: Maximum amount of memory the DS3 container can use.
  displayName: DS3 App Memory Limit
  name: DS3_MEMORY_LIMIT
  required: true
  value: 512Mi
- description: The StorageClass to use to request the volume
  displayName: StorageClass for MySQL backend
  name: STORAGE_CLASS
  required: true
  value: glusterfs-storage
- description: Existing PersistentVolumeClaim to mount to MySQL on /tmp/data
  displayName: Sample Data PVC
  name: SAMPLE_DATA_PVC
  required: true
  value: sample-data
```

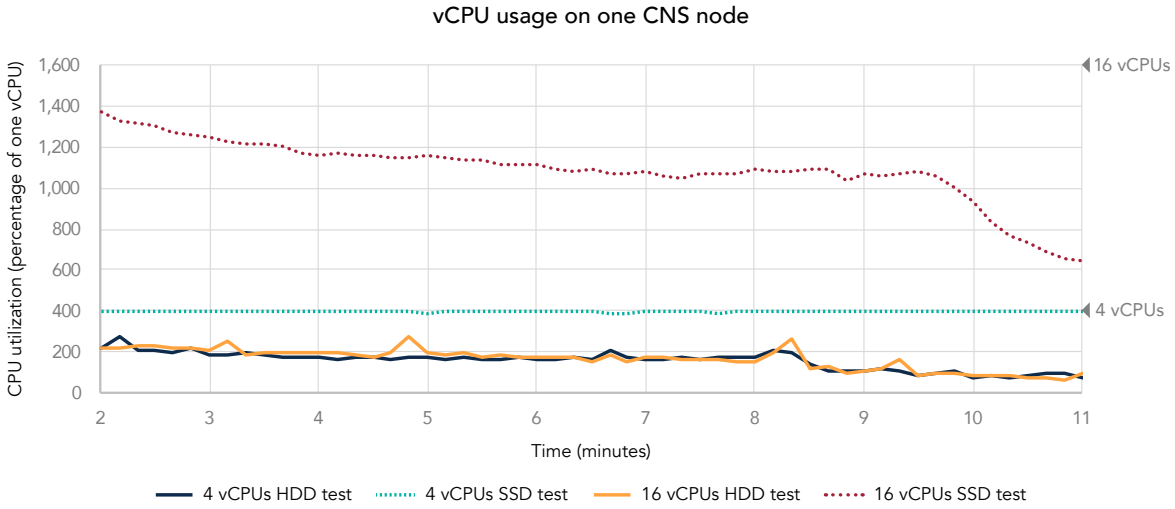
Appendix D: Detailed test results

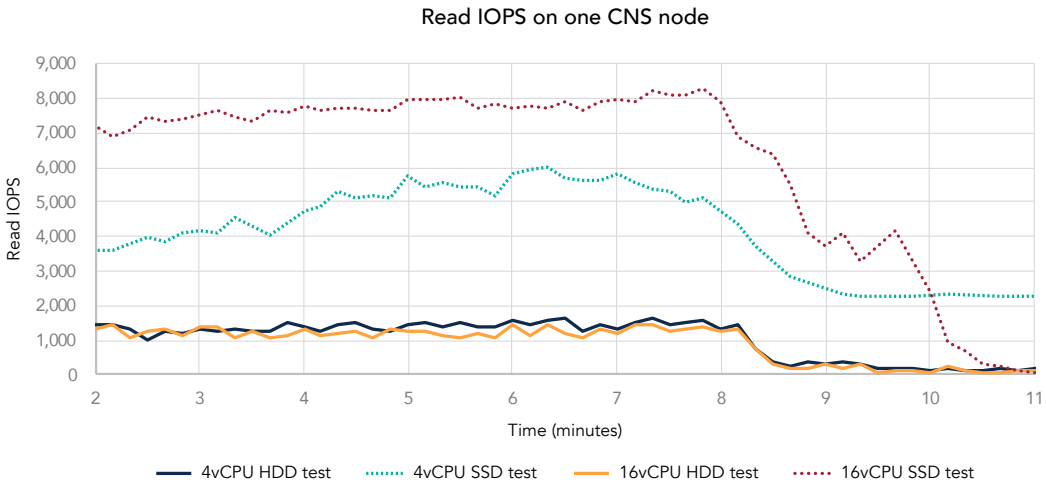
In the charts that follow, we show the key CPU and IO performance data taken from representative nodes during our test runs. For the CNS nodes, we found CPU usage and IO to the disks used by CNS to be most helpful metrics. For the OpenShift app nodes, we concentrated on CPU usage. In all cases, we exclude a 2-minute warmup period at the beginning of each run and present the performance data at 10-second intervals over the remainder of the run.

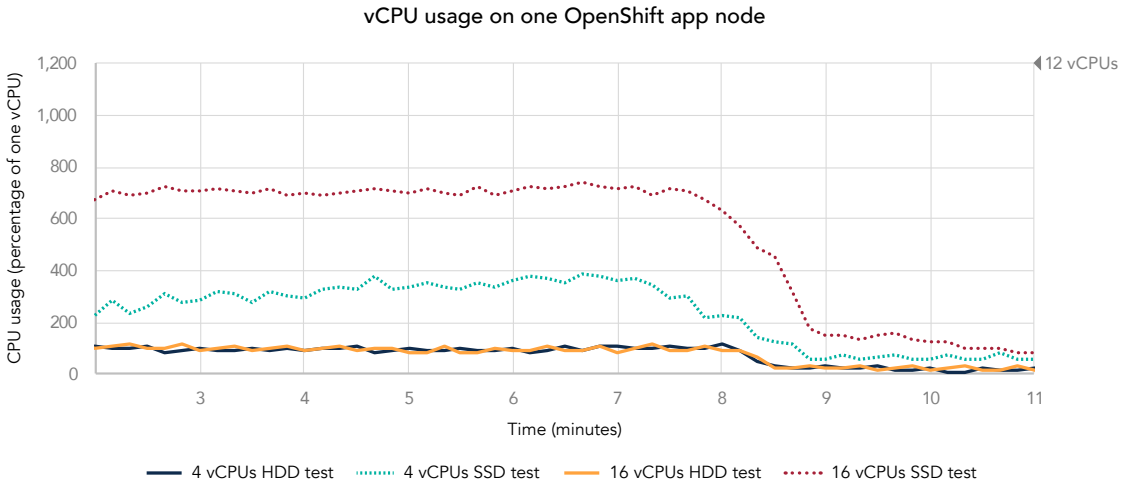
Note: The CPU usage charts use units based on one CPU so that usage can exceed 100 percent on multi-CPU nodes. For example, the nominal maximum CPU usage for the 12-vCPU OpenShift app nodes is 1,200 percent. As a guide, we have included marks at the right of the graphs to indicate the maximum.

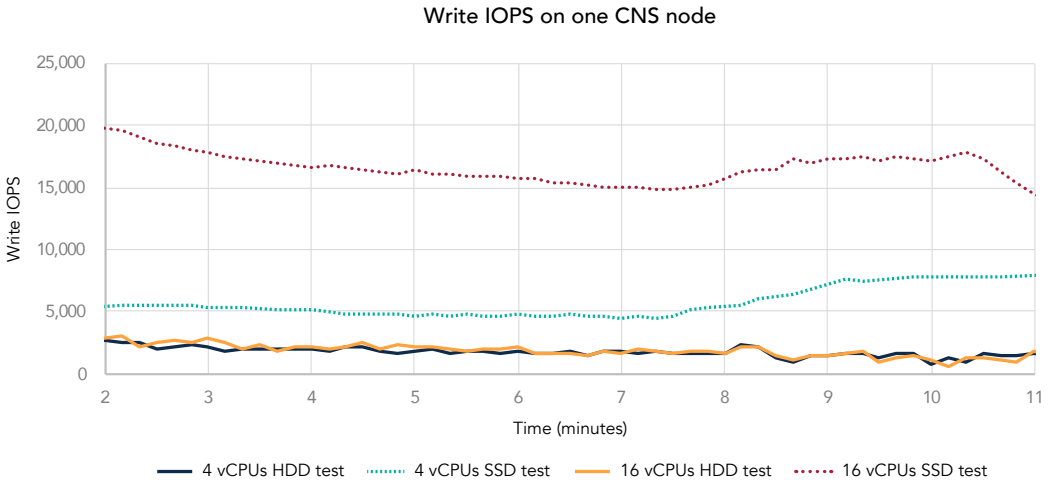
Phase 1: IO-intensive database-only workload

Each of the graphs below use data from the 96-app-instance test.



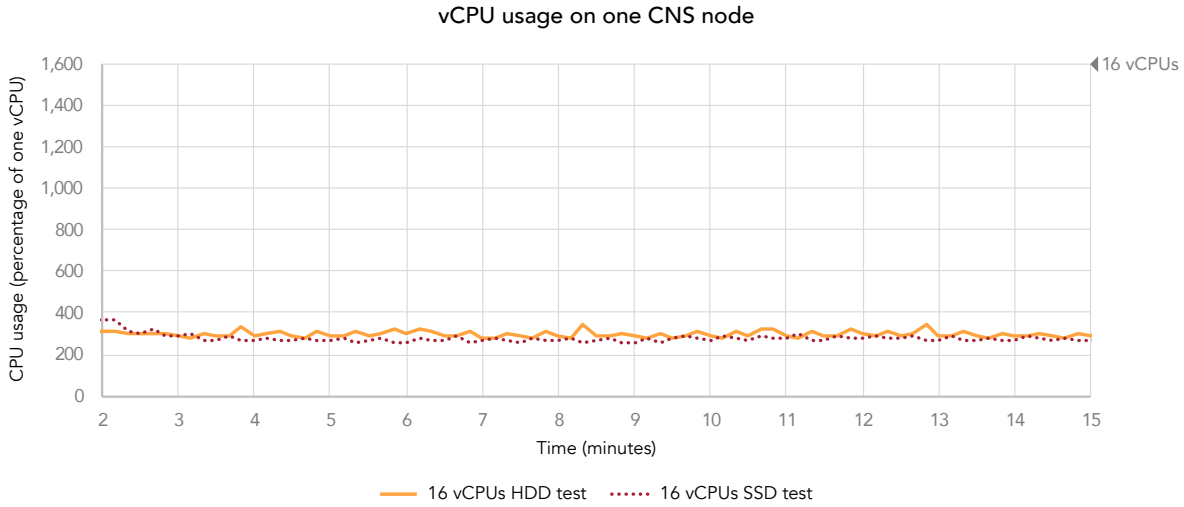




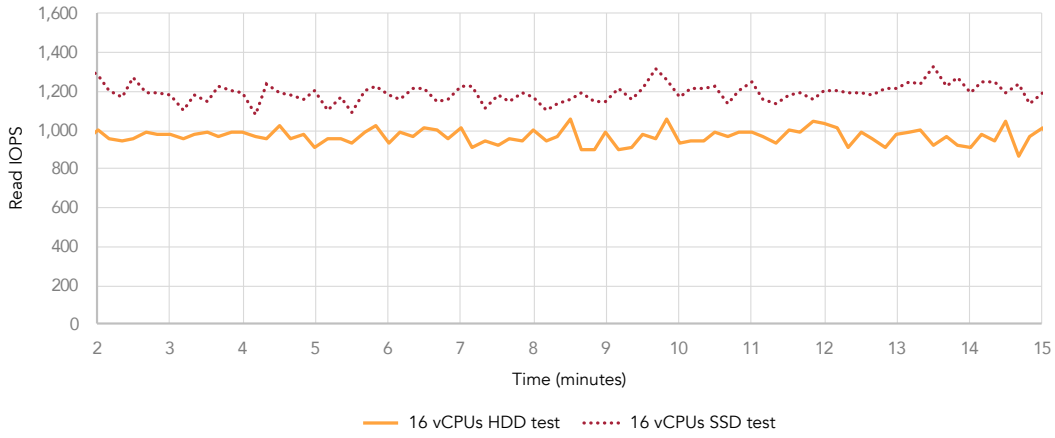


Phase 2: CPU-intensive, full-web app workload

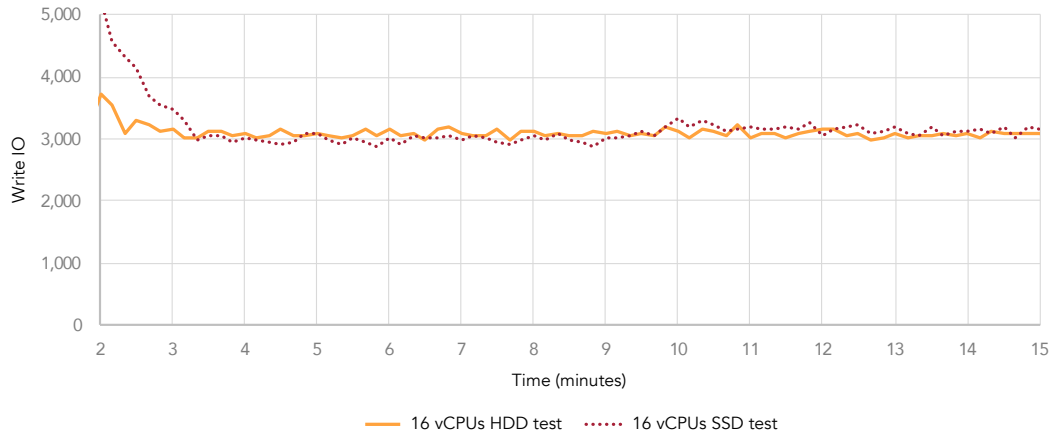
Each of the graphs below use data from the 128-app-instance test.



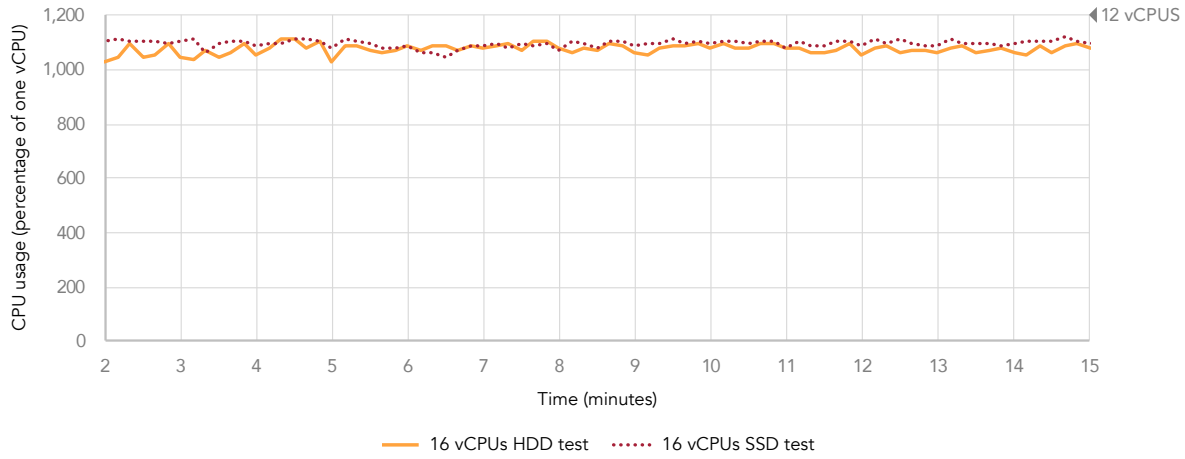
Read IOPS on one CNS node



Write IOPS on one CNS node



vCPU usage on one OpenShift app node



OPM and response time results

DS2 scale out: Total orders per minute

Test		App instances					
Number of vCPUs	Drive type	1	2	8	32	96	128
4	HDD	11,347	19,707	44,226	99,262	79,150	73,541
	SSD	12,612	23,738	73,939	223,441	297,010	238,396
16	HDD	12,473	24,566	57,860	89,489	71,738	
	SSD	12,537	24,629	75,732	258,926	495,000	536,220

DS2 scale out: Average response time (ms)

Test		App instances					
Number of vCPUs	Drive type	1	2	8	32	96	128
4	HDD	24	30	68	141	571	823
	SSD	19	22	31	49	135	174
16	HDD	20	20	46	103	629	
	SSD	19	20	30	39	71	91

DS3 scale out: Total orders per minute

Test		App instances					
Number of vCPUs	Drive type	1	2	8	32	96	128
16	HDD	722	1,430	5,307	19,035	35,786	38,494
	SSD	698	1,420	5,295	19,995	37,424	39,186

DS3 scale out: Average response time (ms)

Test		App instances					
Number of vCPUs	Drive type	1	2	8	32	96	128
16	HDD	323	328	361	443	898	1,180
	SSD	343	330	374	343	873	1,208

Appendix E: Solution pricing

For both solutions, we include prices for the nine HPE ProLiant DL380 Gen9 servers, HGST Ultrastar drives for CNS storage, and the VMware and Red Hat software we ran on the solution. The two solutions included the same software and compute servers. They differed in the configuration and costs of the storage servers. VMware software prices include license cost plus three-year support and Red Hat software prices include three-year subscriptions. Hardware prices do not include rack or networking costs, taxes, or shipping. We gathered this cost data in late December 2017 and early January 2018.

	HGST Ultrastar He[10] hard drive configuration	HGST Ultrastar SS200 solid-state drive configuration
Hardware		
Six HPE ProLiant DL380 Gen9 small form factor servers (base configuration with one SanDisk solid-state drive for operating system)	$142,410	$142,410
Three HPE ProLiant DL380 Gen9 large form factor servers (base configuration with one SanDisk solid-state drive for operating system)	$69,522	$69,522
15 Ultrastar SS200 1.92TB solid-state drives (five for each of three HPE ProLiant DL380 Gen9 small form factor servers)		$23,040
36 Ultrastar He[10] 10TB hard drives (12 for each of three HPE ProLiant DL380 Gen9 large form factor servers)	$15,123	
Hardware total	$227,055	$234,972
Software		
Three-year VMware software licensing and support for nine servers	$120,598	$120,598
Three-year Red Hat software licensing and support for nine servers	$290,700	$290,700
Software total	$411,298	$411,298
Total solution cost	**$638,353**	**$646,270**

Software costs

The two solutions have the same software costs. We include costs for the following:

- 18 licenses with three years of production support for VMware vSphere Enterprise Plus, one license per processor for the nine two-processor servers in the solution
- One license and three years of standard support for one instance of VMware vCenter Server to support the solution.
- Six three-year subscriptions for Red Hat OpenShift Container Platform, Premium (one license for each of the six two-processor servers running the software)
- One three-year subscription covering three nodes for Red Hat Container Storage Add-On for OpenShift Container Platform Premium

The solution requires Red Hat Enterprise Linux, which is included with the OpenShift Container Platform subscription.

Prices for VMware software are from https://www.vmware.com/products/vsphere.html. Red Hat provided us with the subscription prices for the OpenShift Container Platform software.

Software pricing	One-time license	Annual support or subscription	Three-year total	License count	Three-year solution totals
VMware					
VMware vSphere Enterprise Plus with Production Support	$3,495	$874	$6,117	18	$110,106
VMware vCenter Server Standard	$5,995	$1,499	$10,492	1	$10,492
Red Hat					
Red Hat OpenShift Container Platform, Premium		$15,000	$42,750	6	$256,500
Red Hat Container Storage Add-On for OpenShift Container Platform Premium (3 Nodes)		$12,000	$34,200	1	$34,200
Total					$411,298

Hardware costs

A third-party IT service provider provided the list prices for the HPE ProLiant DL380 Gen9 servers. These configurations matched as closely as possible the servers in our solution testbed but did not include the drives with which we tested. Red Hat provided the list prices for the five HGST Ultrastar SS200 1.92TB drives we added to each of the three storage servers in the solid-state drive configuration and for the 12 HGST Ultrastar He[10] 10TB drives we added to each of the three storage servers in the hard drive configuration.

We searched online for the price of the SanDisk Lightning Ascend Gen. II SAS solid-state drive that served as the OS drive for all servers. The solid-state drive configuration we priced differed slightly from our tested configuration in that we tested with a small form factor chassis that supported 24 solid-state drives but, because we used at most six of those slots, we instead priced a chassis that supports eight solid-state drives.

Six HPE ProLiant DL380 Gen9 small form factor servers

The two solutions used the same six SFF servers, each configured with a single SanDisk Lightning Ascend Gen. II SAS solid-state drive to serve as the OS drive. We estimate the cost of those six servers, not including the OS drive, at $22,385 based on list price quotes.

Qty	Product description
1	HPE ProLiant DL380 Gen9 8SFF configure-to-order server
2	HPE DL380 Gen9 Intel Xeon E5-2697Av4 (2.6GHz/16-core/40MB/145W) processors
8	HPE 32GB (1x32GB) Dual Rank x4 DDR4-2400 CAS-17-17-17 Registered Memory Kit
1	HPE H240ar 12Gb 2-ports Int FIO Smart Host Bus Adapter
1	HPE InfiniBand FDR/Ethernet 10Gb/40Gb 2-port 544+FLR-QSFP Adapter
2	HPE 800W Flex Slot Platinum Hot Plug Power Supply Kit
1	HPE iLO Advanced including 3yr 24x7 Tech Support and Updates 1-server LTU
1	HPE 2U Small Form Factor Easy Install Rail Kit
1	HPE 3Y Foundation Care NBD SVC
1	HPE ProLiant DL380 Gen9 Support

We added $1,350 for the OS drive.[1] The total list price for the compute servers was $23,735 each, $142,410 for all six.

1 https://www.disctech.com/SanDisk-SDLTODKM-800G-5CA1-800GB-SAS-SSD

Three HPE ProLiant DL380 Gen9 large form factor servers

We tested with three HPE ProLiant DL380 Gen9 LFF servers that supported a SanDisk Lightning Ascend Gen. II SAS solid-state drive to serve as the OS drive. A third-party IT service provider quoted $22,385 per server, not including the drives.

Qty	Product description
1	HPE ProLiant DL380 Gen9 12LFF Configure-to-order Server
2	HPE DL380 Gen9 Intel Xeon E5-2697v4 (2.3GHz/18-core/45MB/145W) Processors
8	HPE 32GB (1x32GB) Dual Rank x4 DDR4-2400 CAS-17-17-17 Registered Memory Kit
1	HPE H240ar 12Gb 2-ports Int FIO Smart Host Bus Adapter
1	HPE InfiniBand FDR/Ethernet 10Gb/40Gb 2-port 544+FLR-QSFP Adapter
2	HPE 800W Flex Slot Platinum Hot Plug Power Supply Kit
1	HPE iLO Advanced including 3yr 24x7 Tech Support and Updates 1-server LTU
1	HPE DL380 Gen9 12LFF SAS Cable Kit
1	HPE 2U Large Form Factor Easy Install Rail Kit
1	HPE 3Y Foundation Care NBD SVC
1	HPE ProLiant DL380 Gen9 Support

HGST Ultrastar SS200 solid-state drive configuration

We added five Ultrastar SS200 1.92TB drives to each of the three storage servers for this configuration. Western Digital provided the $1,536-per-drive list price for the Ultrastar SS200 1.92TB solid-state drives. The drives for all three servers added $23,040 to the price of the configuration.

HGST Ultrastar He^{10} hard drive configuration

Western Digital provided the $366 per unit list price for the Ultrastar He^{10} 10TB hard drives that we tested with. To support the drives, we also added an HPE Smart Array P440ar/2GB FBWC 12Gb 2-ports Int FIO SAS Controller to each of the three storage servers, priced at $649. These components for all three servers added $15,123 to the price of the configuration.

This project was commissioned by Red Hat.

Facts matter.®