



Technical Report

MongoDB on NetApp HCI

Bobby Oommen, Stephen Carl, NetApp
January 2019 | TR-4633

Abstract

This document introduces the NetApp® HCI solution to infrastructure administrators and provides important design paradigms to consider when using the NetApp HCI solution for NoSQL databases. The document talks about use cases that are ideally suited for HCI and discusses architecture considerations for applications running in the context of NetApp HCI. By following the guidelines in this document, you can learn how to effectively design, implement, and run a MongoDB database on NetApp HCI.

TABLE OF CONTENTS

| | | |
|----------|--|-----------|
| 1 | Introduction | 4 |
| 1.1 | Performance Guarantee | 4 |
| 1.2 | Enterprise Scale | 4 |
| 1.3 | Streamline Operation | 4 |
| 1.4 | Configuration | 4 |
| 2 | NetApp HCI Use Cases | 6 |
| 2.1 | Workload Consolidation | 6 |
| 2.2 | Development and Testing | 6 |
| 3 | MongoDB Storage Configuration | 6 |
| 3.1 | Create Account | 6 |
| 3.2 | Create Volume | 7 |
| 3.3 | Create Initiators | 8 |
| 3.4 | Create Volume Access Groups | 8 |
| 3.5 | Add Volumes to Access Groups | 9 |
| 4 | Operating System Configuration | 10 |
| 4.1 | Add MongoDB Packages | 10 |
| 4.2 | Update Kernel Parameters | 10 |
| 4.3 | Optimize Network Performance | 10 |
| 4.4 | Optimize iSCSI Parameters | 10 |
| 4.5 | Tune I/O Scheduler | 11 |
| 4.6 | Configure Multipath Driver | 11 |
| 4.7 | Enable Multipathing | 12 |
| 4.8 | Disable Transparent Hugepages and Defrag | 12 |
| 4.9 | Set ulimits | 12 |
| 4.10 | Set Up Logical Volume Manager Method | 12 |
| 4.11 | Create SolidFire Volume Using Non-LVM Method | 13 |
| 5 | MongoDB Configuration | 13 |
| 5.1 | Install MongoDB Enterprise | 13 |
| 5.2 | Configure MongoDB Sharded Cluster | 14 |
| 6 | Backup and Recovery Using Storage Snapshot Copies | 14 |
| 7 | MongoDB Database Cloning | 17 |
| 8 | Performance Testing | 20 |

| | |
|---|-----------|
| 9 Conclusion | 24 |
| Appendix A: Additional Setup Information | 24 |
| Appendix B: mongod.conf for Sharded replicaSet Members | 26 |
| Sample mongod.conf for Sharded replicaSet Members | 26 |
| Appendix C: mongod.conf for configReplSet..... | 27 |
| Sample mongod.conf for the configReplSet | 27 |
| Appendix D: Mongos Config File | 27 |
| Mongos Config File..... | 27 |
| Where to Find Additional Information | 28 |
| Version History | 28 |

LIST OF TABLES

| | |
|---|----|
| Table 1) NetApp HCI configuration storage nodes. | 5 |
| Table 2) NetApp HCI configuration compute nodes. | 5 |
| Table 3) MongoDB nodes for clusters. | 22 |
| Table 4) YCSB client server. | 22 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1) Minimum configuration. | 5 |
| Figure 2) Successful Snapshot rollback to Mongo node. | 17 |
| Figure 3) Create volume clone. | 18 |
| Figure 4) MongoDB and NetApp HCI test environment for replicaSet cluster. | 21 |
| Figure 5) MongoDB and NetApp HCI test environment for sharded cluster. | 21 |
| Figure 6) YCSB workloads for three-node replicaSet cluster. | 22 |
| Figure 7) YCSB workload C for three-node replicaSet cluster..... | 23 |
| Figure 8) YCSB workloads for three-node sharded cluster. | 23 |

1 Introduction

NetApp HCI is an enterprise-scale hybrid converged infrastructure solution ideally suited for customers who are looking to break free from first-generation HCI limitations.

NetApp HCI customers can run multiple applications with guaranteed performance to confidently deploy resources across your entire data center. The architecture allows you to deploy your infrastructure by simplifying management and independently scale both compute and storage resources. NetApp HCI is Data Fabric ready out of the box for easy access to all your data across any public, private, or hybrid cloud. By moving to NetApp HCI, IT organizations can transform their data center, driving operational efficiencies and reducing costs.

Data Fabric is a software-defined approach from NetApp for data management that enables businesses to connect disparate data management and storage resources. NetApp HCI can streamline data management between on-premises and cloud storage for enhanced data portability, visibility, and protection.

1.1 Performance Guarantee

A common challenge for a data center is delivering predictable performance, complicated even more by running multiple applications sharing the same infrastructure. An application interfering with other applications creates performance degradations, causing IT administrators to spend valuable time troubleshooting the environment. Mainstream applications, such as virtual desktop infrastructure (VDI) and database applications, have unique I/O patterns that can during normal operations affect one another's performance when deployed in a shared environment. NetApp's HCI quality of service (QoS) feature allows fine-grained control of performance for every application, eliminating noisy neighbors, meeting unique performance needs, and satisfying performance SLAs. The storage architecture that is part of the NetApp HCI solution eliminates performance variance in the context of data locality because the data is distributed across all the nodes in the HCI cluster.

1.2 Enterprise Scale

Unlike previous generations of HCI that have fixed resource ratios, NetApp HCI scales compute and storage resources independently. Independent scaling avoids costly and inefficient overprovisioning, and simplifies capacity and performance planning. Running on innovative SolidFire® technology and delivered on a NetApp designed architecture, NetApp HCI is an enterprise-scale hybrid converged infrastructure solution. NetApp HCI comes in 2RU x 4-node building blocks (chassis) in mix-and-match small, medium, and large storage and compute configurations and can be scaled to allow you to rapidly meet changing business needs and scale on your terms.

1.3 Streamline Operation

A common goal of IT organizations is to automate all routine tasks, eliminating the risk of user errors associated with manual operations and allowing valuable resources to be focused on higher value priorities that drive business efficiencies. The NetApp Deployment Engine (NDE) streamlines day 0 installation from hours to minutes, while simple centralized management through the vCenter plug-in gives you full control of managing your entire infrastructure through an intuitive UI. A robust suite of APIs enables additional seamless integration into higher-level management, orchestration, backup, and disaster recovery tools.

1.4 Configuration

NetApp HCI is available with configuration options for both compute and storage. The nodes are similar to a small blade that sits inside a chassis. A minimum starting configuration must have four storage nodes and two compute nodes.

Diagram illustrating a 2RU 4 Node Chassis configuration. The chassis is divided into two main sections, each containing two rows of components:

- Top Section:**
 - Row 1: Flash Storage (Red)
 - Row 2: Compute w/Hypervisor (Blue)
- Bottom Section:**
 - Row 1: Flash Storage (Red)
 - Row 2: Open (Grey)

Labels and connections:

- 2RU 4 Node Chassis:** Indicated by a vertical arrow on the left side.
- Compute Node:** Indicated by a horizontal arrow pointing to the top section.
- All Flash Storage:** Indicated by a horizontal arrow pointing to the bottom section.
- Grow at 1RU, Half-Width:** Indicated by a horizontal arrow pointing to the right side of the bottom section.

Table 1) NetApp HCI configuration storage nodes.

| | H410S | H610S |
|--------------------------|---------------------------------|----------------------------------|
| RU | 2RU, half-width | 1RU, half-width |
| Drive Capacity | 480GB/960GB/1.92TB | 960GB/1.92TB/3.84TB |
| Performance per node | 50,000 IOPS - 100,000 IOPS | 100,000 IOPS |
| SSD | 6x encrypting or non-encrypting | 12x encrypting or non-encrypting |
| Effective block capacity | 5.5TB–44TB | 20TB–80TB |

| | H410C | H610C |
|-----------------|--|---|
| RU | 2RU, half-width | 1RU |
| Cores for VMs | 8-40 | 32 |
| CPU | 2x Intel Xeon Gold 5122, 4 cores, 3.6GHz 2x Intel Xeon Silver 4110, 8 cores, 2.1GHz 2x Intel Xeon Gold 5120, 14 cores, 2.2GHz 2x Intel Xeon Gold 6138, 20 cores, 2.0GHz | 2x Intel Xeon Gold 6130, 16 cores, 2.1GHz, 2x NVIDIA Tesla M10 GPU cards |
| Memory | 384GB – 1TB | 512GB |
| Base networking | 4x 25/10GbE SFP28/SFP+ 2x 1GbE RJ-45 | 2x 25/10GbE (SFP28) 2x 1GbE RJ-45 |

As an example, a minimum size starting solution would be a configuration with two small compute and four small storage nodes. As requirements change, you can add more compute or storage nodes of any size to the chassis independently of each other. This flexibility of adding only compute or only storage nodes enables unique scalability options for building an efficient and agile cloud in your data center for various use cases.

2 NetApp HCI Use Cases

NetApp HCI can support a wide range of application use cases. This section shows how to identify use cases that are a good fit for an HCI solution.

2.1 Workload Consolidation

End user behaviors and applications' changing requirements are difficult to predict. A key to any workload consolidation is the ability to guarantee some minimum level of performance. The NetApp HCI per-volume QoS control allows individual applications to meet IOPS and throughput requirements without affecting other applications running on the system in parallel. With QoS, administrators can deploy and isolate multiple workloads. In this paper, multiple MongoDB database cluster instances run guaranteed performance without affecting each other.

Administrators have full control of each storage volume on which the database data resides and can perform all maintenance operations, including setting the QoS through the vCenter plug-in. Administrators can also use NetApp HCI REST APIs to further automate and make storage management even easier.

For more information about QoS and mixed workloads, see the [NetApp HCI and mixed workloads](#) technical report.

2.2 Development and Testing

The NetApp Snapshot® technology, which is available on the storage nodes, provides an easy way to deliver a point-in-time view of the contents of an active file system or storage volume. NetApp HCI Snapshot copies can be used for delivering a rapid space-efficient copy of an existing database, which can be used for development and test purposes requiring new database instances. The cloning capability, when coupled with the built-in QoS controls, can create clones without performance impacts on the upstream applications. The CopyVolume feature of storage nodes allows a refresh of an existing cloned copy of a database without file system remount operations. In this use case, you frequently refresh a copy of the database by only taking changes from the production copy.

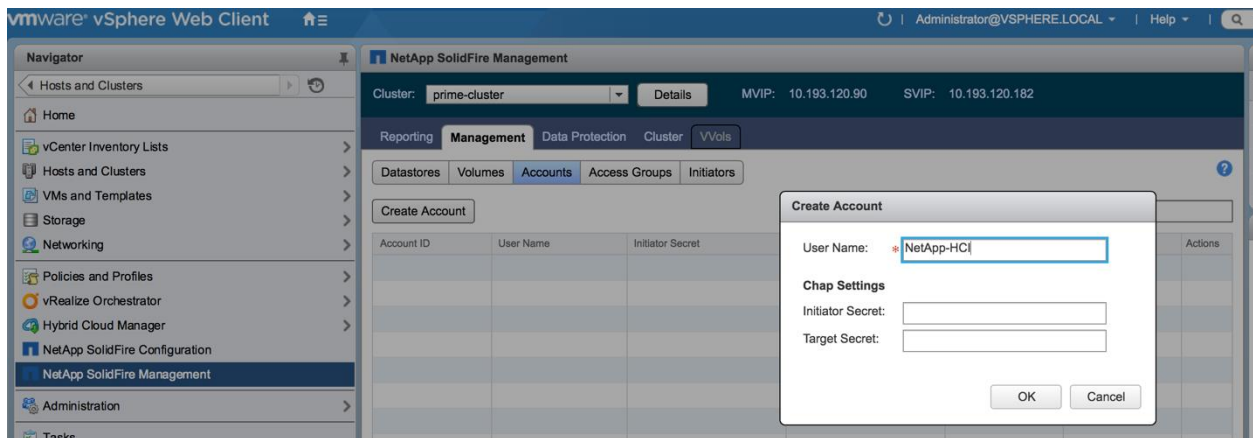
3 MongoDB Storage Configuration

The following sections focus on how to configure an environment of both storage and servers to support a MongoDB database cluster. NetApp recommends that you have all the database components on the storage node. NetApp supports presenting the storage in a 4K sector size (native mode) and in a 512-byte sector size (512e).

3.1 Create Account

To create an account, complete the following steps:

1. Log in to the vCenter as an administrator.
2. Select Home > NetApp SolidFire Management.
3. In the NetApp SolidFire Management pane, select Management > Account > Create Account. The Create Account window appears.



4. Enter a user name; in this case, NetApp-HCI.
5. In the CHAP Settings section, enter the following information:
 - Initiator secret for CHAP node session authentication
 - Target secret for CHAP node session authentication

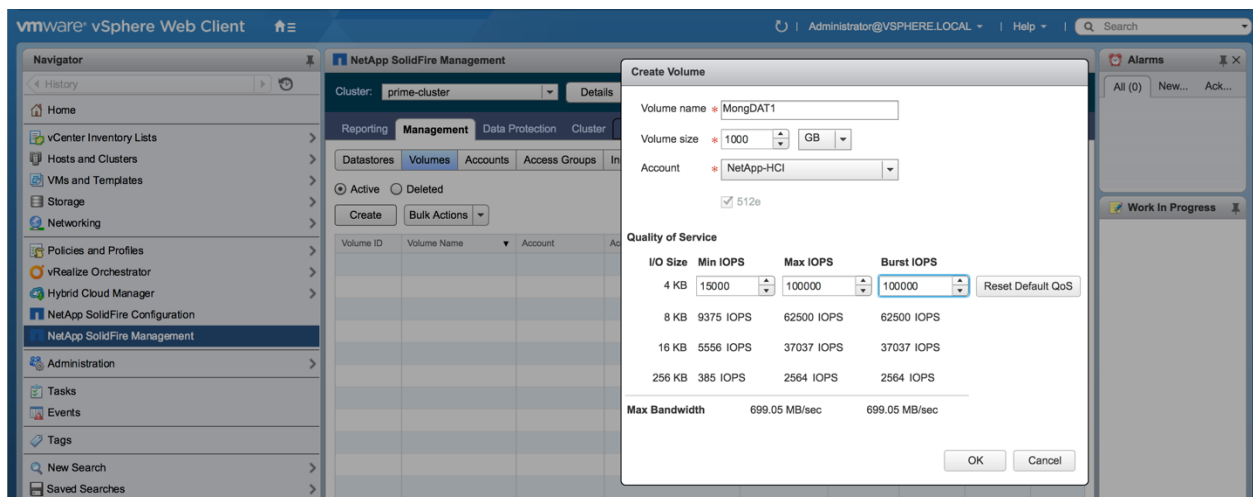
Note: Leave the credentials field blank if you want the passwords to be generated automatically.
6. Click Create Account.

Note: If an account with the same name exists, you get an error message.

3.2 Create Volume

To create a volume, complete the following steps:

1. Log in to the vCenter as an administrator.
2. Select Home > NetApp SolidFire Management.
3. In the NetApp SolidFire Management pane, select Management > Volumes > Create. The Create Volume window appears.



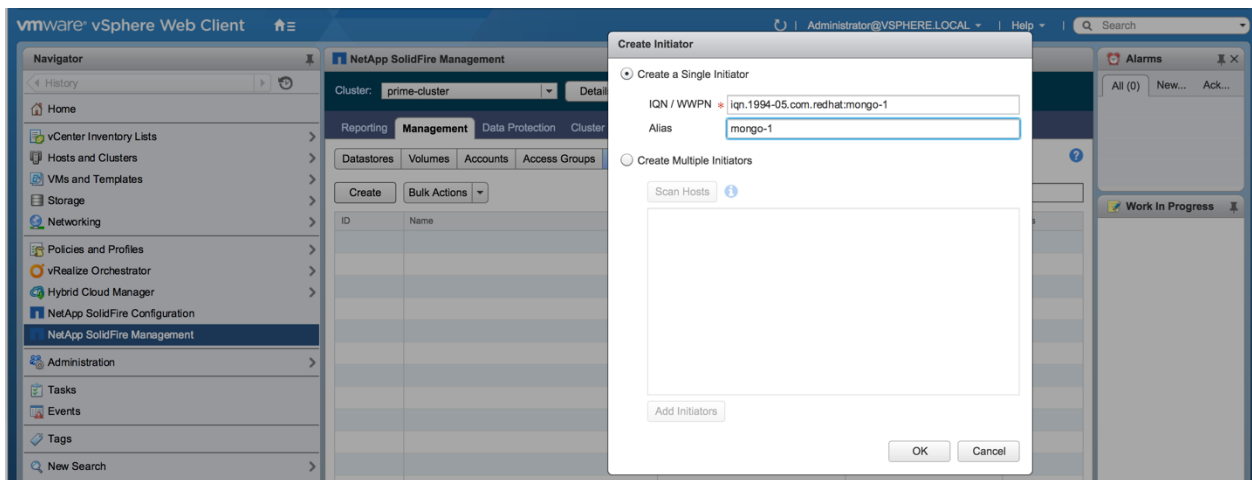
4. Enter the volume name (1 to 64 characters in length). For example, enter the name MongDAT1.
5. Enter the size of the volume.

6. Click the Account drop-down list and select the account that should have access to the volume. In this case, select NetApp-HCI.
7. Enter QoS values. For this setup, the following values were chosen: Min 15K, Max 100K, and Burst.
8. Click Create Volume.
9. Repeat steps 1 through 8 for all the remaining volumes for all the replica sets.

3.3 Create Initiators

To create an initiator, complete the following steps:

1. Log in to the vCenter as an administrator.
2. Select Home > NetApp SolidFire Management.
3. In the NetApp SolidFire Management pane on the right, select Management > Initiators > Create. The Create initiator window appears.



4. From the Windows host, get the initiator name and type it in the IQN/WWPN field; optionally, provide an alias as well.
5. Click OK.

3.4 Create Volume Access Groups

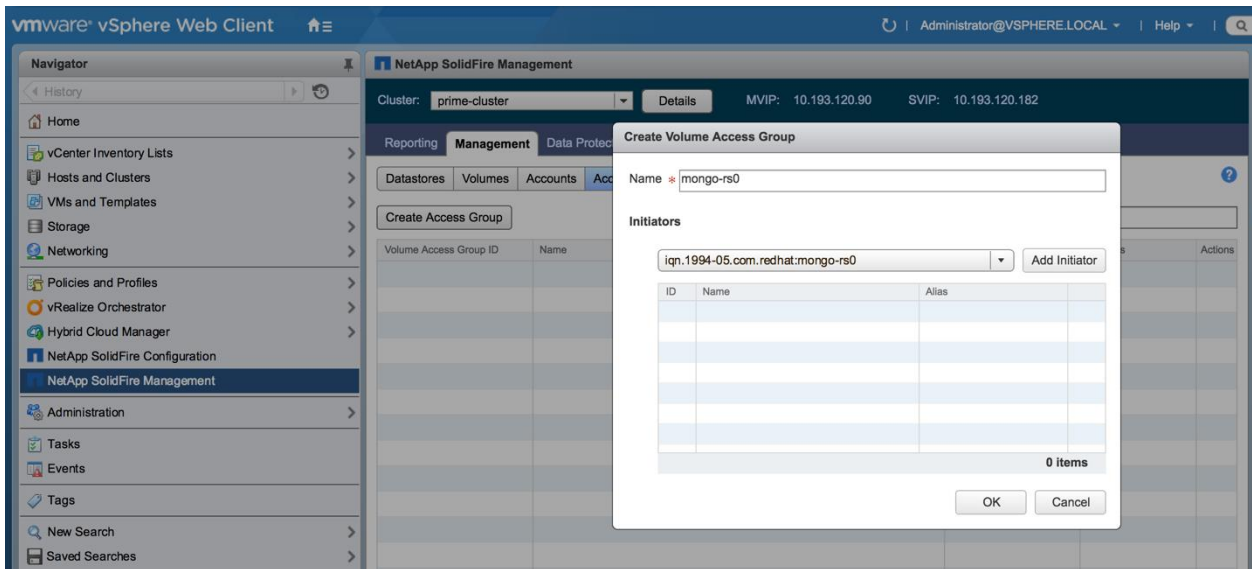
Volume access groups limit connectivity from designated host servers based on a unique identifier, whereas CHAP authentication uses secret keys for unidirectional or bidirectional authentication. In this document, initiator iSCSI qualified names (IQNs) are used to access the volumes.

Volume access groups have the following system limits:

- They can have a maximum of 64 IQNs.
- An IQN can belong to only one access group.
- A single volume can belong to a maximum of four access groups.

To create volume access groups, complete the following steps:

1. Log in to the vCenter as an administrator.
2. Select Home > NetApp SolidFire Management.
3. In the NetApp SolidFire Management pane on the right, select Management > Access Groups > Create Access Group.
4. The Create Access Group window appears.

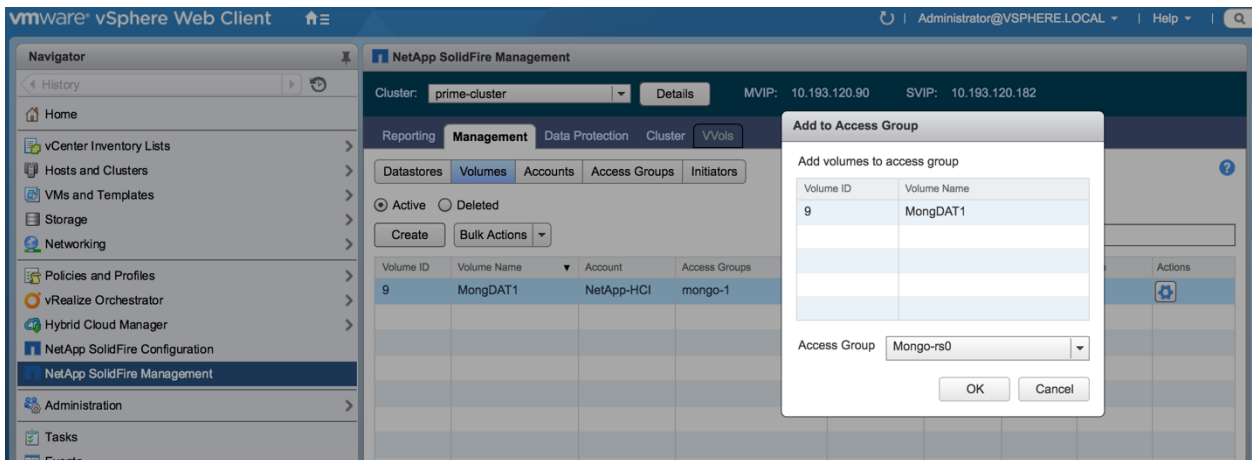


5. Type in the Access group name, mongo-rs0 in this case.
6. From the drop-down list, select the initiator.
7. Click OK.
8. Repeat steps 3 through 7 for all the replica sets.

3.5 Add Volumes to Access Groups

To add the volumes to the access group, complete the following steps:

1. Select Management > Volumes.
2. From the listed volumes, select the volume that is part of the Mongo-rs0.
3. After the volume is selected, click the Bulk Actions drop-down list.
4. Select Add to Access Group. The Add to Access group window appears.



5. From the Access Group drop-down list, select the access group Mongo-rs0.
6. Click OK.
7. Repeat steps 1 through 6 for all the replica sets.

4 Operating System Configuration

The guidelines in this document apply to Linux distributions of the MongoDB Enterprise edition. Ubuntu version 16.04 LTS was used in this document's testing. Alternate distributions can be used if they have full compatibility with the MongoDB software.

4.1 Add MongoDB Packages

After installing the base operating system, you might need to add more packages to update the operating system to meet MongoDB installation requirements. For information about how to meet these requirements, see the latest MongoDB documentation.

4.2 Update Kernel Parameters

Update the kernel parameters for your host operating system to the following values:

```
vm.dirty_ratio = 15
vm.dirty_background_ratio = 5
vm.swappiness = 1
net.core.somaxconn = 4096
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_intvl = 30
net.ipv4.tcp_keepalive_time = 120
net.ipv4.tcp_max_syn_backlog = 4096
```

4.3 Optimize Network Performance

Consider the following guidelines for optimal network performance:

- Enable jumbo frames for all host network interfaces.
- To isolate the data traffic, configure the interface that is used for the MongoDB data traffic with a different subnet from the public network.

4.4 Optimize iSCSI Parameters

The Linux iSCSI initiator configuration works with NetApp SolidFire volumes in its default configuration. To maximize system throughput, increase the number of sessions per target (`nr_sessions`) from the default of 1 to 8.

1. Make the following changes to the iSCSI daemon in the `/etc/iscsi/iscsid.conf` file:

```
iscsid.startup = /etc/rc.d/init.d/iscsid force-start
node.startup = automatic
node.leading_login = No
node.session.timeo.replacement_timeout = 120
node.conn[0].timeo.login_timeout = 15
node.conn[0].timeo.logout_timeout = 15
node.conn[0].timeo.noop_out_interval = 5
node.conn[0].timeo.noop_out_timeout = 5
node.session.err_timeo.abort_timeout = 15
node.session.err_timeo.lu_reset_timeout = 30
node.session.err_timeo.tgt_reset_timeout = 30
node.session.initial_login_retry_max = 8
node.session.cmds_max = 128
node.session.queue_depth = 32
node.session.xmit_thread_priority = -20
node.session.iscsi.InitialR2T = No
node.session.iscsi.ImmediateData = Yes
node.session.iscsi.FirstBurstLength = 262144
node.session.iscsi.MaxBurstLength = 16776192
node.conn[0].iscsi.MaxRecvDataSegmentLength = 262144
node.conn[0].iscsi.MaxXmitDataSegmentLength = 0
discovery.sendtargets.iscsi.MaxRecvDataSegmentLength = 32768
node.conn[0].iscsi.HeaderDigest = None
```

```
node.session.iscsi.FastAbort = Yes
node.startup = automatic
node.session.nr_sessions = 8
```

2. Make discovery of iSCSI devices persistent over reboots.

```
chkconfig iscsid
```

3. To rescan the new storage volumes, run the following commands:

```
iscsiadm -m discovery -t sendtargets -p <SolidFire SVIP> --op update -n node.session.nr_sessions -v 2
iscsiadm -m node -L all
```

4.5 Tune I/O Scheduler

1. Run the following commands to tune the Linux operating system to take advantage of the performance characteristics of the SolidFire storage system (<devpath> is the device name).

```
echo 0 > /sys/<devpath>/queue/rotational
echo noop > /sys/<devpath>/queue/scheduler
echo 128 > /sys/<devpath>/queue/nr_requests
echo 2 > /sys/<devpath>/queue/rq_affinity
echo 0 > /sys/<devpath>/queue/add_random
```

2. To persist reboots, create file `/lib/udev/rules.d/99-solidfire.rules` and the following entries:

```
ACTION=="add", \
SUBSYSTEMS=="scsi", \
KERNEL=="sd*", \
ATTRS{vendor}=="SolidFir", \
ATTR{queue/scheduler}="noop", \
ATTR{queue/add_random}="0", \
ATTR{queue/rq_affinity}="2", \
ATTR{queue/nr_requests}="1024", \
ATTR{queue/max_sectors_kb}="2048"
```

4.6 Configure Multipath Driver

1. Install and configure the Linux multipath driver (`multipathd`) by making the following changes to the `/etc/multipath.conf` file.

```
defaults {
    user_friendly_names yes
}

devices {
    device {
        vendor "SolidFir"
        product "SSD SAN"
        path_grouping_policy multibus
        path_checker tur
        hardware_handler "0"
        failback immediate
        rr_weight uniform
        rr_min_io 10
        rr_min_io_rq 10
        features "0"
        no_path_retry 24
        prio const
    }
}
```

2. Optionally, you can enable persistent mapping of `/dev/mapper` entries by associating the NetApp SolidFire storage system device's worldwide identifier (WWID) with a specific operating system alias. For this option, make the following additions to the `/etc/multipath.conf` file:

```

multipaths {

multipath {
wwid 36f47acc100000000707a646c000003b1
alias mongo-rs0
}
}

```

4.7 Enable Multipathing

1. To enable multipathing on Red Hat Linux distributions, run the following command:

```
systemctl enable multipathd.service
```

2. You can check the status of the multipath daemon with the following command:

```
systemctl list-unit-files|grep multipath
multipathd.service      enabled
```

3. For Ubuntu distributions, install multipath tools:

```
sudo apt-get install multipath-tools
```

4.8 Disable Transparent Hugepages and Defrag

Per MongoDB recommendations, disable hugepages and defrag in the kernel.

```

root@ubuntu-6:~# echo never > /sys/kernel/mm/transparent_hugepage/enabled
root@ubuntu-6:~# echo never > /sys/kernel/mm/transparent_hugepage/defrag

```

To disable permanently and persistent for reboots, see the necessary steps in the following link: [disable transparent hugepages](#).

4.9 Set ulimits

The architecture of MongoDB delegates memory and paging functionality to the operating system. Therefore, you should increase the `ulimit` open file settings from the default value of 4096. In addition, you should set the number of allowed processes to an optimal value for MongoDB. Edit the `/etc/security/limits.conf` file and create the `/etc/security/limits.d/99-mongodb-nproc.conf` file if needed. Add the following `ulimit` values:

```

mongod soft nproc 64000
mongod hard nproc 64000
mongod soft nofile 64000
mongod hard nofile 64000

```

4.10 Set Up Logical Volume Manager Method

For Linux distributions using Logical Volume Manager (LVM), set up an LVM to stripe the data across multiple SolidFire volume devices by completing the following steps:

1. Create a volume group by using the multipath devices `mpatha` and `mpathb`.

```
vgcreate mongovg /dev/mapper/mpatha /dev/mapper/mpathb
```

2. Create a logical volume on this volume group.

```
lvcreate -l 100%FREE -n mongolv mongovg
```

3. Create an XFS file system.

```
mkfs.xfs /dev/mongovg/mongolv
```

4. Create a database directory path for the MongoDB replicas and mount the file system.

```
mkdir -p /data/db/  
mount -t xfs -o nobarrier,discard,noatime /dev/mongovg/mongolv /data/db/
```

Note: Before you install MongoDB, you should set appropriate permissions so that the `mongod` user can access the devices on the SolidFire system. You can set permissions on multipath devices by creating a `udev` rule file that allows appropriate access to the devices.

4.11 Create SolidFire Volume Using Non-LVM Method

1. Create a disk device partition by using the multipath device `mpatha` as shown in this [Configuring SolidFire for Linux](#) guide.

```
root@nosql-ubuntu-6:~# ls -al /dev/mapper/mpath*  
lrwxrwxrwx 1 root root 7 Aug 25 12:31 /dev/mapper/mpatha -> ../dm-2  
lrwxrwxrwx 1 root root 7 Aug 25 12:31 /dev/mapper/mpatha-part1 -> ../dm-3
```

2. Create the file system.

```
mkfs.xfs -K /dev/dm-3
```

3. Update `/etc/fstab` to mount the file system and persist through host reboots.

```
UUID="686277b8-b711-4fc8-b021-3e6430f4a358" /data/db xfs  
noatime,discard,nobarrier,_netdev 0 0
```

4. Set permissions on multipath devices by creating a `udev` rule file that allows appropriate access to the devices by user `mongod`.

5 MongoDB Configuration

5.1 Install MongoDB Enterprise

The server that performs the installations should be configured per the appropriate MongoDB installation files, and any other dependencies can be installed. For this installation, version 3.4 was chosen. For information about the latest stable release, see the [MongoDB installation](#) documents before you start the installation process.

To install the software for Red Hat Linux distributions, complete the following steps:

1. Configure the `yum` repository.
2. Create an `/etc/yum.repos.d/mongodb-enterprise.repo` file by copying the following commands so that the MongoDB Enterprise packages can be installed directly by using `yum`.

```
name=MongoDB Enterprise Repository  
baseurl=https://repo.mongodb.com/yum/redhat/$releasever/mongodb-enterprise/3.4/$basearch/  
gpgcheck=1  
enabled=1 gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc
```

3. Install MongoDB Enterprise.
4. Install the MongoDB Enterprise packages and dependencies by using the `yum` command.

```
yum install -y mongodb-enterprise
```

To install MongoDB on Ubuntu, complete the following steps using advanced packaging tool (`apt`) for the latest MongoDB enterprise version.

1. Generate key:

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv  
0C49F3730359A14518585931BC711F9BA15703C6
```

2. Create a `/etc/apt/sources.list.d/mongodb-enterprise.list` file for MongoDB:

```
$ echo "deb [ arch=amd64,arm64,ppc64el,s390x ] http://repo.mongodb.com/apt/ubuntu xenial/mongodb-enterprise/3.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-enterprise.list
```

3. Reload package database:

```
$ sudo apt-get update
```

4. Install MongoDB Enterprise packages:

```
$ sudo apt-get install -y mongodb-enterprise
```

5.2 Configure MongoDB Sharded Cluster

Set Up Sharded Cluster with `mongod.conf` Files

1. Configure the MongoDB instance.

`/etc/mongod.conf` is used to configure the MongoDB instance. The settings might differ based on the installations. For an example, see Appendix A.

2. Start MongoDB instances.

Use the `systemctl` command to start the MongoDB instance. Verify that the logs that are available in `/var/log/mongodb/mongod.log` do not have any errors.

3. Start the replica, config, and Mongos/query router servers.

```
systemctl start mongod
```

4. Set up all the remaining replica servers by following section 4 and section 5.1 (steps 1 through 6) of this document.

5. Start the `mongo` Shell.

After the MongoDB instance is up and running, you can create replica sets for each shard. For this setup, three replica sets (one primary and two secondary) were configured on each shard, and a total of four shards were configured. To create the replica `rs0`, run the `mongo` Shell with the following command:

```
mongo --host mongo-rs0
```

6. Create a replica set.

To create the replica set `rs0`, run the following commands in the `mongo` Shell. You can check the status of the replica set by running the `rs.status()` command in the `mongo` Shell. A sample output is given in Appendix B.

```
rs.initiate()  
rs.add("mongo-rs1")  
rs.add("mongo-rs2")
```

For more configuration information about the `mongod.conf` file and configuring the sharded cluster, see Appendix A.

6 Backup and Recovery Using Storage Snapshot Copies

Point-in-time (PiT) Snapshot copies can be taken on the SolidFire array for the volumes that are part of the Mongo cluster and are used to recover the database in the event of a data corruption or media failure. Both data and journal files can be in the same SolidFire volume for the PiT recovery to work. SolidFire does support having multiple volumes for the Mongo database and makes sure that all the volumes that are part of the database have the same consistency point during the group Snapshot copy or for a single-volume Snapshot copy.

See [MongoDB best practices on NetApp SolidFire](#) for more information about group Snapshot copies and other information used in this document's testing.

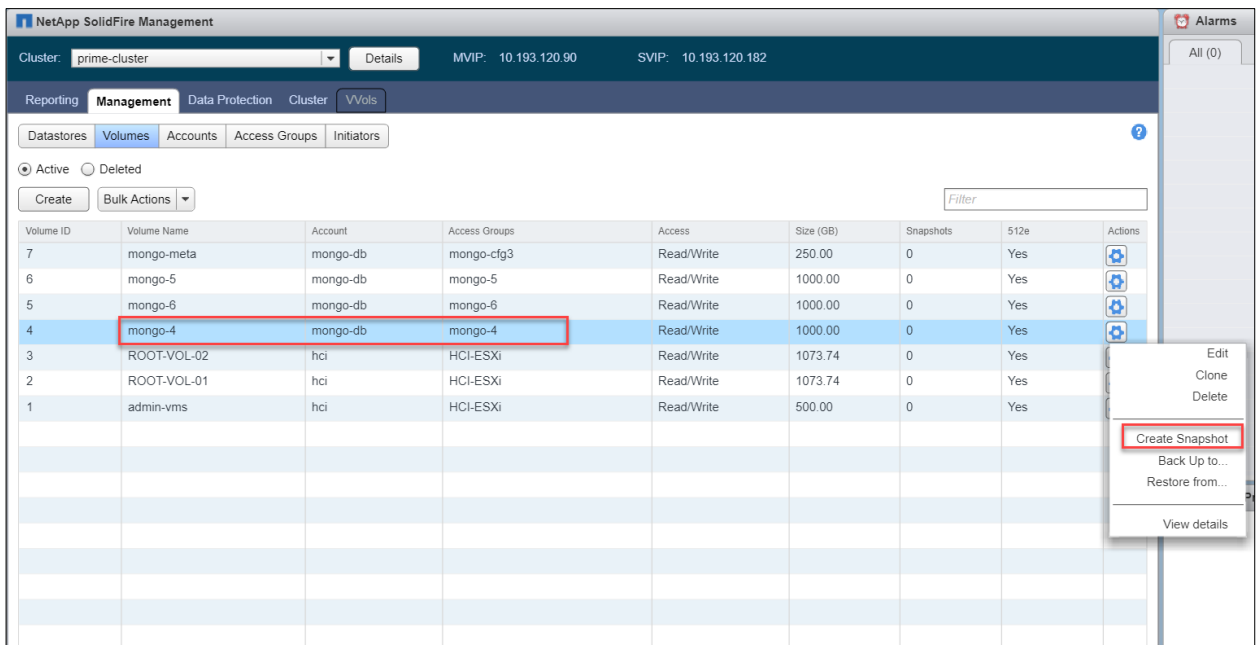
A Snapshot copy in this test environment was made of a single volume of the primary MongoDB sharded cluster node using the NetApp management interface from the vCenter main menu.

The database recovery can be done easily by reverting the volumes on the NetApp HCI SolidFire system using a Snapshot copy. Follow these steps to recover the database:

1. Log in to the mongo cluster and stop the mongod instance on the primary cluster node mongo-4 and unmount the file systems that are part of the database:

```
stop mongod
umount /data/db
```

2. Log in to the NetApp SolidFire management UI.
3. Click Volumes > Create Snapshot.
4. Select the volumes that are part of the database.



The screenshot shows the NetApp SolidFire Management interface. The 'Volumes' tab is active, displaying a table of volumes. The volume 'mongo-4' is selected, and a context menu is open, highlighting the 'Create Snapshot' option.

| Volume ID | Volume Name | Account | Access Groups | Access | Size (GB) | Snapshots | 512e | Actions |
|-----------|-------------|----------|---------------|------------|-----------|-----------|------|---------|
| 7 | mongo-meta | mongo-db | mongo-cfg3 | Read/Write | 250.00 | 0 | Yes | [Icon] |
| 6 | mongo-5 | mongo-db | mongo-5 | Read/Write | 1000.00 | 0 | Yes | [Icon] |
| 5 | mongo-6 | mongo-db | mongo-6 | Read/Write | 1000.00 | 0 | Yes | [Icon] |
| 4 | mongo-4 | mongo-db | mongo-4 | Read/Write | 1000.00 | 0 | Yes | [Icon] |
| 3 | ROOT-VOL-02 | hci | HCI-ESXi | Read/Write | 1073.74 | 0 | Yes | [Icon] |
| 2 | ROOT-VOL-01 | hci | HCI-ESXi | Read/Write | 1073.74 | 0 | Yes | [Icon] |
| 1 | admin-vms | hci | HCI-ESXi | Read/Write | 500.00 | 0 | Yes | [Icon] |

5. Select the volume from the Create Snapshot UI.
6. Configure the Snapshot information.

mongo-4 - Create Snapshot

Volume name: mongo-4
Volume ID: 4
IQN: iqn.2010-01.com.solidfire:j8fh.mongo-4.4
Account: mongo-db
Size: 1000.00 GB
512e: Yes

General

Snapshot name

☐ Include snapshot in replication when volume is paired

Retention

☒ Keep forever
☐ Set retention period

Schedule

☒ Take snapshot now
☐ Create snapshot schedule

7. The Snapshot copy SnapPrimaryNode is created.

NetApp SolidFire Management

Cluster: prime-cluster Details MVIP: 10.193.120.90 SVIP: 10.193.120.182

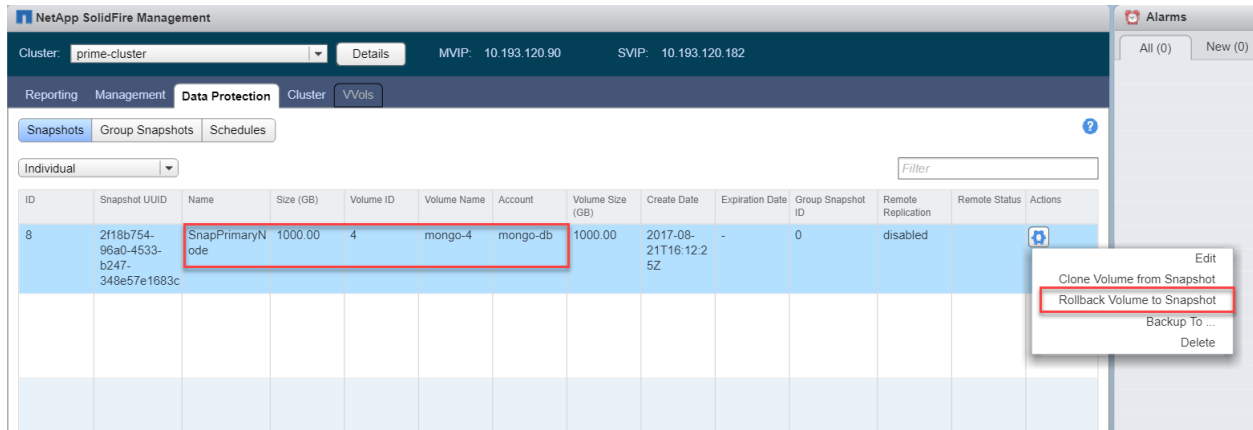
Reporting Management **Data Protection** Cluster VVols

Snapshots Group Snapshots Schedules

Individual Filter

| ID | Snapshot UUID | Name | Size (GB) | Volume ID | Volume Name | Account | Volume Size (GB) | Create Date | Expiration Date | Group Snapshot ID | Remote Replication |
|----|--------------------------------------|-----------------|-----------|-----------|-------------|----------|------------------|----------------------|-----------------|-------------------|--------------------|
| 8 | 2f18b754-96a0-4533-b247-348e57e1683c | SnapPrimaryNode | 1000.00 | 4 | mongo-4 | mongo-db | 1000.00 | 2017-08-21T16:12:25Z | - | 0 | disabled |

Note: The /data/db directory on primary server mongo-4 is deleted, and the mongod processes are shut down. The mongo-5 secondary is now promoted to the primary server. The /data/db directory is restored from the Snapshot copy we created into a new instance of the mongo-4 server. First the volume is rolled back to the Snapshot copy.



8. Click Rollback Volume to Snapshot.
9. After the volume is reverted successfully, log in to the Mongo instance on “mongo-4” server.
10. Mount the file system backup and start the mongod instance.

```
mount /data/db
start mongod
```

Figure 2) Successful Snapshot rollback to Mongo node.

```
{ "os": { "type": "Linux", "name": "Ubuntu", "architecture": "x86_64", "version": "16.04" } }
[NetworkInterfaceASIO-Replication-0] Connecting to mongo-4:37000
[NetworkInterfaceASIO-Replication-0] Successfully connected to mongo-4:37000, took 1ms (
[ReplicationExecutor] Member mongo-4:37000 is now in state SECONDARY
[thread1] connection accepted from 10.193.121.79:38326 #18 (5 connections now open)
```

Note: The /data/db directory on server mongo-4 is rolled back to the Snapshot copy SnapPrimaryNode, and the node is now a secondary cluster member. A different mongo node was promoted to the primary server role while the mongod process was shut down and the Snapshot rollback was executed. The replicaSet is healthy and intact. The mongo-4 node can be manually promoted back to the primary node with mongo shell commands if necessary.

7 MongoDB Database Cloning

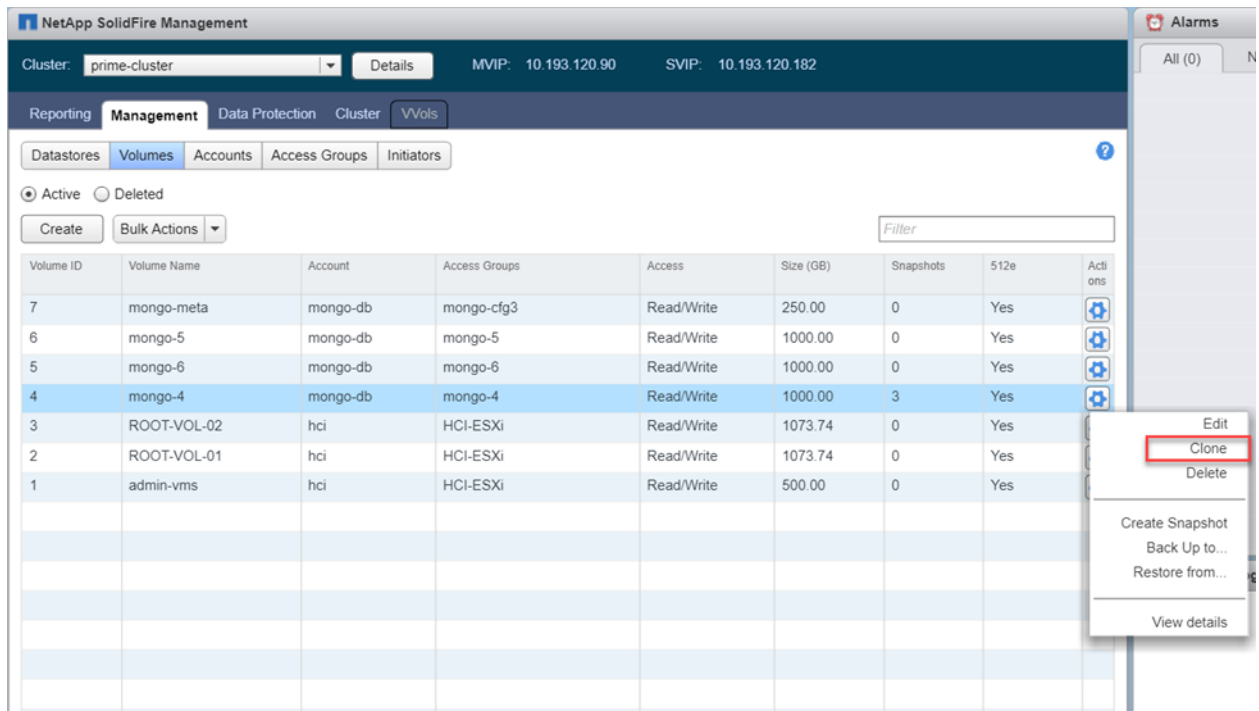
Having the ability to create space- and time-efficient usable database copies quickly and with virtually no effect on the production system is important. SolidFire volume clone is a proven technology that helps database and system administrators deliver a near-instantaneous, space-efficient, point-in-time copy of the production database. Traditional methods of copy process pose various challenges, including system downtime and degraded performance during the cloning process, and a large amount of storage space is usually needed to store each clone. The SolidFire volume cloning process is completed quickly, with virtually no performance effect on the production system. The cloned database is similar to the primary replica copy of a Mongo database cluster, and unlike the traditional database clone, it consumes no extra disk space at the time of creation.

In case the secondary replica copy is corrupted or you would like to add another secondary replicaSet member of the Mongo instance, this can be done as follows.

In this example, we are cloning a volume from the primary MongoDB server mongo-4 with the intention of creating a new secondary replicaSet member for server mongo-7 with cloned volume named mongo-7.

In this example, we are cloning a volume from the primary MongoDB server mongo-4 with the intention of creating a new secondary replicaSet member for server mongo-7 with the cloned volume to a newly named volume mongo-7.

Figure 3) Create volume clone.



1. Select the mongo-4 volume from the Volume UI.
2. Click the actions button and select clone.
3. Create the volume clone.

mongo-4 - Clone Volume

Volume name: mongo-4
 Volume ID: 4
 IQN: iqn.2010-01.com.solidfire:j8fh.mongo-4.4
 Account: mongo-db
 Size: 1000.00 GB
 512e: Yes

New Cloned Volume Details

Volume name * mongo-7
 Volume size * 1000 GB
 Access * Read / Write
 Account * mongo-db

OK Cancel

- We then need to configure an initiator, account, and access group for the new server mongo 7 and the new cloned volume of the same name. The MongoDB cluster is ready to add server Mongo-7.

NetApp SolidFire Management

Cluster: prime-cluster Details MVIP: 10.193.120.90 SVIP: 10.193.120.182

Reporting Management Data Protection Cluster VVols

Datstores Volumes Accounts Access Groups Initiators

☒ Active ☐ Deleted

Create Bulk Actions Filter

| Volume ID | Volume Name | Account | Access Groups | Access | Size (GB) | Snapshots | 512e | Actions |
|-----------|-------------|----------|---------------|------------|-----------|-----------|------|---------|
| 8 | mongo-7 | mongo-db | mongo-7 | Read/Write | 1000.00 | 0 | Yes | |
| 7 | mongo-meta | mongo-db | mongo-cfg3 | Read/Write | 250.00 | 0 | Yes | |
| 6 | mongo-5 | mongo-db | mongo-5 | Read/Write | 1000.00 | 0 | Yes | |
| 5 | mongo-6 | mongo-db | mongo-6 | Read/Write | 1000.00 | 0 | Yes | |
| 4 | mongo-4 | mongo-db | mongo-4 | Read/Write | 1000.00 | 3 | Yes | |
| 3 | ROOT-VOL-02 | hci | HCI-ESXi | Read/Write | 1073.74 | 0 | Yes | |
| 2 | ROOT-VOL-01 | hci | HCI-ESXi | Read/Write | 1073.74 | 0 | Yes | |
| 1 | admin-vms | hci | HCI-ESXi | Read/Write | 500.00 | 0 | Yes | |

- Add the new volume to the new host mongo- and the mount entry to the `/etc/fstab` file to survive reboots. Install and configure MongoDB and start the `mongod` process for the new cluster node mongo-7 after mongoDB is up and running access the mongo shell of the cluster primary server and add mongo-7 to the replicaSet.

```
MongoDB Enterprise rs2:PRIMARY> rs.add("mongo-7")
{ "ok" : 1 }
MongoDB Enterprise rs2:PRIMARY> █
```

6. The new server is now part of the replicaSet as a secondary member from the `rs.status()` command.

```
},
{
  "_id" : 3,
  "host" : "mongo-7:27017",
  "arbiterOnly" : false,
  "buildIndexes" : true,
  "hidden" : false,
  "priority" : 1,
  "tags" : {
  },
  "slaveDelay" : NumberLong(0),
  "votes" : 1
}
```

Note: Adding mongo-7 to a sharded cluster requires additional steps in a mongos shell for the query server to complete sharding the mongo-7 server.

8 Performance Testing

For the testing performed in this paper, each replica set in the MongoDB architecture is configured with separate volumes on the HCI storage nodes, while the global data reduction feature provides efficient space usage regardless of the number of data replicas for each MongoDB cluster.

Database performance is defined by the speed at which a database computes basic CRUD operations. The MongoDB replicaSet and sharded clusters were validated by running Yahoo Cloud Server Benchmarking (YCSB) workloads. YCSB provides a common set of workloads for evaluating the performance of different "key-value" and "cloud" serving stores. YCSB includes a set of core workloads that define a basic benchmark for cloud systems. For more information, see [YCSB core workloads](#).

YCSB has been used extensively by other storage vendors to evaluate Mongo performance. A very common use case is for applications deploying MongoDB in environments where heavy read operations are the most prevalent. YCSB was configured during all tests with the focus on the two following workloads to documents within the loaded database:

- 100% read (workload C)
- 95% reads, 5% updates (workload B)

The test environment for NetApp HCI and MongoDB was configured for two clusters. One was a replicaSet only MongoDB cluster, and the second was a sharded MongoDB cluster. Different size YCSB databases were tested with 20 million, 50 million, and 100 million records loaded in the clusters.

Figure 4) MongoDB and NetApp HCI test environment for replicaSet cluster.

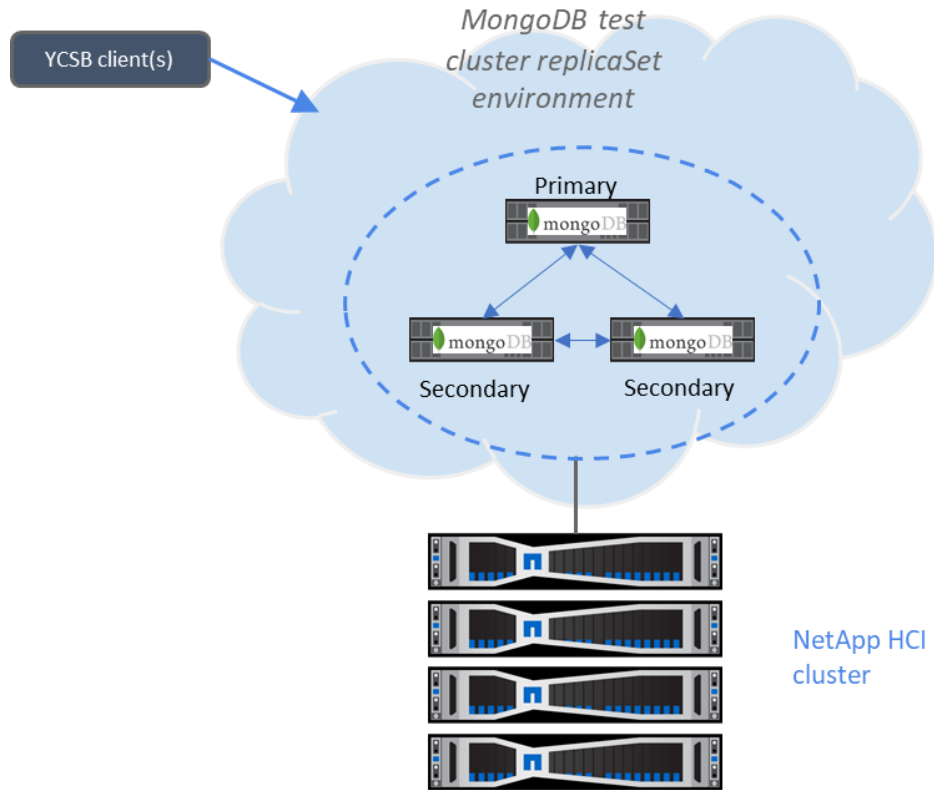
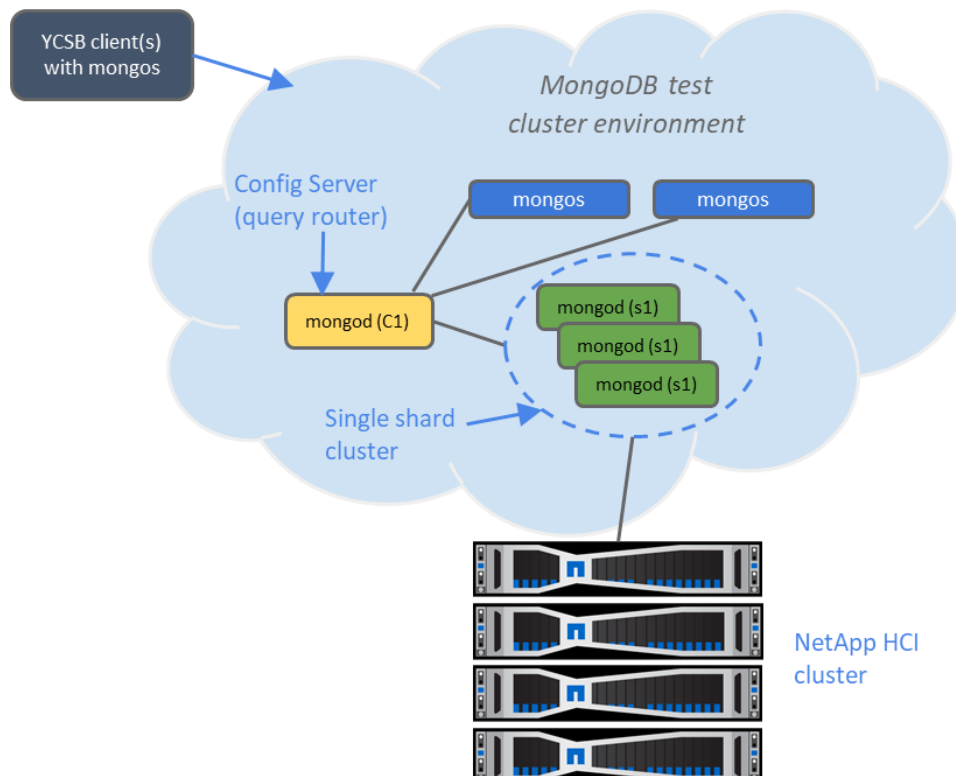


Figure 5) MongoDB and NetApp HCI test environment for sharded cluster.



In Table 3 and Table 4, the VMs and configuration information is listed for the tests.

Table 3) MongoDB nodes for clusters.

| MongoDB Cluster | Qty | MongoDB Version | Type | Cores/vCPUs | RAM | OS |
|---------------------|-----|-----------------|--------|-------------|------|--------------|
| Cluster 1 | 3 | 3.4 | VMware | 8 | 32GB | Ubuntu 16.04 |
| Cluster 2 (sharded) | 3 | 3.4 | VMware | 8 | 32GB | Ubuntu 16.04 |

Table 4) YCSB client server.

| YCSB Client | Qty | YCSB version | Type | Cores/vCPUs | RAM | OS |
|-------------|-----|--------------|--------|-------------|------|--------------|
| YCSB | 1 | 0.12.0 | VMware | 8 | 32GB | Ubuntu 16.04 |

The YCSB workloads were executed from a single client server, and the number of threads of YCSB was increased to an optimal number to get the maximum IOPS and least latency for this setup. Figure 6 through Figure 8 show the results of the YCSB tests.

Figure 6) YCSB workloads for three-node replicaSet cluster.

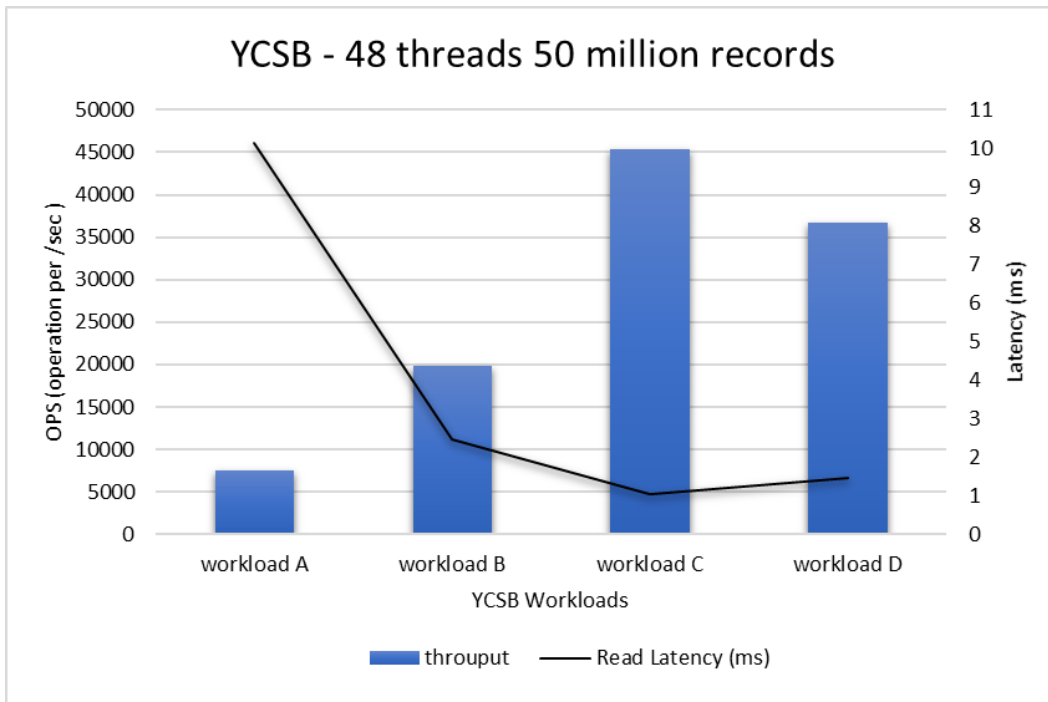


Figure 7) YCSB workload C for three-node replicaSet cluster.

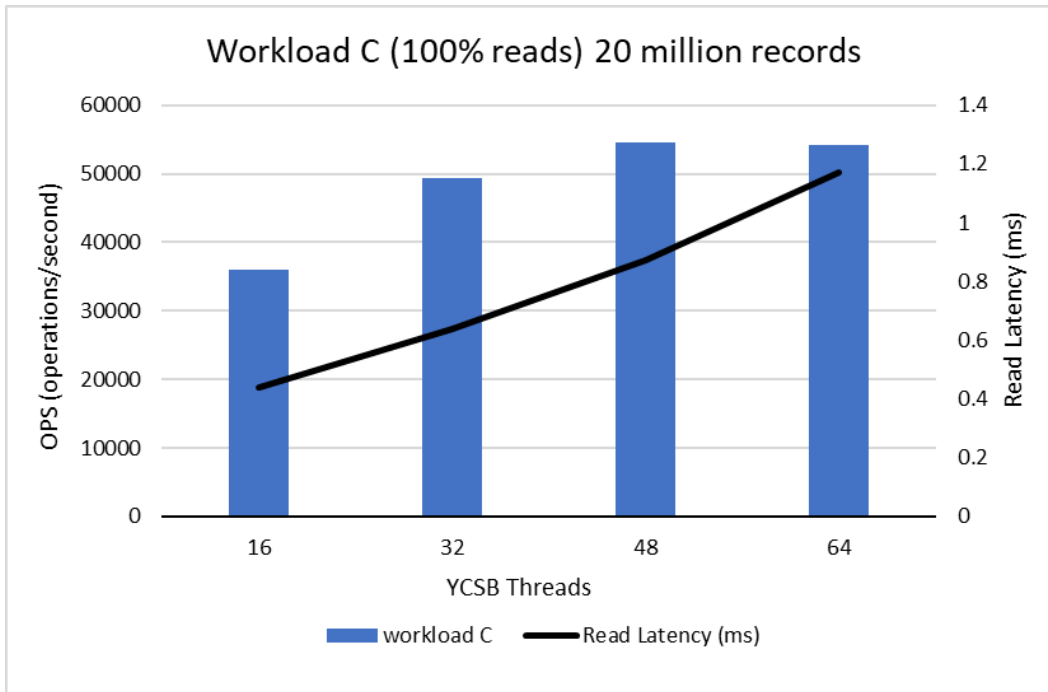
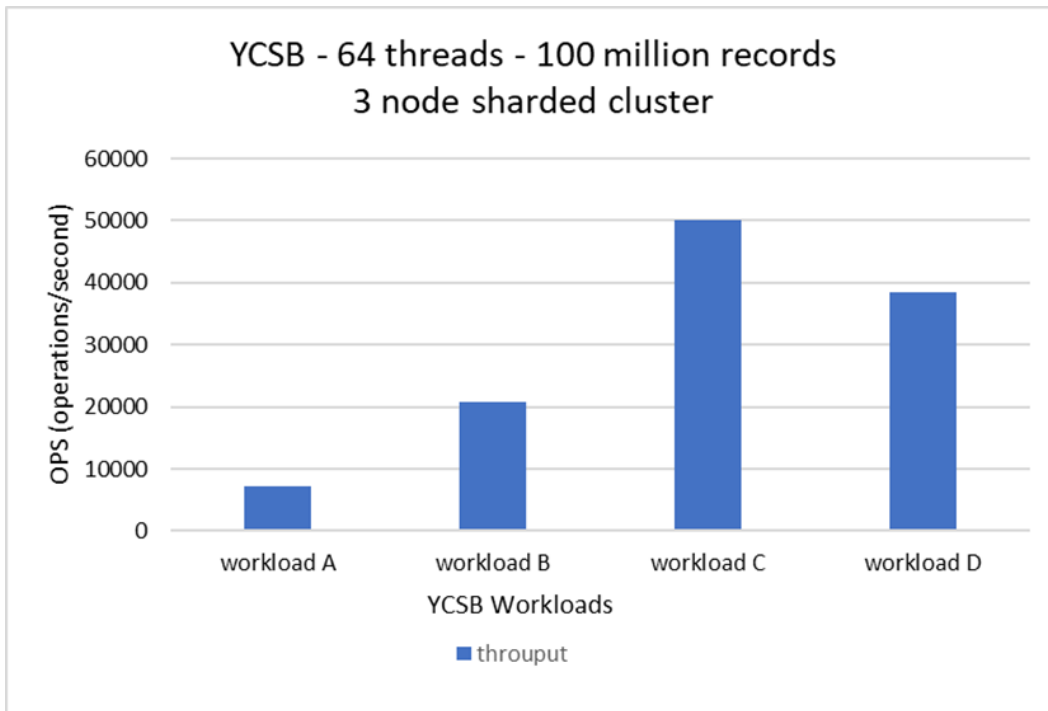


Figure 8) YCSB workloads for three-node sharded cluster.



Test Results

As shown in the figures, a MongoDB replicaSet or sharded cluster running on NetApp HCI provides high throughput of database operations with low latency. The performance combined with the volume backup and cloning capabilities of NetApp HCI provides administrators a compelling method to recover and scale

existing clusters or deploy new database instances for multiple use cases. Additional QoS granular control on volumes can enable performance guarantees for database or mixed workloads. This allows high utilization of NetApp HCI resources for compute and storage nodes, preventing unnecessary scaling.

SolidFire has features for thin provisioning, global in-deduplication, compression, scalability, database consolidation, and QoS control, allowing flexibility for solutions, including Datas IO RecoverX. One of the most beneficial capabilities of NetApp SolidFire storage for a MongoDB deployment is the ability to control IOPS per cluster node. This allows adjustment of the IOPS to a greater throughput if necessary, avoiding bottlenecks of performance and cluster problems for replica set and sharded clusters. SolidFire can also provision additional storage capacity without disruptions. Coupled with point-in-time Snapshot and cloning capabilities, the combined features make a compelling solution.

Based on the VMware virtual machines and settings used in this test, sizing cluster nodes the NetApp HCI configurations storage capacity and compute the size of 8 virtual CPUs and 32GB of RAM were optimal for MongoDB cluster nodes. The sizing is based on each node as a primary or secondary node role in the cluster.

Table 5) HCI MongoDB node sizing information.

| HCI Configuration Size | Qty. of VMs | Type | Cores/vCPUs | RAM | Storage Capacity per Volume (Max. Size Volume) |
|------------------------|-------------|--------|-------------|------|--|
| Small | 12 | VMware | 8 | 32GB | 900GB |
| Medium | 16 | VMware | 8 | 32GB | 1.3TB |
| Large | 24 | VMware | 8 | 32GB | 1.8TB |

9 Conclusion

As data centers migrate away from dedicated/siloed platforms and consolidate environments, the need to scale granular components and control the performance of applications is critical. NetApp HCI provides a unique solution with quality of service (QoS), allowing the granular control of every application, eliminating noisy neighbors, enabling administrators to set and meet all performance SLAs. As environments change, administrators can easily add compute and/or storage without overprovisioning any element of the solution.

The underlying storage nodes of NetApp HCI using all-flash media eliminate the variable performance of an HDD and, coupled with built-in efficiencies, thin provisioning, and inline data reduction features, makes the NetApp HCI a cost-effective option. The solution yields significant efficiency and agility when deploying applications and helps the business in consolidating workloads with confidence. This benefits system planners and administrators when deploying and maintaining database applications such as MongoDB. For additional information, contact SolidFire directly at info@solidfire.com.

Appendix A: Additional Setup Information

This appendix describes additional setup information used to deploy a three-node sharded cluster on Ubuntu Linux version 16.04 running on VMware virtual machines. This is the configuration used in the testing and results of this document.

1. Configure the config server, and Mongos/query router servers. For testing purposes, we used one query router. In production environments, a replicaSet of three query routers is deployed for high availability.

```
Query router (config server):
```



```

mongod --configsvr --replSet cfgSF3 --port 30001 --dbpath /mongo-meta/

then

from the mongo shell to the configsvr on port 30001

rs.initiate({_id:"cfgSF3", configsvr:true, members:[{_id:0, host:"mongo-cfg3:30001"}] } )

rs.status()

```

2. Configure the sharded cluster from a YCSB server using a mongos. Starting a mongos shell on the YCSB server is a routing service for MongoDB shard configurations that processes queries from the application layer and determines the location of this data in the sharded cluster, in order to complete these operations.

```

mongos --port 47017 --configdb cfgSF3/mongo-cfg3:30001

```

3. Connect to the mongos port 47017 in a mongos shell and configure the shards.

```

mongo --port 47017
MongoDB Enterprise mongos> sh.addShard("rs2/mongo-4:37000")
MongoDB Enterprise mongos> sh.addShard("rs2/mongo-5:38000")
MongoDB Enterprise mongos> sh.addShard("rs2/mongo-6:39000")
MongoDB Enterprise mongos> sh.enableSharding("ycsb")
MongoDB Enterprise mongos> use ycsb
MongoDB Enterprise mongos> db.createCollection("usertable")
MongoDB Enterprise mongos> db.usertable.createIndex({"_id": 1 })
MongoDB Enterprise mongos> sh.shardCollection("ycsb.usertable", {"_id": 1})

```

4. Start the mongod processes using the `/etc/mongod.conf` file or like the following for the replicaset MongoDB cluster.

The ReplicaSet servers (mongo-4, mmongo-5, mongo-6) for a sharded cluster where the Query routers has a replicaSet named "rs2".

```

mongo-4: mongod --port 37000 --shardsvr --replSet rs2 --dbpath /data/db

mongo-5: mongod --port 38000 --shardsvr --replSet rs2 --dbpath /data/db

mongo-6: mongod --port 39000 --shardsvr --replSet rs2 --dbpath /data/db

then

mongo-4: mongo --port 37000

execute in the mongo shell at port 37000 on server mongo-4:

config = {_id:"rs2", members: [ { _id:0, host:"mongo-4:37000"}, { _id:1, host:"mongo-5:38000"},
{ _id:2, host:"mongo-6:39000"} ] }

rs.initiate( config)

or from the primary server mongo-4 for a repoliceSet of rs0:

rs.initiate()
rs.add("mongo-5:38000")
rs.add("mongo-6:39000")

```

Note: The mongo-4 server is promoted to the primary Mongo server when the cluster members are connected and establish the replicaSet.

Appendix B: mongod.conf for Sharded replicaSet Members

Sample mongod.conf for Sharded replicaSet Members

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

# where to write logging data.
systemLog:
  component:
    network:
      verbosity: 2
    storage:
      verbosity: 2
    replication:
      verbosity: 2
    sharding:
      verbosity: 2

  path: /var/log/mongodb/mongod.log
  logRotate: rename
  destination: file
  timeStampFormat: ctime

# Where and how to store data.
storage:
  dbPath: /data/db
  journal:
    enabled: false
  directoryPerDB: true
  engine: wiredTiger
  wiredTiger:
    engineConfig:
      journalCompressor: snappy
    collectionConfig:
      blockCompressor: snappy
    indexConfig:
      prefixCompression: true

# how the process runs
processManagement:
  fork: true # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid # location of pidfile

# network interfaces
net:
  bindIp: mongo-rs0p
  port: 27017
  wireObjectCheck: true
  http:
    enabled: false
    JSONPEnabled: false
    RESTInterfaceEnabled: false

#security:
#operationProfiling:

replication:
  replSetName: rs0

sharding:
  clusterRole: shardsvr

## Enterprise-Only Options
#auditLog:
#snmp:
```

Appendix C: mongod.conf for configReplSet

The following is the mongod.conf file for the config server (query router). In this nonproduction test environment, a single config server was used. In production systems, three config servers are used for HA.

Sample mongod.conf for the configReplSet

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# Where and how to store data.
storage:
  dbPath: /mongo-metadata
  journal:
    enabled: true
# how the process runs
processManagement:
  fork: true # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid # location of pidfile

# network interfaces
net:
  port: 27019
  bindIp: mongo-cl

replication:
  replSetName: configReplSet

sharding:
  clusterRole: "configsvr"
```

Appendix D: Mongos Config File

The following is a mongod.conf file for the mongos used for connection to the query router. In this test environment, we configure the mongos on the YCSB server to provide the I/O load for the workloads.

Mongos Config File

```
mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# how the process runs
processManagement:
  fork: true # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid # location of pidfile

# network interfaces
net:
  port: 47017
```

```
bindIp: mongo-cfg3
sharding:
  configDB: configReplSet/mongo-cfg3:30001
```

Where to Find Additional Information

To learn more about the information described in this document, refer to the following documents and/or websites:

- Configuring SolidFire on Windows for NetApp Element software:
<http://www.netapp.com/us/media/configuring-solidfire-for-windows.pdf>
- Install MongoDB Enterprise on Linux
<https://docs.mongodb.com/manual/administration/install-enterprise-linux/>
- MongoDB Best Practices on NetApp SolidFire
<http://www.netapp.com/us/media/tr-4600.pdf>
- MongoDB Disable Transparent Huge Pages (THP)
<https://docs.mongodb.com/manual/tutorial/transparent-huge-pages/index.html>
- YCSB Core Workloads
<https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads>
- SolidFire All-Flash Array: Achieve the Next-Generation Data Center
<http://www.netapp.com/us/products/storage-systems/all-flash-array/solidfire-web-scale.aspx>
- NetApp Product Documentation
<http://docs.netapp.com>
- NetApp HCI Documentation Resources
<https://www.netapp.com/us/documentation/hci.aspx>

Version History

| Version | Date | Document Version History |
|-------------|---------------|---|
| Version 1.0 | October 2017 | Initial document creation |
| Version 1.1 | November 2017 | Changed "References" title to "Where to Find Additional Information" and other minor edits. |
| Version 1.2 | January 2019 | Updated configuration options and other minor edits |

Refer to the [Interoperability Matrix Tool \(IMT\)](#) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

Copyright Information

Copyright © 2019 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

Data contained herein pertains to a commercial item (as defined in FAR 2.101) and is proprietary to NetApp, Inc. The U.S. Government has a non-exclusive, non-transferrable, non-sublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b).

Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.