A Principled Technologies report: Hands-on testing. Real-world results.

Enable greater data reduction, storage performance, and manageability with Dell EMC PowerStore storage arrays

Dell EMC PowerStore arrays outperformed a competitor array in areas including data reduction, performance, and out-of-the-box VM deployment

Organizations face different challenges for storing and accessing data. Some need powerful arrays to maximize performance, and some want the flexibility of an infrastructure that combines both storage and compute into a single system. New all-flash, NVMe™-based Dell EMC™ PowerStore™ storage arrays help organizations meet these needs. PowerStore arrays deliver high data reduction and speed in a 2U form factor, and a hypervisor-enabled PowerStore array can internally host VMware ESXi™ VMs while also providing storage resources to external hosts.

We tested two Dell EMC PowerStore arrays against the equivalent NVMe-based array of a competitor ("Vendor A") across a range of performance and usability metrics. The Dell EMC PowerStore arrays had higher data reduction ratios, supported more input/output operations per second (IOPS), responded in less time, and provided higher bandwidth than the array from Vendor A. In addition, the hypervisor-enabled PowerStore array allowed our admins to start deploying a VM right out of the box. With Dell EMC PowerStore storage arrays, organizations can maximize storage capacity, increase storage performance, and cut management overhead time.

Up to 3x the data reduction*
Maximize storage efficiency

Up to 125% more bandwidth**
Minimize bottlenecks during periods of high user traffic

Up to 53% more IOPS**
Satisfy more users with faster storage performance

Up to 10.6x faster out-of-the-box VM deployment†
Save IT time and effort

Note: The standard PowerStore model is the 9000T, and the hypervisor-enabled model is the 9000X.

*Dell EMC PowerStore 9000T and 9000X vs. Vendor A array
**Dell EMC PowerStore 9000T vs. Vendor A array
†Dell EMC PowerStore 9000X vs. Vendor A array

Enable greater data reduction, storage performance, and manageability with Dell EMC PowerStore storage arrays April 2020

The science behind the report:

Enable greater data reduction, storage performance, and manageability with Dell EMC PowerStore storage arrays

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Enable greater data reduction, storage performance, and manageability with Dell EMC PowerStore storage arrays.

We concluded our hands-on testing on April 2, 2020. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on March 26, 2020 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

Table 1: Results of our testing on the Dell EMC™ PowerStore™ 9000T.

	Dell EMC PowerStore 9000T	Vendor A array	Win
VDbench testing			
Data reduction ratio			
256KB 100% write	9.6:1 (reduced 32 TB of data to 3.3 TB)	3.1:1 (reduced 32 TB of data to 10.1 TB)	3.09x the data reduction
Input/output operations per second (IOPS)			
32KB random read (16T*)	566,145	370,983	52% more IOPS
32KB, 80/20 read/write mix (8T)	294,793	218,890	34% more IOPS
Simulated DSS workload (8T)	130,374	86,090	51% more IOPS
4KB random write (4T)	218,054	142,042	53% more IOPS
Bandwidth (MB/s)			
256KB random read (8T)	25,612	11,358	125% more bandwidth
256KB sequential read (8T)	25,450	16,356	55% more bandwidth
Latency (ms)			
32KB random reads (8T paced at 360,000 IOPS)	0.70	1.48	52% lower latency
Storage provisioning on clustered systems			
Time to provision 8 LUNs (min:sec)	1:25	8:57	5.3x faster
Steps to provision 8 LUNs	13	100	87 fewer steps

*"T" stands for "threads per LUN."

Table 2: Results of our testing on the Dell EMC PowerStore 9000X.

	Dell EMC PowerStore 9000X	Vendor A array	Win
Data reduction ratio			
256KB 100% write	9.4:1 (reduced 32 TB of data to 3.6 TB)	3.1:1 (reduced 32 TB of data to 10.1 TB)	3.03x the data reduction
Vdbench testing at 256KB random reads (4T)**			
IOPS	65,358	47,035	38% more IOPS
Bandwidth (MB/s)	16,339	11,758	38% more bandwidth
Latency (ms)	3.91	5.43	27% lower latency
MongoDB testing (40T)			
Operations per second	208,178	N/A	N/A
Read application latency (ms)	0.76	N/A	N/A
Update (write) application latency (ms)	0.86	N/A	N/A
Out-of-the-box VM deployment			
Time (min:sec)	00:53	10:21	10.6x faster
Steps	12	59	47 fewer steps

**The PowerStore 9000X achieved these results while running Vdbench and MongoDB simultaneously. The Vendor A array was only running Vdbench.

System configuration information

Table 3: Detailed information on the server we tested.

Server configuration information	12 x Dell EMC PowerEdge™ R740
BIOS name and version	Dell EMC PowerEdge R740 2.3.10
Non-default BIOS settings	Virtualization enabled
Operating system name and version/build number	VMware ESXi™ 6.7.0 Update 3 Build 14320388
Date of last OS updates/patches applied	8/20/2019
Power management policy	Performance
Processor	
Number of processors	2
Vendor and model	Intel® Xeon® Gold 6230
Core count (per processor)	20
Core frequency (GHz)	2.10
Memory module(s)	
Total memory in system (GB)	256
Number of memory modules	16
Vendor and model	Hynix HMA82GR7CJR8N-WM
Size (GB)	16 GB
Type	PC4-2666
Speed (MHz)	2,666
Speed running in the server (MHz)	2,666
Storage controller	
Vendor and model	Dell PERC H330
Cache size (GB)	N/A
Firmware version	25.5.6.0009
Driver version	7.708.07.00
Local storage	
Number of drives	2
Drive vendor and model	Intel SSDSC2KB480G8R
Drive size (GB)	480 GB
Drive information (speed, interface, type)	6Gbps, SATA, SSD
Network adapter	
Vendor and model	Broadcom Gigabit Ethernet BCM5720
Number and type of ports	2 x 1Gb, 2 x 10Gb
Driver version	21.40.21

Server configuration information	12 x Dell EMC PowerEdge™ R740
Storage adapter	
Vendor and model	Emulex LPe35002-M2-D
Number and type of ports	2 x 2-port 32Gb Fibre Channel
Firmware version	12.0.257.15
Power supplies	
Vendor and model	Dell 0PJMDNA01
Number of power supplies	2
Wattage of each (W)	750

Table 4: Detailed information on the storage we tested.

Storage configuration information	Dell EMC PowerStore 9000T	Dell EMC PowerStore 9000X	Vendor A array
Software version	1.0.0.1.3.622	1.0.0.0.4.032	9.6P4
Number of storage shelves	1	1	2
Number of data drives	21	21	22
Drive part number	119000746-01	118000740	X4001A
Drive size (TB)	1.92	1.92	1.92

Detailed testing procedure

We received three separate testbeds from Dell EMC: one for PowerStore 9000T testing, one for PowerStore 9000X testing, and one for Vendor A testing. In addition to the storage arrays we tested, each testbed had four Dell EMC PowerEdge R740 servers equipped with two dual-port 32GB Emulex Fibre Channel adapters and VMware ESXi 6.7. Where possible, we verified that the setups on the testbeds were identical, and we configured the arrays as closely as possible following best practices for each storage vendor.

We began by creating 64 volumes on each storage array. We then created a LUN for each volume on the Vendor A array. After finishing the volume and LUN creation, we mapped the volumes from both arrays to the four ESXi servers connected to each storage array. After mapping the volumes/LUNs to the hosts, we added four RDM disks to 16 Linux®-based virtual machines on both testbeds. We tuned the host according to each vendor's best practices, which included setting up multi-pathing on the hosts in each environment. We then prefilled the LUNs and ran through the series of Vdbench configurations listed below and collected the results. After we completed the set of tests, we unmapped the Raw Device Mapping and LUNs from the VMs and hosts. We then repeated the above volume creation process on the Dell EMC storage arrays, this time creating only 32 volumes. On the Vendor A array, we created two volumes, each 4.6 TB in capacity, and then created 32 LUNs, distributing them between the two volumes. After creating the 32 volumes and LUNs, we mapped them to the hosts and mapped the RDMs to the 16 test VMs, just as in previous tests. Again, we prefilled the 32 LUNs and then ran through the series of tests listed below, collecting the data once the solutions completed the tests. When we finished the 64 and 32 LUN tests, we unmapped the RDMs and LUNs from the VMs and hosts and removed all volumes and LUNs from each storage array. We then let the storage arrays sit idle to allow them to reclaim space, and restarted the process of creating LUNs, running tests, and collecting data.

We repeated this process two times for a total of three test runs. The results we present are the median results out of these three test runs.

Performance testing

Provisioning and testing 64 LUNs

PowerStore 9000T and Vendor A array

Prefilling LUN data

To fill the LUNs with data, we used 256KB sequential writes with a single thread.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root,jvms=1
hd=hd1,system=QT_001
sd=default,openflags=o_direct
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
sd=sd3,hd=hd1,lun=/dev/sdd
sd=sd4,hd=hd1,lun=/dev/sde
```

We continued this command sequence up to 64 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_prefill,sd=sd*,xfersize=256k,seekpct=eof,rdpct=0
rd=default
rd=rd_prefill,wd=wd_prefill,elapsed=10h,interval=10,iorate=max,forthreads=(1)
```

Testing random reads

For these tests, we deployed 64 500GB LUNs and configured Vdbench to run at 32K and 256K block sizes at 16 and 8 thread counts, respectively.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root
hd=hd1,system=QT_001
sd=default,openflags=o_direct
```

```
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
sd=sd3,hd=hd1,lun=/dev/sdd
sd=sd4,hd=hd1,lun=/dev/sde
```

We continued this command sequence up to 64 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_32k,sd=sd*,xfersize=32k,seekpct=100
wd=wd_256k,sd=sd*,xfersize=256k,seekpct=100
rd=default
rd=read32k_test,wd=wd_32k,iorate=max,warmup=30,interval=10,forrdpct=(100),elapsed=180,forthreads=16
rd=read256k_test,wd=wd_256k,iorate=max,warmup=30,interval=10,forrdpct=(100),elapsed=180,forthreads=8
```

Testing sequential reads

For these tests, we deployed 64 500GB LUNs and configured Vdbench to run 256K sequential reads with 8 threads.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root
hd=hd1,system=QT_001
sd=default,openflags=o_direct
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
sd=sd3,hd=hd1,lun=/dev/sdd
sd=sd4,hd=hd1,lun=/dev/sde
```

We continued this command sequence up to 64 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_256k,sd=sd*,xfersize=256k,seekpct=seqnz
rd=default
rd=read256k_test,wd=wd_256k,iorate=max,warmup=30,interval=10,forrdpct=(100),elapsed=180,forthreads=8
```

Testing paced random reads

For these tests, we deployed 64 500GB LUNs and configured Vdbench to generate a 360K IOPs workload using 32KB random reads with 8 threads.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root
hd=hd1,system=QT_001
sd=default,openflags=o_direct
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
sd=sd3,hd=hd1,lun=/dev/sdd
sd=sd4,hd=hd1,lun=/dev/sde
```

We continued this command sequence up to 64 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_32k,sd=sd*,xfersize=32k,seekpct=100
rd=default
rd=read32k_
test1,wd=wd_32k,iorate=(360000),warmup=30,interval=10,forrdpct=(100),elapsed=180,forthreads=8
```

Provisioning and testing 32 LUNs

PowerStore 9000T and Vendor A array

Prefilling LUN data

To fill the LUNs with data, we used 256KB sequential writes with a single thread.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root,jvms=1
hd=hd1,system=QT_001
sd=default,openflags=o_direct
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
```

We continued this command sequence up to 32 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_prefill,sd=sd*,xfersize=256k,seekpct=eof,rdpct=0
rd=default
rd=rd_prefill,wd=wd_prefill,elapsed=10h,interval=10,iorate=max,forthreads=(1)
```

Testing 128K Decision Support System (DSS)-like workload

For these tests, we deployed 32 400GB LUNs and configured Vdbench to run a workload that emulates a typical DSS workload.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root,jvms=1
hd=hd1,system=QT_001
sd=default,openflags=o_direct
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
```

We continued this command sequence up to 32 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_DSS128K_RRH,sd=*,rhpct=100,rdpct=100,xfersize=64K,skew=18,range=(0,6m)
wd=wd_DSS128K_RM,sd=*,rdpct=100,xfersize=64k,skew=18,range=(0,100)
wd=wd_DSS128K_RW,sd=*,rdpct=0,xfersize=64K,skew=4,range=(0,100)
wd=wd_DSS128K_SR,sd=*,rdpct=100,seekpct=seqnz,range=(0,100),xfersize=128K,skew=48
wd=wd_DSS128K_SW,sd=*,rdpct=0,seekpct=seqnz,range=(0,100),xfersize=128K,skew=12
rd=default
rd=rd_DSS128K,wd=wd_DSS128K_*,iorate=max,elapsed=180,interval=10,warmup=30,forthreads=8
```

Testing 32K 80/20 read/write mix

For these tests, we deployed 32 400GB LUNs and configured Vdbench to run a 32KB mixed 80/20 read/write workload with 8 threads.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root
hd=hd1,system=CB_001
sd=default,openflags=o_direct
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
```

We continued this command sequence up to 32 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_32k,sd=sd*,xfersize=32k,seekpct=100
rd=default
rd=read32k_test,wd=wd_32k,iorate=max,warmup=30,interval=10,forrdpct=(80),elapsed=180,forthreads=8
```

Testing 4K random writes

For these tests, we deployed 32 400GB LUNs and configured Vdbench to run 4KB random writes with 4 threads.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root
hd=hd1,system=CB_001
sd=default,openflags=o_direct
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
```

We continued this command sequence up to 32 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_4k,sd=sd*,xfersize=4k,seekpct=100
rd=default
rd=read4k_test,wd=wd_4k,iorate=max,warmup=30,interval=10,forrdpct=(0),elapsed=180,forthreads=4
```

Providing storage resources to external hosts while hosting database VMs internally

PowerStore 9000X and Vendor A array

We used external servers from Vendor A to run the external Vdbench workload. The PowerStore 9000X array came equipped with its own VMware ESXi virtualized enviroment and presented its two storage processors as two ESXi 6.7 hosts. We grouped both ESXi hosts inside a vCenter cluster, and used this cluster to host our internal VMs running MongoDB. We set up a MongoDB environment with three MongoDB servers and one YCSB server for executing the benchmark tests.

Using our vSphere 6.7 environment, we created three VMs for MongoDB and installed CentOS 7 on each. We then went through the configuration steps below to prepare the VMs for MongoDB. We installed the community version of MongoDB on each of the MongoDB VMs, and created a replica set with the three MongoDB VMs. The replica set consists of one primary VM and two secondary VMs. After configuring our MongoDB VMs, we created another VM for YCSB, installed CentOS 7, and followed the configuration steps below to set up and install YCSB.

We began by creating 64 volumes on each storage array. We then created a LUN for each volume on the Vendor A array. After finishing the volume and LUN creation, we mapped the volumes from both arrays to the four ESXi servers connected to each storage array. After mapping the volumes and LUNs to the hosts, we added four RDM disks to 16 Linux®-based virtual machines on both testbeds. We tuned the host according to each vendor's best practices, which included setting up multi-pathing on the hosts in each environment. We prefilled the LUNs on each array using the Vdbench prefill script below. After we completed the installation and configuration steps, we moved to loading the database and running the tests. We loaded the database with 100,000,000 records (roughly 135 GB per VM, for 405 GB total) using our YCSB VM and four separate processes. After loading the database, we ran a test with the following procedure on the PowerStore 9000X:

1. Start the Vdbench test for 40 minutes.
2. Wait 3 minutes, and start the YCSB test.
3. Wait for both to finish (YCSB will finish first).
4. Collect the data from the Vdbench and YCSB VMs.
5. Take the average Vdbench score during the time interval that the YCSB tests were running.
6. Report the scores.

We repeated this procedure twice more for a total of three runs. We present the median run. Because the Vendor A array cannot host internal VMs, we only ran the Vdbench workload on the external hosts for Vendor A.

Prefilling LUN data

To fill the LUNs with data, we used 256KB sequential writes with a single thread.

```
compratio=3
dedupratio=3
dedupunit=4k
messagescan=no
hd=default,shell=ssh,user=root,jvms=1
hd=hd1,system=QT_001
sd=default,openflags=o_direct
sd=sd1,hd=hd1,lun=/dev/sdb
sd=sd2,hd=hd1,lun=/dev/sdc
```

We continued this command sequence up to 64 LUNs, then entered the following:

```
wd=default,sd=*
wd=wd_prefill,sd=sd*,xfersize=256k,seekpct=eof,rdpct=0
rd=default
rd=rd_prefill,wd=wd_prefill,elapsed=10h,interval=10,iorate=max,forthreads=(1)
```

Configuring the VMs for testing

We used the following configuration for the test VMs.

VM type	Number of VMs	vCPUs	vRAM	Virtual disks
MongoDB	3	16	64 GB	1 x 50 GB (OS) 1 x 200 GB (DB)
YCSB	1	32	64 GB	1 x 50 GB (OS)

Installing and configuring MongoDB on the MongoDB VMs

We installed CentOS 7 onto each VM, then performed the following steps on each server:

1. Log into the VM you are configuring.
2. Set a static IP address by editing the interface file at /etc/sysconfig/network-scripts/ifcfg-<INTERFACE>. For example:

```
NAME=ens192
IPADDR=192.168.200.31
PREFIX=24
DEVICE=ens192
ONBOOT=yes
USERCTL=no
BOOTPROTO=none
PEERDNS=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
```

3. Stop and disable the firewall service by typing the following commands:

```
systemctl stop firewalld
systemctl disable firewalld
```

4. Enter the SELinux configuration file by typing the following command:

```
vim /etc/sysconfig/selinux
```

5. Once inside the SELinux configuration file, change its configuration from enabled to disabled.
6. Change the hostname with the following command:

```
hostnamectl set-hostname <hostname>
```

7. Perform system updates:

```
sudo yum update -y
```

8. Edit the file /etc/default/grub and add the 'transparent_hugepage=never' parameter. For example:

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=centos_vsh-centos7/root rd.lvm.lv=centos_vsh-centos7/
swap rhgb quiet transparent_hugepage=never"
GRUB_DISABLE_RECOVERY="true"
```

9. Run grub2-mkconfig to regenerate a grub.cfg file with the following command:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

10. Reboot the VM.
11. Add the Mongo community database to your yum repo, and install MongoDB to your VM by typing the following commands:

```
vim /etc/yum.repos.d/mongodb-org-4.2.repo
yum install -y mongodb-org
```

12. Create a directory for the MongoDB database and log files:

```
mkdir /data/
```

13. Add the database drive to the VM, and create the database folders for MongoDB by typing the following commands:

```
mkfs.xfs -K /dev/sdb
mount /dev/sdb /data
mkdir /data/mongodb
```

14. Edit the file /etc/fstab and add the following line to ensure the database drive mounts at boot:

```
/dev/sdb /data xfs defaults 0 0
```

15. Change the ownership of the MongoDB database folder:

```
chown -R mongod:mongod /data/mongodb
```

16. Open the MongoDB configuration file by typing the following command:

```
vim /etc/mongod.conf
```

17. Inside the MongoDB configuration file, make the following changes, filling in the # with the appropriate number:

```
storage:
  dbPath: /data/mongodb
  journal:
    enabled: true
directoryPerDB: True
wiredTiger:
    engineConfig:
journalCompressor: snappy
directoryforIndexes: true
collectionConfig:
blockCompressor: snappy
indexConfig:
prefixCompression: true
systemLog:
destination: file
logAppend: true
path: /data/mongodb/mongod.log
net:
  port: 27017
  bindIp: "[your server's IP],localhost"
replication:
    replSetName: rs0
```

18. Save the MongoDB configuration file.
19. Start your MongoDB server by typing the following command:

```
mongod -f /etc/mongod.conf
```

20. Repeat steps 1 through 19 on your two remaining MongoDB servers.

Creating the MongoDB replica sets

1. To enter the MongoDB console, type `mongo`
2. In the MongoDB console, create the replica set by typing the following command, filling in the # in the command with the appropriate number:

```
rs.initiate(
  {
    _id : "rs0",
    members: [
      { _id : 0, host : "mongod#.test.local:27017" },
      { _id : 1, host : "mongod#.test.local:27017" },
      { _id : 2, host : "mongod#.test.local:27017" }
    ]
  }
)
```

3. To exit the MongoDB terminal, type `exit`
4. To enter the MongoDB console again, type `mongo`
5. To verify which of the three VMs is the primary, type `rs.isMaster()`

Installing the YCSB driver VMs

1. Log into the VM you are configuring.
2. Set a static IP address by editing the interface file at /etc/sysconfig/network-scripts/ifcfg-<INTERFACE>. For example:

```
NAME=ens192
IPADDR=192.168.200.31
PREFIX=24
DEVICE=ens192
ONBOOT=yes
USERCTL=no
BOOTPROTO=none
PEERDNS=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
```

3. Stop and disable the firewall service by typing the following commands:

```
systemctl stop firewalld
systemctl disable firewalld
```

4. Enter the SELinux configuration file by typing the following command:

```
vim /etc/sysconfig/selinux
```

5. Once inside the SELinux configuration file, change its configuration from enabled to disabled.
6. Change the hostname with the following command:

```
hostnamectl set-hostname <hostname>
```

7. Perform system updates:

```
sudo yum update -y
```

8. Add the Mongo enterprise database to your yum repo, and install MongoDB to your VM by typing the following commands:

```
vim /etc/yum.repos.d/mongodb-org-4.2.repo
yum install -y mongodb-org
```

9. Install Java with the following command:

```
sudo yum install java-devel
```

10. Install Maven with the following commands:

```
wget “http://apache.cs.utah.edu/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz”
sudo tar xzf apache-maven-*-bin.tar.gz -C /usr/local
cd /usr/local
sudo ln -s apache-maven-* maven
```

11. Add the following lines to the end of your profile at /etc/profile.d/maven.sh:

```
export M2_HOME=/usr/local/maven
export PATH=${M2_HOME}/bin:${PATH}
```

12. Download and unpack YCSB into /usr/local on your driver VM with the following commands:

```
run(“curl -O --location https://github.com/brianfrankcooper/YCSB/releases/download/0.17.0/ycsb-
0.17.0.tar.gz”)
run(“tar xfvz ycsb-0.17.0.tar.gz”)
```

Creating the YCSB database

1. Log into a YCSB VM.
2. Open four terminal windows and navigate to /usr/local in each.
3. Run each of the following commands in a separate terminal to create a roughly 130GB database for MongoDB using four YCSB processes:

```
./bin/ycsb load mongodb-async -s -P ./workloads/workloadb -threads 32 -p mongodb
.url=”mongodb://mongod1.test.local:27017/ycsb” -p recordcount=100000000 -p insertstart=0 -p
insertcount=25000000

./bin/ycsb load mongodb-async -s -P ./workloads/workloadb -threads 32 -p mongodb
.url=”mongodb://mongod1.test.local:27017/ycsb” -p recordcount=100000000 -p insertstart=25000000 -p
insertcount=25000000
```

```
./bin/ycsb load mongodb-async -s -P ./workloads/workloadb -threads 32 -p mongodb
.url=”mongodb://mongod1.test.local:27017/ycsb” -p recordcount=100000000 -p insertstart=50000000 -p
insertcount=25000000

./bin/ycsb load mongodb-async -s -P ./workloads/workloadb -threads 32 -p mongodb
.url=”mongodb://mongod1.test.local:27017/ycsb” -p recordcount=100000000 -p insertstart=75000000 -p
insertcount=25000000
```

Running Vdbench on the array from Vendor A

1. Log into the Vdbench controller VM, and navigate to the Vdbench directory.
2. Type the following command to start the Vdbench test, replacing # with the correct run number:

   ```
   ./vdbench -f 256K_random_read -o 256K_random_read_run#
   ```

   For this step, we used the following Vdbench script:

   ```
   compratio=3
   dedupratio=3
   dedupunit=4k
   messagescan=no
   hd=default,shell=ssh,user=root
   hd=hd1,system=JZ_001
   sd=default,openflags=o_direct
   sd=sd1,hd=hd1,lun=/dev/sdb
   sd=sd2,hd=hd1,lun=/dev/sdc
   sd=sd3,hd=hd1,lun=/dev/sdd
   sd=sd4,hd=hd1,lun=/dev/sde
   ```

   We continued this command sequence up to 64 LUNs, then entered the following:

   ```
   wd=default,sd=*
   wd=wd_256k_RR,sd=sd*,xfersize=256k,seekpct=100
   rd=default
   rd=read256k_test,wd=wd_256k_
   RR,iorate=max,warmup=30,interval=10,forrdpct=(100),elapsed=2400,forthreads=(4)
   ```

Running the MongoDB YSCB test in parallel with external bandwidth tests using Vdbench

1. On each MongoDB VM, run the following command to clear all caches:

   ```
   echo 3 > /proc/sys/vm/drop_caches
   ```

2. Log into the Vdbench controller VM, and navigate to the Vdbench directory.
3. Type the following command to start the Vdbench test, replacing # with the correct run number:

   ```
   ./vdbench -f 256K_random_read -o 256K_random_read_run#
   ```

   For this step, we used the following Vdbench script:

   ```
   compratio=3
   dedupratio=3
   dedupunit=4k
   messagescan=no
   hd=default,shell=ssh,user=root
   hd=hd1,system=JZ_001
   sd=default,openflags=o_direct
   sd=sd1,hd=hd1,lun=/dev/sdb
   sd=sd2,hd=hd1,lun=/dev/sdc
   sd=sd3,hd=hd1,lun=/dev/sdd
   sd=sd4,hd=hd1,lun=/dev/sde
   ```

   We continued this command sequence up to 64 LUNs, then entered the following:

   ```
   wd=default,sd=*
   wd=wd_256k_RR,sd=sd*,xfersize=256k,seekpct=100
   rd=default
   rd=read256k_test,wd=wd_256k_
   RR,iorate=max,warmup=30,interval=10,forrdpct=(100),elapsed=2400,forthreads=(4)
   ```

4. Log into your YCSB driver VM, and open four terminal windows.
5. In each terminal window, run the following command to launch four YCSB processes, and run the test for roughly 30 minutes:

```
./bin/ycsb run mongodb-async -s -P ./workloads/workloadb -threads 40 -p mongodb
.url="mongodb://mongod1.test.local:27017/ycsb?w=1" -p operationcount=75000000 > ~/output/[filename].
txt
```

6. Record the results from the test.
7. Repeat steps 1 through 6 twice more for a total of three runs.

Usability testing

Provisioning storage on clustered systems

PowerStore 9000T cluster

Creating 8 LUNs

1. Open a browser and navigate to PowerStorage Manager UI.
2. Enter credentials and click Login.
3. Click on Storage, and select Volumes.
4. At Properties screen, enter a name or prefix, description, quantity, and size. If necessary, update the Performance policy. Click Next.
5. At Host Mapping screen, select the hosts or host groups you are going to map to the volumes, and click Next.
6. At the Summary screen, click Create.

Discovering LUNs in a virtualized ESXi environment

1. Open a browser, and navigate to the vCenter vSphere client.
2. Enter your credentials, and click Login.
3. If necessary, expand Datacenter and cluster.
4. Select the target host.
5. Click the Configure tab.
6. Under Storage→Storage adapters, click Rescan Storage to discover newly added storage.
7. Select Scan for new Storage Devices, and click OK.

Vendor A array cluster

Creating aggregates

1. Open a browser, and navigate to Vendor A cluster management UI.
2. Enter your credentials, and click Login.
3. Under Storage, click Aggregates.
4. Click the Create icon, and click Submit to accept the recommended settings.
5. Click Close.

Creating 8 LUNs

Note: This process will automatically create the necessary volumes as well.

1. Under Storage, click LUNs.
2. Under the LUN Management tab, click the Create icon.
3. Select the appropriate SVM, and click Select.
4. At the Welcome to Create LUN Wizard screen, click Next.
5. At the General Properties screen, enter a LUN name, select the OS type, and enter a LUN size. Click Next.
6. At the LUN Container screen, select Create a new flexible volume in, and click Choose.
7. Select the appropriate aggregate, click OK, and click Next.
8. At the Initiators Mapping screen, select the appropriate Initiator group name, and click Next.
9. At the Storage Quality of Service Screen, select Next.
10. At the LUN Summary screen, click Next.
11. Click Finish.

Repeat steps 1 through 11 to create 7 more volumes and LUNs, each time manually selecting a different aggregate to keep volumes and LUNs balanced across the entire cluster.

Discovering LUNs in a virtualized ESXi environment

1. Open a browser and navigate to the vCenter vSphere client.
2. Enter credentials, and click Login.
3. Expand Datacenter, and cluster if necessary.
4. Select the target host.
5. Click on the Configure tab.
6. Under Storage→Storage adapters, click Rescan Storage to discover newly added storage
7. Select Scan for new Storage Devices, and click OK.

Deploying VMs out of the box

PowerStore 9000X

Deploying VMs on a VMware vSphere environment

1. Open a browser, and navigate to the vCenter vSphere client.
2. Enter credentials, and click Login
3. Expand the PX9000 Data Center.
4. Expand the PSX9000 Cluster.
5. Right-click on an ESXi host, and select New Virtual Machine.
6. Select Create a new virtual machine, and click Next.
7. Enter a virtual machine name, and click Next.
8. At Select Storage screen, select PowerStore PSX9000 storage, and click Next.
9. At compatibility screen, select ESXi 6.7, and click Next.
10. Select appropriate OS details, and click Next.
11. At Customize hardware screen, click Next.
12. At ready to complete screen, click Finish.

Vendor A array

Creating aggregates

1. Open a browser, and navigate to Vendor A cluster management UI.
2. Enter credentials, and click Login.
3. Under Storage, click Aggregates.
4. Click the Create icon, and click Submit to accept the recommended settings.
5. Click Close.

Creating an SVM

1. Under Storage, click SVMs.
2. Click the Create icon.
3. Enter a name, select a FC/FCoE data protocol, and click Submit & Continue.
4. At the Configure FC/FCoE protocol screen, select Configure Data LIFs for FC.
5. At the SVM administration screen, click Skip, and click OK.

Configuring Fibre Channel switch zoning

1. Log into Fibre Channel switch UI.
2. Click on Configure→Zone Admin.
3. Click Zone tab→New Zone.
4. Create a new fibre channel zone with appropriate hosts HBAs and storage ports.
5. Click Zone Config tab, and add the new zones to Zone Config Members.
6. Click Save Config.
7. Click Enable Config, and select the appropriate configuration to enable.

Mapping hosts

1. On Vendor A cluster management UI, Under Storage, click LUNs.
2. Select the Initiator Groups tab.
3. Click the Create icon.
4. Enter a name and operating system, and select the FC/FCoE data protocol.
5. Click the Initiators tab, and use the drop-down menu to add appropriate initiators.
6. Click Create.

Creating one volume and one LUN

1. Under Storage, click LUNs.
2. Under the LUN Management tab, click the Create icon.
3. Select the appropriate SVM, and click Select.
4. At the Welcome to Create LUN Wizard screen, click Next.
5. At the General Properties screen, enter a LUN name, select the OS type, and enter a LUN size. Click Next.
6. At the LUN Container screen, select Create a new flexible volume in, and click Choose.
7. Select the appropriate aggregate, click OK, and click Next.
8. At the Initiators Mapping screen, select the appropriate Initiator group name, and click Next.
9. At the Storage Quality of Service Screen, select Next.
10. At the LUN Summary screen, click Next.
11. Click Finish.

Adding LUNs to a virtualized ESXi environment

1. Open a browser, and navigate to the vCenter vSphere client.
2. Enter credentials, and click Login.
3. Expand Datacenter, and cluster if necessary.
4. Select the target host.
5. Click the Configure tab.
6. Under Storage→Storage adapters, click Rescan Storage to discover newly added storage.
7. Select Scan for new Storage Devices, and click OK.

Creating a single VMFS datastore from a LUN

1. Right-click on target host, and select Storage→New Datastore.
2. At the Type screen, select VMFS, and click Next.
3. At Name and device selection screen, enter a Datastore name and select a LUN for provisioning the datastore. Click Next.
4. At VMFS version screen, select VMFS 6, and click Next.
5. At the Partition configuration screen, use default settings, and click Next.
6. At the Ready to complete screen, click Finish.

Deploying VMs in a VMware vSphere environment

1. On the vCenter UI, expand the Data Center.
2. Expand the cluster.
3. Right-click on an ESXi host, and select New Virtual Machine.
4. Select Create a new virtual machine, and click Next.
5. Enter a virtual machine name, and click Next.
6. At Select Storage screen, select the newly created VMFS datastore, and click Next.
7. At Compatibility screen, select ESXi 6.7, and click Next.
8. Select appropriate OS details, and click Next.
9. At Customize hardware screen, click Next.
10. At ready to complete screen, click Finish.

Read the report at http://facts.pt/sgqbpyp ▶

This project was commissioned by Dell EMC.

Principled Technologies®

Facts matter.®