The science behind the report:

Achieve more storage performance with Dell PowerEdge R750 servers equipped with Broadcom PCIe Gen4 switches

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 Achieve more storage performance with Dell PowerEdge R750 servers equipped with Broadcom PCIe Gen4 switches.

We concluded our hands-on testing on March 30, 2022. 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 15, 2022 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to https://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: Results of our 4KB random read workload FIO benchmark testing.

Random read testing results	Dell PowerEdge™ R750	Dell PowerEdge R740xd	Up to 2.1x the raw IOPS
8 NVMe™ drives			
8 cores (Thousands of IOPS (kIOPS))	5,011.33	2,996.66	Up to 12 million IOPS
16 cores (kIOPS)	7,864.00	3,156.00	
36 cores (kIOPS)	12,100.00	3,163.66	
56 cores (kIOPS)	12,000.00	N/A	

Random read testing results	Dell PowerEdge™ R750	Dell PowerEdge R740xd	Up to 2.1x the raw IOPS
12 NVMe drives			
8 cores (kIOPS)	4,975.66	4,203.00	Up to 11.4 million IOPS
16 cores (kIOPS)	9,183.66	4,502.66	
36 cores (kIOPS)	11,100.00	4,658.00	
56 cores (kIOPS)	11,466.67	N/A	
24 NVMe drives			
8 cores (kIOPS)	4,952.33	4,729.33	Up to 12.3 million IOPS
16 cores (kIOPS)	9,924.33	5,574.33	
36 cores (kIOPS)	12,000.00	5,744.00	
56 cores (kIOPS)	12,300.00	N/A	

Table 2: Results of our 4KB random write workload FIO benchmark testing. Note: Our random write testing was limited by NVMe drive performance and could not properly show the differences between the two architectures.

Random write testing results	Dell PowerEdge R750	Dell PowerEdge R740xd	Up to 1.1x the raw IOPS
8 NVMe drives			
8 cores (kIOPS)	1,985.67	2,052.33	Up to 2.8 million IOPS
16 cores (kIOPS)	2,805.67	2,140.67	
36 cores (kIOPS)	2,878.33	2,142.67	
56 cores (kIOPS)	2,882.00	N/A	
12 NVMe drives			
8 cores (kIOPS)	1,947.67	2,048.33	Up to 4.2 million IOPS
16 cores (kIOPS)	3,513.67	3,198.67	
36 cores (kIOPS)	4,268.67	3,210.00	
56 cores (kIOPS)	4,293.33	N/A	
24 NVMe drives			
8 cores (kIOPS)	1,905.33	2,003.67	Up to 7.3 million IOPS
16 cores (kIOPS)	4,127.00	3,779.67	
36 cores (kIOPS)	7,614.00	6,242.33	
56 cores (kIOPS)	7,363.67	N/A	

Table 3: Results of our 1MB sequential read workload FIO benchmark testing.

Sequential read testing results	Dell PowerEdge R750	Dell PowerEdge R740xd	Up to 2.2x the GiB/s
8 NVMe drives			
8 cores: Throughput (GiB/s)	37.8	23.0	Up to 50.9 GiB/s
16 cores: Throughput (GiB/s)	40.5	23.7	
36 cores: Throughput (GiB/s)	38.1	23.8	
56 cores: Throughput (GiB/s)	50.9	N/A	
12 NVMe drives			
8 cores: Throughput (GiB/s)	53.1	23.8	Up to 53.2 GiB/s
16 cores: Throughput (GiB/s)	52.7	23.7	
36 cores: Throughput (GiB/s)	53.0	23.7	
56 cores: Throughput (GiB/s)	53.2	N/A	
24 NVMe drives			
8 cores: Throughput (GiB/s)	53.2	23.7	Up to 53.2 GiB/s
16 cores: Throughput (GiB/s)	53.2	23.7	
36 cores: Throughput (GiB/s)	53.2	23.5	
56 cores: Throughput (GiB/s)	53.2	N/A	

Table 4: Results of our 1MB sequential write workload FIO benchmark testing.

Sequential write testing results	Dell PowerEdge R750	Dell PowerEdge R740xd	Up to 1.9x the GiB/s
8 NVMe drives			
8 cores: Throughput (GiB/s)	31.3	23.5	Up to 31.6 GiB/s
16 cores: Throughput (GiB/s)	30.4	23.4	
36 cores: Throughput (GiB/s)	30.3	23.2	
56 cores: Throughput (GiB/s)	31.6	N//A	
12 NVMe drives			
8 cores: Throughput (GiB/s)	47.4	25.1	Up to 47.5 GiB/s
16 cores: Throughput (GiB/s)	47.2	25.7	
36 cores: Throughput (GiB/s)	45.8	25.6	
56 cores: Throughput (GiB/s)	47.5	N/A	
24 NVMe drives			
8 cores: Throughput (GiB/s)	50.0	25.7	Up to 49.3 GiB/s
16 cores: Throughput (GiB/s)	49.9	25.7	
36 cores: Throughput (GiB/s)	49.6	25.7	
56 cores: Throughput (GiB/s)	49.3	N/A	

System configuration information

Table 5: Detailed information on the systems we tested.

Server configuration information	Dell PowerEdge R750	Dell PowerEdge R740xd
BIOS name and version	Dell 1.3.8	Dell 2.12.2
Operating system name and version/ build number	Red Hat® Enterprise Linux® release 8.4 (Ootpa)/ 4.18.0-305.25.1.el8_4.x86_64	Red Hat Enterprise Linux release 8.4 (Ootpa)/ 4.18.0-305.25.1.el8_4.x86_64
Date of last OS updates/patches applied	1/13/22	1/13/22
Processor		
Number of processors	2	2
Vendor and model	Intel® Xeon® Gold 6330	Intel Xeon Gold 6240L Processor
Core count (per processor)	28	18
Core frequency (GHz)	2.00	2.60
Stepping	6	7
Memory module(s)		
Total memory in system (GB)	256	256
Number of memory modules	16	16
Vendor and model	Samsung® M393A2K43DB3-CWE	Micron 18ASF2G72PDZ-3G2R1
Size (GB)	16	16
Type	DDR4	DDR4
Speed (MHz)	3,200	3,200
Speed running in the server (MHz)	2,933	2,933
Storage controller		
Vendor and model	Broadcom PEX8000 series PCIe® Gen4 switch	Broadcom PEX 9733 33-lane, 9-port PCI Express Gen3 (8.0 GT/s) switch
Firmware version	N/A	4.35
Local storage		
Number of drives	24	24
Drive vendor and model	Dell Ent NVMe CM6 MU 3.2 TB	Dell NVMe MU U.2 3.2TB (P5600)
Drive size (GB)	3,200	3,200
Drive information	NVMe SSD	NVMe SSD
Network adapter		
Vendor and model	Broadcom NetXtreme BCM5720 2-port Gigabit Ethernet PCIe, Broadcom BCM57414 NetXtreme-E 10Gb/25Gb RDMA Ethernet Controller	2 x Broadcom NetXtreme Gigabit Ethernet BCM95720
Number and type of ports	2 x 1GbE, 2 x 10GbE	2 x 1GbE

Server configuration information	Dell PowerEdge R750	Dell PowerEdge R740xd
Cooling fans		
Number of cooling fans	6	6
Power Supplies		
Vendor and model	Dell 0M63JNA00	Dell 0CMPGMA03
Number of power supplies	2	2
Wattage of each (W)	2,400	1,100

System diagrams

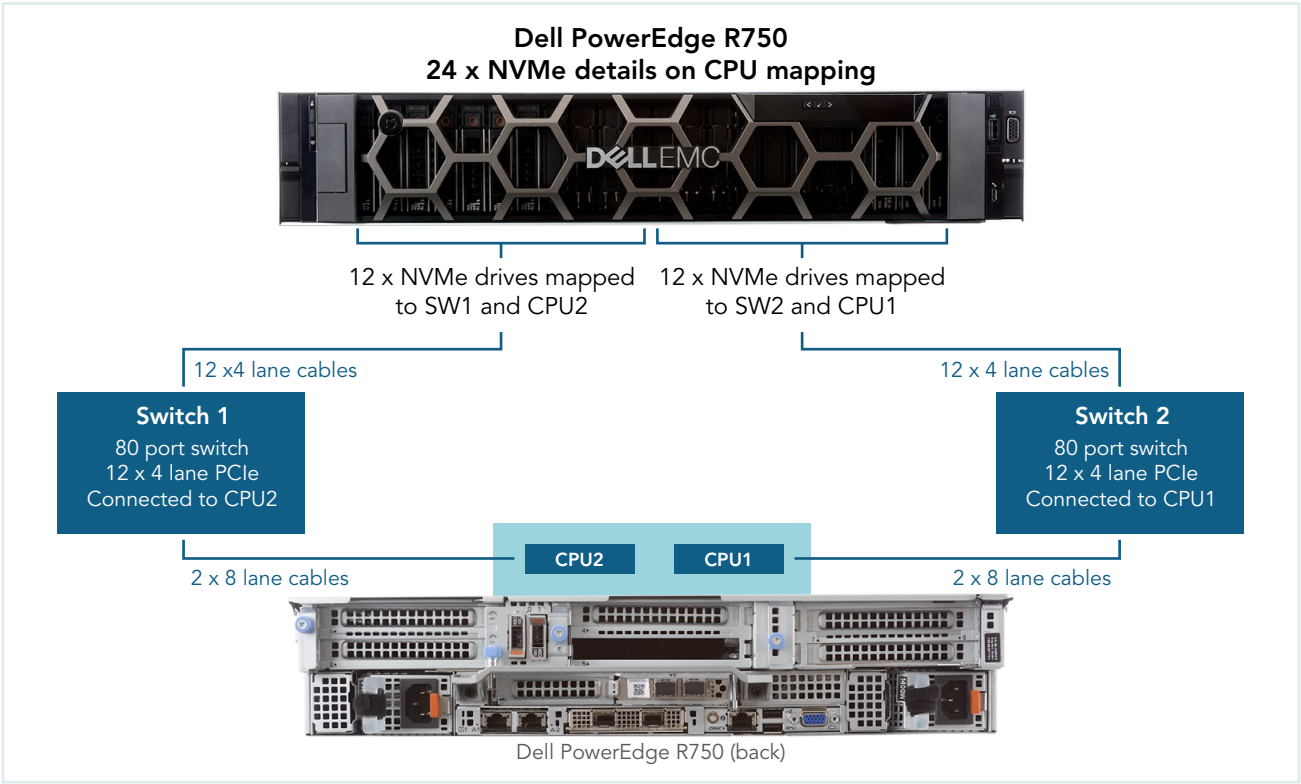

Figure 1: Dell PowerEdge R750 CPU mapping diagram. Source: Principled Technologies.

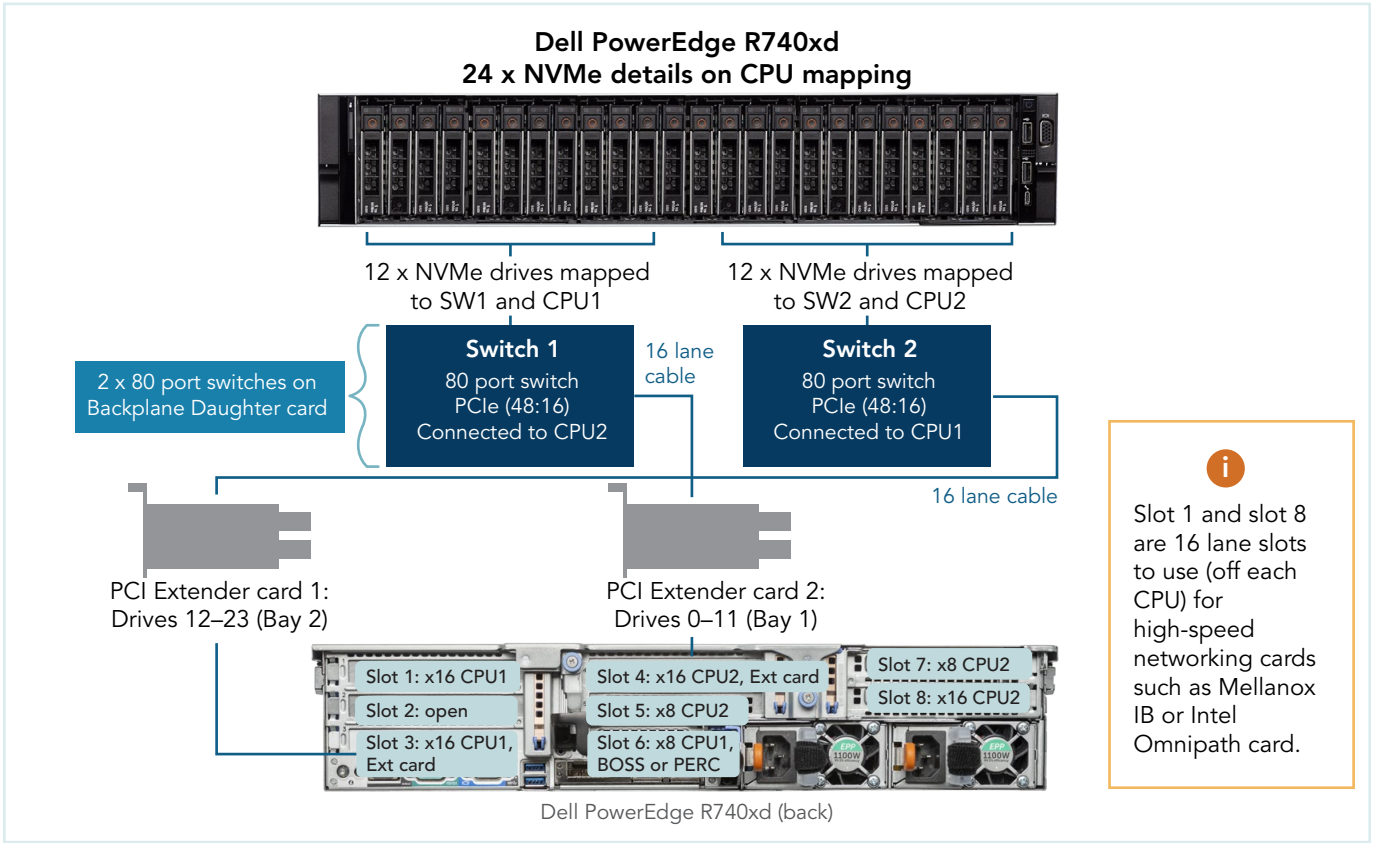

Figure 2: Dell PowerEdge R740xd CPU mapping diagram. Source: Principled Technologies.

How we tested

Both systems were dual-processor, and each processor controlled a single switch. We tested with 8, 12, and 24 NVMe drives, evenly splitting the drives between the switches. The R740xd and R750 had different NVMe drives. We also used the NVMe Command Line Interface (NVMe-CLI) software to manage 8, 12, and 24 NVMe SSDs on the PowerEdge R750.

Installing Red Hat 8.4

Install the Operating System on both the Dell PowerEdge R740xd and Dell PowerEdge R750 systems.

1. Open a browser window and connect to the iDRAC.
2. Log into the iDRAC.
3. To open it, click Virtual Console.
4. Click Connect Virtual Media.
5. Next to Map CD/DVD, click Browse.
6. Browse to the ISO for Red Hat 8.4.
7. Click Map Device.
8. Click Boot.
9. Click the Virtual CD/DVD/ISO text.
10. Click yes.
11. Click Power, and boot the machine.
12. At the Red Hat Enterprise Linux boot menu, press Up, select Install Red Hat Enterprise Linux 8.4.0, and press Enter.
13. At the Welcome screen, make sure English is selected, and click Continue.
14. At the Installation Summary screen, click Time & Date.
15. To adjust your location, use the down arrows next to Region and City, and click Done.
16. At the Installation Summary screen, click Software Selection.
17. At the Software Selection screen, click Minimal Install, and click Done.
18. At the Installation Summary screen, click Installation Destination.
19. At the Installation Destination screen, select the only non-NVMe drive. Leave Automatic selected for the Storage Configuration, and click Done.
20. At the Installation Summary screen, click Network & Host Name.
21. At the Network & Host Name screen, where it says Host Name, enter `R740xd` or `R750`, and click Apply.
22. At the top-right of the screen, next to Ethernet information, click the OFF slider to ON, allow the NIC connect, pick up an IP address from DHCP, and click Done.
23. Click Begin Installation.
24. At the Configuration screen, click Root Password. Next to Root Password and Confirm, type in your preferred password, and click Done.
25. When the Reboot button appears, click it.

Preparing scripts and running FIO tests

Prepare FIO and bash script files to automate testing and gathering results on both systems. We used NVMe-CLI to optimize drive interrupts. We created 1drive configuration files and varied the iodepth, numjobs, and other parameters to find the configuration that yielded the maximum results for 1 drive. In general, we tended to the same parameters in the 8-,12-, and 24-multidrive configurations unless we thought we could get better performance by making further adjustments.

1. Create the following bash scripts, and install the appropriate software:
    a. runtest.sh:

```
test_config=$1
cpu=$2
testno=$3
stat_collection_time=$4
hostname='hostname'
cur_time=$(date +"%Y.%m.%d_%H.%M")
host='echo $hostname | cut -d "." -f1'
OUTDIR=STATS_${cur_time}_${host}_${test_config}_${cpu}_${testno}_${RANDOM}
mkdir -p $OUTDIR
echo "Directory name is $OUTDIR"
echo "Starting fio"
fio --output-format=normal --terse-version=3 ./${test_config}.cfg output=./$OUTDIR/${test_
```

```
config}_${cpu}CPU_TEST${testno}.txt &
fiopid=$!
echo "Waiting 60 seconds for system to be in a steady state"
sleep 60
echo "Getting system stats for $stat_collection_time seconds"
mpstat -P ALL 1 > mpstat_out.txt &
pidmpstat=$!
dstat -t -v --nocolor > dstat_out.txt &
piddstat=$!
iostat -xmt 1 > iostat_out.txt &
pidiostat=$!
sleep $stat_collection_time
echo "Killing mpstat"
kill -9 $pidmpstat
echo "Killing dstat"
kill -9 $piddstat
echo "Killing iostat"
                    kill -9 $pidiostat
echo "Waiting for fio to complete"
while kill -0 $fiopid 2> /dev/null; do
     sleep 10
done
mv mpstat_out.txt ${OUTDIR}/mpstat_out.txt
mv dstat_out.txt ${OUTDIR}/dstat_out.txt
mv iostat_out.txt ${OUTDIR}/iostat_out.txt
echo "Test $OUTDIR_complete"
```

b. Get_Sys_Info_NVMeDir.sh:

```
#!/bin/sh'
################################################################################'
### THE FOLLOWING VARIABLES CAN BE MODIFIED FOR YOUR SPECIFIC TEST REQUIREMENTS ###'
################################################################################'
'
'
storcli="storcli64"'
controller_number=0'
hostname='hostname''
cur_time=$(date "+%Y.%m.%d-%H.%m")'
host='echo $hostname | cut -d "." -f1''
OUTDIR=SysInfo_${RANDOM}_${host}_${cur_time}'
echo "*** All results and system information can be found in the $OUTDIR directory."'
mkdir -p $OUTDIR'
echo'
'
#########################################################'
############## Collecting System Information #############'
#########################################################'
'
collect_system_info() {'
echo'
echo "#########################################"'
echo "####     COLLECTING SYSTEM INFO       ####"'
echo "#########################################"'
echo'
'
'
'
# Collect System Info'
echo -e 'hostname' >> $OUTDIR/Sys_Config.txt'
echo -e "OS: 'cat /etc/redhat-release'\n" >> $OUTDIR/Sys_Config.txt'
echo -e "Proc Version: 'cat /proc/version'\n" >> $OUTDIR/Sys_Config.txt'
echo -e "\nMemory Information:" >> $OUTDIR/Sys_Config.txt'
cat /proc/meminfo >> $OUTDIR/Sys_Config.txt'
echo -e "\nProcessor Information:" >> $OUTDIR/Sys_Config.txt'
cat /proc/cpuinfo >> $OUTDIR/Sys_Config.txt'
```

```
echo -e "\nPCI Information:" >> $OUTDIR/Sys_Config.txt'
lspci -vvv  >> $OUTDIR/PCI_Config.txt'
echo -e "\nSystem DMI Information:" >> $OUTDIR/Sys_Config.txt'
dmidecode >> $OUTDIR/Sys_Config.txt'
echo -e "\nChecking SCSI MQ Settings, use_blk_mq is set to: 'cat /sys/module/scsi_mod/
parameters/use_blk_mq'\n" >> $OUTDIR/Sys_Config.txt'
numactl --hardware >> $OUTDIR/NUMA.txt  2> /dev/null'
nvme list >> $OUTDIR/nvmelist.txt'
'
rm -rf storcli.log'
echo "*** System specific information can be found in $OUTDIR\\SystemInfo directory"'
echo'
sleep 1'
}'
'
'
'
############################################################'
###################### MAIN PROGRAM ######################'
############################################################'
'
'
'
collect_system_info'
'
Enter the following commands'
yum -y install sysstat'
yum -y install pcp-system-tools'
yum -y install numactl'
chmod a+x *.sh
```

2. Create the FIO configuration files with *.cfg as the file extension. We provide an example file used for each workload here, along with a table 6 listing what parameters we adjusted for individual tests. All files we provide are for the 8-drive tests but have additional lines that you can uncomment in order to test more drives.

    a. Random read configuration file:

```
[global]
name=4k_random_read
scramble_buffers=1
buffered=0
sync=0
ioengine=libaio
norandommap
do_verify=0
randrepeat=0
random_generator=tausworthe64
offset=0
thinktime=0
time_based=1
ramp_time=5m
direct=1
overwrite=0
group_reporting=1
bs=4k
blockalign=4k
rw=randread
numa_cpu_nodes=0
cpus_allowed_policy=shared
iodepth=32
numjobs=4
runtime=60s

[part1]
filename=/dev/nvme0n1
numa_cpu_nodes=0
```

```
[part2]
filename=/dev/nvme12n1
numa_cpu_nodes=1

[part3]
filename=/dev/nvme2n1
numa_cpu_nodes=0

[part4]
filename=/dev/nvme14n1
numa_cpu_nodes=1

[part5]
filename=/dev/nvme4n1
numa_cpu_nodes=0

[part6]
filename=/dev/nvme16n1
numa_cpu_nodes=1

[part7]
filename=/dev/nvme6n1
numa_cpu_nodes=0

[part8]
filename=/dev/nvme18n1
numa_cpu_nodes=1

#[part9]
#filename=/dev/nvme8n1
#numa_cpu_nodes=0

#[part10]
#filename=/dev/nvme20n1
#numa_cpu_nodes=1

#[part11]
#filename=/dev/nvme10n1
#numa_cpu_nodes=0

#[part12]
#filename=/dev/nvme22n1
#numa_cpu_nodes=1

#[part13]
#filename=/dev/nvme1n1
#numa_cpu_nodes=0

#[part14]
#filename=/dev/nvme13n1
#numa_cpu_nodes=1

#[part15]
#filename=/dev/nvme3n1
#numa_cpu_nodes=0

#[part16]
#filename=/dev/nvme15n1
#numa_cpu_nodes=1

#[part17]
#filename=/dev/nvme5n1
#numa_cpu_nodes=0

#[part18]
#filename=/dev/nvme17n1
#numa_cpu_nodes=1
```

```
#[part19]
#filename=/dev/nvme7n1
#numa_cpu_nodes=0

#[part20]
#filename=/dev/nvme19n1
#numa_cpu_nodes=1

#[part21]
#filename=/dev/nvme9n1
#numa_cpu_nodes=0

#[part22]
#filename=/dev/nvme21n1
#numa_cpu_nodes=1

#[part23]
#filename=/dev/nvme11n1
#numa_cpu_nodes=0

#[part24]
#filename=/dev/nvme23n1
#numa_cpu_nodes=1

Random write configuration file:
    [global]
name=4k_random_write
scramble_buffers=1
buffered=0
sync=0
ioengine=libaio
norandommap
do_verify=0
randrepeat=0
random_generator=tausworthe64
offset=0
thinktime=0
time_based=1
ramp_time=5m
direct=1
overwrite=0
group_reporting=1
bs=4k
blockalign=4k
rw=randwrite
numa_cpu_nodes=0
cpus_allowed_policy=shared
iodepth=4
numjobs=2

runtime=60s
[part1]
filename=/dev/nvme0n1
numa_cpu_nodes=0

[part2]
filename=/dev/nvme12n1
numa_cpu_nodes=1

[part3]
filename=/dev/nvme2n1
numa_cpu_nodes=0

[part4]
filename=/dev/nvme14n1
numa_cpu_nodes=1
```

```
[part5]
filename=/dev/nvme4n1
numa_cpu_nodes=0

[part6]
filename=/dev/nvme16n1
numa_cpu_nodes=1

[part7]
filename=/dev/nvme6n1
numa_cpu_nodes=0

[part8]
filename=/dev/nvme18n1
numa_cpu_nodes=1

#[part9]
#filename=/dev/nvme8n1
#numa_cpu_nodes=0

#[part10]
#filename=/dev/nvme20n1
#numa_cpu_nodes=1

#[part11]
#filename=/dev/nvme10n1
#numa_cpu_nodes=0

#[part12]
#filename=/dev/nvme22n1
#numa_cpu_nodes=1

#[part13]
#filename=/dev/nvme1n1
#numa_cpu_nodes=0

#[part14]
#filename=/dev/nvme13n1
#numa_cpu_nodes=1

#[part15]
#filename=/dev/nvme3n1
#numa_cpu_nodes=0

#[part16]
#filename=/dev/nvme15n1
#numa_cpu_nodes=1

#[part17]
#filename=/dev/nvme5n1
#numa_cpu_nodes=0

#[part18]
#filename=/dev/nvme17n1
#numa_cpu_nodes=1

#[part19]
#filename=/dev/nvme7n1
#numa_cpu_nodes=0

#[part20]
#filename=/dev/nvme19n1
#numa_cpu_nodes=1

#[part21]
#filename=/dev/nvme9n1
```

```
#numa_cpu_nodes=0

#[part22]
#filename=/dev/nvme21n1
#numa_cpu_nodes=1

#[part23]
#filename=/dev/nvme11n1
#numa_cpu_nodes=0

#[part24]
#filename=/dev/nvme23n1
#numa_cpu_nodes=1
```

b. Sequential read configuration file:

```
[global]
name=1m_sequential_read
scramble_buffers=1
buffered=0
sync=0
ioengine=libaio
norandommap
do_verify=0
randrepeat=0
random_generator=tausworthe64
offset=0
thinktime=0
time_based=1
ramp_time=5m
direct=1
overwrite=0
group_reporting=1
bs=1024k
blockalign=4k
rw=read
numa_cpu_nodes=0
cpus_allowed_policy=shared
iodepth=32
numjobs=1
runtime=5m

[part1]
filename=/dev/nvme0n1
numa_cpu_nodes=0

[part2]
filename=/dev/nvme12n1
numa_cpu_nodes=1

[part3]
filename=/dev/nvme2n1
numa_cpu_nodes=0

[part4]
filename=/dev/nvme14n1
numa_cpu_nodes=1

[part5]
filename=/dev/nvme4n1
numa_cpu_nodes=0

[part6]
filename=/dev/nvme16n1
```

```
numa_cpu_nodes=1

[part7]
filename=/dev/nvme6n1
numa_cpu_nodes=0

[part8]
filename=/dev/nvme18n1
numa_cpu_nodes=1

#[part9]
#filename=/dev/nvme8n1
#numa_cpu_nodes=0

#[part10]
#filename=/dev/nvme20n1
#numa_cpu_nodes=1

#[part11]
#filename=/dev/nvme10n1
#numa_cpu_nodes=0

#[part12]
#filename=/dev/nvme22n1
#numa_cpu_nodes=1

#[part13]
#filename=/dev/nvme1n1
#numa_cpu_nodes=0

#[part14]
#filename=/dev/nvme13n1
#numa_cpu_nodes=1

#[part15]
#filename=/dev/nvme3n1
#numa_cpu_nodes=0

#[part16]
#filename=/dev/nvme15n1
#numa_cpu_nodes=1

#[part17]
#filename=/dev/nvme5n1
#numa_cpu_nodes=0

#[part18]
#filename=/dev/nvme17n1
#numa_cpu_nodes=1

#[part19]
#filename=/dev/nvme7n1
#numa_cpu_nodes=0

#[part20]
#filename=/dev/nvme19n1
#numa_cpu_nodes=1

#[part21]
#filename=/dev/nvme9n1
#numa_cpu_nodes=0

#[part22]
#filename=/dev/nvme21n1
#numa_cpu_nodes=1

#[part23]
```

```
#filename=/dev/nvme11n1
#numa_cpu_nodes=0

#[part24]
#filename=/dev/nvme23n1
#numa_cpu_nodes=1
```

c. Sequential write configuration file:

```
[global]
name=1m_sequential_write
scramble_buffers=1
buffered=0
sync=0
ioengine=libaio
norandommap
do_verify=0
randrepeat=0
random_generator=tausworthe64
offset=0
thinktime=0
time_based=1
ramp_time=5m
direct=1
overwrite=0
group_reporting=1
bs=1024k
blockalign=4k
rw=write
numa_cpu_nodes=0
cpus_allowed_policy=shared
iodepth=32
numjobs=1
runtime=5m

[part1]
filename=/dev/nvme0n1
numa_cpu_nodes=0

[part2]
filename=/dev/nvme12n1
numa_cpu_nodes=1

[part3]
filename=/dev/nvme2n1
numa_cpu_nodes=0

[part4]
filename=/dev/nvme14n1
numa_cpu_nodes=1

[part5]
filename=/dev/nvme4n1
numa_cpu_nodes=0

[part6]
filename=/dev/nvme16n1
numa_cpu_nodes=1

[part7]
filename=/dev/nvme6n1
numa_cpu_nodes=0

[part8]
filename=/dev/nvme18n1
```

```
numa_cpu_nodes=1

#[part9]
#filename=/dev/nvme8n1
#numa_cpu_nodes=0

#[part10]
#filename=/dev/nvme20n1
#numa_cpu_nodes=1

#[part11]
#filename=/dev/nvme10n1
#numa_cpu_nodes=0

#[part12]
#filename=/dev/nvme22n1
#numa_cpu_nodes=1

#[part13]
#filename=/dev/nvme1n1
#numa_cpu_nodes=0

#[part14]
#filename=/dev/nvme13n1
#numa_cpu_nodes=1

#[part15]
#filename=/dev/nvme3n1
#numa_cpu_nodes=0

#[part16]
#filename=/dev/nvme15n1
#numa_cpu_nodes=1

#[part17]
#filename=/dev/nvme5n1
#numa_cpu_nodes=0

#[part18]
#filename=/dev/nvme17n1
#numa_cpu_nodes=1

#[part19]
#filename=/dev/nvme7n1
#numa_cpu_nodes=0

#[part20]
#filename=/dev/nvme19n1
#numa_cpu_nodes=1

#[part21]
#filename=/dev/nvme9n1
#numa_cpu_nodes=0

#[part22]
#filename=/dev/nvme21n1
#numa_cpu_nodes=1

#[part23]
#filename=/dev/nvme11n1
#numa_cpu_nodes=0

#[part24]
#filename=/dev/nvme23n1
#numa_cpu_nodes=1
```

Table 6: Parameters we adjusted for each test.

System	Workload	Drives	Cores	numjobs	iodepth
R740xd	Random read	8	8	4	16
R740xd	Random read	8	16	4	16
R740xd	Random read	8	36	4	16
R740xd	Random read	12	8	4	16
R740xd	Random read	12	16	4	16
R740xd	Random read	12	36	4	16
R740xd	Random read	24	8	4	16
R740xd	Random read	24	16	4	16
R740xd	Random read	24	36	4	16
R740xd	Random write	8	8	8	32
R740xd	Random write	8	16	8	32
R740xd	Random write	8	36	8	32
R740xd	Random write	12	8	8	32
R740xd	Random write	12	16	8	32
R740xd	Random write	12	36	8	32
R740xd	Random write	24	8	8	32
R740xd	Random write	24	16	8	32
R740xd	Random write	24	36	8	32
R740xd	Sequential read	8	8	1	32
R740xd	Sequential read	8	16	1	32
R740xd	Sequential read	8	36	1	32
R740xd	Sequential read	12	8	1	32
R740xd	Sequential read	12	16	1	32
R740xd	Sequential read	12	36	1	32
R740xd	Sequential read	24	8	1	32
R740xd	Sequential read	24	16	1	32
R740xd	Sequential read	24	36	1	32
R740xd	Sequential write	8	8	1	32
R740xd	Sequential write	8	16	1	32
R740xd	Sequential write	8	36	1	32
R740xd	Sequential write	12	8	1	32
R740xd	Sequential write	12	16	1	32
R740xd	Sequential write	12	36	1	32
R740xd	Sequential write	24	8	1	32
R740xd	Sequential write	24	16	1	32
R740xd	Sequential write	24	36	1	32
R750	Random read	8	8	4	32
R750	Random read	8	16	4	32
R750	Random read	8	36	8	64
R750	Random read	8	56	8	64
R750	Random read	12	8	4	32
R750	Random read	12	16	4	32
R750	Random read	12	36	8	64

R750	Random read	12	56	8	64
R750	Random read	24	8	4	32
R750	Random read	24	16	4	32
R750	Random read	24	36	4	32
R750	Random read	24	56	4	32
R750	Random write	8	8	2	4
R750	Random write	8	16	2	4
R750	Random write	8	36	2	4
R750	Random write	8	56	2	4
R750	Random write	12	8	2	4
R750	Random write	12	16	2	4
R750	Random write	12	36	2	4
R750	Random write	12	56	2	4
R750	Random write	24	8	2	4
R750	Random write	24	16	2	4
R750	Random write	24	36	2	4
R750	Random write	24	56	2	4
R750	Sequential read	8	8	1	32
R750	Sequential read	8	16	1	32
R750	Sequential read	8	36	1	32
R750	Sequential read	8	56	1	32
R750	Sequential read	12	8	1	32
R750	Sequential read	12	16	1	32
R750	Sequential read	12	36	1	32
R750	Sequential read	12	56	1	32
R750	Sequential read	24	8	1	32
R750	Sequential read	24	16	1	32
R750	Sequential read	24	36	1	32
R750	Sequential read	24	56	1	32
R750	Sequential write	8	8	1	32
R750	Sequential write	8	16	1	32
R750	Sequential write	8	36	1	32
R750	Sequential write	8	56	1	32
R750	Sequential write	12	8	1	32
R750	Sequential write	12	16	1	32
R750	Sequential write	12	36	1	32
R750	Sequential write	12	56	1	32
R750	Sequential write	24	8	1	32
R750	Sequential write	24	16	1	32
R750	Sequential write	24	36	1	32
R750	Sequential write	24	56	1	32

3. To optimize and run the NVMe drives, install the appropriate software, and create the bash scripts. Note: You must rerun these after every reboot.

    a. Create the fixnvme.sh bash script file on each server:

```
nvme set-feature /dev/nvme0 -f=8 --value=0x108
nvme set-feature /dev/nvme1  -f=8 --value=0x108
nvme set-feature /dev/nvme2  -f=8 --value=0x108
nvme set-feature /dev/nvme3  -f=8 --value=0x108
nvme set-feature /dev/nvme4  -f=8 --value=0x108
nvme set-feature /dev/nvme5  -f=8 --value=0x108
nvme set-feature /dev/nvme6  -f=8 --value=0x108
nvme set-feature /dev/nvme7  -f=8 --value=0x108
nvme set-feature /dev/nvme8  -f=8 --value=0x108
nvme set-feature /dev/nvme9  -f=8 --value=0x108
nvme set-feature /dev/nvme10 -f=8 --value=0x108
nvme set-feature /dev/nvme11 -f=8 --value=0x108
nvme set-feature /dev/nvme12 -f=8 --value=0x108
nvme set-feature /dev/nvme13 -f=8 --value=0x108
nvme set-feature /dev/nvme14 -f=8 --value=0x108
nvme set-feature /dev/nvme15 -f=8 --value=0x108
nvme set-feature /dev/nvme16 -f=8 --value=0x108
nvme set-feature /dev/nvme17 -f=8 --value=0x108
nvme set-feature /dev/nvme18 -f=8 --value=0x108
nvme set-feature /dev/nvme19 -f=8 --value=0x108
nvme set-feature /dev/nvme20 -f=8 --value=0x108
nvme set-feature /dev/nvme21 -f=8 --value=0x108
nvme set-feature /dev/nvme22 -f=8 --value=0x108
nvme set-feature /dev/nvme23 -f=8 --value=0x108
```

    b. To install the appropriate software, enter the following commands for each server:

```
yum install nvme-cli
chmod a+x *.sh
```

4. Adjust the number of cores per processor as desired, by performing the following steps:

    a. At the virtual console, click Boot.
    b. Click BIOS setup.
    c. Click Yes.
    d. At the console, enter the command `reboot`.
    e. At the System Setup Main Menu, click System BIOS.
    f. At the System BIOS menu, click Processor Settings.
    g. At the Processor Settings menu, click the down arrow, next to Number of Cores per Processor, and select half the number of cores that you want to work with (two processors together will give you the total cores you want).
    h. Click Back.
    i. Click Finish.
    j. At the Warning - Saving Changes dialogue box, click Yes.
    k. At the Success - Saving Changes dialogue box, click OK.
    l. Click Finish.
    m. At the Warning - Confirm Exit dialogue box, click Yes.

5. After the system boots, run your tests using the bash scripts by entering the following commands at the console:

```
./fixnvme.sh
./Get_Sys_Info_NVMeDir.sh
./runtest.sh <desired fio test config> <number of processors configured> <test number> 60
```

6. View your results in the created, appropriately named directory.

Read the report at https://facts.pt/yjZGZ6O ▶

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®