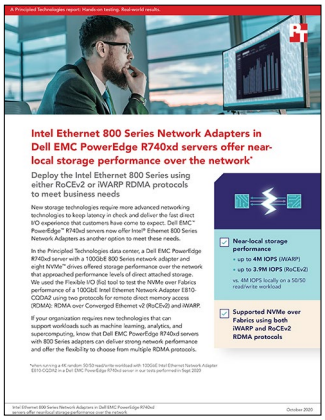

The science behind the report:

Intel Ethernet 800 Series Network Adapters in Dell EMC PowerEdge R740xd servers offer near-local storage performance over the network

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 Intel Ethernet 800 Series Network Adapters in Dell EMC PowerEdge R740xd servers offer near-local storage performance over the network.

We concluded our hands-on testing on September 12, 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 September 9, 2020 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

Table 1: Combined IOPS, CPU utilization, and bandwidth for each read/write ratio we tested across three configurations. We present the median result of three runs.

Protocol	% Read	Avg. Total IOPS	Avg. server CPU % utilization	Avg. client CPU % utilization	Avg. bandwidth (MB/s)
Four NVMe drives					
iWARP	0	2,410,234	11.89	22.62	9,415
RoCEv2	0	2,413,310	11.89	21.22	9,427
Direct attached storage	0	2,406,702	32.42	-	9,401
iWARP	50	2,009,693	11.82	17.2	7,850
RoCEv2	50	2,010,771	11.8	16.43	7,855
Direct attached storage	50	2,011,716	25.03	-	7,858
iWARP	70	2,069,574	12.04	16.51	8,084
RoCEv2	70	2,070,212	11.9	16.17	8,087
Direct attached storage	70	2,070,630	24.85	-	8,088
iWARP	100	2,349,313	12.59	15.1	9,177
RoCEv2	100	2,349,431	12.56	15.41	9,177
Direct attached storage	100	2,349,803	27.64	-	9,179

Protocol	% Read	Avg. Total IOPS	Avg. server CPU % utilization	Avg. client CPU % utilization	Avg. bandwidth (MB/s)
Six NVMe drives					
iWARP	0	2,725,289	11.89	20.56	10,646
RoCEv2	0	2,708,243	11.98	20.52	10,579
Direct attached storage	0	3,244,172	42.6	-	12,673
iWARP	50	3,013,320	14.41	22.45	11,771
RoCEv2	50	3,015,917	14.42	20.93	11,781
Direct attached storage	50	3,015,675	40.02	-	11,780
iWARP	70	3,102,797	14.55	20.79	12,120
RoCEv2	70	3,105,282	14.83	19.75	12,130
Direct attached storage	70	3,104,904	40.08	-	12,129
iWARP	100	2,773,029	13.25	16.63	10,832
RoCEv2	100	2,738,298	13.17	15.9	10,696
Direct attached storage	100	3,302,506	41.46	-	12,900
Eight NVMe drives					
iWARP	0	2,734,628	11.85	19.64	10,682
RoCEv2	0	2,708,037	12	20.83	10,578
Direct attached storage	0	3,268,552	43.12	-	12,768
iWARP	50	4,012,470	16.93	27.15	15,674
RoCEv2	50	3,811,799	16.55	24.16	14,890
Direct attached storage	50	4,019,408	46	-	15,701
iWARP	70	3,774,315	16.17	21.92	14,743
RoCEv2	70	3,729,388	16.14	21.45	14,568
Direct attached storage	70	4,078,633	45.86	-	15,932
iWARP	100	2,773,089	13.3	15.8	10,832
RoCEv2	100	2,738,625	13.45	15.67	10,698
Direct attached storage	100	3,255,987	41.27	-	12,719

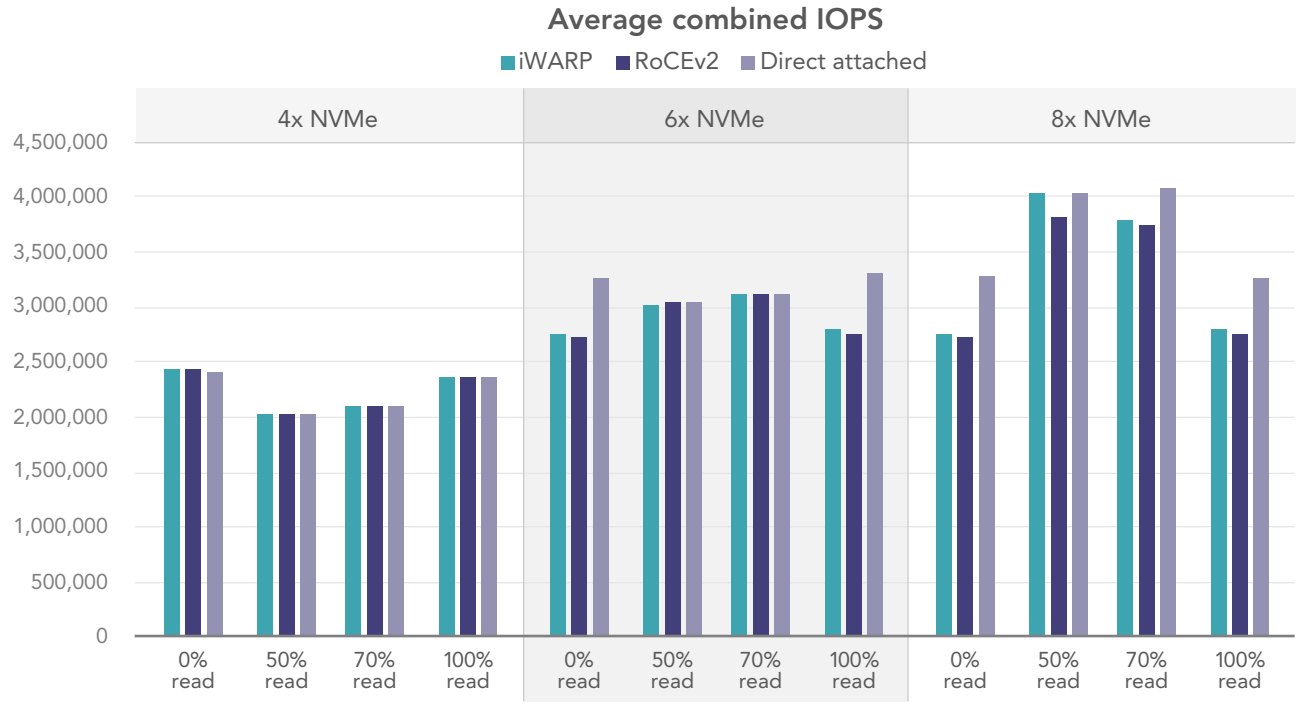

Figure 1: Combined IOPS for each read/write ratio we tested across three configurations. Higher is better.
Source: Principled Technologies

System configuration information

Table 2: Detailed information for our test systems.

Server configuration information	Dell EMC PowerEdge R740xd	Dell EMC PowerEdge R740
BIOS name and version	Dell 2.8.1	Dell 2.8.1
Non-default BIOS settings	N/A	N/A
Operating system name and version/build number	CentOS 8.2 kernel 4.18.0-193.14.2.el8_2.x86_64	CentOS 8.2 kernel 4.18.0-193.14.2.el8_2.x86_64
Date of last OS updates/patches applied	8/21/2020	8/21/2020
Power management policy	Performance	Performance
Processor		
Number of processors	2	2
Vendor and model	Intel® Xeon® Gold 6242	Intel Xeon Platinum 8168
Core count (per processor)	16	24
Core frequency (GHz)	2.8	2.7
Stepping	6	4
Memory module(s)		
Total memory in system (GB)	192	96
Number of memory modules	12	12
Vendor and model	Hynix® HMA82GR7AFR8N-VK	Hynix HMA41GR7MFR8N-TF
Size (GB)	16	8
Type	DDR4	DDR4
Speed (MHz)	2,666	2,133
Speed running in the server (MHz)	2,666	2,133
Storage controller		
Vendor and model	Dell PERC H730P	Dell PERC H740P
Cache size (MB)	2,048	8,192
Firmware version	25.5.6.0009	50.9.4-3025
Driver version	07.710.50.00-rc1	07.710.50.00-rc1
Local storage (type A)		
Number of drives	2	2
Drive vendor and model	Dell ST1200MM0099 (Seagate)	Dell ST1200MM0099 (Seagate)
Drive size (GB)	1,200	1,200
Drive information (interface, type)	SAS HDD	SAS HDD

Server configuration information	Dell EMC PowerEdge R740xd	Dell EMC PowerEdge R740
Local storage (type B)		
Number of drives	8	N/A
Drive vendor and model	Intel Optane™ SSD DC P4800X	N/A
Drive size (GB)	375	N/A
Drive information (interface, type)	PCIe NVMe	N/A
Network adapter		
Vendor and model	Intel Ethernet Connection E810-C	Intel Ethernet Connection E810-C
Number and type of ports	2 x 100GbE	2 x 100GbE
Cooling fans		
Vendor and model	Nidec® UltraFlo 4VXP3-X30	Nidec UltraFlo 4VXP3-X30
Number of cooling fans	6	6
Power supplies		
Vendor and model	Dell 0PJMDN	Dell 0GRTNK
Number of power supplies	2	2
Wattage of each (W)	750	495

How we tested

Changing the BIOS System Profile to Performance

1. Power on the System.
2. Enter BIOS and change the System Profile to Performance.

Installing CentOS Linux 8.2

1. Boot to the CentOS Linux 8.2 installation media using the CentOS-8.2.2004-x86_64-minimal.iso image.
2. Select Install CentOS Linux 8.
3. Choose English, and click Continue.
4. Under Installation Destination, select the desired disk to install the OS.
5. Under Storage Configuration, select Custom, and click Done.
6. Select Click here to create them automatically.
7. If it exists, remove the /home partition.
8. Expand the swap partition to 4GB.
9. Assign all remaining free space to the / partition.
10. Click Done.
11. Click Accept Changes.
12. Select Network & Hostname.
13. Enter the desired hostname for the system.
14. Turn on the desired network ports, and click Configure.
15. On the General tab, select Connect automatically with priority.
16. On the IPv4 Settings tab, choose the Method drop-down, and select Manual.
17. Under Addresses, click Add, and enter the desired static IP information for the server.
18. Enter the desired DNS information.
19. Click Save, and click Done.
20. Select Date & Time, and ensure the correct date, time, and time zone are set.
21. Click the cog next to the Network Time On/Off switch to add your NTP server.
22. Add the IP address of your NTP server and click +.
23. Uncheck all other NTP servers.
24. Click OK.
25. Click Done.
26. Click Software Selection.
27. Choose the Base Environment of Minimal Install.
28. Click Done.
29. Click Begin Installation.
30. Select Root Password.
31. Enter the desired root password, and click Done.
32. When the installation completes, select Reboot to restart the server.

Configuring CentOS Linux 8

1. Log into the server as root.
2. Disable the firewall and SELinux with the following commands:

```
setenforce 0
sed -i 's/SELINUX=.*/SELINUX=disabled/' /etc/selinux/config
systemctl disable --now firewalld
```

3. Tune the system for low network latency using the following command:

```
tuned-adm profile network-latency
```

4. Install the EPEL Repository and additional tools::

```
dnf install -y epel-release
dnf install -y wget vim tar zip unzip numactl sysstat nmon ksh pciutils ipmitool gdisk
```

5. Update the host:

```
dnf update -y
```

6. Reboot the host.

Installing the test software, network adapter driver, and network adapter firmware

1. Extract the test software:

```
dnf groupinstall -y "Development Tools"
dnf install -y elfutils-libelf-devel
```

2. Build and install the driver:

```
cd PROCGB/Linux/
tar -xf ice-1.0.4.tar.gz
cd ice-1.0.4/src/
make
make install
modinfo ice
modprobe -r ice
modprobe ice
cd ../../../../
```

3. Update the network adapter firmware:

```
cd NVMUpdatePackage/E810/
tar -xf E810_NVMUpdatePackage_v2_00_Linux.tar.gz
cd E810/Linux_x64/
./nvmupdate64e -u -b -rd
modprobe -r ice
modprobe ice
cd ../../../../
```

Installing and configuring RDMA-related software

1. Install IRDMA:

```
cd PROCGB/Linux/
tar -xf irdma-1.0.13.tgz
cd irdma-1.0.13/
./build.sh
modprobe irdma roce_ena=1
```

2. Patch and install RDMA Core userspace libraries and daemons:

```
dnf erase -y rdma-core
dnf install -y epel-release
dnf config-manager --set-enabled PowerTools
dnf install -y valgrind valgrind-devel libnl3 libnl3-devel cmake3 python3 python3-docutils python3-
devel python3-Cython libudev-devel ninja-build pandoc systemd-devel perl-generators
wget https://github.com/linux-rdma/rdma-core/releases/download/v27.0/rdma-core-27.0.tar.gz
tar -xzvf rdma-core-27.0.tar.gz
cd rdma-core-27.0
patch -p2 < ../libirdma-27.0.patch
cd ../
chgrp -R root rdma-core-27.0/redhat
tar -zcvf rdma-core-27.0.tgz rdma-core-27.0
mkdir -p ~/rpmbuild/SOURCES
mkdir -p ~/rpmbuild/SPECS
cp rdma-core-27.0.tgz ~/rpmbuild/SOURCES/
cd ~/rpmbuild/SOURCES
tar -xzvf rdma-core-27.0.tgz
cp ~/rpmbuild/SOURCES/rdma-core-27.0/redhat/rdma-core.spec ~/rpmbuild/SPECS/
cd ~/rpmbuild/SPECS/
rpmbuild -ba rdma-core.spec
cd ~/rpmbuild/RPMS/x86_64
dnf install -y *27.0*.rpm
```

3. Verify iWARP on ports:

```
ibv_devices
```

4. Enable flow control on network ports:

```
ethtool -A <NETWORK_DEVICE> rx on tx on
```

5. To enable RoCE, follow the remaining steps:

```
echo 'options irdma roce_ena=1' > /etc/modprobe.d/irdma.conf
dracut -f
```

6. Reboot the host.
7. Verify RoCE on ports:

```
ibv_devices
```

Setting up and configuring NVMe over Fabrics (NVMeoF) on the target server

1. On the target server, install the tools for NVMeoF:

```
dnf install -y nvme-cli nvmetcli
```

2. Format NVMe devices:

```
for device in $(ls /dev/nvme*n*); do
nvme format $device
done
```

3. Make and load the NVMe target configuration file:

```
vim gen_target_config_kernel_doc.sh
chmod +x gen_target_config_kernel_doc.sh
./gen_target_config_kernel_doc.sh
nvmetcli clear
nvmetcli restore nvme-target-setup-auto.json
```

Setting up and configuring NVMeoF on the client system

1. On the client system, install the tools for NVMeoF:

```
dnf install -y nvme-cli
```

2. Connect to the target NVMeoF server:

```
modprobe configfs
modprobe nvme
modprobe nvme_rdma
nvme connect-all -t rdma -s 4420 -a <TARGET_SERVER_IP>
```

Running the benchmark

1. On the target server, run the benchmark script by issuing the following command:

```
./run_test.sh <test_host>
```

Scripts and configuration files

gen_target_config_kernel_doc.sh

```
#!/bin/bash
#This script creates nvmetcli configuration file that uses all nvme partitions in /dev/nvme*n*p* for
target subsystem creation.
#Note: be sure to change $ip to match your RDMA interface IP
#On NVMe target, after running this script, use 'nvmetcli restore <filen>' to create nvme subsystems

#file name for config file.
filen="nvme-target-setup-auto.json"

#RDMA target interface IP
ip="192.168.79.11"

#get all NVMe drive partition paths
nvmePartitions=( $(ls /dev/nvme*n*p* | cut -d'/' -f3) )

#get last partition path in the list
lastnvmePartition=${nvmePartitions[$(( ${#nvmePartitions[*]} - 1))]}

#create config file
cat > $filen << EOF
{
    "hosts": [],
    "ports": [
       {
       "addr": {
          "adrfam": "ipv4",
       "traddr": "$ip",
       "treq": "not specified",
       "trsvcid": "4420",
       "trtype": "rdma"
       },
    "portid": 1,
    "referrals": [],
    "subsystems": [
EOF

for dev in "${nvmePartitions[@]}"; do
if [[ $dev == "$lastnvmePartition" ]]; then
cat >> $filen << EOF
    "${dev}"
EOF
else
cat >> $filen << EOF
    "${dev}",
EOF
fi
done

cat >> $filen << EOF
        ]
      }
    ],
    "subsystems": [
EOF

i=1
for dev in "${nvmePartitions[@]}"; do
cat >> $filen << EOF
      {
        "allowed_hosts": [],
        "attr": {
```

```
        "allow_any_host": "1"
      },
      "namespaces": [
      {
        "device": {
        "nguid": "00000000-0000-0000-0000-000000000000",
        "path": "/dev/$dev"
      },
      "enable": 1,
      "nsid": 1
    }
  ],
  "nqn": "$dev"
EOF

if [[ $dev == "$lastnvmePartition" ]]; then
cat >> $filen << EOF
  }
EOF
else
cat >> $filen << EOF
  },
EOF
fi
done

cat >> $filen << EOF
  ]
}
EOF
```

run_test.sh

```
#!/bin/bash
RUNTIME=90
WARMUP=10
STEP=3
COUNT=$((RUNTIME/STEP))
TIMESTAMP=$(date '+%Y%m%d_%H%M%S')
PREFIX=iwarp

TEST_HOST=${1}
NVME_HOST=nvme-server
NVME_CLIENT=nvme-client

GLOBAL_FILES="global.fio"
#JOB_FILES="$(echo ${TEST_HOST}_{4,6,8}xNVMe.fio)"
JOB_FILES="$(echo ${TEST_HOST}_8xNVMe.fio)"
JOB_TYPES="randrw" # --rw
#JOB_MIXES="100 70 50 0" # --rwmixread
JOB_MIXES="50"
JOB_BLOCKSIZES="4K" # --bs

echo "TEST HOST: "
ssh ${TEST_HOST} "hostname" || exit
echo "NVME SERVER: "
ssh ${NVME_HOST} "hostname" || exit
sleep 1
echo

echo "Copying job files:"
if [[ "$TEST_HOST" != "$NVME_HOST" ]]; then
  for file in ${GLOBAL_FILES} ${JOB_FILES}; do
    scp -p $file ${TEST_HOST}: || exit
  done
```

```
fi
sleep 1
echo


for job_file in ${JOB_FILES};
do
  for job_type in ${JOB_TYPES};
  do
    for job_bs in ${JOB_BLOCKSIZES};
    do
      for job_mix in ${JOB_MIXES};
      do
    RESULTS_PREFIX=${PREFIX}_${job_file%.*}_${job_type}_${job_bs}_${job_mix}r_${TIMESTAMP}
    RESULTS_DIR=results/${RESULTS_PREFIX}
    RESULTS_FILE=${RESULTS_DIR}/${RESULTS_PREFIX}.log
    mkdir -p ${RESULTS_DIR}

    if [[ "$TEST_HOST" != "$NVME_HOST" ]]; then
      ssh ${TEST_HOST} "pkill nmon ; sleep $((WARMUP-STEP)) ; nmon -F /tmp/${TEST_HOST}.nmon
-s${STEP} -c${COUNT} -J -t" &
    fi
    ssh ${NVME_HOST} "pkill nmon ; sleep $((WARMUP-STEP)) ; nmon -F /tmp/${NVME_HOST}.nmon -s${STEP}
-c${COUNT} -J -t" &

    echo "Running FIO for $((WARMUP+RUNTIME)) seconds: ${RESULTS_PREFIX}"
    ssh ${TEST_HOST} "fio --runtime=${RUNTIME}s --ramp_time=${WARMUP}s --output-format=normal,terse
--output=/tmp/fio_${TEST_HOST}.log ${job_file} --rw=${job_type} --bs=${job_bs} --rwmixread=${job_mix}"
    echo "Saving results:"
    scp -p ${TEST_HOST}:/tmp/fio_${TEST_HOST}.log ${RESULTS_FILE}
    grep '3;fio-3.7;' ${RESULTS_FILE} > ${RESULTS_DIR}/${RESULTS_PREFIX}.csv
    grep -v '3;fio-3.7;' ${RESULTS_FILE} > ${RESULTS_DIR}/${RESULTS_PREFIX}.txt

    ssh ${NVME_HOST} "pkill nmon"
    if [[ "$TEST_HOST" != "$NVME_HOST" ]]; then
      ssh ${TEST_HOST} "pkill nmon"
      wait
      scp -p ${TEST_HOST}:/tmp/${TEST_HOST}.nmon ${RESULTS_DIR}/${RESULTS_PREFIX}_${TEST_HOST}.nmon
      ssh ${TEST_HOST} "ibv_devices" > ${RESULTS_DIR}/ibv_devices.txt
    fi
    wait
    scp -p ${NVME_HOST}:/tmp/${NVME_HOST}.nmon ${RESULTS_DIR}/${RESULTS_PREFIX}_${NVME_HOST}.nmon

    for nmonfile in `find ${RESULTS_DIR}/*.nmon`;
    do
      ./nmonchart $nmonfile
    done

    for file in ${GLOBAL_FILES} ${job_file}; do
      cp -pf ${file} ${RESULTS_DIR}/
    done
    cp -pf ${0} ${RESULTS_DIR}/

    echo -e "Complete: ${RESULTS_PREFIX}\n"
      done
    done
  done
done
exit
```

global.fio

```
ioengine=libaio
iodepth=32
iodepth_batch=16
iodepth_batch_complete_max=32
iodepth_batch_complete_min=1
direct=1
time_based=1
thread=1
gtod_reduce=0
disable_lat=0
refill_buffers
```

nvme-client_8xNVMe.fio

```
[global]
include global.fio
cpus_
allowed=1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63,
65,67,69,71,73,75,77,79,81,83,85,87,89,91,93,95
cpus_allowed_policy=split
numa_mem_policy=bind:1
numjobs=6
[job-nvme0n1]
filename=/dev/nvme0n1
[job-nvme1n1]
filename=/dev/nvme1n1
[job-nvme2n1]
filename=/dev/nvme2n1
[job-nvme3n1]
filename=/dev/nvme3n1
[job-nvme4n1]
filename=/dev/nvme4n1
[job-nvme5n1]
filename=/dev/nvme5n1
[job-nvme6n1]
filename=/dev/nvme6n1
[job-nvme7n1]
filename=/dev/nvme7n1
```

nvme-client_6xNVMe.fio

```
[global]
include global.fio
cpus_
allowed=1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63,
65,67,69,71,73,75,77,79,81,83,85,87,89,91,93,95
cpus_allowed_policy=split
numa_mem_policy=bind:1
numjobs=8
[job-nvme0n1]
filename=/dev/nvme0n1
[job-nvme1n1]
filename=/dev/nvme1n1
[job-nvme2n1]
filename=/dev/nvme2n1
[job-nvme4n1]
filename=/dev/nvme4n1
[job-nvme5n1]
filename=/dev/nvme5n1
[job-nvme6n1]
filename=/dev/nvme6n1
```

nvme-client_4xNVMe.fio

```
[global]
include global.fio
cpus_
allowed=1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63,
65,67,69,71,73,75,77,79,81,83,85,87,89,91,93,95
cpus_allowed_policy=split
numa_mem_policy=bind:1
numjobs=12
[job-nvme0n1]
filename=/dev/nvme0n1
[job-nvme1n1]
filename=/dev/nvme1n1
[job-nvme4n1]
filename=/dev/nvme4n1
[job-nvme5n1]
filename=/dev/nvme5n1
```

nvme-server_8xNVMe.fio

```
[global]
include global.fio
cpus_
allowed=0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62
cpus_allowed_policy=split
numa_mem_policy=bind:0
numjobs=6
[job-nvme0n1]
filename=/dev/nvme0n1
[job-nvme1n1]
filename=/dev/nvme1n1
[job-nvme2n1]
filename=/dev/nvme2n1
[job-nvme3n1]
filename=/dev/nvme3n1
[job-nvme4n1]
filename=/dev/nvme4n1
[job-nvme5n1]
filename=/dev/nvme5n1
[job-nvme6n1]
filename=/dev/nvme6n1
[job-nvme7n1]
filename=/dev/nvme7n1
```

nvme-server_6xNVMe.fio

```
[global]
include global.fio
cpus_
allowed=0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62
cpus_allowed_policy=split
numa_mem_policy=bind:0
numjobs=8
[job-nvme0n1]
filename=/dev/nvme0n1
[job-nvme1n1]
filename=/dev/nvme1n1
[job-nvme2n1]
filename=/dev/nvme2n1
[job-nvme4n1]
filename=/dev/nvme4n1
[job-nvme5n1]
filename=/dev/nvme5n1
[job-nvme6n1]
filename=/dev/nvme6n1
```

nvme-server_4xNVMe.fio

```
[global]
include global.fio
cpus_
allowed=0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62
cpus_allowed_policy=split
numa_mem_policy=bind:0
numjobs=12
[job-nvme0n1]
filename=/dev/nvme0n1
[job-nvme1n1]
filename=/dev/nvme1n1
[job-nvme4n1]
filename=/dev/nvme4n1
[job-nvme5n1]
filename=/dev/nvme5n1
```

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

This project was commissioned by Dell EMC.

Principled Technologies®

Facts matter.®