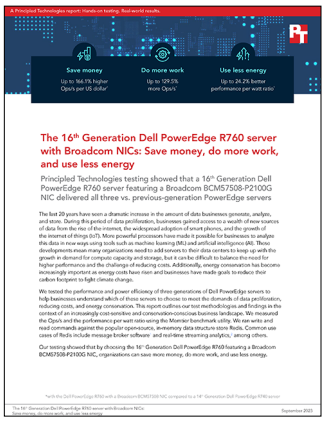

The science behind the report:

The 16th Generation Dell PowerEdge R760 server with Broadcom NICs: Save money, do more work, and use less energy

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 The 16th Generation Dell PowerEdge R760 server with Broadcom NICs: Save money, do more work, and use less energy.

We concluded our hands-on testing on June 20, 2023. 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 June 15, 2023 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 http://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 testing.

	Dell PowerEdge R760	Dell PowerEdge R750	Dell PowerEdge R740
Operations per second – Higher is better	64,282,525	49,215,573	28,000,329
Throughput (MB/s) – Higher is better	2,533	1,936	1,116
Performance per watt (Ops/s per watt) – Higher is better	77,265	70,698	62,179

System configuration information

Table 2: Detailed information on the systems we tested.

System configuration information	Dell R760	Dell R750	Dell R740
BIOS name and version			
Non-default BIOS settings	1.3.2	1.9.2	2.18.1
Operating system name and version/build number	Red Hat Enterprise Linux release 8.7	Red Hat Enterprise Linux release 8.7	Red Hat Enterprise Linux release 8.7
Date of last OS updates/patches applied	5/30/23	5/30/23	5/30/23
Power management policy	Performance	Performance	Performance
Processor			
Number of processors	2	2	2
Vendor and model	Intel Xeon Gold 6430 Model 143	Intel Xeon Gold 6330 Model 106	Intel Xeon Gold 6230 Model 85
Core count (per processor)	32	28	20
Core frequency (GHz)	2.10	2.00	2.10
Stepping	8	6	7
Memory module(s)			
Total memory in system (GB)	1024	1024	1024
Number of memory modules	16	32	16
Vendor and model	Hynix HMCG94AEBRA109N	Hynix HMAA4GR7AJR8N-XN	Hynix HMAA8GR7AJR4N
Size (GB)	64	32	64
Type	DRAM DDR5	DRAM DDR4	DRAM DDR4
Speed (MHz)	4800	3200	3200
Speed running in the server (MHz)	4400	2933	2933
Storage controller (A)			
Vendor and model	BOSS-N1	Dell PERC H755 Front	Dell PERC H740P
Cache size (GB)	0	8192 MB	8192 MB
Firmware version	2.1.13.2014	52.21.0-4606	51.16.0-4076
Storage controller (B)			
Vendor and model	PERC H965i Front	–	–
Cache size (GB)	8361 MB	–	–
Firmware version	8.0.0.0.18-84	–	–
Driver version	8.0.0.69.0	–	–

System configuration information	Dell R760	Dell R750	Dell R740
Local storage (type A)			
Number of drives	2	2	2
Drive vendor and model	Dell EC NVMe ISE 7400 RI M.2	Toshiba THNSF8120CCSE	Toshiba THNSF8120CCSE
Drive size (GB)	480	120	120
Drive information (speed, interface, type)	8 GT/s PCIe, NVMe	6Gb SATA, SSD	6Gb SATA, SSD
Network adapter			
Vendor and model	Broadcom BCM57508-P2100G	Broadcom BCM57414	Broadcom BCM57712
Number and type of ports	2 x 100G QSFP	Ethernet2 x 25Gb DA QSFP+	2 x 10Gb DA SFP+
Driver version	21.85.21.91 (family version)	bnxt_en 1.10.2-223.0.183.0	bnx2x 1.715.20
Cooling fans			
Fan type	Silver	Standard	Standard
Number of cooling fans	6	6	6
Power supplies			
Vendor and model	Dell 07DWXYA01	Dell 0CYHHJA01	Dell 0Y26KXA02
Number of power supplies	2	2	2
Wattage of each (W)	1400	1400	1100

How we tested

Installing Red Hat Enterprise Linux 8.7

Install the operating system on both the system under test and client system

1. Open a browser window, and connect to the iDRAC.
2. Log into the iDRAC.
3. Click Virtual Console.
4. Click Connect Virtual Media.
5. Next to Map CD/DVD, click Browse.
6. Browse to the ISO for Red Hat 8.7.
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 8.7.0, 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 internal JBOSS 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 r750-server or r750-client, and click Apply.
22. At the top-right of the screen, next to Ethernet information, click the OFF slider to ON, allow the NIC to 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.
26. Wait for the system to reboot.

Configuring Red Hat Enterprise Linux 8.7 for the system under test

Run the following commands on both the system under test and client system.

27. Set the time zone, disable SELinux, disable the firewall, and set the tuned profile:

```
sudo timedatectl set-timezone America/New_York
setenforce 0
sed -i 's/SELINUX=.*/SELINUX=disabled/' /etc/selinux/config
systemctl disable --now firewalld
tuned-adm profile throughput-performance
```

28. Register the system with RHN, and configure subscriptions:

```
subscription-manager register
subscription-manager service-level --set="Self-Support"
subscription-manager usage --set="Development/Test"
subscription-manager role --set="Red Hat Enterprise Linux Server"
subscription-manager attach
subscription-manager release --set=8.7
```

29. Install updates, and reboot if new kernel is installed:

```
dnf update -y
reboot
```

30. Install EPEL and the prerequisites for iPerf3 and Redis:

```
subscription-manager repos --enable codeready-builder-for-rhel-8-$(arch)-rpms
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
dnf install -y wget tar smartmontools vim sysstat numactl pcp-system-tools nvme-cli
net-tools usbutils
dnf groupinstall "Development Tools" -y
dnf install autoconf automake make gcc-c++ pcre-devel zlib-devel libevent-devel openssl-devel wget
```

31. Install the iPerf3 package from RHN:

```
dnf install -y iperf3
```

32. Make the following tuning changes to the OS for Redis:

- Add the following lines to the end of /etc/sysctl.conf to allow larger Redis deployments to work:

```
vm.overcommit_memory=1
net.core.somaxconn=65535
```

- Add the following lines to the end of /etc/security/limits.conf:

```
*   soft  nofile         10240
*   hard  nofile         10240
```

33. Download the set_irq_affinity script, make it executable, and move it into the bin folder:

```
wget https://raw.githubusercontent.com/majek/ixgbe/master/scripts/set_irq_affinity
chmod +x set_irq_affinity
mv set_irq_affinity /usr/local/sbin/
```

34. On the 10Gb, 25Gb, and 100Gb NICs, assign a static IP address and set the MTU to 9000.
35. Modify /etc/ssh/sshd_config and change the following lines:

```
MaxSessions 100
MaxStartups 100:30:100
```

Preparing for a Redis deployment

Run the following commands only on the system under test.

36. Download and untar Redis:

```
wget https://download.redis.io/redis-stable.tar.gz
tar -xzvf redis-stable.tar.gz
```

37. Compile and install Redis:

```
cd redis-stable
make
make install
```

38. Format the database drive for Redis:

```
mkfs.xfs /dev/nvme1n1
```

39. Create the directories you will use for Redis:

```
mkdir -p /var/lib/redis
mkdir -p /etc/redis/cluster
mkdir -p /var/log/redis
```

40. Mount the drive to the database folder:

```
mount /dev/nvme1n1 /var/lib/redis
```

41. Create a redis user and add it to the redis group:

```
useradd -m redis -p Password1
usermod -a -G redis redis
```

42. Change the ownership of the Redis directories to the redis user:

```
chown -R redis:redis /var/lib/redis
chown -R redis:redis /etc/redis/cluster
chown -R redis:redis /var/log/redis
```

43. Change to the redis user for the rest of the configuration:

```
sudo su - redis
```

44. Create a script to create folders and configuration files for the Redis nodes (72, 104, and 120 nodes on the R740, R750, and R760 respectively):

```
rm -rf /var/lib/redis/7*
rm -rf /etc/redis/cluster/7*

NUM_REDIS=${1}
MAX_REDIS=$((NUM_REDIS+7000))
for REDIS_NUM in `seq 7001 $MAX_REDIS`;
do
        mkdir -p /var/lib/redis/${REDIS_NUM}
        mkdir -p /etc/redis/cluster/${REDIS_NUM}
        cat <<EOF >/etc/redis/cluster/${REDIS_NUM}/redis_${REDIS_NUM}.conf
port ${REDIS_NUM}
dir /var/lib/redis/${REDIS_NUM}/
appendonly no
protected-mode no
cluster-enabled yes
cluster-node-timeout 5000
cluster-config-file /etc/redis/cluster/${REDIS_NUM}/nodes_${REDIS_NUM}.conf
pidfile /var/run/redis/redis_${REDIS_NUM}.pid
logfile /var/log/redis/redis_${REDIS_NUM}.log
loglevel notice
bind 0.0.0.0
EOF
done
```

45. Create a script to kick off Redis and make it executable:

```
cat <<EOF >/home/redis/multi_redis.sh
#!/bin/bash
```

```
NUM_REDIS=${1}
MAX_REDIS=$((NUM_REDIS+7000))
for REDIS_NUM in `seq 7001 $MAX_REDIS`;
do
    COREBIND=$((REDIS_NUM-7001))
    numactl --physcpubind=${COREBIND} --localalloc redis-server /etc/redis/cluster/${REDIS_NUM}/
redis_${REDIS_NUM}.conf &
done
EOF
chmod +x multi_redis.sh
```

Deploying a Redis cluster

46. Log into the system under test via SSH.
47. Mount the database drive:

```
mount /dev/nvme1n1 /var/lib/redis
```

48. Change to the redis user:

```
sudo su - redis
```

49. Launch the appropriate number of Redis instances (we used 72 for the R740, 96 for the R750, and 120 for the R760):

```
./multi_redis.sh [72, 96, or 120 depending on the server]
```

50. Create multiple 24-node clusters with Redis (3 on the R740, 4 on the R750, and 5 on the R760). The following is an example of how to create a single 24-node cluster:

```
redis-cli --cluster create 172.16.0.31:{7001..7072}From this point on, whenever you start your Redis
cluster with the multi_redis.sh script, it will automatically restart the cluster.
```

Installing Memtier on the client

51. Log into the client host via SSH.
52. Install the prerequisites for Memtier:

```
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
dnf install autoconf automake make gcc-c++ pcre-devel zlib-devel libevent-devel openssl-devel -y
```

53. Pull Memtier from Github:

```
git clone https://github.com/RedisLabs/memtier_benchmark.git
```

54. Compile and install Memtier:

```
cd memtier_benchmark/
autoreconf -ivf
./configure --disable-tls
sudo make
sudo make install
```

Running a memtier_benchmark test

55. Log into the client instance via SSH as ec2-user.
56. Using Memtier, create the dataset Redis will run against:

```
memtier_benchmark -s [SUT IP address] -p 7001 -t 16 -c 1 --key-maximum=240000000 -n allkeys -d 25
--pipeline=15 --key-pattern=P:P --ratio=1:0 --cluster-mode
```

57. Once the dataset has been created, Redis needs a few minutes to finish writing the changes to disk. Once it has finished, you can run a test with the following command:

```
memtier_benchmark -s [SUT IP address] -p 7000 -t 10 -c 1 --key-maximum=240000000 -d 25 --randomize
--test-time=600 --pipeline=30 --ratio=0:1 --cluster-mode
```

Preparing scripts and running iPerf tests

Prepare FIO and bash script files to automate testing and gathering results on both systems. We modified a number of parameters (e.g., buffer length, window size).

58. Create the following bash scripts and run as instructed:

- On the client, create and run the following script after every boot to tune the network interface: tune_client.sh

```
#!/bin/bash
systemctl stop irqbalance ; sleep 3 ; pgrep irqbalance ;

for IFNAME in ens6f0np0 ;
do
  ethtool -L ${IFNAME} combined 56
  sleep 1
  set_irq_affinity local ${IFNAME}
done
On the SUT, create and run the following script after every boot to tune the network
interfaces: tune_server.sh
#!/bin/bash
systemctl stop irqbalance ; sleep 3 ; pgrep irqbalance ;

for IFNAME in ens6f0np0 ens4f0 ens4f1 ens5f0 ens5f1;
do
  ethtool -L ${IFNAME} combined 56
  sleep 1
  set_irq_affinity local ${IFNAME}
done
```

- On the client, create the following script: run_test.sh

```
#!/bin/bash

APP=iperf3
BUFFER_LENGTH=128k
TCP_WINDOW=512k
NIC_SPEED=100
TARGET_IP=192.168.${NIC_SPEED}.1
PORT_BASE=5200
SERVER_HOST=r750-server
CLIENT_HOST=r750-client
PROCESSES=1
INSTANCES=$1
RUNS=15
WARMUP=3
RUNTIME=20
PAUSE=3
RAMPDELAY=1
CONGESTION=cubic
```

```
STEP=1
TIMESTAMP=$(date '+%Y%m%d_%H%M%S')
SERVER_CPU_START=1
SERVER_CPU_SKIP=2
CLIENT_CPU_START=1
CLIENT_CPU_SKIP=2

TOTAL_TIME=$((WARMUP+RUNTIME))

# Prepare nmon on client and server
#sudo killall -q -w nmon ; sudo sync ; sudo rm -f /tmp/client.nmon
#ssh ${TEST_HOST} "sudo killall -q -w nmon ; sudo sync ; sudo rm -f /tmp/server.nmon"

# Start nmon on client and server and wait 1 step
#sudo nmon -F /tmp/client.nmon -s${STEP} -c$((SAMPLES_TOTAL)) -J -t
#ssh ${TEST_HOST} "sudo nmon -F /tmp/server.nmon -s${STEP} -c$((SAMPLES_TOTAL)) -J -t"
#sleep ${STEP}

for PROC in ${PROCESSES};
do
  # Make results folder
  RESULTS_DIR=results/${APP}_${INSTANCES}I_${PROC}P_${RUNS}RC_${NIC_SPEED}G_${TIMESTAMP}
  mkdir -p ${RESULTS_DIR}
  RESULTS_FINAL=${RESULTS_DIR}/${APP}_${INSTANCES}I_${PROC}P_${RUNS}RC_${NIC_SPEED}G_${TIMESTAMP}.csv
  echo "RESULTS_DIR: ${RESULTS_DIR}"
  echo
  for RUN in `seq -w 1 ${RUNS}`;
  do
    echo "RUN: ${RUN}"
    ssh ${SERVER_HOST} "killall -q -w ${APP}"
    ssh ${CLIENT_HOST} "killall -q -w ${APP}"
    sleep ${PAUSE}
    for INSTANCE in `seq -w 1 ${INSTANCES}`;
    do
      SERVER_CPU_PIN=`expr \( \( ${INSTANCE} - 1 \) \* ${SERVER_CPU_SKIP} \) + ${SERVER_CPU_START}`
      CLIENT_CPU_PIN=`expr \( \( ${INSTANCE} - 1 \) \* ${CLIENT_CPU_SKIP} \) + ${CLIENT_CPU_START}`
      SERVER_PORT=`expr ${PORT_BASE} + ${INSTANCE}`
      SERVER_FILE=${RESULTS_DIR}/${APP}_server_R${RUN}_I${INSTANCE}
      CLIENT_FILE=${RESULTS_DIR}/${APP}_client_R${RUN}_I${INSTANCE}
      echo "INSTANCE: ${INSTANCE}"
      TOTAL_TIME=${RUNTIME}
      echo "TOTAL_TIME: ${TOTAL_TIME}"
      OMIT_TIME=`expr ${WARMUP} + \( ${INSTANCES} \* ${RAMPDELAY} \) - \( ${INSTANCE} \*
${RAMPDELAY} \)`
      echo "OMIT_TIME: ${OMIT_TIME}"
      ssh ${SERVER_HOST} "numactl -C ${SERVER_CPU_PIN} -l ${APP} --server --one-off --port ${SERVER_
PORT}" > ${SERVER_FILE}.txt &
      ssh ${CLIENT_HOST} "sleep `expr ${PAUSE} + \( ${INSTANCE} \* ${RAMPDELAY} \)` ; numactl -C
${CLIENT_CPU_PIN} -l ${APP} --forceflush --format g --client ${TARGET_IP} --port ${SERVER_PORT} --time
${TOTAL_TIME} --omit ${OMIT_TIME} --length ${BUFFER_LENGTH} --window ${TCP_WINDOW} --parallel ${PROC}
--congestion ${CONGESTION}" | tee ${CLIENT_FILE}.txt &
    done
    wait
    sync
    echo -n "Combined average throughput (Gb/s): "
    awk '/receiver/{sum+=$7}END{print sum}' ${RESULTS_DIR}/${APP}_client_R${RUN}_I*.txt | tee -a
${RESULTS_FINAL}
    awk '/receiver/{print $7}' ${RESULTS_DIR}/${APP}_client_R${RUN}_I*.txt > ${RESULTS_DIR}/${APP}_
client_R${RUN}.csv
    echo
  done
  echo "Final results:"
  paste ${RESULTS_DIR}/${APP}_client_R*.csv | tee ${RESULTS_DIR}/${APP}_client.csv
  echo
  echo "Final combined results:"
  cat ${RESULTS_FINAL}
done
echo

# Save script, timestamp, and environment variables to results directory
cp -pvf ${0} ${RESULTS_DIR}/
echo ${TIMESTAMP} > ${RESULTS_DIR}/timestamp.txt
```

```
set > ${RESULTS_DIR}/set.txt
echo
Modify the variables at the top of the run_test.sh script as needed. To start the test, execute
the script using the following command and by sure to include the first parameter to set the
number of instances:
./run_test.sh <NUMBER_OF_INSTANCES>
```

Read the report at https://facts.pt/2SXLepY ▶

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®