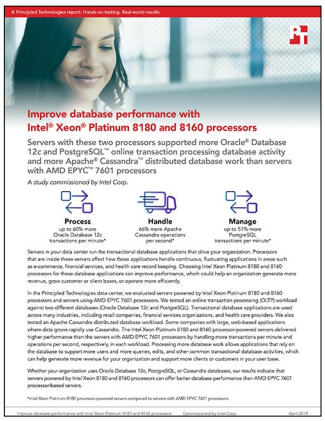

The science behind the report:

Improve database performance with Intel® Xeon® Platinum 8180 and 8160 processors

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 "Improve database performance with Intel® Xeon® Platinum 8180 and 8160 processors."

We concluded our hands-on testing on December 14, 2018. 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 December 10, 2018 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

Oracle® Database 12c workload

Oracle licensing agreements prohibit publishing benchmark scores, so we report only relative performance for the two relational database management systems. To get the relative performance numbers, we designated the lowest performance output as the baseline (the servers with AMD EPYC™ 7601 processors for this study), assigned the baseline output a value of 1, divided the outputs of the Intel Xeon Platinum 8180 and 8160 processor-powered servers by the baseline output, and assigned the results of the division to the respective configurations.

Processor	Relative transactions per minute
Intel Xeon Platinum 8180	1.60
AMD EPYC 7601	1
Percent difference for Intel Xeon Platinum 8180	60%
Intel Xeon Platinum 8160	1.39
AMD EPYC 7601	1
Percent difference for Intel Xeon Platinum 8160	39%

PostgreSQL 9.6 workload

Processor	Transactions per minute
Intel Xeon Platinum 8180	1,757,375
AMD EPYC 7601	1,156,705
Percent difference for Intel Xeon Platinum 8180	51%
Intel Xeon Platinum 8160	1,471,016
AMD EPYC 7601	1,156,705
Percent difference for Intel Xeon Platinum 8160	27%

Apache® Cassandra™ 3.11.3 workload

Processor	Operations per second
Intel Xeon Platinum 8180	377,464
AMD EPYC 7601 (20 VMs)	226,099
AMD EPYC 7601 (24 VMs)	219,228
Percent difference for Intel Xeon Platinum 8180 vs AMD EPYC 7601 (24 VMs)	72%
Percent difference for Intel Xeon Platinum 8180 vs AMD EPYC 7601 (20 VMs)	66%
Intel Xeon Platinum 8160	320,654
AMD EPYC 7601 (20 VMs)	226,099
AMD EPYC 7601 (24 VMs)	219,228
Percent difference for Intel Xeon Platinum 8160 vs AMD EPYC 7601 (24 VMs)	46%
Percent difference for Intel Xeon Platinum 8160 vs AMD EPYC 7601 (20 VMs)	41%

System configuration information

Why we chose our configurations

We focused our configuration choices for these tests on the CPUs. We chose the Intel Xeon Platinum 8180 processors to compare to the AMD EPYC 7601 processors because they are top bin offerings from the two processor companies. We chose to test the Intel Xeon Platinum 8160 processors because they are a mid-range tier option from the Platinum processor family that offers a high number of CPU cores.

Per the tables below, you can see that to keep the memory configurations fair, we populated each system with one DIMM per CPU channel to optimize both memory bandwidth and capacity. As the Intel Xeon Platinum 8180 and 8160 processors had six channels per CPU and the AMD EPYC 7601 processors had eight channels per CPU, each Intel Xeon Platinum 8180 and 8160 processor-powered system had 384 GB of RAM and each AMD EPYC 7601 processor-based system had 512 GB of RAM. Finally, we chose a database size that would fit in memory and stored the data and logs on NVMe SSDs to help reduce the potential for storage bottlenecks.

The tables below present detailed information on the systems we tested.

Hammer DB using Oracle Database 12*c* testing

For the memory of the servers, we used dual-rank RDIMMs and populated one DIMM per channel in a two DIMM per channel box. AMD best practices for a general performance and capacity configuration of the AMD EPYC 7601 processors recommends 2,666 RDIMMs operate at 2,400 MHz.[1]

Server configuration information	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10	HPE ProLiant DL385 Gen10
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	11/20/2018	11/28/2018	11/20/2018
BIOS name and version	U30 v1.42 (06/20/2018)	U30 v1.42 (06/20/2018)	A40 v1.30 (06/07/2018)
Non-default BIOS settings	N/A	N/A	Power deterministic
Operating system name and version/build number	Oracle Enterprise Linux® 7.5 4.1.12-124.20.7.el7uek.x86_64	Oracle Enterprise Linux 7.5 4.1.12-124.20.7.el7uek.x86_64	Oracle Enterprise Linux 7.5 4.1.12-124.20.7.el7uek.x86_64
Date of last OS updates/ patches applied	11/04/2018	11/04/2018	11/04/2018
Power management policy	Static high-performance mode	Static high-performance mode	Static high-performance mode
No. of nodes	1	1	1
Workload			
Name and version	HammerDB 3.0	HammerDB 3.0	HammerDB 3.0
Other software	Oracle Database 12*c* Release 2	Oracle Database 12*c* Release 2	Oracle Database 12*c* Release 2
Processor			
Number of processors	2	2	2
Vendor and model	Intel Xeon Platinum 8180	Intel Xeon Platinum 8160	AMD EPYC 7601
Ucode	0x200004d	0x200004d	0x8001227
Core count (per processor)	28	24	32
Thread count (per processor)	56	48	64
Core frequency (GHz)	2.5	2.1	2.2
Stepping	4	0	2
Hyperthreading	On	On	On
Turbo	On	On	On
Mitigation variants	1, 2, 3, 3a, 4, L1tf	1, 2, 3, 3a, 4, L1tf	1, 2, 3, 3a, 4, L1tf

Server configuration information	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10	HPE ProLiant DL385 Gen10
Memory module(s)			
Total memory in system (GB)	384	384	512
Number of memory modules	12	12	16
Vendor and model	HPE SmartMemory 840758-191	HPE SmartMemory 840758-191	HPE SmartMemory 840758-191
Size (GB)	32	32	32
Type	DDR4	DDR4	DDR4
Speed (MHz)	2,666	2,666	2,400
Speed running in the server (MHz)	2,666	2,666	2,400
Storage controller			
Vendor and model	HPE Smart Array P408i-a SR Gen10	HPE Smart Array P408i-a SR Gen10	HPE Smart Array P408i-a SR Gen10
Cache size (GB)	2	2	2
Firmware version	1.65	1.65	1.65
Driver version	1.1.2-126	1.1.2-126	1.1.2-126
Local storage (OS & ASM Instance)			
Number of drives	6	6	6
Drive vendor and model	Intel SSD 530	Intel SSD 530	Intel SSD 530
Drive size (GB)	480	480	480
Drive information (speed, interface, type)	6 Gbps, SATA, SSD	6 Gbps, SATA, SSD	6 Gbps, SATA, SSD
Local storage (Data)			
Number of drives	2	2	2
Drive vendor and model	Intel SSD DC P3700	Intel SSD DC P3700	Intel SSD DC P3700
Drive size (GB)	800	800	800
Drive information (speed, interface, type)	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD
Local storage (Logs)			
Number of drives	2	2	2
Drive vendor and model	Intel SSD DC P3700	Intel SSD DC P3700	Intel SSD DC P3700
Drive size (GB)	2,000	2,000	2,000
Drive information (speed, interface, type)	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD
Network adapter			
Vendor and model	Intel 82599ES	Intel 82599ES	Intel 82599ES
Number and type of ports	2 x 10GbE	2 x 10GbE	2 x 10GbE
Driver version	5.1.0	5.1.0	5.1.0
Cooling fans			
Vendor and model	Delta Electronics PFR0612XHE	Delta Electronics PFR0612XHE	Delta Electronics PFM0612XHE
Number of cooling fans	6	6	6

Server configuration information	HPE ProLiant DL380 Gen10	HPE ProLiant DL380 Gen10	HPE ProLiant DL385 Gen10
Power supplies			
Vendor and model	HPE 865414-B21	HPE 865414-B21	HPE 865414-B21
Number of power supplies	2	2	2
Wattage of each (W)	800	800	800

HammerDB using PostgreSQL 9.6 testing

Server configuration information	Intel Xeon 8180 processor-powered white box server	Intel Xeon 8160 processor-powered white box server	AMD EPYC 7601 processor-based server
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	11/20/2018	11/28/2018	12/04/2018
Platform	Intel S2600WFD	Intel S2600WFD	Supermicro H11DSU-iN
BIOS name and version	Intel SE5C620.86B.0D.01.0134.10042181737	Intel SE5C620.86B.0D.01.0134.100420181737	Supermicro H11DSU-iN 1.1b
Non-default BIOS settings	N/A	N/A	Memory Clock set to 2666 MHz Power Deterministic
Operating system name and version/build number	Red Hat® Enterprise Linux® 7.5 3.10.0-862.11.6.el7.x86_64	Red Hat Enterprise Linux 7.5 3.10.0-862.11.6.el7.x86_64	Red Hat Enterprise Linux 7.5 3.10.0-862.11.6.el7.x86_64
Date of last OS updates/ patches applied	9/24/2018	9/24/2018	9/24/2018
Power management policy	Performance	Performance	Performance
No. of nodes	1	1	1
Workload			
Name and version	HammerDB 3.0	HammerDB 3.0	HammerDB 3.0
Other software	PostgreSQL 9.6	PostgreSQL 9.6	PostgreSQL 9.6
Processor			
Number of processors	2	2	2
Vendor and model	Intel Xeon Platinum 8180	Intel Xeon Platinum 8160	AMD EPYC 7601
Ucode	0x2000050	0x2000050	0x8001227
Core count (per processor)	28	24	32
Thread count (per processor)	56	48	64
Core frequency (GHz)	2.5	2.1	2.2
Stepping	4	0	2
Hyperthreading	On	On	On
Turbo	On	On	On
Mitigation variants	1, 2, 3, 3a, 4, L1tf	1, 2, 3, 3a, 4, L1tf	1, 2, 3, 3a, 4, L1tf
Memory module(s)			
Total memory in system (GB)	384	384	512
Number of memory modules	12	12	16
Vendor and model	Micron 36ASF4G72PZ-2G6B2	Micron 36ASF4G72PZ-2G6B2	Micron 36ASF4G72PZ-2G6D1
Size (GB)	32	32	32
Type	DDR4	DDR4	DDR4
Speed (MHz)	2,666	2,666	2,666
Speed running in the server (MHz)	2,666	2,666	2,666

Server configuration information	Intel Xeon 8180 processor-powered white box server	Intel Xeon 8160 processor-powered white box server	AMD EPYC 7601 processor-based server
Local storage (OS)			
Number of drives	1	1	1
Drive vendor and model	Intel SSD DC S3710	Intel SSD DC S3710	Intel SSD DC S3710
Drive size (GB)	400	400	400
Drive information (speed, interface, type)	6Gbps, SATA, SSD	6Gbps, SATA, SSD	6Gbps, SATA, SSD
Local storage (Data)			
Number of drives	2	2	2
Drive vendor and model	Intel SSD P4600	Intel SSD P4600	Intel SSD P4600
Drive size (GB)	2,000	2,000	2,000
Drive information (speed, interface, type)	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD
Local storage (Logs)			
Number of drives	2	2	2
Drive vendor and model	Intel SSD P4600	Intel SSD P4600	Intel SSD P4600
Drive size (GB)	2,000	2,000	2,000
Drive information (speed, interface, type)	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD
Network adapter			
Vendor and model	Intel 82599ES	Intel 82599ES	Intel 82599ES
Number and type of ports	2 x 10GbE	2 x 10GbE	2 x 10GbE
Driver version	5.1.0	5.1.0	5.1.0
Cooling fans			
Vendor and model	Nidec UltraFlo Bb60E12BS1B5-07A016	Nidec UltraFlo Bb60E12BS1B5-07A016	Nidec UltraFlo V80E12BGA5-57
Number of cooling fans	6	6	4
Power supplies			
Vendor and model	FLEXTRONICS G84027-009	FLEXTRONICS G84027-009	SUPERMICRO PWS-1K62A-1R
Number of power supplies	2	2	2
Wattage of each (W)	1,100	1,100	1,600

Cassandra 3.11.3 testing

Server configuration information	Intel Xeon 8180 processor-powered white box server	Intel Xeon 8160 processor-powered white box server	AMD EPYC 7601 processor-based server
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	12/14/2018	12/14/2018	12/14/2018
Platform	Intel S2600WFD	Intel S2600WFD	Supermicro H11DSU-iN
BIOS name and version	Intel SE5C620.86B. 0D.01.0134.100420181737	Intel SE5C620.86B. 0D.01.0134.100420181737	Supermicro H11DSU-iN 1.1b
Non-default BIOS settings	Virtualization enabled	Virtualization enabled	Memory Clock set to 2666 MHz Power Deterministic
Operating system name and version/build number	Ubuntu Server 18.04.1 4.15.0-42-generic	Ubuntu Server 18.04.1 4.15.0-42-generic	Ubuntu Server 18.04.1 4.15.0-42-generic
Date of last OS updates/ patches applied	12/10/2018	12/10/2018	12/10/2018
Power management policy	Performance	Performance	Performance
No. of nodes	2	2	2
Workload			
Name and version	Cassandra 3.11.3	Cassandra 3.11.3	Cassandra 3.11.3
Processor			
Number of processors	2	2	2
Vendor and model	Intel Xeon Platinum 8180	Intel Xeon Platinum 8160	AMD EPYC 7601
Ucode	0x2000050	0x2000050	0x8001227
Core count (per processor)	28	24	32
Thread count (per processor)	56	48	64
Core frequency (GHz)	2.5	2.1	2.2
Stepping	4	0	2
Hyperthreading	On	On	On
Turbo	On	On	On
Mitigation Variants	1, 2, 3, 3a, 4, L1tf	1, 2, 3, 3a, 4, L1tf	1, 2, 3, 3a, 4, L1tf
Memory module(s)			
Total memory in system (GB)	384	384	512
Number of memory modules	12	12	16
Vendor and model	Micron 36ASF4G72PZ-2G6B2	Micron 36ASF4G72PZ-2G6B2	Micron 36ASF4G72PZ-2G6D1
Size (GB)	32	32	32
Type	DDR4	DDR4	DDR4
Speed (MHz)	2,666	2,666	2,666
Speed running in the server (MHz)	2,666	2,666	2,666

Server configuration information	Intel Xeon 8180 processor-powered white box server	Intel Xeon 8160 processor-powered white box server	AMD EPYC 7601 processor-based server
Local storage (type A)			
Number of drives	1	1	1
Drive vendor and model	Intel SSD DC S3710	Intel SSD DC S3710	Intel SSD DC S3710
Drive size (GB)	400	400	400
Drive information (speed, interface, type)	6Gbps, SATA, SSD	6Gbps, SATA, SSD	6Gbps, SATA, SSD
Local storage (type B)			
Number of drives	2	2	2
Drive vendor and model	Intel SSD P4600	Intel SSD P4600	Intel SSD P4600
Drive size (GB)	2,000	2,000	2,000
Drive information (speed, interface, type)	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD	PCIe NVMe 3.0, PCIe, SSD
Network adapter			
Vendor and model	Intel 82599ES	Intel 82599ES	Intel 82599ES
Number and type of ports	2 x 10GbE	2 x 10GbE	2 x 10GbE
Driver version	5.1.0	5.1.0	5.1.0
Cooling fans			
Vendor and model	Nidec UltraFlo Bb60E12BS1B5-07A016	Nidec UltraFlo Bb60E12BS1B5-07A016	Nidec UltraFlo V80E12BGA5-57
Number of cooling fans	6	6	4
Power supplies			
Vendor and model	FLEXTRONICS G84027-009	FLEXTRONICS G84027-009	SUPERMICRO PWS-1K62A-1R
Number of power supplies	2	2	2
Wattage of each (W)	1,100	1,100	1,600

How we tested the HammerDB workload for Oracle

We tested two servers with three processor configurations. We equipped the first server, an HPE ProLiant DL385 Gen10, with AMD EPYC 7601 processors. We equipped the second server, an HPE ProLiant DL380 Gen10, with Intel Xeon Platinum 8180 processors first and then re-tested with Intel Xeon Platinum 8160 processors. We populated the RAM in each server so each CPU channel would have one memory stick, which gave the AMD EPYC 7601 processor-based solution 512 GB of RAM and the Intel Xeon Platinum 8180 and 8160 solutions 384 GB. For each configuration, we configured two SSDs in a RAID1 pair and installed the operating system. For the Automatic Storage Management (ASM) instance and the Fast Recovery Area where our database backups were held, we used an additional four SSDs in a RAID10 pair. We installed the grid infrastructure and database software on the OS drives. Using the NVMe SSDs in the system, we created ASM disks and put them into ASM disk groups, one for data and one for logs.

We generated an 800-warehouse database with 220 users running on our client VMs. We set a 15-minute warm-up for the test, which had a 30-minute runtime. We backed up and restored the database with Recovery Manager (RMAN). We ran the test three times and chose the median runs for our calculations.

Installing Oracle Enterprise Linux 7.5

1. Insert an Oracle Enterprise Linux 7.5 bootable USB stick to the server, and boot to it.
2. Select Install or upgrade an existing system.
3. Choose the language you wish to use, and click Continue.
4. Select Installation Destination.
5. For the OS, select the desired disk.
6. Under Other Storage Options, select I will configure partitioning.
7. Click Done.
8. Select Click here to create them automatically.
9. Remove the /home partition.
10. Expand the swap partition to 16GB.
11. Assign all remaining free space to the / partition.
12. Click Done.
13. Click Accept Changes.
14. Select Kdump.
15. Uncheck Enable kdump, and click Done.
16. Select Network & Hostname.
17. Enter the desired hostname for the VM.
18. Turn on the desired network port, and click Configure.
19. On the General tab, select Automatically connect to this network when it is available.
20. Under Method on the IPv4 Settings tab, select Manual.
21. Under Addresses, click Add, and enter the desired static IP information for the server.
22. Enter the desired DNS information.
23. Click Save, and click Done.
24. Select Date & Time, and ensure the correct date, time, and time zone are set.
25. To add an NTP server, click the cog next to the Network Time On/Off switch.
26. Add the IP address of your NTP server, and click +.
27. Uncheck all other NTP servers.
28. Click OK.
29. Click Done.
30. Click Begin Installation.
31. Select Root Password.
32. Enter the desired root password, and click Done.
33. To restart the server when the installation completes, select Reboot.

Configuring Oracle Enterprise Linux 7.5 for Oracle

1. Log onto the server as root.
2. Disable the firewall:

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

3. Disable SELinux:

```
vi /etc/selinux/config
SELINUX=disabled
```

4. Install the Oracle 12*c* preinstall RPM:

```
yum install oracle-database-server-12cR2-preinstall
```

5. Update Oracle Enterprise Linux 7.5:

```
yum update
```

6. Using yum, install the following prerequisite packages for Oracle Database:

```
yum install compat-libstdc++-33.i686
yum install glibc-devel.i686
yum install libstdc++-devel.i686
yum install libaio.i686
yum install libaio-devel.i686
yum install libXext.i686
yum install libXtst.i686
yum install unixODBC
yum install unixODBC-devel
yum install zlib-devel
yum install zlib-devel.i686
yum install xhost
```

7. Disable auditd:

```
systemctl disable auditd
```

8. Create Oracle users and groups:

```
groupadd -g 54327 asmdba
groupadd -g 54328 asmoper
groupadd -g 54329 asmadmin
usermod -g 54321 -g oinstall -G dba,oper,backupdba,dgdba,kmdba,asmdba,asmoper,asmadmin oracle
```

9. Create passwords for the Oracle account with passwd.
10. Create the following directories, and assign the following permissions:

```
mkdir -p /u01/app/12.2.0.1/grid
mkdir -p /u01/app/oracle/product/12.2.0.1/db_1
chown -R oracle:oinstall /u01
chmod -R 775 /u01/
```

11. Append the following to the /etc/security/limits.conf:

```
oracle - nofile 65536
oracle - nproc 16384
oracle - stack 32768
oracle - memlock 134217728
* soft memlock unlimited
* hard memlock unlimited
```

12. To modify the system's kernel parameters, append the following to /etc/systctl.conf:

```
vm.nr_hugepages = 131072
vm.hugetlb_shm_group = 54321
```

13. Add the following lines to the .bash_profile for the Oracle user:

```
export TMP=/tmp
export TMPDIR=$TMP

export ORACLE_HOSTNAME= <HOSTNAME>
export ORACLE_UNQNAME=orcl
export ORACLE_BASE=/u01/app/oracle
export GRID_HOME=/u01/app/12.2.0.1/grid
export DB_HOME=$ORACLE_BASE/product/12.2.0.1/db_1
export ORACLE_HOME=$DB_HOME
export ORACLE_SID=orcl
export ORACLE_TERM=xterm
export BASE_PATH=/usr/sbin:$PATH
export PATH=$ORACLE_HOME/bin:$BASE_PATH
```

```
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib
export CLASSPATH=$ORACLE_HOME/JRE:$ORACLE_HOME/jlib:$ORACLE_HOME/rdbms/jlib

alias grid_env='. /home/oracle/grid_env'
alias db_env='.    /home/oracle/db_env'
```

14. In the Oracle users home folder, create the following files:

```
>>>grid_env<<<
export ORACLE_SID=+ASM
export ORACLE_HOME=$GRID_HOME
export PATH=$ORACLE_HOME/bin:$BASE_PATH

export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib
export CLASSPATH=$ORACLE_HOME/JRE:$ORACLE_HOME/jlib:$ORACLE_HOME/rdbms/jlib
>>>db_env<<<
export ORACLE_SID=orcl
export ORACLE_HOME=$DB_HOME
export PATH=$ORACLE_HOME/bin:$BASE_PATH

export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib
export CLASSPATH=$ORACLE_HOME/JRE:$ORACLE_HOME/jlib:$ORACLE_HOME/rdbms/jlib
```

15. Install the oracleasmlib packages:

```
yum install -y oracleasm-support-*
```

16. To create a partition on all disks, use fdisk.
17. Edit /etc/sysconfig/oracleasm to contain the following:

```
# ORACLEASM_ENABLED: 'true' means to load the driver on boot.
ORACLEASM_ENABLED=true

# ORACLEASM_UID: Default UID owning the /dev/oracleasm mount point.
ORACLEASM_UID=oracle

# ORACLEASM_GID: Default GID owning the /dev/oracleasm mount point. ORACLEASM_GID=oinstall

# ORACLEASM_SCANBOOT: 'true' means fix disk perms on boot
ORACLEASM_SCANBOOT=true

# ORACLEASM_USE_LOGICAL_BLOCK_SIZE: 'true' means use the logical block
# size reported by the underlying disk instead of the physical. The
# default is 'false' ORACLEASM_USE_LOGICAL_BLOCK_SIZE=false
```

18. Initialize Oracle ASM:

```
oracleasm init
```

19. Configure all the disks for Oracle ASM:

```
oracleasm createdisk DATA       /dev/sdc1
oracleasm createdisk NVME_DATA1 /dev/nvme0n1p1
oracleasm createdisk NVME_DATA2 /dev/nvme1n1p1
oracleasm createdisk NVME_LOG1  /dev/nvme2n1p1
oracleasm createdisk NVME_LOG2  /dev/nvme3n1p1
```

20. Edit the /etc/default/grub file, and remove "numa=off" from the GRUB_CMDLINE.
21. For the rebuilt changes to take effect, run the following command:

```
grub2-mkconfig -o /etc/grub2-efi.cfg
```

22. Reboot the server.

Installing Oracle Grid Infrastructure 12c

1. Log in as the Oracle user.
2. Unzip linuxx64_12201_grid_home.zip
3. Open a terminal to the unzipped database directory.

4. To set the Oracle grid environment, type `grid_env`.
5. To start the installer, type `./gridSetup`.
6. In the Select Installation Option screen, select Install and Configure Grid Infrastructure for a Standalone Server, and click Next.
7. Choose the language, and click Next.
8. In the Create ASM Disk Group screen, choose the Disk Group Name, and change redundancy to Normal.
9. Change the path to /dev/oracleasm/disks, and select the data disk to use for the ASM instance.
10. In the Specify ASM Password screen, choose Use same password for these accounts, enter the passwords for the ASM users, and click Next.
11. At the Management Options screen, click Next.
12. Leave the default Operating System Groups, and click Next.
13. Leave the default installation, and click Next.
14. Leave the default inventory location, and click Next.
15. Under Root script execution, select Automatically run configuration scripts, and enter root credentials.
16. In the Prerequisite Checks screen, make sure there are no errors.
17. In the Summary screen, verify that everything is correct. To install Oracle Grid Infrastructure, click Finish.
18. The installation will prompt you to execute two configuration scripts as root. Follow the instructions to run the scripts.
19. At the Finish screen, click Close.
20. To run the ASM Configuration Assistant, type `asmca`.
21. In the ASM Configuration Assistant, click Create.
22. In the Create Disk Group window, name the new disk group `NVME_DATA`, choose redundancy Normal Redundancy, and select the two NVME_DATA disks.
23. In the Create Disk Group window, name the new disk group `NVME_LOG`, choose redundancy Normal Redundancy, and select the two NVME_LOG disks.
24. Click Advanced Options.
25. Set the database compatibility level to 12.2.0.1.0, and click OK.
26. Exit the ASM Configuration Assistant.

Installing Oracle Database 12c

1. Unzip linux_12201_database.zip.
2. Open a terminal to the unzipped database directory.
3. To set the first Oracle database environment, type `db_env`.
4. Run ./runInstaller.sh.
5. Wait for the GUI installer to load.
6. On the Configure Security Updates screen, enter the credentials for My Oracle Support. If you do not have an account, uncheck I wish to receive security updates via My Oracle Support, and click Next.
7. At the warning, click Yes.
8. On the Download Software Updates screen, enter the desired update option, and click Next.
9. On the Select Installation Option screen, select Install database software only, and click Next.
10. On the Grid Installation Options screen, select Single instance database installation, and click Next.
11. On the Select Product Languages screen, leave the default setting of English, and click Next.
12. On the Select Database Edition screen, select Enterprise Edition, and click Next.
13. On the Specify Installation Location, leave the defaults, and click Next.
14. On the Create Inventory screen, leave the default settings, and click Next.
15. On the Privileged Operating System groups screen, keep the defaults, and click Next.
16. Allow the prerequisite checker to complete.
17. On the Summary screen, click Install.
18. Once the Execute Configuration Scripts prompt appears, SSH into the server as `root`, and run the following command:

```
# /u01/app/oracle/product/12.2.0.1/db_1/root.sh
```

19. Return to the prompt, and click OK.
20. Once the installer completes, click Close.

Creating and configuring the database

1. Using Putty with X11 forwarding enabled, SSH to the VM.
2. To switch to the first database environment, type `db_env`.
3. Type `dbca`, and press Enter to open the Database configuration assistant.
4. At the Database Operation screen, select Create Database, and click Next.
5. Under Creation Mode, select Advanced Mode, and click Next.

6. At the Deployment Type screen, select General Purpose or Transaction Processing. Click Next.
7. Enter a Global database name and the appropriate SID, and uncheck Create as Container database. Click Next.
8. At the storage option screen, select Use following for the database storage attributes.
9. In the drop-down menu, select Automatic Storage Management (ASM), and select +NVMe_DATA for the file location.
10. At the Fast Recovery Option screen, check Specify Fast Recovery Area.
11. In the drop-down menu, select ASM, select +DATA for the Fast Recovery Area, and type `500GB` for the size.
12. At the Network Configuration screen, select the listener, and click Next.
13. At the Data Vault Option screen, leave as default, and click Next.
14. At the Configuration Options screen, set the SGA size to 147456 and the PGA size to 49152, and click Next.
15. At the Management Options screen, select Configure Enterprise Manager (EM) Database Express, and click Next.
16. At the User Credentials screen, select Use the same administrative password for all accounts, enter and confirm the desired password, and click Next.
17. At the Creation Options screen, select Create Database, and click Next.
18. At the summary screen, click Finish.
19. Close the Database Configuration Assistant.

Configuring Oracle Tablespaces and redo log

Alter the tablespaces on both systems as shown below. Type `sqlplus / as sysdba` to enter SQL prompt.

```
ALTER DATABASE ADD LOGFILE GROUP 11 ( '/tmp/temp1.log' ) SIZE 50M;
ALTER DATABASE ADD LOGFILE GROUP 12 ( '/tmp/temp2.log' ) SIZE 50M;

ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM CHECKPOINT;

ALTER DATABASE DROP LOGFILE GROUP 1;
ALTER DATABASE DROP LOGFILE GROUP 2;
ALTER DATABASE DROP LOGFILE GROUP 3;

ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM CHECKPOINT;

alter system set "_disk_sector_size_override"=TRUE scope=both;

ALTER DATABASE ADD LOGFILE GROUP 1 ( '+NVME_LOGS/redo01.log' ) SIZE 800G
BLOCKSIZE 4K;
ALTER DATABASE ADD LOGFILE GROUP 2 ( '+NVME_LOGS/redo02.log' ) SIZE 800G
BLOCKSIZE 4K;

ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM CHECKPOINT;
ALTER DATABASE DROP LOGFILE GROUP 11;
ALTER DATABASE DROP LOGFILE GROUP 12;

HOST rm -f /tmp/temp*.log

CREATE BIGFILE TABLESPACE "TPCC"
DATAFILE '+NVME_DATA/ORCL/DATAFILE/tpcc.dbf' SIZE 200G AUTOEXTEND ON NEXT 1G
BLOCKSIZE 8K
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
SEGMENT SPACE MANAGEMENT AUTO;

CREATE BIGFILE TABLESPACE "TPCC_OL"
DATAFILE '+NVME_DATA/ORCL/DATAFILE/tpcc_ol.dbf' SIZE 100G AUTOEXTEND ON NEXT 1G
BLOCKSIZE 16K
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
SEGMENT SPACE MANAGEMENT AUTO;

ALTER DATABASE DATAFILE '+NVME_DATA/ORCL/DATAFILE/<UNDO FILE>' RESIZE 32760M;
```

Configuring the Oracle pfile

Alter the Oracle pfile as shown below. To make Oracle use it, type the following, and restart Oracle:

```
CREATE SPFILE = '+NVME_DATA/ORCL/spfileorcl.ora' FROM PFILE='$ORACLE_HOME/pfile.ora'

_ash_enable=FALSE
_awr_restrict_mode=FASLE
_check_block_after_checksum=FALSE
_collect_undo_stats=FALSE
_db_block_check_objtyp=FALSE
_db_block_prefetch_limit=0
_disable_highres_ticks=TRUE
_first_spare_parameter=1
_numa_shift_enabled=FALSE
_resource_manager_always_off=TRUE
_trace_pool_size=0
_use_adaptive_log_file_sync=FALSE
_disable_logging=FALSE
_disable_selftune_checkpointing=TRUE
_disk_sector_size_override=TRUE
_enable_NUMA_interleave=TRUE
_enable_NUMA_support=TRUE
_fast_cursor_reexecute=TRUE
_in_memory_undo=TRUE
_kgl_hot_object_copies=8
aq_tm_processes=0
audit_trail=NONE
commit_logging=IMMEDIATE
commit_wait=WAIT
compatible=12.2.0.1.0
control_files='+NVME_DATA/ORCL/CONTROLFILE/Current.260.992224907','+DATA/ORCL/CONTROLFILE/
Current.265.992224907'
db_16k_cache_size=34359738368
db_4k_cache_size=34359738368
db_block_checking=false
db_block_checksum=false
db_block_size=8192
db_cache_size=128g
db_create_file_dest='+NVME_DATA'
db_name='orcl'
db_recovery_file_dest='+DATA'
db_recovery_file_dest_size=500g
db_file_multiblock_read_count=4
db_writer_processes=4
diagnostic_dest='/u01/app/oracle'
disk_asynch_io=TRUE
dispatchers='(PROTOCOL=TCP) (SERVICE=hammerXDB)'
dml_locks=500
fast_start_mttr_target=0
java_pool_size=3221225472
large_pool_size=3221225472
lock_sga=TRUE
log_buffer=2147418112
log_checkpoint_interval=0
log_checkpoint_timeout=0
log_checkpoints_to_alert=TRUE open_cursors=2000
parallel_max_servers=0
parallel_min_servers=0
pga_aggregate_target=3589934592
pre_page_sga=FALSE
processes=1000
query_rewrite_enabled=FALSE
remote_login_passwordfile=EXCLUSIVE
replication_dependency_tracking=FALSE
```

```
result_cache_max_size=0
shared_pool_size=11811160064
statistics_level='BASIC'
timed_statistics=false
trace_enabled=FALSE
transactions=2000
transactions_per_rollback_segment=1
undo_management=AUTO
undo_retention=2
undo_tablespace=UNDOTBS1
use_large_pages=ONLY
_undo_autotune=FALSE
_db_writer_flush_imu=false
plsql_optimize_level=3
plsql_code_type="NATIVE"
```

Configuring the HammerDB client for Oracle Database

We used the same HammerDB client for the PostgreSQL and Oracle testing.

1. Create an Oracle user:

   ```
   useradd oracle
   ```

2. Assign the Oracle user a password:

   ```
   passwd oracle
   ```

3. Log in as the Oracle user.
4. Navigate to the folder with the client installer, and launch it.
5. In Select Installation Type, select Administrator (1.8 GB) as the installation type, and click Next.
6. In Specify Installation Location, accept default locations provided, and click Next.
7. In Create Inventory, accept the defaults, and click Next.
8. In Summary, review the information. To begin installation, click Install.
9. In Install Product, follow the instructions to execute the scripts. When the scripts have completed, click OK.
10. In Finish, click Close to exit the installer.
11. Append the following to ~/.bash_profile.

    ```
    export ORACLE_HOME=/home/oracle/app/oracle/product/12.2.0/client_1
    export LD_LIBRARY_PATH=$ORACLE_HOME/lib
    export ORACLE_LIBRARY=$ORACLE_HOME/lib/libclntsh.so
    export PATH=$ORACLE_HOME/bin:$PATH
    ```

Generating the HammerDB TPC-C-like 800-warehouse database

1. Log into the HammerDB client as the Oracle user.
2. Navigate to the HammerDB directory.
3. Start the HammerDB cli:

   ```
   ./hammerdbcli
   ```

4. Set the following build parameters:

   ```
   dbset db ora
   dbset bm TPC-C
   diset connection system_password <Password>
   diset connection instance <IP ADDRESS/orcl>
   diset tpcc count_ware 800
   diset tpcc num_vu 24
   diset tpcc tpcc_def_tab tpcc
   diset tpcc tpcc_ol_tab tpcc_ol
   diset tpcc partition true
   diset tpcc hash_clusters true
   ```

5. Build the schema:

   ```
   buildschema
   ```

Running the test

We ran the tests five times on each configuration. We used RMAN to backup and restore the database between runs.

1. Log into the HammerDB client as the Oracle user.
2. Navigate to the HammerDBB directory.
3. Start the HammerDB cli:

```
./hammerdbcli
```

4. Set the following run parameters:

```
dbset db ora
dbset bm TPC-C
diset connection system_password <Password>
diset connection  instance <IP ADDRESS/orcl>
diset tpcc count_ware 800
diset tpcc total_iteration 1000000000
diset tpcc driver timed
diset tpcc rampup 15
diset tpcc duration 30
diset tpcc timeprofile true
```

5. Reload the newly created script:

```
loadscript
```

6. Configure and create the virtual users:

```
vuset vu 450
vuset showoutput 1
vuset logtotemp 1
vuset unique 1
vucreate
```

7. Start the test:

```
vurun
```

8. When the test is finished, kill the virtual users:

```
vudestroy
```

How we tested the HammerDB workload for PostgreSQL

We tested on two different servers with three different processor configurations. We equipped the first server, a Supermicro H11DSU-iN, with AMD EPYC 7601 processors. We equipped the second server, an Intel S2600WFD, with Intel Xeon Platinum 8180 processors, and then re-tested with Intel Xeon Platinum 8160 processors. We populated the RAM in each server so each CPU channel would have one memory stick, which gave the AMD EPYC 7601 processor-based system 512 GB of RAM and the Intel Xeon Platinum 8180 and 8160 processor-powered system 384 GB. On each system, we installed the OS on a single SSD because the servers were not equipped with RAID controllers. This installation is fine for testing purposes, but we do not advise using the installation for a production environment. We then configured two NVMe SSDs in a Linux software RAID for the database data and two NVMe SSDs in a Linux software RAID for the logs.

We installed PostgreSQL on each system and generated an 800-warehouse database with 150 users running on client VMs. We set a 15-minute warm-up for the test, which had a 30-minute runtime. We backed up and restored the database with tar. We ran the test five times and chose the median runs for our calculations.

Installing and configuring Red Hat Enterprise Linux 7.5 for PostgreSQL

1. Boot from the Red Hat Enterprise Linux 7.5 installation DVD.
2. Remove the home partition, and allocate the remaining disk space to the root partition.
3. Choose Minimal Install.
4. Disable kdump.
5. Enable the first active network port, and configure it (for example, assign static IP address of 10.200.99.1/16 with gateway of 10.200.0.1 and DNS server of 10.41.0.10).
6. Set the hostname.
7. Set the time zone to Eastern/US, and enable NTP to sync only from 10.40.0.1.
8. Click Install.
9. Set the root password.
10. Once the install has finished, reboot.
11. Log in as root.
12. Connect to the Red Hat Repository, and update the operating system software:

```
subscription-manager register
subscription-manager attach --auto
yum -y update
```

13. Set SELINUX to disabled in the following config file:

```
vi /etc/selinux/config
```

14. Disable the firewall:

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

15. Create a user `postgres`, and give it a password:

```
useradd postgres
passwd postgres
```

16. Create a partition on each NVMe SSD with fdisk.
17. Install mdadm:

```
yum install mdadm
```

18. Using the NVMe SSDs that you just created a partition on, create two RAID 1 volumes:

```
mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/nvme0n1 /dev/nvme1n1
mdadm --create --verbose /dev/md1 --level=1 --raid-devices=2 /dev/nvme2n1 /dev/nvme3n1
```

19. On each RAID 1 volume, create a file system:

```
mkfs.ext4 -F /dev/md0
mkfs.ext4 -F /dev/md1
```

Installing and configuring PostgreSQL 9.6

1. Log in as the root user. Install the repository RPM:

```
yum install https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-
redhat96-9.6-3.noarch.rpm
```

2. Install the client packages:

```
yum install postgresql96
```

3. Install the server packages:

```
yum install postgresql96-server
```

4. Create a folder for the logs:

```
mkdir /var/lib/pgsql/9.6/log
```

5. Mount the RAID volumes to the pgsql data and log directories:

```
mount /dev/md0 /var/lib/pgsql/9.6/data/
mount /dev/md1 /var/lib/pgsql/9.6/log/
```

6. To make the RAID volume mounts persistent, append the following lines to /etc/fstab:

```
/dev/md0 /var/lib/pgsql/9.6/data ext4 defaults 0 0
/dev/md1 /var/lib/pgsql/9.6/log ext4 defaults 0 0
```

7. To enable hugepages, append the following to /etc/sysctl.conf:

```
vm.nr_hugepages = 98304
```

8. Initialize the database, and enable automatic startup:

```
/usr/pgsql-9.6/bin/postgresql96-setup initdb
systemctl enable postgresql-9.6
systemctl start postgresql-9.6
```

9. Change ownership of the pgsql directory to the postgres user.

```
chown -R postgres /var/lib/pgsql/
```

10. Log in as the postgres user.
11. Append the following lines to ~/.bash_profile:

```
LD_LIBRARY_PATH=/usr/pgsql-9.6/lib
export LD_LIBRARY_PATH
```

12. Shut down the database:

```
/usr/pgsql-9.6/bin/pg_ctl -D /var/lib/pgsql/9.6/data/ stop
```

13. Move the WAL to the log directory:

```
mv /var/lib/pgsql/9.6/data/pg_xlog /var/lib/pgsql/9.6/log/pg_xlog
```

14. Create a symbolic link to pg_xlog:

```
ln -s /var/lib/pgsql/9.6/log/pg_xlog /var/lib/pgsql/9.6/data/pg_xlog
```

15. Append the following to /var/lib/pgsql/9.6/data/pg_hba.conf:

```
Host all all <IP ADDRESS/PREFIX> trust
```

16. Edit /var/lib/pgsql/9.6/data/postgresql.conf to match the PostgreSQL config.
17. Start the database.

```
/usr/pgsql-9.6/bin/pg_ctl -D /var/lib/pgsql/9.6/data/ start
```

18. Reboot the server.

```
shutdown -r now
```

Setting up the HammerDB client

We created VMware ESXi 6.5 virtual machines with 24 vCPUs, 32 GB of RAM, and 10GbE network connections. We installed Red Hat Enterprise Linux 7.5 following the steps at the beginning of this appendix. We used HammerDB version 3.0.

1. Download and copy the HammerDB-3.0.tar.gz file to the Linux machine.
2. Install the tar package:

```
tar -zxvf HammerDB-3.0.tar.gz
```

Configuring the HammerDB client for PostgreSQL

1. Install the repository RPM:

```
yum install https://download.postgresql.org/pub/repos/yum/9.6/redhat/rhel-7-x86_64/pgdg-redhat96-9.6-3.noarch.rpm
```

2. Install the client packages:

```
yum install postgresql96
```

3. Append the following lines to ~/.bash_profile:

```
LD_LIBRARY_PATH=/usr/pgsql-9.6/lib
export LD_LIBRARY_PATH
```

Generating the HammerDB TPC-C-like 800-scale database

1. Log into the HammerDB client.
2. Navigate to the HammerDB directory.
3. Start the HammerDB cli:

```
./hammerdbcli
```

4. Set the following build parameters:

```
dbset db pg
dbset bm TPC-C
diset connection pg_host <IP ADDRESS>
diset tpcc pg_count_ware 800 diset tpcc pg_num_vu 24 diset tpcc pg_superuserpass <Password>
```

5. Build the schema:

```
buildschema
```

Running the test

We used tar to back up and restore the database between runs.

1. Log into the HammerDB client.
2. Navigate to the HammerDB directory.
3. Start the HammerDB cli:

```
./hammerdbcli
```

4. Set the following run parameters:

```
dbset db pg
dbset bm TPC-C
diset connection pg_host <IP ADDRESS>
diset tpcc pg_count_ware 800
diset tpcc pg_superuserpass <Password>
diset tpcc pg_total_iteration 1000000000
diset tpcc pg_driver timed
diset tpcc pg_rampup 15
diset tpcc pg_duration 30
diset tpcc pg_timeprofile true
```

5. Reload the newly created script:

```
loadscript
```

6. Configure and create the virtual users:

```
vuset vu 250
vuset showoutput 1
vuset logtotemp 1
vuset unique 1
vucreate
```

7. Start the test:

```
vurun
```

8. When the test is finished, kill the virtual users:

```
vudestroy
```

Configuring PostgreSQL

Postgresql.conf

```
listen_addresses = '<IP ADDRESS>'
port = 5432
max_connections = 256
shared_buffers = 64000MB
huge_pages = on
temp_buffers = 4000MB
work_mem = 4000MB
maintenance_work_mem = 512MB
autovacuum_work_mem = -1
max_stack_depth = 7MB
dynamic_shared_memory_type = posix
max_files_per_process = 4000
effective_io_concurrency = 32
wal_level = minimal
synchronous_commit = off
wal_buffers = 512MB
checkpoint_timeout = 1h
checkpoint_completion_target = 1.0
checkpoint_warning = 0
effective_cache_size = 192GB
log_destination = 'stderr'
logging_collector = on
log_directory = 'pg_log'
log_filename = 'postgresql-%a.log'
log_truncate_on_rotation = on
log_rotation_age = 1d
log_rotation_size = 0
log_min_messages = error
log_min_error_statement = error
log_timezone = 'US/Eastern'
autovacuum = off
datestyle = 'iso, mdy'
timezone = 'US/Eastern'
lc_messages = 'en_US.UTF-8'
lc_monetary = 'en_US.UTF-8'
lc_numeric = 'en_US.UTF-8'
lc_time = 'en_US.UTF-8'
default_text_search_config = 'pg_catalog.english'
max_locks_per_transaction = 64
max_pred_locks_per_transaction = 64
```

How we tested the Apache Cassandra workload

Configuring the servers under test

We enabled virtualization in the Intel BIOS, and in the AMD BIOS, we set determinism to power and statically set the memory clock speed to 2,666Mhz. Each server had a 1Gb connection to an infrastructure network and a 10Gb connection to a testing network. We installed Ubuntu 18.04 with default packages onto the OS drive in each server, leaving the two NVMe SSDs for VM volumes.

Installing Ubuntu

1. Boot the server to the Ubuntu 18.04 installation media.
2. Select English as the preferred language, and press Enter.
3. Select English (US) as the keyboard layout, and press Enter.
4. Select Install Ubuntu, and press Enter.
5. Ensure the 1Gb adapter has DHCP and the 10Gb adapter has no address, and press Enter.
6. Press Enter to skip the proxy address page.
7. Press Enter to accept the default mirror address.
8. Select Use an Entire Disk, and press Enter.
9. Select the OS disk, and press Enter.
10. Select Done, and press Enter.
11. Select Continue, and press Enter.
12. Enter a name, hostname, username, and password. Select Done, and press Enter.
13. Select Done, and press Enter.
14. When the installation completes, remove the installation media, and reboot the server.

Updating and configuring the operating system

1. Log into the operating system as the user you created in the previous section.
2. Disable the system firewall by running the following command:

```
sudo ufw disable
```

3. Update the operating system and reboot by running the following commands:

```
sudo apt -y upgrade
sudo apt -y update
sudo reboot
```

4. Configure password-less SSH:
   a. On one of the hosts, configure SSH by running the following commands:

```
rm -rf ~/.ssh
mkdir -p ~/.ssh
chmod 700 ~/.ssh
cd ~/.ssh
ssh-keygen -t rsa -q
cp id_rsa.pub authorized_keys
echo "StrictHostkeyChecking=no" > config
echo > known_hosts
cp -rp ~/.ssh ~/clean_ssh
```

   b. Copy the keys to the root user:

```
sudo su
rm -rf /root/.ssh
cp -rp ../clean_ssh /root/.ssh
chown -R root:root /root/.ssh
```

   c. Copy to other hosts:

```
ssh <remotehost> 'mkdir ~/.ssh'
scp -rp ~/.clean_ssh/* remotehost:~/.ssh
```

   d. SSH to each host, and copy the keys to the root user:

```
ssh remotehost
```

```
cd ~
sudo su
rm -rf /root/.ssh
cp -rp .ssh /root/.ssh
chown -R root:root /root/.ssh
exit
exit
```

5. Configure the servers to run in performance mode by typing the following commands:

```
sudo apt -y install cpufrequtils
echo 'GOVERNOR="performance"' > /etc/default/cpufrequtils
sudo systemctl disable ondemand
sudo /etc/init.d/cpufrequtils restart
```

Configuring the NVMe SSDs

1. Create a physical volume using the two NVMe SSDs by running the following command:

```
sudo pvcreate /dev/nvme0n1 /dev/nvme1n1
```

2. Create a volume group:

```
sudo vgcreate vgdata /dev/nvme0n1 /dev/nvme2n1
```

Installing the hypervisor

1. Install the required packages by running the following command:

```
sudo apt -y install qemu-kvm libvirt-bin
```

2. Edit the file `/etc/netplan/01-netcfg.yaml` to reflect the following (replace `192.168.0.21/24` with the desired host IP address and replace adapter names if necessary):

```
network:
    version: 2
    renderer: networkd
    ethernets:
        eno1:
            addresses: []
            dhcp4: true
            optional: true
        eno2:
            addresses: []
            dhcp4: true
            optional: true
        eno3:
            addresses: []
            dhcp4: true
        eno4:
            addresses: []
            dhcp4: true
            optional: true
        enp225s0f0:
            addresses: []
            dhcp4: true
            optional: true
        enp225s0f1:
            addresses: []
            dhcp4: true
    bridges:
        br0:
            dhcp4: false
            addresses: [192.168.0.21/24]
            interfaces:
              - enp225s0f1
```

3. Apply the networking configuration by running the following command:

```
sudo netplan apply
```

4. Create a file named `br0.xml`, and type the following to configure the bridge network:

```
<network>
  <name>br0</name>
  <forward mode='bridge'/>
  <bridge name='br0'/>
</network>
```

5. Define and start the bridge network with the following commands:

```
sudo virsh net-define ~/br0.xml
sudo virsh net-start br0
sudo virsh net-autostart br0
```

6. Download the Ubuntu 18.04 ISO by running the following command:

```
wget http://www.cdimage.ubuntu.com/ubuntu/releases/18.04/release/ubuntu-18.04.1-server-amd64.iso
```

Creating the VMs

We used a separate server with Ubuntu 16.04 to host Virtual Machine Manager (VMM). We updated the operating system, disabled the firewall, and configured password-less SSH.

Installing Virtual Machine Manager and connecting the hosts

1. Install Ubuntu Desktop by running the following commands:

```
sudo apt -y install ubuntu-desktop
sudo reboot
```

2. Install VMM by running the following command:

```
sudo apt -y install virt-manager
```

3. Open VMM, and click File→Add Connection.
4. Select QEMU/KVM as the hypervisor.
5. Check the Connect to remote host checkbox.
6. Select SSH as the connection method.
7. Enter the host username and hostname.
8. Check the Autoconnect checkbox.
9. Click Connect.

Creating the storage pool for the installation media

1. Right-click the desired host, and click Details.
2. Click the Storage tab.
3. Click the plus sign for add pool.
4. Enter a name for the pool, and click Forward.
5. Enter the file location of the Ubuntu 18.04 ISO, and click Finish.

Creating the storage pool on the NVMe SSDs for VM volumes

1. Right-click the desired host, and click Details.
2. Click the Storage tab.
3. Click the plus sign for add pool.
4. Enter a name for the pool, select logical: LVM Volume Group, and click Forward.
5. Enter `/dev/vgdata` for the Target Path, and click Finish.

6. Enter the file location of the Ubuntu 18.04 ISO, and click Finish.
7. Creating the VM Select the desired host, and click New VM.
8. Click Forward.
9. Click Browse.
10. Select the installation media storage pool in the left pane, then select the Ubuntu 18.04 ISO in the right pane, and click Choose Volume.
11. Click Forward.
12. Enter 8 for the number of CPUs for the VM and 32768 for the amount of memory. Click Forward.
13. Select the radio button for Select or create custom storage, and click Manage. Select the NVMe storage pool in the left pane, and click the Create new volume button.
14. Enter a name for the volume and a Max Capacity to match the VM storage required. Click Finish.
15. Click Choose Volume.
16. Click Forward.
17. Enter a name for the VM, select Virtual network 'br0': Bridge network for the network selection, and click Finish.
18. Right-click the VM, and click Open.
19. Click Show virtual hardware details.
20. In the left pane, select IDE Disk 1. Click Advanced options, and select VirtIO for the Disk bus. Click Apply.
21. In the left pane, select NIC. Select virtio for the Device model, and click Apply.
22. In the left pane, select the Sound Controller, and click Remove.
23. In the left pane, select Display, and select Spice server for Type, All interfaces for Address, and en-us for Keymap. Click Apply.
24. Power on the VM, and follow the steps in the OS configuration section to install Ubuntu 18.04, update the OS, disable the firewall, and configure password-less SSH.

Installing the Cassandra workload

We created 10 Cassandra VMs on each server in the Intel Xeon Platinum 8180 and 8160 processor-powered cluster and 12 Cassandra VMs on each server in the AMD EPYC 7601 processor-based cluster. We configured Cassandra so that all the Intel Xeon Platinum 8180 and 8160 processor-powered VMs were in one cluster, and all the AMD EPYC 7601 processor-based VMs were in another cluster. We used a bare-metal server with threaded cassandra-stress instances to drive the Cassandra workload.

Installing Cassandra on the target VMs

1. Install Cassandra by running the following commands:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt -y update
sudo apt -y install oracle-java8-installer

echo "deb http://www.apache.org/dist/cassandra/debian 311x main" | sudo tee -a /etc/apt/sources.
list.d/cassandra.sources.list

curl https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -

sudo apt -y update
sudo apt -y install cassandra cassandra-tools
```

2. Edit `/etc/cassandra/cassandra.yaml` to reflect the following:

```
seeds: "<IP address of VM1>,<IP address of VM2>,<etc>"
start_rpc: true
# rpc_address: localhost
rpc_interface: ens3
# listen_address: localhost
listen_interface: ens3
```

3. Apply the new Cassandra configuration by running the following commands:

```
sudo systemctl stop cassandra
sudo rm -rf /var/lib/cassandra/*
sudo systemctl start cassandra
```

Installing the workload driver on the client server

1. Install the Cassandra toolset by running the following commands:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt -y update
sudo apt -y install oracle-java8-installer

echo "deb http://www.apache.org/dist/cassandra/debian
311x main" | sudo tee -a
/etc/apt/sources.list.d/cassandra.sources.list

curl https://www.apache.org/dist/cassandra/KEYS |
sudo apt-key add -

sudo apt -y update
sudo apt -y install cassandra cassandra-tools
```

2. Create a file named `cluster.<Intel or AMD>` with a comma-separated list of the Cassandra node IP addresses, for example:

```
192.168.0.101,192.168.0.102,192.168.0.103,<etc>
```

3. We used a script to run `cassandra-stress` on each VM, spread evenly across NUMA nodes. Create a script named `numarun.sh`:

```
rm ~/casstest*

count=$1
cpus=$[<number of CPU cores>/$count]

echo count=$count
echo cpus=$cpus

for i in  $(seq 1 1 $count)
do
        numactl -C $[($i-1)*$cpus]-$[$i*$cpus-2] -l cassandra-stress write no-warmup duration=600s
-rate threads=1000 -schema 'replication(factor=3)' -node $(cat $2) > casstest$i.txt &
done
```

---

1 "Memory Population Guidelines for AMD EPYC™ Processors," accessed January 22, 2019, https://developer.amd.com/wp-content/resources/56301_1.0.pdf

Read the report at http://facts.pt/m7yd6e1 ▶

This project was commissioned by Intel.