The science behind the report:

AMD EPYC™ 7763 processor-based servers can offer a better value for MySQL workloads

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 AMD EPYC™ 7763 processor-based servers can offer a better value for MySQL workloads.

We concluded our hands-on testing on September 14, 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 September 2, 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 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.

	3rd Gen AMD EPYC 7763 processor-based server cluster	3rd Gen Intel Xeon Platinum 8380 processor-based server cluster
Cost per server with three-year support (USD)	$29,265.25	$39,197.62
Total hardware cost per four-node cluster with three-year support (USD)	$117,061.00	$156,790.48
Transactions per minute (TPM)	24,913,848	20,046,107
TPM/dollar ratio	212.83	127.85
Average cluster CPU utilization (percentage)	84	94

CPU utilization statistics

Figures 1 and 2 show the CPU utilization during testing for the two clusters.

Figure 1: CPU utilization during testing for the Intel Xeon Platinum 8380 processor-based cluster.

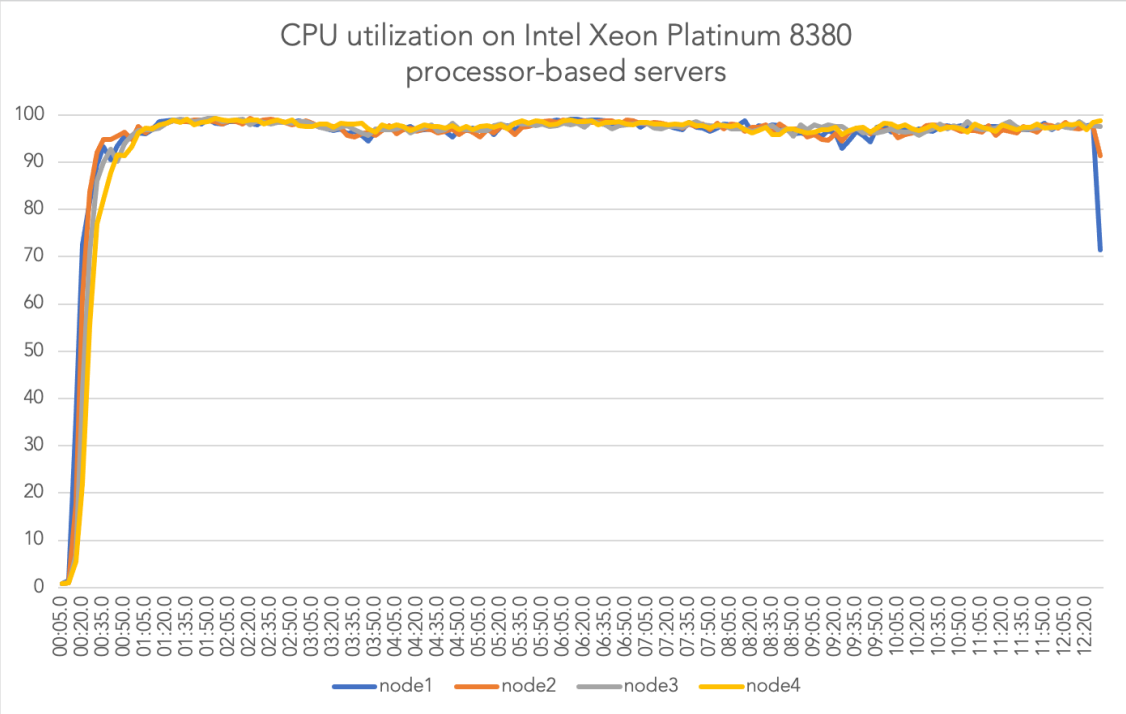
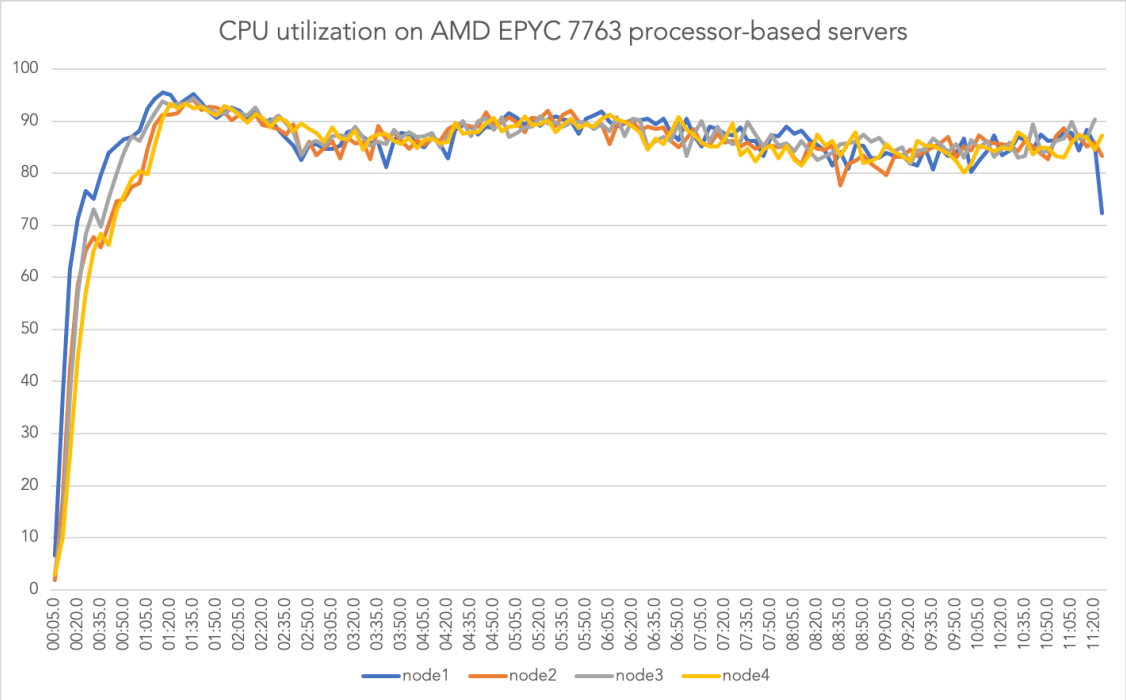


Figure 2: CPU utilization during testing for the AMD EPYC 7763 processor-based cluster.



System configuration information

Table 2: Detailed information on the system we tested.

System configuration information	Supermicro SYS-620U-TNR	Supermicro AS-1124US-TNRP
BIOS name and version	1.1A	American Megatrends Inc. 2.3a
Operating system name and version/build number	VMware® ESXi™ 7.0U3e-19898904-standard	VMware ESXi 7.0U3e-19898904-standard
Date of last OS updates/patches applied	6/17/2022	6/17/2022
Power management policy	Extreme Performance	Determinism Slider: Power
Processor		
Number of processors	2	2
Vendor and model	Intel® Xeon® Platinum 8380	AMD EPYC 7763
Core count (per processor)	40	64
Core frequency (GHz)	2.30	2.45
Stepping	N/A	N/A
Memory module(s)		
Total memory in system (GB)	1,024	1,024
Number of memory modules	16	16
Vendor and model	Micron MTA36ASF8G72PZ-3G2E1	Micron MTA36ASF8G72PZ-3G2E1
Size (GB)	64	64
Type	PC4-3200	PC4-3200
Speed (MHz)	3,200	3,200
Speed running in the server (MHz)	3,200	3,200
Local storage – OS		
Number of drives	1	1
Drive vendor and model	Intel SSD SSDSC2KB240G8	Micron MTFDHBA256TCK
Drive size (GB)	240	240
Drive information (speed, interface, type)	6Gb, SATA, SSD	PCIe 3.0, NVMe
Local storage – VMware vSAN™ capacity		
Number of drives	4	4
Drive vendor and model	KIOXIA KCD6XLUL3T84	KIOXIA KCD6XLUL3T84
Drive size (GB)	3.84	3.84
Drive information (speed, interface, type)	PCIe® 4.0, NVMe® 1.4, 64GT/s	PCIe 4.0, NVMe 1.4, 64GT/s
Local storage – VMware vSAN cache		
Number of drives	2	2
Drive vendor and model	KIOXIA KCM61VUL1T60	KIOXIA KCM61VUL1T60
Drive size (GB)	1.60	1.60
Drive information (speed, interface, type)	PCIe 4.0, NVMe SSD, 1.4	PCIe 4.0, NVMe SSD, 1.4

System configuration information	Supermicro SYS-620U-TNR	Supermicro AS-1124US-TNRP
Network adapter 1		
Vendor and model	Supermicro AOC-2UR68G4-I2XT	Supermicro AOC-URG4N4-i4XTS
Number and type of ports	2x 10GbE	4x 10GbE
Driver version	8.10 0x80009661	N/A
Network adapter 2		
Vendor and model	Supermicro AOC-S25G-M2S-O	Supermicro AOC-S25G-M2S-O
Number and type of ports	2x 25GbE SFP28	2x 25GbE SFP28
Driver version	14.28.2006	14.28.2006
Cooling fans		
Vendor and model	Sunon VF80381B1	Sunon PF40561BX-Q40U-S9H
Number of cooling fans	4	8
Power supplies		
Vendor and model	Supermicro PWS-2K08A-1R	Supermicro PWS-1K22A-1R
Number of power supplies	2	2
Wattage of each (W)	2,000	1,200

How we tested

We installed and configured the latest available version of VMware vSphere® 7.0.3 build 19717403 on a cluster of four Supermicro AS-1124US-TNRP servers with AMD EPYC 7763 processors, and a cluster of four Supermicro SYS-620U-TNR servers with Intel Xeon Platinum 8380 processors. We installed the hypervisor on internal SSDs. On each cluster, we created a 55TB VMware vSAN datastore using six NVMe drives on each server: two disk groups, each with one 1.60TB drive for cache and two 3.84TB drives for capacity. The vSAN datastore served as a shared datastore for VMware SQL Server/HammerDB deployment.

We deployed 32 VMs (8 per host) in each cluster. Each VM had 112 GB of memory fully reserved, a 60GB OS drive, and a 500GB data drive. Due to the core count difference between the AMD EPYC and Intel Xeon Scalable processors, we configured the 32 VMs to use all threads in each environment: 32 vCPUs per VM on the AMD EPYC processor-based cluster and 20 vCPUs on the Intel Xeon Scalable processor-based cluster. On each VM, we installed MySQL Community Server, version 8.0.3. To test the database performance of each environment, we used the HammerDB benchmark suite TPROC-C workload. Each VM had a single MySQL® instance with a 1,000-warehouse TPROC-C database. We targeted the maximum transactions per minute (TPM) each cluster could achieve by increasing the user count until performance degraded. The max user count was 144 for both. We deployed two servers for HammerDB clients to drive the workload. Each client host ran 16 client VMs. We configured each client VM with 6 vCPUs, 18 GB of memory, and 60GB OS storage.

We used a Dell Networking 1Gb X1052 switch for the VMware management network. We used a Dell 5048-ON 25Gb switch for vMotion®, vSAN, and the VM network. On the 25Gb switch, we configured the following VLANs: 2 for the VM network, 20 for the AMD EPYC processor-based cluster vMotion network, 30 for the AMD EPYC processor-based cluster vSAN network, 40 for the Intel Xeon Scalable processor-based cluster vMotion network, and 60 for the Intel Xeon Scalable processor-based vSAN network. We also configured a port channel for each host using the two 25Gb ports from each host. We set the 25Gb switch MTU setting to 9216. We configured a two-port LAG on a Distributed vSwitch using two 25Gb ports from each host connected to the 25Gb switch. On the Distributed vSwitch, we created vSAN, vMotion, and workload port groups.

We optimized the AMD EPYC processor-based cluster BIOS based on the VMware vSphere tuning guide for AMD EPYC 7003, most of which was already the default. We deviated slightly by setting the NPS to 4 instead of the default of 1. We also set the fans to max speed to keep the CPU as cool as possible for greater power consumption. For the Intel Xeon Scalable processor-based cluster, we did not find a tuning guide specifically for VMware, so we set the system to Extreme Performance and the sub NUMA clustering to 2. We performed tuning runs on the default sub NUMA clustering set to disabled and achieved a higher score with it set to 2, so we tested with that setting out of fairness. In vSphere on the AMD EPYC processor-based cluster, we pinned the VMs in a round robin fashion to the NUMA nodes 1 to 1. Because the maximum number of NUMA nodes we could set in Intel was half that of AMD, we pinned 2 VMs per NUMA in a round-robin fashion.

The following sections describe the steps we took to configure the test environment and run the test.

Installing VMware vCenter Server Appliance version 7.0.3 build 19717403

Note: We deployed the vCenter appliance on one of the client hosts for management of the vSphere environment.

1. Download VMware vCenter version 7.0.3 build 19717403 from the VMware support portal at https://my.vmware.com.
2. Mount the image on your local system, and browse to the vcsa-ui-installer folder. Expand the folder for your OS, and launch the installer if it doesn't automatically begin.
3. When the vCenter Server Installer wizard opens, click Install.
4. To begin installation of the new vCenter server appliance, click Next.
5. Check the box to accept the license agreement, and click Next.
6. Enter the IP address of the infrastructure server with VMware ESXi version 7.0.3 build 19717403. Provide the root password, and click Next.
7. To accept the SHA1 thumbprint of the server's certificate, click Yes.
8. Accept the VM name, and provide and confirm the root password for the VCSA. Click Next.
9. Set the size for environment you're planning to deploy. We selected Medium. Click Next.
10. Select the datastore on which to install vCenter. Accept the datastore defaults, and click Next.
11. Enter the FQDN, IP address information, and DNS servers you want to use for the vCenter server appliance. Click Next.
12. To begin deployment, click Finish.
13. When Stage 1 has completed, click Close. To confirm, click Yes.
14. Open a browser window and connect to https://[vcenter.FQDN:5480/.
15. On the Getting Started - vCenter Server page, click Set up.
16. Enter the root password, and click Log in.
17. Click Next.
18. Enable SSH access, and click Next.
19. To confirm the changes, click OK.
20. Type `vsphere.local` for the Single Sign-On domain name. Enter a password for the administrator account, confirm it, and click Next.
21. Click Next.
22. Click Finish.

Installing vSphere version 7.0.3 build 19717403 on the Supermicro servers

1. Download ESXi Version 7.0.3 build 19717403 from https://my.vmware.com/group/vmware/evalcenter?p=vsphere-eval-7#tab_download.
2. Open a new browser tab, and connect to the IP address of the Supermicro server BMC.
3. Log in with the BMC credentials.
4. In the main screen, click Launch Virtual Console.
5. In the console menu bar, select Virtual Media, and select Virtual Storage.
6. In the Virtual Storage popup window, select ISO from the drop-down menu, and click Open Image. Browse local computers, and select the image you downloaded in step 1.
7. To mount the ISO image, click Plug In, and click OK.
8. On the console menu bar, click the Power Control, and select Power Reset.
9. The system will boot to the mounted image and the Loading ESXi installer screen will appear. When prompted, press Enter to continue.
10. To Accept the EULA and Continue, press F11.
11. Select the storage device to target for installation. We selected the internal SD card. To continue, press Enter.
12. To confirm the storage target, press Enter.
13. Select the keyboard layout, and press Enter.
14. Provide a root password, and confirm the password. To continue, press Enter.
15. To install, press F11.
16. Upon completion, reboot the server by pressing Enter.
17. Complete steps 1 through 16 for each server under test.

Creating a vSphere cluster in vCenter

1. Open a browser, and enter the address of the vCenter server you deployed. For example: `https://[vcenter.FQDN]/ui`
2. In the left panel, select the vCenter server, right-click, and select New Datacenter.
3. Provide a name for the new data center, and click OK.
4. Select the data center you just created, right-click, and select New Cluster.
5. Give a name to the cluster, and click OK.
6. In the cluster configuration panel, under Add hosts, click Add.
7. Check the box for Use the same credentials for all hosts. Enter the IP address and root credentials for the first host, and enter the IP addresses of all remaining hosts. Click Next.
8. Check the box beside Hostname/IP Address to select all hosts. Click OK.
9. Click Next.
10. Click Finish.

Creating a distributed vSwitch and port group

1. From vSphere client, click Home→Networking.
2. Select your Datastore.
3. On the right panel, in the Actions drop-down menu, select Distributed vSwitch→New Distributed vSwitch.
4. Give your vSwitch a name, or accept the default. Click Next.
5. Select 7.0.2 - ESXi 7.0.2 and later as the version, and click Next.
6. Select the number of uplinks per ESXi host you'll give to the vSwitch. We selected two. Click Next.
7. Click Finish.
8. Select new DSwitch, and click configure.
9. Select Properties, and click edit.
10. Under advanced, set MTU (Bytes) to 9000.
11. Click OK.
12. Select LACP, click + NEW, and set mode to active.
13. Click OK.
14. Enter appropriate name, set ports to two, and click OK.
15. Right-click the new DSwitch, and select Add and Manage Hosts.
16. Leave Add hosts selected, and click Next.
17. To add new hosts, click the +.
18. To select all the hosts in your target cluster, check the box for Host. Click OK. Click Next.
19. Select the NIC you want to use for this DSwitch, and click Assign Uplink.
20. Select the first LAG port.

21. Check the box for Apply this uplink assignment to the rest of the hosts. Click OK.
22. Complete steps 16 through 18 for the second NIC, and assign to the second LAG port.
23. Click Next.
24. Do not assign VMkernel adapters. Click Next.
25. Do not migrate any VM networking. Click Next.
26. Click Finish.
27. Right-click the DvSwitch, and select Distributed Port Group→New Distributed Port Group.
28. Name it `Workload Network`, and click Next.
29. Change the VLAN type to VLAN, and set the VLAN ID to VLAN 2. Click Next.
30. Click Finish.
31. Complete steps 17 through 20 two times on the AMD EPYC processor-based cluster using the following settings:
    - Name: vMotion; VLAN ID 20
    - Name: vSAN; VLAN ID 30
32. For the Intel cluster use the following port group settings:
    - Name: vMotion; VLAN ID 40
    - Name vSAN; VLAN ID 60

Creating a VMkernel adapter

Note: We completed these steps on each server under test for the private network, vSAN network, and vMotion network using the appropriate IPv4 settings for each network and enabling the corresponding service for each VMkernel adapter. We enabled the management service on the private network VMkernel and re-added the hosts to vCenter Server using the private IP.

1. From the vCenter UI in the Hosts and Clusters view, right-click a host, and click Add Networking.
2. Select VMkernel Network Adapter, and click Next.
3. Click Browse, and select existing network.
4. Select the desired Distributed Port Group you created in the previous section.
5. If necessary, assign a different network label.
6. Select the desired service or services for the VMkernel, and click Next.
7. Select use static IPv4 settings, and enter the desired IPv4 address, subnet mask, and default gateway.
8. Click Next.
9. Click Finish.

Configuring the VMware vSAN datastore

1. In the left panel of the vCenter, right-click the test cluster, select vSAN, and select configuration.
2. Select single site cluster, and click Next.
3. In the Services screen, click Next.
4. In the Claim disks screen, click the Group by drop-down menu, and select Host.
5. Select disks for each host. We selected one drive for cache and two drives for capacity. Click Next.
6. In the Review screen, click Finish.
7. Click the test cluster from vCenter, and in the right panel, click the Configure tab.
8. In the vSAN section, select Services.
9. To create a distributed switch, in the Configure Cluster section, click CONFIGURE.
10. In the Physical adapters section, choose two uplink adapters for the distributed switch, and click Next.
11. In the Storage traffic screen, set the VLAN ID assigned for vSAN, choose Static IPs, and fill in the static IP address, Subnet mask, and default gateway. Click Next.
12. In the Advanced options screen, click Next.
13. In the Claim disks screen, ensure that the 1.92TB drives are selected for cache and the 3.84TB drives are selected for capacity.
14. Click Next.
15. In the Proxy settings screen, click Next.
16. Click Finish.

Creating the gold test VM

1. In vCenter, select a host to create a VM.
2. Right-click, and select New Virtual Machine.
3. Click Create a new virtual machine, and click Next.
4. Enter an appropriate VM name, select Datacenter, and click Next.
5. Select an appropriate host, and click Next.
6. Choose an appropriate datastore, and click Next.
7. Select an appropriate compatibility mode (ESXi 7.0 U2 and later), and click Next.
8. For the Guest OS Family, select Linux. For the Guest OS Version, select Red Hat Enterprise Linux 8 (64-bit). Click Next.
9. In customize hardware, select the following:
    - CPU (32 vCPU for the AMD EPYC processor-based solution, and 20 vCPU for the Intel Xeon Scalable processor-based solution)
    - 112GB memory fully reserved
    - 60GB OS hard drive
    - 500GB data hard drive
    - Choose the appropriate VM Network
    - Point the DVD drive to the installation media ISO in your datastore
    - Click Next.
10. Review, and click Finish.

Installing Red Hat Enterprise Linux 8.6 on the gold client VM

1. Power on the VM.
2. Boot to the installation media attached to the VM.
3. Select Install Red Hat Enterprise Linux 8.6.
4. Select English, and click Continue.
5. For the Installation destination, select automatic partitioning, and click Done.
6. For the Software Selection, select Minimal Install, and click Done.
7. Configure a static IP address, and click Done.
8. Set a password for the root user account, and click Done.
9. Click Begin Installation

Configuring Red Hat Enterprise Linux 8 and installing MySQL on the gold test VM

1. Log into the MySQL instance via SSH.
2. In /etc/selinux/config, set `SELINUX=Permissive`.
3. Stop and disable the firewall:

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

4. Set the tuned profile to virtual-guest:

```
tuned-adm profile virtual-guest
```

5. Add the following lines to the end of /etc/sysctl.conf:

```
vm.swappiness = 1
fs.aio-max-nr = 1048576
vm.nr_hugepages = 49152
```

6. Install prerequisite tools:

```
yum -y install wget vim tar zip unzip lz4 pigz nmon sysstat numactl ksh screen
```

7. Create a directory for MySQL:

```
mkdir -p /mnt/mysqldata
```

8. Create an XFS filesystem on the data drive:

```
mkfs.xfs /dev/sdb
```

9. Mount the data drive to the MySQL directory:

```
mount -o defaults,nofail,x-systemd.device-timeout=5 /dev/sdb /mnt/mysqldata
```

10. Edit the /etc/fstab file, and add the following line to the end:

```
/dev/sdb /mnt/mysqldata xfs defaults,nofail,x-systemd.device-timeout=5 0 2
```

11. Download the appropriate MySQL bundle for either x86 or Arm architecture on Red Hat Enterprise Linux from https://dev.mysql.com/downloads/mysql/.
12. Extract the MySQL bundle:

```
tar -xf mysql-community-server-8.0.30-1.el8.x86_64.rpm
```

13. Install MySQL Community Server 8 and all dependencies:

```
sudo yum --disablerepo=* localinstall mysql-community-server-8.0.30-1.el8.x86_64.rpm
```

14. Stop the MySQL service:

```
sudo service mysqld stop
```

15. Copy the MySQL data directory to your data disk:

```
sudo cp -R -p /var/lib/mysql /mnt/mysqldata/
```

16. Start the MySQL service:

```
sudo service mysqld start
```

17. Log into the MySQL instance as the root user:

```
mysql -u root -p
```

18. Create a new user named mysql with full permissions:

```
CREATE USER 'mysql'@'localhost' IDENTIFIED BY '[password]';
GRANT ALL PRIVILEGES ON *.* TO 'mysql'@'localhost'
->  WITH GRANT OPTION;
CREATE USER 'mysql'@'%' IDENTIFIED BY '[password]';
GRANT ALL PRIVILEGES ON *.* TO 'mysql'@'%'
->  WITH GRANT OPTION;
```

19. Enable mysql_native_password authentication on the mysql user:

```
ALTER USER 'mysql'@'localhost' IDENTIFIED WITH mysql_native_password BY '[password]';
ALTER USER 'mysql'@'%' IDENTIFIED WITH mysql_native_password BY '[password]';
```

20. Edit the /etc/my.cnf file to match what we have in the Scripts we used for testing section.
21. Reboot the gold test VM.

```
Reboot
```

Creating the gold client VM

1. In vCenter, select a host to create a VM.
2. Right-click, and select New Virtual Machine.
3. Click Create a new virtual machine, and click Next.
4. Enter an appropriate VM name, select Datacenter, and click Next.
5. Select an appropriate host, and click Next.
6. Choose an appropriate datastore, and click Next.
7. Select an appropriate compatibility mode (ESXi 7.0 U2 and later), and click Next.
8. Select Linux for the Guest OS Family and CentOS 8 (64-bit) for the Guest OS Version. Click Next.
9. In customize hardware, select the following:
    - 6 vCPU
    - 18GB Memory fully reserved
    - 40GB OS hard drive
    - Choose appropriate VM Network
    - Point the DVD drive to the installation media ISO in your datastore
    - Click Next.
10. Review, and click Finish.

Installing CentOS 8 on the gold client VM

1. Power on the VM.
2. Boot to the installation media attached to the VM.
3. Select Install CentOS Linux 8.
4. Select English, and click Continue.
5. For the Installation destination, select automatic partitioning, and click Done.
6. For the Software Selection, select Minimal Install, and click Done.
7. Configure a static IP address, and click Done.
8. Set a password for the root user account, and click Done.
9. Click Begin Installation

Configuring CentOS 8 and installing HammerDB 4.4 on the gold client VM

1. SSH into the gold client VM.
2. Download HammerDB 4.4:

```
wget https://github.com/TPC-Council/HammerDB/releases/download/v4.4/HammerDB-4.4-Linux.tar.gz
```

3. Unzip HammerDB:

```
tar -xzf HammerDB-4.4-Linux.tar.gz
```

4. Install prerequisites:

```
dnf install -y epel-release
dnf install -y wget vim tar zip unzip lz4 pigz nmon sysstat numactl ksh screen
```

5. Install MySQL client:

```
dnf install -y https://dev.mysql.com/get/mysql80-community-release-el8-4.noarch.rpm
```

6. Place the run.tcl file from the Scripts we used for testing section into the HammerDB directory. Create a scripts directory:

```
mkdir ~/scripts
```

7. Change directories into the scripts directory:

```
cd ~/scripts
```

8. Create a bash script called run_hammerdb.sh with the contents from the Scripts we used for testing section.

Setting up password-less SSH on the VMs and hosts

1. Open two different terminals, one for the gold test VM and one for the gold client VM.
2. In each VM, generate an RSA keypair:

```
ssh-keygen
```

3. Accept the default storage location, and leave the password blank.
4. Cat out the public key on each VM:

```
cat ~/.ssh/id_rsa.pub
```

5. Copy the output, and paste it into the `~/.ssh/authorized_keys` file on the other VM.
6. Open two terminals, one for the first AMD EPYC processor-based cluster host and one for the first Intel Xeon Scalable processor-based host, and SSH into each.
7. Copy the public key from the gold test client VM into the `/etc/ssh/keys-root/authorized_keys` file on each host.
8. In `/etc/ssh/sshd_config`, make sure `PermitRootLogin yes` is set and that `ChallengeResponseAuthentication` and `PasswordAuthentication` are set to `no`.
9. Reload the SSH service:

```
/etc/init.d/SSH restart
```

10. Complete steps 6 through 9 three more times for the remaining hosts in the cluster.

Building the 1,000-warehouse TPROC-C schema on the gold test VM

1. On the gold client VM, change directories into the HammerDB directory.
2. Enter the HammerDB CLI:

```
./hammerdbcli
```

3. Set up the build parameters:

```
dbset db mysql
diset connection mysql_host [Hostname or IP Address]
diset tpcc mysql_count_ware 1000
diset tpcc mysql_num_vu 32
diset tpcc mysql_pass [Password]
diset tpcc mysql_partition true
```

4. Build the schema:

```
buildschema
```

Backing up the 1,000-warehouse schema

1. SSH into the gold test VM.
2. Stop the MySQL service:

```
systemctl stop mysqld
```

3. Create a compressed backup of the MySQL directory:

```
tar -cf- /mnt/mysqldata/mysql/ | pigz -9 -c > /mnt/mysqldata/backup/mysql_
tpcc_1000warehouses_mysql.tar.gz
```

4. Once the backup is complete, start the MySQL service:

```
systemctl start mysqld
```

5. Shut down the VM:

```
poweroff
```

Running the test

Prior to running the test, we cloned out the gold test VM and the gold client VM so that we had 32 AMD EPYC processor-based cluster test VMs, 32 Intel Xeon Scalable processor-based test VMs, and 32 client VMs from which to run the test. Our first client VM also doubled as our controller, and we ran all of our tests from scripts in our scripts directory on client1. Our run script sets up all the clients with the correct parameters, starts performance collection, runs the test, restores the database, and stops, collects, and parses the performance data.

1. SSH into client1.
2. Change directories into the scripts directory:

```
cd ~/scripts
```

3. For whichever cluster you are testing, run `./check_mysql_status.sh [amd | intel]`. The output should look like the following for all 32 VMs:

```
amd1: (running)
```

4. Once you've confirmed the VMs are all up and running MySQL, run the test.

```
./run.sh [amd | intel]
```

5. After the test is complete, wait for the restore to complete before rebooting the VMs and running again. You can run the /check_mysql_status.sh script to verify when the restore is finished.

We ran the test three times on each system, and use the median for reporting purposes.

Scripts we used in our testing

run.tcl

```
#!/bin/tclsh
puts "SETTING CONFIGURATION"
global complete
proc wait_to_complete {} {
global complete
set complete [vucomplete]
if {!$complete} { after 5000 wait_to_complete } else { exit }
}

dbset db mysql

diset connection mysql_host [Hostname or IP address]
diset tpcc mysql_count_ware 1000
diset tpcc mysql_pass [Password]
diset tpcc mysql_partititon true
diset tpcc mysql_driver timed
diset tpcc mysql_rampup 5
diset tpcc mysql_duration 10
diset tpcc mysql_num_vu 144

vuset logtotemp 1

loadscript

vuset vu 144
vuset showoutput 1
vucreate

vurun

wait_to_complete
vwait forever
```

my.cnf

```
[mysqld]

datadir=/mnt/mysqldata/mysql
default_authentication_plugin=mysql_native_password
socket=/mnt/mysqldata/mysql/mysqld.sock
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
port=3306
bind_address=0.0.0.0
large-pages

# general

max_connections=4000
table_open_cache=8000
table_open_cache_instances=16
back_log=1500
default_password_lifetime=0
ssl=0
performance_schema=OFF
max_prepared_stmt_count=128000
skip_log_bin=1
character_set_server=latin1
collation_server=latin1_swedish_ci
transaction_isolation=REPEATABLE-READ

# files
```

```
innodb_file_per_table
innodb_log_file_size=1024M
innodb_log_files_in_group=32 #scale
innodb_open_files=4000

# buffers
innodb_buffer_pool_size=90112M #scale
innodb_buffer_pool_instances=64
innodb_log_buffer_size=1024M

# tune

innodb_doublewrite=0
innodb_thread_concurrency=0
innodb_flush_log_at_trx_commit=0
innodb_max_dirty_pages_pct=90
innodb_max_dirty_pages_pct_lwm=10
join_buffer_size=32K
sort_buffer_size=32K
innodb_use_native_aio=1
innodb_stats_persistent=1
innodb_spin_wait_delay=6
innodb_max_purge_lag_delay=300000
innodb_max_purge_lag=0
innodb_flush_method=O_DIRECT_NO_FSYNC
innodb_checksum_algorithm=none
innodb_io_capacity=8000
innodb_io_capacity_max=16000
innodb_lru_scan_depth=9000
innodb_change_buffering=none
innodb_read_only=0
innodb_page_cleaners=4
innodb_undo_log_truncate=off

# perf special

innodb_adaptive_flushing=1
innodb_flush_neighbors=0
innodb_read_io_threads=16
innodb_write_io_threads=16
innodb_purge_threads=4
innodb_adaptive_hash_index=0

# monitoring

innodb_monitor_enable='%'

[client]

socket=/mnt/mysqldata/mysql/mysqld.sock
```

check_mysql_status.sh

```
#!/bin/bash

CPU=${1}

for i in {1..32};
do
      echo "${CPU}${i}: $(ssh ${CPU}${i} 'systemctl status mysqld' | grep Active: | awk '{ print $3 }')"

done
```

run_hammerdb.sh – called by run.sh on each client VM to run the test with the parameters from run.tcl

```
#!/bin/bash

cd HammerDB-4.4
./hammerdbcli auto run.tcl
```

run.sh

```
#!/bin/bash

CPU=${1}
RESULTS_DIR=/root/results
TIMESTAMP=$(date '+%Y%m%d_%H%M%S')
HDB_DIR=/root/HammerDB-4.4
SCRIPT_DIR=/root/scripts/
HDB_SCRIPT=run.tcl
HDB_RUN=${HDB_DIR}/${HDB_SCRIPT}
HDB_LOG=/tmp/hammerdb.log

#HammerDB run parameters
WAREHOUSE_COUNT=1000
RAMPUP=5  #  minutes
DURATION=10  #  minutes
VIRTUAL_USERS=144
TOTAL_ITERATIONS=10000000000

#Run length
WARMUP=$((RAMPUP*60))
RUNTIME=$((DURATION*60))
RUN_LENGTH=$(((WARMUP+RUNTIME)+60))

#nmon parameters
STEP=2  # seconds
SAMPLES_TOTAL=$(((WARMUP+RUNTIME)/STEP+5))

#Prepare HammerDB run script
if [[ ${CPU} == "amd" ]];
then
      for i in {1..32};
      do
        ssh client${i} "sed -i 's/_host.*/_host amd${i}/' ${HDB_RUN}"
        ssh client${i} "sed -i 's/_total_iterations.*/_total_iterations ${TOTAL_ITERATIONS}/' ${HDB_RUN}"
        ssh client${i} "sed -i 's/_count_ware.*/_count_ware ${WAREHOUSE_COUNT}/' ${HDB_RUN}"
        ssh client${i} "sed -i 's/_rampup.*/_rampup ${RAMPUP}/' ${HDB_RUN}"
        ssh client${i} "sed -i 's/_duration.*/_duration ${DURATION}/' ${HDB_RUN}"
        ssh client${i} "sed -i 's/_num_vu.*/_num_vu ${VIRTUAL_USERS}/' ${HDB_RUN}"
        ssh client${i} "sed -i 's/vuset vu .*/vuset vu ${VIRTUAL_USERS}/' ${HDB_RUN}"
      done
elif [[ ${CPU} == "intel" ]];
then
        for i in {1..32};
        do
                ssh client${i} "sed -i 's/_host.*/_host intel${i}/' ${HDB_RUN}"
        ssh client${i} "sed -i 's/_total_iterations.*/_total_iterations ${TOTAL_ITERATIONS}/' ${HDB_RUN}"
                ssh client${i} "sed -i 's/_count_ware.*/_count_ware ${WAREHOUSE_COUNT}/' ${HDB_RUN}"
                ssh client${i} "sed -i 's/_rampup.*/_rampup ${RAMPUP}/' ${HDB_RUN}"
                ssh client${i} "sed -i 's/_duration.*/_duration ${DURATION}/' ${HDB_RUN}"
        ssh client${i} "sed -i 's/_num_vu.*/_num_vu ${VIRTUAL_USERS}/' ${HDB_RUN}"
                ssh client${i} "sed -i 's/vuset vu .*/vuset vu ${VIRTUAL_USERS}/' ${HDB_RUN}"
        done
else
      echo "CPU type not set. Enter either amd or intel after run.sh."
      exit
fi
```

```
#Make results folder for run and copy hammerdb config files
mkdir -p ${RESULTS_DIR}/${CPU}/${TIMESTAMP}/client_configs
RUN_DIR=${RESULTS_DIR}/${CPU}/${TIMESTAMP}
CLIENT_CFG_DIR=${RESULTS_DIR}/${CPU}/${TIMESTAMP}/client_configs
mkdir -p ${CLIENT_CFG_DIR}/hammerdb
cp ${HDB_RUN} ${CLIENT_CFG_DIR}/hammerdb/client1.run.tcl

for i in {2..32};
do
      scp client${i}:${HDB_RUN} ${CLIENT_CFG_DIR}/hammerdb/client${i}.run.tcl
done

#Start performance monitoring on hosts and clients
if [[ ${CPU} == "amd" ]];
then
      for i in {1..32};
        do
          ssh amd${i} "killall -q -w nmon; sync; rm -f /tmp/amd${i}.nmon"
        done

        for i in {1..4};
        do
                ssh amd-host${i} "pkill nmon; rm -f vmfs/volumes/datastore-amd${i}/esxoutput/
amd${i}esxout.*"
        done

      for i in {1..4};
      do
        ssh amd-host${i} "vmfs/volumes/datastore-amd${i}/esxoutput/startesxtop.sh $((WARMUP+RUNTIME))" &
      done

      for i in {1..32};
      do
        ssh amd${i} "nmon -F /tmp/amd${i}.nmon -s${STEP} -c$((SAMPLES_TOTAL)) -J -t"
      done

elif [[ ${CPU} == "intel" ]];
then
      for i in {1..32};
        do
          ssh intel${i} "killall -q -w nmon; sync; rm -f /tmp/intel${i}.nmon"
        done

        for i in {1..4};
        do
          ssh intel-host${i} "pkill nmon; rm -f vmfs/volumes/datastore-intel${i}/esxoutput/
intel${i}esxout.*"
        done

        for i in {1..4};
        do
                ssh intel-host${i} "vmfs/volumes/datastore-intel${i}/esxoutput/startesxtop.sh
$((WARMUP+RUNTIME))" &
        done

      for i in {1..32};
        do
          ssh intel${i} "nmon -F /tmp/intel${i}.nmon -s${STEP} -c$((SAMPLES_TOTAL)) -J -t"
        done
fi


#Run Test
echo -e "\nRunning test for $((RAMPUP+DURATION)) minutes!"
rm -f ${HDB_LOG}

for i in {2..32};
do
      ssh client${i} "rm -f ${HDB_LOG}"
done

sleep 5
```

```
for i in {1..32};
do
      ssh client${i} "screen -d -m ${SCRIPT_DIR}/run_hammerdb.sh"
done

sleep ${RUN_LENGTH}


#Stop performance monitoring and copy files
mkdir -p ${RUN_DIR}/nmon
mkdir -p ${RUN_DIR}/esxtop

NMON_DIR=${RUN_DIR}/nmon
ESXTOP_DIR=${RUN_DIR}/esxtop

if [[ ${CPU} == "amd" ]];
then
      for i in {1..32};
        do
          ssh amd${i} "killall -q -w nmon; sync"
        done

        for i in {1..4};
        do
          ssh amd-host${i} "pkill nmon"
        done

        for i in {1..4};
        do
                scp amd-host${i}:/vmfs/volumes/datastore-amd${i}/esxoutput/amd${i}esxout* ${ESXTOP_DIR}
        done

      for i in {1..32};
        do
          scp amd${i}:/tmp/amd${i}.nmon ${NMON_DIR}
        done

elif [[ ${CPU} == "intel" ]];
then
        for i in {1..32};
        do
          ssh intel${i} "killall -q -w nmon; sync"
        done

        for i in {1..4};
        do
          ssh intel-host${i} "pkill nmon"
        done

      for i in {1..4};
        do
                scp intel-host${i}:/vmfs/volumes/datastore-intel${i}/esxoutput/intel${i}
esxout* ${ESXTOP_DIR}
        done

      for i in {1..32};
        do
          scp intel${i}:/tmp/intel${i}.nmon ${NMON_DIR}
        done
fi


#Copy HammerDB results
cp ${HDB_LOG} ${RUN_DIR}/client1-hammerdb.log

for i in {2..32};
do
      scp client${i}:${HDB_LOG} ${RUN_DIR}/client${i}-hammerdb.log
done
```

```
sleep 60

#Restore database after the run finishes
if [[ ${CPU} == "amd" ]];
then
        for i in {1..32};
      do
                ssh amd${i} "screen -d -m ~/restoreDB.sh"

        done
elif [[ ${CPU} == "intel" ]];
then
      for i in {1..32};
        do
          ssh intel${i} "screen -d -m ~/restoreDB.sh"

        done
fi

#Parse results and output run summary
for i in {1..32};
do
      cat ${RUN_DIR}/client${i}-hammerdb.log | grep TPM | awk '{ print $10 }' >> ${RUN_DIR}/TPM.txt
        cat ${RUN_DIR}/client${i}-hammerdb.log | grep NOPM | awk '{ print $7 }' >> ${RUN_DIR}/NOPM.txt
done

touch ${RUN_DIR}/run_summary.txt
SUMMARY=${RUN_DIR}/run_summary.txt

echo "HammerDB run $(date)" >> ${SUMMARY}
echo "" >> ${SUMMARY}
echo "" >> ${SUMMARY}

echo "Number of virtual users: ${VIRTUAL_USERS}" >> ${SUMMARY}
echo "Rampup (min): ${RAMPUP}" >> ${SUMMARY}
echo "Duration (min): ${DURATION}" >> ${SUMMARY}
echo "" >> ${SUMMARY}

TPM_TOTAL=0
NOPM_TOTAL=0
for i in $(cat ${RUN_DIR}/TPM.txt);
do
      TPM_TOTAL=$((${TPM_TOTAL}+${i}))
done

for i in $(cat ${RUN_DIR}/NOPM.txt);
do
        NOPM_TOTAL=$((${NOPM_TOTAL}+${i}))
done

for i in {1..32};
do
      echo "Client${i} TPM: $(cat ${RUN_DIR}/client${i}-hammerdb.log | grep TPM | awk '{ print $10
}')" >> ${SUMMARY}
done

echo "" >> ${SUMMARY}
echo "TPM total: ${TPM_TOTAL}" >> ${SUMMARY}
echo "" >> ${SUMMARY}

for i in {1..32};
do
        echo "Client${i} NOPM: $(cat ${RUN_DIR}/client${i}-hammerdb.log | grep NOPM | awk '{ print $7
}')" >> ${SUMMARY}
done

echo "" >> ${SUMMARY}
echo "NOPM total: ${NOPM_TOTAL}" >> ${SUMMARY}


if [[ ${CPU} == "amd" ]];
then
```

```
      for i in {1..32};
        do
          ~/nmonchart ${NMON_DIR}/amd${i}.nmon ${NMON_DIR}/amd${i}.html
        done
elif [[ ${CPU} == "intel" ]];
then
      for i in {1..32};
        do
          ~/nmonchart ${NMON_DIR}/intel${i}.nmon ${NMON_DIR}/intel${i}.html
      done
fi
```

Read the report at https://facts.pt/RXRk9yi

This project was commissioned by AMD.

Principled Technologies®

Facts matter.®