How We Got Here—the Prowess Approach to Analyzing Dell Technologies versus HPE Cloud Price/Performance

This document provides the methodology used by Prowess Consulting to evaluate cloud price versus performance data for Dell Technologies and HPE.

Methodology Summary

Prowess engineers took the following steps to test the Dell™ PowerEdge™ R650xs and the HPE® ProLiant® DL360 Gen10 Plus servers. This document provides the system configuration details and the step-by-step procedures that Prowess Consulting used to collect data. Testing was concluded in May 2022. We tested one of each server, highlighting the performance of each.

For the full analysis, read the report: https://www.prowesscorp.com/project/dell-servers-outperform-HPE-price-performance/.

Table 1: Configurations used in testing

System Configuration Info		
Model Name	HPE® ProLiant® DL360 Gen10 Plus	Dell™ PowerEdge™ R650xs
CPU	Intel® Xeon® Silver 4310 processor at 2.10 GHz	Intel® Xeon® Silver 4310 processor at 2.10 GHz
Number of CPUs	2	2
Cores/Threads per CPU	12/24	12/24
Cores/Threads Total	24/48	24/48
Frequency (Base/SCT/MCT)	2,100 MHz	2,100 MHz
Storage Controller 01	HPE® Smart Array P816i-a SR Gen10	Dell™ PowerEdge RAID 10 Controller 10 (PERC 10) H745 Front
Disk	HPE® Hynix® MK000480GWXFF	Samsung® MZ7KH480HAHQ0D3
Number of Disks	4	4
Installed Memory	128 GB	128 GB
Memory DIMM	Samsung® M393A2K43DB3-CWE	Samsung® M393A2K43DB3-CWE
Memory Speed	3,200 MHz	3,200 MHz
Number of Memory DIMMs	8	8
Operating System (OS)	Red Hat® Enterprise Linux® (RHEL)	Red Hat® Enterprise Linux® (RHEL)
OS Version	8.6	8.6
OS Kernel	4.18.0-372.9.1.el8.x86_64	4.18.0-372.9.1.el8.x86_64
Microsoft® SQL Server® Version	8.0.26	8.0.26
BIOS Version	1.60	1.6.5

Process

Our engineers installed Red Hat® Enterprise Linux® (RHEL) on the two servers under test using the following method.

Operating System Installation

1. Configure the following settings for the Red Hat installation:

- **Time and date: Pacific time zone**
- **Software selection: Server without a GUI**
- **System:** Select the install disk, set the volume group so that **Root** has majority of disk space, and then set swap to 16 GB.

2. Click **Install.**

MySQL® Installation and Testing

MySQL® was tested with HammerDB in the following method:

MySQL Installation and Setup

1. Enter the following command to install MySQL:

```
dnf install -y @mysql
```

2. After completing the installation, enable the MySQL service to auto-start on system start. Also start service manually for the first time:

```
sudo systemctl start mysqld

sudo systemctl enable --now mysqld.service

sudo systemctl start mysqld.service
```

3. Check the service's current status using the following command:

```
sudo systemctl status mysqld.service
```

4. Connect to MySQL as a root user. (Note that because we did not set up security, a password is not required.)

```
mysql -u root
```

5. Create a database with the following command:

```
CREATE DATABASE tpcc;
```

6. Create a new user with the following command:

```
CREATE USER '<user>'@'localhost' IDENTIFIED BY 'TestPassword1';
```

7. Grant permissions for the new user to the database with the following command:

```
GRANT ALL privileges ON *.* to '<user>'@'localhost';
```

8. Exit MySQL with the following command:

```
exit
```

9. Connect to the database with the new user using the following command:

```
mysql -u user -p
```

10. To list the current databases, enter the following command:

```
show databases;
```

11. Enter the following command to configure the MySQL client libraries path:

```
export LD_LIBRARY_PATH=/var/lib/mysql:$LD_LIBRARY_PATH
```

Install and Configure HammerDB

1. Download HammerDB to the system using the following command:

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

2. Unzip (untar) the file with the following command:

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

3. Build the HammerDB TPC-C® database:

   a. Enter the following command to change the directory to HammerDB-4.4:

```
cd HammerDB-4.4/config
```

   b. Edit the mysql.xml file with the following command:

```
nano mysql.xml
```

   c. Update the following entries:

```
mysql_socket: location to mysql.sock

mysql_user: user previously created

mysql_pass: password previously created
```

   d. Save and close the file.

Start HammerDB

1. Enter the following command to start HammerDB:

```
./hammerdbcli
```

2. Enter the following command to set the database type:

```
dbset db mysql
```

3. Verify the database type with the following command:

```
print db
```

4. Enter the following command to set the benchmark workload type:

```
dbset bm TPROC-C
```

5. Enter the following commands to set the settings to build the TPC-C database:

```
diset tpcc mysql_count_ware 310

diset tpcc mysql_num_vu 45

diset tpcc mysql_allwarehouse true
```

6. Enter the following command to build the TPC-C test database:

```
buildschema
```

7. Enter the following command to verify the status of the build:

```
vustatus
```

8. Enter the following to destroy the build's virtual users:

```
vudestroy
```

9. Exit out of the HammerDB command-line interface (CLI) or open a new Secure Shell (SSH) session.

10. Enter the following command to back up the TPC-C database:

```
mysqldump -u <user> tpcc > tpccbackup.sql
```

11. Create a test script with the following settings, and save it as **mysqltest.tcl:**

```
#!/bin/tclsh
proc runtimer { seconds } {
set x 0
set timerstop 0
while {!$timerstop} {
incr x
after 1000
if { ![ expr {$x % 60} ] } {
set y [ expr $x / 60 ]
puts "Timer: $y minutes elapsed"
}
update
if { [ vucomplete ] || $x eq $seconds } { set timerstop 1 }
}
return
}
puts "SETTING CONFIGURATION"
dbset db mysql
diset tpcc mysql_driver timed
diset tpcc mysql_rampup 5
diset tpcc mysql_duration 20
diset tpcc mysql_allwarehouse true
vuset vu 63
vuset iterations 1
vuset showoutput 1
vuset logtotemp 1
vuset timestamps 1
vuset unique 1
foreach z { 1 } {
puts "$z iteration"
loadscript
vucreate
vurun
runtimer 1500
vudestroy
after 1620
```

```
}
puts "TESTING COMPLETE"
```

12. Enter the following command to enter the HammerDB CLI:

```
./hammerdbcli
```

13. Run the source script three times, taking the median of the results:

```
source mysqltest.tcl
```

14. Delete and restore the TPC-C database between each run:

```
mysql -u <user> -p <password> tpcc < tpccbackup.sql
```

Virtual Machine Capacity Installation and Testing

For the purposes of this test, we used Red Hat Enterprise Linux virtualization.

1. Enter the following command to install and set up the Red Hat Enterprise Linux cockpit:

```
sudo yum install -y cockpit
rpm -qa | grep cockpit
```

2. Enter the following command to enable the cockpit service:

```
systemctl enable --now cockpit.socket
```

3. Add the cockpit service exception to the firewall:

```
sudo firewall-cmd --add-service=cockpit --permanent
sudo firewall-cmd --reload
```

4. Install virtual machine (VM) management with the following command:

```
sudo yum -y install cockpit-machines
```

5. Install virt-viewer with the following command:

```
sudo yum -y install virt-viewer
```

6. Install and configure Red Hat® Virtualization:
    a. Enter the following command to install Red Hat Virtualization:

```
sudo yum module install virt -y
sudo yum install virt-install virt-viewer -y
```

    b. Start the service with the following command:

```
systemctl start libvirtd
```

    c. Verify the service installed correctly, and then verify all services pass with the following command:

```
virt-host-validate
```

Note: If you receive a warning about an input/output memory management unit (IOMMU), run the following command and reboot:

```
sudo grubby --update-kernel=ALL --args="intel_iommu=on"
```

7. Create a CentOS® VM template with the following configuration:
    a. Log on to the web console at **https://100.67.96.210:9090.**
    b. Go to **Virtual Machines**, and then select **Create VM.**
    c. On the host, download CentOS with the following command:

```
wget https://mirrors.xtom.com/centos/7.9.2009/isos/x86_64/CentOS-7-x86_64-DVD-2009.iso
```

d. Using the web console, create a new VM with the following configuration:

- **vCPU:** 2 cores
- **Memory:** 8 GB
- **Storage:** 25 GB

e. Select **Virtual Machines**, and then click **Create VM.**

f. Enter a unique name.

g. For **Connection Type**, select **System**.

h. For the **Installation Type**, select **Local install media**.

i. Under **Installation Source**, select the path to the ISO that was downloaded earlier.

j. The operation system will be automatically chosen based on the ISO.

k. For **Storage**, leave it as **Create new volume**.

l. Set **Size** to **25GiB**.

m. Set **Memory** to **8GiB**.

8. Log on to the new VM, and then update and configure the VM.

a. Enter the following command to install updates on the VM:

```
sudo yum –y update && sudo yum –y upgrade
```

b. Install stress-ng with the following commands:

```
sudo yum –y install epel–release
sudo yum makecache
sudo yum –y install stress–ng
```

In order to simulate a workload, stress-ng will be utilized on the VMs:

9. Enter the following command to create a startup script:

```
vi /home/user/startup.sh
```

10. Copy the following to the startup.sh file:

```
#! /bin/bash
stress–ng --cpu 2 --io 2 --vm 4 --vm–bytes 1G --timeout 5
```

11. Enter the following command to set the script as executable:

```
sudo chmod +wx /home/user/startup.sh
```

12. Enter the following command to add a cron job to rerun the startup.sh script:

```
crontab –e
```

13. Enter the following into the cron file:

```
*5/ * * * * /home/user/startup.sh
```

14. Enter the following command to add an auto-startup service:

```
sudo vi /etc/systemd/system/testscript.service
```

15. Enter the following into the testscript.service file:

```
[Unit]
Description=stress test
```

```
After=network.target
[Service]
Type=simple
ExecStart=/home/user/startup.sh
TimeoutStartSec=0
[Install]
WantedBy=default.target
```

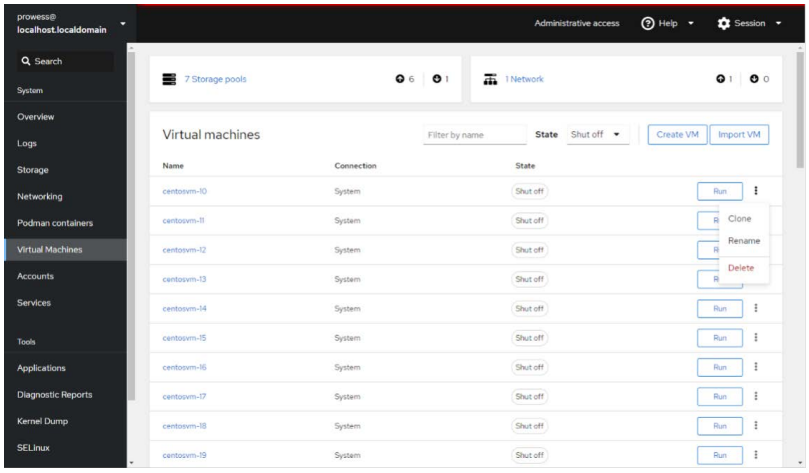
16. Enter the following command to modify the permissions on the new scripts:

```
sudo chmod +xw /etc/systemd/system/testscript.service
sudo chmod +xw /home/user/startup.sh
```

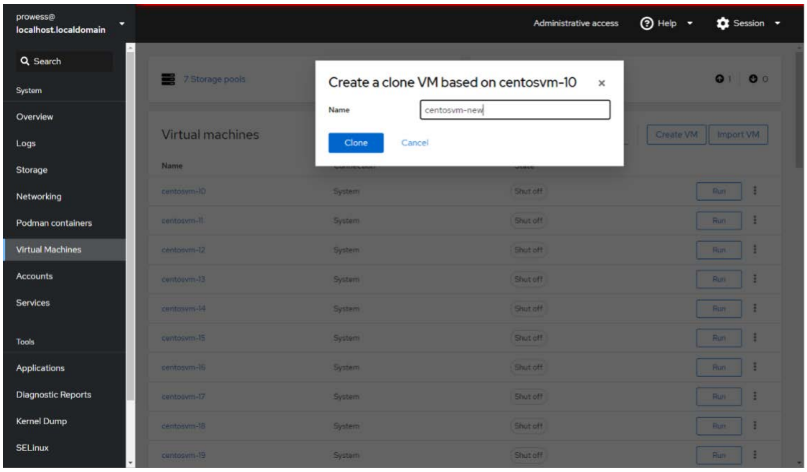
17. Enter the following command to configure the test script to startup automatically:

```
systemctl daemon-reload
systemctl start testscript.service
systemctl enable testscript.service
```

18. From the Red Hat cockpit web console, create 25 clones of the VM:
    a. Shut down the VM previously created.
    b. Select the ellipses, and then choose **Clone**.



    c. Enter a new name for the VM, and then click **Clone**.



    d. Repeat until all of the Vms have been created.

19. From the Red Hat host servers, create a script that will start four of the VMs at bootup and then start an additional VM every five minutes:

- `vi /home/user/vm-stress-gm.sh`

```
#!/bin/bash
#echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-1
echo -e $1 | sudo -S virsh start centosvm-2
echo -e $1 | sudo -S virsh start centosvm-3
echo -e $1 | sudo -S virsh start centosvm-4
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-5
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-6
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-7
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-8
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-9
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-10
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-11
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-12
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-13
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-14
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-15
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-16
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-17
echo `sleep 300`
```

```
echo -e $1 | sudo -S virsh start centosvm-18
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-19
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-20
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-21
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-22
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-23
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-24
echo `sleep 300`
echo -e $1 | sudo -S virsh start centosvm-25
echo `echo "Test Complete"`
```

20. Gather metrics utilizing ATOP and DSTAT on the host server. Max out the capacity of the servers with additional VMs.

    In the case of this test, we maxed out memory capability by starting new VMs until we received an alert that the server could no longer start any additional VMs. Therefore, we could no longer deploy any additional VM workloads.