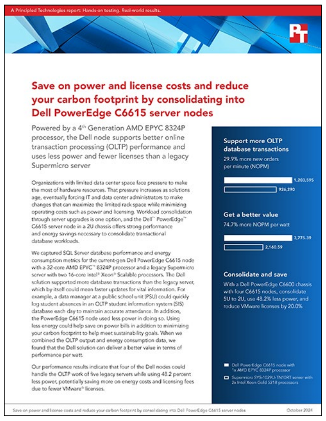

A Principled Technologies report: Hands-on testing. Real-world results.

Save on power and license costs and reduce your carbon footprint by consolidating into Dell PowerEdge C6615 server nodes

Powered by a 4th Generation AMD EPYC 8324P processor, the Dell node supports better online transaction processing (OLTP) performance and uses less power and fewer licenses than a legacy Supermicro server

Organizations with limited data center space face pressure to make the most of hardware resources. That pressure increases as solutions age, eventually forcing IT and data center administrators to make changes that can maximize the limited rack space while minimizing operating costs such as power and licensing. Workload consolidation through server upgrades is one option, and the Dell™ PowerEdge™ C6615 server node in a 2U chassis offers strong performance and energy savings necessary to consolidate transactional database workloads.

We captured SQL Server database performance and energy consumption metrics for the current-gen Dell PowerEdge C6615 node with a 32-core AMD EPYC™ 8324P processor and a legacy Supermicro server with two 16-core Intel® Xeon® Scalable processors. The Dell solution supported more database transactions than the legacy server, which by itself could mean faster updates for vital information. For example, a data manager at a public school unit (PSU) could quickly log student absences in an OLTP student information system (SIS) database each day to maintain accurate attendance. In addition, the PowerEdge C6615 node used less power in doing so. Using less energy could help save on power bills in addition to minimizing your carbon footprint to help meet sustainability goals. When we combined the OLTP output and energy consumption data, we found that the Dell solution can deliver a better value in terms of performance per watt.

Our performance results indicate that four of the Dell nodes could handle the OLTP work of five legacy servers while using 48.2 percent less power, potentially saving more on energy costs and licensing fees due to fewer VMware® licenses.

Support more OLTP database transactions

29.9% more new orders per minute (NOPM)

1,203,595

926,290

Get a better value

74.7% more NOPM per watt

3,775.39

2,160.59

Consolidate and save

With a Dell PowerEdge C6600 chassis with four C6615 nodes, consolidate 5U to 2U, use 48.2% less power, and reduce VMware licenses by 20.0%

Dell PowerEdge C6615 node with 1x AMD EPYC 8324P processor

Supermicro SYS-1029U-TN10RT server with 2x Intel Xeon Gold 5218 processors

Save on power and license costs and reduce your carbon footprint by consolidating into Dell PowerEdge C6615 server nodes October 2024

The science behind the report:

Save on power and license costs and reduce your carbon footprint by consolidating into Dell PowerEdge C6615 server nodes

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 Save on power and license costs and reduce your carbon footprint by consolidating into Dell PowerEdge C6615 server nodes.

We concluded our hands-on testing on July 26, 2024. 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 July 22, 2024 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins.
Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: Results of our testing.

	Dell™ PowerEdge™ C6615	Supermicro SYS-1029U-TN10RT
CPU utilization	91.8%	95.3%
NOPM	1,203,595	926,290
Power	318.80	428.72
Perf/watt	3,775.39	2,160.59

System configuration information

Table 2: Detailed information on the systems we tested.

System configuration information	Dell PowerEdge C6615	Supermicro SYS-1029U-TN10RT
BIOS name and version	Dell 1.3.3	Supermicro X11DPU 4.2
Operating system name and version/ build number	VMware® ESXi™ 8.0 U3 - 24022510	VMware ESXi 8.0 U3 - 24022510
Date of last OS updates/patches applied	7/23/24	7/22/24
Power management policy	Performance	Balanced
Processor		
Number of processors	1	2
Vendor and model	AMD EPYC™ 8324P	Intel® Xeon® Gold 5218 CPU
Core count (per processor)	32	16
Core frequency (GHz)	2.65	2.30
Stepping	2	7
Memory module(s)		
Total memory in system (GB)	576	512
Number of memory modules	6	16
Vendor and model	Micron® MTC40F204WS1RC56BB1	SK Hynix® HMAA4GR7AJR8N-XN
Size (GB)	96	32
Type	ECC DDR5	ECC DDR4
Speed (MHz)	5,600	3,200
Speed running in the server (MHz)	4,800	2,666
Storage controller		
Vendor and model	Direct-attached	Direct-attached
Cache size (GB)	N/A	N/A
Firmware version	N/A	N/A
Driver version	N/A	N/A
Local storage		
Number of drives	2	4
Drive vendor and model	Dell NVMe™ PM1743 RI E3.S	Samsung MZ-WLJ1T60 (PM1735)
Drive size (GB)	3.84	1.6
Drive information (speed, interface, type)	NVMe U.2 Gen4	NVMe U.2 Gen4
Network adapter		
Vendor and model	Broadcom® Adv. Quad 25Gb	Intel 10G 2P X520
Number and type of ports	4x 25GbBase-T	2x 10GBase-T

System configuration information	Dell PowerEdge C6615	Supermicro SYS-1029U-TN10RT
Cooling fans		
Vendor and model	Dell PowerEdge 60mm fan	Supermicro heavy duty fan
Number of cooling fans	8	8
Power supplies		
Vendor and model	07GN3CA05	Supermicro PWS-1K02A-1R
Number of power supplies	2	2
Wattage of each (W)	2,400	1,000

How we tested

We performed the following steps in a fully updated VMware vSphere® 8 environment. We used a separate VMware vSphere 8 host to run infrastructure and client VMs. To prevent bottlenecking, we attached the host to the Supermicro system under test through a 10GbE switch and attached the host to the Dell PowerEdge C6615 solution through a 25GbE switch.

Installing and configuring the base Ubuntu image

We first created a base VM that we later customized for our client and SQL Server VMs.

1. Log into the vCenter console.
2. Right-click the infrastructure host, and select New Virtual Machine.
3. In Select a creation type, select Create a new virtual machine, and click Next.
4. In Select a name and folder, name the VM `gold-VM`, and click Next.
5. In Select a compute resource, click Next to accept defaults.
6. In Select storage, choose the storage you set up for your client VMs, and click Next.
7. In Select compatibility, click Next to accept defaults.
8. In Select a guest OS, choose Linux → Ubuntu Linux (64-bit), and click Next.
9. Choose the following options for the new VM:
    - CPU: 8
    - RAM: 8 GB
    - Hard disk: 60 GB
    - Network
        - Network connection 1: Your network connection for the Internet (optional if your test network has a pre-configured gateway)
        - Network connection 2: Your test network connection
10. Verify that you chose the correct options, and click Next.
11. In Ready to complete, verify that you've applied the correct details, and click Next.
12. Right-click your new VM, and select Power → Power On.
13. Open a console to the new VM.
14. Attach an Ubuntu 22.04 LTS ISO to the VM, and accept defaults (with the following exceptions):
    - Select the minimal installation option.
    - Configure a static IP for both network connections.
    - Select the username and password you prefer (we chose `user` as our username).
15. Once the OS has booted, log into the console for the system, and run a package update for the OS:

```
sudo apt-get update
sudo apt-get upgrade
```

16. Install Open VM tools:

```
sudo apt-get install -y open-vm-tools
```

17. Add the Microsoft SQL Server package repositories to your OS:

```
curl https://packages.microsoft.com/keys/microsoft.asc | sudo tee /etc/apt/trusted.
gpg.d/microsoft.asc
curl -fsSL https://packages.microsoft.com/config/ubuntu/22.04/mssql-server-2022.list | sudo tee /etc/
apt/sources.list.d/mssql-server-2022.list
curl https://packages.microsoft.com/config/ubuntu/22.04/prod.list | sudo tee /etc/apt/sources.list.d/
mssql-release.list
sudo apt-get update
```

18. Install the tools you will use later:

```
sudo apt-get install -y wget curl vim tar zip unzip nmon sysstat numactl ksh mssql-
tools18 unixodbc-dev
```

19. Add the SQL Server tools to the console path:

```
echo 'export PATH="$PATH:/opt/mssql-tools18/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools18/bin"' >> ~/.bashrc
```

20. Power down the base VM down:

```
sudo shutdown now
```

Installing and configuring the HammerDB Client VM on the Ubuntu base image

1. Clone your base VM into the infrastructure server, and when prompted, name it `client-VM-base`.
2. Right-click the client VM, and select Power → Power On.
3. Open a console to the client VM.
4. Once the OS has booted, log in, and change the IP address:

```
sudo vi 00-installer-config.yaml
```

5. In the yaml file, change the IP addresses of the VM, and save your changes.
6. Back in the console, apply the IP changes:

```
sudo netplan apply
```

7. Change the client VM hostname:

```
sudo hostnamectl set-hostname client-gold.mssql.test
```

8. Download and untar HammerDB:

```
wget https://github.com/TPC-Council/HammerDB/releases/download/v4.11/HammerDB-4.11-Linux.tar.gz
tar -zxvf HammerDB-4.11-Linux.tar.gz
```

Installing and configuring Microsoft SQL Server on the Ubuntu base image

1. Clone the base image onto the system under test host, and name it `sql-base`.
2. Right-click the SQL base image, and select Edit Settings.
3. In the Edit Settings window, make the following changes:
   - Change the RAM from 8GB to 64 GB.
   - Add a new thick provision eager zeroed 100GB disk for the database and backup.
   - Add a new thick provision eager zeroed 60GB disk for the log.
4. Click OK.
5. Right-click the SQL VM, and select Power → Power On.
6. Open a console to the SQL VM.
7. Once the OS has booted, log in and change the IP address:

```
sudo vi 00-installer-config.yaml
```

8. In the yaml file, change the IP addresses of the VM, and save your changes.
9. Back in the console, apply the IP changes:

```
sudo netplan apply
```

10. Change the client VM hostname:

```
sudo hostnamectl set-hostname sql-gold.mssql.test
```

11. Download the SQL Server 2022 packages to the system:

```
sudo apt-get install -y mssql-server
```

12. Create the database and log directories:

```
sudo mkdir -p /data
sudo mkdir -p /logs/log
```

13. Format the database and log disks:

```
sudo mkfs.xfs -f /dev/sdb
sudo mkfs.xfs -f /dev/sdc
```

14. Open the filesystem table, and make the database and log disks automatically attach to the OS on boot:

```
sudo vim /etc/fstab
```

15. In the filesystem table, add the following lines to the end:

```
/dev/sdb /data xfs rw,attr2,noatime 0 0
/dev/sdc /logs/log xfs rw,attr2,noatime 0 0
```

16. Apply the filesystem table to mount the database and log drives (this also provides an opportunity to ensure that you edited your table correctly. If you did not, these commands will not work. Check `/etc/fstab` and ensure you entered your disk information correctly):

```
sudo mount -v /data
sudo mount -v /logs/log/
```

17. Add new directories in the database drive for the database and backup, and add all permissions to the database, backup, and log directories:

```
sudo mkdir -p /data/db
sudo mkdir -p /data/backup
sudo chmod -R 777 /data
sudo chmod -R 777 /logs
```

18. Start the SQL Server database installation, and accept defaults (we chose `Password1` as our system administrator password):

```
sudo /opt/mssql/bin/mssql-conf setup
```

19. (Optional) Verify that SQL is installed:

```
systemctl status mssql-server --no-pager
```

20. For performance purposes, edit the SQL Server limits file:

```
sudo vim /etc/security/limits.d/99-mssql-server.conf
```

21. Add the following lines to the end of the file, and save:

```
mssql hard nofile 32727
mssql soft nofile 16000
```

22. Add the following tuning options to SQL:

```
sudo /opt/mssql/bin/mssql-conf traceflag 3979 on
sudo /opt/mssql/bin/mssql-conf set control.writethrough 1
sudo /opt/mssql/bin/mssql-conf set control.alternatewritethrough 0
```

23. Install Tuned:

```
sudo apt install tuned
```

24. Edit the Tuned settings for SQL:

```
sudo vim /usr/lib/tuned/mssql/tuned.conf
```

25. Ensure the SQL Tuned file looks like this:

```
#
# tuned configuration
#

[main]
summary=Optimize for MS SQL Server
include=throughput-performance

[cpu]
force_latency=5

[vm]
transparent_hugepage.defrag=always

[sysctl]
vm.swappiness = 1
vm.dirty_background_ratio = 3
vm.dirty_ratio = 80
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100
vm.transparent_hugepages=always
vm.max_map_count=1600000
net.core.rmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_default = 262144
net.core.wmem_max = 1048576
kernel.numa_balancing=0
```

26. Make the SQL Tuned file executable:

```
sudo chmod +x /usr/lib/tuned/mssql/tuned.conf
```

27. Change the Tuned profile to mssql:

```
sudo tuned-adm profile mssql
```

28. Reboot the server:

```
sudo reboot now
```

Initializing the HammerDB TPROC-C database and backing up the database

We used a combination of Microsoft SQL Server Management Studio (SMSS) and the Ubuntu command line to perform the following steps.

1. Open SMSS, and connect to the SQL Server installation.
2. In SMSS, right-click Databases, and select New Database.
3. In the New Database window, name your database tpcc, and give the database four database files and one log file. Ensure the database files are in `/data/db/`, and the log file is in `/logs/log/`.
4. In the Options tab, change the Recovery mode to Simple, and click OK.
5. Log into the client VM via SSH, and navigate to the HammerDB directory:

```
cd HammerDB-4.11
```

6. Open the HammerDB cli:

```
./hammerdbcli
```

7. Inside the HammerDB cli, set the benchmark to Microsoft SQL and TPROC-C:

```
dbset db mssqls
dbset bm tpcc
```

8. Configure the connection details for the SQL server:

```
diset connection mssqls_linux_server 192.168.0.20
diset connection mssqls_uid sa
diset connection mssqls_pass Password1
diset connection mssqls_trust_server_cert true
```

9. Configure the size of the database and the number of users to create the database:

```
diset tpcc mssqls_count_ware 500
diset tpcc mssqls_num_vu 4
```

10. Change the HammerDB client so it doesn't use locally cached data when initializing (Note: this is for compatibility purposes with our setup. If you have configured your setup to use locally cached data, you can leave this at defaults):

```
diset tpcc mssqls_use_bcp false
```

11. Start the database creation:

```
buildschema
```

12. While the database is initializing, create a TPROC-C automation file for HammerDB:

```
vi tproc-c.tcl
```

13. Inside the tproc-c.tcl, add the following lines:

```
dbset db mssqls
dbset bm TPC-C

diset connection mssqls_linux_server 192.168.0.21
diset connection mssqls_linux_authent sql
diset connection mssqls_uid sa
diset connection mssqls_pass Password1

diset tpcc mssqls_count_ware 500
diset tpcc mssqls_use_bcp false
diset tpcc mssqls_total_iterations 1000000000
diset tpcc mssqls_driver timed
diset tpcc mssqls_rampup 10
diset tpcc mssqls_duration 20
diset tpcc mssqls_allwarehouse false

loadscript
puts "TEST STARTED"
vuset vu 16
vuset logtotemp 1
vucreate
tcstart
tcstatus
set jobid [ vurun ]
vudestroy
tcstop
puts "TEST COMPLETE"
```

14. After you create the database, use SSMS to log into the SQL server.
15. Right-click the tpcc database, and select Properties.
16. In the Files tab of Database Properties, click the log file, and expand it to 50GB.
17. In the Options tab of Database Properties, change Recovery model to full, and click OK.
18. Right-click the tpcc database, and select Tasks → Back Up.
19. In the Back Up Database window, change the destination to `/data/backup/backup.bak`.
20. In the Backup Options tab of the Back Up Database window, set backup compression to Compress backup, and click OK.
21. After a few minutes, the database backup will complete.
22. Shut down the HammerDB client, and SQL VMs.

Cloning and configuring the HammerDB client VMs

1. Right-click the gold client VM, and select Clone → Clone to Virtual Machine.
2. In Select a name and folder, enter the name for the first HammerDB client, and click Next.
3. In Select a compute resource, select your infrastructure host, and click Next.
4. In Select storage, select your client VM storage, and click Next.
5. In Select clone options, accept defaults, and click Next.
6. In Ready to complete, verify your options, and click Finish.
7. When the VM has finished cloning, log into it, and make the IP address of the VM unique:

```
sudo vi /etc/netplan/00-installer-config.yaml
```

8. Apply the networking changes:

```
sudo netplan apply
```

9. Change the VM hostname:

```
sudo hostnamectl set-hostname hammerdb-01.mssql.test
```

10. Edit the HammerDB automation file to target the correct SQL server:

```
vi HammerDB-4.11/tproc-c.tcl
```

11. Complete steps 1 through 10 for all remaining clients.

Cloning and configuring the SQL Server host VMs

1. Right-click the gold SQL VM, and select Clone → Clone to Virtual Machine.
2. In Select a name and folder, enter the name for the first SQL Server VM, and click Next.
3. In Select a compute resource, select your system under test host, and click Next.
4. In Select storage, select Configure per Disk.
5. Clone the configuration file, operating system, and log disks to one drive, and clone the database disk to a second drive. Keep track of which drives you used—when cloning new VMs, we recommend spreading out the OS, log, and database drives to reduce resource contention. Ensure the log and database drives are thick provision eager zeroed, and click Next.
6. In Select clone options, accept defaults, and click Next.
7. In Ready to complete, verify your options, and click Finish.
8. When the VM has finished cloning, log into it, and make the IP address of the VM unique:

```
sudo vi /etc/netplan/00-installer-config.yaml
```

9. Change the VM hostname:

```
sudo hostnamectl set-hostname mssql-06.mssql.test
```

10. Complete steps 1 through 9 for all remaining SQL Server VMs.

Performing a test run

1. On all HammerDB client VMs, navigate to the HammerDB directory:

```
cd HammerDB-4.11
```

2. On all HammerDB client VMs, type the following command but do not execute it:

```
./hammerdbcli auto ~/HammerDB-4.11/tpcroc-c.tcl
```

3. On all HammerDB client VMs, simultaneously execute the command from step 2.

Performing a restore after a test run

We performed the following steps on all database VMs after completing a test run.

1. Reboot the database VM.
2. Use SSMS to log into the VM database.
3. Right-click the tpcc database, and select Delete.
4. In Delete Object, check Close existing connections, and click OK.
5. Right-click Databases, and select Restore Files and Filegroups.
6. In Destination to restore, type `tpcc`.
7. In Source for restore, select From Device.
8. In Select backup devices, navigate to `/data/backup/backup.bak`, and click OK.
9. Verify that the system is restoring the files to their appropriate locations, and click OK.
10. When the database has restored, click OK.
11. Complete steps 1 through 10 for the remaining database VMs.

Read the report at https://facts.pt/e4rvBfP ▶

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®