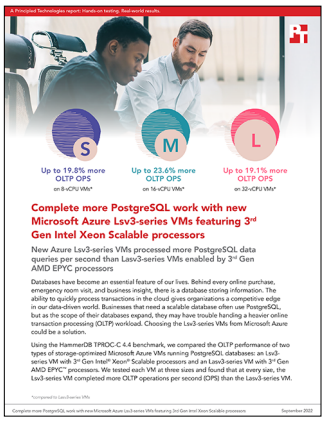

The science behind the report:

Complete more PostgreSQL work with new Microsoft Azure Lsv3-series VMs featuring 3rd Gen Intel Xeon Scalable 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 Complete more PostgreSQL work with new Microsoft Azure Lsv3-series VMs featuring 3rd Gen Intel Xeon Scalable processors.

We concluded our hands-on testing on July 29, 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 July 29, 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.

VM type	Operations per second	Lsv3-series advantage
L8s_v3 VMs	845,366	N/A
L8as_v3 VMs	705,485	19.82%
L16s_v3 VMs	1,715,299	N/A
L16as_v3 VMs	1,386,734	23.69%
L32s_v3 VMs	2,722,690	N/A
L32as_v3 VMs	2,285,972	19.10%

System configuration information

Table 2: Detailed information on the systems we tested.

System configuration information	Standard_Lsv3 Intel Ice Lake 8 / 16 / 32 vCPU	Standard_Lasv3 AMD Milan 8 / 16 / 32 / vCPU
Tested by	Principled Technologies	Principled Technologies
Test date	7/29/2022	7/29/2022
CSP / Region	Microsoft Azure – South Central US	Microsoft Azure – South Central US
Workload & version	HammerDB TPROC-C 4.4 PostgreSQL 14.4	HammerDB TPROC-C 4.4 PostgreSQL 14.4
WL specific parameters	800 / 1600 / 3200 warehouses, 16 / 32 / 64 virtual users, 28672 / 57344 / 114688 huge pages	800 / 1600 / 3200 warehouses, 16 / 32 / 64 virtual users, 28672 / 57344 / 114688 huge pages
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_L8sv3, Standard_L16sv3, Standard_L32sv3	Standard_L8sv3, Standard_L16sv3, Standard_L32sv3
BIOS name and version	Hyper-V UEFI v4.1 5/9/2022	Hyper-V UEFI v4.1 5/9/2022
Operating system name and version/build number	RHEL 9.0 5.14.0-70.17.1.el9_0.x86_64	RHEL 9.0 5.14.0-70.17.1.el9_0.x86_64
Date of last OS updates/patches applied	7/29/2022	7/29/2022
Processor		
Number of processors	1	1
Vendor and model	Intel® Xeon® Platinum 8370C CPU @ 2.80GHz	AMD EPYC™ 7763v @ 2.45 GHz
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6,106,6	B1
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	8 / 16 / 32	8 / 16 / 32
Memory module(s)		
Total memory in system (GB)	64 / 128 / 256	64 / 128 / 256
NVMe memory present?	No	No
Total DDR memory (GB)	64 / 128 / 256	64 / 128 / 256
General HW		
Storage: NW or direct att / instance	Direct att / instance	Direct att / instance

System configuration information	Standard_Lsv3 Intel Ice Lake 8 / 16 / 32 vCPU	Standard_Lasv3 AMD Milan 8 / 16 / 32 / vCPU
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information (speed, interface, type)	PremiumSSD	PremiumSSD
Data drive		
Number of drives	1 / 2 / 4	1 / 2 / 4
Drive size	1.92 TB	1.92 TB
Drive information (speed, interface, type)	Direct-attached NVMe	Direct-attached NVMe

How we tested

Creating the test and client VM

This section contains the steps we took to create the test VM hosting the PostgreSQL database and the client VM for remotely running the HammerDB benchmark client software.

Creating the test instance

1. Log into the Azure Portal, and navigate to Virtual Machines.
2. Click Create, and then click Virtual Machine.
3. Enter the subscription and resource group names, and give the virtual machine a name.
4. Choose the geographical region for the instance. In our case we used South Central US.
5. Select the RHEL 9.0 image.
6. Select the instance size. We tested the following sizes:
    a. Standard_L8s_v3
    b. Standard_L16s_v3
    c. Standard_L32s_v3
    d. Standard_L8as_v3
    e. Standard_L16as_v3
    f. Standard_L32as_v3
7. Select SSH public key, and generate a new key pair for the desired username.
8. Select Allow selected ports, and ensure SSH is enabled.
9. Click Disks.
10. Click Add a new disk.
11. Select a 2TB disk for database backups. We used the P50 tier Premium SSD.
12. Click Save.
13. Click Review + create.
14. Click Create.

Creating the HammerDB 4.4 client instance

1. Log into the Azure Portal, and navigate to the Virtual Machines.
2. Click Create, and then click Virtual Machine.
3. Enter the subscription and resource group names, and give the virtual machine a name.
4. Choose the geographical region for the instance. In our case we used South Central US.
5. Select the RHEL 9.0 image.
6. Select the Standard_D32ds_v4 instance size.
7. Select SSH public key, and generate a new key pair for the desired username.
8. Select Allow selected ports, and ensure SSH is enabled.
9. Click Review + create.
10. Click Create.

Configuring Red Hat Enterprise Linux 9 and installing PostgreSQL

1. Log into the PostgreSQL instance via ssh.
2. Run the host preparation script from the Scripts section:

```
sudo ./host_prepare.sh
```

3. Make directories for the database data and backups:

```
sudo mkdir /backups
sudo mkdir /pgmnt
```

4. Create a filesystem on the backup drive:

```
sudo mkfs.xfs -f /dev/<backup drive>
```

5. If testing a size with multiple NVMe drives, create a striped logical volume:

```
sudo pvcreate /dev/nvme0n1 /dev/nvme1n1 /dev/nvm2n1 /dev/nvme3n1
sudo vgcreate volgroup01 /dev/nvme0n1 /dev/nvme1n1 /dev/nvm2n1 /dev/nvme3n1
sudo lvcreate -i 2 -l 100%FREE -I 64 -n striped_logical_volume volgroup01
```

6. Create a filesystem on the logical volume (use /dev/nvme0n1 if single data drive):

```
sudo mkfs.xfs -f /dev/volgroup01/striped_logical_volume
```

7. Mount the backup and data drives to their respective locations:

```
sudo mount /dev/<backup drive> /backups
sudo mount /dev/volgroup01/striped_logical_volume /pgmnt
```

8. Create the data directory:

```
sudo mkdir /pgmnt/data
```

9. Install the PostgreSQL repository:

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/pgdg-redhat-
repo-latest.noarch.rpm
```

10. Install PostgreSQL:

```
sudo dnf install -y postgresql14-server
```

11. Give the postgres user a password:

```
sudo passwd postgres
```

12. Change to the postgres user, initialize the database, and exit:

```
su - postgres
/usr/pgsql-14/bin/postgresql-14-setup initdb -D /pgmnt/data
exit
```

Configuring the client instance

1. Log into to the client instance via ssh.
2. Disable SELINUX.

```
sudo sed -I 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
sudo setenforce 0
```

3. Install the PostgreSQL repository:

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

4. Install PostgreSQL:

```
sudo dnf install -y postgresql14-server
```

5. Download HammerDB 4.4.

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

6. Extract the HammerDB package.

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

Creating the database schema with HammerDB

1. Log into to the client instance via ssh.
2. Navigate to the HammerDB directory.

```
cd HammerDB-4.4
```

3. Start the database build using the appropriate pg-build-*.tcl script:

```
./hammerdbcli auto pg-build-*vCPU.tcl
```

Backing up the database

1. Log into to the PostgreSQL instance.
2. Change to the postgres user, stop the database, and exit:

```
su - postgres
/usr/pgsql-14/bin/pg_ctl -D /pgmnt/data stop
exit
```

3. Backup the database:

```
sudo tar -cf- /pgmnt/data/ | pigz -9 -c > /backups/pg*vCPU_backup.tar.gz
```

Running the tests

In this section, we list the steps to run the HammerDB TPROC-C test on the instances under test.

1. Log into to the PostgreSQL instance via ssh.
2. Make sure PostgreSQL is stopped, and delete the database:

```
sudo rm -rf /pgmnt/data
```

3. Unzip the backup database:

```
cd /pgmnt
sudo tar -xvf /backups/pg*vCPU_backup.tar.gz
```

4. Copy the appropriate pgconf_*vCPU.conf file to the data directory, replacing the original configuration file:

```
sudo cp pgconf_*vCPU.conf /pgmnt/data/postgresql.conf
```

5. Reboot the instance.
6. Change to the postgres user, start the database, and exit:

```
su - postgres
/usr/pgsql-14/bin/pg_ctl -D /pgmnt/data start
exit
```

7. Log into to the client instance via ssh.
8. Navigate to the HammerDB directory.

```
cd HammerDB-4.4
```

9. Run the test using the TCL file from the Scripts section.

```
./hammerdbcli auto pg-run-*vCPU.tcl
```

10. Repeat these steps 2 more times for a total of 3 runs.

Scripts

In this section, we print the scripts used in the methodology steps above.

host_prepare.sh (adjust Huge Pages for instance size):

```
#!/bin/bash

yum update -y
yum upgrade -y
setenforce 0
sed -i 's/SELINUX=.*/SELINUX=Permissive/' /etc/selinux/config
systemctl disable --now firewalld
#### System tuning ####
tuned-adm profile virtual-guest

sed -i \
-e '/vm.swappiness/d' \
-e '/fs.aio-max-nr/d' \
-e '/vm.nr_hugepages/d' \
/etc/sysctl.conf
cat <<EOF >>/etc/sysctl.conf
vm.swappiness = 1
fs.aio-max-nr = 1048576
vm.nr_hugepages = <Huge Pages>
EOF
sysctl -p
```

pgconf_8vCPU.conf:

```
listen_addresses = '*'
port = 5432
max_connections = 256
shared_buffers = 51200MB
huge_pages = on
temp_buffers = 2048MB
work_mem = 2048MB
maintenance_work_mem = 1024MB
autovacuum_work_mem = -1
max_stack_depth = 5MB
dynamic_shared_memory_type = posix
max_files_per_process = 4000
effective_io_concurrency = 32
wal_level = minimal
synchronous_commit = off
wal_buffers = 1024MB
checkpoint_timeout = 1h
max_wal_size = 1GB
min_wal_size = 80MB
checkpoint_completion_target = 1
checkpoint_warning = 0
max_wal_senders = 0
log_destination = 'stderr'
logging_collector = on
log_directory = '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_line_prefix = '%m [%p] '
log_timezone = 'America/New_York'
autovacuum = off
datestyle = 'iso, mdy'
timezone = 'America/New_York'
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
```

pgconf_16vCPU.conf:

```
listen_addresses = '*'
port = 5432
max_connections = 512
shared_buffers = 102400MB
huge_pages = on
temp_buffers = 4096MB
work_mem = 4096MB
maintenance_work_mem = 1024MB
autovacuum_work_mem = -1
max_stack_depth = 6MB
dynamic_shared_memory_type = posix
max_files_per_process = 4000
effective_io_concurrency = 32
wal_level = minimal
synchronous_commit = off
wal_buffers = 768MB
checkpoint_timeout = 1h
max_wal_size = 2GB
min_wal_size = 80MB
checkpoint_completion_target = 1
checkpoint_warning = 0
max_wal_senders = 0
log_destination = 'stderr'
logging_collector = on
log_directory = '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_line_prefix = '%m [%p] '
log_timezone = 'America/New_York'
autovacuum = off
datestyle = 'iso, mdy'
timezone = 'America/New_York'
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
```

pgconf_32vCPU.conf:

```
listen_addresses = '*'
port = 5432
max_connections = 1000
shared_buffers = 204800MB
huge_pages = on
temp_buffers = 4096MB
work_mem = 4096MB
maintenance_work_mem = 1024MB
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
max_wal_size = 5GB
min_wal_size = 80MB
checkpoint_completion_target = 1
checkpoint_warning = 0
max_wal_senders = 0
effective_cache_size = 128GB
log_destination = 'stderr'
logging_collector = on
log_directory = '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_line_prefix = '%m [%p] '
log_timezone = 'America/New_York'
autovacuum = off
datestyle = 'iso, mdy'
timezone = 'America/New_York'
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
```

pg-build-*vCPU.tcl:

```
dbset db pg
diset tpcc pg_count_ware <800, 1600, 3200>
diset tpcc pg_num_vu 32
diset connection pg_host <Target Instance IP>
buildschema
waittocomplete
```

pg-run-*vCPU.tcl:

```
dbset db pg
diset tpcc pg_driver timed
diset tpcc pg_rampup 2
diset tpcc pg_duration 5
diset connection pg_host <Target Instance IP>
diset tpcc pg_count_ware <800, 1600, 3200>
diset tpcc pg_total_iterations 10000000
loadscript
vuset vu <16, 32, 64>
vuset logtotemp 1
vucreate
vurun
runtimer 500
vudestroy
```

Read the report at https://facts.pt/7fU7cwD ▶

This project was commissioned by Intel.

Principled Technologies®

Facts matter.®