

The science behind the report:

Eadsv7 and Famdsv7 Azure VMs delivered higher SQL Server performance and value than Eadsv5 VMs

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 Eadsv7 and Famdsv7 Azure VMs delivered higher SQL Server performance and value than Eadsv5 VMs.

We concluded our hands-on testing on October 6, 2026. 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 October 6, 2026 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 https://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 TPROC-H Power (VU1) testing. Time to complete (sec). Lower is better. Source: PT.

TPCH	Run 1	Run 2	Run 3	Median	% less time
E16ads_v5	1,412	1,344	2,099	1,412	-
E16ads_v7	869	1,053	830	869	38.45%
F16-8amds_v7	759	703	1,074	759	46.24%
E64ads_v5	278	273	270	273	-
E64ads_v7	200	299	219	219	19.78%
F64-32amds_v7	197	203	200	200	26.73%

Table 2: Results of our TPROC-H throughput (VU7) testing. Time to complete (sec). Lower is better. Source: PT.

TPCH	Run 1	Run 2	Run 3	Median	% less time
E16ads_v5	15,477	15,523	16,469	15,523	-
E16ads_v7	5,796	5,685	5,585	5,685	63.37%
F16-8amds_v7	4,134	4,139	4,219	4,139	73.33%
E64ads_v5	1,009	1,055	960	1,009	-
E64ads_v7	681	711	728	711	29.53%
F64-32amds_v7	1,057	946	905	946	6.24%

Table 3: Results of our TPC-E-like testing. Transactions per second (TPS). Higher is better. Source: PT.

2x 20,000 customers	Run 1	Run 2	Run 3	Median	% more TPS
E16ads_v5	1,883	1,890	1,875	1,883	-
E16ads_v7	2,994	3,026	3,051	3,026	60.73%
F16-8amds_v7	2,329	2,353	2,371	2,353	24.98%
E64ads_v5	4,363	4,309	3,960	4,309	-
E64ads_v7	4,718	4,769	4,903	4,769	10.69%
F64-32amds_v7	4,392	4,384	4,369	4,384	1.74%

System configuration information

Table 4: Detailed information on the systems we tested.

System configuration information	
General information	
Date testing ended	10/6/2026
Cloud service provider (CSP)	Microsoft Azure
Region	East US
Workload information	
Workload name and version	HammerDB 6.0 TPROC-H, Benchmark Factory 10.0 TPCE-like
Workload or software specific parameters	RHEL: firewall/SELINUX disabled, nofile 1048576, tuned mssql profile, blockdev setra 4096 SQL Server: Process affinity all CPU and NUMA, MAXDOP 8, tempdb 8x 3GB files, trace flag 3979, control.writethrough=1, control.alternatewritethrough=0
Iterations and result choice	3 test runs, median reported
Cloud VM or instance details	
Number of VMs	1 (six different sizes tested)
VM or instance size	E16ads_v5, E16ads_v7, F16-8amds_v7, E64ads_v5, E64ads_v7, F64-32amds_v7
BIOS name and version	Azure VM
vCPU	E16ads_v5: 16 E16ads_v7: 16 F16-8amds_v7: 8 E64ads_v5: 64 E64ads_v7: 64 F64-32amds_v7: 32
Number of cores/threads	E16ads_v5: 8/16 E16ads_v7: 8/16 F16-8amds_v7: 8/8 E64ads_v5: 32/64 E64ads_v7: 32/64 F64-32amds_v7: 32/32
Memory (GB)	E/F16: 128 GB E/F64: 512 GB
Underlying processor model (if attainable)	V5: AMD EPYC 7763v V7: AMD EPYC 9005
OPTIONAL/If applicable – virtual NUMA information	Single NUMA node
vNIC information/Underlying NIC speed (if attainable)	E16ads_v5: ConnectX max. 12.5 Gbps E16ads_v7: MANA max. 25 Gbps F16-8amds_v7: MANA max. 25 Gbps E64ads_v5: ConnectX max. 32 Gbps E64ads_v7: MANA max. 45 Gbps F64-32amds_v7: MANA max. 75 Gbps

System configuration information			
vNIC driver	NetVSC		
Other instance hardware or parameter details	All testing: Availability Zone 1, Standard security type TPCE-like testing: Two WS2025 client instances used to drive traffic over private (10.0.0.X) network		
Operating system information			
Image or template name and UUID	Red Hat Enterprise Linux 10 (latest minor version) – x64 Gen2, rhel-20190605, 10-lvm		
Operating system name	Red Hat Enterprise Linux 10.2		
Operating system build number	10.2		
Kernel version	6.12.0-211.7.1.el10_2		
Date patches last applied	10/6/2026		
Changes made from CSP image	No		
Instance storage (volume type 1)			
Number of volumes	E/F16: 1x OS, 1x Data, 1x Logs E/F64: 1x OS, 2x Data, 1x Logs		
Volume use in this test	OS, data, logs		
CSP volume type	OS: Premium SSD Data: Premium SSD v2 Logs: Premium SSD v2		
Volume size (GB)	OS: 64GB Data: 1x 2TB (E/F16), 2x 1TB (E/F64) Logs: 512GB		
IOPS requested	TPROC-H	Data	Logs
	E16ads_v5	20,000	5,600
	E16ads_v7	25,000	7,000
	F16-8amds_v7	50,000	14,000
	E64ads_v5	30,000 (per disk)	20,000
	E64ads_v7	55,000 (per disk)	35,200
	F64-32amds_v7	80,000 (per disk)	60,886
	TPCE-like	Data	Logs
	E16ads_v5	10,000	3,000
	E16ads_v7	15,000	3,000
	F16-8amds_v7	15,000	3,000
	E64ads_v5	15,000 (per disk)	3,000
	E64ads_v7	15,000 (per disk)	5,000
	F64-32amds_v7	15,000 (per disk)	5,000

System configuration information			
Throughput requested	TPROC-H	Data	Logs
	E16ads_v5	800	384
	E16ads_v7	2,000	384
	F16-8amds_v7	2,000	384
	E64ads_v5	1,000 (per disk)	384
	E64ads_v7	2,000 (per disk)	384
	F64-32amds_v7	2,000 (per disk)	384
	TPCE-like	Data	Logs
	E16ads_v5	125	125
	E16ads_v7	250	125
	F16-8amds_v7	250	125
	E64ads_v5	500 (per disk)	125
	E64ads_v7	500 (per disk)	125
	F64-32amds_v7	500 (per disk)	125
Encryption type	None		

How we tested

In our tests, we compared database performance on the following Microsoft Azure VM sizes:

- E16ads_v5
- E16ads_v7
- F16-8amds_v7
- E64ads_v5
- E64ads_v7
- F64-32amds_v7

We ran two benchmarks against each size: HammerDB TPROC-H and Benchmark Factory TPCE-like. We used Red Hat Enterprise Linux 10.2 and Microsoft SQL Server 2025 to host the databases. For TPROC-H, we created a single database with scale factor 1,000 and ran the HammerDB client inside the VM. For TPCE-like, we used two 200GB (20-scale) databases and ran two Benchmark Factory clients on separate Windows Server 2025 Datacenter VMs, in the same Azure region and over the private (10.0.0.X) network. For TPROC-H, we report the time required for all queries to finish. For TPCE-like, we report the transactions per second (TPS) summed over both databases. For both benchmarks, we performed three runs at each size and report the median result.

Preparing the operating system

1. Disable the firewall:

```
systemctl disable --now firewalld
```

2. Disable SELinux:

```
sed -i 's/^SELINUX=.*/SELINUX=disabled/g' /etc/selinux/config
```

3. Disable auditd:

```
systemctl disable auditd
```

4. Increase the open-file limit for the mssql user:

```
echo "mssql - nofile 1048576" | tee -a /etc/security/limits.conf
```

5. Install and apply the tuned profile for SQL Server:

```
yum install -y tuned-profiles-mssql
chmod +x /usr/lib/tuned/profiles/mssql/tuned.conf
tuned-adm profile mssql
```

Formatting the data and log drives

1. For E/F16 sizes, skip this step and proceed to step 2. On the E/F64 sizes, create a RAID 0 array from the data drives:

```
mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb /dev/sdc
mdadm --detail --scan | tee -a /etc/mdadm.conf
dracut -f
reboot
```

2. Confirm you have the correct drives with lsblk before proceeding. Make filesystems directly on the drives, with no partitioning:

```
lsblk
mkfs.xfs /dev/sdb -L datavolume
mkfs.xfs /dev/sdc -L logvolume
```

3. Create the mount point directories:

```
mkdir /data1
mkdir /log1
```

4. Find each drive's UUID, and add entries to /etc/fstab:

```
blkid
vi /etc/fstab
UUID="<data volume UUID>" /data1 xfs rw,attr2,noatime 0 0
UUID="<log volume UUID>" /log1 xfs rw,attr2,noatime 0 0
```

5. Mount the filesystems, and reboot:

```
mount -a
reboot
```

6. Set the block device readahead value to 4096:

```
blockdev --setra 4096 /dev/sdb
blockdev --setra 4096 /dev/sdc
```

Installing SQL Server on RHEL 10

1. Add the SQL Server repository:

```
curl -o /etc/yum.repos.d/mssql-server.repo https://packages.microsoft.com/config/rhel/10/mssql-
server-2025.repo
```

2. Install SQL Server:

```
yum install -y mssql-server
```

3. Run mssql-conf setup, and answer the prompts as follows:

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

   a. Enter your edition: `1`
   b. Do you accept the license terms? `Yes`
   c. Enter the SQL Server administrator password: `<password>`
   d. Confirm the SQL Server administrator password: `<password>`

Installing SQLCMD

1. Add the Microsoft production repository:

```
curl https://packages.microsoft.com/config/rhel/9/prod.repo | tee /etc/yum.repos.d/mssql-release.repo
```

2. If needed, remove any existing mssql-tools packages:

```
yum remove mssql-tools unixODBC-utf16 unixODBC-utf16-devel
```

3. Install mssql-tools18 and unixODBC-devel, accepting the license terms at the prompt:

```
yum install -y mssql-tools18 unixODBC-devel
```

4. Update mssql-tools18:

```
yum check-update
yum update mssql-tools18
```

5. Add mssql-tools18 to the PATH, and set the TMP directory, in both .bash_profile and .bashrc:

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

Setting SQL Server process affinity

1. Use ALTER SERVER CONFIGURATION to bind SQL Server to all CPUs and NUMA nodes. Replace the CPU count with the number of CPUs shown inside the VM:

```
sqlcmd -S localhost -U sa -P '<password>' -C -Q "
ALTER SERVER CONFIGURATION SET PROCESS AFFINITY CPU = 0 TO 15;
ALTER SERVER CONFIGURATION SET PROCESS AFFINITY NUMANODE = 0;
"
```

Growing the tempdb files

1. Add space to /var before growing tempdb:

```
lvextend -L +20G /dev/mapper/rootvg-varlv
xfs_growfs /var
```

2. Grow each tempdb data file:

```
sqlcmd -S localhost -U sa -P '<password>' -C -Q "
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev, SIZE = 3072MB, FILEGROWTH = 256MB);
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev2, SIZE = 3072MB, FILEGROWTH = 256MB);
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev3, SIZE = 3072MB, FILEGROWTH = 256MB);
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev4, SIZE = 3072MB, FILEGROWTH = 256MB);
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev5, SIZE = 3072MB, FILEGROWTH = 256MB);
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev6, SIZE = 3072MB, FILEGROWTH = 256MB);
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev7, SIZE = 3072MB, FILEGROWTH = 256MB);
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev8, SIZE = 3072MB, FILEGROWTH = 256MB);
"
```

Setting default file locations, trace flags, and write-through

1. Set the default data and log directories, enable trace flag 3979, configure write-through, and restart SQL Server:

```
/opt/mssql/bin/mssql-conf set filelocation.defaultdatadir /data1
/opt/mssql/bin/mssql-conf set filelocation.defaultlogdir /log1
/opt/mssql/bin/mssql-conf traceflag 3979 on
/opt/mssql/bin/mssql-conf set control.writethrough 1
/opt/mssql/bin/mssql-conf set control.alternatewritethrough 0
systemctl restart mssql-server.service
```

Installing HammerDB and building the TPROC-H schema

1. Download HammerDB:

```
wget https://github.com/TPC-Council/HammerDB/releases/download/v6.0/HammerDB-6.0-Prod-
Lin-RHEL9.tar.gz
```

2. Extract the tarball, and change to the HammerDB directory:

```
tar -zxf HammerDB-6.0-Prod-Lin-RHEL9.tar.gz
cd HammerDB-6.0
```

3. In mssqls_tproch_buildschema.tcl, set the following:

```
set tmpdir "/tmp"
dbset db mssqls
dbset bm TPC-H
diset connection mssqls_tcp false
diset connection mssqls_port 1433
diset connection mssqls_azure false
diset connection mssqls_encrypt_connection false
diset connection mssqls_trust_server_cert true
diset connection mssqls_authentication sql
diset connection mssqls_server {(local)}
diset connection mssqls_linux_server {localhost}
diset connection mssqls_uid sa
diset connection mssqls_pass <password>
diset connection mssqls_linux_authent sql
diset connection mssqls_linux_odbc {ODBC Driver 18 for SQL Server}
diset tpch mssqls_scale_fact 1000
diset tpch mssqls_maxdop 2
diset tpch mssqls_num_tpch_threads 16
diset tpch mssqls_tpch_dbase tpch
diset tpch mssqls_colstore true
```

4. Build the schema:

```
hammerdbcli auto mssqls_tproch_buildschema.tcl
```

5. Grow the TPROC-H log file:

```
sqlcmd -S localhost -U sa -P '<password>' -C -Q "
ALTER DATABASE tpch MODIFY FILE (NAME = tpch_log, SIZE = 300GB);
"
```

Running the TPROC-H benchmark

1. In mssqls_tproch_run.tcl, set the following (change `vuset vu` to `7` for a throughput run):

```
set tmpdir "/tmp"
dbset db mssqls
dbset bm TPC-H
diset connection mssqls_tcp false
diset connection mssqls_port 1433
diset connection mssqls_azure false
diset connection mssqls_encrypt_connection false
diset connection mssqls_trust_server_cert true
diset connection mssqls_authentication sql
diset connection mssqls_server {(local)}
diset connection mssqls_linux_server {localhost}
diset connection mssqls_uid sa
diset connection mssqls_pass <password>
diset connection mssqls_linux_authent sql
diset connection mssqls_linux_odbc {ODBC Driver 18 for SQL Server}
diset tpch mssqls_scale_fact 1000
diset tpch mssqls_maxdop 8
diset tpch mssqls_tpch_dbase tpch
diset tpch mssqls_total_querysets 1
vuset logtotemp 1
loadscript
puts "TEST STARTED"
vuset vu 1
vucreate
set jobid [ vurun ]
vudestroy
puts "TEST COMPLETE"
set of [ open $tmpdir/mssqls_tproch w ]
puts $of $jobid
close $of
```

2. Run the test:

```
./hammerdbcli auto mssqls_tproch_run.tcl
```

Installing Benchmark Factory on the Windows Server clients

1. On each Windows Server client, download and run the Benchmark Factory 10 installer.
2. Follow the wizard, selecting all default options.
3. Click Finish.
4. Run the Benchmark Factory 10 software as Administrator.
5. Enter the license key and activate the license.

Adding the SQL Server certificate to the Windows certificate store

Apply these steps to both Windows Server clients, for all SQL Server VMs.

1. On the Windows Server clients, install SQLCMD using the instructions at https://learn.microsoft.com/en-us/sql/tools/sqlcmd/sqlcmd-download-install.
2. On the SQL Server VMs, generate a self-signed certificate:

```
openssl req -x509 -nodes -newkey rsa:2048 \
  -keyout /var/opt/mssql/mssql.key \
  -out /var/opt/mssql/mssql.pem \
  -days 3650 \
  -subj "/CN=<hostname>" \
  -addext "subjectAltName=DNS:<hostname>"
```

3. Set ownership and permissions on the certificate and key:

```
chown mssql:mssql /var/opt/mssql/mssql.key /var/opt/mssql/mssql.pem
chmod 600 /var/opt/mssql/mssql.key
```

4. Configure SQL Server to use the certificate, and restart the service:

```
/opt/mssql/bin/mssql-conf set network.tlscert /var/opt/mssql/mssql.pem
/opt/mssql/bin/mssql-conf set network.tlskey /var/opt/mssql/mssql.key
systemctl restart mssql-server
```

5. Copy mssql.pem to the Windows Server client, and add it to the trusted root store:

```
certutil -addstore "Root" C:\path\to\mssql.pem
```

6. Add the VM's hostname and IP address to the Windows hosts file (the hostname must match the certificate's CN).

Creating the TPCE-like databases

1. On one of the Windows Server clients, download and install SQL Server Management Studio using the instructions at https://learn.microsoft.com/en-us/ssms/install/install
2. Open SQL Server Management Studio and connect to one of the SQL Server VMs.
3. Right-click Databases, and click New Database.
4. Name the database TPCE.
5. In the Files section, add seven data files for a total of eight data files. Make all eight data files 32GB and the log file 10GB.
6. Click OK.
7. Repeat steps 2 through 6 for a second database named TPCE2.
8. Repeat the previous steps for all six SQL Server VMs, creating two databases on each VM.

Configuring the database connections

Complete these steps on both Windows Server clients, creating a connection for TPCE on the first client, and for TPCE2 on the second client. Create additional connections (one TPCE and one TPCE2) for each SQL Server VM.

1. Run Benchmark Factory as Administrator.
2. Click Create Connection Profile.
3. Select Microsoft SQL Server from the drop-down menu.
4. Click the ODBC tab.
5. Click Add DSN.
6. Select User Data Source, and click Next.
7. Select ODBC Driver 18 for SQL Server, and click Next.
8. Click Finish.
9. Enter a name for the data source, enter the name of the target SQL Server (the server name must match the name on the certificate and in the hosts file), and click Next.
10. Select SQL Server authentication, enter the Login ID and Password, and click Next.
11. Select Change the default database and choose TPCE (TPCE2 on the second client), and click Next. If the connection fails, complete the next step first and come back to this one.
12. On the Connection Encryption drop-down menu, select Optional. Check the Trust server certificate checkbox, and click Finish.
13. Click Test Data Source, and click OK.
14. In the Data Source Name drop-down menu, select the newly created DSN.
15. Enter the User Name and Password for SQL Server.
16. Enter a name for the connection, and click Create Connection.

Creating a Benchmark Factory job

1. In Benchmark Factory, click Add Workload.
2. Select TPC-E (Improved TPCC), and click Add Test.
3. Click the Test Options tab.
4. Enter 20 for the Benchmark Scale.
5. Click the Options sub-tab.
6. Select Create/Load objects if objects don't exist (no backup sets created).
7. Click TPC-E Transaction Mix on the left-hand pane.
8. Click the Test Options tab.
9. For each transaction, edit the latency to No Delay (0 ms).
10. Click the User Load sub-tab.
11. Remove all existing user loads. For the E/F16 sizes, enter 100 users. For the E/F64 sizes, enter 400 users.
12. Click the Timing sub-tab.
13. Set the warmup period to 10 minutes, and the test period to 20 minutes.
14. Click Save/Close.
15. Repeat steps 1 through 14 on the second Windows Server client, using TPCE2 as the target database. Create new jobs as needed to cover all SQL Server target VMs.

Building the TPCE databases

1. Open Benchmark Factory.
2. In the left-hand pane, copy one of the jobs to a new job.
3. On the job copy, delete the TPC-E Transaction Mix item.
4. Click Run Job.
5. Repeat steps 1 through 4 on the second Windows Server client for TPCE2.

Backing up the TPCE database

1. Connect to the database using SQL Server Management Studio.
2. Right-click the TPCE database, and click Tasks → Back Up.
3. Click OK to back up the database.
4. Repeat steps 1 through 3 for TPCE2.

Running the Benchmark Factory TPCE-like test

1. In Benchmark Factory, select the job still containing TPC-E Transaction Mix from the left-hand pane.
2. Repeat step 1 on the second Windows Server client, targeting the same SQL Server VM.
3. Click Run Job on both Windows Server clients.
4. When the tests complete, use the following script to restore each database (replace TPCE with TPCE2 to restore the second database):

```
USE [master];
GO
IF DB_ID(N'tpce') IS NOT NULL
BEGIN
    ALTER DATABASE [tpce] SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE [tpce];
END
GO
USE [master]
RESTORE DATABASE [tpce] FROM  DISK = N'/tmp/backup/tpce.bak' WITH  FILE = 1,
MOVE N'tpce_data1' TO N'/data1/tpce_data1.mdf',
MOVE N'tpce_data2' TO N'/data1/tpce_data2.ndf',
MOVE N'tpce_data3' TO N'/data1/tpce_data3.ndf',
MOVE N'tpce_data4' TO N'/data1/tpce_data4.ndf',
MOVE N'tpce_data5' TO N'/data1/tpce_data5.ndf',
MOVE N'tpce_data6' TO N'/data1/tpce_data6.ndf',
MOVE N'tpce_data7' TO N'/data1/tpce_data7.ndf',
MOVE N'tpce_data8' TO N'/data1/tpce_data8.ndf',
MOVE N'tpce_log' TO N'/log1/tpce_log.ldf',
NOUNLOAD,  STATS = 5
GO
```

5. When both databases have finished restoring, reboot the SQL Server VMs and allow them to idle for at least five minutes.
6. Repeat steps 1 through 5 two more times to get a total of three runs. For each run, sum the two TPS scores together. Report the median sum.

This project was commissioned by Microsoft.

Read the report ▶

Primary contributors

Tech: Max R.

Writing: Laura W.

Design: Laura K.

PM: Scott Luchene

How we created this report

A PT team, which includes the contributors we've listed and others, created this report and performed the technical work behind it. We used AI to create scripts for collecting and parsing performance counters such as CPU, IO, and memory usage, and to draft sections of the report.

Principled Technologies®

Facts matter.®

Principled Technologies is a registered trademark of Principled Technologies, Inc. All other product names are the trademarks of their respective owners. For additional information, review the science behind this report.

DISCLAIMER OF WARRANTIES; LIMITATION OF LIABILITY:

Principled Technologies, Inc. has made reasonable efforts to ensure the accuracy and validity of its testing, however, Principled Technologies, Inc. specifically disclaims any warranty, expressed or implied, relating to the test results and analysis, their accuracy, completeness or quality, including any implied warranty of fitness for any particular purpose. All persons or entities relying on the results of any testing do so at their own risk, and agree that Principled Technologies, Inc., its employees and its subcontractors shall have no liability whatsoever from any claim of loss or damage on account of any alleged error or defect in any testing procedure or result.

In no event shall Principled Technologies, Inc. be liable for indirect, special, incidental, or consequential damages in connection with its testing, even if advised of the possibility of such damages. In no event shall Principled Technologies, Inc.'s liability, including for direct damages, exceed the amounts paid in connection with Principled Technologies, Inc.'s testing. Customer's sole and exclusive remedies are as set forth herein.