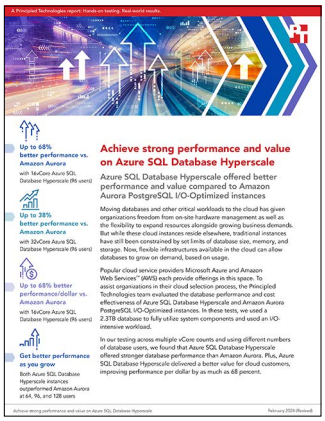

The science behind the report:

Achieve strong performance and value on Azure SQL Database Hyperscale

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 Achieve strong performance and value on Azure SQL Database Hyperscale.

We concluded our hands-on testing on November 29, 2023. The results in this report reflect configurations that we finalized on October 9, 2023 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 HammerDB testing, in new orders per minute (NOPM).

Users	Azure SQL Database Hyperscale NOPM	Amazon Aurora NOPM	Win %
16 vCore			
64	52,710	34,749	51.69%
96	55,323	32,866	68.33%
128	63,493	37,995	67.11%
32 vCore			
64	69,865	58,523	19.38%
96	87,511	63,296	38.26%
128	88,577	69,076	28.23%

Table 2: Pricing for the systems we tested.

	Azure SQL Database Hyperscale			Amazon Aurora			
Users	**Monthly cost**	**NOPM**	**Perf/$**	**Monthly cost**	**NOPM**	**Perf/$**	**Perf/$ win % for SQL Database Hyperscale**
16 vCore							
64	$2,727.72	52,710	19.32	$2,736.51	34,749	12.69	52.25%
96	$2,727.72	55,323	20.28	$2,736.51	32,866	12.01	68.86%
128	$2,727.72	63,493	23.27	$2,736.51	37,995	13.88	67.65%
32 vCore							
64	$4,861.19	69,865	14.37	$4,938.19	58,523	11.85	21.27%
96	$4,861.19	87,511	18.00	$4,938.19	63,296	12.81	40.52%
128	$4,861.19	88,577	18.22	$4,938.19	69,076	13.98	30.33%

System configuration information

Table 3: Detailed information on the systems we tested.

Server configuration information	SQL Database Hyperscale	Amazon Aurora
Tested by	Principled Technologies	Principled Technologies
CSP / region	East US 1	us-east-1
Workload and version	HammerDB 4.8	HammerDB 4.8
Workload specific parameters	Use all warehouses, 15 minute rampup, 30 minute run duration	Use all warehouses, 15 minute rampup, 30 minute run duration
Server platform	Hyperscale Premium 16 vCore	db.r6i.4xlarge
Database engine version	Azure SQL Database Engine version 16[1]	Aurora PostgreSQL 15.3
Processor		
Number of vCPU	16	16
Vendor and model	Intel® Xeon® Platinum 8370C	Intel Xeon Platinum 8375c
Core count (per processor)	16	16
Core frequency (GHz)	2.8	2.9
Stepping	6	6
Hyper-threading	Yes	Yes
Turbo	Yes	Yes
Memory module(s)		
Total memory in system (GB)	81.6	128
Storage		
Storage capacity used (GB)	2,377	2,377

1. Microsoft, "ALTER DATABASE (Transact-SQL) compatibility level," accessed January 24, 2024, https://learn.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql-compatibility-level?view=sql-server-ver16.

Table 4: Detailed information on the systems we tested.

Server configuration information	SQL Database Hyperscale	Amazon Aurora
Tested by	Principled Technologies	Principled Technologies
CSP / region	East US 1	us-east-1
Workload and version	HammerDB 4.8	HammerDB 4.8
Workload specific parameters	Use all warehouses, 15 minute rampup, 30 minute run duration	Use all warehouses, 15 minute rampup, 30 minute run duration
Server platform	Hyperscale Premium 32 vCore	db.r6i.8xlarge
Database engine version	Azure SQL Database Engine version 16	Aurora PostgreSQL 15.3
Processor		
Number of vCPU	32	32
Vendor and model	Intel Xeon Platinum 8370C	Intel Xeon Platinum 8375c
Core count (per processor)	32	32
Core frequency (GHz)	2.8	2.9
Stepping	6	6
Hyper-threading	Yes	Yes
Turbo	Yes	Yes
Memory module(s)		
Total memory in system (GB)	163.2	256
Storage		
Storage capacity used (GB)	2,377	2,377

How we tested

For this project, we compared the database performance of Azure SQL Database Hyperscale vs. Amazon Aurora PostgreSQL using the TPROC-C benchmark HammerDB. We tested Azure SQL Database Hyperscale at 16 and 32 vCores. On AWS, we tested Amazon Aurora PostgreSQL I/O-Optimized with the db.r6i.4xlarge and db.r6i.8xlarge instances. We created a client VM for each environment and installed HammerDB 4.8. We then used the client to connect to each database and build a 25,000 warehouse TPROC-C schema. Once the database was created, we took a snapshot/copy of the database to create a baseline to revert to between tests. We then used HammerDB to run the TPROC-C test with a 15 minute warmup and 30-minute run duration at multiple user counts. After each run we deleted the database and restored from the snapshot/copy.

Using our methodology to aid your own deployments

While the methodology below describes in great detail how we accomplished our testing, it is not a deployment guide. However, because we include many basic installation steps for operating systems and testing tools, reading our methodology may help with your own installation.

Creating the SQL Database Hyperscale database

1. Log into the Azure portal at portal.azure.com.
2. In the search field, type Azure SQL.
3. Click Create.
4. In the SQL databases box, leave the resource type as Single database, and click Create.
5. From the drop-down menus, select Subscription and Resource groups.
6. In the Database name field, enter a name.
7. Under the Server drop-down, click Create new.
8. In the Server name field, enter a name for the database server.
9. From the Location drop-down field, select the location of the server.
10. Next to Authentication method, select Use SQL authentication.
11. Enter a name and password for the SQL authentication, and click OK.
12. Next to Workload environment, click the Production radio button.
13. Next to Compute + storage, click Configure database.
14. From the Service tier drop-down, select Hyperscale (Highly scalable compute and storage).
15. Next to Compute tier, select the Provisioned radio button.
16. Next to Hardware Configuration, click Change configuration.
17. Select Premium-series, and click OK.
18. Assign the number of provisioned vCores. We tested the following two configurations:
    - 16 vCores
    - 32 vCores
19. Click Apply.
20. Next to Backup storage redundancy, click the Locall-redundant backup storage radio button.
21. Click Next: Networking.
22. Next to Connectivity method, select Public endpoint.
23. Next to Allow Azure services and resources to access this server, toggle the button to yes.
24. Next to Add current client IP address, toggle the button to yes.
25. Click Next: Security.
26. Leave the defaults, and click Next: Additional settings.
27. Click Next: Tags.
28. Enter any relevant tags, and click Next: Review + create.
29. Click Create.

Creating the Azure HammerDB client VM

1. In the Azure portal, navigate to the Virtual Machine tab.
2. Click Create→Azure virtual machine.
3. From the drop-down menus, select Subscription and Resource groups.
4. Enter a name for the Virtual Machine.
5. Next to Availability options, select No infrastructure redundancy required.
6. In the Image drop-down menu, select Windows Server 2022 Datacenter: Azure Edition - x64 gen 2.
7. From the Size drop-down menu, select Standard_D16ds_v5.
8. Enter an Administrator username and password.
9. Click Next: Disks.
10. From the OS disk type drop-down menu, select Standard SSD (locally-redundant storage).
11. Click Next: Networking.
12. Check the box next to Delete public IP and NIC when VM is deleted.
13. Click Next: Management.
14. Click Next: Monitoring.
15. Click Next: Advanced.
16. Click Next: Tags.
17. Fill in any necessary tags, and Click Next: Review + create.
18. Click Create.

Creating firewall rule for the Azure HammerDB client VM communication to the Azure SQL DB

1. In the Azure portal, navigate to the SQL Database.
2. Under Getting started, click Configure access→Configure.
3. Click Add a virtual network rule.
4. Select the virtual network that your client VM is assigned to, and click OK.
5. Click Add a firewall rule.
6. Enter the Rule name external IP address of the client VM, and click OK.
7. Click Save.

Configuring the Azure HammerDB client VM and installing HammerDB

1. Log into the Windows VM you created.
2. In Server Manager, navigate to the Local Server tab.
3. Disable the firewall.
4. Disable Windows defender.
5. Disable IE Enhanced Security Configuration.
6. Install the latest updates.
7. Download and install SQL Server Management Studio 19.0 from https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16#download-ssms.
8. Download and install the SQL Server ODBC Driver 18 from https://learn.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server?view=sql-server-ver16.
9. Download and install HammerDB 4.8 from https://www.hammerdb.com/download.html.

Building the TPROC-C schema on Azure

1. Log into the HammerDB client.
2. Launch the HammerDB 4.8 program.
3. Click Options→Benchmark.
4. Change the Benchmark to SQL Server, and click OK.
5. Click TPROC-C→Schema→Options.
6. In the SQL Server field, enter the Server name.
7. Click the TCP check box.
8. Click the Azure check box.
9. Click the SQL Server Authentication radio button.
10. Enter the SQL Username and Password.
11. For the number of warehouses, enter 25000.
12. For the number of virtual users to build the schema, enter 64.
13. Click OK.
14. Double-click Build.

Creating a copy of the SQL database

1. In the Azure portal, navigate to the Azure SQL Database.
2. Click Copy.
3. Click Next: Review + create >.
4. Click Create.

Restoring the SQL database from a copy

1. In the Azure portal, navigate to the Azure SQL Database.
2. Click the database copy.
3. Click Restore.
4. Select the Restore point time you'd like to restore to.
5. Give the database a name.
6. In the Compute + storage option, click Configure database.
7. Make sure Hyperscale and Provisioned are selected.
8. Change the Hardware Configuration to Premium-series, and click OK.
9. Set the number of vCores, and click Apply.
10. Click Review + create.
11. Click Create.

Running the TPROC-C test on Azure

1. In the left-hand pane, click Driver Script→Options.
2. Change the Rampup time to 15 minutes and the Test Duration to 30 minutes.
3. Check the Use All Warehouses checkbox.
4. Click OK.
5. Double-click Load.
6. Click VirtualUser →Options.
7. Change the number of virtual users to [64, 96, 128].
8. Check the box next to Log Output to Temp and Use Unique Log Name.
9. Click OK.
10. Double-click Create.
11. Double-click Run.

Creating the Amazon Aurora PostgreSQL cluster on AWS

1. Log into AWS, and navigate to the AWS Management Console.
2. Click RDS.
3. Click Create database.
4. For the Engine type, select Aurora (PostgreSQL Compatible).
5. Under the Available versions drop-down menu, select Aurora PostgreSQL (Compatible with PostgreSQL 15.3).
6. Give DB cluster a name and Master password.
7. From the Configuration options, select Aurora I/O-Optimized.
8. Under the Instance configuration drop-down, select db.r6i.4xlarge.
9. Under Availability and durability options, select Don't create an Aurora Replica.
10. Under Public access, select Yes.
11. Click Create database.

Creating the HammerDB client EC2 instance

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. Click Launch instance.
4. Give the instance a name.
5. In the search window under Application and OS Images (Amazon Machine Image), type `Red Hat Enterprise Linux 8` and press enter.
6. Click AWS Marketplace AMIs.
7. Next to Red Hat Enterprise Linux 8, click Select.
8. Click Continue.
9. Under Instance type, select m6i.4xlarge.
10. Under Key pair (login), select the proper Key pair from the drop-down.
11. Under Network settings, click edit, select the subnet that your RDS cluster is in from the drop-down, and leave the rest as defaults.
12. Under Configure storage, set the storage size to 100 GiB and gp3.
13. Click Launch Instance.

Configuring the HammerDB client on AWS

1. SSH into the EC2 instance:

```
ssh -i [ec2 Key pair] ec2-user@[IP Address]
```

2. Disable SELinux:

```
sudo vi /etc/selinux/config
SELINUX=disabled
```

3. Register your RHEL subscription:

```
sudo subscription-manager register --username <RHEL_USERNAME> --password <RHEL_
PASSWORD> --auto-attach
```

4. Install nano & wget:

```
sudo dnf install -y nano wget
```

5. Update RHEL 8:

```
sudo dnf update -y
```

6. Install the PostgreSQL 15 client repo:

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86_64/pgdg-redhat-
repo-latest.noarch.rpm
```

7. Disable the built-in PostgreSQL module:

```
sudo dnf -qy module disable postgresql
```

8. Install the PostgreSQL 15 client:

```
sudo dnf install -y postgresql15
```

9. Download HammerDB 4.8:

```
wget https://github.com/TPC-Council/HammerDB/releases/download/v4.8/HammerDB-4.8-RHEL8.tar.gz
tar -xzf HammerDB-4.8-RHEL8.tar.gz
```

Building the TPROC-C schema on AWS

1. Change into the HammerDB directory:

```
cd HammerDB-4.8
```

2. Switch to the HammerDB command line interface:

```
./hammerdbcli
```

3. Change the following parameters in the build script:

```
dbset db pg
dbset bm tproc-c
diset connection pg_host [Writer instance Endpoint name]
diset tpcc pg_count_ware 25000
diset tpcc pg_num_vu 64
diset tpcc pg_superuserpass [Password]
diset tpcc pg_pass [Password]
diset tpcc pg_storedprocs true
diset tpcc pg_partition true
```

4. Build the schema:

```
buildschema
```

5. After the build is finished, destroy the virtual users.

```
vudestroy
```

Taking a snapshot of the RDS cluster

1. Log into AWS, and navigate to the AWS Management Console.
2. Click RDS.
3. In the left-hand pane, click Databases.
4. Click the Regional cluster Db identifier radio button.
5. Click Actions→Take snapshot.
6. Give the snapshot a name, and click Take snapshot.

Restoring the RDS cluster from a snapshot

1. Log into AWS, and navigate to the AWS Management Console.
2. Click RDS.
3. In the left-hand pane, click Snapshots.
4. Click the check box next to your snapshot.
5. Click Actions → Restore snapshot.
6. Give the cluster a name in the DB instance identifier field.
7. Under the Instance configuration section, select the instance size.
8. Under Public acces, click the Yes radio button.
9. Choose the existing VPC security group and Availability zone that your client instance is in.
10. Review, and click Restore DB cluster.

Running the TPROC-C test on AWS

1. Change into the HammerDB directory:

```
cd HammerDB-4.8
```

2. Switch to the HammerDB command line interface:

```
./hammerdbcli
```

3. Change the following parameters in the load script:

```
dbset db pg
dbset bm tproc-c
diset connection pg_host [Writer instance Endpoint name]
diset tpcc pg_count_ware 25000
diset tpcc pg_superuserpass [Password]
diset tpcc pg_pass [Password]
diset tpcc pg_storedprocs true
diset tpcc pg_partition true
diset tpcc pg_rampup 15
diset tpcc pg_duration 30
diset tpcc pg_allwarehouse true
```

4. Load the driver script:

```
loadscript
```

5. Setup the virtual users:

```
vuset vu [64, 96, 128]
vuset showoutput 1
vuset logtotemp 1
vuset unique 1
vucreate
```

6. Run the test:

```
vurun
```

7. After the run is complete, delete the virtual users:

```
vudestroy
```

Read the report at https://facts.pt/6lj1RCw

This project was commissioned by Microsoft.

Principled Technologies®

Facts matter.®