The science behind the report:

Improving database performance and value with an easy migration to Azure Database for MySQL – Flexible Server with AMD EPYC

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 Improving database performance and value with an easy migration to Azure Database for MySQL - Flexible Server with AMD EPYC.

We concluded our hands-on testing on January 24, 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 January 24, 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: HammerDB results of our testing.

Sysbench results Flexible Server (autoscale IOPS enabled) vs. Single Server						
Threads	**4**	**8**	**16**	**32**	**64**	**128**
8vCPU Flexible Server (128GB) (with autoscale IOPS enabled)						
TPS	314.80	616.49	1,104.58	1,709.32	2,073.65	2,180.52
QPS	6,296.07	12,329.74	22,091.59	34,186.37	41,472.97	43,610.48
Latency (ms)	12.70	12.98	14.48	18.72	30.86	58.69
8vCPU Single Server (4266GB)						
TPS	144.73	283.73	564.04	1,076.71	1,586.37	1,905.61
QPS	2,894.69	5,674.63	11,280.74	21,534.25	31,727.84	38,113.42
Latency (ms)	27.63	28.19	28.36	29.72	40.34	67.16
Flexible Server TPS win %	117.51%	117.28%	95.83%	58.75%	30.72%	14.43%

Table 2: How we calculated costs for Azure Database for MySQL – Flexible Server and Azure Database for MySQL – Single Server. Cost data comes from the Azure cost calculator as of January 24, 2024. Source: Principled Technologies.

	Flexible Server (AutoIOPS)	Single Server
Compute (monthly)	$564.23	$613.78
Storage (monthly)	$16.64	$588.71
Off-Peak MReqs/hr	3	-
Off-Peak Hrs (max=730)	140	-
Peak MReqs/hr	12	-
Peak Hrs (max=730)	20	-
$/Mreqs	0.2	-
Req cost	$132.00	-
Total monthly cost	$712.87	$1,202.49

System configuration information

Table 3: Detailed information on the systems we tested.

System configuration information	Azure Database for MySQL – Flexible Server instance	Azure Database for MySQL – Single Server instance
Tested by	Principled Technologies	Principled Technologies
Test date	1/8/2024	1/8/2024
CSP / region	Central US (Zone 3)	Central US (Zone 3)
Workload and version	Sysbench 1.0.18	Sysbench 1.0.18
Workload specific parameters	10 tables, 5M rows per table, 4/8/16/32/64/128 threads, 5-minute tests, 70/30 R/W ratio OLTP	10 tables, 5M rows per table, 4/8/16/32/64/128 threads, 5-minute tests, 70/30 R/W ratio OLTP
Iterations and result choice	Three runs, median	Three runs, median
Server platform	Standard_D8ads_v5	GP_Gen5_8
SQL version	MySQL™ 8.0.32	MySQL 8.0.32
Date of last OS updates/patches applied	1/8/2024	1/8/2024
Processor		
Number of vCPU	8	8
Core count (per processor)	64	20
Core frequency (GHz)	2.45	2.30
Stepping	6	6
Hyper-threading	Yes	Yes
Turbo	Yes	Yes
Memory module(s)		
Total memory in system (GB)	32	40
Data drive		
Number of drives	1	1
Drive size (GB)	128	4,266
Drive information (speed, interface, type)	Autoscale IOPS	GPv2

How we tested

We compared two Azure Database for MySQL offerings. First, we deployed an 8vCore General Purpose Azure Database for MySQL – Flexible Server instance (Standard_D8ads_v5) with a 128GB volume, with Autoscale IOPS and storage autogrowth enabled. Second, we provisioned an 8vCore General Purpose Azure Database for MySQL – Single Server instance with a 4,266GB volume. This increased volume size allowed us to match the Flexible Server instance's maximum disk IOPS of 128,000. We compared their performance using the OLTP workload from Sysbench 1.0.18. We sized the db to have 10 tables with 5M rows each, running the 70/30 R/W workload at 4, 8, 16, 32, 64, and 128 threads on each of the managed database services (Azure Database for MySQL – Flexible Server and Azure Database for MySQL – Single Server).

While testing, we used out-of-the-box parameters with the exception of the parameter max_prepared_stmt_count, which we set to 655,350 on both solutions.

Setting up MySQL Flexible Server and Single Server environments on Azure

Creating the client VM

1. Log into the Azure Portal, and navigate to the Virtual Machines service.
2. Click Add to open the Add VM wizard.
3. On the Basics tab, set the following:
    a. Choose your Subscription from the drop-down menu.
    b. Choose your Resource group from the drop-down menu.
    c. Name the Virtual Machine.
    d. Choose your Region from the drop-down menu. We used Central US.
    e. Under Availability options, choose Availability zone.
    f. Choose your Availability zone. We used Zone 1.
    g. From the Image drop-down menu, choose Ubuntu 20.04 LTS.
    h. Under VM architecture, select x64.
    i. Leave Azure Spot instance set to No.
    j. Select the instance size you wish to use; we used Standard D8ds_v5.
    k. Keep the Authentication type set to SSH public key.
    l. Choose a Username and Key pair name, and select generate a new key pair.
    m. Leave Public inbound ports set to Allow selected ports.
    n. For Select inbound ports, choose SSH (22).
    o. Click Next: Disks.
4. On the Disks tab, set the following:
    a. For the OS disk size, select 30GB.
    b. For the OS disk type, choose Standard SSD from the drop-down menu.
    c. Leave the default Encryption type.
    d. Click Next: Networking.
5. On the Networking tab, set the following:
    a. Leave all settings as default.
    b. Ensure the checkbox for accelerated networking is enabled.
    c. Click Next: Management.
6. On the Management tab, leave all defaults.
7. On the Advanced tab, leave all defaults.
8. On the Tags tab, add any tags you wish to use.
9. On the Review + create tab, review your settings, and click Create.

Creating the Azure Database for MySQL – Flexible Server instance

1. Log into the Azure Portal, and navigate to the Azure Database for MySQL Flexible Server service.
2. Click Create, and select Flexible Server.
3. Select your Subscription and Resource group.
4. Enter a name for the instance, and select the region you wish to work in. We used Central US.
5. For the MySQL version, select 8.0.
6. For the Workload type, select For small or medium size databases.
7. Under Compute + storage, click Configure server.
    a. Compute tier: General Purpose
    b. Compute processor: AMD
    c. Compute size: Standard_D8ads_v5
    d. Storage size (in GiB): 128
    e. IOPS: Autoscale IOPS
    f. Storage Auto-growth: Enabled
    g. Enable high availability: Disabled
    h. Backup retention period (in days): 7
    i. Geo-redundancy: Disabled
    j. Click Save.
8. Choose your Availability zone. We used Zone 1.
9. Under Authentication method, select MySQL authentication only, and enter a username and password.
10. Click Next: Networking.
11. On the Networking tab, set the following:
    a. Connectivity method: Public access (allowed IP addresses) and Private endpoint.
    b. Public access: enabled.
    c. Add a firewall rule with your Ubuntu client IP.
    d. Click Next: Security.
12. On the Security tab, leave all defaults.
13. On the Tags tab, add any tags you wish to use.
14. On the Review + create tab, review your settings, and click Create.
15. Once the deployment is complete, click the Server Parameters blade.
16. Click All, and in the search bar, search for max_prepared_stmt_count.
17. Change the value for max_prepared_stmt_count to 655350, and click Save.

Creating the Azure Database for MySQL – Single Server instance

1. Log into the Azure Portal, and navigate to the Azure Cloud Shell UI.
2. Use the following command to deploy your Single Server instance:

```
az mysql server create --resource-group [resource group] --name [name] --location centralus --admin-user [admin name] --admin-password [password] --sku-name GP_Gen5_8 --storage-size 4368384 --version 8.0 --tags [key=value]
```

3. Configure the max_prepared_stmt_count parameter:

```
az mysql server configuration set --name max_prepared_stmt_count --resource-group [resource group] --server [server name] --value 655350
```

4. Create a firewall rule for the Ubuntu client:

```
az mysql server firewall-rule create --resource-group [resource group] --server [server name] --name public --start-ip-address [client IP] --end-ip-address [client IP]
```

Configuring Ubuntu 20.04 LTS client (both solutions)

Configuring Ubuntu 20.04 LTS client

1. SSH into the Ubuntu client VM.
2. Run updates and upgrade:

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

3. Install required packages (sysbench and MySQL client tools):

```
sudo apt-get install sysbench -y
sudo apt-get install mysql-client -y
```

Creating and preparing the test database (both solutions)

Creating the database

1. SSH into the Ubuntu client VM.
2. Connect to the MySQL instance:

```
mysql -h [mysql-instance-name] -u [username] -p
```

3. Enter your password.
4. Create a database called testdb:

```
CREATE DATABASE testdb;
```

5. Exit MySQL:

```
exit;
```

Preparing the database

1. Navigate to the sysbench folder:

```
cd /usr/share/sysbench
```

2. Run the sysbench prepare statement with 10 tables and 5M rows per table:

```
date ; sysbench oltp_write_only.lua --tables=10 --table-size=5000000 --threads=10 --time=720000
--mysql-host=[mysql-instance-name] --mysql-db=testdb --mysql-user=[username] --mysql-
password='[password]' --mysql-port=3306 --report-interval=10 --percentile=99 prepare ; date
```

3. Once the data insertion is complete, wait 20 minutes for background tasks to complete.

Running the tests

1. On the client system, navigate to the sysbench folder:

```
cd /usr/share/sysbench
```

2. Run the sysbench 70/30 R/W workload for 30 minutes at 64 threads as a warmup:

```
sysbench oltp_read_write.lua --tables=10 --table-size=5000000 --db-driver=mysql --threads=64
--time=1800 --mysql-host=[mysql-instance-name] --mysql-db=testdb --mysql-user=[username] --mysql-
password='[password]' --mysql-port=3306 --report-interval=1 --percentile=99 run | tee ~/warmup.log
```

3. Reboot the client VM.
4. Return to the sysbench folder:

```
cd /usr/share/sysbench
```

5. Set the ulimit to two times the number of threads to be tested:

```
ulimit -n =[threads x2]
```

6. Run the 70/30 R/W workload for 5 minutes at 4, 8, 16, 32, 64, and 128 threads, repeating steps 3-5 after each test at a particular thread size and runtime.
7. Repeat step 6 two more times and collect the median results.

Migrating a 50GB database from Single Server to Flexible Server

Creating the Azure Database for MySQL - Single Server instance

1. Log into the Azure Portal, and navigate to the Azure Cloud Shell UI.
2. Use the following command to deploy your Single Server instance:

```
az mysql server create --resource-group [resource group] --name single-server --location centralus
--admin-user [admin name] --admin-password [password] --sku-name GP_Gen5_8 --storage-size 131072
--version 8.0 --tags [key=value]
```

3. Configure the max_prepared_stmt_count parameter:

```
az mysql server configuration set --name max_prepared_stmt_count --resource-group [resource group]
--server single-server --value 655350
```

4. Create a firewall rule for the Ubuntu client:

```
az mysql server firewall-rule create --resource-group [resource group] --server single-server --name
public --start-ip-address [client IP] --end-ip-address [client IP]
```

Creating the database

1. SSH into the Ubuntu client VM.
2. Connect to the MySQL instance:

```
mysql -h [mysql-instance-name] -u [username] -p
```

3. Enter your password.
4. Create a database called testdb:

```
CREATE DATABASE testdb;
```

5. Exit MySQL:

```
exit;
```

Preparing the database

1. Navigate to the sysbench folder:

```
cd /usr/share/sysbench
```

2. Run the sysbench prepare statement with 25 tables and 10M rows per table:

```
date ; sysbench oltp_write_only.lua --tables=25 --table-size=10000000 --threads=10 --time=720000
--mysql-host=[mysql-instance-name] --mysql-db=testdb --mysql-user=[username] --mysql-
password='[password]' --mysql-port=3306 --report-interval=10 --percentile=99 prepare ; date
```

3. Once the data insertion is complete, wait 20 minutes for background tasks to complete.

Migrating the database with Azure Database for MySQL Import CLI

1. Log into the Azure Portal, and navigate to the Azure Cloud Shell UI.
2. Use the following command to create a Flexible Server instance with the same vCore count as the Single Server instance and import the previously created database to the new instance:

```
az mysql flexible-server import create --data-source-type "mysql_single" --data-source "single-
server" --resource-group [resource group] --name "flexible-server" --sku-name "Standard_D8ads_v5"
--iops 12800 --high-availability Disabled --auto-scale-iops Disabled
```

Note: The Azure Database for MySQL Import command maps over the corresponding tier, version, sku-name, storage-size, location, geo-redundant-backup, public access, tags, auto grow, backup-retention-days, admin-user and admin-password properties from Single Server to Flexible Server as smart defaults if no inputs are provided to the CLI command.

You can choose to override the smart defaults by providing inputs for these optional parameters.

Read the report at https://facts.pt/WA4Av1Z

This project was commissioned by Microsoft.

Principled Technologies®

Facts matter.®