TaurusDB

Performance White Paper

Issue 01

Date 2024-12-30

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Test Method

GaussDB(for MySQL) is a MySQL-compatible, enterprise-grade distributed database service. It uses a compute-storage decoupled architecture and supports up to 128 TB of storage. With GaussDB(for MySQL), there is no need to do sharding, and no need to worry about data loss. It provides the superior performance of commercial databases at the price of open-source databases.

Test Environment

GaussDB(for MySQL) test environment is as follows:

- Region: AP-Singapore
- AZ: AZ1
- Instance: The instance contains a primary node and a read replica for the test.
- Elastic Cloud Server (ECS): general computing-plus | c7.8xlarge.4 | 32 vCPUs | 128 GB, CentOS 7.6 (64 bit). The ECS and instance nodes are in the same AZ. Bind an EIP to the ECS because additional compilation tools need to be installed on stress testing tools.

Test Tool

Table 1-1 Test tool

Tool	Description	Version
Sysbench	Sysbench is a multi-threaded benchmark tool based on LuaJIT. It is most frequently used for database benchmarks. With sysbench, you can quickly get an impression of database performance. For details, visit **https://github.com/akopytov/sysbench**	**Sysbench 1.0.18**

Perform the following commands to install sysbench:

Log in to an ECS and download the sysbench software package:

wget https://codeload.github.com/akopytov/sysbench/zip/refs/tags/1.0.18

yum install -y autoconf libtool mysql mysql-devel vim unzip

Decompress the software package.

unzip 1.0.18

Install the software package.

cd sysbench-1.0.18

./autogen.sh

./configure

make

make install

Test Procedure

NOTICE

The following tests are performed on an ECS. Replace the number of concurrent threads, connection IP address, connection port, username, and user password based on the site requirements.

Performance test data (including SQL) is automatically generated by the sysbench tool.

The ECS and the instance are in the same AZ.

To ensure that sysbench runs properly in high-concurrency scenarios (concurrent requests: 512-1000), increase the value of **max_prepared_stmt_count**. The recommended value is **1048576**. Too many Prepare statements consume a lot of memory space, resulting in out-of-memory (OOM). For an instance with 4 vCPUs and 16 GB memory, set this parameter to **400000**.

To improve performance, configure the following parameter and reboot the instance. Modifying the parameter value does not affect the product functions and reliability.

log-bin: **OFF**

Testing write-only performance:

Step 1 Import data.

1. Create the test database **sbtest**.

   mysql -u<*user*>**-P** <*port*> **-h** <*host*> **-p -e "create database sbtest"**

2. Import the test background data to the **sbtest** database.

   sysbench --db-driver=mysql --mysql-host=<*host*> **--mysql-port=**<*port*> **--mysql-user=**<*user*> **--mysql-password=**<*password*> **--mysql-db=sbtest --table_size=25000 --tables=250 --threads=**<*thread_num*> **oltp_common prepare**

Step 2 Test the write-only performance. The process takes about 10 minutes.

sysbench --db-driver=mysql --mysql-host=<*host*> --mysql-port=<*port*> --mysql-user=<*user*> --mysql-password=<*password*> --mysql-db=sbtest --table_size=25000 --tables=250 --time=600 --threads=<*thread_num*> --percentile=95 --report-interval=1 oltp_write_only run

Step 3 Delete data.

sysbench --db-driver=mysql --mysql-host=<*host*> --mysql-port=<*port*> --mysql-user=<*user*> --mysql-password=<*password*> --mysql-db=sbtest --table_size=25000 --tables=250 --threads=<*thread_num*> oltp_common cleanup

----End

Testing read-only performance.

Step 1 Import data.

1. Create the test database **sbtest**.

   mysql -u<*user*> -P<*port*> -h<*host*> -p -e "create database sbtest"

2. Import the test background data to the **sbtest** database.

   sysbench --db-driver=mysql --mysql-host=<*host*> --mysql-port=<*port*> --mysql-user=<*user*> --mysql-password=<*password*> --mysql-db=sbtest --table_size=25000 --tables=250 --threads=<*thread_num*> oltp_common prepare

Step 2 Test the read-only performance. The process takes about 10 minutes.

sysbench --db-driver=mysql --mysql-host=<*host*> --mysql-port=<*port*> --mysql-user=<*user*> --mysql-password=<*password*> --mysql-db=sbtest --table_size=25000 --tables=250 --time=600 --range_selects=0 --skip-trx=1 --threads=<*thread_num*> --percentile=95 --report-interval=1 oltp_read_only run

Step 3 Delete data.

sysbench --db-driver=mysql --mysql-host=<*host*> --mysql-port=<*port*> --mysql-user=<*user*> --mysql-password=<*password*> --mysql-db=sbtest --table_size=25000 --tables=250 --threads=<*thread_num*> oltp_common cleanup

----End

Testing read/write performance:

Step 1 Import data.

1. Create the test database **sbtest**.

   mysql -u<*user*> -P<*port*> -h <*host*> **-p -e "create database sbtest"**

2. Import the test background data to the **sbtest** database.

   sysbench --db-driver=mysql --mysql-host=<*host*> --mysql-port=<*port*> --mysql-user=<*user*> --mysql-password=<*password*> --mysql-db=sbtest --table_size=250000 --tables=25 --threads=<*thread_num*> oltp_common prepare

Step 2 Test the read and write performance. The process takes about 10 minutes.

sysbench --db-driver=mysql --mysql-host=<*host*> --mysql-port=<*port*> --mysql-user=<*user*> --mysql-password=<*password*> --mysql-db=sbtest --table_size=250000 --tables=25 --time=600 --threads=<*thread_num*> --percentile=95 --report-interval=1 oltp_read_write run

Step 3 Delete data.

sysbench --db-driver=mysql --mysql-host=<*host*> --mysql-port=<*port*> --mysql-user=<*user*> --mysql-password=<*password*> --mysql-db=sbtest --table_size=250000 --tables=25 --threads=<*thread_num*> oltp_common cleanup

----End

Test Metric

- Transactions per second (TPS) indicates the number of transactions executed per second.
- Queries per second (QPS) indicates the number of SQL statements, including INSERT, SELECT, UPDATE, and DELETE statements, executed per second.

2 Performance Data

2.1 Read/Write Mode

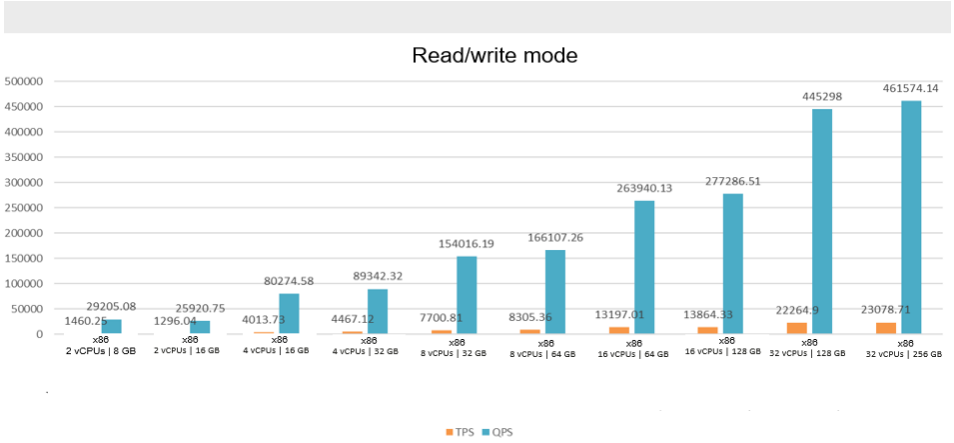
Dedicated DB Instance Test Data

Table 2-1 Test data in read/write mode

Mode	Tables	Data Volume	Threads	Instance Specifications	TPS	QPS
Read/ write	25	250,000	128	x86 | 2 vCPUs | 8 GB	1460.25	29205.08
			128	x86 | 2 vCPUs | 16 GB	1296.04	25920.75
			128	x86 | 4 vCPUs | 16 GB	4013.73	80274.58
			128	x86 | 4 vCPUs | 32 GB	4467.12	89342.32
			256	x86 | 8 vCPUs | 32 GB	7700.81	154016.19
			256	x86 | 8 vCPUs | 64 GB	8305.36	166107.26
			512	x86 | 16 vCPUs | 64 GB	13197.01	263940.13
			512	x86 | 16 vCPUs | 128 GB	13864.33	277286.51
			512	x86 | 32 vCPUs | 128 GB	22264.9	445298
			512	x86 | 32 vCPUs | 256 GB	23078.71	461574.14
			512	x86 | 60 vCPUs | 256 GB	22638.79	452775.89
			512	x86| 64 vCPUs | 512 GB	22638.21	452764.3

Dedicated DB Instance Test Results

Figure 2-1 Test results



2.2 Read-only Mode

Dedicated DB Instance Test Data

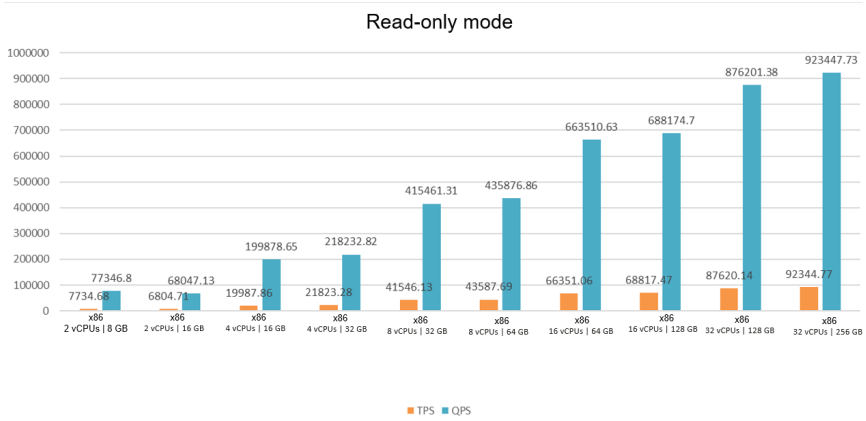
Table 2-2 Test data in read-only mode

Mode	Tables	Data Volume	Threads	Instance Specifications	TPS	QPS
Read-only	250	25000	64	x86 | 2 vCPUs | 8 GB	7734.68	77346.8
			64	x86 | 2 vCPUs | 16 GB	6804.71	68047.13
			64	x86 | 4 vCPUs | 16 GB	19987.86	199878.65
			64	x86 | 4 vCPUs | 32 GB	21823.28	218232.82
			128	x86 | 8 vCPUs | 32 GB	41546.13	415461.31
			128	x86 | 8 vCPUs | 64 GB	43587.69	435876.86
			256	x86 | 16 vCPUs | 64 GB	66351.06	663510.63
			256	x86 | 16 vCPUs | 128 GB	68817.47	688174.7
			512	x86 | 32 vCPUs | 128 GB	87620.14	876201.38

			512	x86 | 32 vCPUs | 256 GB	92344.77	923447.73
			512	x86 | 60 vCPUs | 256 GB	90295.49	902954.92
			1000	x86| 64 vCPUs | 512 GB	91093.96	910939.6

Dedicated DB Instance Test Results

Figure 2-2 Test results



2.3 Write-only Mode

Dedicated DB Instance Test Data

Table 2-3 Test data in write-only mode

Mode	Tables	Data Volume	Threads	Instance Specifications	TPS	QPS
Write-only	250	25000	128	x86 | 2 vCPUs | 8 GB	4972.9	29837.37
			128	x86 | 2 vCPUs | 16 GB	4848.42	29090.52
			128	x86 | 4 vCPUs | 16 GB	15117.9	90707.38
			128	x86 | 4 vCPUs | 32 GB	17651.49	105908.94
			256	x86 | 8 vCPUs | 32 GB	31456.27	188737.65

			256	x86 | 8 vCPUs | 64 GB	34088.75	204532.49
			512	x86 | 16 vCPUs | 64 GB	58271.73	349630.37
			512	x86 | 16 vCPUs | 128 GB	60286.91	361721.43
			512	x86 | 32 vCPUs | 128 GB	81209.8	487258.82
			512	x86 | 32 vCPUs | 256 GB	85428.83	512573
			512	x86 | 60 vCPUs | 256 GB	81580.39	489482.33
			512	x86| 64 vCPUs | 512 GB	81922.84	491537.02

Dedicated DB Instance Test Results

Figure 2-3 Test results