Distributed Database Middleware

Performance White Paper

Issue	01
Date	2022-01-07

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.

Contents

1 Sysbench..... 1

1 Sysbench

Test Environment

- **Region**: CN-Hong Kong
- **AZ**: AZ4
- **Shards per DB instance**: 16
- **Logical tables**: 1
- **Data volume**: 320 million lines
- **RDS instance specifications**: 8 vCPUs | 16 GB, 16 vCPUs | 32 GB, 32 vCPUs | 64 GB, ultra-high I/O
- **One testing tool (sysbench)**: 32 vCPUs | 64 GB

Test Tool

Sysbench is a multi-threaded modular benchmark tool based on LuaJIT. It is mostly used for testing database benchmarks. With sysbench, you can perform multi-thread concurrent operations using the built-in database test model to evaluate the database performance. For details, visit **https://github.com/akopytov/sysbench**.

Sysbench 1.0.20 is used in this test. Run the following commands to install it:

 # wget -c https://github.com/akopytov/sysbench/archive/1.0.20.zip

 # yum install autoconf libtool mysql mysql-devel vim unzip

 # unzip 1.0.20.zip

 # cd sysbench-1.0.20

 # ./autogen.sh

 # ./configure

 # make

 # make install

Test Procedure

Replace the database name, connection IP address, and user password based on site requirements.

Step 1 Import test data.

1. Create a schema.
2. Log in to the target DDM database using the CLI and create a table.

   Table structure: CREATE TABLE sbtest1(

   id int UNSIGNED NOT NULL auto_increment PRIMARY KEY,

   k INTEGER UNSIGNED DEFAULT '0' NOT NULL,

   c varCHAR(1200) DEFAULT '' NOT NULL,

   pad varCHAR(1200) DEFAULT '' NOT NULL

) dbpartition by hash(id);
3. Run the following command to import test data into the database:

   sysbench --test=/usr/local/share/sysbench/tests/include/oltp_legacy/oltp.lua --oltp_tables_count=1 --report-interval=5 --oltp-table-size=<*data*> **--mysql-user=**<*user*> **--mysql-password=**<*password*> **--mysql-table-engine=innodb --rand-init=on --mysql-host=**<*host*> **--mysql-port=*5066*** **--mysql-db=**<*db-name*> **--max-time=300 --max-requests=0 --oltp_skip_trx=off --oltp_auto_inc=on --oltp_range_size=5 --num-threads=256 --oltp_secondary --id-start-val=1 --id-step-val=1000 prepare**

   NOTE

   Comment out line **db_query(query)** in script **common.lua**.

Step 2 Run the following command to perform stress testing:

sysbench --test=/usr/local/share/sysbench/tests/include/oltp_legacy/oltp.lua --oltp_tables_count=1 --report-interval=1 --oltp-table-size=<*data*> **--mysql-user=**<*user*> **--mysql-password=**<*password*> **--mysql-table-engine=innodb --rand-init=on --mysql-host=<***host***> --mysql-port=5066 --mysql-db=<***db_name***> --max-time=300 --max-requests=0 --oltp_skip_trx=off --oltp_auto_inc=on --oltp_range_size=5 --num-threads=256 --oltp_secondary --forced-shutdown=0 run**

Step 3 Run the following command to delete the test data:

sysbench --test=/usr/local/share/sysbench/tests/include/oltp_legacy/oltp.lua --db-driver=mysql --mysql-db=<*db_name*> **--mysql-user=**<*user*> **--mysql-password=**<*password*> **--mysql-port=5066 --mysql-host=**<*host*> **--oltp-tables-count=64 --oltp-table-size=**<*data*> **--max-time=3600 --max-requests=0 --num-threads=200 cleanup**

----End

Concepts

- Transaction Per Second (TPS) refers to the number of transactions executed per second by a database.
- Query Per Second (QPS) refers to the number of SQL statements, including INSERT, SELECT, UPDATE, and DELETE statements, executed per second.

Test Data

Table 1-1 Required test data

Specifications	Metrics in SysBench OLTP Test			Remarks
DDM	Concurrency	TPS	QPS	
1 x 8 vCPUs | 16 GB	512	2168.748	43389.45	RDS instance specifications: 2 x 8 vCPUs | 16 GB, ultra-high I/O
1 x 16 vCPUs | 32 GB	512	4117.23	82362.996	RDS instance specifications: 2 x 16 vCPUs | 32 GB, ultra-high I/O
1 x 32 vCPUs | 64 GB	1000	5627.754	112585.59	RDS instance specifications: 2 x 32 vCPUs | 64 GB, ultra-high I/O

CAUTION

The DDM kernel has the overload protection capability. If overload occurs, the kernel may trigger a series of actions, for example, attempting to release the SQL statements that have occupied system resources for a long period of time. If overload protection is triggered, a performance bottleneck occurs, and jitter may be found when you observe the stress testing curve. If you continue to increase the pressure, the performance cannot be improved. Try to reduce the pressure to obtain higher performance.