1. **testing environment**

|  |  |
| --- | --- |
| Hardware information | SG2042 |
| architecture | RISC-V64 |
| operating system | openEuler 24.03 (LTS) |
| Python edition | 3.11.6 |
| GCC edition | 12.3.1 |
| G++ edition | 12.3.1 |
| Mysql edition | 8.0.39 |
| Kernel version | 6.6.0 |
| Perf edition | 6.6.0 |

1. **Benchmark tests**
2. **Test function items**
3. Test using MTR framework

# Enter the test directory

|  |
| --- |
| pushd /usr/local/mysql/mysql-test |

# You can execute the test command once

|  |
| --- |
| perl mysql-test-run.pl --nowarnings --force --max-test-fail=0 --timestamp --parallel=5 |

# Shield the failed test items according to the interruption situation of the first test result

# Enter the collections folder and edit the disabled.def file

The format for adding an entry is <testcasename>: BUG#<xxxx> <date disabled> <disabler> <comment>

屏蔽测试项应根据第一次实际测试情况进行.

# Note: These masked test items are marked as diasbled in the second test and will be filtered out after the mysql\_process\_data.py script processes the log file. They need to be manually added to the summary data table.

ARM and RV

|  |
| --- |
| echo "main.component-upgrade : BUG#123 TMP" >> collections/disabled.def  echo "main.log\_tables\_upgrade : BUG#123 TMP" >> collections/disabled.def  echo "main.plugin\_auth : BUG#123 TMP" >> collections/disabled.def  echo "funcs\_2.innodb\_charset : BUG#123 TMP" >> collections/disabled.def  echo "funcs\_2.memory\_charset : BUG#123 TMP" >> collections/disabled.def  echo "funcs\_2.myisam\_charset : BUG#123 TMP" >> collections/disabled.def  echo "rpl.rpl\_spec\_variables\_myisam : BUG#123 TMP" >> collections/disabled.def  echo "rpl.rpl\_charset : BUG#123 TMP" >> collections/disabled.def  echo "rpl.rpl\_drop\_view : BUG#123 TMP" >> collections/disabled.def  echo "sys\_vars.character\_set\_database\_func\_myisam : BUG#123 TMP" >> collections/disabled.def  echo "x.admin\_list\_objects\_docpath : BUG#123 TMP" >> collections/disabled.def |

# Execute the second test instruction

|  |
| --- |
| nohup perl mysql-test-run.pl --nowarnings --force --max-test-fail=0 --timestamp --parallel=5 > mysql-arm.log 2>&1 & |

1. Test instruction parameters

|  |
| --- |
| **--no-warnings: Ignore warnings errors; when warnings errors occur after setting this parameter, the--max-test-fail is no longer accumulated.**  **--force: Ignore the error and continue executing the next case until all cases have been executed and exit.**  **--max-test-fail=0: The test will exit when the number of failed cases reaches a certain value. The default value is 10, and setting it to 0 will ignore the count.**  **--timestamp: Add a timestamp**  **----parallel=5: Specifies the number of parallel threads to run the test case.** |

1. Process test data (mysql\_process\_data.py)

Use Python to process mysql-arm.log log to obtain test item name Y or N test result

|  |
| --- |
| import re  *Open the file and read its contents*  file\_name = mysql-arm.log  with open(fC:/CAS/MySQL/{file\_name}, r, encoding=iso-8859-1) as file:  lines = file.readlines()  *# Store the last test result content of duplicate items*  latest\_results = {}  for line in lines:  parts = line.split()  *# pattern = r\[ (pass|skipped|fail|retry-fail|disabled) \]*  *# Filter skipped and disabled*  pattern = r\[ (pass|fail|retry-fail) \]  match = re.search(pattern, line)  *# Extract test item name and test result*  if match:  *# Special handling of items with non-standard names*  if parts[5] in ["mix", "row", "stmt"]:  test\_name = (parts[4]+" "+parts[5])  else:  test\_name = (parts[4])  test\_result = match.group(1)  *# Update the results of duplicate items*  latest\_results[test\_name] = test\_result  *# Correct test results*  for test\_name, result in latest\_results.items():  if result == retry-fail:  latest\_results[test\_name] = fail  *# Add Y or N according to the result of test\_result*  output\_lines = [f"{test\_name}\t{Y if test\_result == pass else N}\t{test\_result}" for test\_name, test\_result in latest\_results.items()]  *Remove the last line of content*  *# The last line of content is a test completion prompt, not test content*  if output\_lines:  output\_lines.pop()  *Write the results to a new file*  with open(foutput-{file\_name}, w) as file:  for line in output\_lines:  file.write(line + \n)  Print ("Processing is complete and the results have been saved to the file") |

1. **Use benchmarksql to test performance items**
2. Create a test directory and copy the perf performance test script to the test directory

|  |
| --- |
| mkdir -p /home/openeuler/BMarkSQL  cp performance\_counter.sh /home/openeuler/BMarkSQL/ |

1. Download the benchmarksql source code

|  |
| --- |
| git clone https://github.com/jackysp/benchmarksql.git -b 5.0-mysql-support |

1. Modify benchmarksql-5.0/run/runDatabaseBuild.sh to remove extraHistID

|  |
| --- |
| sed -i /AFTER\_LOAD=/s/extraHistID //g ./run/runDatabaseBuild.sh |

1. To add the mysql java connector driver, you need to download it by yourself, decompress it and add it to the lib directory of benchmarksql.

|  |
| --- |
| cd /benchmarksql-5.0/lib  wget https://cdn.mysql.com/archives/mysql-connector-java-8.0/mysql-connector-j-8.0.33.[zip](https://cdn.mysql.com/archives/mysql-connector-java-8.0/mysql-connector-j-8.0.33.zip)  mkdir mysql  unzip mysql-connector-j-8.0.33.zip  cp mysql-connector-j-8.0.33.zip /benchmarksql-5.0/lib/mysql/ |

1. Then compile it with ant.

|  |
| --- |
| [root@localhost benchmarksql-5.0]# ant  Buildfile: /benchmarksql-5.0/build.xml  init:  compile:  [javac] Compiling 11 source files to /benchmarksql-5.0/build  dist:  [mkdir] Created dir: /benchmarksql-5.0/dist  [jar] Building jar: /benchmarksql-5.0/dist/BenchmarkSQL-5.0.jar  BUILD SUCCESSFUL  Total time: 2 seconds |

1. Modify the configuration file

Modify the test configuration file props.mysql and modify the database configuration corresponding to mysql

|  |
| --- |
| db= mysql  driver=org.mysql .jdbc.Driver  conn=jdbc: mysql://localhost:330 9/benchmarksql  //conn=jdbc:mysql://127.0.0.1:3306/test?useSSL=false&useServerPrepStmts=true&useConfigs=maxPerformance&rewriteBatchedStatements=true  user=root  password=open12#$  defaults-file=/etc/ mysql/my.cnf  warehouses=10  loadWorkers=4  terminals=10  // To run specified transactions per terminal- runMins must equal zero  runTxnsPerTerminal=0  // To run for specified minutes- runTxnsPerTerminal must equal zero  runMins= 5  // Number of total transactions per minute  limitTxnsPerMin=10000  // Set to true to run in 4.x compatible mode. Set to false to use the  // entire configured database evenly.  terminalWarehouseFixed=true  // Set to true to use the stored procedure/function implementations. Not  // all of them exist for all databases and the use of stored procedures  // is strongly discouraged for comparing different database vendors as  // they may not have been implemented ideally for all of them. This is  // however useful to test how much network IO can be saved by using  // stored procedures.  useStoredProcedures=true  // The following five values must add up to 100  newOrderWeight=45  paymentWeight=43  orderStatusWeight=4  deliveryWeight=4  stockLevelWeight=4  // Directory name to create for collecting detailed result data.  // Comment this out to suppress.  resultDirectory=my\_result\_%tY-%tm-%td\_%tH%tM%tS  osCollectorScript=./misc/os\_collector\_linux.py  osCollectorInterval=1  //osCollectorSSHAddr=user@dbhost  osCollectorDevices=net\_enp125s0f0 blk\_sda  //osCollectorDevices=net\_eth0 blk\_sda |

explanatory note ：

db= Specifies the database type of mysql. The current type is postgres

driver= org.The JDBC driver for the mysql.jdbc.Driver database

The connection string for the PostgreSQL is conn=jdbc:postgresql://IP:port/database name

user=benchmarksql is the user name to connect to postgres

password=PostgreSQL5432 Password for the user name to connect to postgres

Warehouse = 1 The number of warehouses, each warehouse data volume is about 100MB, so the database size is about 1000MB, default 1 warehouse

loadWorkers=4 The number of processes used to initialize the database data. The default is 4 load loading processes

Terminals=1 specifies the number of terminals, default is 1

runTxnsPerTerminal=10 Specifies the number of transactions to execute per terminal for stress testing. If this parameter is configured to a non-zero value, the runMins parameter below must be set to 0

runMins=0 Specifies the duration of the pressure test (in minutes). If this value is set to a non-zero value, runTxnsPerTerminal must be set to 0.

Limit the total number of transactions per minute to 300. This parameter controls the number of transactions processed per minute, which is affected by the terminals parameter. The value of limitTxnsPerMin/terminals must be a positive integer.

terminalWarehouseFixed=true This setting binds each terminal to a specific warehouse. When set to true, it enables 4.x compatibility mode, meaning each terminal is assigned a fixed warehouse. Setting it to false allows for a more balanced use of the database configuration. According to TPCC, each terminal must be bound to a specific warehouse, so the default value of true is typically used.

The sum of the following five values must equal 100. The default values are: 45,43,4,4,4, which is consistent with the ratio defined by TPC-C test. In actual operation, the proportion can be adjusted to adapt to various scenarios.

NewOrderWeight=45 The new order transaction accounts for 45% of the total transaction

PaymentWeight=43 Payment order transactions account for 43% of the total transactions

orderStatusWeight=4 The order status transaction accounts for 4% of the total transaction

DeliveryWeight=4 The arrival date transaction accounts for 4% of the total transaction

stockLevelWeight=4 The transaction to view current inventory items accounts for 4% of the total transaction

resultDirectory=my\_result\_%tY-%tm-%td\_%tH%tM%tS The directory where system performance data is collected during the stress test (do not change)

osCollectorScript=./misc/os\_collector\_linux.py Operating system performance collection scripts (no modifications required)

osCollectorInterval=1 The operating system collection interval, which is 1 second by default

osCollectorSSHAddr=user@dbhost The host that needs to collect system performance

osCollectorDevices=net\_eth0 blk\_sda The name of the network card and disk name of the server being collected in the operating system, which can be adjusted according to your personal environment

1. Start the database and create the data table to be tested

|  |
| --- |
| mysql -u root -p -h 127.0.0.1 -P 3306  Createdatabase benchmarksql; |

1. Generate test data

|  |
| --- |
| ./runDatabaseBuild.sh props.mysql |

1. Start the test

|  |
| --- |
| ./runBenchmark.sh props.mysql |

1. Clean test data

|  |
| --- |
| ./runDatabaseDestroy.sh props.mysql |

Ensure that the `root` user has access to connect to mysql from `127.0.0.1`. You can check the permissions of the `root` user with the following command:

|  |
| --- |
| SELECT User, Host FROM mysql.user WHERE User=root; |

If the Host column in the result contains only localhost, you may need to add a record that allows the root user to connect from 127.0.0.1:

|  |
| --- |
| GRANT ALL PRIVILEGES ON \*.\* TO root@127.0.0.1 IDENTIFIED BY your\_password WITH GRANT OPTION;  FLUSH PRIVILEGES; |

The above steps are implemented in the script benchmarksql\_test.sh. By executing the script, you can conduct the benmarksql test. The test data is located in the benchmarksql\_report.txt file within the current directory /home/openeuler/BMarkSQL. Additionally, other test item data are available in the /home/openeuler/BMarkSQL/benchmarksql/run/my\_result\_xxx directory, which can be viewed simultaneously.

1. **Use the TPCH test performance item**
2. Download and install the package and decompress the file:

Download https://www.tpc.org/tpc\_documents\_current\_versions/current\_specifications5.asp and upload it to the server.

|  |
| --- |
| unzip 1B81A62A-118E-4CEF-A0A8-671BABA32136-TPC-H-Tool.zip |

(2) Switch directories

|  |
| --- |
| cd TPC-H V3.0.1  cd/dbgen/ |

(3) Backup makefile.suite, or not.

The makefile is mainly used for make compilation

|  |
| --- |
| cp makefile.suite makefile |

1. Modify the makefile

|  |
| --- |
| vim makefile |

Lines 103 to 111 are revised as follows:

|  |
| --- |
| CC = gcc  # Current values for DATABASE are: INFORMIX, DB2, TDAT (Teradata)  # SQLSERVER, SYBASE, ORACLE, VECTORWISE  # Current values for MACHINE are: ATT, DOS, HP, IBM, ICL, MVS,  # SGI, SUN, U2200, VMS, LINUX, WIN32  # Current values for WORKLOAD are: TPCH  DATABASE= MARIADB  MACHINE = LINUX  WORKLOAD = TPCH |

1. Modify tpcd.h (This step is required because the database format is set to mariaDB. If the above supported database format is used, this step is not required)

vim tpcd.h Add to blank space:

|  |
| --- |
| #ifdef MARIADB  #define GEN\_QUERY\_PLAN ""  #define START\_TRAN "START TRANSACTION"  #define END\_TRAN "COMMIT"  #define SET\_OUTPUT ""  #define SET\_ROWCOUNT "limit %d;\n"  #define SET\_DBASE "use %s;\n"  #endif |

1. Compile and generate dbgen

|  |
| --- |
| make |

1. Create the tbl data file

|  |
| --- |
| ./dbgen -s 1 |

-s 1 means that 1G of data is generated. If you have previously generated data, you can clean it up by make clean

1. Modify the initialization script

After compilation, there are two scripts: dss.ddl is the table creation script, and dss.ri is the key-foreign key association script.

Because the TPC-H benchmark itself does not support mariadb, you need to modify these two scripts.

Modify dss.ddl and add the following statement at the beginning of the line.

|  |
| --- |
| drop database tpch;  create database tpch;  use tpch; |

The above steps can be performed by executing the script test-tpch.sh

Modify dss.ri, you can back up a copy and change the content to the following

|  |
| --- |
| ALTER TABLE tpch.PARTSUPP  ADD PRIMARY KEY (PS\_PARTKEY, PS\_SUPPKEY);  COMMIT;  -- For table CUSTOMER  ALTER TABLE tpch.CUSTOMER  ADD PRIMARY KEY (C\_CUSTKEY);  ALTER TABLE tpch.CUSTOMER  ADD FOREIGN KEY CUSTOMER\_FK1 (C\_NATIONKEY) REFERENCES tpch.NATION(N\_NATIONKEY);  COMMIT;  -- For table LINEITEM  ALTER TABLE tpch.LINEITEM  ADD PRIMARY KEY (L\_ORDERKEY, L\_LINENUMBER);  COMMIT;  -- For table ORDERS  ALTER TABLE tpch.ORDERS  ADD PRIMARY KEY (O\_ORDERKEY);  COMMIT;  -- For table PARTSUPP  ALTER TABLE tpch.PARTSUPP  ADD FOREIGN KEY PARTSUPP\_FK1 (PS\_SUPPKEY) REFERENCES tpch.SUPPLIER(S\_SUPPKEY);  ALTER TABLE tpch.PARTSUPP  ADD FOREIGN KEY PARTSUPP\_FK2 (PS\_PARTKEY) REFERENCES tpch.PART(P\_PARTKEY);  COMMIT;  -- For table ORDERS  ALTER TABLE tpch.ORDERS  ADD FOREIGN KEY ORDERS\_FK1 (O\_CUSTKEY) REFERENCES tpch.CUSTOMER(C\_CUSTKEY);  COMMIT;  -- For table LINEITEM  ALTER TABLE tpch.LINEITEM  ADD FOREIGN KEY LINEITEM\_FK1 (L\_ORDERKEY) REFERENCES tpch.ORDERS(O\_ORDERKEY);  ALTER TABLE tpch.LINEITEM  ADD FOREIGN KEY LINEITEM\_FK2 (L\_PARTKEY, L\_SUPPKEY) REFERENCES tpch.PARTSUPP(PS\_PARTKEY, PS\_SUPPKEY);  COMMIT;  -- Rename tables  ALTER TABLE tpch.CUSTOMER RENAME TO customer;  ALTER TABLE tpch.LINEITEM RENAME TO lineitem;  ALTER TABLE tpch.NATION RENAME TO nation;  ALTER TABLE tpch.ORDERS RENAME TO orders;  ALTER TABLE tpch.PART RENAME TO part;  ALTER TABLE tpch.PARTSUPP RENAME TO partsupp;  ALTER TABLE tpch.REGION RENAME TO region;  ALTER TABLE tpch.SUPPLIER RENAME TO supplier;  COMMIT; |

1. Create a table

|  |
| --- |
| cd ../  /usr/local/mariadb/bin/mariadb -u root -pz --socket=/tmp/mariadb.sock  CREATE DATABASE tpch;  USE tpch;  source ./dbgen/dss.ddl; |

Show tables; you can see 8 tables

customer ;

lineitem ;

nation ;

orders ;

part ;

partsupp ;

region ;

supplier ;

1. Establish associations

|  |
| --- |
| source ./dbgen/dss.ri; |

If you want to see whether the association is established successfully, you can view it by showing create table customer\G;

1. Import the tbl script

It is executed under the m ariadb command line. The first seven tables are imported first, and the last table needs to be split and imported after it is too large.

|  |
| --- |
| LOAD DATA INFILE ./dbgen/part.tbl INTO TABLE part FIELDS TERMINATED BY |;  LOAD DATA INFILE ./dbgen/region.tbl INTO TABLE region FIELDS TERMINATED BY | LINES TERMINATED BY \n;  LOAD DATA INFILE ./dbgen/nation.tbl INTO TABLE nation FIELDS TERMINATED BY |;  LOAD DATA INFILE ./dbgen/customer.tbl INTO TABLE customer FIELDS TERMINATED BY |;  LOAD DATA INFILE ./dbgen/supplier.tbl INTO TABLE supplier FIELDS TERMINATED BY | LINES TERMINATED BY \n;  LOAD DATA INFILE ./dbgen/orders.tbl INTO TABLE orders FIELDS TERMINATED BY |;  LOAD DATA INFILE ./dbgen/partsupp.tbl INTO TABLE partsupp FIELDS TERMINATED BY |;  LOAD DATA INFILE ./dbgen/lineitem.tbl INTO TABLE lineitem FIELDS TERMINATED BY |; |

Note that the association relationship is added, and the data is imported in the order of the association relationship.

The last three tables are too large to be imported through a script.

Edit the script

|  |
| --- |
| vi split\_import\_file.sh |

Script content

|  |
| --- |
| #!/bin/bash  # The file name does not include.tbl, that is, the corresponding table name  FileName=lineitem # Modify 1: Replace the file names orders, partsupp, and lineitem  # Get the total number of lines in the original file totalline  totalline=$(cat $filename.tbl | wc -l)  echo totalline=$totalline  # The number of lines in each small file to be split line  line=1000000  a=$(($totalline / $line))  b=$(($totalline % $line))  # Get the number of small files filenum  if (( $b == 0 ))  then  filenum=$a  else  filenum=$(($a + 1))  fi  echo filenum=$filenum  # Perform file segmentation, and the first small file name suffix is i, where i is the minimum value of 1  i=1 # Amend 2: Amend to 1  while (( i <= $filenum ))  do  # The number of lines to be extracted from each small file should be within the range of min and max in the original file  p=$(($i - 1))  min=$(($p \* $line + 1))  max=$(($i \* $line))  sed -n "$min,$max"p ./$filename.tbl > ./$filename.tbl.$i  # Import small files into the database. MariaDB login information and small file path are modified according to actual requirements  /usr/local/mariadb/bin/mariadb -u root -pz --socket=/tmp/mariadb.sock --local-infile -Dtpch -e "load data local infile ./$filename.tbl.$i into table $filename fields terminated by |;"    i=$(($i + 1))  done |

1. View the import results

|  |
| --- |
| SHOW TABLE STATUS FROM tpch; |

1. Run the query statement

Use the qgen tool to generate the sql query and enter the dbgen directory

|  |
| --- |
| cd TPC-HV3.0.1/dbgen |

Copy the qgen and dists.dss files to the queries directory.

|  |
| --- |
| cp qgen queries  cp dists.dss queries |

Create a gen.sh under the queries directory, and the script generates the query SQL.

|  |
| --- |
| #!/usr/bin/bash  for i in {1..22}  do  ./qgen -d $i -s 100 > db"$i".sql  done |

Run gen.sh script.

|  |
| --- |
| chmod +x gen.sh  ./gen.sh |

After the operation is completed, 22 query statement files named d\*.sql are generated in the queries directory

1. entry mysql

Run 22 queries in MySQL.

|  |
| --- |
| use tpch; |
| LOAD DATA INFILE /home/wangtiancheng/TPC-H V3.0.1/dbgen/part.tbl INTO TABLE part FIELDS TERMINATED BY |;  LOAD DATA INFILE /home/wangtiancheng/TPC-H V3.0.1/dbgen/region.tbl INTO TABLE region FIELDS TERMINATED BY | LINES TERMINATED BY \n;  LOAD DATA INFILE /home/wangtiancheng/TPC-H V3.0.1/dbgen/nation.tbl INTO TABLE nation FIELDS TERMINATED BY |;  LOAD DATA INFILE /home/wangtiancheng/TPC-H V3.0.1/dbgen/customer.tbl INTO TABLE customer FIELDS TERMINATED BY |;  LOAD DATA INFILE /home/wangtiancheng/TPC-H V3.0.1/dbgen/supplier.tbl INTO TABLE supplier FIELDS TERMINATED BY | LINES TERMINATED BY \n;  LOAD DATA INFILE /home/wangtiancheng/TPC-H V3.0.1/dbgen/orders.tbl INTO TABLE orders FIELDS TERMINATED BY |;  LOAD DATA INFILE /home/wangtiancheng/TPC-H V3.0.1/dbgen/partsupp.tbl INTO TABLE partsupp FIELDS TERMINATED BY |;  LOAD DATA INFILE /home/wangtiancheng/TPC-H V3.0.1/dbgen/lineitem.tbl INTO TABLE lineitem FIELDS TERMINATED BY |; |

At this point, an error may occur. You need to delete (3) in d1.sql and run it again. Note that you should use the absolute path

1. **Use sysbench to test performance items**
2. Install sysbench:

|  |
| --- |
| curl -s https://packagecloud.io/install/repositories/akopytov/sysbench/script.rpm.sh | sudo bash  sudo yum -y install sysbench |

1. test

After installing sysbench, the /usr/share/sysbench directory contains 11 general Lua test scripts. Each script includes three steps: prepare, run, and cleanup. The prepare step sets up the necessary data for testing, the run step performs the actual testing, and the cleanup step clears the database of the preset data after the test is completed. For example, in oltp\_read\_only.lua, the following commands are used to execute the entire test process, where the run step retrieves the required test data.

|  |
| --- |
| sysbench /usr/share/sysbench/oltp\_read\_only.lua --mysql-host=localhost --mysql-port=3309 --mysql-user=root --mysql-password=open12#$ --mysql-db=sysbenchtest --mysql-socket=/data/mysql/run/mysql.sock --db-driver=mysql --tables=10 --table-size=200000 --report-interval=10 --threads=10 --time=120 prepare  ./performance\_counter.sh "sysbench /usr/share/sysbench/oltp\_read\_only.lua --mysql-host=localhost --mysql-port=3309 --mysql-user=root --mysql-password=open12#$ --mysql-db=sysbenchtest --mysql-socket=/data/mysql/run/mysql.sock --db-driver=mysql --tables=10 --table-size=200000 --report-interval=10 --threads=10 --time=120 run" ./result  sysbench /usr/share/sysbench/oltp\_read\_only.lua --mysql-host=localhost --mysql-port=3309 --mysql-user=root --mysql-password=open12#$ --mysql-db=sysbenchtest --mysql-socket=/data/mysql/run/mysql.sock --db-driver=mysql --tables=10 --table-size=200000 --report-interval=10 --threads=10 --time=120 cleanup |

The corresponding test results are as follows:

|  |
| --- |
| Starting oltp\_read\_only.lua, prepare...  Command: sysbench /usr/share/sysbench/oltp\_read\_only.lua --mysql-host=localhost --mysql-port=3309 --mysql-user=root --mysql-password=open12#$ --mysql-db=sysbenchtest --mysql-socket=/data/mysql/run/mysql.sock --db-driver=mysql --tables=10 --table-size=200000 --report-interval=10 --threads=10 --time=120 prepare  sysbench 1.0.20 (using system LuaJIT 2.1.ROLLING)  Initializing worker threads...  Creating table sbtest7...  Creating table sbtest2...Creating table sbtest4...  Creating table sbtest1...  Creating table sbtest3...  Creating table sbtest5...Creating table sbtest10...  Creating table sbtest9...  Creating table sbtest6...  Creating table sbtest8...  Inserting 200000 records into sbtest10  Inserting 200000 records into sbtest9  Inserting 200000 records into sbtest4  Inserting 200000 records into sbtest2  Inserting 200000 records into sbtest1  Inserting 200000 records into sbtest5  Inserting 200000 records into sbtest6  Inserting 200000 records into sbtest8  Inserting 200000 records into sbtest7  Inserting 200000 records into sbtest3  Creating a secondary index on sbtest3...  Creating a secondary index on sbtest4...  Creating a secondary index on sbtest7...  Creating a secondary index on sbtest10...  Creating a secondary index on sbtest2...  Creating a secondary index on sbtest1...  Creating a secondary index on sbtest8...  Creating a secondary index on sbtest6...  Creating a secondary index on sbtest5...  Creating a secondary index on sbtest9...  Starting oltp\_read\_only.lua, run...  Command: sysbench /usr/share/sysbench/oltp\_read\_only.lua --mysql-host=localhost --mysql-port=3309 --mysql-user=root --mysql-password=open12#$ --mysql-db=sysbenchtest --mysql-socket=/data/mysql/run/mysql.sock --db-driver=mysql --tables=10 --table-size=200000 --report-interval=10 --threads=10 --time=120 run  parameter1=sysbench /usr/share/sysbench/oltp\_read\_only.lua --mysql-host=localhost --mysql-port=3309 --mysql-user=root --mysql-password=open12#$ --mysql-db=sysbenchtest --mysql-socket=/data/mysql/run/mysql.sock --db-driver=mysql --tables=10 --table-size=200000 --report-interval=10 --threads=10 --time=120 run  file name : mysql.sock--db-driver=mysql--tables=10--table-size=200000--report-interval=10--threads=10--time=120run  sysbench 1.0.20 (using system LuaJIT 2.1.ROLLING)  Running the test with following options:  Number of threads: 10  Report intermediate results every 10 second(s)  Initializing random number generator from current time  Initializing worker threads...  Threads started!  [ 10s ] thds: 10 tps: 779.00 qps: 12469.43 (r/w/o: 10910.44/0.00/1558.99) lat (ms,95%): 14.46 err/s: 0.00 reconn/s: 0.00  [ 20s ] thds: 10 tps: 798.50 qps: 12776.73 (r/w/o: 11179.74/0.00/1596.99) lat (ms,95%): 14.46 err/s: 0.00 reconn/s: 0.00  [ 30s ] thds: 10 tps: 809.38 qps: 12952.55 (r/w/o: 11334.00/0.00/1618.56) lat (ms,95%): 14.21 err/s: 0.00 reconn/s: 0.00  [ 40s ] thds: 10 tps: 813.72 qps: 13019.67 (r/w/o: 11392.04/0.00/1627.63) lat (ms,95%): 13.22 err/s: 0.00 reconn/s: 0.00  [ 50s ] thds: 10 tps: 821.59 qps: 13146.74 (r/w/o: 11503.56/0.00/1643.18) lat (ms,95%): 13.70 err/s: 0.00 reconn/s: 0.00  [ 60s ] thds: 10 tps: 855.48 qps: 13687.19 (r/w/o: 11976.23/0.00/1710.96) lat (ms,95%): 12.98 err/s: 0.00 reconn/s: 0.00  [ 70s ] thds: 10 tps: 843.55 qps: 13495.73 (r/w/o: 11808.74/0.00/1686.99) lat (ms,95%): 14.21 err/s: 0.00 reconn/s: 0.00  [ 80s ] thds: 10 tps: 836.55 qps: 13384.83 (r/w/o: 11711.62/0.00/1673.20) lat (ms,95%): 14.21 err/s: 0.00 reconn/s: 0.00  [ 90s ] thds: 10 tps: 836.82 qps: 13391.52 (r/w/o: 11717.88/0.00/1673.64) lat (ms,95%): 14.46 err/s: 0.00 reconn/s: 0.00  [ 100s ] thds: 10 tps: 828.55 qps: 13253.17 (r/w/o: 11596.17/0.00/1657.01) lat (ms,95%): 14.46 err/s: 0.00 reconn/s: 0.00  [ 110s ] thds: 10 tps: 809.48 qps: 12953.19 (r/w/o: 11334.13/0.00/1619.06) lat (ms,95%): 15.00 err/s: 0.00 reconn/s: 0.00  [ 120s ] thds: 10 tps: 828.75 qps: 13260.82 (r/w/o: 11603.42/0.00/1657.40) lat (ms,95%): 14.46 err/s: 0.00 reconn/s: 0.00  SQL statistics:  queries performed:  read: 1380848  write: 0  other: 197264  total: 1578112  transactions: 98632 (821.71 per sec.)  queries: 1578112 (13147.29 per sec.)  ignored errors: 0 (0.00 per sec.)  reconnects: 0 (0.00 per sec.)  General statistics:  total time: 120.0260s  total number of events: 98632  Latency (ms):  min: 10.13  avg: 12.16  max: 30.12  95th percentile: 14.21  sum: 1199193.76  Threads fairness:  events (avg/stddev): 9863.2000/369.51  execution time (avg/stddev): 119.9194/0.01  Avg 10 times duration time: 120122118277  Avg 10 times task clock: 290154.510  Avg 10 times cpu-cycles: 558751145155  Avg 10 times instructions: 232440917845  Avg 10 times cache references: 5775461543  Avg 10 times cache misses: 5775462232  Avg 10 times branches: 53370981978  Avg 10 times branch misses: 888115531  Avg 10 times L1 dcache loads: 46548985366  Avg 10 times L1 dcache load misses: 967612852  Avg 10 times LLC load misses: 1196907642  Avg 10 times LLC load: 5775465135  Avg 10 times IPC: 0.416  120122118277  290154.51  558751145155  232440917845  5775461543  5775462232  53370981978  888115531  46548985366  967612852  1196907642  5775465135  0.416  Starting oltp\_read\_only.lua, cleanup...  Command: sysbench /usr/share/sysbench/oltp\_read\_only.lua --mysql-host=localhost --mysql-port=3309 --mysql-user=root --mysql-password=open12#$ --mysql-db=sysbenchtest --mysql-socket=/data/mysql/run/mysql.sock --db-driver=mysql --tables=10 --table-size=200000 --report-interval=10 --threads=10 --time=120 cleanup  sysbench 1.0.20 (using system LuaJIT 2.1.ROLLING)  Dropping table sbtest1...  Dropping table sbtest2...  Dropping table sbtest3...  Dropping table sbtest4...  Dropping table sbtest5...  Dropping table sbtest6...  Dropping table sbtest7...  Dropping table sbtest8...  Dropping table sbtest9...  Dropping table sbtest10... |