Open Gauss test document

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 |
| Openauss version | 5.1.0 |
| Kernel version | 6.6.0 |
| Perf edition | 6.6.0 |

## Test preparation-benchmarksql

Benchmark address: https://gitee.com/YH0204/shlibs

Branch: master

Commit ID: f4ebb181fb7dd70d4cab11566fbd19daab5c043c

The specific operation is as follows:

1. Download the TPCC support package:

wget https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.0.0/MOT-TPCC-Benchmark.tar.gz

1. Replace jdbc support package:

git clone https://gitee.com/opengauss/openGauss-connector-jdbc.git

cd ./openGauss-connector-jdbc/

git checout 5.1.0

sh https://build.sh

After compilation, the package: openGauss-5.0.0-OPENEULER-64bit-Jdbc.tar.gz appears in the current directory. Replace this support package with the PostgreSQL support package under /lib/postgres in the benchmar directory.

cp openGauss-5.0.0-OPENEULER-64bit-Jdbc.tar.gz ../benchmarsql-5.0/lib/postgres

tar -xvf openGauss-5.0.0-OPENEULER-64bit-Jdbc.tar.gz

cd ../../

ant

1. Modify the benchmark configuration

Enter benchmarksql-5.0/run, modify props.pg, and configure as follows:

db=postgres

driver=org.postgresql.Driver

conn=jdbc:postgresql://localhost:5432/tpcc1000?prepareThreadshold=1&batchMode=on&fetchsize=10

user=bot

password=GS.initdb

warehouses=8

loadWorkers=40

terminals=37

//To run specified transactions per terminal- runMins must equal zero

runTxnsPerTerminal=10

//To run for specified minutes- runTxnsPerTerminal must equal zero

runMins=0

//Number of total transactions per minute

limitTxnsPerMin=300

//Set to true to run in 4.x compatible mode. Set to false to use the

//entire configured database evenly.

terminalWarehouseFixed=true

//The following five values must add up to 100

//The default percentages of 45, 43, 4, 4 & 4 match the TPC-C spec

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

1. Import test data:

Modify the tableCreates.sql file under benchmark-5.0/run/sql.common/ directory.

CREATE TABLESPACE example2 relative location tablespace2;

CREATE TABLESPACE example3 relative location tablespace3;

create table bmsql\_config (

cfg\_name varchar(30),

cfg\_value varchar(50)

);

create table bmsql\_warehouse (

w\_id integer not null,

w\_ytd decimal(12,2),

w\_tax decimal(4,4),

w\_name varchar(10),

w\_street\_1 varchar(20),

w\_street\_2 varchar(20),

w\_city varchar(20),

w\_state char(2),

w\_zip char(9)

) WITH (FILLFACTOR=80);

create table bmsql\_district (

d\_w\_id integer not null,

d\_id integer not null,

d\_ytd decimal(12,2),

d\_tax decimal(4,4),

d\_next\_o\_id integer,

d\_name varchar(10),

d\_street\_1 varchar(20),

d\_street\_2 varchar(20),

d\_city varchar(20),

d\_state char(2),

d\_zip char(9)

) WITH (FILLFACTOR=80);

create table bmsql\_customer (

c\_w\_id integer not null,

c\_d\_id integer not null,

c\_id integer not null,

c\_discount decimal(4,4),

c\_credit char(2),

c\_last varchar(16),

c\_first varchar(16),

c\_credit\_lim decimal(12,2),

c\_balance decimal(12,2),

c\_ytd\_payment decimal(12,2),

c\_payment\_cnt integer,

c\_delivery\_cnt integer,

c\_street\_1 varchar(20),

c\_street\_2 varchar(20),

c\_city varchar(20),

c\_state char(2),

c\_zip char(9),

c\_phone char(16),

c\_since timestamp,

c\_middle char(2),

c\_data varchar(500)

) WITH (FILLFACTOR=80) tablespace example2;

create sequence bmsql\_hist\_id\_seq;

create table bmsql\_history (

hist\_id integer,

h\_c\_id integer,

h\_c\_d\_id integer,

h\_c\_w\_id integer,

h\_d\_id integer,

h\_w\_id integer,

h\_date timestamp,

h\_amount decimal(6,2),

h\_data varchar(24)

) WITH (FILLFACTOR=80);

create table bmsql\_new\_order (

no\_w\_id integer not null,

no\_d\_id integer not null,

no\_o\_id integer not null

) WITH (FILLFACTOR=80);

create table bmsql\_oorder (

o\_w\_id integer not null,

o\_d\_id integer not null,

o\_id integer not null,

o\_c\_id integer,

o\_carrier\_id integer,

o\_ol\_cnt integer,

o\_all\_local integer,

o\_entry\_d timestamp

) WITH (FILLFACTOR=80);

create table bmsql\_order\_line (

ol\_w\_id integer not null,

ol\_d\_id integer not null,

ol\_o\_id integer not null,

ol\_number integer not null,

ol\_i\_id integer not null,

ol\_delivery\_d timestamp,

ol\_amount decimal(6,2),

ol\_supply\_w\_id integer,

ol\_quantity integer,

ol\_dist\_info char(24)

) WITH (FILLFACTOR=80);

create table bmsql\_item (

i\_id integer not null,

i\_name varchar(24),

i\_price decimal(5,2),

i\_data varchar(50),

i\_im\_id integer

);

create table bmsql\_stock (

s\_w\_id integer not null,

s\_i\_id integer not null,

s\_quantity integer,

s\_ytd integer,

s\_order\_cnt integer,

s\_remote\_cnt integer,

s\_data varchar(50),

s\_dist\_01 char(24),

s\_dist\_02 char(24),

s\_dist\_03 char(24),

s\_dist\_04 char(24),

s\_dist\_05 char(24),

s\_dist\_06 char(24),

s\_dist\_07 char(24),

s\_dist\_08 char(24),

s\_dist\_09 char(24),

s\_dist\_10 char(24)

) WITH (FILLFACTOR=80) tablespace example3;

1. Create a database user and change the password at xxxxxxxxx to your own password:

create user bot identified by XXXXXXXX profile default;

alter user bot sysadmin;

create database tpcc1000 encoding UTF8 template=template0 owner tpcc5q;

1. Import data and run test scripts:

./runDatabaseBuild.sh props.opengauss.1000

./runBenchmark.sh props.opengauss.1000

1. perf test

./performance\_counter\_920.sh "./runBenchmark.sh props.opengauss.1000

" ./result/

## Test preparation-sysbench

1. Get sysbench

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

sudo yum -y install sysbench

1. Run sysbench. Here, bulk\_insert is used as an example. You need to modify the host, port, user, password, and db of the following configuration items. After modification, run prepare to generate data and then run to complete the test.

sysbench /usr/share/sysbench/bulk\_insert.lua \

-- --db-driver=pgsql --pgsql-host=127.0.0.1 --pgsql-port=5432 \

--pgsql-user=openeuler --pgsql-password=GS.initdb \

--pgsql-db=postgres --tables=10 --table-size=1000000 \

--threads=4 --time=300 --report-interval=10 \

prepare

sysbench /usr/share/sysbench/bulk\_insert.lua \

--db-driver=pgsql --pgsql-host=127.0.0.1 --pgsql-port=5432 \

--pgsql-user=openeuler --pgsql-password=GS.initdb \

--pgsql-db=postgres --tables=10 --table-size=1000000 \

--threads=4 --time=300 --report-interval=10 \

run

1. Perf tests performance

./performance\_counter\_920.sh “sysbench /usr/share/sysbench/bulk\_insert.lua \

-- --db-driver=pgsql --pgsql-host=127.0.0.1 --pgsql-port=5432 \

--pgsql-user=openeuler --pgsql-password=GS.initdb \

--pgsql-db=postgres --tables=10 --table-size=1000000 \

--threads=4 --time=300 --report-interval=10 \

run“ ./result/