云数据库 GaussDB

性能白皮书

文档版本 01

发布日期 2024-10-12

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 测试方法

本章提供GaussDB使用BenchmarkSQL进行性能测试的方法和测试数据报告。

BenchmarkSQL，一个JDBC基准测试工具，内嵌了TPC-C测试脚本，支持很多数据库，如PostgreSQL、Oracle和Mysql等。

TPC-C是专门针对联机交易处理系统（OLTP系统）的规范，一般情况下我们也把这类系统称为业务处理系统。几乎所有在OLTP市场提供软硬平台的国外主流厂商都发布了相应的TPC-C测试结果，随着计算机技术的不断发展，这些测试结果也在不断刷新。

测试环境

- JDK：建议使用1.8及以上版本。
- Ant：建议使用apache-ant-1.10及以上版本。
- Benchmark SQL：建议使用 Benchmark SQL 5.0。
- ECS客户端：建议使用32U64GB及以上规格。
- 数据库：建议提前准备GaussDB数据库，并同时创建压测数据库$db_name和压测用户$db_user。

安装 Benchmark SQL

1. 下载安装包。
   ```
   wget https://sourceforge.net/projects/benchmarksql/files/latest/download
   ```
2. 解压安装包。
   ```
   unzip download -d /home
   ```
3. 进入Benchmark SQL解压后的目录，使用ant编译 Benchmark SQL。
   ```
   cd /home/benchmarksql-5.0
   ant
   ```
   返回如下信息，表示编译成功。
   ```
   [root@test benchmarksql-5.0]# ant
   Buildfile: /tools/benchmarksql-5.0/build.xml

   init:
       [mkdir] Created dir: /tools/benchmarksql-5.0/build

   compile:
       [javac] Compiling 11 source files to /tools/benchmarksql-5.0/build

   dist:
       [mkdir] Created dir: /tools/benchmarksql-5.0/dist
         [jar] Building jar: /tools/benchmarksql-5.0/dist/BenchmarkSQL-5.0.jar

   BUILD SUCCESSFUL
   Total time: 1 second
   ```

GaussDB 适配 Benchmark SQL

1. 更换驱动包。

   请根据实际的数据库引擎版本、实例类型和操作系统获取对应的**驱动包**，此处以获取分布式V2.0-8.x实例、Euler2.5操作系统对应的驱动包为例。

   ```
   cd /home
   wget https://dbs-download.obs.cn-north-1.myhuaweicloud.com/GaussDB/1716897684140/GaussDB_driver.zip
   unzip GaussDB_driver.zip
   cd GaussDB_driver/Distributed/Euler2.5_X86_64
   tar -zxvf GaussDB-Kernel_505.1.0_Euler_64bit_Jdbc.tar.gz
   rm -rf /home/benchmarksql-5.0/lib/postgres/postgresql-9.3-1102.jdbc41.jar
   cp -rp gsjdbc4.jar /home/benchmarksql-5.0/lib/postgres/
   ```

2. 修改配置文件。

   备份并重写/home/benchmarksql-5.0/run/props.pg。

   ```
   db=postgres
   driver=org.postgresql.Driver
   //集中式版
   //conn=jdbc:postgresql://$host_ip:$host_port/$db_name?targetServerType=master&loggerLevel=OFF
   //分布式版
   conn=jdbc:postgresql://$host_ip1:$host_port1,host_ip2:$host_port2,host_ip3:$host_port3/$db_name?autoBalance=true&loggerLevel=OFF
   user=$db_user
   password=$db_user_passwd
   warehouses=10
   loadWorkers=10
   terminals=5
   //To run specified transactions per terminal- runMins must equal zero
   runTxnsPerTerminal=0
   //To run for specified minutes- runTxnsPerTerminal must equal zero
   runMins=3
   //Number of total transactions per minute
   limitTxnsPerMin=0
   //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_eth0 blk_sda
   ```

说明

prop.pg中的关键参数说明：

- conn：JDBC连接串，请根据实际的实例类型选择对应的连接串。
  - 集中式版：conn=jdbc:postgresql://$host_ip:$host_port/$db_name?targetServerType=master&loggerLevel=OFF
  - 分布式版：conn=jdbc:postgresql://$host_ip1:$host_port1,host_ip2:$host_port2,host_ip3:$host_port3/$db_name?autoBalance=true&loggerLevel=OFF
- warehouses/loadWorkers：用于设置导入的数据量，可以适当调整。
- terminals/runMins：用于设置压测的并发数和压测时长，可以适当调整。

  Terminals需要满足：0<terminals num<=10* warehouses num，否则可能出现报错。
- osCollector相关参数可根据实际情况选择是否需要注释。

3. 改造 BenchMarkSQL5中的SQL。

   a. 新建/home/benchmarksql-5.0/run/sql.postgres/tableCreates.sql。

   - 集中式版建表语句：

```
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) ;

create sequence bmsql_hist_id_seq cache 1000;

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) ;
```

- 分布式建表语句：

```
create table bmsql_config (
        cfg_name    varchar(30),
        cfg_value   varchar(50)
) DISTRIBUTE BY REPLICATION;

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) DISTRIBUTE BY hash(w_id);

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) DISTRIBUTE BY hash(d_w_id);

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) DISTRIBUTE BY hash(c_w_id);

create sequence bmsql_hist_id_seq cache 1000;
```



```
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) DISTRIBUTE BY hash(h_w_id);

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) DISTRIBUTE BY hash(no_w_id);

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) DISTRIBUTE BY hash(o_w_id);

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) DISTRIBUTE BY hash(ol_w_id);

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
) DISTRIBUTE BY REPLICATION;

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) DISTRIBUTE BY hash(s_w_id);
```

b. 新建/home/benchmarksql-5.0/run/sql.postgres/tableDrops.sql。

```
drop table bmsql_config;
drop table bmsql_new_order;
drop table bmsql_order_line;
drop table bmsql_oorder;
drop table bmsql_history;
drop table bmsql_customer;
drop table bmsql_stock;
drop table bmsql_item;
drop table bmsql_district;
drop table bmsql_warehouse;
drop sequence bmsql_hist_id_seq;
```

c. 新建/home/benchmarksql-5.0/run/sql.postgres/indexCreates.sql。

```
set maintenance_work_mem='4GB';
alter table  bmsql_warehouse add constraint bmsql_warehouse_pkey
  primary key (w_id);

alter table  bmsql_district add constraint bmsql_district_pkey
  primary key (d_w_id, d_id);

alter table  bmsql_customer add constraint bmsql_customer_pkey
  primary key (c_w_id, c_d_id, c_id);

alter table  bmsql_oorder add constraint bmsql_oorder_pkey
  primary key (o_w_id, o_d_id, o_id);

alter table  bmsql_new_order add constraint bmsql_new_order_pkey
  primary key (no_w_id, no_d_id, no_o_id);

alter table  bmsql_order_line add constraint bmsql_order_line_pkey
  primary key (ol_w_id, ol_d_id, ol_o_id, ol_number);

alter table  bmsql_stock add constraint bmsql_stock_pkey
  primary key (s_w_id, s_i_id);

alter table  bmsql_item add constraint bmsql_item_pkey
  primary key (i_id);

create index bmsql_oorder_idx1 on  bmsql_oorder(o_w_id, o_d_id, o_c_id, o_id);

create index bmsql_customer_idx1
  on   bmsql_customer (c_w_id, c_d_id, c_last, c_first);
```

d. 新建/home/benchmarksql-5.0/run/sql.postgres/indexDrops.sql。

```
set statement_timeout=0;
set maintenance_work_mem='4GB';
alter table  bmsql_warehouse drop constraint bmsql_warehouse_pkey;
alter table  bmsql_district drop constraint bmsql_district_pkey;
alter table  bmsql_customer drop constraint bmsql_customer_pkey;
drop index bmsql_customer_idx1;
alter table  bmsql_oorder drop constraint bmsql_oorder_pkey;
drop index bmsql_oorder_idx1;
drop index bmsql_oorder_idx2;
alter table  bmsql_new_order drop constraint bmsql_new_order_pkey;
alter table  bmsql_order_line drop constraint bmsql_order_line_pkey;
alter table  bmsql_stock drop constraint bmsql_stock_pkey;
alter table  bmsql_item drop constraint bmsql_item_pkey;
alter table bmsql_history drop constraint bmsql_history_pkey;
```

4. 更新./runDatabaseBuild.sh。

   执行以下命令打开“runDatabaseBuild.sh”文件。

   ```
   vim /home/benchmarksql-5.0/run/runDatabaseBuild.sh
   ```

   将光标移至AFTER_LOAD="indexCreates foreignKeys extraHistID buildFinish"中foreignKeys的第一个字符，按“x”键依次删除foreignKeys及其后边的空格，输入“:wq”命令保存并退出。删除后的结果如下。


```
AFTER_LOAD="indexCreates extraHistID buildFinish"
```

运行 Benchmark SQL

1. 导入数据。
   ```
   ./runDatabaseBuild.sh props.pg
   ```
2. 进行压测。
   ```
   ./runBenchmark.sh props.pg
   ```

   注意

   请根据实际需要进行参数调优。

3. 删除数据。
   ```
   ./runDatabaseDestroy.sh props.pg
   ```

2 测试数据

关于 IOPS

GaussDB支持的IOPS取决于云硬盘（Elastic Volume Service，简称EVS）的IO性能，具体请参见《云硬盘产品介绍》中“**磁盘类型及性能介绍**”的内容。

测试数据

1. 实例类型：分布式版，集中式版。
2. 实例规格：16U128GB和32U256GB等。
3. 集群规模：分布式版：3CN，3分片，3副本；集中式版：1主2备。
4. 数据量：1000wh。
5. 压测时长：30min（预热5min）。

表 2-1 性能数据

实例类型	部署形态	规格	并发数	tpmC
集中式版	高可用（1主2备）	16U128G	256	70057
		32U256G	384	132196
分布式版	独立部署	16U128G	1024	466930
		32U256G	2048	658805

测试指标

流量指标(Throughput，简称tpmC)：按照TPC组织的定义，流量指标描述了系统在执行支付操作、订单状态查询、发货和库存状态查询这4种交易的同时，每分钟可以处理多少个新订单交易。所有交易的响应时间必须满足TPC-C测试规范的要求，且各种交易数量所占的比例也应该满足TPC-C测试规范的要求。在这种情况下，流量指标值越大说明系统的联机事务处理能力越高。