GaussDB

性能白皮书

文档版本 01

发布日期 2023-02-08

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。

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

- 局点：华为云。
- 实例类型：分布式，主备版。
- 规格选择：16U128G和32U256G。
- 集群规模：分布式：3CN，3分片，3副本；主备版：1主2备。

测试方法

1. 修改连接配置。

   配置文件所在目录为：./run/props.pg

2. 重点参数修改。

```
//连接配置
conn=jdbc:postgresql://127.0.0.1:8000/postgres?autoBalance=true
//连接用户名
user=****
//连接密码
password=****
//压入数据量
warehouses=1000
//压入并发
loadWorkers=10
//业务并发
terminals=2048
//运行时间
runMins=30
```

3. 压数据

   cd ~/BenchmarkSQL-5.0/run

   ./runDatabaseBuild.sh props.pg

4. 运行tpcc业务场景
   cd ~/BenchmarkSQL-5.0/run
   ./runBenchmark.sh props.pg

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);
```


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。

2 测试数据

1. 实例类型：分布式，主备版。
2. 实例规格：16U128G和32U256G。
3. 集群规模：分布式：3CN，3分片，3副本；主备版：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
主备版	高可用（1主2备）	16U128G	256	70057
		32U256G	384	132196
分布式	独立部署	16U128G	1024	466930
		32U256G	2048	658805