

# 实验三报告

## 关卡一：openGauss 数据库的编译和安装

### 1. 关卡验证

步骤 1 首先需要对数据库状态进行验证。

```
[omm@opengauss01 openGauss-server]$ gs_ctl status
```

(截图语句和执行结果)

```
[2022-12-06 14:55:02.072][225310][][gs_ctl]: gs_ctl status,datadir is /opt/software/openGauss/data  
gs_ctl: server is running (PID: 225270)  
/opt/software/openGauss/bin/gaussdb "-D" "/opt/software/openGauss/data"
```

步骤 2 对数据库进程进行截图验证，需包含数据库服务器的主机名。

```
[omm@opengauss01 openGauss-server]$ ps -ef|grep omm
```

(截图语句和执行结果)

```
[omm@opengauss01 ~]$ ps -ef|grep omm  
root      225075      5176    0 14:50 pts/0    00:00:00 su - omm  
omm       225076      225075    0 14:50 pts/0    00:00:00 -bash  
omm       225270        1    1 14:54 pts/0    00:00:01 /opt/software/openGauss/bin/gaussdb -D /opt/software/openGauss/data  
omm       225317      225076    0 14:55 pts/0    00:00:00 ps -ef  
omm       225318      225076    0 14:55 pts/0    00:00:00 grep --color=auto omm
```

## 关卡二：openGauss 数据导入及基本操作

### 1. 关卡验证

步骤 12 登录数据库验证

```
[omm@opengauss01 dbgen]$ gsql -d tpch -p 5432 -r
```

```
tpch=# select count(*) from supplier;
```

(截图语句和执行结果)

```
[omm@opengauss01 dbgen]$ gsql -d tpch -p 5432 -r  
gsql ((GaussDB Kernel V500R002C00 build b2ff10be) compiled at 2022-12-06 14:33:15 commit 0 last mr debug)  
Non-SSL connection (SSL connection is recommended when requiring high-security)  
Type "help" for help.  
  
tpch=# select count(*) from supplier;  
count  
-----  
10000  
(1 row)
```

步骤 21 登录数据库进行验证

```
[omm@opengauss01 ~]$ gsql -d tpch -p 5432 -r
```

```
tpch=# \dt
```

(截图语句和执行结果)

```
[omm@opengauss01 ~]$ gsql -d tpch -p 5432 -r
gsql ((GaussDB Kernel V500R002C00 build b2ff10be) compiled at 2022-12-06 14:33:15 commit 0 last mr debug)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

tpch=# \dt
          List of relations
Schema |      Name      | Type | Owner | Storage
-----+-----+-----+-----+-----
public | address_dimension | table | omm    | {orientation=row,compression=no}
public | customer        | table | omm    | {orientation=row,compression=no}
public | date_dimension  | table | omm    | {orientation=row,compression=no}
public | lineitem        | table | omm    | {orientation=row,compression=no}
public | littell_orders  | table | omm    | {orientation=row,compression=no}
public | nation          | table | omm    | {orientation=row,compression=no}
public | orders          | table | omm    | {orientation=row,compression=no}
public | part            | table | omm    | {orientation=row,compression=no}
public | partsupp        | table | omm    | {orientation=row,compression=no}
public | region          | table | omm    | {orientation=row,compression=no}
public | supplier        | table | omm    | {orientation=row,compression=no}
public | user_dimension  | table | omm    | {orientation=row,compression=no}
(12 rows)
```

步骤 22 查询 customer 表的数据

```
tpch=# select * from customer limit 10;
```

(截图语句和执行结果)

```
tpch=# select * from customer limit 10;
 c_custkey |      c_name      |      c_address      | c_nationkey | c_phone | c_acctbal | c_mktsegment |      c_c
-----+-----+-----+-----+-----+-----+-----+-----
1 | Customer#000000001 | IVhziApeRb ot,c,E | 15 | 25-989-741-2988 | 711.56 | BUILDING | I to the even, regular platelets. regular, ironic epitaph
2 | Customer#000000002 | XSTf4,NCwDVaWNe6tEgvrfaRchXak | 13 | 23-768-687-3665 | 121.65 | AUTOMOBILE | I l accounts. blithely ironic theodolites integrate boldl
3 | Customer#000000003 | M09kdTD2V8Hm | 1 | 11-719-748-3364 | 7498.12 | AUTOMOBILE | I deposits eat slyly ironic, even instructions. express
4 | Customer#000000004 | XxVSJSLAgtN | 4 | 14-128-198-5944 | 2866.83 | MACHINERY | I requests. final, regular ideas sleep final accou
5 | Customer#000000005 | KvpvUHCpLr98AwqA10VesTp2q7I | 3 | 13-758-942-6364 | 794.47 | HOUSEHOLD | I n accounts will have to unwind. foxes cajole accor
6 | Customer#000000006 | xZ2z9CmMO7nq4Kd9rBvx,LRZIKUMah yVn | 20 | 38-114-988-4951 | 7638.57 | AUTOMOBILE | I tions. even deposits boost according to the slyly bold
7 | Customer#000000007 | Tc0e5GaZNgVePxU5KrrvX8fkasDtea | 18 | 28-198-982-9759 | 9561.95 | AUTOMOBILE | I ainst the ironic, express theodolites. express, even pi
8 | Customer#000000008 | I8B18b88Aymc, 0PrY8CP1yG38xc8PmwhL5 | 17 | 27-147-574-9335 | 6819.74 | BUILDING | I among the slyly regular theodolites kindle blithely cou
9 | Customer#000000009 | x5IAFTJUsCuxfeteNqefumTrJs | 8 | 18-338-986-3675 | 8324.87 | FURNITURE | I r theodolites according to the requests wake thinly exc
10 | Customer#000000010 | 6LrEaV8KR6PLVcgl2ArL Q3rqzLzcT1 v2 | 5 | 15-741-346-9878 | 2753.54 | HOUSEHOLD | I es regular deposits haggle. fur
(10 rows)
```

## 2. 思考题

数据初始化中出现了 TPC-H，这是什么？

TPC-H（商业智能计算测试） 是美国交易处理效能委员会（TPC,Transaction Processing Performance Council）组织制定的用来模拟决策支持类应用的一个测试集。目前，在学术界和工业界普遍用来评价决策支持技术方面应用的性能。

## 关卡三： openGauss 的 AI4DB 特性应用

### 1. 关卡验证

(1) 使用 X-Tuner 进行参数优化

步骤 2 在原来 CloudShell 连接窗口中查看 queries01.log。

```
[omm@opengauss01 ~]$ tail -10 /opt/software/tpch-kit/dbgen/queries/queries01.log
```

(截图执行语句和结果)

```
[omm@opengauss01 ~]$ tail -10 /opt/software/tpch-kit/dbgen/queries/queries01.log
13 |      | 888 | 6737713.99
17 |      | 861 | 6460573.72
18 |      | 964 | 7236687.40
23 |      | 892 | 6701457.95
29 |      | 948 | 7158866.63
30 |      | 909 | 6808436.13
31 |      | 922 | 6806670.18
(7 rows)

total time: 1208295  ms
```

步骤 3 切换至 root 用户，执行 X-Tuner 进行参数建议优化

```
[omm@opengauss01 ~]$ exit
[root@opengauss01 ~]# gs_xtuner recommend --db-name tpch --db-user omm --port 5432
--host 127.0.0.1 --host-user omm
```

(截图执行语句和结果)

name	recommend	min	max	restart
default_statistics_target	1000	100	1000	False
effective_cache_size	21602334	186752	21602334	False
effective_io_concurrency	200	150	250	False
enable_mergejoin	off	0	1	False
enable_nestloop	off	0	1	False
max_connections	370	50	741	True
max_prepared_transactions	370	50	741	True
max_process_memory	28803112	22402420	28803112	True
random_page_cost	1.0	1.0	2.0	False
shared_buffers	186752	186756	214768	True
wal_buffers	5836	2048	5836	True

步骤 6 获取参数值

```
[omm@opengauss01 ~]$ cd /opt/software/openGauss/data
[omm@opengauss01 data]$ cat postgresql.conf|grep -E
'shared_buffers|max_connections|effective_cache_size|effective_io_concurrency|wal_buffers|rando
m_page_cost|default_statistics_target'
```

(截图执行语句和结果)

```
omm@opengauss01 data$ cat postgresql.conf|grep -E 'shared_buffers|max_connections|effective_cache_size|effective_io_concurrency|wal_buffers|random_page_cost|default_statistics_target'
max_connections = 370                                # (change requires restart)
# Note: Increasing max_connections costs ~400 bytes of shared memory per
shared_buffers = 187398                               # min 128kB
bulk_write_ring_size = 260                            # for bulkload, max shared_buffers
#standby_shared_buffers_fraction = 0.3 #control shared buffers use in standby, 0.1-1.0
effective_io_concurrency = 200                        # 1-1000; 0 disables prefetching
wal_buffers = 5855                                    # min 32kB
random_page_cost = 1                                  # same scale as above
effective_cache_size = 21602948                       # range 1-10000
default_statistics_target = 1000                      # max locks per transaction * (max_connections + max prepared transactions)
```

步骤 7 再次执行步骤 2，对比优化前的执行时间。

(截图执行语句和结果)

```
[omm@opengauss01 ~]$ tail -10 /opt/software/tpch-kit/dbgen/queries/queries01.log
13 |      | 888 | 6737713.99
17 |      | 861 | 6460573.72
18 |      | 964 | 7236687.40
23 |      | 892 | 6701457.95
29 |      | 948 | 7158866.63
30 |      | 909 | 6808436.13
31 |      | 922 | 6806670.18
(7 rows)

total time: 1171820  ms
```

步骤 8 【附加题】有兴趣的同学可以尝试并截图记录于此。

(截图执行语句和结果)

## (2) Index-advisor: 索引推荐

步骤 4 使用 explain, 对该 SQL 加以分析

```
tpch=# EXPLAIN
SELECT ad.province AS province, SUM(o.actual_price) AS GMV
FROM litemall_orders o,
     address_dimension ad,
     date_dimension dd
WHERE o.address_key = ad.address_key
     AND o.add_date = dd.date_key
     AND dd.year = 2020
     AND dd.month = 3
GROUP BY ad.province
ORDER BY SUM(o.actual_price) DESC;
```

(截图执行语句和结果)

```
tpch=# EXPLAIN
tpch=# SELECT ad.province AS province, SUM(o.actual_price) AS GMV
tpch=# FROM litemall_orders o,
tpch=# address_dimension ad,
tpch=# date_dimension dd
tpch=# WHERE o.address_key = ad.address_key
tpch=# AND o.add_date = dd.date_key
tpch=# AND dd.year = 2020
tpch=# AND dd.month = 3
tpch=# GROUP BY ad.province
tpch=# ORDER BY SUM(o.actual_price) DESC;

QUERY PLAN

-----
Sort (cost=4593.80..4593.88 rows=31 width=47)
Sort Key: (sum(o.actual_price)) DESC
-> HashAggregate (cost=4592.72..4593.03 rows=31 width=47)
    Group By Key: ad.province
    -> Hash Join (cost=4354.43..4585.97 rows=1351 width=15)
        Hash Cond: (ad.address_key = o.address_key)
        -> Seq Scan on address_dimension ad (cost=0.00..188.02 rows=8002 width=14)
        -> Hash (cost=4337.54..4337.54 rows=1351 width=9)
            -> Hash Join (cost=1031.78..4337.54 rows=1351 width=9)
                Hash Cond: (o.add_date = dd.date_key)
                -> Seq Scan on litemall_orders o (cost=0.00..3041.00 rows=100000 width=13)
                -> Hash (cost=1031.76..1031.76 rows=2 width=4)
                    -> Seq Scan on date_dimension dd (cost=0.00..1031.76 rows=2 width=4)
                        Filter: ((year = 2020) AND ((month)::bigint = 3))

(14 rows)
```

步骤 10 使用 explain, 对该 SQL 加以分析

```
tpch=# EXPLAIN
SELECT ad.province AS province, SUM(o.actual_price) AS GMV
FROM litemall_orders o,
     address_dimension ad,
     date_dimension dd
WHERE o.address_key = ad.address_key
     AND o.add_date = dd.date_key
     AND dd.year = 2020
     AND dd.month = 3
GROUP BY ad.province
ORDER BY SUM(o.actual_price) DESC;
```

(截图执行语句和结果)



```

tpch=# EXPLAIN
tpch=# SELECT ad.province AS province, SUM(o.actual_price) AS GMV
tpch=# FROM litemall_orders o,
tpch=#         address_dimension ad,
tpch=#         date_dimension dd
tpch=# WHERE o.address_key = ad.address_key
tpch=#        AND o.add_date = dd.date_key
tpch=#        AND dd.year = 2020
tpch=#        AND dd.month = 3
tpch=# GROUP BY ad.province
tpch=# ORDER BY SUM(o.actual_price) DESC;

QUERY PLAN

-----
Sort (cost=3579.58..3579.65 rows=31 width=47)
  Sort Key: (sum(o.actual_price)) DESC
  -> HashAggregate (cost=3578.58..3578.81 rows=31 width=47)
    Group By Key: ad.province
    -> Hash Join (cost=3340.21..3571.74 rows=1351 width=15)
      Hash Cond: (ad.address_key = o.address_key)
      -> Seq Scan on address_dimension ad (cost=0.00..188.02 rows=8002 width=14)
      -> Hash (cost=3323.32..3323.32 rows=1351 width=9)
        -> Hash Join (cost=17.56..3323.32 rows=1351 width=9)
          Hash Cond: (o.add_date = dd.date_key)
          -> Seq Scan on litemall_orders o (cost=0.00..3041.00 rows=100000 width=13)
          -> Hash (cost=17.53..17.53 rows=2 width=4)
            -> Index Scan using <16503-btree_date_dimension_year on date_dimension dd (cost=0.00..17.53 rows=2 width=4)
              Index Cond: (year = 2020)
              Filter: ((month)::bigint = 3)
(15 rows)

```

步骤 11 【附加题】有兴趣的同学可以尝试并截图记录于此。仅需要从 queries.sql 文件里选择一条或多条进行索引优化即可。

（截图执行语句和结果）

## 关卡四【附加题】：openGauss 的 DB4AI 特性应用

\*本关卡为附加题，有兴趣的同学可以尝试实验并记录于此。

### 1. 关卡验证

步骤 10 利用训练好的逻辑回归模型预测数据，并与 SVM 算法进行比较，将执行结果截图。

```

openGauss=# SELECT tax, bath, size, price, price < 100000 AS price_actual, PREDICT BY
house_binary_classifier (FEATURES tax, bath, size) AS price_svm_pred, PREDICT BY
house_logistic_classifier (FEATURES tax, bath, size) AS price_logistic_pred FROM houses;

```

（截图执行语句和结果）

```

NOTICE: Hyperparameter verbose (axes value DEFAULT) (FALSE)
NOTICE: GD shuffle cache size 4143
MODEL CREATED. PROCESSED 1
openGauss=# SELECT tax, bath, size, price, price < 100000 AS price_actual, PREDICT BY house_binary_classifier (FEATURES tax, bath, size) AS price_svm_pred, PREDICT BY house_logistic_classifier (FEATURES tax, bath, size) AS price_logistic_pred FROM houses;
 tax | bath | size | price | price_actual | price_svm_pred | price_logistic_pred
-----+-----+-----+-----+-----+-----+-----
 590 |    1 |   770 |  50000 | t             | t             | t
1050 |    2 |  1410 |  85000 | t             | t             | t
  20 |    1 |  1060 |  22500 | t             | t             | t
  870 |    2 |  1300 |   9000 | t             | t             | t
1320 |    2 |  1500 | 133000 | f             | t             | t
1350 |    1 |   820 |   90500 | t             | f             | f
2790 |   2.5 |  2130 | 260000 | f             | f             | f
  680 |    1 |  1170 | 142500 | f             | t             | t
1840 |    2 |  1500 | 160000 | f             | f             | f
3680 |    2 |  2790 | 240000 | f             | f             | f
1660 |    1 |  1030 |   87000 | t             | f             | f
1620 |    2 |  1250 | 118000 | f             | f             | f
3100 |    2 |  1700 | 140000 | f             | f             | f
2070 |    3 |  1550 | 140000 | f             | f             | f
  550 |   1.5 |  1450 |   65000 | t             | t             | t
(15 rows)

openGauss=#

```

# 清理工作：资源释放

## 1. 关卡验证

步骤 3 查看到列表中已没有资源时，表示弹性云服务器已删除。

（截图执行语句和结果）

