# 分析型数据库 PostgreSql

# 用户指南

天翼云科技有限公司

# 目 录

## 目录

修订历史

| 版本号 | 日期 | 修订人 | 修订内容 | 修订原因 |
| --- | --- | --- | --- | --- |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# 1. 产品概述

## 1.1 产品定义

分析型数据库 PostgreSql 基于开源项目 GreenPlum 构建的面向数据仓库应用的关系型数据库。支持 ANSI SQL 2003 语法，全面兼容 PostgreSql 数据库生态，是一种采用 MPP 全并行处理 Share-Nothing 架构实现对海量数据的即席查询分析、ETL 处理的数据仓库解决方案。

## 1.2 产品优势

- **易用易适配**

  兼容 ANSI SQL 2003，支持主流 SQL 语法；兼容 PostgreSql 数据库生态，学习使用成本低。

- **海量数据秒级分析**

  采用 MPP 全并行处理和 Share-Nothing 架构以及向量化计算及列存储智能索引，支持海量数据秒级分析。

- **高可用**

  Mirror 机制同步容错，故障自动切换。

- **高性价比**

  硬件选项自由，相对其他数据仓库平台资源投资低，易维护

- **实时 BI**

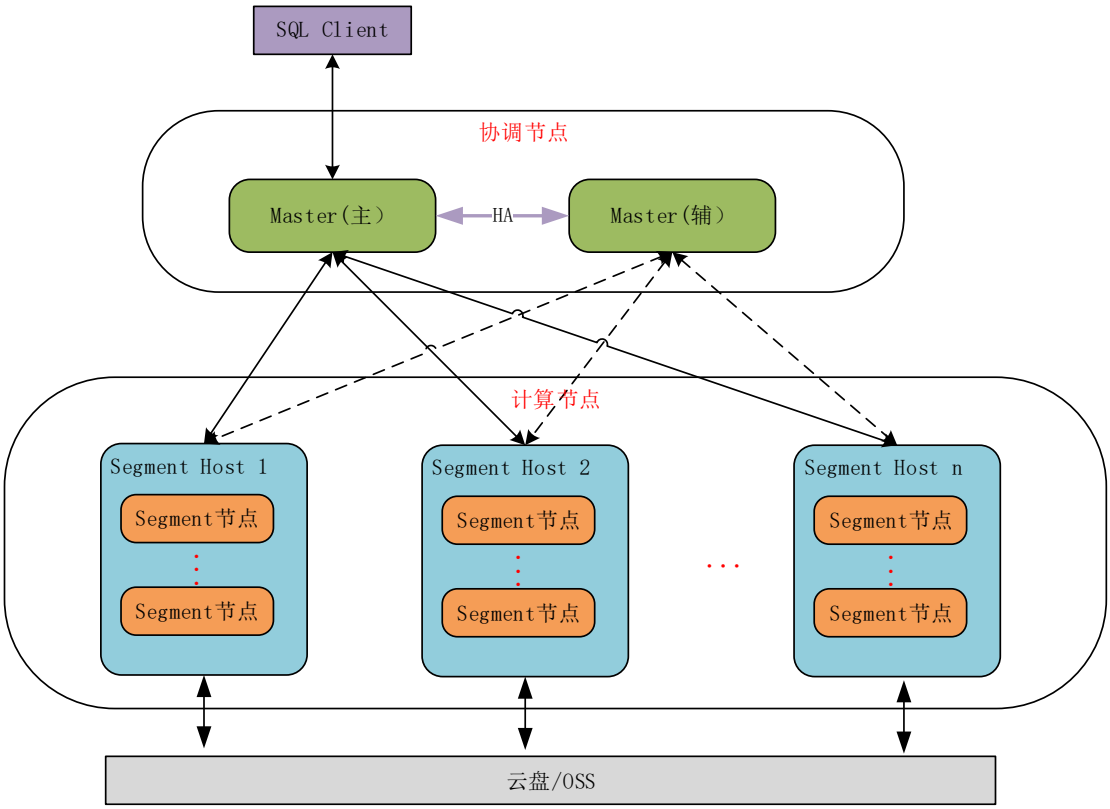
  准实时、实时数据加载，支持数据仓库实时更新，提供业务数据 BI 实时分析能力。

## 1.3 基本概念

| | |
|---|---|
| | 大规模并行处理，一种分布式 Shared Nothing 计算架构，多个无共享节点 |

| | |
|---|---|
| | 执行全并行计算，计算能力随节点增加线性提升。 |
| 计算节点 | 计算节点是用户购买计算资源单位，一个分析型数据库 PostgreSql 实例由多个计算节点组成。 |
| Segment 节点 | 每个计算节点运行多个 Segment 节点，分为 primary Segment 和 mirror segment，分布在不同的计算节点。某计算节点故障后，mirror 会切换成 primary。是计算和存储单元。 |
| Segment 节点数 | 实例所包含 primary Segment 节点的数量，实例存储空间额计算资源随节点数增加而线性增加。 |
| Segment 存储空间 | 每个 Segment 节点分配的数据存储空间 |
| 协调节点(Master) | 协调节点接收客户端的命令，解析并生成执行计划下发至对应 segment 节点进行执行，收集执行结果汇总返回给客户端。 |
| 数据分区 | 协调节点接收客户端的命令，解析并生成执行计划下发至对应 segment 节点进行执行，收集执行结果汇总返回给客户端。 |

## 1.4 产品架构




1、 Master 节点，分为主备角色，高可用。

2、 Segment 节点，包含 primary 和 mirror 角色，分布在不同的 Segment Host 计算节点上。

## 1.5 应用场景

分析型数据库 PostgreSql 引擎是为新一代数据仓库和大规模分析处理而建立的软件解决方案，其最大特点是不需要高端的硬件支持可以支撑大规模的高性能数据仓库和商业智能查询，主要适用于面向分析的 OLAP 应用领域，以下是典型的场景：

构建企业级数据仓库 EDW 服务，实时在线分析应用场景，同时也支持离线分析场景，常用作汇集库接受多源业务系统数据，并对外提供各种数据分析类查询业务，支持多维统计分析、智能模型构建、商业化 BI 报表分析、数据资源管理和对数据有高安全需求等多种应用场景。


# 2. 快速入门

## 2.1 产品规格

节点规格（segment：2C16G、4C32G

节点数量（segment）：4~128；

存储类型：普通 IO（SATA）

segment 节点存储空间：不超过 2000G。

## 2.2 计费方式

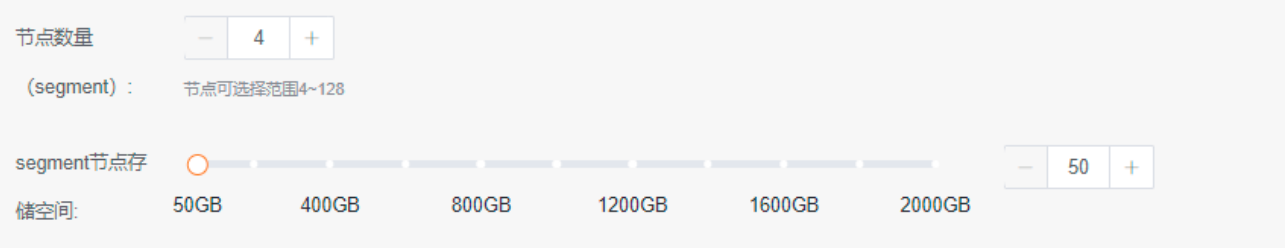
| 计费方式 | 说明 | 注意事项 |
| --- | --- | --- |
| 公测期 | 公测期不收取费用 | 遵守天翼云《公测产品服务协议》 |
| 预付费（包年包月） | 预付费方式，在新建实例时就需要支付费用。<br>对于包月的实例，购买 1 年即可享受价格 85 折优惠，且购买的年数越长，折扣更低。 | 无法变更为按量付费。（公测期结束后） |
| 按量付费 | 计费周期为 1 个小时。不足 1 小时，也按 1 小时收费。<br>账单生成后会自动从您的账户余额中扣除费用以结算账单。 | 实例可以随时释放，释放后不再收费。不可变更成包年包月。（暂不支持） |

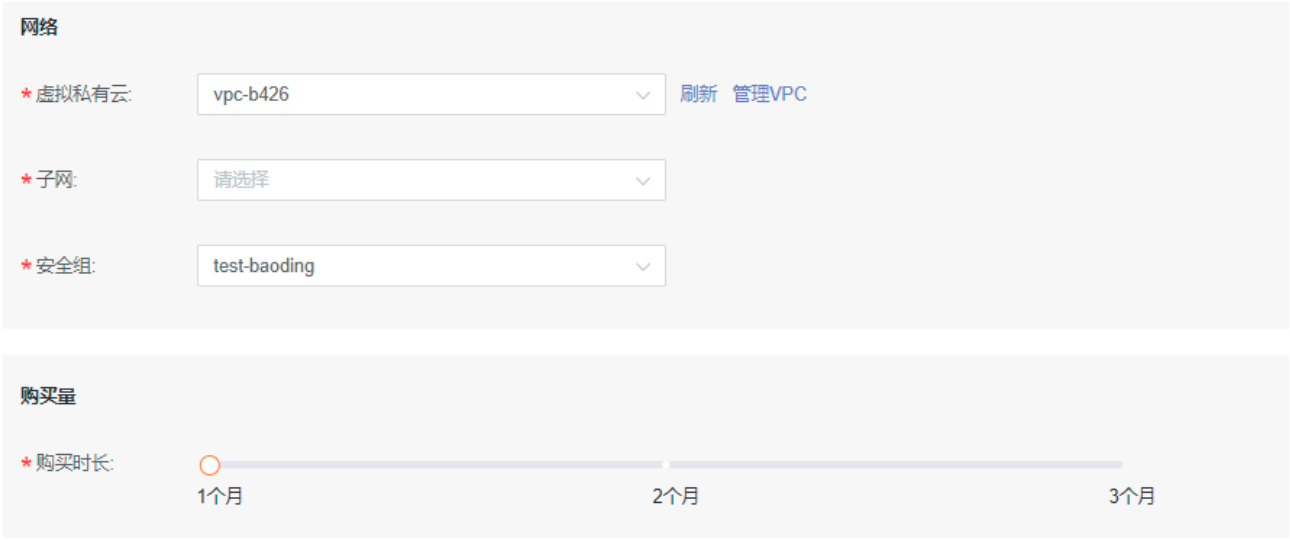
# 2.3 创建实例

1、登入天翼云门户

2、选择相应的资源池

3、进入服务列表，选择分析型数据库 PostgreSql，进入控制台。实例列表页点击“订购实例”进入开通页进行实例创建。




**基础配置：**

| 参数 | 描述 |
| --- | --- |
| 计费模式 | “包年/包月”或“按需计费”。<br>• 包年包月用户选购完服务配置后，可以根据需要设置购买时长，系统会一次性按照购买价格对账户余额进行扣费。<br>• 按需付费,用户选购完服务配置后，无需设置购买时长，系统会根据消费时长对账户余额进行扣费。（暂不支持） |
| 版本 | 6.16，参考社区 GreenPlum 6.16 |
| 存储类型 | 普通 IO，SATA 类型 |
| Master 节点 | 规格 4 核 32G，主备 2 个，高可用 |
| 节点规格（segment） | Segment 节点规格，2 核 16G，4 核 32G； |

| 参数 | 描述 |
| --- | --- |
| 节点数量（segment） | 支持的 segment 节点数量，4~128 |
| segment 节点存储空间 | 每个 Segment 节点的存储空间大小，50~2000G |

## 2.4 连接实例

1、创建数据库用户名和密码；

2、获取访问地址，进入 Cassandra 控制台，实例管理-实例详情，查看连接地址

3、通过客户端进行访问

### 2.4.1 创建数据库账号

分析型数据库 PostgreSql 提供两种角色数据库账号：

超级管理员：拥有所有数据库所有权限

普通账号：拥有已授权数据库的所有操作权限（注：对访问数据库已存在的其他用户创建的对象，需授权）

数据库账号创建步骤如下：

A、 分析型数据库 PostgreSql 控制台，实例列表->管理->账号管理

B、 点击“创建账号”，可创建超级管理员和普通用户账号


### 2.4.2 创建数据库

可通过 psql 连接后进行数据库创建。

psql postgres -U 用户名 -h 连接地址

create database 数据库名。

### 2.4.3 数据库授权

对于创建的普通用户账号，需要进行数据库授权。控制台账号管理，点击“数据库授权”为普通用户配置数据库


注：all 表示所有数据库，多个数据库以逗号分隔。对于需要访问授权的数据库其他用户创建的对象，需要通过超级管理员或创建对象的用户进行授权。

其他授权操作可通过 psql 命令行功能实现。

### 2.4.4 客户端连接

分析型数据库 PostgreSql 兼容 PostgreSql 数据库生态，可以直接使用支持 PostgreSQL 消息协议的工具 psql、libpq、JDBC、ODBC、psycopg2 等进行管理和开发。

### 2.4.5 psql 连接

psql 是分析型数据库 PostgreSql 较常用的命令行工具，提供了丰富的命令。

支持两种方式进行连接

1、 连接串方式

```
psql "host=xxx port=5432 dbname=postgres user=xxxx password=xxxx"
```

2、 指定参数的方式

```
psql  -h xxxx -p 5432 -d xxx -U xxx
```

参数说明：

-h：指定主机地址。

-p：指定端口号。

-d：指定数据库（默认的数据库是 postgres），

-U：指定连接的用户。

具体可通过 psql --help 查看更多选项

# 3. 操作指导

## 3.1 实例生命周期管理

进入分析型数据库 PostgreSql 控制台实例列表页，可进行实例的开通、续费、退订；

包周期实例到期不可访问，续费可恢复，退订后实例默认保留半个月后进行销毁。

按需计费实例退订立即释放，数据不保留。

### 3.1.1 变更实例

#### 3.1.1.1 存储空间扩容

随着业务的发展，数据库存储容量不能满足数据的增加时，可以进行存储空间扩容。

需要注意的是：存储空间只允许扩容，不能缩容。无需重启实例，在此期间，服务不中断，不影响您正常使用数据库。

1、 登录云数据库分析型数据库 PostgreSql 控制台

2、 在实例列表，选择目标实例，更多-存储空间扩容

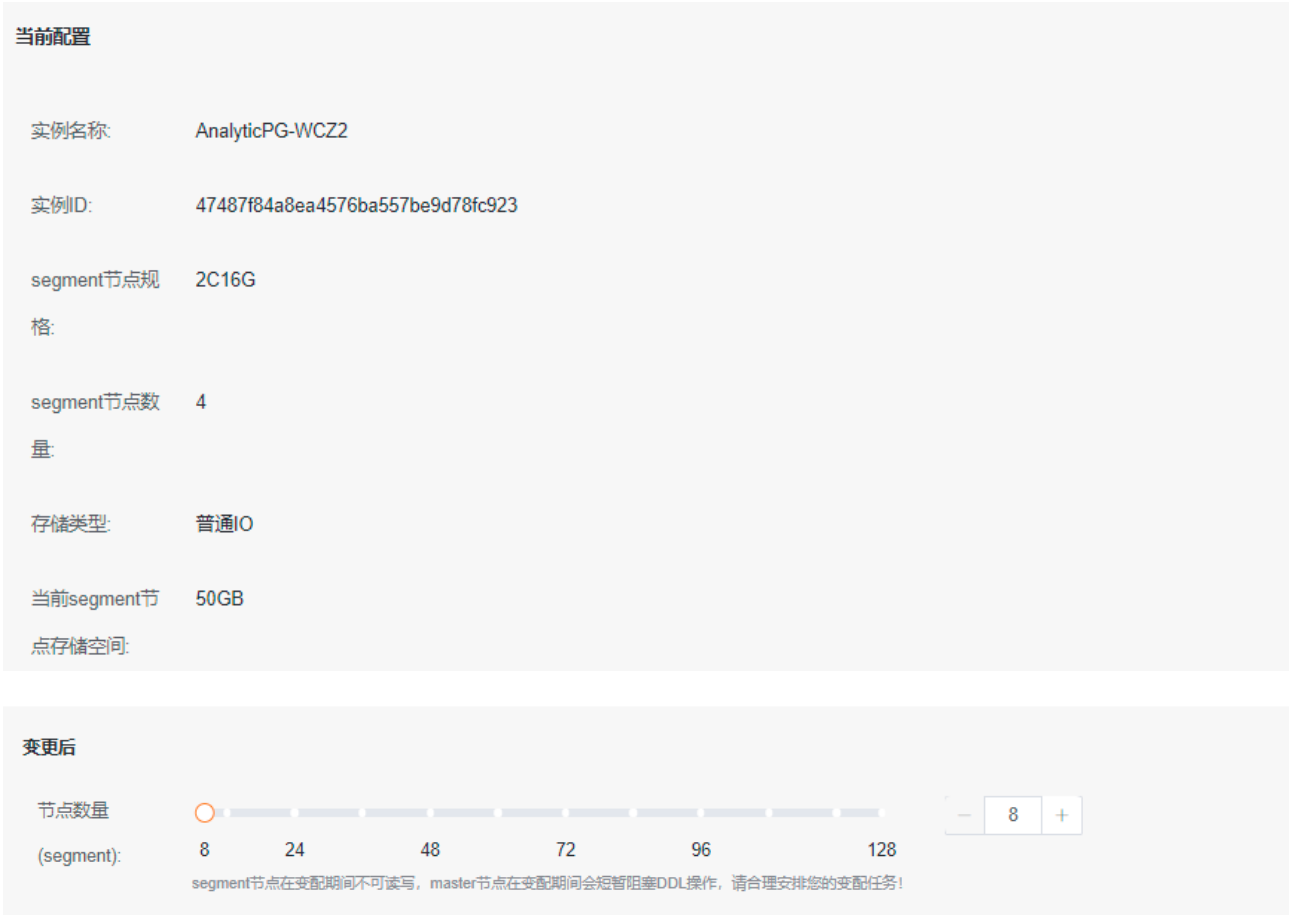

存储空间扩容是针对每个 Segment 节点，扩容的总容量=Segment 节点数*segment 节点扩容的容量。

### 3.1.1.2 加节点

随着业务发展，当前实例节点不能满足需求时，可以为实例添加节点，不支持节点缩容。

1、登录分析型数据库 PostgreSql 控制台

2、实例列表，选择目标实例，更多-增加节点




增加的 segment 节点个数受最大 128 的限制。

## 3.2 账号管理

账号管理模块提供数据库用户的创建和数据库授权。

### 3.2.1 新增数据库用户

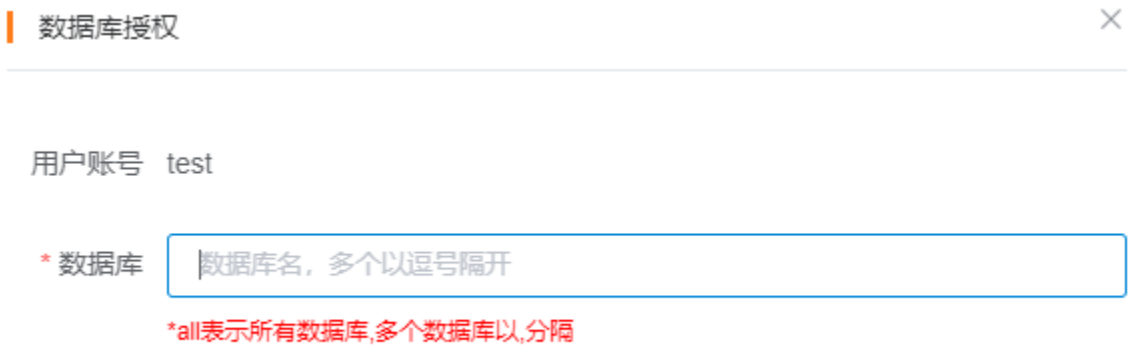
可以创建普通用户和超级用户，创建完普通用户后可通过 psql 命令行功能进行授权操作。


### 3.2.2 密码重置

提供用户密码的重置修改能力。


### 3.2.3 数据库授权

普通用户创建后，需要授予数据库权限，这样才具备连接访问能力。




all 表示具备所有数据库权限，多个以逗号隔开。对于数据库其他用户创建的对象，访问需授权。

## 3.3 数据库参数配置

数据库参数配置提供常用参数的调整，各参数功能可详见参数列表备注。

| 参数名称 | 默认参数值 | 运行参数值 | 是否需要重启 | 可修改值范围 | 参数描述 |
| --- | --- | --- | --- | --- | --- |
| authentication_timeout | 60s | 60s | 是 | [1,600] | 完成客户端认证的最大时间，这样可以... |
| deadlock_timeout | 1s | 1s | 是 | | 在检查以查看是否存在死锁情况之前等... |
| default_transaction_isolation | read committed | read committed | 否 | [read committed,r... | 控制每个新事务的默认隔离级别 |
| effective_cache_size | 512MB | 512MB | 否 | | 设置对于遗传查询优化器（计划器）的... |
| gp_initial_bad_row_limit | 1000 | 1000 | 否 | | 对于参数值 n，当用户使用COPY 命令... |
| gp_vmem_idle_resource_timeout | 18s | 18s | 否 | | 如果一个数据库会话空闲时间超过指定... |
| max_connections | 250 | 250 | 是 | [10,1500] | 与数据库服务器并发连接的最大数量,增... |

有些参数需要重启数据库，需慎重修改。

## 3.4 SQL 审计

SQL 审计提供相关 DDL、DML、DCL 以及 DQL 语句执行情况查询。


其中 SELECT 只提供超过 log_min_duration_statement（默认 2000ms）的语句查询。

## 3.5 节点管理

节点管理提供分析型数据库 PostgreSql Segment 节点状态、初始角色、当前角色，同步状态情况；并可支持实例重启、重平衡和节点恢复操作。


| dbid | 节点类型 | 角色 | 初始角色 | 同步状态 | 状态 | 端口 |
|---|---|---|---|---|---|---|
| 1 | Master | 主 | 主 | 未同步 | 存活 | 5432 |
| 4 | Segment-2 | 主 | 主 | 同步 | 存活 | 6000 |
| 8 | Segment-2 | 备 | 备 | 同步 | 存活 | 7000 |
| 5 | Segment-3 | 主 | 主 | 同步 | 存活 | 6001 |
| 9 | Segment-3 | 备 | 备 | 同步 | 存活 | 7001 |
| 3 | Segment-1 | 主 | 主 | 同步 | 存活 | 6001 |
| 7 | Segment-1 | 备 | 备 | 同步 | 存活 | 7001 |

# 4. 最佳实践

## 4.1 数据膨胀例行清理

分析型数据库 PostgreSql 使用 MVCC 事务并发模型的设计意味着被删除或者被更新的数据行仍

在磁盘上占据物理空间， 即便它们已经对新事务不可见。如果数据库进行了很多更新和删除，会有很多过期行存在并且它们所使用的空间必须使用 VACUUM 命令来回收。

不锁表的情况下，可以回收部分垃圾，具体方式如下：

连接每个数据库，以数据库的所有者身份登录，执行 VACUUM 命令。
执行频率： 如果有大批量实时更新的情况（即不断执行 INSERT VALUES、UPDATE、DELETE 等操作），最好每天执行一次，或每周至少一次。如果更新是每天一次批量进行的，建议每周执行一次，或不要超过一个月执行一次。
注：该操作会导致 CPU、I/O 使用率增加，可能影响查询的性能。

可以使用如下的 Linux Shell 脚本文件，作为 crontab 定期任务来执行。

```
#!/bin/bash
export PGHOST=xxxx
export PGPORT=5432
export PGUSER=xxxx
export PGPASSWORD=xxx
#do not echo command, just get a list of db
dblist=`psql -d postgres -c "copy (select datname from pg_stat_database) to
stdout"`
for db in $dblist ; do
    #skip the system databases
    if [[ $db == template0 ]] ||  [[ $db == template1 ]] || [[ $db ==
postgres ]] || [[ $db == gpdb ]] ; then
        continue
    fi
    echo processing $db
    #vacuum all tables (catalog tables/user tables)
    psql -d $db -e -a -c "VACUUM;"
done
```

业务暂停，可以回收所有垃圾。具体方式如下：

连接每个数据库，以数据库的所有者身份登录（需要对所有操作对象有所有者权限）。
A、执行 REINDEX SYSTEM <database name>。
B、对每张数据表，执行 VACUUM FULL <table name>，REINDEX TABLE <table name>。

C、对于系统表（包括 pg_class, pg_attribute, pg_index 等），当有频繁建删表，建删索引等操作时，也建议执行 VACUUM FULL <table name>进行定期维护。注意：该操作需要业务停止访问数据库。

频率：至少每周执行一次。如果每天会更新几乎所有数据，需要每天做一次。
对系统影响：会对正在进行 VACUUM FULL 或 REINDEX 的表进行锁定，无法读写。会导致 CPU、I/O 使用率增加。

可以使用如下的 Linux Shell 脚本文件，作为 crontab 定期任务来执行。

```
#!/bin/bash
export PGHOST=xxxx
export PGPORT=5432
export PGUSER=myuser
export PGPASSWORD=mypass
#do not echo command, just get a list of db
dblist=`psql -d postgres -c "copy (select datname from pg_stat_database) to
stdout"`
for db in $dblist ; do
    #skip system databases
    if [[ $db == template0 ]] ||  [[ $db == template1 ]] || [[ $db ==
postgres ]] || [[ $db == gpdb ]] ; then
        continue
    fi
    echo processing db "$db"
    #do a normal vacuum
    psql -d $db -e -a -c "VACUUM;"
    #reindex system tables firstly
    psql -d $db -e -a -c "REINDEX SYSTEM $db;"
    #use a temp file to store the table list, which could be vary large
    cp /dev/null tables.txt
    #query out only the normal user tables, excluding partitions of parent
tables
    psql -d $db -c "copy (select '\"'||tables.schemaname||'\".' ||
'\"'||tables.tablename||'\"' from (select nspname as schemaname, relname as
tablename from pg_catalog.pg_class, pg_catalog.pg_namespace,
pg_catalog.pg_roles where pg_class.relnamespace = pg_namespace.oid and
pg_namespace.nspowner = pg_roles.oid and pg_class.relkind='r' and
(pg_namespace.nspname = 'public' or pg_roles.rolsuper = 'false' ) ) as
tables(schemaname, tablename) left join pg_catalog.pg_partitions on
pg_partitions.partitionschemaname=tables.schemaname and
pg_partitions.partitiontablename=tables.tablename where
pg_partitions.partitiontablename is null) to stdout;" > tables.txt
    while read line; do
        #some table name may contain the $ sign, so escape it
        line=`echo $line |sed 's/\\\$/\\\\\\\$/g'`
        echo processing table "$line"
        #vacuum full this table, which will lock the table
        psql -d $db -e -a -c "VACUUM FULL $line;"
        #reindex the table to reclaim index space
        psql -d $db -e -a -c "REINDEX TABLE $line;"
    done <tables.txt
done
```

## 4.2 数据倾斜诊断处理

在 MPP 无共享环境中，查询的总体响应时间由所有节点的完成时间来度量。 系统只能与最慢的节点一样快。 如果数据偏斜，具有更多数据的节点将花费更多时间来完成，因此每个节点必须具有大致相等的行数并执行大致相同的处理量。 如果一个节点具有比其他节点更多的处理数据，则可能导致性能不

佳和内存不足，尤其是大表做连接查询时，最佳分布至关重要。

- 针对单个表或库，可以通过查看空间占用的方式来确定是否有倾斜。
  - 判断某个库是否出现数据的倾斜：

```
select pg_size_pretty(pg_database_size('dbname')) from gp_dist_random('gp_id');
```

通过上述语句，可以看出名为 dbname 的库在每个节点上占用的空间，当某个或个别几个节点的空间明显大于其他节点时，说明该库上有数据倾斜存在。

  - 判断某张表是否出现数据的倾斜：

```
select pg_size_pretty(pg_relation_size('tblname')) from gp_dist_random('gp_id');
```

通过上述语句，可以看出名为 tblname 的表在各个节点上占用的空间，当某个或个别几个表的空间明显大于其他表时，说明该表上有数据倾斜存在，需要修改该表的分布键以调整数据分布。

- 利用系统视图判断数据倾斜。
  - 利用下面语句判断是否在存储上有倾斜，原理与查看空间占用类似：

```
select * from gp_toolkit.gp_skew_coefficients;
```

该视图会基于表的行的数据量来检查，如果表数据量越大，检查时间就会越长。

  - 通过视图 gp_toolkit.gp_skew_idle_fractions 计算在表扫描过程中系统闲置资源的百分比，来判断数据倾斜的情况：

```
select * from gp_toolkit.gp_skew_idle_fractions;
```

## 4.3 执行中的 SQL 诊断

视图 pg_stat_activity 每行显示一个服务器进程同时详细描述与之关联的用户会话和查询，以下是视图具体详情：

| 列 | 类型 | 参考 | 描述 |
| --- | --- | --- | --- |
| datid | oid | pg_database.oid | 数据库 OID |
| datname | name | | 数据库名称 |
| pid | integer | | 服务进程的进程 ID |
| sess_id | integer | | 会话 ID |

| 列 | 类型 | 参考 | 描述 |
| --- | --- | --- | --- |
| usesysid | oid | pg_authid.oid | 登录此后端的用户的 OID |
| usename | name | | 登录到此后端的用户的名称 |
| application_name | text | | 连接到此后端的应用程序的名称 |
| client_addr | inet | | 连接到此后端的客户端的 IP 地址。 如果此字段为空，则表示客户端通过服务器计算机上的 Unix 套接字连接，或者这是内部进程（如 autovacuum）。 |
| client_hostname | text | | 客户端的主机名，由 client_addr 的反向 DNS 查找报告。 对于 IP 连接，此字段仅为非 null，并且仅在启用 log_hostname 时才为空。 |
| client_port | integer | | 客户端用于与此后端通信的 TCP 端口号，如果使用 Unix 套接字，则为-1 |
| backend_start | timestamptz | | 后端进程启动时间 |
| xact_start | timestamptz | | 事务开始时间 |
| query_start | timestamptz | | 查询开始执行时间 |
| state_change | timestampz | | 状态最后一次改变的时间 |
| waiting | boolean | | 如果等待一个锁为 True，否则为 false |
| state | text | | 此后端的当前整体状态。可能的值是：<br><br>active：后端正在执行查询。 |

| 列 | 类型 | 参考 | 描述 |
| --- | --- | --- | --- |
| | | | idle：后端正在等待新的客户端命令。<br><br>idle in transaction：后端处于事务中，但当前未执行查询。<br><br>idle in transaction (aborted)：此状态类似于事务中的空闲，除了事务中的一个语句导致错误。<br><br>fastpath function call：后端正在执行快速路径功能。<br><br>disabled：如果在此后端禁用 track_activities，则报告此状态。 |
| query | text | | 此后端的最新查询的文本。 如果状态为活跃，则此字段显示当前正在执行的查询。 在所有其他状态中，它显示最后执行的查询。 |
| waiting_reason | text | | 服务器进程正在等待的原因。 值可以是：<br>lock， replication 或 resgroup |
| rsgid | oid | pg_resgroup.oid | 资源组 OID |
| rsgname | text | pg_resgroup.rsgname | 资源组名称 |
| rsgqueueduration | interval | | 对于排队查询，查询排队的总时间 |

### 查看连接信息

通过下述 SQL 就能确认当前的连接用户和对应的连接机器。

```
SELECT datname,usename,client_addr,client_port FROM pg_stat_activity ;
```

### 查看 SQL 运行信息

获取当前用户执行 SQL 信息：

```
SELECT datname,usename,query FROM pg_stat_activity ;
```

只看当前正在运行的 SQL 信息：

```
SELECT datname,usename,query FROM pg_stat_activity WHERE state !=
'idle' ;
```

**查看耗时较长的查询**

查看当前运行中的耗时较长的 SQL 语句

```
select current_timestamp - query_start as runtime, datname, usename,query
  from pg_stat_activity where state != 'idle'order by 1 desc;
```

**异常 SQL 诊断及修复**

如果一个 SQL 运行很长时间没有结果，需要检查该 SQL 还在运行中还是已经被 block 了：

```
SELECT datname,usename,query FROM pg_stat_activity WHERE waiting;
```

需要注意的是这个输出只能获取当前因为 lock 而被 block 的 SQL，因为其他原因被 block 的 SQL 这里获取不到。绝大多数情况下 SQL 都是因为 lock 而被 block，但也会有一些其他情况例如等待 i/o、定时器等。如果上述 SQL 有结果输出说明有 SQL 被 lock 阻塞，进一步明确相互 block 的 SQL 信息：

```
SELECT
      w.current_query as waiting_query,
      w.procpid as w_pid,
      w.usename as w_user,
      l.current_query as locking_query,
      l.procpid as l_pid,
      l.usename as l_user,
      t.schemaname || '.' || t.relname as tablename
   from pg_stat_activity w
   join pg_locks l1 on w.procpid = l1.pid and not l1.granted
   join pg_locks l2 on l1.relation = l2.relation and l2.granted
   join pg_stat_activity l on  l2.pid = l.procpid
   join pg_stat_user_tables t on l1.relation = t.relid
   where w.waiting;
```

通过这个 SQL 的输出信息就能确认相互 block 的 SQL 和对应的执行 pid。在明确了 SQL 的阻塞信息后，可以通过 cancel 或者 kill query 的方式进行恢复。通过 cancel 取消一个正在运行的 query：

```
SELECT pg_cancel_backend(pid)
```

需要在一个运行 query 的 session 中执行，如果 session 本身就是 idle 的，执行不起作用。另外取消这个 query 需要花费一定的时间来做清理和事务的回滚。使用 pg_terminate_backend 来清理 idle session，也可以用来终止 query：

```
SELECT pg_terminate_backend(pid);
```

该用户的连接会被断开。尽量避免在正在运行 query 的进程 pid 上执行。需要注意的是文中提到操作需要用户有 superuser 的权限。

## 4.4 使用 ANALYZE

分析型数据库 PostgreSQL 优化器在进行查询优化时，会根据统计信息进行查询代价估算和优化。如果参与查询的表没有收集过统计信息或统计信息过旧，系统将按照默认值或老旧的统计信息进行优化，往往无法生成最优执行计划。所以，建议在大批量数据加载完成，或者有较多数据（超过 20%）更新后，进行统计信息收集。

采用 ANALYZE 命令收集统计信息时，可以对所有表收集、对某个表的所有列收集或对表的指定列收集。对于大部分用户，建议采用对所有表收集或对表的所有列收集的方式。如果想对统计信息收集环节做精细化控制，可以采用对表的指定列收集方式，针对关联（JOIN）的条件列、过滤条件列、有索引的列进行统计信息收集。

**示例**

收集所有表的统计信息示例（推荐数据大批量入库后使用）：

```
ANALYZE;
```

收集表 t 的所有列的统计信息示例（推荐某个表插入/更新/删除较多数据后使用）：

```
ANALYZE t;
```

收集表 t 的 a 列的统计信息示例：

```
ANALYZE t(a);
```

## 4.5 优化器选择

**分析型数据库 PostgreSql 提供两种优化器，**默认优化器为 ORCA 优化器。两个优化器在不同的场景下，各有优势。

**Legacy 优化器：**SQL 优化耗时较短，适合高并发的简单查询场景（3 表以内关联），或者高并发的数据写入或更新场景（INSERT/UPDATE/DELTE）。

**ORCA 优化器：**面向复杂 SQL 语句的优化器，会遍历更多执行路径，制定最优执行计划，但 SQL 优化过程相对耗时稍长。建议对复杂查询（3 表以上关联为主的场景）为主的 ETL 场景和报表场景采用此外，ORCA 优化器具有相关子查询的解关联优化及动态分区裁剪优化等能力，含有相关子查询的语句及含有带参数化过滤条件的分区表的语句建议使用 ORCA 优化器。

查看当前的优化器的方式：

```
show optimizer;
-- 值为 on：表示当前优化器为 ORCA 优化器
-- 值为 off：表示当前优化器为 Legacy 优化器
```

Session 会话级设置方式：

```
-- 使用 Legacy 优化器
set optimizer = off;
-- 使用 ORCA 优化器
set optimizer = on;
```

## 4.6 锁检查

*pg_locks* 系统目录视图允许用户查看有关未解除的锁的信息，提供了数据库系统中所有锁的一个全局视图。用户可以把 pid 列 连接到 pg_stat_activity.procpid 以查看更多会话持有或者等待持有锁的信息。

例如：

```
SELECT locktype, database, c.relname, l.relation,
l.transactionid, l.transaction, l.pid, l.mode, l.granted,
a.current_query
        FROM pg_locks l, pg_class c, pg_stat_activity a
        WHERE l.relation=c.oid AND l.pid=a.procpid
        ORDER BY c.relname;
```

如果用户把资源队列用于负载管理，在一个队列中等候的查询也会显示在 *pg_locks* 中。要查看一个资源队列中有多少查询正在等待运行，可使用 *gp_resqueue_status* 系统目录视图。例如：

```
SELECT * FROM gp_toolkit.gp_resqueue_status;
```

# 5. 常见问题

## 5.1 实例创建失败或长时间处于创建中

实例创建后，一般在 20 分钟内完成。如果新建实例长时间处于创建中，一般是由于后端资源分配时间过长或资源不足，这情况请提工单处理或者更好其他资源池尝试购买。

## 5.2 实例开通后是否支持 VPC 切换

实例创建完成后暂不支持更换 VPC，开通时需结合应用服务选择相应虚拟子网。

## 5.3 可开通资源池

具体以开通页面可选资源池为准。

## 5.4 计算节点与 Segment 节点

计算节点是资源分配的单位，每个计算节点相当于一台主机，独享。

Segment 节点是部署在计算节点上。一个节点计算节点包含 2 个 Primary segment 节点和 2 个 Mirror segment 节点。

## 5.5 Master 节点

Master 节点主备模式，主 master 处于工作模式；主 Master 节点不可用时，备 Master 节点接管服务。

## 5.6 终止异常 SQL 执行

若需要终止特定的 SQL 或会话来恢复系统状态，先通过 pg_stat_activity 系统视图查询当前的查询. 终止非当前连接的 SELECT：

```
SELECT pg_cancel_backend(pg_stat_activity.procpid)
FROM pg_stat_activity
WHERE procpid <> pg_backend_pid() and current_query like 'SELECT%';--注意 SELECT 的大小写
```

## 5.7 查看表和数据库数据量

假设表的模式为<schemaname>，表名为<tablename>。

- 执行以下命令，查询一张表的总大小（单位为 MB，包含表的索引和数据）：

```
select
pg_size_pretty(pg_total_relation_size('<schemaname>.<tablename>'));
```

- 执行以下命令，查询表的数据大小（单位 MB，不包括索引）：

```
select pg_size_pretty(pg_relation_size('<schemaname>.<tablename>'));
```

- 执行以下命令，查询分区表所有分区的总大小（单位 MB，包含表的索引和数据）：

```
select schemaname,tablename,round(sum(pg_total_relation_size(schemaname
|| '.' || partitiontablename))/1024/1024) "MB" from pg_partitions where
schemaname='<schemaname>' and tablename='<tablename>' group by 1,2;
```

- 执行以下命令，查询一个 Schema 下面的所有表的总大小（单位 MB，包括索引和数据）：

```
select
schemaname ,round(sum(pg_total_relation_size(schemaname||'.'||tablename))
```

```
/1024/1024) "Size_MB" from pg_tables where schemaname='<schemaname>'
group by 1;
```

- 执行以下命令，查询每个数据库的大小（单位 MB）：

```
select datname,pg_size_pretty(pg_database_size(datname)) from
pg_database;
```