GreenPlum使用操作指南

V0.0.2

**最近修订记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 修订日期 | 版本 | 创建人 | 修订记录 |
| 20170718 | v0.0.1 | 彭仲颐 | 创建文档 |
| 20170727 | V0.0.2 | 彭仲颐 | 增加进阶内容 |
|  |  |  |  |
|  |  |  |  |

目录

[第一章-GP基础操作 4](#_Toc488153053)

[第二章-GP结构介绍 4](#_Toc488153054)

[1.表切片 4](#_Toc488153055)

[2.分布式join 4](#_Toc488153056)

[3.分布式事务 4](#_Toc488153057)

[第三章-GP上线部署 4](#_Toc488153058)

[3.1硬件选型标准: 4](#_Toc488153059)

[3.2 OS配置: 4](#_Toc488153060)

[3.3 测试: 5](#_Toc488153061)

[3.4 数据库参数优化: 5](#_Toc488153062)

[第四章-GP数据库管理 5](#_Toc488153063)

[4.1 用户与权限管理 5](#_Toc488153064)

[4.2 资源队列 6](#_Toc488153065)

[4.3 锁的管理 6](#_Toc488153066)

[4.4 文件系统 7](#_Toc488153067)

[4.5 表空间 7](#_Toc488153068)

[第五章-GP数据库监控 7](#_Toc488153069)

[5.1.系统监控: 7](#_Toc488153070)

[5.2.数据库监控: 7](#_Toc488153071)

[5.3.详细指标监控: 7](#_Toc488153072)

[第六章-GP数据库运维 8](#_Toc488153073)

[6.1.启动/停止脚本gpstart/gpstop 8](#_Toc488153074)

[6.2.参数修改脚本 gpconfig 8](#_Toc488153075)

[第七章-GP数据库备份与恢复 8](#_Toc488153076)

[第八章-GP数据库扩容 8](#_Toc488153077)

[8.1.group mirror和spread mirror 8](#_Toc488153078)

## 第一章-GP基础操作

后续加上

## 第二章-GP结构介绍

GreenPlum是一种Share Nothing架构的 MPP数据库

Share Nothing : 不同节点不共享硬件设备,仅通过网络进行通信与合并

MPP数据库 : 分布式数据库的代表,硬件性能与机器数量成正比

### 1.表切片

GP中,所有的表都是分布式的,切片类型可以是hash或者随机切片

随机切片不适用于join/group by出现的表

hash切片尽量基于唯一且常用于join的列,这样数据会平摊至节点

### 2.分布式join

如果是join两个hash join的列,那么本地节点join后再合并结果即可

其他情况下,需要将数据distribute分布到各节点上,与Vertica类似

### 3.分布式事务

主要解决两阶段提交(2PC)问题,主要由master来协调,实现一致回滚或提交

## 第三章-GP上线部署

GP的性能与Vertica一样,依据最差的机器而定,遵循木桶原则

### 3.1硬件选型标准:

1.依据:高性能原则(CPU,内存,磁盘,网络) 高可靠原则(单点故障问题) 可拓展性原则(预计何时升级)

2.CPU选型:并发主要由cpu核数决定

3.内存选型:用于存放join,排序等,要满足大部分sql的内存需求

4.磁盘选型:磁盘与内存的交互,磁盘内部的交互,都是较为关键的部分.SAS盘或SSD盘为佳,SATA盘转速较低,会有一定的性能瓶颈.做raid哪个级别也是需要考虑的问题

5.网络:千兆足以,注意冗余

### 3.2 OS配置:

基于官网文档

<http://gpdb.docs.pivotal.io/500Beta/install_guide/prep_os_install_gpdb.html#topic3>

/etc/sysctl.conf 内存与ipv4配置

/etc/security/limits.conf 文件句柄设置

推荐基于xfs文件系统(centos7可用,6似乎不推荐使用)

echo deadline > /sys/block/sda/queue/scheduler 针对大量小文件的优化

/sbin/blockdev --setra 16384 /dev/sda 文件预读大小

/etc/hosts 设置合适的别名来互联,建议gp001,002...

echo never > /sys/kernel/mm/redhat\_transparent\_hugepage/enabled 关闭透明大页功能

ntp时钟同步 使用crontab+ntpdate命令

### 3.3 测试:

1.使用gpcheck命令来测试 /app/database/greenplum-db/bin/gpcheck -h v001

可以检查出哪些配置异常的节点

2.使用gpcheckperf命令来测试

### 3.4 数据库参数优化:

1.shared buffer 在不引起swap的情况下,越大越好,一般15%左右

2.effective cache size 参数越大越倾向索引读,否则全表扫

3.work mem 排序时使用,越大则分片的文件越少,提高IO效率

4.temp\_buffer 临时表缓冲区

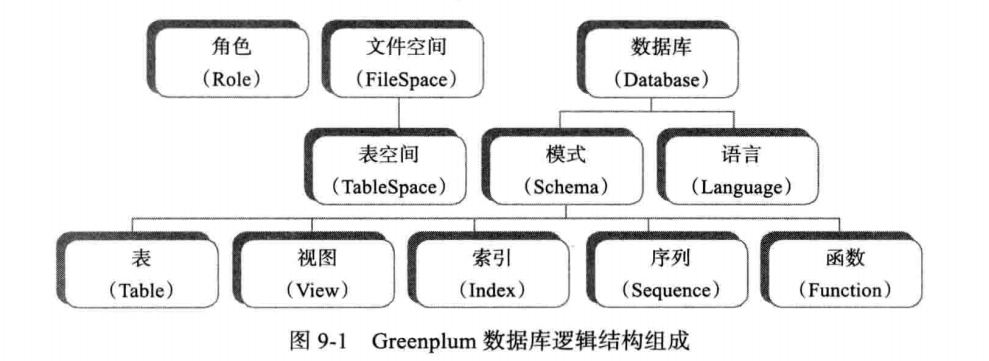
5.client\_min\_message 发送客户端的消息级别,panic即可

6.debug\_print\_plan/parse 打开执行计划

7.gp\_hashjoin\_tuples\_per\_bucket 此bucket越大,对大表join的支持就越好.

## 第四章-GP数据库管理

### 4.1 用户与权限管理



1.database下有多个schema,也叫作namespace

2.一个table对应多个tablespace

3.创建用户,登录权限,创建用户和数据库权限,连接数为5

create role testreole1 createdb createrole connection limit 5 login;

4.授权

grant select on test\_table\_1 to testrole1;

revoke select on test\_table1 from testrole1;

5.获取权限控制信息

select \* from pg\_class a,pg\_attribute b ,pg\_namespace c

where a.oid=b.attrelid and a.relnamespace=c.oid and c.nspname='pg\_catalog' and b.attname like '%acl%'

### 4.2 资源队列

1.创建资源限制队列

这个队列平均分配内存

create resource queue adhoc with (active\_statements=30,memory\_limit='10240MB',priority=HIGH);

应该分为快速队列,查询队列和缓慢队列

快速队列:时间短,内存中等,排队少,数量多+ active\_statements=20,memory\_limit='30GB',priority=HIGH

查询队列:时间中等,内存中等,排队中等,数量多 active\_statements=10,memory\_limit='30GB',priority=MEDIUM

缓慢队列:时间长,内存多,排队多,数量少

active\_statements=5,memory\_limit='60GB',priority=LOW

--查询资源队列配置

select \* from pg\_resqueue\_attributes;

--现有资源队列使用

select \* from pg\_resqueue\_status;

### 4.3 锁的管理

1.access share 访问表select申请锁

2.row share 行锁

3.row exclusive 在update,delete,insert会使用

4.share update exclusive 在analysis,create index concurrently使用

5.share 普通create index使用

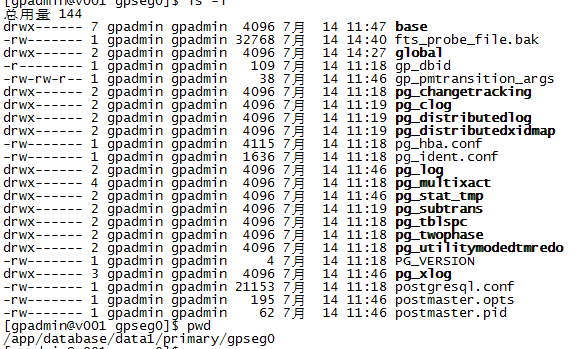
6.access exclusive 保证该事务是唯一访问模式 drop table/alter table/reindex

数字越大影响越大

查询锁情况

select \* from pg\_locks

### 4.4 文件系统



### 4.5 表空间

表空间是因为当一个文件夹下有大量文件时,Linux会有一定的I/O瓶颈

因此通过多个表空间来分散文件数

要在每个节点上创建文件夹

然后执行gpfilespace命令来生成

## 第五章-GP数据库监控

### 5.1.系统监控:

oralce em监控即可

### 5.2.数据库监控:

自带的performce monitor亦可

### 5.3.详细指标监控:

1.检测segment切换

select \* from gp\_segment\_configuration where mode !='s' or status='d'

2.彻底删除数据vacuum

<http://gpdb.docs.pivotal.io/500Beta/ref_guide/sql_commands/VACUUM.html#topic1__section6>

如果数据库的事务号超过2亿,则会触发auto-vacuum,导致数据库响应缓慢,所以要定时进行vacuum

由于vacuum full的代价很高(大量资源),所以必要时可以create table as 再drop

3.数据均匀分布

可以使用gp\_segment\_id,这个字段每个表都有,是个隐藏字段

可以使用pg\_relation=-size来通过容量确认分布均匀

4.sql运行状态

select \* from pg\_stat\_activity

5.清理过期临时数据

## 第六章-GP数据库运维

### 6.1.启动/停止脚本gpstart/gpstop

-M fast 终止事务并回滚

-M imeedaite 不回滚直接关闭,慎用

-u 不重启,只是重新加载防火墙策略和配置文件

### 6.2.参数修改脚本 gpconfig

## 第七章-GP数据库备份与恢复

1.使用gprecoverseg进行恢复失效的segment

注意,会使得整个数据库暂时hang,可通过gpstate -m来查看进度

-r参数可以在mirror启动,primary恢复后导致的节点数过多情况下,重新平衡

需要注意节点状态已同步(synchronized)

2.失败的primary节点会在恢复后成为mirror节点,两者身份互换

3.导出方式:gpdump是传统串行pgdump的2-5倍

4.primary master和standby master互为master节点的备份

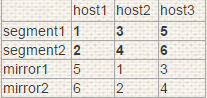
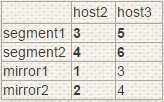
## 第八章-GP数据库扩容

### 8.1.group mirror和spread mirror

group mirror(gm) 是指1的备份在2,2的备份在3,以此类推的方式

这样情况下,如果有一台宕机,必然有一台会承受双倍的压力(host2)

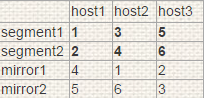
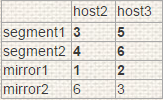
新增节点则需要至少是2的倍数,保证新主机1的备份在新主机2上

 🡺

而spread mirror则是将segment分散,这样的话压力还是能分散到每个节点

这样的话,如果新增节点,则需要segment数+1的机器(保证能够分散)

这和vertica的理念类似了,只不过vertica将数据已经切分为了节点数份数

 🡺

在数据库初始化时,使用-S参数来设置spread mirror

## 第九章-GP进阶内容

### 9.1.分布列法则

1.情况1-id分布列.

Select \* from t1,t2 where t1.id=t2.id

当JOIN的列都是分布列时，不需要重分布或广播小表，可以在segment内完成JOIN。

在单独的segment段内.可以直接连接查询

2.情况2-大小表join

Select \* from t1,t2 where t1.prov\_id=t2.prov\_id

两个表在JOIN时，如果JOIN列不是表的分布列，那么其中一个更小的表会发生数据重分布，或者broadcast，以继续完成JOIN的动作。

如果表T1更小,则会将表T1 broadcast(广播)到每个segment上.以供查询使用

这种情况下,需要网络来传输

结论:

1. 分布列尽量选择需要经常join的列
2. 更小的表,或者是维度表,应该建成broadcast状态,每个segment都分布一份
3. 高频率的查询,可将查询条件作为分布列,这样可以减少查询次数与时间
4. 随机分布(hash)不是一个优秀的分布方式

### 9.2.分区法则

原则:

1. 缩小查询范围
2. 选择与查询条件相关的字段
3. 多个查询条件时,使用子分区来缩小查询范围

### 9.3.拓展阅读与参考资料

Mysql与PG互迁工具dbsync

<https://github.com/aliyun/rds_dbsync>

使用 OSS 外部表同步数据

<https://help.aliyun.com/document_detail/35457.html>

update最佳实践

<https://yq.aliyun.com/articles/86604>

### 9.2.group mirror和spread mirror