

Greenplum 管理实战

目录

Greenplum 管理实战.....	1
目录.....	1
1 catalog 管理.....	2
1.1 catalog 基本原理.....	2
1.1.1 仅在 master 保存的元数据.....	3
1.1.2 表-元数据表(部分).....	4
1.1.2.1 查询实例.....	4
1.2 gpcheckcat 命令的使用.....	6
1.2.1 catalog 检查不一致问题.....	6
1.2.2 gpcheckcat 在线测试.....	6
1.2.3 gpcheckcat 离线测试.....	7
1.2.4 gpcheckcat 提示自动修复.....	7
1.2.5 最佳实践和常见问题.....	8
2 master 管理.....	8
2.1 master 基本原理和操作.....	8
2.1.1 master 管理工具.....	8
2.1.2 添加 standby.....	9
2.1.3 master-standby 数据同步原理.....	9
2.1.4 standby 同步状态.....	10
2.1.5 standby 特定配置.....	10
2.1.6 日志查看和收集.....	11
2.2 master 故障管理.....	11
2.2.1 master 节点异常退出:尝试恢复 master.....	11
2.2.2 master 节点异常退出:master 无法恢复.....	12
2.2.3 gpactivatestandby 内部逻辑.....	12
2.2.4 提升 standby 操作过程.....	12
2.2.5 重建 standby 操作过程.....	13
2.3 standby 故障管理.....	13
2.3.1 standby 异常.....	13
2.3.2 standby 异常处理.....	14
2.4 集群启动过程.....	15
2.4.1 gpstart 内部逻辑.....	15
2.4.2 集群无法启动.....	15
3 segment 管理.....	17
3.1 基本原理和操作.....	17
3.1.1 segment 管理工具.....	17

3.1.2 配置 mirror.....	17
3.1.3 primary-mirror 数据同步.....	18
3.1.4 mirror 异常退出.....	18
3.1.5 primary 异常退出.....	19
3.2 segment 故障管理.....	19
3.2.1 segment 异常退出.....	19
3.2.2 segment 异常处理:主机错误.....	20
3.2.3 segment 异常处理:进程错误.....	20
3.2.4 segment 异常处理:数据错误.....	21
3.2.5 primary segment 异常处理:再平衡.....	21
3.2.6 增量和全量恢复.....	23
3.2.7 gprecoverseg 常见错误.....	24
3.3 集群扩容.....	24
3.3.1 集群扩容准备.....	24
3.3.2 扩容实例.....	25
3.3.2.1 扩容之前的状态.....	25
3.3.2.2 交互式生成输入文件.....	25
3.3.2.3 添加节点.....	26
3.3.2.4 数据重分布.....	26
3.3.2.5 清除扩容的状态.....	26
3.3.2.6 扩容后.....	27
4 资源管理.....	27
4.1 资源管理 resource group 概述.....	27
4.2 资源组属性.....	28
4.3 启用 resource group.....	28
4.4 创建 resource group.....	29
4.5 查看 resource group 信息.....	30
4.6 常见问题和最佳实践.....	30

1 catalog 管理

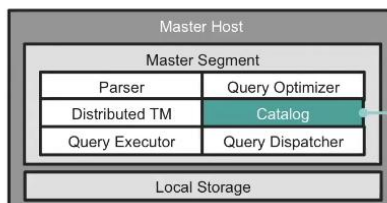
1.1 catalog 基本原理

有些数据是在 Master 上而有些数据是在 Standby Master 上

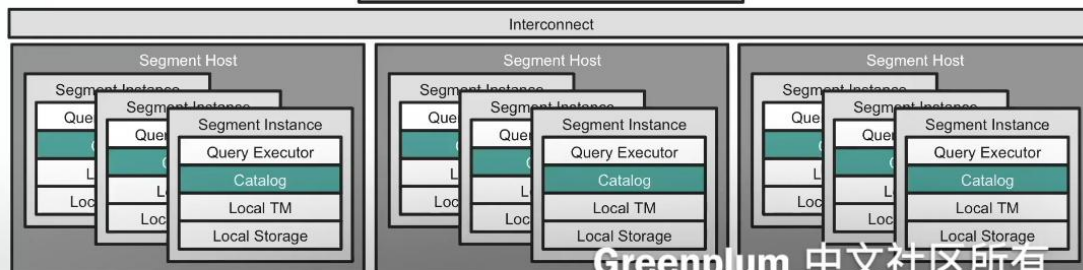
元数据 (Catalog)



存储和管理数据库、表、列等的元数据



大多数元数据在master和segment上同时保存



Greenplum 中文社区所有

1.1.1 仅在 master 保存的元数据

gp_segment_configuration 既在 master 保存元数据, 又在 segment 上可以查询, 但在 segment 上不保存数据

仅在master保存的元数据



```
postgres=# select * from gp_segment_configuration;
dbid | content | role | preferred_role | mode | status | port | hostname | address | replication_port
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
  1 |      -1 | p   | p             | s   | u      | 15432 | myvm2    | myvm2    |
  6 |      -1 | m   | m             | s   | u      | 16432 | myvm2    | myvm2    |
  2 |       0 | p   | p             | s   | u      | 25432 | myvm2    | myvm2    |
  4 |       0 | m   | m             | s   | u      | 25434 | myvm2    | myvm2    |
  3 |       1 | p   | p             | s   | u      | 25433 | myvm2    | myvm2    |
  5 |       1 | m   | m             | s   | u      | 25435 | myvm2    | myvm2    |
(6 rows)
```

```
[gpadmin@myvm2 ~]$ PGOPTIONS='-c gp_session_role=utility' psql -p 25432
psql (8.3.23)
Type "help" for help.

postgres=# select * from gp_segment_configuration;
dbid | content | role | preferred_role | mode | status | port | hostname | address | replication_port
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
(0 rows)
```

Greenplum 中文社区所有

1.1.2 表-元数据表(部分)

表-元数据表（部分）



pg_class	每张表在pg_class中有一行记录（relkind='r'）
pg_attribute	每张表每一列在pg_attribute中有一行记录，包含系统和隐藏列
pg_appendonly	每个AO或者AOCO表在pg_appendonly中有一行，并通过relid列与pg_class.oid关联

Greenplum 中文社区所有

1.1.2.1 查询实例

表-元数据示例



```
$ psql postgres ## master
postgres=# \d t1
      Table "public.t1"
  Column | Type | Modifiers
-----+-----+-----
 c1      | integer |
 c2      | numeric |
Distributed by: (c1)

postgres=# select oid, relname from pg_class where relname = 't1';
 oid | relname
-----+-----
41271 | t1

postgres=# select attrelid, attname from pg_attribute where attrelid = 41271;
attrelid | attname
-----+-----
41271 | c1
41271 | c2
41271 | ctid
41271 | xmin
41271 | cmin
41271 | xmax
41271 | cmax
41271 | tableoid
41271 | gp_segment_id

$ PGOPTIONS='-c gp_session_role=utility' psql -p 25432 ## segment
会看到同样的输出
```

Greenplum 中文社区所有

所有的元数据表以及依赖关系请查看：
<https://stephendotcarter.github.io/greenplum-syscat-ref/diagram.html>



1.2 gpcheckcat 命令的使用

1.2.1 catalog 检查不一致问题

catalog检查工具



gpcheckcat	检查Greenplum数据库系统表的一致性
------------	-----------------------

- segment内不一致
- segment间不一致
- persistent table不一致

示例：

```
$ nohup $GPHOME/bin/lib/gpcheckcat -A -v > ~/gpcheckcat_output_`date`  
%Y%m%d_%H%M%S` 2>&1 &
```

Greenplum 中文社区所有

1.2.2 gpcheckcat 在线测试

在集群中正在运行时可以测试以下选项-o 是全部参数

gpcheckcat: 在线测试



这些测试可以在任何时候执行：

- duplicate
- missing_extraneous
- inconsistent
- foreign_key
- acl
- owner
- part_integrity
- part_constraint
- duplicate_persistent

```
$ gpcheckcat -O
```

Greenplum 中文社区所有

1.2.3 gpcheckcat 离线测试

gpcheckcat: 离线（限制模式）测试



在有其他查询运行时执行这些测试，可能产生误报：

- persistent
- pg_class
- namespace
- distribution_policy
- dependency

Greenplum 中文社区所有

1.2.4 gpcheckcat 提示自动修复

当出现检测问题时，gpcheckcat 会把需要修复的命令存放在当前的目录下，生成的文件是 *.sql 与 *.sh 管理员只需要打开执行就可以

gpcheckcat异常: 修复提示



```
[gpadmin@myvm2 ~]$ gpcheckcat -R owner db_test
Truncated batch size to number of primaries: 3

Connected as user 'gpadmin' to database 'db_test', port '15432', gpdb version '5.19'
-----
Batch size: 3
Performing test 'owner'

catalog issue(s) found , repair script(s) generated in dir gpcheckcat.repair.20190810205158

Total runtime for test 'owner': 0:00:00.08

SUMMARY REPORT
=====
Completed 1 test(s) on database 'db_test' at 2019-08-10 20:51:59 with elapsed time 0:00:00
Failed test(s) that are not reported here: owner
See /home/gpadmin/gpAdminLogs/gpcheckcat_20190810.log for detail

-----
[gpadmin@myvm2 ~]$ ls -l gpcheckcat.repair.20190810205158
total 8
-rw-rw-r-- 1 gpadmin gpadmin 116 Aug 10 20:51 db_test_fixowner_20190810205158.sql
-rwx----- 1 gpadmin gpadmin 188 Aug 10 20:51 runsql_20190810205158.sh

[gpadmin@myvm2 ~]$ cat gpcheckcat.repair.20190810205158/db_test_fixowner_20190810205158.sql
-- owner for "public"."t1"
ALTER TABLE "public"."t1" OWNER TO "wolf";
ALTER TABLE "public"."t1" OWNER TO "gpadmin";

[gpadmin@myvm2 ~]$ ./gpcheckcat.repair.20190810205158/runsql_20190810205158.sh
```

Greenplum 中文社区所有

1.2.5 最佳实践和常见问题

- 定期vacuum catalog
- 不要取消未完成的catalog vacuum
- 如果catalog vacuum长时间没有进展，请检查是否有其他查询阻塞
 - pg_locks
 - pg_stat_activity
 - 其他catalog查询: [known issue 29766](#)
- 定期gpcheckcat

2 master 管理

2.1 master 基本原理和操作

2.1.1 master 管理工具

master管理工具

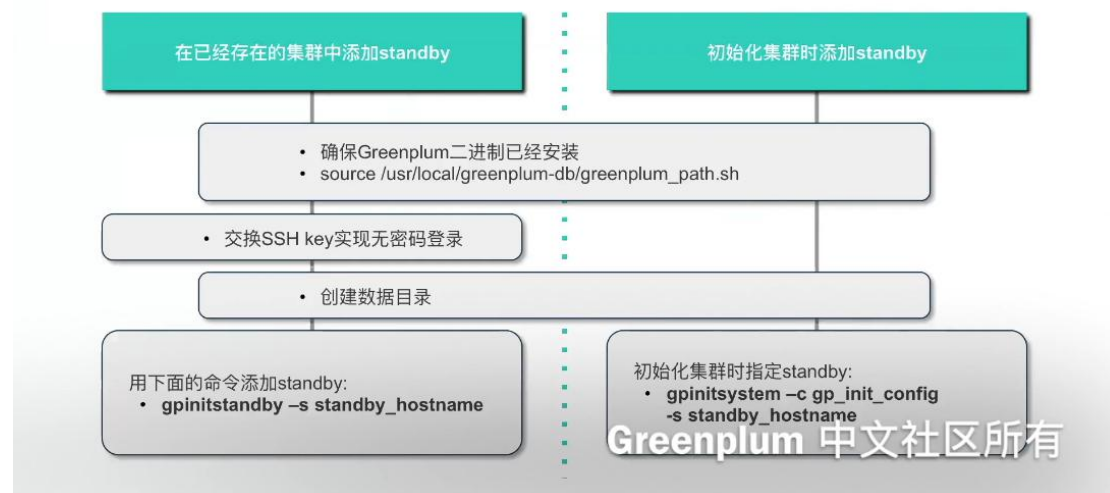


gpinitstandby	添加并初始化一个Standby Master节点
gpactivatstandby	激活Standby Master，使之成为集群中新的Master节点

2.1.2 添加 standby

初始化 standby 使用 gpinitssystem 命令,添加 standby 使用 gpinnitstandby

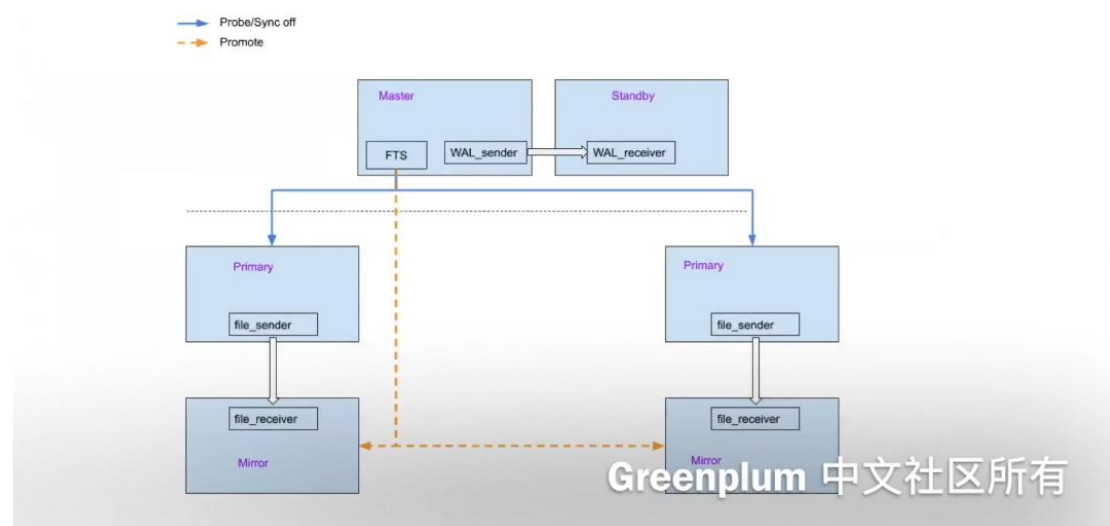
添加 standby



2.1.3 master-standby 数据同步原理

master 与 standby 是通过 WAL(WAL_sedder 与 WAL_receiver)日志的发送和接受来同步数据，master 与 segment 之间是通过 FTS 来进行同步数据

master-standby数据同步



2.1.4 standby 同步状态

standby同步状态



```
[gpadmin@myvm2 pg_log]$ gpstate -f
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:-Starting gpstate with args: -f
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:-local Greenplum Version: 'postgres (Greenplum Database) 5.19.0+
0860b7ca build dev'
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:-master Greenplum Version: 'PostgreSQL 8.3.23 (Greenplum Databas
ev.44.gdf0860b7ca build dev) on x86_64-pc-linux-gnu, compiled by GCC gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-36), 64-b
d on Jun 21 2019 16:40:17'
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:-Obtaining Segment details from master...
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:-Standby master details
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:-----
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:- Standby address          = myvm2
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:- Standby data directory      = /home/gpadmin/workspace/gpdb5/gpA
datadirs/standby
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:- Standby port              = 16432
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:- Standby PID                = 18365
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:- Standby status           = Standby host passive
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:-----
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:--pg_stat_replication
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:-----
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:--WAL Sender State: streaming
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:--Sync state: sync
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:--Sent Location: 2/3E4750D0
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:--Flush Location: 2/3E4750D0
20190811:13:02:36:019238 gpstate:myvm2:gpadmin-[INFO]:--Replay Location: 2/3E4750D0
```

Greenplum 中文社区所有

2.1.5 standby 特定配置

如果 master 与 standby 数据不进行同步了，可以在 standby 上查看 recovery.conf 配置文件

standby特定配置



```
[gpadmin@myvm2 datadirs]$ cat standby/recovery.conf
standby_mode = 'on'
primary_conninfo = 'user=gpadmin host=myvm2 port=15432 sslmode=disable sslcompression=1'
```

Greenplum 中文社区所有

2.1.6 日志查看和收集

日志查看和收集



- 节点:
 - 节点类型: Master, standby, segments
 - 关联节点: master/standby, master/segments, primary/mirror
 - `select * from gp_segment_configuration order by content`
- 日志:
 - `/home/gpadmin/gpAdminLogs/gpstart_<timestamp>.log`
 - `<data_directory>/pg_log/startup.log`
 - `<data_directory>/pg_log/gpdb-<timestamp>.csv`

Greenplum 中文社区所有

2.2 master 故障管理

2.2.1 master 节点异常退出:尝试恢复 master

master节点异常退出: 尝试恢复master

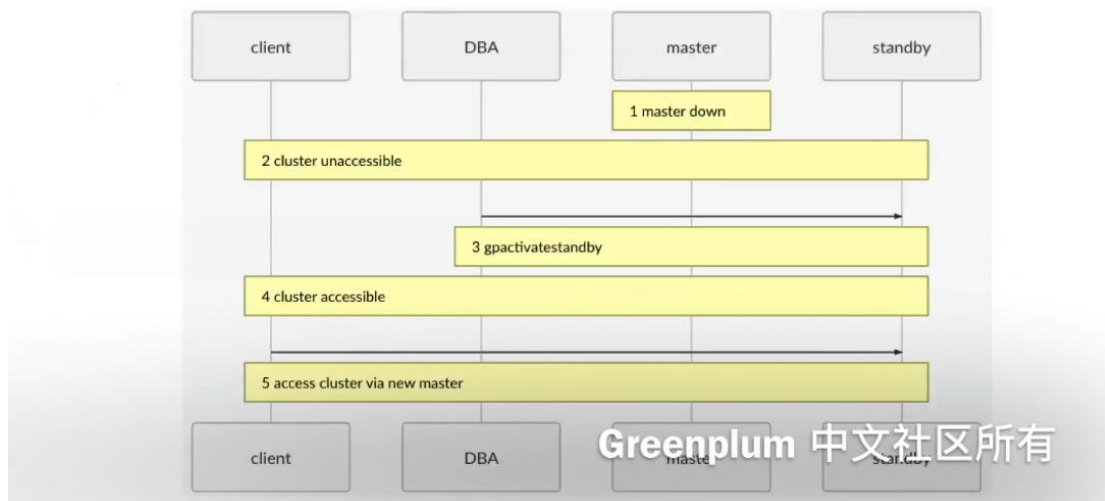


- 核对: 检查master节点和所有segment服务器数据库服务进程
- 尝试只启动master, `gpstart -m`
- 正常停止集群, `gpstop -af`
- 检查所有数据库服务进程停止
- 正常启动集群, `gpstart -a`
- 使用`gpstate`检查数据库状态并执行checkpoint

Greenplum 中文社区所有

2.2.2 master 节点异常退出:master 无法恢复

master节点异常退出: master无法恢复



2.2.3 gpactivatestandby 内部逻辑

gpactivatestandby内部逻辑



- 停止日志接收和重放进程walreceiver
- 使用接收到的日志更新系统catalog
- 将standby master提升为系统新的master
- 使用新的master重启集群

Greenplum 中文社区所有

2.2.4 提升 standby 操作过程

此操作需要谨慎,确保 master 所有的进程已经停止,否则会发生脑裂的情况,数据不一致问

题。

提升standby操作过程



如果原master节点可以访问，在原master上执行：

- 备份master目录\$MASTER_DATA_DIRECTORY
- 确保master进程已停止
- 如果\$MASTER_DATA_DIRECTORY/postmaster.pid还在，删除
- 复制\$MASTER_DATA_DIRECTORY/pg_hba.conf到standby

在standby节点执行：

- 备份standby master数据目录
- **gpactivestandby** -d /data/gpmaster/gpseg-1
- 检查数据库状态并Analyze

Greenplum 中文社区所有

2.2.5 重建 standby 操作过程

重建standby操作过程



- 等待原master物理节点恢复，或者新的standby节点就绪
- 在新的master节点上，使用gpinitstandby及时重建新的standby

Greenplum 中文社区所有

2.3 standby 故障管理

2.3.1 standby 异常

使用 `gpstate -f` 查看 standby 的状态

standby异常



```
[gpadmin@myvm2 pg_log]$ gpstate -f
20190811:13:06:11:019800 gpstate:myvm2:gpadmin-[INFO]:-Starting gpstate with args: -f
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-local Greenplum Version: 'postgres (Greenplum Database) 5.19.0+
0860b7ca build dev
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-master Greenplum Version: 'PostgreSQL 8.3.23 (Greenplum Databas
ev.44.gdf0860b7ca build dev) on x86_64-pc-linux-gnu, compiled by GCC gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-36), 64-b
d on Jun 21 2019 16:40:17'
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-Obtaining Segment details from master...
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-Standby master details
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-----
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:- Standby address          = myvm2
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:- Standby data directory   = /home/gpadmin/workspace/gpdb5/gpA
datadirs/standby
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:- Standby port              = 16432
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[WARNING]:-Standby PID              = 0
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[WARNING]:-Standby status          = Standby process not running
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-----
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-pg_stat_replication
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-----
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-No entries found.
20190811:13:06:12:019800 gpstate:myvm2:gpadmin-[INFO]:-----
```

Greenplum 中文社区所有

2.3.2 standby 异常处理

使用一下命令重启 standby

standby异常处理



尝试重启standby:

- gpinitstandby -n

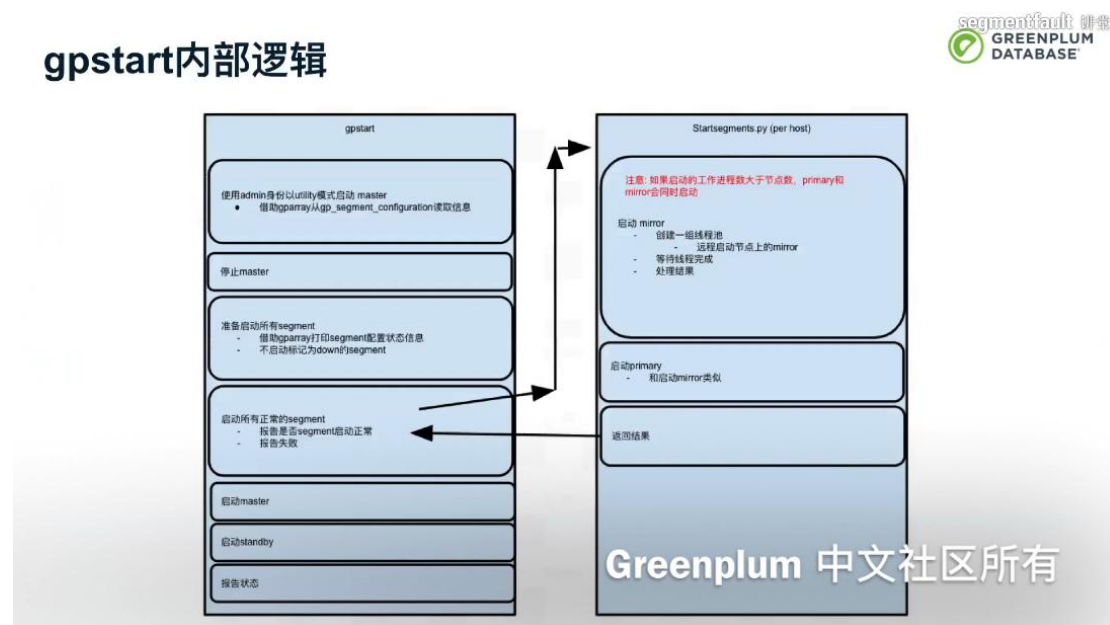
重建standby:

- 备份standby数据目录
- gpinitstandby -r
- gpinitstandby -s smdw

Greenplum 中文社区所有

2.4 集群启动过程

2.4.1 gpstart 内部逻辑



2.4.2 集群无法启动

一下的情况会导致集群无法启动

- 无效的GUC设置: gpconfig, postgresql.conf
- 无效的库: gpconfig -c shared_preload_libraries
- invalid IP mask: 10.1.xx.xxx/32 in pg_hba.conf
- segment connection refused: recovery?
- 启动时间太长: 检查没有完成启动的segment
- verbose模式: gpstart -a -v

Greenplum 中文社区所有

- 1、无效的 GUC 设置:确保 postgresql.conf 确保正确
- 2、无效的库:在安装无效的库
- 3、Invalid IP mask : 配置无效的 pg_hba.conf
- 4、recovery 重启:集群没有正常的关闭, segment 有大量的日志, 会报启动超时
- 5、启动时间过长:检查没有完成请的 segment 的节点
- 6、查看 verbose 详细信息:gpstart -a -v 查看详细信息

3 segment 管理

3.1 基本原理和操作

3.1.1 segment 管理工具

segment管理工具



gpaddmirrors	为Greenplum数据库系统添加Mirror Segment
gprecoverseg	恢复宕机的Primary Segment或Mirror Segment
gpexpand	扩容Greenplum，增加新的Segment节点

Greenplum 中文社区所有

3.1.2 配置 mirror

配置mirror



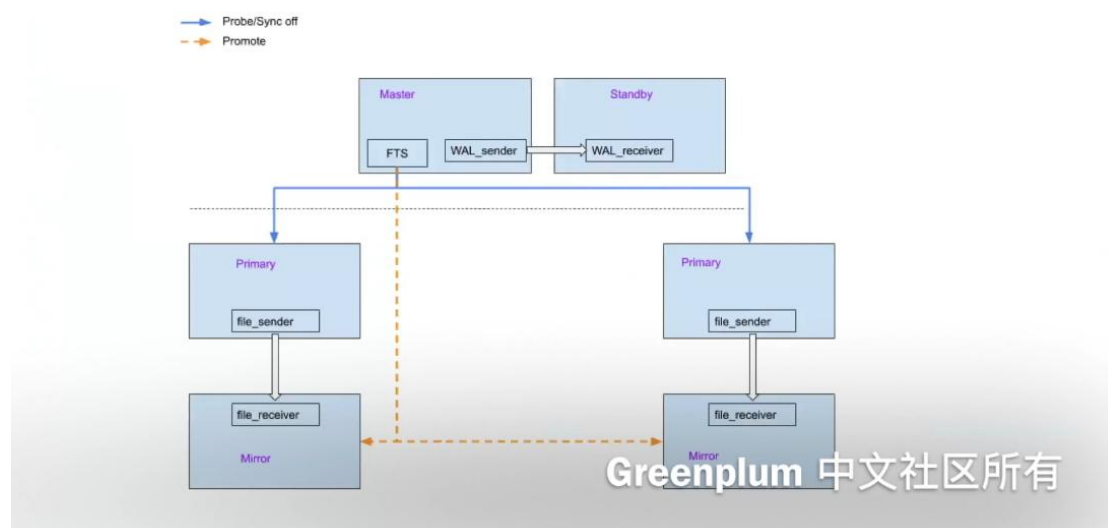
- Mirror做什么用:
 - primary segment的一个备份
 - 数据冗余以提高可用性
- 什么时候启用mirror:
 - 集群初始化时 (在gp_init_config设置参数)
 - 通过**gpaddmirrors**命令往一个运行中的集群添加:
 - gpaddmirrors -o /home/gpadmin/sample_mirror_config
 - gpaddmirrors -i mirror_config_file

Greenplum 中文社区所有

3.1.3 primary-mirror 数据同步

Greenplum5.x 中的 Master 使用 FTS 来进行同步的，FTS 会不断的收集 primary 与 mirror 的信息，查看 primary 与 mirror 是不是都在线和数据是不是都同步，FTS 直连 primary，因为 primary 根据返回的状态知道 mirror 的状态。

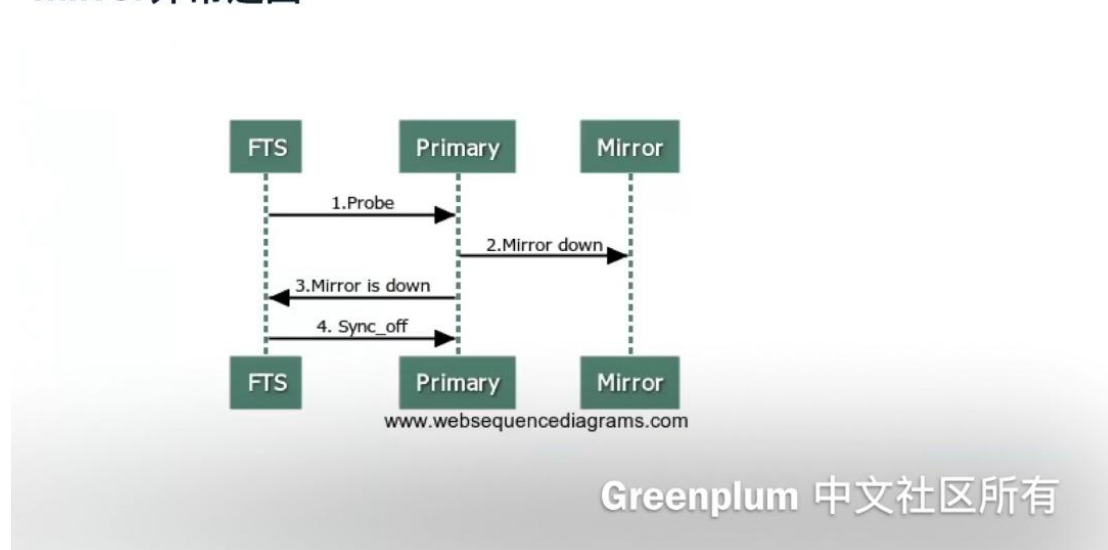
primary-mirror数据同步



3.1.4 mirror 异常退出

mirror 异常退出时，primary 进入到 sync_off 的状态，并记录用户的操作历史，等待 mirror 重新启动后，再进行同步。

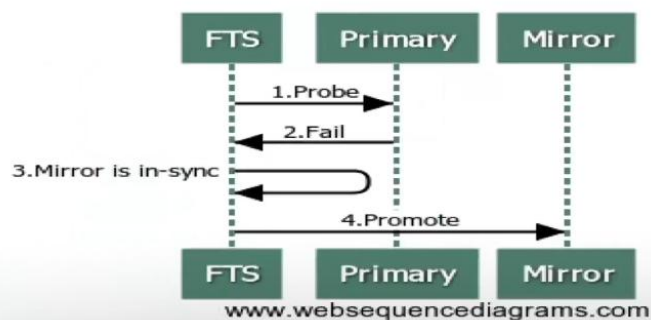
mirror异常退出



3.1.5 primary 异常退出

Primary 异常包括节点异常或进程退出了，FTS 会尝试连接 primary，尝试多次后 primary 无响应，FTS 会去连接相对应的在 sync 状态的 mirror，并把 mirror 提升为 primary，并把 primary 标记为 down，全部过程都是系统自动完成不需要用户的干预。

primary异常退出



Greenplum 中文社区所有

3.2 segment 故障管理

3.2.1 segment 异常退出

segment异常处理



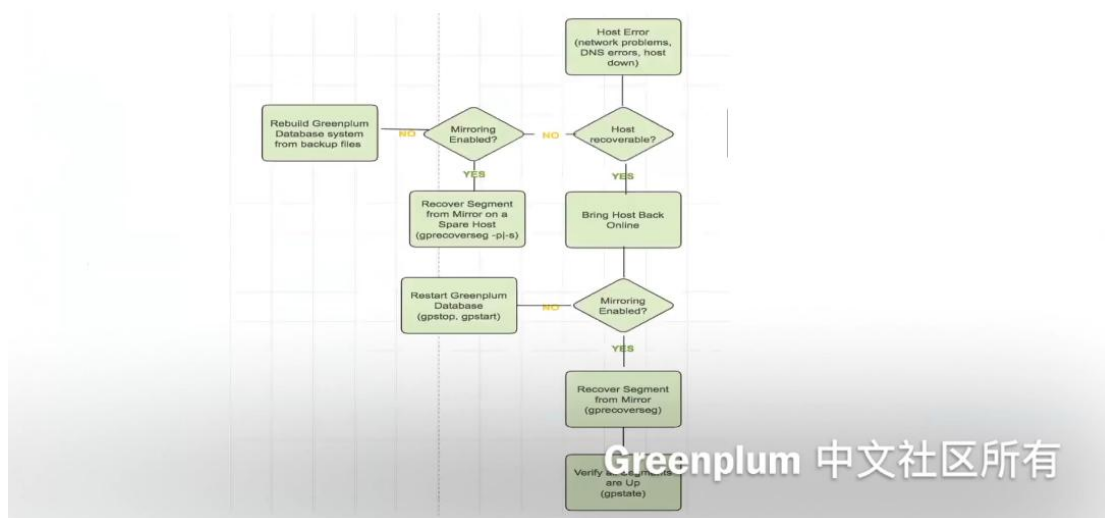
- 确定异常原因或者收集信息:
 - 日志
 - gpstate
 - 错误信息
- 错误分类
 - 主机错误
 - 进程错误
 - 数据错误

Greenplum 中文社区所有

3.2.2 segment 异常处理:主机错误

如果发现主机错误，判断当前的节点是否可以恢复，查看当前的节点是否有对应的 mirror，如果有 mirror 使用 `gprecoverseg` 把失败的节点进行恢复。如果没有 mirror 使用 `gpstop` 和 `gpstart` 对集群进行恢复正常。如果发现物理节点无法恢复，查看是否有 mirror，如果没有 mirror，表示集群无法恢复，只能通过备份文件对新集群进行恢复。发现有 mirror 使用 `gprecoverseg -p|-s` 进行恢复。

segment异常处理：主机错误

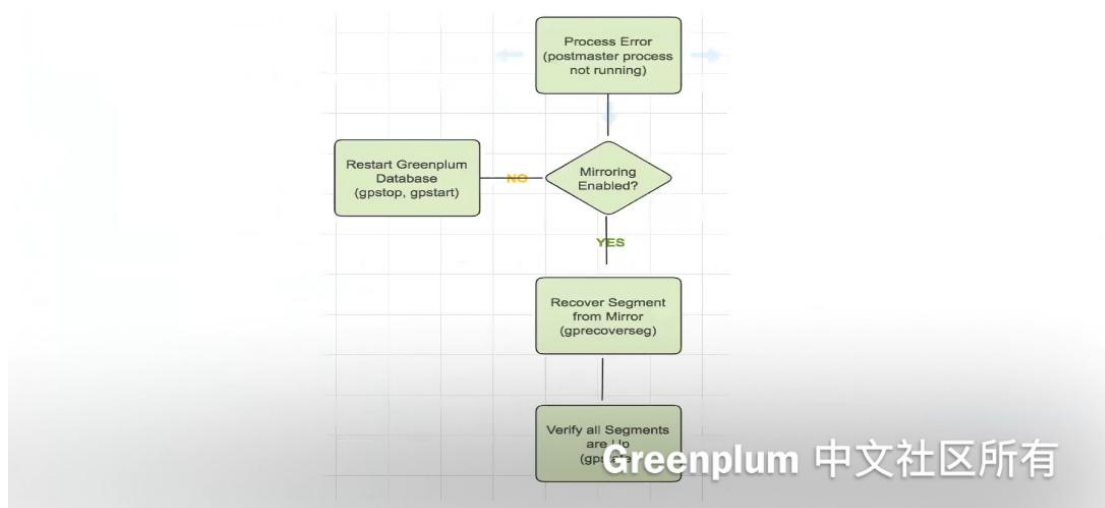


Greenplum 中文社区所有

3.2.3 segment 异常处理:进程错误

如果发现进程有错误，如果有 mirror 可以在集群运行时使用 `gprecoverseg` 来进行恢复，如果没有 mirror 只能使用 `gpstop` 和 `gpstart` 来进行恢复。

segment异常处理：进程错误

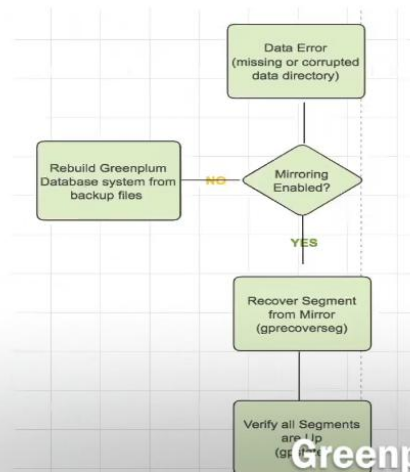


Greenplum 中文社区所有

3.2.4 segment 异常处理:数据错误

当数据发生错误是，如果有 mirror 可以使用 `gprecoverseg` 来恢复，如果没有 mirror 可以通过备份文件来恢复。

segment异常处理：数据错误

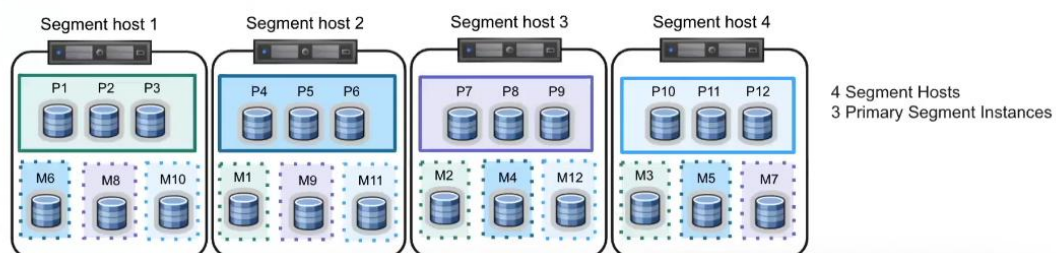


Greenplum 中文社区所有

3.2.5 primary segment 异常处理:再平衡

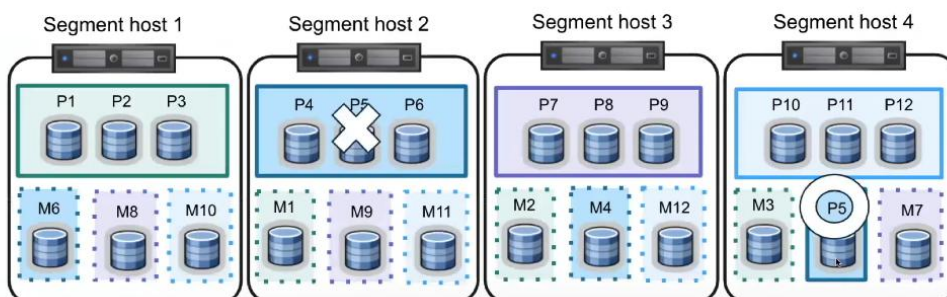
当使用 `gprecoverseg` 进行恢复完成后，请在使用 `gprecoverseg -r` 把 primarr 和 mirror 角色互换过来，是集群达到平衡。

primary segment异常处理：再平衡（1）



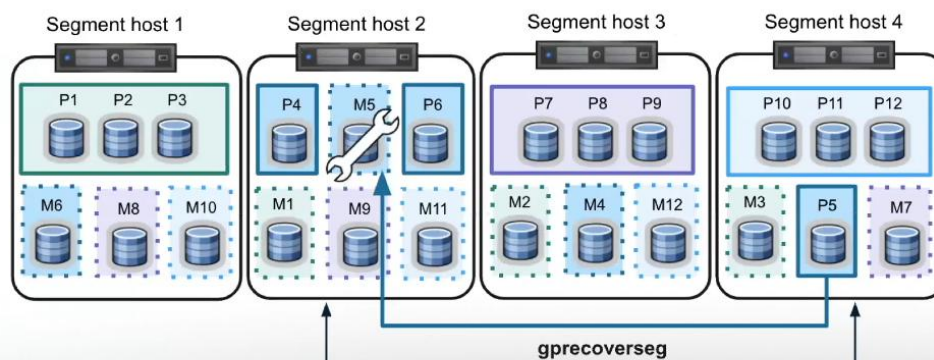
Greenplum 中文社区所有

primary segment异常处理：再平衡（2）



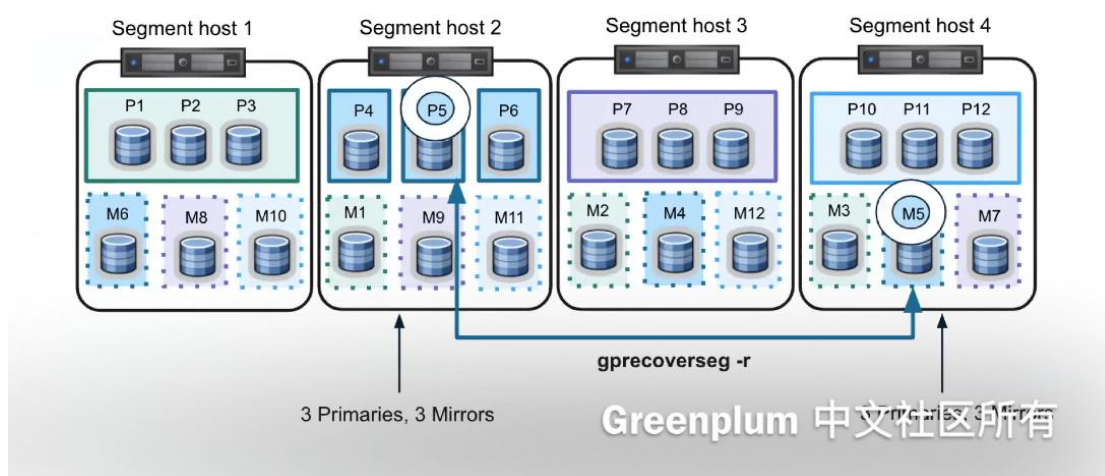
Greenplum 中文社区所有

primary segment异常处理：再平衡（3）



Greenplum 中文社区所有

primary segment异常处理：再平衡（4）



3.2.6 增量和全量恢复

如果使用增量的方式表示从上次终端的点进行恢复，建议先使用这种方式。
如果增量出错了或 catalog 修改过，使用全量的方式。

增量和全量恢复

- gprecoverseg -a (增量)
- gprecoverseg -aF (全量)

3.2.7 gprecoverseg 常见错误

- gprecoverseg成功但是gpstate显示mirror还是down
 - 检查primary和master日志，可能数据损坏
- 发现warning: Failed to inform primary segment of updated mirroring state
 - gpstate -e
 - 检查列Objects to resync的内容是否还在变化

3.3 集群扩容

3.3.1 集群扩容准备

扩容



- 目的：增加资源
 - 存储
 - 计算：CPU和内存
- 准备：
 - 用gpsegininstall在新segment主机上安装Greenplum
 - 配置SSH免密登录，调整内核参数
- 过程：
 - 1 生成input文件
 - 2 添加节点
 - 3 重分布数据
 - 4 清除
- 期望：
 - 数据分布均匀
 - 最小化扩容对业务的影响

Greenplum 中文社区所有

3.3.2 扩容实例

3.3.2.1 扩容之前的状态

注意以下只是测试的环境

扩容示例：扩容前



```
postgres=# select * from gp_segment_configuration ;
dbid | content | role | preferred_role | mode | status | port | hostname | address |          datadir
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
 1 |   -1 | p | p | n | u | 5432 | mdw | mdw | /home/gpadmin/test/master/gpseg-1
 2 |    0 | p | p | n | u | 6000 | sdw1 | sdw1 | /home/gpadmin/test/p1/gpseg0
 5 |    3 | p | p | n | u | 6000 | sdw2 | sdw2 | /home/gpadmin/test/p1/gpseg3
 3 |    1 | p | p | n | u | 6001 | sdw1 | sdw1 | /home/gpadmin/test/p2/gpseg1
 6 |    4 | p | p | n | u | 6001 | sdw2 | sdw2 | /home/gpadmin/test/p2/gpseg4
 4 |    2 | p | p | n | u | 6002 | sdw1 | sdw1 | /home/gpadmin/test/p3/gpseg2
 7 |    5 | p | p | n | u | 6002 | sdw2 | sdw2 | /home/gpadmin/test/p3/gpseg5
(7 rows)
```

Greenplum 中文社区所有

3.3.2.2 交互式生成输入文件

扩容示例：1 交互式生成输入文件



```
$ gpexpand
```

```
Please refer to the Admin Guide for more information.
Would you like to initiate a new System Expansion Yy|Nn (default=N):
y
Enter a comma separated list of new hosts you want
to add to your array. Do not include interface hostnames.
Enter a blank line to only add segments to existing hosts[]:
sdw3
By default, new hosts are configured with the same number of primary segments as
existing hosts. Optionally, you can increase the number of segments per host. Fo
r example, if existing hosts have two primary segments, entering a value of 2 wil
l initialize two additional segments on existing hosts, and four segments on new
hosts. In addition, mirror segments will be added for these new primary segments
if mirroring is enabled.
How many new primary segments per host do you want to add? (default=0):
0
Generating configuration file...
20190423:16:38:49:028224 gpexpand:mdw:gpadmin-[INFO]:-Generating input file...
Input configuration file was written to 'gpexpand_inputfile_20190423_163849'.
Please review the file and make sure that it is correct then re-run
with: gpexpand -i gpexpand_inputfile_20190423_163849
20190423:16:38:49:028224 gpexpand:mdw:gpadmin-[INFO]:-Exiting...
```

Greenplum 中文社区所有

3.3.2.3 添加节点

扩容示例: 2 添加节点



```
$ gpexpand -i gpexpand_inputfile_20190423_163849 -D postgres
```

- 在这一步最后，gpexpand会自动重启数据库
- 将所有表的分布方式改为Random分布
- 在新节点没有添加到系统Catalog之前出错，可以回滚，gpexpand -r -D postgres

Greenplum 中文社区所有

3.3.2.4 数据重分布

扩容示例: 3 数据重分布



- 在重分布之前，可以更改db1中的系统表gpexpand.status_detail来控制重分布表的顺序
 - UPDATE gpexpand.status_detail SET rank=10;
 - UPDATE gpexpand.status_detail SET rank=2 WHERE fq_name = 'public.orders';
- 开始重分布，分布期间，集群在线可以访问:
\$ gpexpand
- 查看重分布状态:

```
=# SELECT * FROM gpexpand.expansion_progress;
      name      |      value
-----+-----
Bytes Left      | 5534842880
Bytes Done      | 142475264
Estimated Expansion Rate | 680.75667095996092 MB/s
Estimated Time to Completion | 00:01:01.008047
Tables Expanded | 4
Tables Left     | 4
(6 rows)
```

Greenplum 中文社区所有

3.3.2.5 清除扩容的状态

通过 gpexpand -c 清除集群的扩容后的状态

扩容示例: 4 清除扩容状态

```
$ gpexpand -c
```

Greenplum 中文社区所有

3.3.2.6 扩容后

通过 `select * from gp_segment_configuration;` 查看同步的状态

扩容示例: 扩容后

```
postgres=# select * from gp_segment_configuration ;
dbid | content | role | preferred_role | mode | status | port | hostname | address |          datadir
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
  1 |    -1 | p   | p              | n    | u      | 5432 | mdw      | mdw      | /home/gpadmin/test/master/gpseg-1
  2 |     0 | p   | p              | n    | u      | 6000 | sdw1     | sdw1     | /home/gpadmin/test/p1/gpseg0
  5 |     3 | p   | p              | n    | u      | 6000 | sdw2     | sdw2     | /home/gpadmin/test/p1/gpseg3
  3 |     1 | p   | p              | n    | u      | 6001 | sdw1     | sdw1     | /home/gpadmin/test/p2/gpseg1
  6 |     4 | p   | p              | n    | u      | 6001 | sdw2     | sdw2     | /home/gpadmin/test/p2/gpseg4
  4 |     2 | p   | p              | n    | u      | 6002 | sdw1     | sdw1     | /home/gpadmin/test/p3/gpseg2
  7 |     5 | p   | p              | n    | u      | 6002 | sdw2     | sdw2     | /home/gpadmin/test/p3/gpseg5
  8 |     6 | p   | p              | n    | u      | 6000 | sdw3     | sdw3     | /home/gpadmin/test/p1/gpseg6
  9 |     7 | p   | p              | n    | u      | 6001 | sdw3     | sdw3     | /home/gpadmin/test/p2/gpseg7
 10 |     8 | p   | p              | n    | u      | 6002 | sdw3     | sdw3     | /home/gpadmin/test/p3/gpseg8
(10 rows)
```

Greenplum 中文社区所有

4 资源管理

4.1 资源管理 resource group 概述

Greenplum 5.1x 支持 resource group 分配资源

资源管理resource group概述

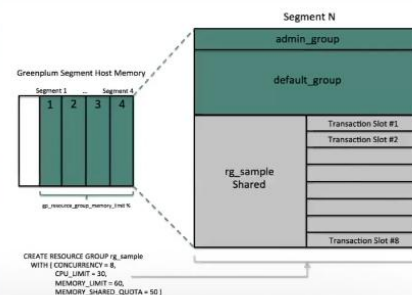
- 资源
 - CPU
 - 内存
- 目标：
 - 在不同租户和不同业务负载之间实现资源隔离
 - 最大化系统资源利用率
 - 确保系统稳定
- 特性：
 - 利用Linux kernel cgroup来分配CPU
 - cpuset
 - cpu_rate_limit
 - 如果一个用cpu_rate_limit的组CPU空闲，可以被其他组超额使用
 - 指定最大并发，按事务计算
 - 内存可以完全隔离，也可以充分共享

Greenplum 中文社区所有

4.2 资源组属性

资源组属性

名称	描述
MEMORY_AUDITOR	内存是由Greenplum管理 (vmtracker) 还是cgroup管理. 缺省是vmtracker, 如果一个组是给其他外部模块使用, 例如PL/Container, 需要制定为cgroup并把concurrency设为0
CONCURRENCY	最大并发事务数
CPU_RATE_LIMIT	CPU份额, 所有组的CPU_RATE_LIMIT之和小于100
CPuset	CPU核的编号, 与CPU_RATE_LIMIT二选一, 为减少延迟, 可以给一些组制定特定的CPU核
MEMORY_LIMIT	内存份额, 所有组memory_limit之和小于100.可以设为0, 让所有组共享内存。
MEMORY_SHARED_QUOTA	组内共享的内存比例, 内存较小或者并发较高时, 建议此值设大一些, 例如80
MEMORY_SPILL_RATIO	0-100, 越大表示一条语句的初始内存越多。0表示初始内存为固定的系统statement_mem。



Greenplum 中文社区所有

4.3 启用 resource group

可以使用 root 用户创建 gpdb.conf 文件

启用resource group

- 建议操作系统CentOS/Redhat 7.x, SuSE 12
- 创建cgroups配置文件

```
# vi /etc/cgconfig.d/gpdb.conf
```

- 安装cgroups系统服务

```
# yum install libcgroup-tools  
# cgconfigparser -l /etc/cgconfig.d/gpdb.conf  
# systemctl enable cgconfig.service
```

- 启用

```
$ gpconfig -c gp_resource_manager -v group  
$ gpstop -af  
$ gpstart -a
```

```
group gpdb {  
  perm {  
    task {  
      uid = gpadmin;  
      gid = gpadmin;  
    }  
    admin {  
      uid = gpadmin;  
      gid = gpadmin;  
    }  
  }  
  cpu {  
    cpucact {  
    }  
    memory {  
    }  
    cpuset {  
    }  
  }  
}
```

Greenplum 中文社区所有

4.4 创建 resource group

创建resource group

- 创建group
 - 至少提供CPU_RATE_LIMIT（或者CPUSET）和MEMORY_LIMIT

```
CREATE RESOURCE GROUP rgoltp WITH (CPU_RATE_LIMIT=30, MEMORY_LIMIT=0,  
memory_spill_ratio=0);  
CREATE RESOURCE GROUP rgadhoc WITH (CPUSET='2,3', MEMORY_LIMIT=0,  
memory_spill_ratio=0);  
CREATE RESOURCE GROUP rgbatch WITH (CPU_RATE_LIMIT=30, MEMORY_LIMIT=30,  
MEMORY_SPILL_RATIO=0, MEMORY_SHARED_QUOTA=90);
```

- 将resource group和用户关联

```
ALTER ROLE uoltp RESOURCE GROUP rgoltp;  
CREATE ROLE uoltp RESOURCE GROUP rgoltp;
```

Greenplum 中文社区所有

4.5 查看 resource group 信息

查看resource group信息



- 查看资源组的设置

```
SELECT * FROM gp_toolkit.gp_resgroup_config;
```

- 查看查询状态和CPU/内存使用

```
SELECT * FROM gp_toolkit.gp_resgroup_status;
```

- 查看资源组和用户关联

```
SELECT rolname, rsgname
FROM pg_roles, pg_resgroup
WHERE pg_roles.rolresgroup = pg_resgroup.oid;
```

- 查看每个组运行和等待的语句

```
SELECT rolname, g.rsgname, procpid, waiting, current_query, datname
FROM pg_roles, gp_toolkit.gp_resgroup_status g, pg_stat_activity
WHERE pg_roles.rolresgroup = g.groupid
AND pg_stat_activity.username = pg_roles.rolname;
```

Greenplum 中文社区所有

4.6 常见问题和最佳实践

详细的可以查看:<http://greenplum.org/calc>

- 正确设置vm.overcommit_ratio, 建议90或者95
- 可以在会话级别设置memory_spill_ratio或者statement_mem
- 更大的memory_spill_ratio或者statement_mem可能会让语句避免spill, 运行更快, 但也可能导致OOM
- 大多数OOM都和资源组的设置有关, 不正确的设置会让每个语句的初始内存太小
- 对小查询, 小的memory_spill_ratio或者statement_mem可能更好
 - set memory_spill_ratio=0;
 - set statement_mem='10MB'
- 如果还在使用resource queue, 请正确设置gp_vmem_protect_limit
 - <http://greenplum.org/calc>

Greenplum 中文社区所有