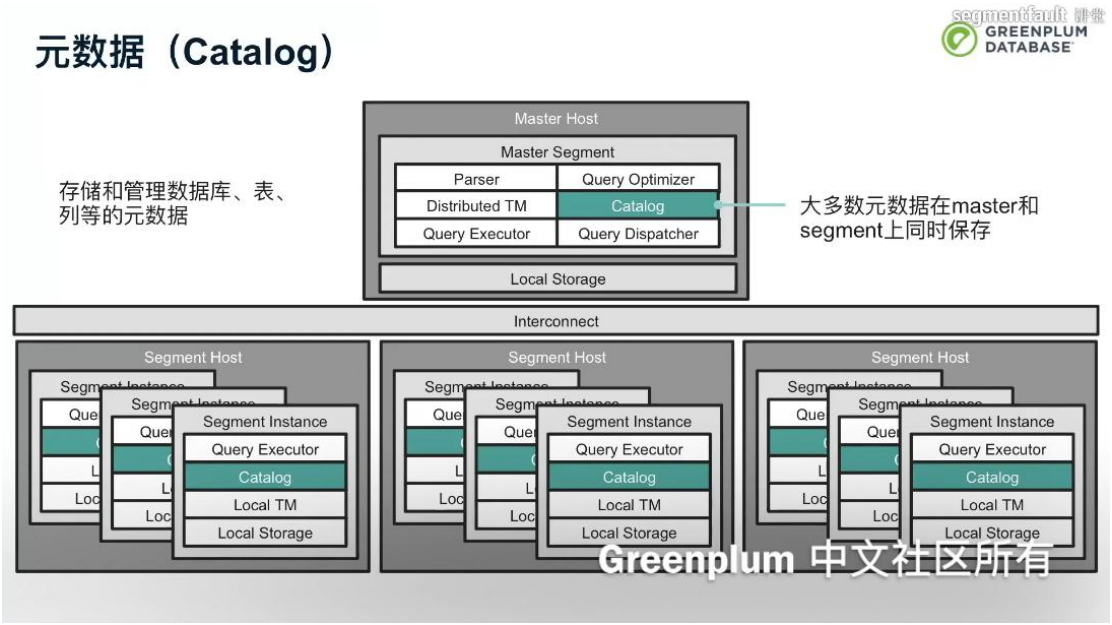
Greenplum 管理实战

目录

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

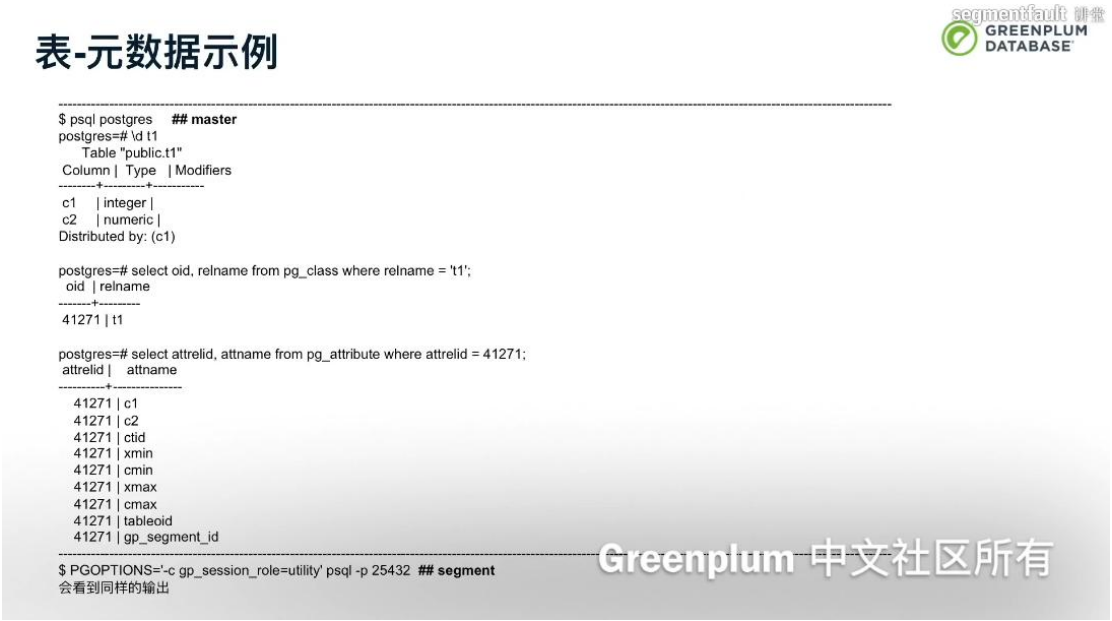


1.1.1 仅在 master 保存的元数据

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

1.1.2 表-元数据表(部分)

1.1.2.1 查询实例

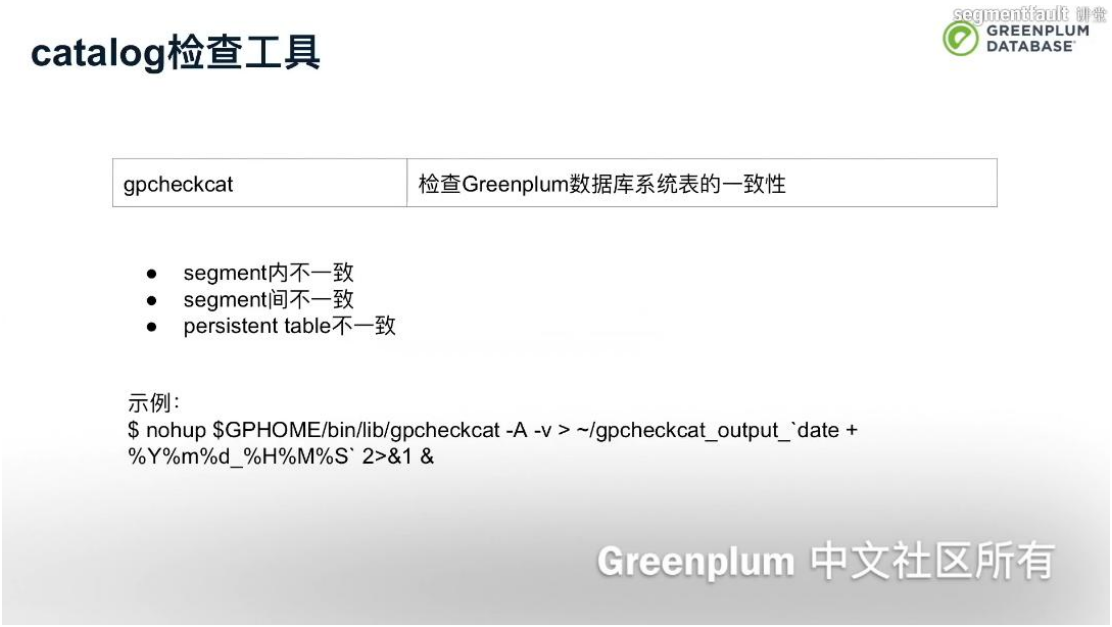


所有的元数据表以及依赖关系请查看:
https://stephendotcarter.github.io/greenplum-syscat-ref/diagram.html


pg_type - typanalyze
pg_type - typbasetype
pg_type - typelem
pg_type - typinput
pg_type - typnamespace
pg_type - typoutput
pg_type - typowner
pg_type - typreceive
pg_type - typrelid
pg_type - typsend
pg_user_mapping - umuser
pg_window - winfinfunc
pg_window - winfnoid
pg_window - winfunc
pg_window - winprefunc
pg_window - winpretype
gp_configuration_history - dbid
gp_db_interfaces - dbid
gp_db_interfaces - interfaceid
gp_distributed_log - dbid
gp_distributed_log - segment_id
gp_distribution_policy - attrnums
gp_distribution_policy - localoid
gp_fastsequence - objid
gp_interfaces - interfaceid
gp_persistent_database_node - database_oid
gp_persistent_database_node - tablespace_oid
gp_persistent_filespace_node - filespace_oid
gp_persistent_relation_node - database_oid
gp_persistent_relation_node - relfilenode_oid
gp_persistent_relation_node - tablespace_oid
gp_persistent_tablespace_node - filespace_oid
gp_persistent_tablespace_node - tablespace_oid
gp_pgdatabase - content
gp_pgdatabase - dbid
gp_pgdatabase - definedprimary
gp_pgdatabase - isprimary
gp_relation_node - relfilenode_oid
gp_san_configuration - oid
gp_segment_configuration - content
gp_segment_configuration - dbid
gp_segment_configuration - preferred_role
gp_segment_configuration - role
gp_segment_configuration - san_mounts
gp_transaction_log - dbid
gp_transaction_log - segment_id
pg_aggregate - aggfinalfn
pg_aggregate - aggfnoid
pg_aggregate - agginvprelimfn
pg_aggregate - agginvtransfn
pg_aggregate - aggsortop
pg_aggregate - aggtransfn
pg_aggregate - aggtranstype
pg_am - ambeginscan
pg_am - ambuild
pg_am - ambulkdelete
pg_am - amcostestimate
pg_am - amendscan
pg_am - amgetmulti
pg_am - amgettuple
pg_am - aminsert
pg_am - ammarkpos
pg_am - amoptions
pg_am - amrescan
pg_am - amrestrpos
pg_am - amvacuumcleanup
pg_am - oid
pg_amop - amopclaid
pg_amop - amopopr
pg_amop - amopsubtype
pg_amproc - amopclaid
pg_amproc - amproc
pg_amproc - amprocsubtype
pg_attrdef - adnum
pg_attrdef - adrelid
pg_attribute - attnum
pg_attribute - attrelid
pg_attribute - atttypid
pg_auth_members - grantor
pg_auth_members - member
pg_auth_members - roleid
pg_authid - oid
pg_cast - castfunc
pg_cast - castsource
pg_cast - casttarget
pg_class - oid
pg_class - relam
pg_class - relfilenode
pg_class - relnamespace
pg_class - relowner
pg_class - reltablespace
pg_class - reltoastidxid
pg_class - reltoastrelid
pg_class - reltype
pg_constraint - confkey
pg_constraint - confrelid
pg_constraint - conkey
pg_constraint - connamespace
pg_constraint - conrelid
pg_constraint - contypid
pg_conversion - connamespace
pg_conversion - conowner
pg_conversion - conproc
pg_database - datdba
pg_database - dattablespace
pg_database - oid
pg_depend - classid
pg_depend - refclassid
pg_description - classoid
pg_exttable - fmterrtbl
pg_exttable - reloid
pg_filespace - fsowner
pg_filespace - oid
pg_filespace_entry - fsedbid
pg_filespace_entry - fsefsoid
pg_index - indclass
pg_index - indexrelid
pg_index - indrelid
pg_inherits - inhparent
pg_inherits - inhrelid
pg_language - lanplcallfoid
pg_language - lanvalidator
pg_language - oid
pg_locks - classid
pg_locks - database
pg_locks - relation
pg_namespace - nspowner
pg_namespace - oid
pg_opclass - oid
pg_opclass - opcamid
pg_opclass - opcintype
pg_opclass - opckeytype
pg_opclass - opcnamespace
pg_opclass - opcowner
pg_operator - oid
pg_operator - oprcode
pg_operator - oprcom
pg_operator - oprgtcmpop
pg_operator - oprjoin
pg_operator - oprleft
pg_operator - oprlsortop
pg_operator - oprltcmpop
pg_operator - oprnamespace
pg_operator - oprnegate
pg_operator - oprowner
pg_operator - oprrest
pg_operator - oprresult
pg_operator - oprright
pg_operator - oprrsortop
pg_partition - oid
pg_partition - parrelid
pg_partition - parclass
pg_partition_rule - parchildrelid
pg_partition_rule - paroid
pg_partition_rule - parparentrule
pg_proc - oid
pg_proc - proallargtypes
pg_proc - proargtypes
pg_proc - prolang
pg_proc - pronamespace
pg_proc - proowner
pg_proc - prorettype
pg_resourcetype - restypid
pg_resqueue - oid
pg_resqueue - rsqname
pg_resqueue_attributes - rsqname
pg_resqueuecapability - restypid
pg_resqueuecapability - rsqueueid
pg_rewrite - ev_class
pg_roles - oid
pg_roles - rolresqueue
pg_shdepend - classid
pg_shdepend - dbid
pg_shdepend - refclassid
pg_shdescription - classoid
pg_stat_activity - datid
pg_stat_activity - usesysid
pg_stat_last_operation - classid
pg_stat_last_operation - stasysid
pg_stat_last_shoperation - classid
pg_statistic - staattnum
pg_statistic - staopN
pg_statistic - starelid
pg_tablespace - oid
pg_tablespace - spcfsoid
pg_tablespace - spcowner
pg_trigger - tgconstrrelid
pg_trigger - tgfoid
pg_trigger - tgrelid
pg_type - oid

1.2 gpcheckcat 命令的使用

1.2.1 catalog 检查不一致问题



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

1.2.3 gpcheckcat 离线测试

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

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

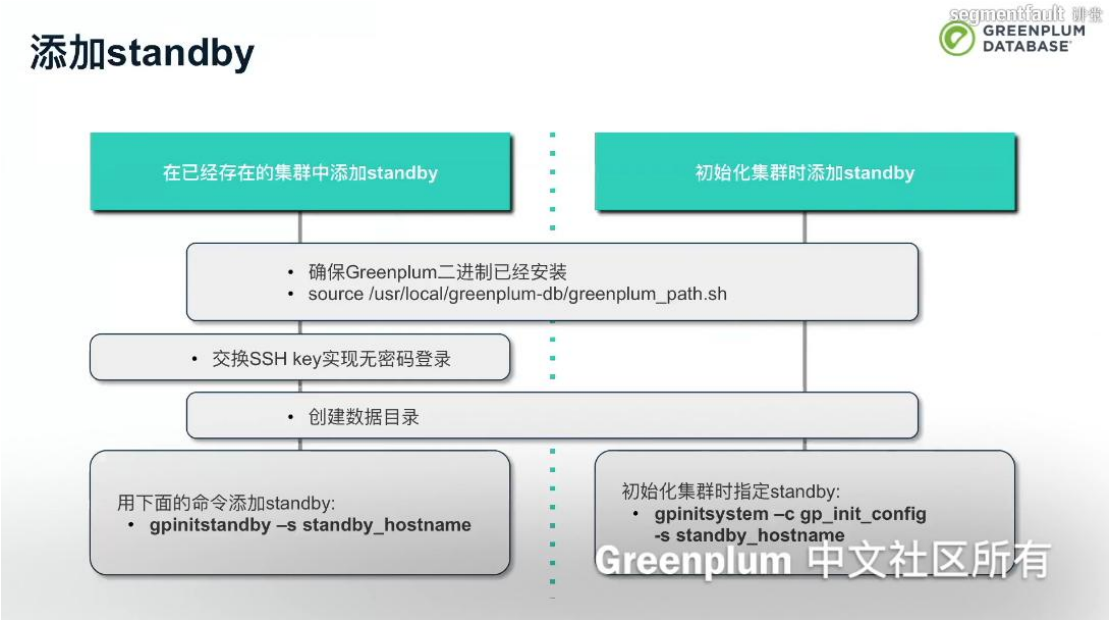
2.1.1 master 管理工具

master管理工具

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

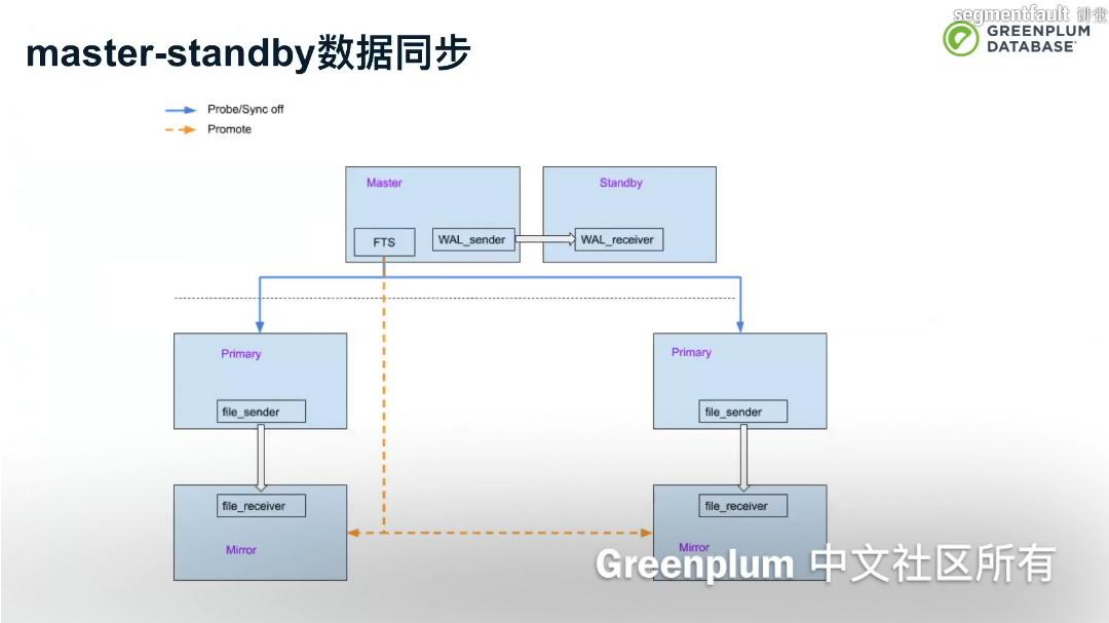
2.1.2 添加 standby

初始化 standby 使用 gpinitsystem 命令,添加 standby 使用 gpinnitstandby



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



2.1.4 standby 同步状态

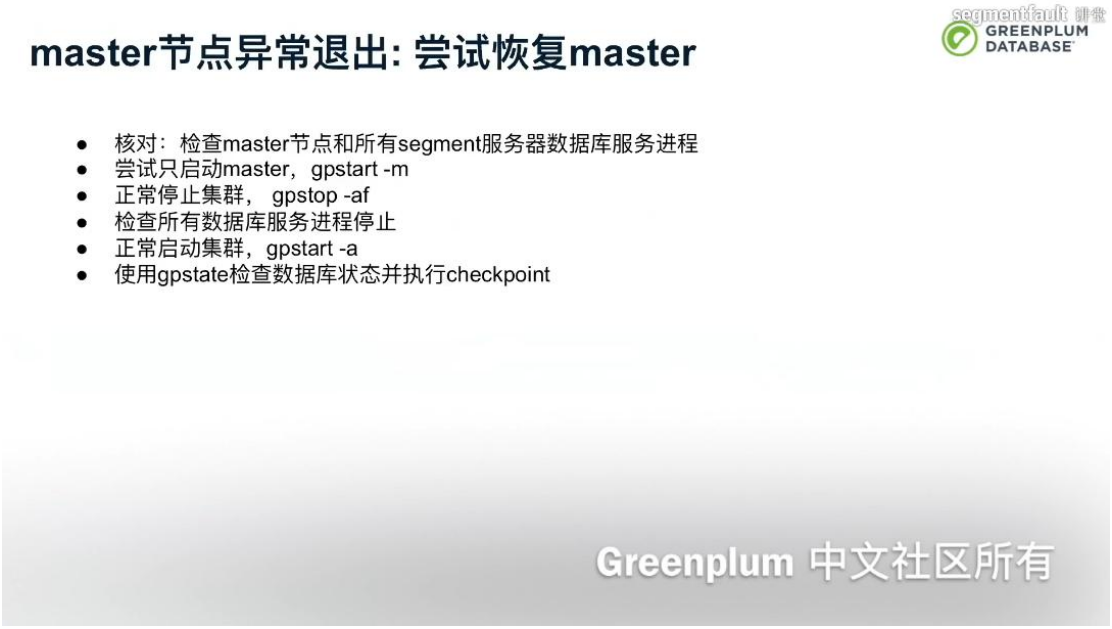
2.1.5 standby 特定配置

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

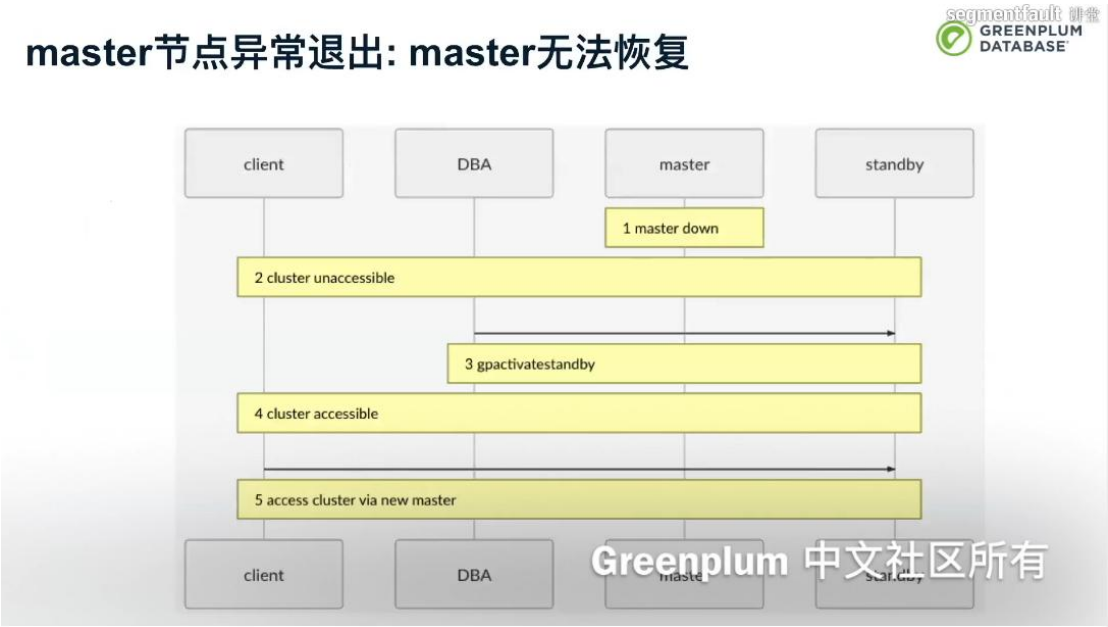
2.1.6 日志查看和收集

2.2 master 故障管理

2.2.1 master 节点异常退出:尝试恢复 master



2.2.2 master 节点异常退出:master 无法恢复



2.2.3 gpactivatestandby 内部逻辑

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。

2.2.5 重建 standby 操作过程

2.3 standby 故障管理

2.3.1 standby 异常

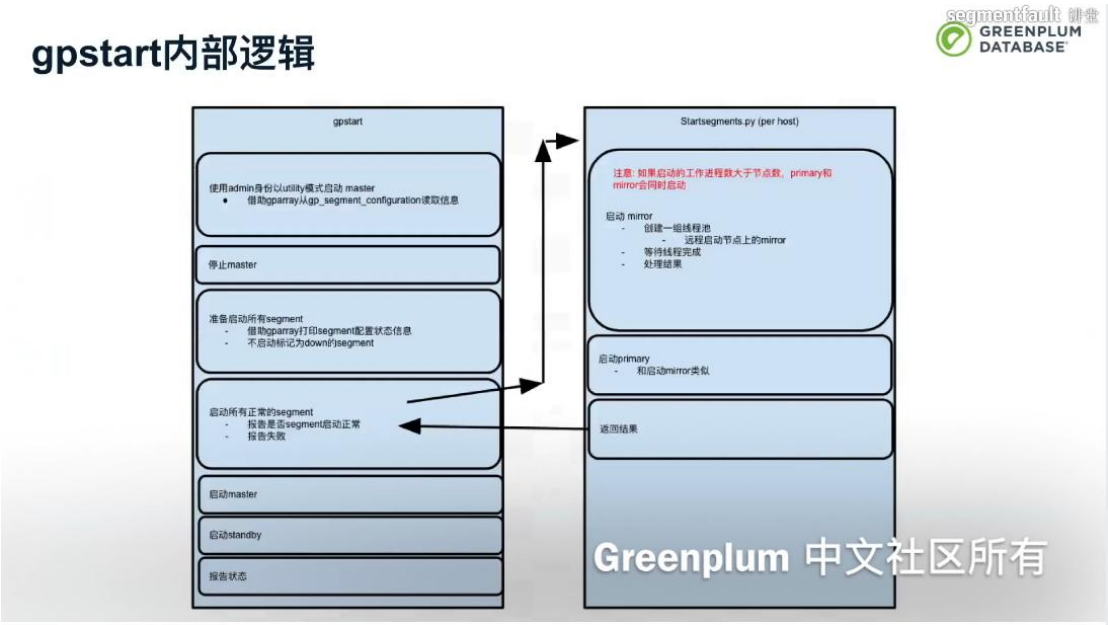
使用 gpstate -f 查看 standby 的状态

2.3.2 standby 异常处理

使用一下命令重启 standby

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

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

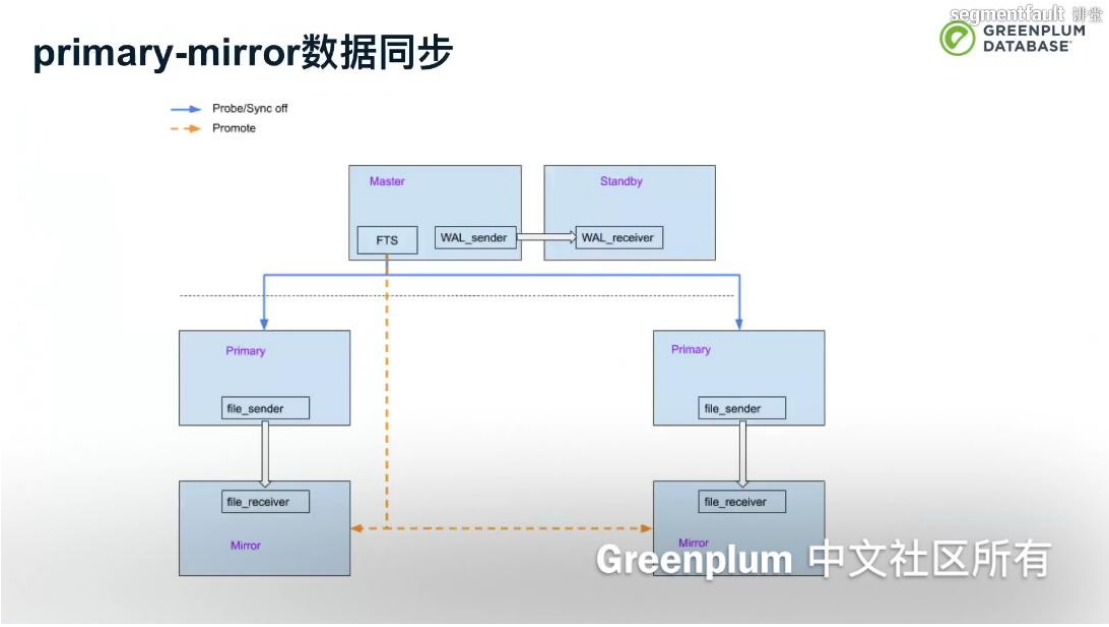
3.1 基本原理和操作

3.1.1 segment 管理工具

3.1.2 配置 mirror

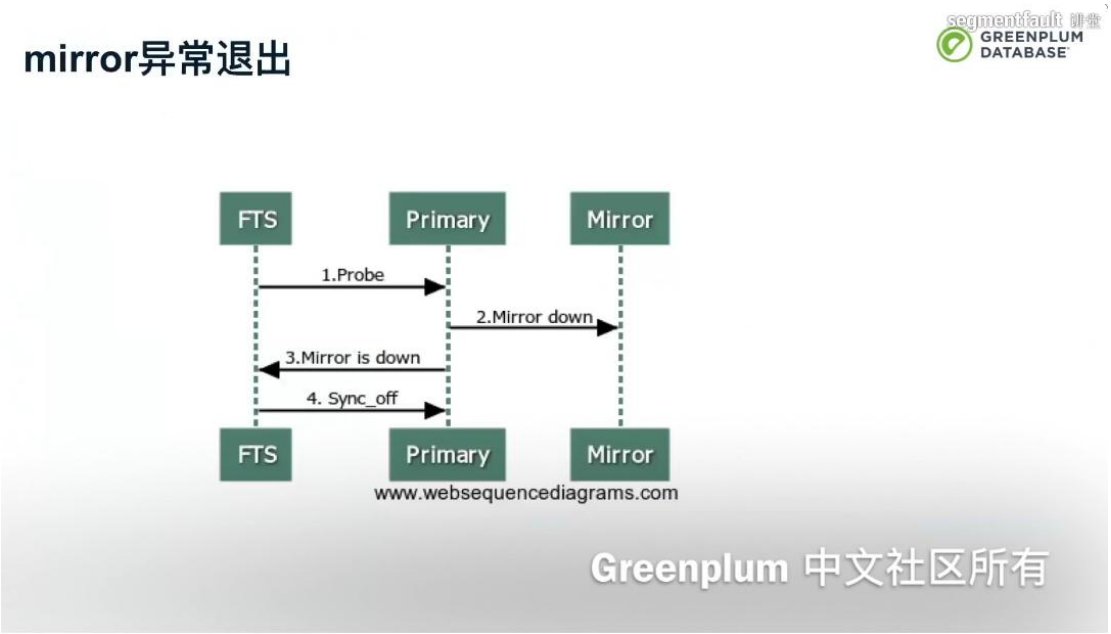
3.1.3 primary-mirror 数据同步

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。



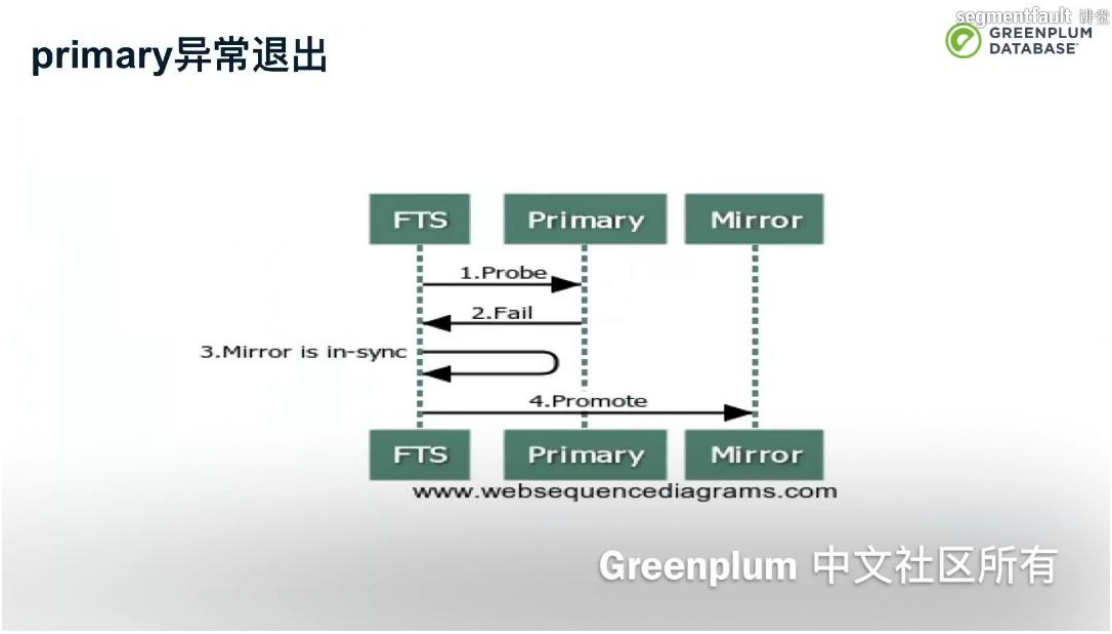
3.1.4 mirror 异常退出

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。



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。

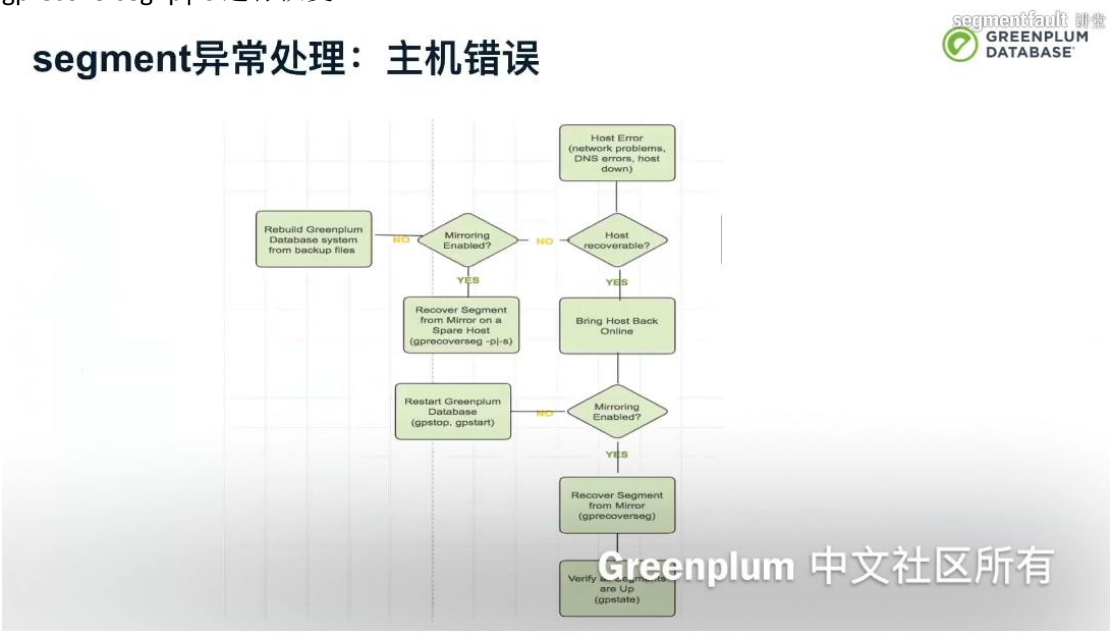


3.2 segment 故障管理

3.2.1 segment 异常退出

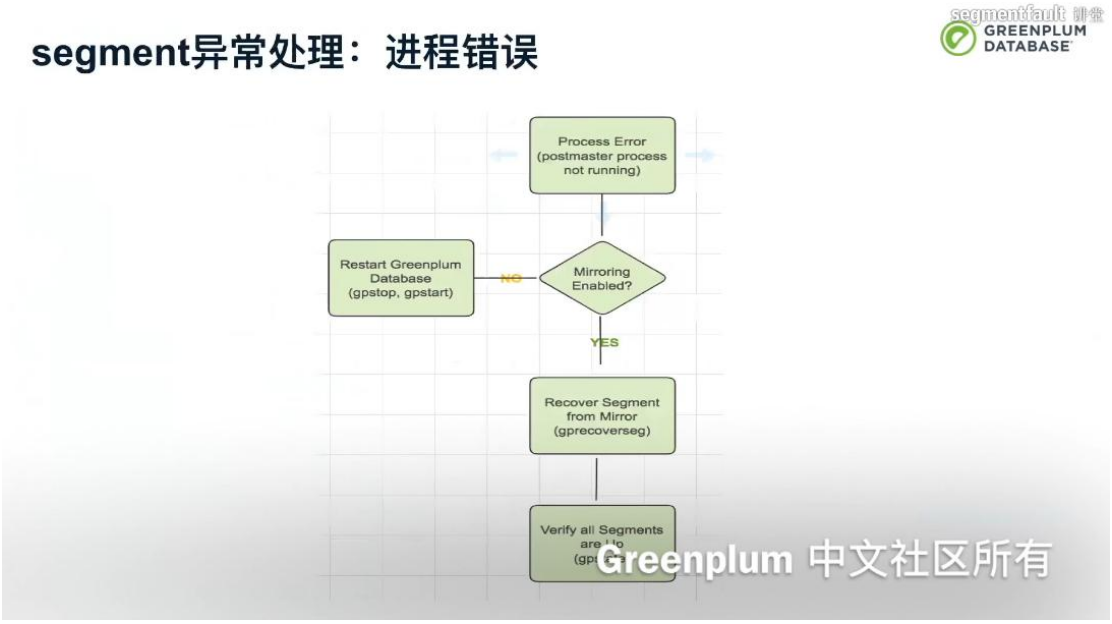
3.2.2 segment 异常处理:主机错误

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。



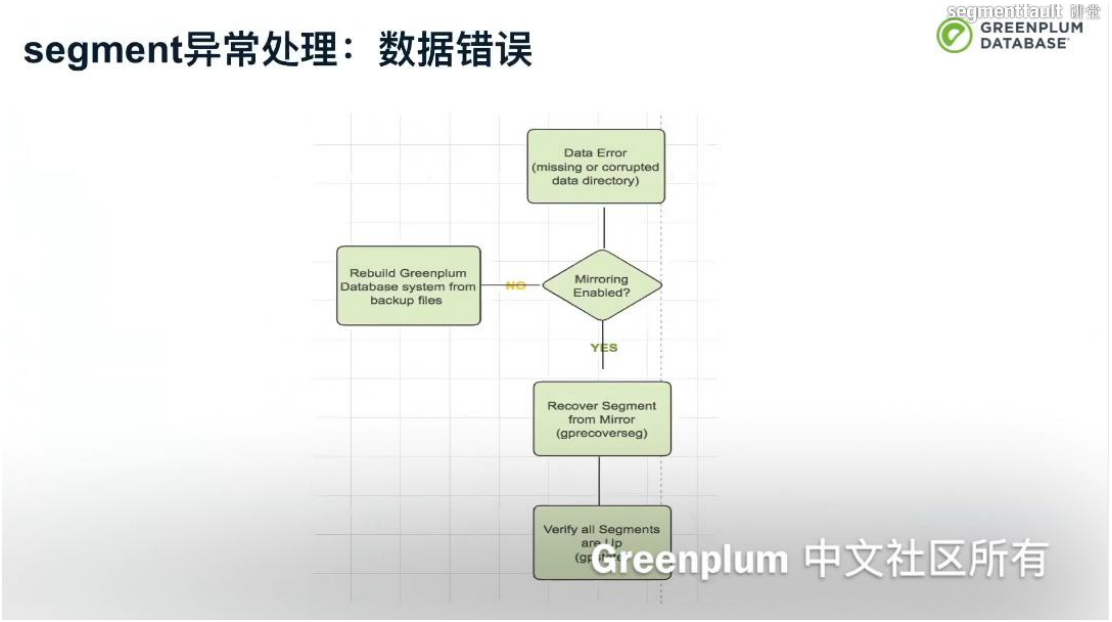
3.2.3 segment 异常处理:进程错误

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。



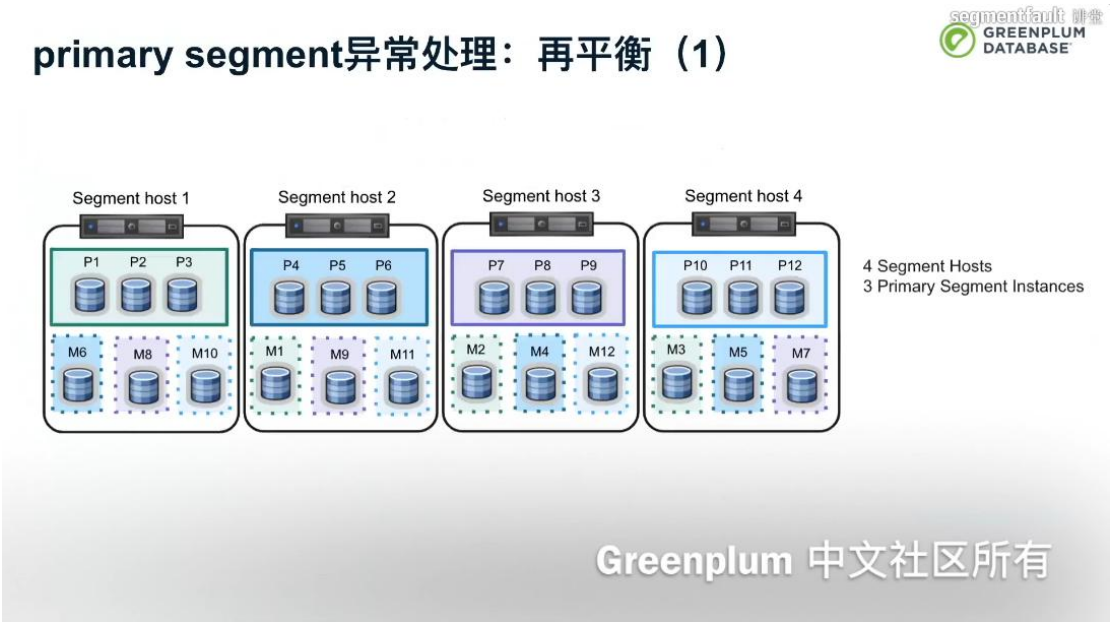
3.2.4 segment 异常处理:数据错误

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。

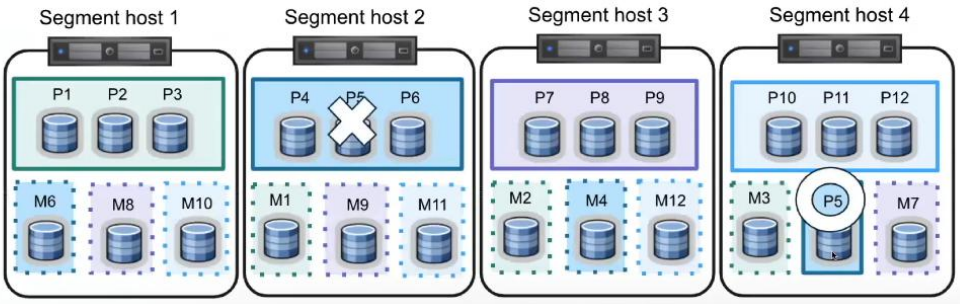


3.2.5 primary segment 异常处理:再平衡

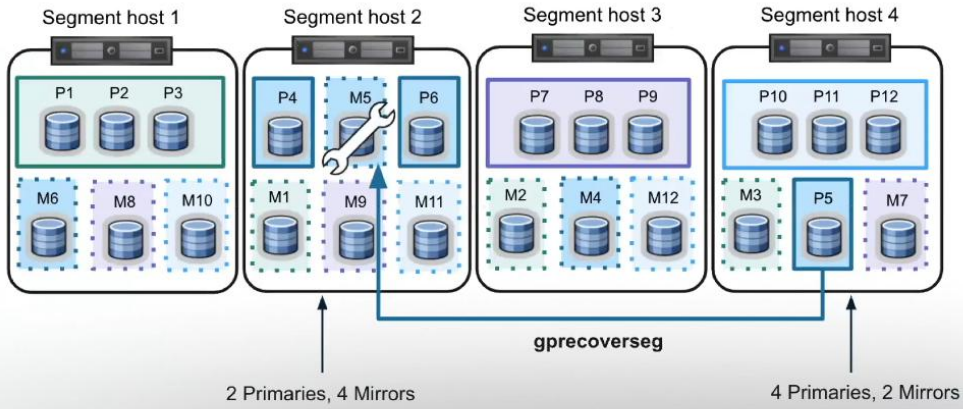
当使用 gprecoverseg 进行恢复完成后，请在使用 gprecoverseg -r 把 primarr 和 mirror 角色互换过来，是集群达到平衡。

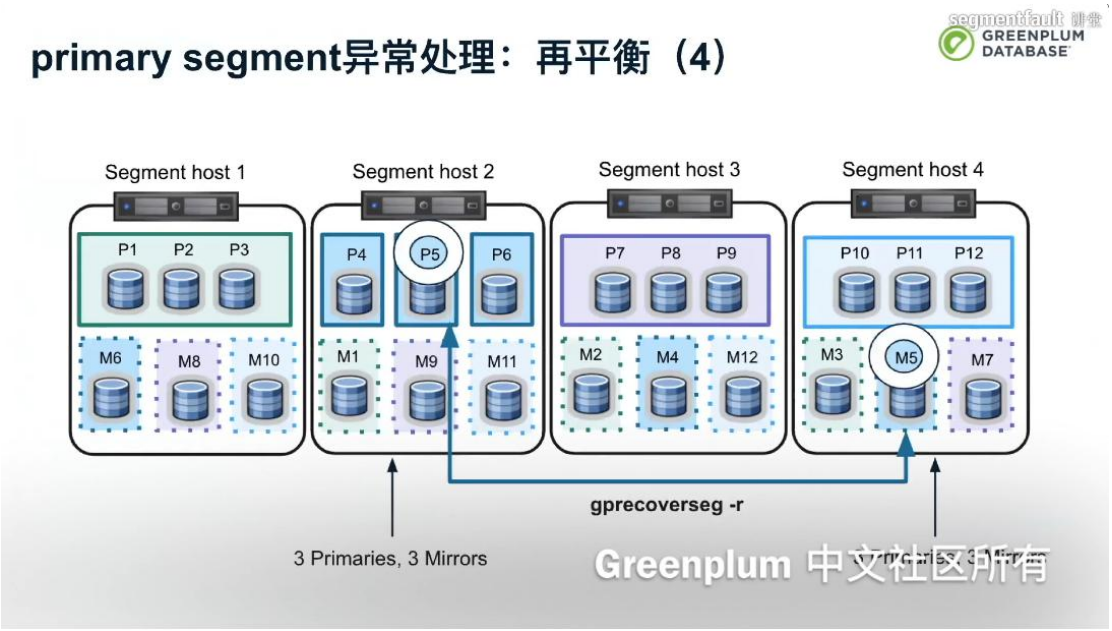


primary segment异常处理：再平衡（2）



primary segment异常处理：再平衡（3）





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。

3.2.7 gprecoverseg 常见错误

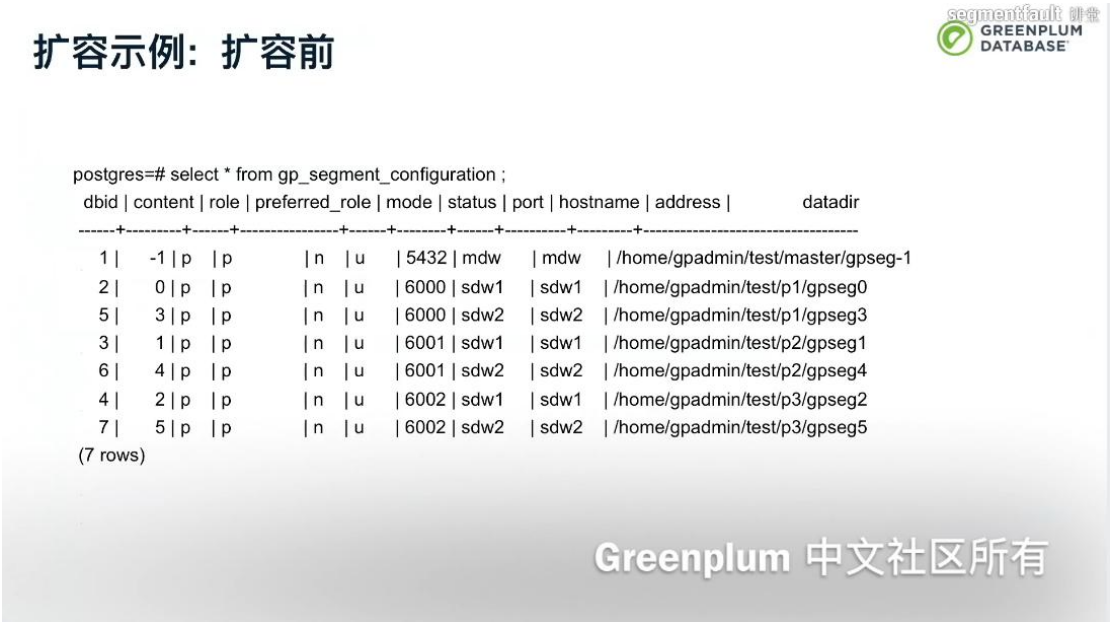
3.3 集群扩容

3.3.1 集群扩容准备

3.3.2 扩容实例

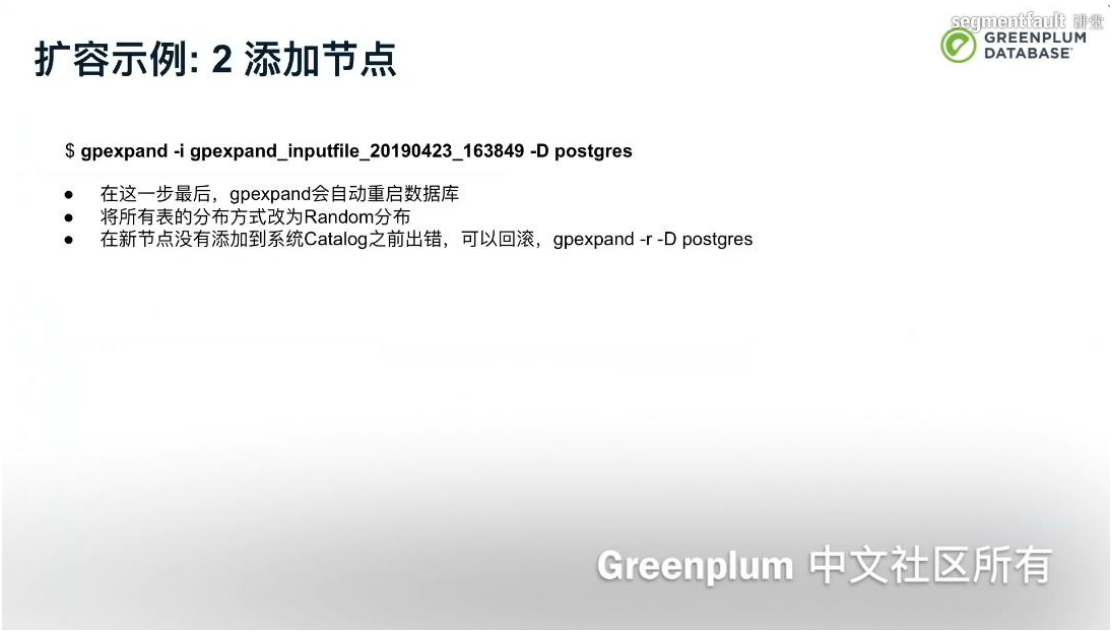
3.3.2.1 扩容之前的状态

注意以下只是测试的环境



3.3.2.2 交互式生成输入文件

3.3.2.3 添加节点



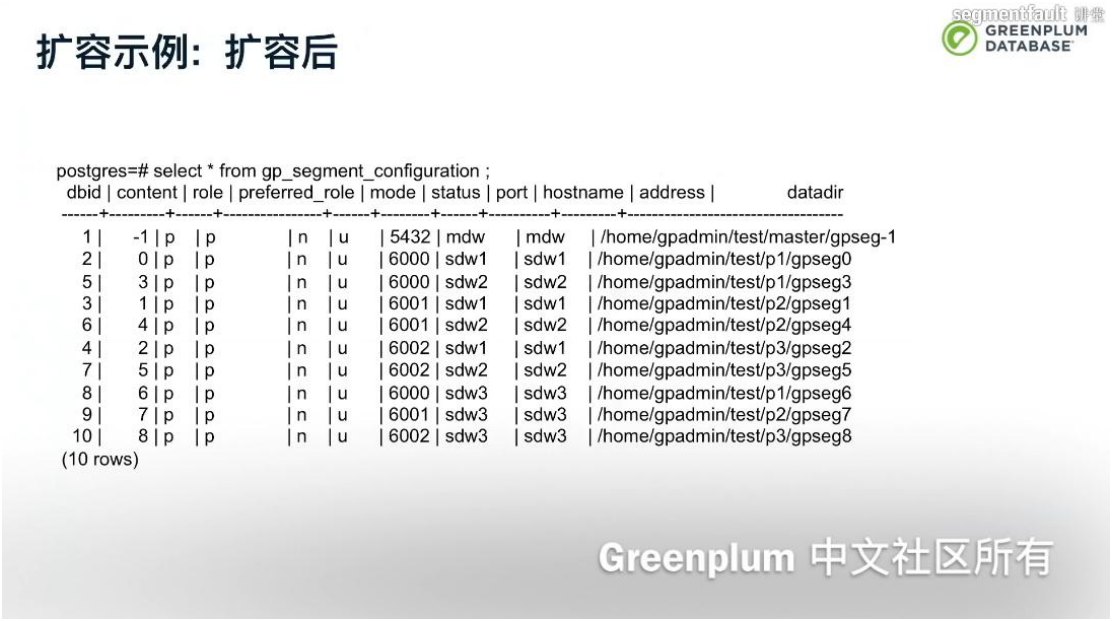
3.3.2.4 数据重分布

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

3.3.2.6 扩容后

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



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

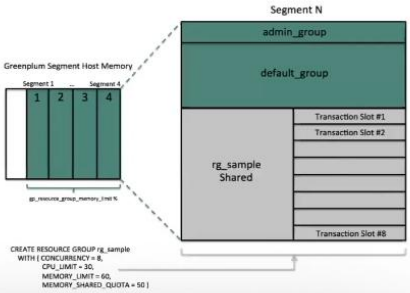
资源管理resource group概述

- 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

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。



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 {
      }
      cpuacct {
      }
      memory {
      }
      cpuset {
      }
}
```


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_QUOUTA=90);
```

- 将resource group和用户关联

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

4.5 查看 resource group 信息

4.6 常见问题和最佳实践

详细的可以查看:http://greenplum.org/calc