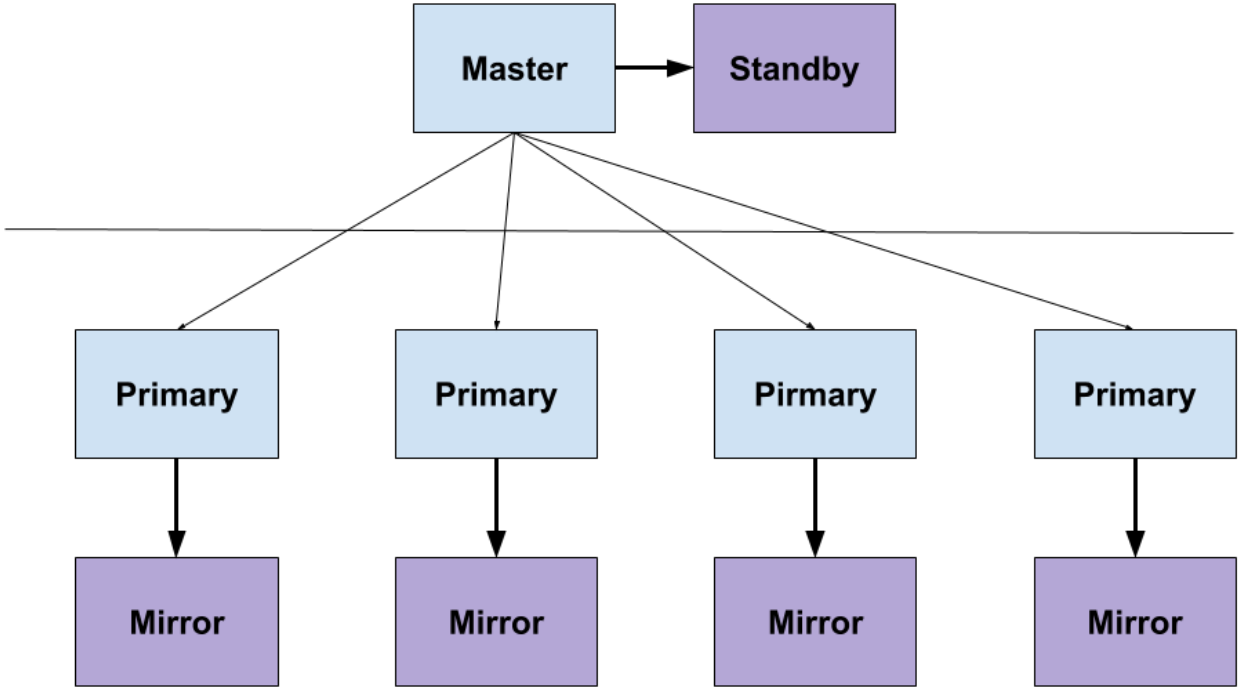
Greenplum 6新特性:
在线扩容工具GPexpand剖析

杜佳伦 (jdu@pivotal.io)

大纲

- Greenplum 集群部署
- GPExpand简介与具体用法
- Greenplum 6中GPExpand的改进与实现

Greenplum 集群部署



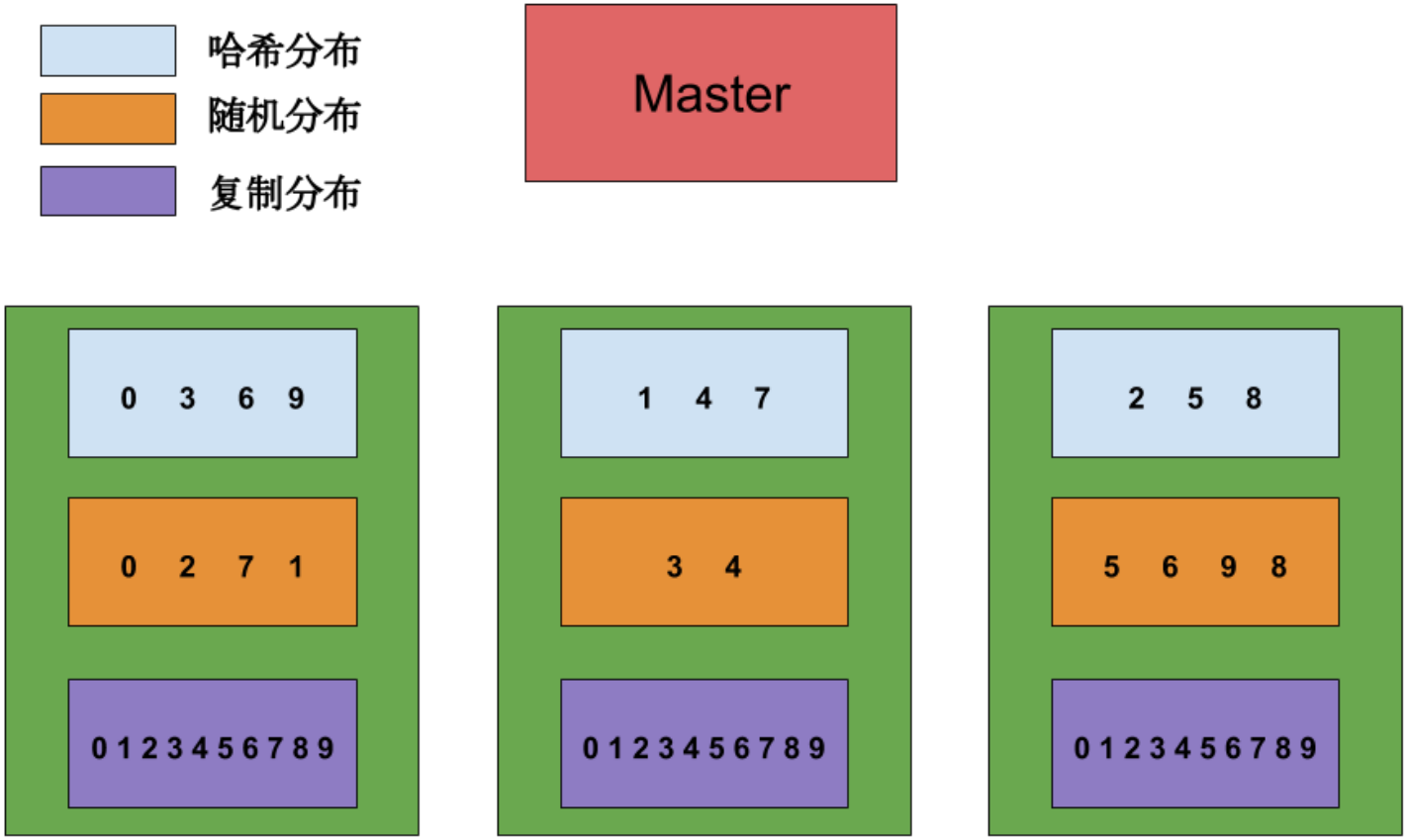
Pivotal

Greenplum 集群部署

- gp_segment_configuration

字段名	描述
dbid	每个节点的唯一id
content	每个pair组的id，master-standby为-1，primary-mirror从0开始递增
role	‘p’ primary，‘m’ mirror
preferred_role	初始化时的值，对于一个被promote成primary的mirror节点，role为’p’，preferred_role为’m’
mode	主从同步状态，’s’同步，’n’不同步
status	运行状态，’u’在线，’d’不在线
port	该节点的运行端口
hostname	节点的hostname
address	通常和hostname相同
datadir	该节点的数据目录

Greenplum 集群部署



Pivotal

Greenplum 集群部署

- gp_distribution_policy

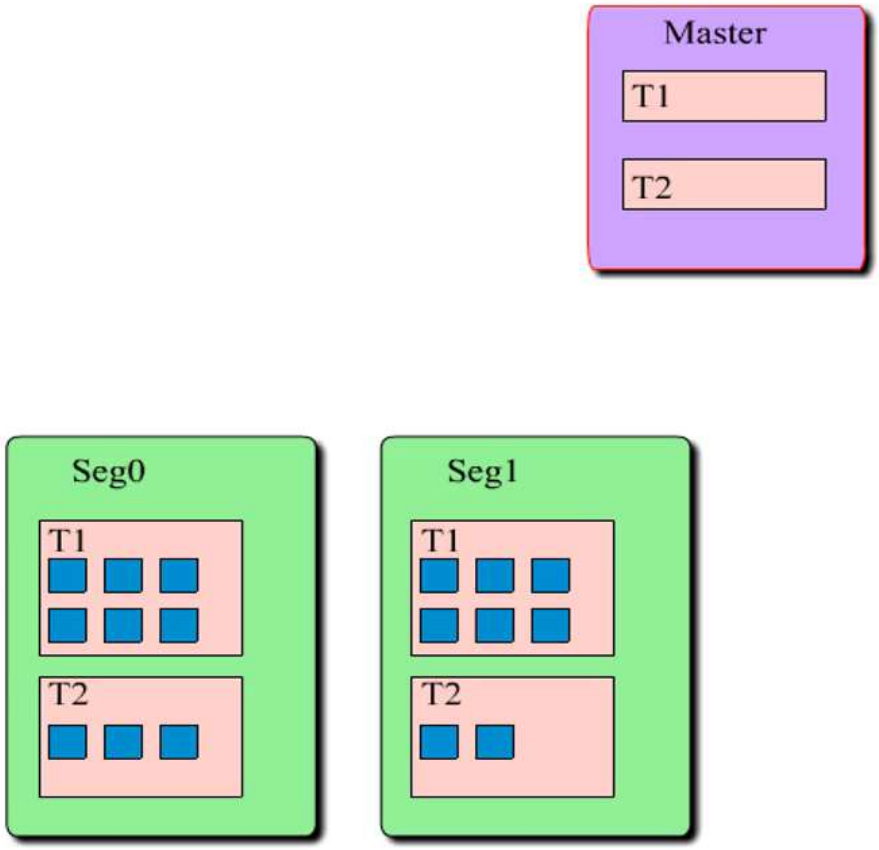
字段名	描述
localoid	表的OID
policytype	分布类型 ‘p’ 分区 ‘r’ 复制表
numsegments	表分布在多少个节点上
diskkey	分布列的序号
distclass	分布列的操作类

GPExpand简介与具体用法

- GPExpand是Greenplum的扩容工具，可以为集群增加新的节点来支持更大容量的存储和更高的计算能力。
- 随着Greenplum一起安装发布，在$GPHOME/bin下面，和其他辅助工具，如gpstart,gpstop,gpactivatestandby一样，是一个用python写的命令行脚本。

GPExpand简介与具体用法

- GPExpand工作流程
  - 建立并添加新节点
  - 数据重分布


Master
T1
T2
Seg0
T1
T2
Seg1
T1
T2

GPExpand简介与具体用法

- 增加新节点
  - gpexpand –i 配置文件
- 数据重分布
  - gpexpand
- 清理
  - gpexpand -c

GPExpand简介与具体用法

- 增加新节点
  - gpexpand –i 配置文件（gpexpand生成或手动编辑）

sdw:sdw:25438:/data/expand1/primary:9:3:p
sdw:sdw:25439:/data/expand1/mirror:10:3:m

GPExpand简介与具体用法

- **新增新**节点
  - postgres下创建gpexpand schen
  - gpexpand schema下面会创建几
    - status
      - —扩容状态
    - status_detail
      - —**将所有需要**扩容的表都存到这个表里
    - expansion_progress
      - —记录扩容时的状态

```
postgres=# select * from gpexpand.status_detail;
-[ RECORD 1 ]-------+------------
dbname              | gpadmin
fq_name             | public.t1
table_oid           | 16385
root_partition_name |
rank                | 2
external_writable   | f
status              | NOT STARTED
expansion_started   |
expansion_finished  |
source_bytes        | 0
-[ RECORD 2 ]-------+------------
dbname              | gpadmin
fq_name             | public.t2
table_oid           | 16388
root_partition_name |
rank                | 2
external_writable   | f
status              | NOT STARTED
expansion_started   |
expansion_finished  |
source_bytes        | 0
```

GPExpand简介与具体用法

- 数据重分布
  - GPExpand
  - 会遍历postgres数据库下面gpe…
  ALTER TABLE {schema.table} E…

```
postgres=# select * from gpexpand.status_detail;
-[ RECORD 1 ]-------+-------------------------
dbname              | gpadmin
fq_name             | public.t2
table_oid           | 16388
root_partition_name |
rank                | 2
external_writable   | f
status              | COMPLETED
expansion_started   | 2019-05-02 10:15:35.713601
expansion_finished  | 2019-05-02 10:15:35.799893
source_bytes        | 0
-[ RECORD 2 ]-------+-------------------------
dbname              | gpadmin
fq_name             | public.t1
table_oid           | 16385
root_partition_name |
rank                | 2
external_writable   | f
status              | COMPLETED
expansion_started   | 2019-05-02 10:15:35.852455
expansion_finished  | 2019-05-02 10:15:35.924996
source_bytes        | 0
```

GPExpand简介与具体用法

- 清理
  - gpexpand –c
  - 会将gpexpand schema和下面关于扩容的表都清理掉

Greenplum 6中GPExpand的改进与实现

- 在线不停机
- 数据重分布优化
- 并行的优化

Pivotal

改进与实现

- **如何做到不停机**
  - **增加新**节点只要在gp_segment_configuration**里添加新**节点信息即可
  - **新**节点以Master为模板生成，只包含catalog，**没有数据**

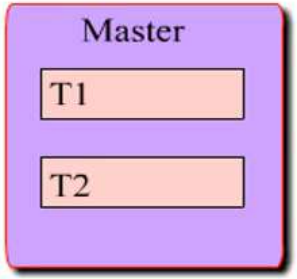
通过Master生成模板

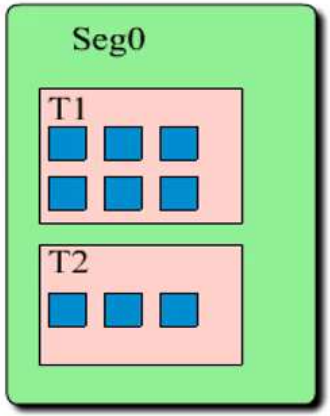
拷贝模板到新节点并启动新节点

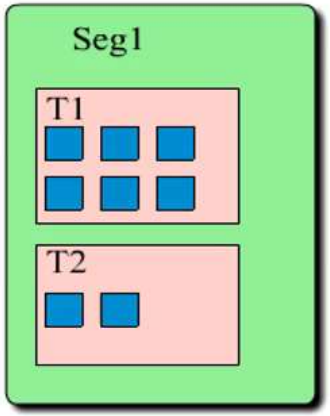
删除模板

将新节点信息加入到catalog表

gp_segment_configuration
Seg0
Seg1







改进与实现

- 问题
  - 生成模板的过程中，如果catalog被修改怎么保证一致性

在线扩容

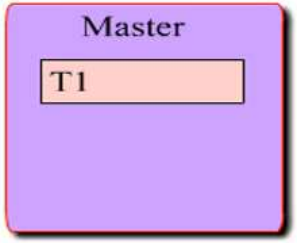
通过master生成模板

拷贝模板到新节点并启动新节点
删除模板
将新节点信息加入到catalog表

正在运行的查询

创建表T2

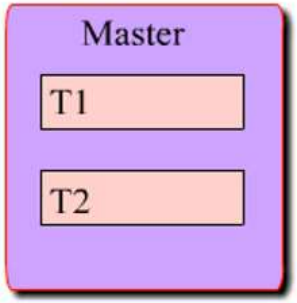
gp_segment_configuration
Seg0
Seg1

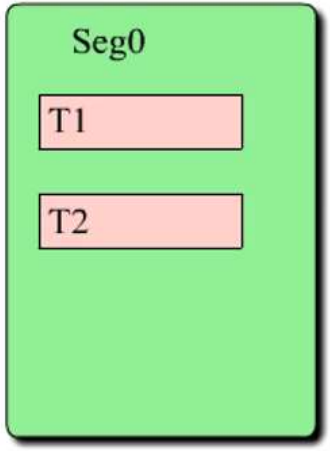


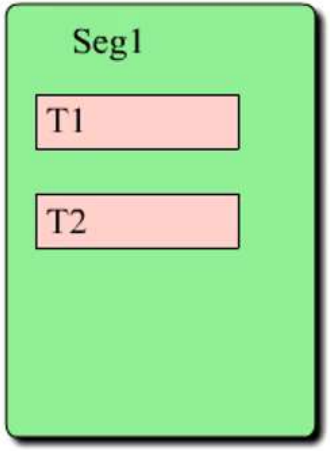
改进与实现

- 问题
  - 生成模板的过程中，如果catalog被修改怎么保证一致性
    - 新增catalog锁
    - select gp_expand_lock_catalog()
    - expand过程中申请写锁
    - 其他修改catalog操作时也会申请锁来实现与expand的互斥

gp_segment_configuration







改进与实现

- 数据重分布的优化
  - 扩容后，新节点没有数据，查询Plan如何做？？？
    - 在Greenplum 5和之前的版本里会将所有的表改成随机分布，然后再ALTER成按列分布
    - Greenplum 6里引入了numsegments
    - Numsegments描述了该表连续分布的segment数量，默认与集群大小一致。对每个表执行操作时也会按照numsegment值分配Gang
    - 增加新节点后，对每个表做EXPAND后该值会随着改成新集群的大小

改进与实现

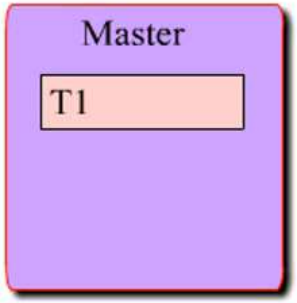
```
gpadmin=# select * from gp_distribution_policy ;
 localoid | policytype | numsegments | distkey | distclass
----------+------------+-------------+---------+-----------
    16385 | p          |           3 | 1       | 10027
    16388 | p          |           3 | 1       | 10027
(2 rows)

gpadmin=# alter table t1 expand table;
ALTER TABLE
gpadmin=# select * from gp_distribution_policy ;
 localoid | policytype | numsegments | distkey | distclass
----------+------------+-------------+---------+-----------
    16388 | p          |           3 | 1       | 10027
    16385 | p          |           4 | 1       | 10027
(2 rows)
```

Pivotal

T1上的查询(未做重分布)

通过catalog创建工作进程

查询仅仅运行在旧节点上

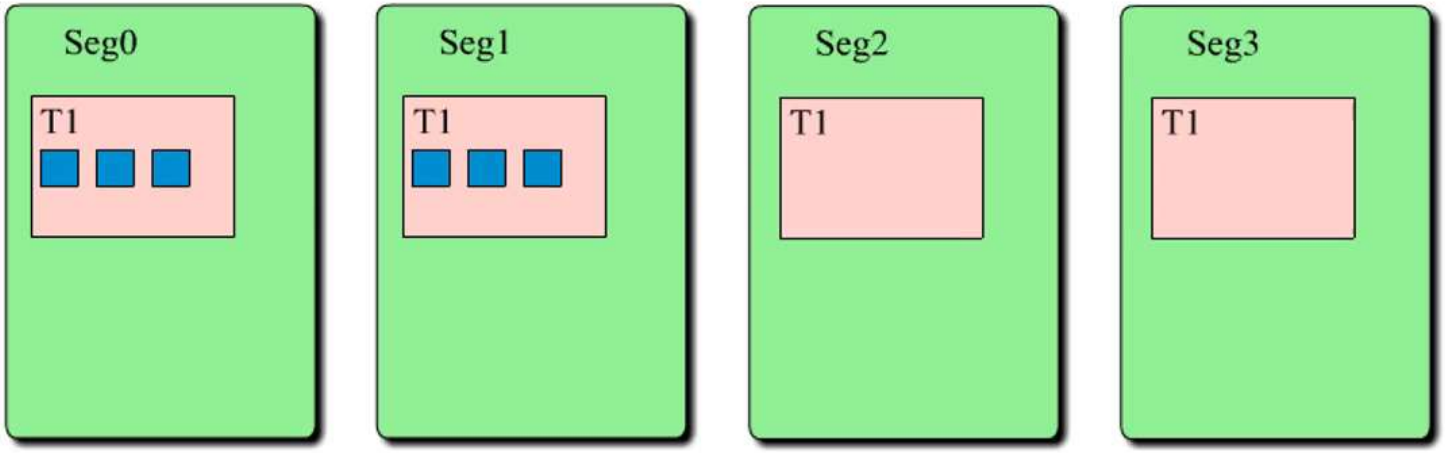


gp_distribution_policy

T1:numsegments(2)

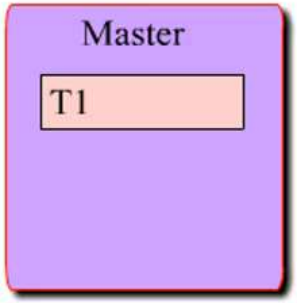
gp_segment_configuration

Seg0
Seg1
Seg2
Seg3



Pivotal

创建新表
根据catalog表创建工作进程
在master上创建新表
下发到工作进程节点上
执行查询
根据catalog表创建工作进程
下发到工作进程节点上

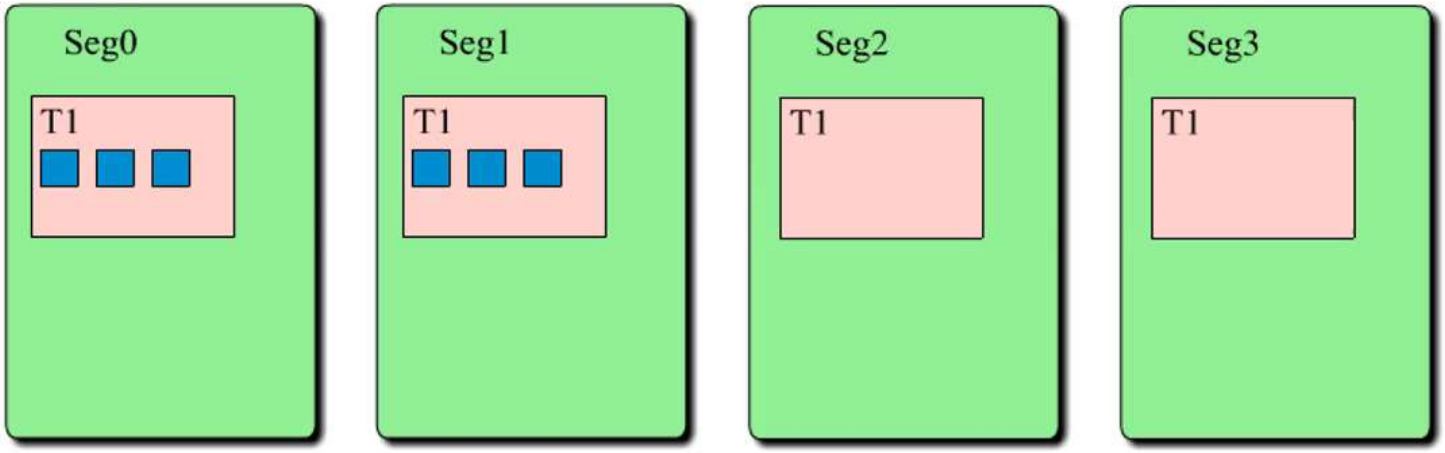


gp_distribution_policy

T1:numsegments(2)

gp_segment_configuration

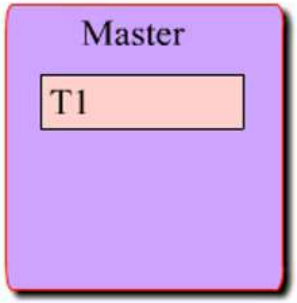
Seg0
Seg1
Seg2
Seg3



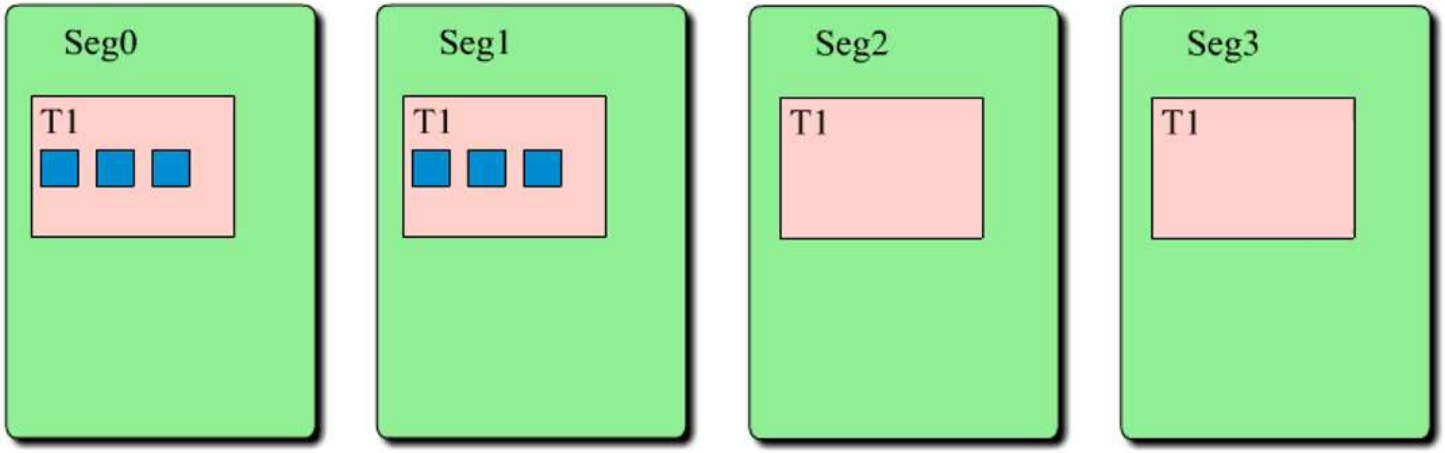
改进与实现

- EXPAND每个表
  - 对表加最高级别锁（其他读写均被阻塞）
  - 移动数据到新节点
  - 修改numsegments
  - 释放锁

数据重分布之前
(数据只分布在旧节点上)
T1操作只运行在Seg0和Seg1上
数据重分布
锁T1 (其他T1查询阻塞)
数据重分布到新节点上
更新numsegments
释放T1锁
数据重分布之后
T1操作运行在所有节点上



gp_distribution_policy
T1:numsegments(2)

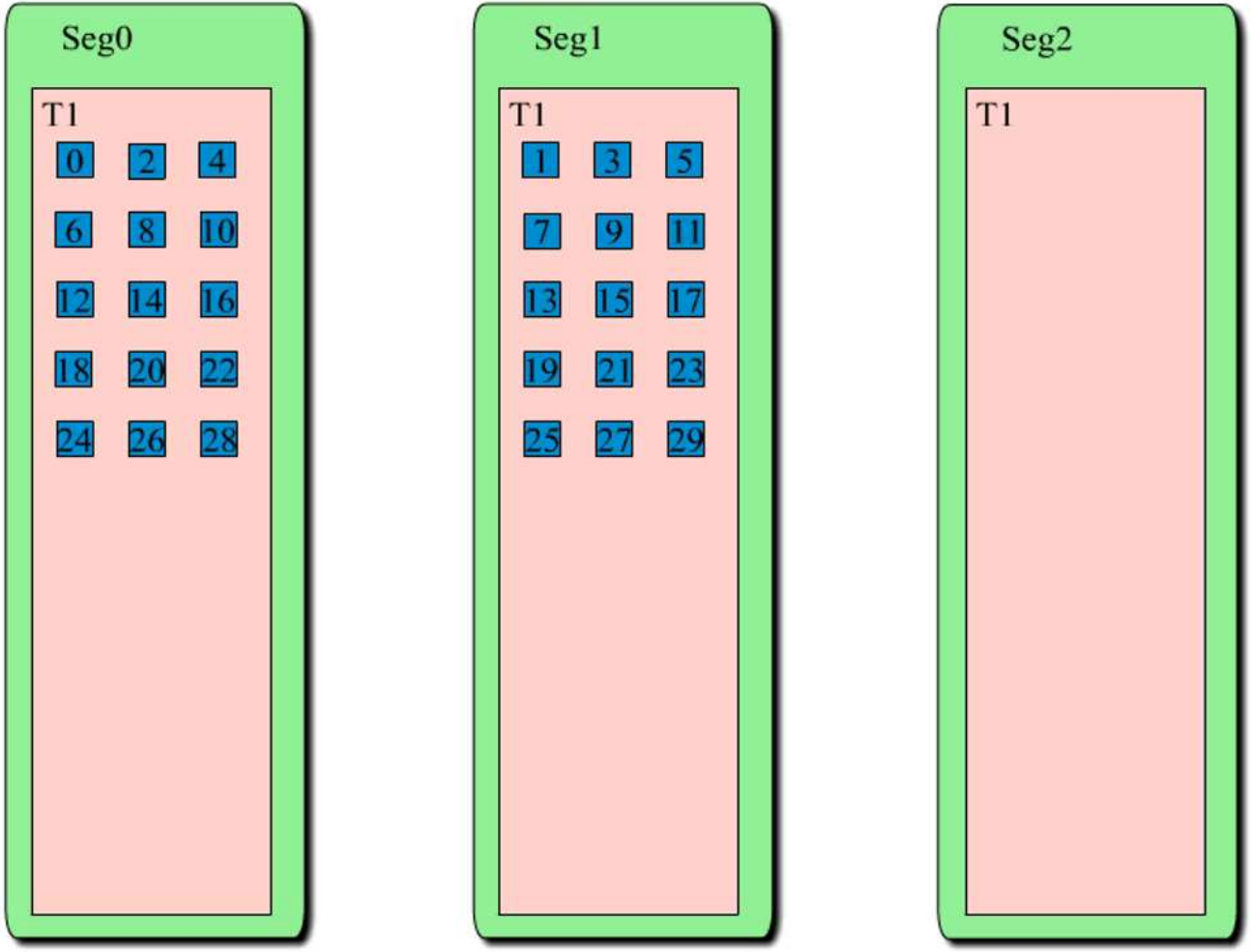


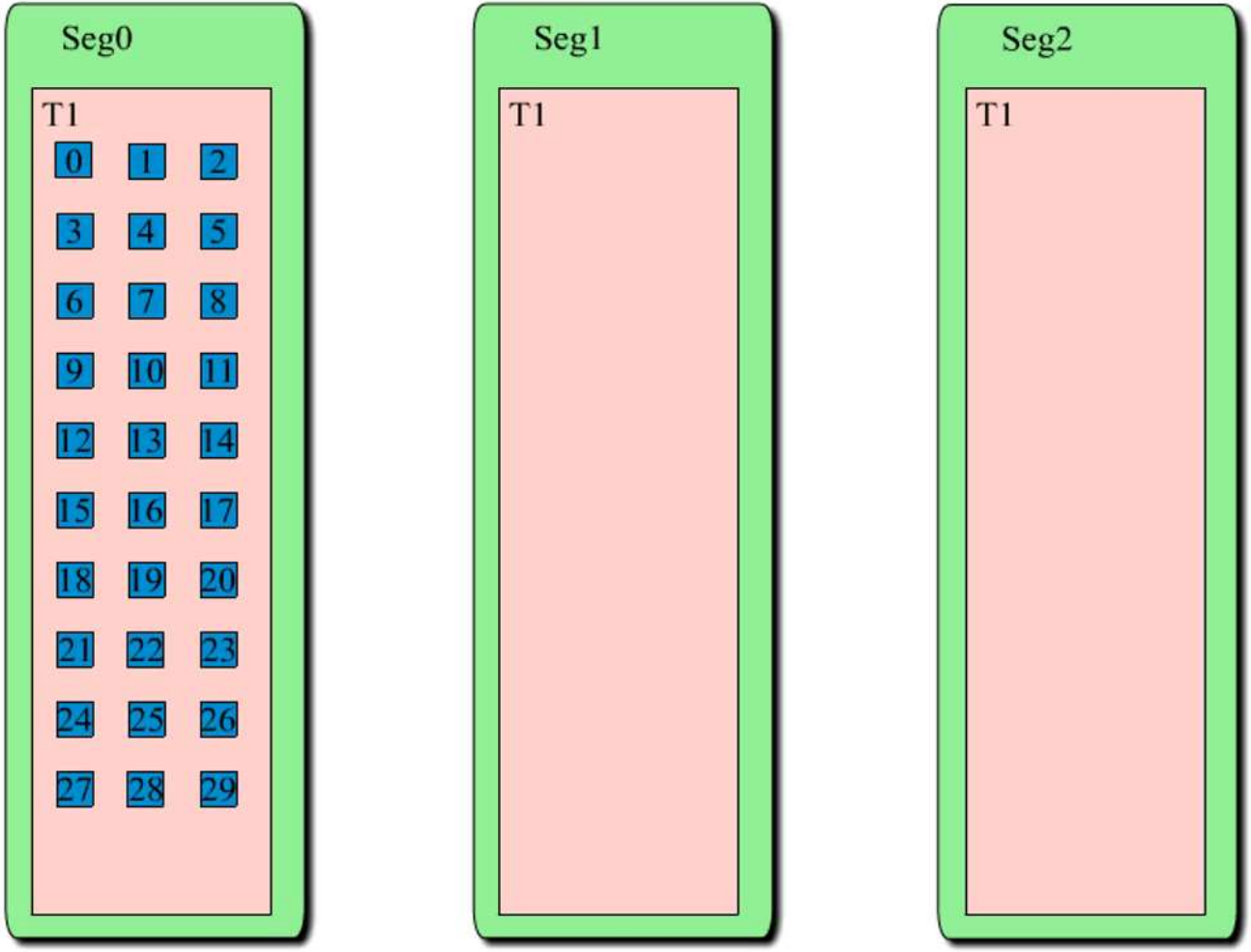
改进与实现

- numsegments的收益
  - **不需要将表改成随机分布**，单表查询可以做优化
  - 对于Join查询，如果分布状态相同的情况下，可以被优化

改进与实现

- **减少重分布数据移**动量
  - Greenplum 5及之前版本采用取模分布
  - 节点数量发生变化后重新计算取模，移动数据量大
  - **不**仅存在新旧节点间的移动，旧节点之间也要移动


Seg0
T1
0 2 4
6 8 10
12 14 16
18 20 22
24 26 28
Seg1
T1
1 3 5
7 9 11
13 15 17
19 21 23
25 27 29
Seg2
T1


Seg0
T1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
Seg1
T1
Seg2
T1

改进与实现

- 减少重分布数据移动量
  - Jump Consistent Hash
    - 均匀性：通过概率做到均匀分布
    - 稳定性：在相同集群大小下，同一个Tuple每次计算结果相同
    - 单调性：扩容过程中，旧节点之间没有数据迁移
    - 高效性：对于集群大小为N的时候，时间复杂度为Log(N)
  - 更多算法细节请参考链接。https://arxiv.org/pdf/1406.2294.pdf
  - 通过GUC gp_use_legacy_hashops 可以控制，默认是Jump Consistent Hash算法

改进与实现

- 并行度控制
  - gpexpand –B
  - 可以并行初始化每个新节点
  - 并行执行expand表
    - 对表执行expand之后要更新gpexpand.status_detail表的状态
    - Greenplum 5及之前的版本对表的更新操作是串行的，所以大量小表做expand会在更新状态表时遇到瓶颈
    - Greenplum 6中因为全局死锁检测的引入可以对heap表做并行更新

改进与实现

- 扩容期间对查询的影响
  - **新增**节点阶段无法修改catalog
  - 对于正在重分布的表的读写访问均会被阻塞
  - 对于分布状态不相同的哈希分布表的Join**无法做**优化

T1,T2重分布之前
分布节点相同，不需要做数据移动

T1重分布,T2未重分布
分布节点变化，需要做数据移动

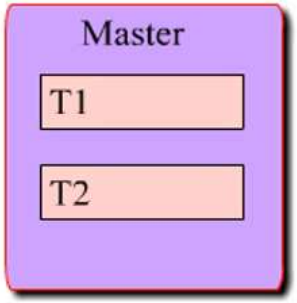
T1,T2重分布之后
分布节点相同，不需要做数据移动

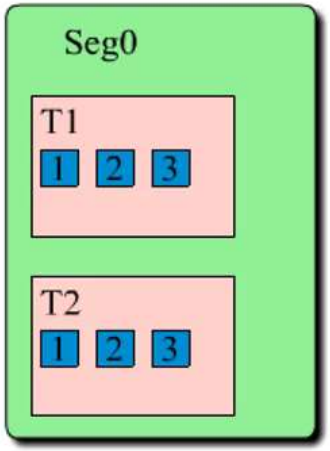
SELECT * FROM T1 JOIN T2
ON T1.C1 = T2.C1

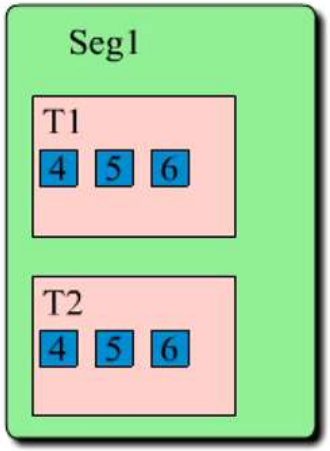
gp_distribution_policy

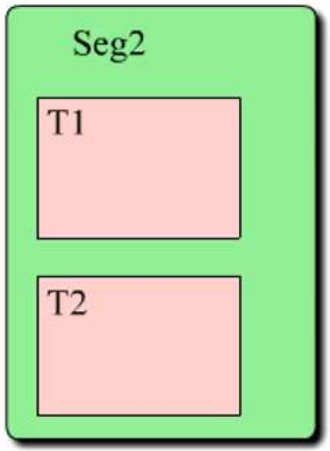
T1:numsegments(2)
T2:numsegments(2)

T1 (C1 INT) 按C1分布
T2 (C1 INT) 按C1分布









Pivotal

Q&A

Thank you

Pivotal