PostgreSQL中文社区

GREENPLUM DATABASE

2021 PostgreSQL China Conference

主办：PostgreSQL 中文社区

第 11 届 PostgreSQL 中国技术大会

开源论道 × 数据驱动 × 共建数字化未来

GREENPLUM DATABASE®

Greenplum中的资源管理策略

王尧, 主任工程师
Greenplum，VMware

2021.11

Index

- Greenplum简介
- 基于角色的资源管理
- Resource Queue
- Resource Group
- 转换策略需要注意的问题
- FAQ
- 参考文档

Greenplum简介

- Greenplum 是全球领先的大数据分析引擎，专为分析、机器学习和AI而打造
  - 线性扩展能力：大规模并行处理架构，PB级存储、处理和实时分析能力
  - 支持多种数据源和数据格式：括 Hadoop、HIVE、HBase、 S3、Gemfire、各种数据库和文件等
  - 多态数据存储：支持堆表，以及面向行存或列存的追加表
  - 集成数据库分析处理数据科学：内建可扩展数据库分析开源库Apache MADlib
  - Greenplum提供的查询优化器是业界优秀的开源的基于代价的查询优化器
- Pivotal公司产品，现在属于VMware公司

基于角色的资源管理

- 基于角色的资源管理
  - CPU
  - 内存
  - 并发数
- 目前支持两种策略
  - Resource Queue
  - Resource Group
- RQ和RG的选择
  - （差异见下页）

Resource Queue和Resource Groups的差异一览

Metric	Resource Queues	Resource Groups
Concurrency	Managed at the query level	Managed at the transaction level
CPU	Specify query priority	Specify percentage of CPU resources; uses Linux Control Groups
Memory	Managed at the queue and operator level; users can over-subscribe	Managed at the transaction level, with enhanced allocation and tracking; users cannot over-subscribe
Memory Isolation	None	Memory is isolated between resource groups and between transactions within the same resource group
Users	Limits are applied only to non-admin users	Limits are applied to SUPERUSER and non-admin users alike
Queueing	Queue only when no slot available	Queue when no slot is available or not enough available memory
Query Failure	Query may fail immediately if not enough memory	Query may fail after reaching transaction fixed memory limit when no shared resource group memory exists and the transaction requests more memory
Limit Bypass	Limits are not enforced for SUPERUSER roles and certain operators and functions	Limits are not enforced on SET, RESET, and SHOW commands
External Components	None	Manage PL/Container CPU and memory resources

Resource Queue: 范例

- CREATE RESOURCE QUEUE myqueue WITH (ACTIVE_STATEMENTS=20, MEMORY_LIMIT='2000MB');
- CREATE ROLE myname WITH LOGIN RESOURCE QUEUE myqueue;
- SET ROLE myname;

Resource Queue: 范例

```
SELECT * FROM gp_toolkit.gp_resqueue_status;
 queueid |     rsqname      | rsqcountlimit | rsqcountvalue | rsqcostlimit | rsqcostvalue |
rsqmemorylimit | rsqmemo
ryvalue | rsqwaiters | rsqholders
---------+-----------------+--------------+--------------+-------------+-------------+---------------+--------
--------+-----------+-----------
  24629 | myqueue         |           20 |            0 |          -1 |           0 |    2.09715e+09 |
    0 |         0 |         0
   6055 | pg_default      |           20 |            0 |          -1 |           0 |             -1 |
    0 |         0 |         0
(2 rows)
```

Resource Queue: 范例

```
testdb=# SELECT * FROM gp_toolkit.gp_locks_on_resqueue WHERE lorwaiting='true';
 lorusename | lorrsqname |  lorlocktype   | lorobjid | lortransaction | lorpid
|    lormode    | lorgranted | lorwai
ting
------------+------------+---------------+---------+---------------+-------+--------------+------------+-------
-----
 wayao2     | myqueue    | resource queue |    24634 |               | 133766 | ExclusiveLock |
t          | t
 wayao2     | myqueue    | resource queue |    24634 |               | 133766 | ExclusiveLock |
f          | t
(2 rows)
```

Resource Queue: 属性

- MEMORY_LIMIT
  - The amount of memory used by all the queries in the queue (per segment). For example, setting MEMORY_LIMIT to 2GB on the ETL queue allows ETL queries to use up to 2GB of memory in each segment.
- ACTIVE_STATEMENTS
  - The number of slots for a queue; the maximum

Resource Queue: 属性

- PRIORITY
  - The relative CPU usage for queries. This may be one of the following levels: LOW, MEDIUM, HIGH, MAX. The default level is MEDIUM. The query prioritization mechanism monitors the CPU usage of all the queries running in the system, and adjusts the CPU usage for each to conform to its priority level. For example, you could set MAX priority to the executive resource queue and MEDIUM to other queues to ensure that executive queries receive a greater share of CPU.
- MAX_COST
  - Query plan cost limit.
  - The Greenplum Database optimizer assigns a numeric cost to each query. If the cost exceeds the MAX_COST value set for the resource queue, the query is rejected as too expensive.

Resource Group: 简介

- 当RG的资源管理限制没有达到上限，并且当前query没有导致所属的group超过当前事务的限制时，Greenplum会立即执行query。否则的话，Greenplum会将query放入执行队列。
- 你可以使用RG管理外部组件（例如PL/Container ）的CPU和内存资源
- RG使用Linux control groups (cgroups)管理CPU资源
- RG使用**vmtracker/cgroups**管理内存

Resource Group: 范例

- CREATE RESOURCE GROUP rgroup1 WITH (CPU_RATE_LIMIT=20, MEMORY_LIMIT=25, MEMORY_SPILL_RATIO=20);
- CREATE ROLE myrole RESOURCE GROUP rgroup1;
- SELECT * FROM gp_toolkit.gp_resgroup_config;
- SELECT * FROM gp_toolkit.gp_resgroup_status;
- SELECT * FROM gp_toolkit.gp_resgroup_status_per_segment;
- SELECT query_start, waiting, rsgname, rsgqueueduration FROM pg_stat_activity;

Resource Group: 范例

```
SELECT * FROM gp_toolkit.gp_resgroup_config;
 groupid |   groupname   | concurrency | proposed_concurrency | cpu_rate_limit | memory_limit | proposed_memory_limi
t | memory_shared_quota | proposed_memory_shared_quota | memory_spill_ratio | proposed_memory_spill_ratio
---------+---------------+-------------+----------------------+----------------+--------------+----------------------
--+---------------------+------------------------------+--------------------+------------------------------
   6437 | default_group | 20          | 20                   | 30             | 30           | 30
  | 50                  | 50                           | 20                 | 20
   6438 | admin_group   | 10          | 10                   | 10             | 10           | 10
  | 50                  | 50                           | 20                 | 20
  24639 | rgroup1       | 2           | 2                    | 20             | 25           | 25
  | 20                  | 20                           | 20                 | 20
(3 rows)
```

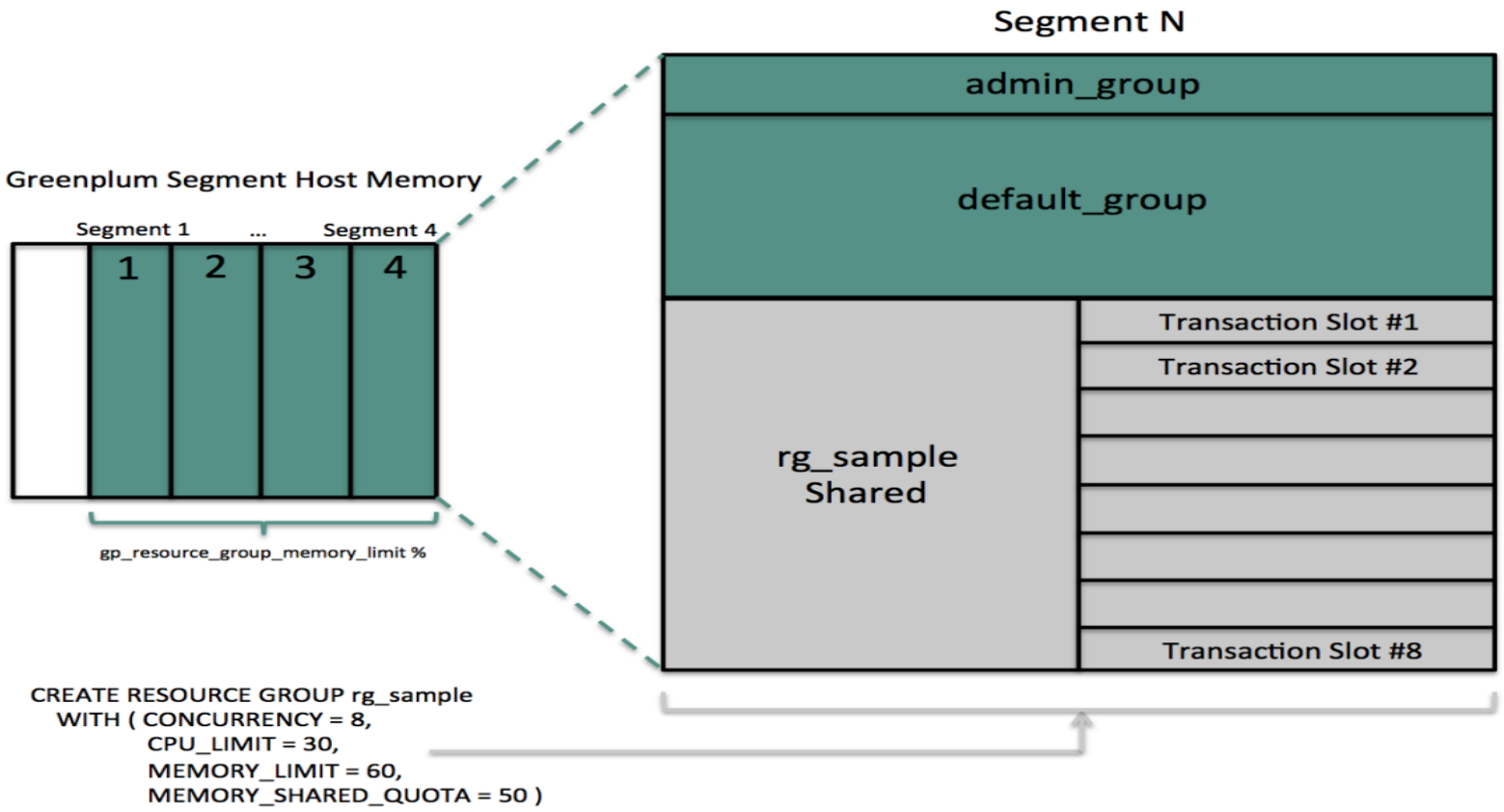
Resource Group: 属性

Limit Type	Description
MEMORY_AUDITOR	The memory auditor in use for the resource group. vmtracker (the default) is required if you want to assign the resource group to roles. Specify cgroup to assign the resource group to an external component.
CONCURRENCY	The maximum number of concurrent transactions, including active and idle transactions, that are permitted in the resource group.
CPU_RATE_LIMIT	The percentage of CPU resources available to this resource group.
CPUSET	The CPU cores to reserve for this resource group.
MEMORY_LIMIT	The percentage of reserved memory resources available to this resource group.
MEMORY_SHARED_QUOTA	The percentage of reserved memory to share across transactions submitted in this resource group.
MEMORY_SPILL_RATIO	The memory usage threshold for memory-intensive transactions. When a transaction reaches this threshold, it spills to disk.

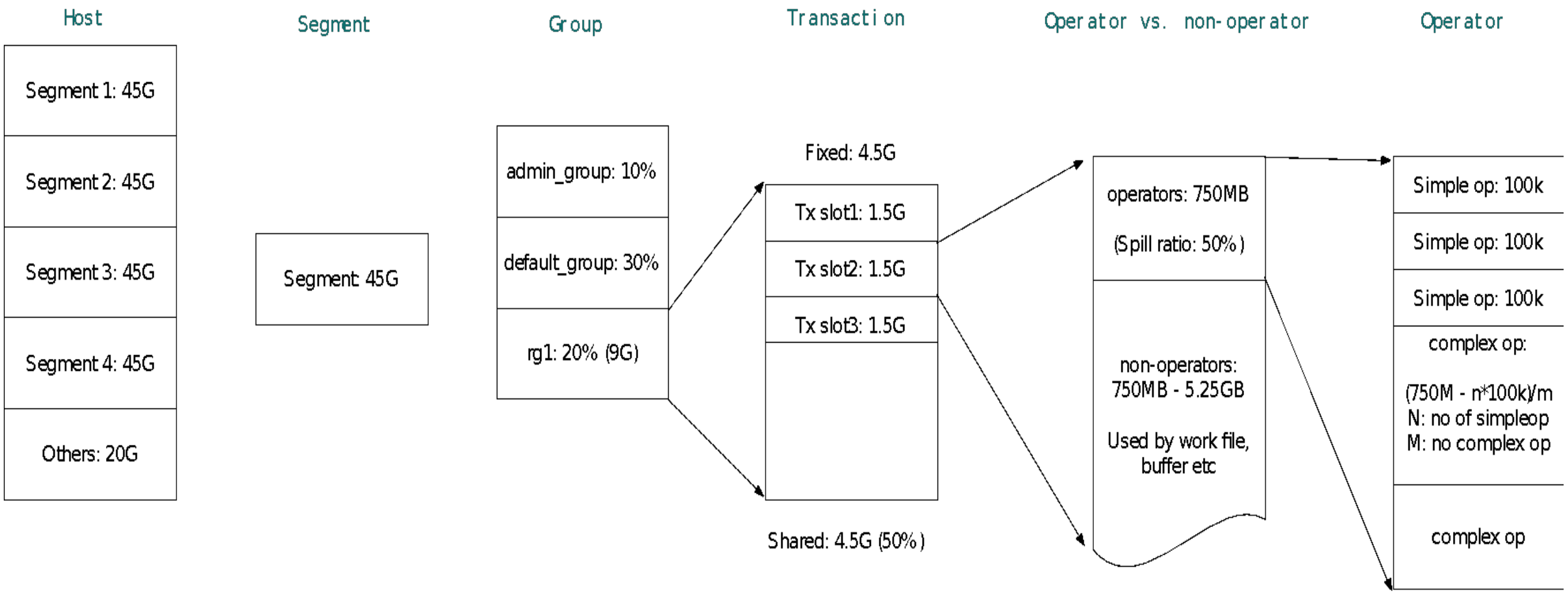
Resource Group: 其他设置

- gp_resource_group_memory_limit
  - The percentage of system memory to allocate to Greenplum Database. The default value is .7 (70%).
- gp_workfile_limit_files_per_query
  - Set gp_workfile_limit_files_per_query to limit the maximum number of temporary spill files (workfiles) allowed per query. Spill files are created when a query requires more memory than it is allocated. When the limit is exceeded the query is terminated. The default is zero, which allows an unlimited number of spill files and may fill up the file system.
- gp_workfile_compression
  - If there are numerous spill files then set gp_workfile_compression to compress the spill files. Compressing spill files may help to avoid overloading the disk subsystem with IO operations.
- memory_spill_ratio
  - Set memory_spill_ratio to increase or decrease the amount of query operator memory Greenplum Database allots to a query. When memory_spill_ratio is larger than 0, it represents the percentage of resource group memory to allot to query operators. If concurrency is high, this memory amount may be small even when memory_spill_ratio is set to the max value of 100. When you set memory_spill_ratio to 0, Greenplum Database uses the statement_mem setting to determine the initial amount of query operator memory to allot.

Resource Group:内存模型



Resource Group:内存模型



Resource Group: 内存管理

For example, a segment node has 30G physical memory and 8G SWAP space with overcommit ratio 50, then the system total memory will be:

```
total_memory = total_RAM * overcommit_ratio/100 + total_Swap
= 30*0.5 + 8 = 23G
```

With default GUC gp_resource_group_memory_limit 0.9 and suppose that there are 4 segments on this node, the total memory for a segment will be:

```
total_segment_memory = total_memory *
gp_resource_group_memory_limit/segment_count = 23*0.9/4 =
5.2G
```

Resource Group:内存管理

If your resource group rg1 has the following definition:

```
create resource group rg1 with (concurrency=5,
cpu_rate_limit=20, memory_limit=30, memory_shared_quota=20,
memory_spill_ratio=30);
```

resource group rg1 will have the following memory allocation on each segment:

```
resource_group_memory_per_segment =
total_segment_memory*memory_limit/100 = 5.2*30/100 = 1.56G
```

Resource Group:内存管理

- For every transaction in this group, memory limit will include two parts:
  - **Fixed memory quota:** Fixed memory quota will be allocated to this transaction dedicatedly, guaranteed and cannot be used by any other transactions. This part will be
    - `transaction_fixed_memory_per_segment = resource_group_memory_per_segment * (100-memory_shared_quota)/(100*concurrency) = 1.56 * (100 - 20)/(100*5) ~= 250M`
  - **Shared memory quota:** Shared memory quota will be shared by all running transactions in this group, and first come first serve if available. This part of memory cannot be guaranteed to specific transactions in this group.
    - `transaction_shared_memory_per_segment = resource_group_memory_per_segment * memory_shared_quota)/100 = 1.56 * 20/ 100 ~= 300M`

Resource Group:内存管理

- So in theory, the maximum memory for a transaction on one segment will be:
  - `transaction_memory_per_segment = transaction_fixed_memory_per_segment + transaction_shared_memory_per_segment = 250+300 = 550M.`
- For any new transaction, GPDB allocates some initial memory quota according to memory_spill_ratio:
  - `transaction_initial_quota_per_segment = (resource_group_memory_per_segment * MEMORY_SPILL_RATIO)/(100*concurrency) = (1.56 * 30)/(100*5) ~= 94M`

转换策略需要注意的问题

- statement vs transaction
- CPU PRIORITY vs CPU percentage
- Memory limit override vs cannot
- MAX_COST在RQ中不可用
- 使用自动化工具来实现转换?

FAQ

- gp_vmem_protect_limit在RG的环境下无效
- memory_spill_ratio 在设置为非0值时，statement_mem和max_statement_mem 无效
- 大多数GPDB内存配置是用于避免操作系统级别的内存不足错误（OOM）导致的数据库实例崩溃，而不是避免OOM本身。单个的查询仍然可能由于超于内存限制而出现OOM
- 对于操作系统可见的任何CPU都被计入到CPU core的数量中。例如，虚拟出的CPU core也被RG视为CPU core。
- 为什么我们允许query的操作符在初始内存以外申请更多的内存?
  - memory_spill_ratio用于定义一个内存百分比，超出这个内存使用量后query可以开始溢出到磁盘。但即使query开始溢出到磁盘后，query仍然可能使用更多的内存，因为我们无法简单地把所有的内存需求都转换到磁盘上。所以在这个时候，仍然可能要申请更多的内存。

参考文档

- Managing Resources
  - https://gpdb.docs.pivotal.io/6-18/admin_guide/wlmgmt.html
- Memory and Resource Management with Resource Queues
  - https://gpdb.docs.pivotal.io/6-18/best_practices/workloads.html
- Memory and Resource Management with Resource Groups
  - https://gpdb.docs.pivotal.io/6-18/best_practices/resgroups.html
- https://github.com/greenplum-db/gpdb/wiki/Resource-Group-(Experimental)

感谢观看

Greenplum中文社区公众号

Greenplum微信技术讨论群

GREENPLUM DATABASE®

THANKS

谢谢观看