GREENPLUM DATABASE 腾讯云大学

六节课

快速上手Greenplum

第四节课

Greenplum快速调优

线上直播 | 10月31日 14：00 - 15：00

GREENPLUM DATABASE®

微信技术讨论群
微信搜索添加“gp_assistant”
加入技术讨论

微信公众号
搜索添加“Greenplum中文社区”
技术干货、行业热点、活动预告

GREENPLUM DATABASE 腾讯云大学

Greenplum 中文社区 公众号

- 技术研讨会PPT
- 活动预告
- 行业热点
- 技术干货
- Greenplum动态

GREENPLUM DATABASE 腾讯云大学

Greenplum中文社区网站

https://cn.greenplum.org

博客 · 资料 · 文档 · 项目 · 活动

GREENPLUM DATABASE 腾讯云大学

全新的技术问答论坛

有问题？askGP！

https://cn.greenplum.org/askgp

原厂专家值班 · 优质内容沉淀

GREENPLUM DATABASE 腾讯云大学

六节课快速上手Greenplum

第四课

Greenplum 快速调优

南网数据中心大数据平台DBA：叶健锋

目录

GREENPLUM DATABASE 腾讯云大学

第一节 集群规划中影响性能的因素

第二节 数据库性能优化内容

第三节 日常维护对性能的提升

第四节 SQL优化技巧

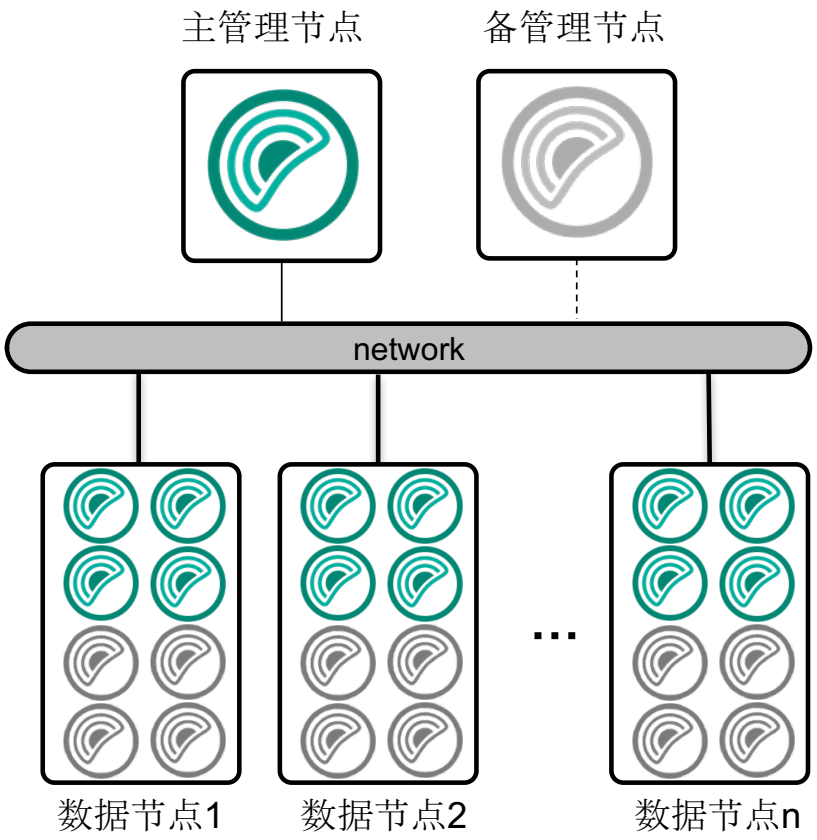
第五节 常见性能问题

集群规划中影响性能的因素

GREENPLUM DATABASE

腾讯云大学

架构设计



影响性能因素

- 并行处理的木桶效应
  - 节点服务器配置
  - 实例处理数据量不均衡
- 镜像分布策略
  - GROUP
  - SPREAD

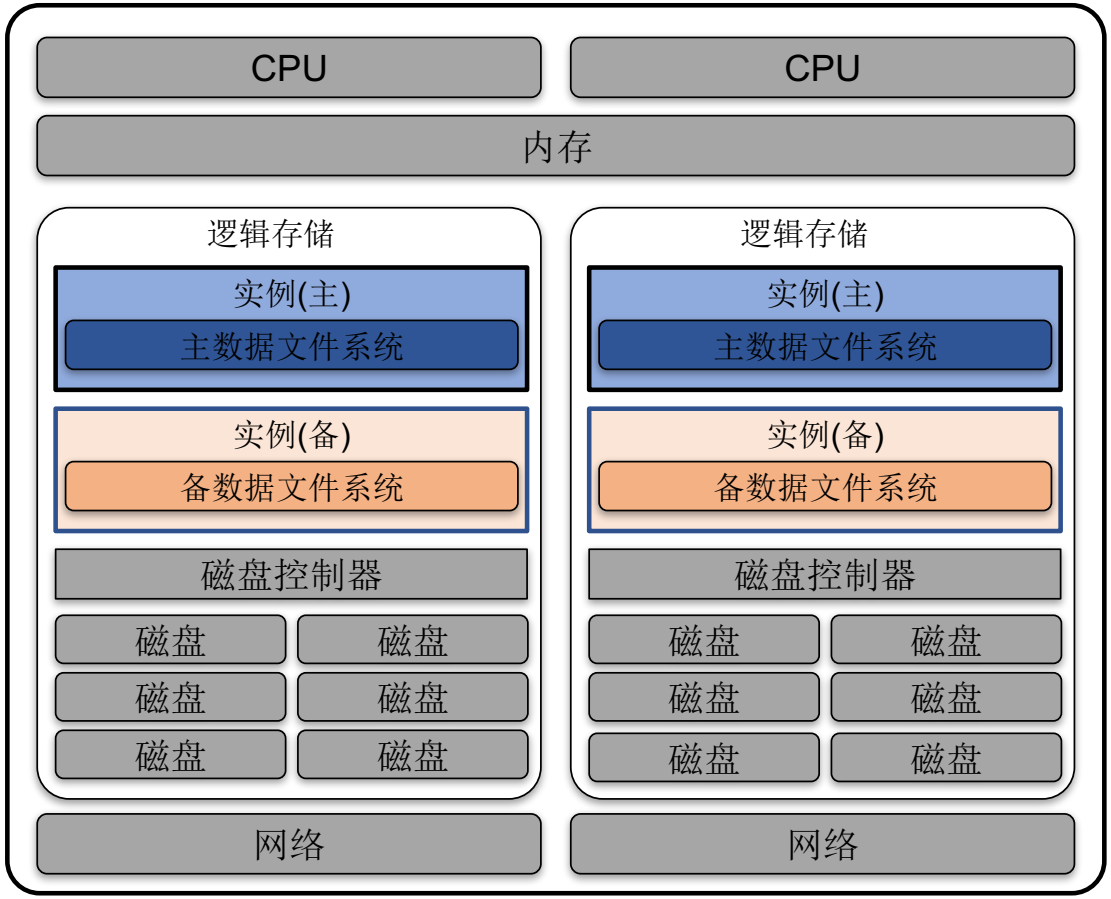
服务器配置

关注点

- CPU开启超线程
- 磁盘IO性能
- 万兆网络

性能测试

- gpcheckperf

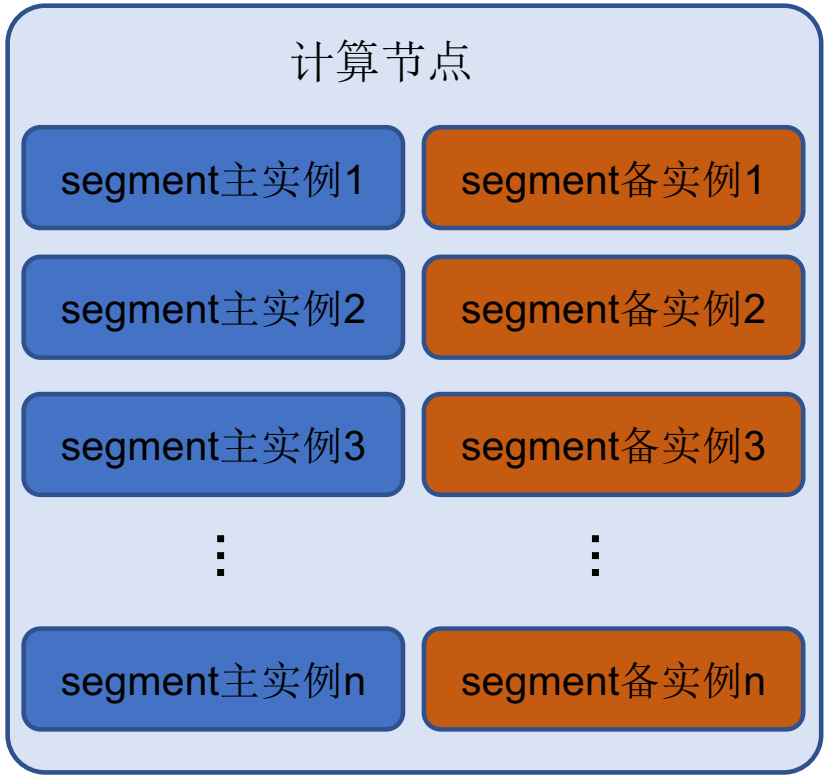


Segment实例数量

GREENPLUM DATABASE 腾讯云大学

考虑因素

- CPU核数
- 物理内存
- 网络速度
- 主备实例同时工作
- 服务器有运行其它进程
- 预期的并发数



数据库优化内容

GREENPLUM DATABASE

腾讯云大学

内存管理

GREENPLUM DATABASE 腾讯云大学

操作系统参数设置

vm.overcommit_memory = 2

内核不允许分配超过所有物理内存和交换空间总和的内存

vm.overcommit_ratio = 95

为进程分配内存的百分比,默认50，vm.overcommit_memory=2的情况下才生效

数据库参数设置

gp_vmem_protect_limit

限制每个节点所有语句使用内存总的上限

计算公式:

(SWAP + (RAM * vm.overcommit_ratio)) * 0.9 / number_Segments_per_server

资源队列的内存管理

GREENPLUM DATABASE 腾讯云大学

gp_resqueue_memory_policy

auto:内存的消耗由参数statement_mem和资源队列的memory_limit限制

eager_free:内存消耗由参数max_statement_mem和资源队列的memory_limit限制

max_statement_mem

限制每个查询最大使用的内存，默认2000MB

(seghost_physical_memory) / (average_number_concurrent_queries)

statement_mem

节点数据库上单个查询可以使用的内存总量,默认125MB，如果需要更多内存完成操作，则会溢出到磁盘

(gp_vmem_protect_limit * 0.9) / max_expected_concurrent_queries

资源队列

- 限制并发的查询数
  - ACTIVE_STATEMENTS
- 限制查询使用的内存总量
  - MEMORY_LIMIT
- 控制查询的优先级
  - PRIORITY
- 根据查询成本代价做限制
  - MIN_COST
  - MAX_COST
  - COST_OVERCOMMIT

```
postgres=# select * from pg_resqueue_attributes
postgres-# where rsqname = 'pg_default';
  rsqname   |      resname       | ressetting | restypid
------------+--------------------+------------+----------
 pg_default | active_statements  | 80         |        1
 pg_default | max_cost           | -1         |        2
 pg_default | min_cost           | 0          |        3
 pg_default | cost_overcommit    | 0          |        4
 pg_default | priority           | medium     |        5
 pg_default | memory_limit       | -1         |        6
(6 rows)
```

资源组

GREENPLUM DATABASE 腾讯云大学

内存管理模式	• 基于角色(vmtracker) • 基于外部组件(cgroup)
CPU配额	• 按百分比分配 • 按CORE分配
内存配额	• 按百分比分配
并发事务限制	• 基于角色的管理模式才有效

资源组和资源队列的区别

GREENPLUM DATABASE 腾讯云大学

参数	资源队列	资源组
并行	在查询级别管理	在事务级别管理
CPU	指定队列顺序	指定CPU的使用百分比；使用Linux控制组
内存	在队列和操作级别管理；用户可以过量使用	在事务级别管理，可以进一步分配和追踪；用户不可以过量使用。
内存隔离	无	同资源组下的事务使用的内存是隔离的，不同资源组使用的内存也是隔离的。
用户	仅非管理员用户有限制。	非管理员用户和超级用户都有限制
排序	当没有可用槽位时，才开始排序	当槽位或内存不足时，开始排序
查询失效	当内存不足时，查询可能会立即失效	在没有更多的共享资源组内存的情况下，若事务到达了内存使用量限制后仍然提出增加内存的申请，查询可能会失效
避开限制	超级用户角色以及特定的操作者和功能不受限制。	SET、RESET和SHOW指令不受限制
外部组件	无	管理PL/Container CPU和内存资源

表存储

GREENPLUM DATABASE 腾讯云大学

堆(HEAP)存储

Postgresql的堆存储，所有操作都会产生REDO记录，写到事务日志文件，并由它来控制刷盘时间，适合OLTP型业务。

追加优化(AO)存储

追加存储，删除更新数据时，通过BITMAP文件来标记被删除的行，事务结束时，需要调用FSYNC刷盘。

表存储

GREENPLUM DATABASE 腾讯云大学

行存储

- 一行为一个元组的形式，所有列都存到一个文件上
- 读取任意列的成本不一样，越靠后的列成本越高

列存储

- 一列存一个单独的文件
- 读取任意列的成本一样
- 压缩比高于行存储
- 访问的列越多，开销越大

存储模式及对比

GREENPLUM DATABASE 腾讯云大学

数据库中表存储的模式

<table>
<tr><td>HEAP表</td><td>行存</td><td>不可压缩</td></tr>
<tr><td rowspan="2">AO表
(appendonly=true)</td><td>行存
(orientation=row)</td><td rowspan="2">可压缩
(compresstype=zlib,
COMPRESSLEVEL=5)</td></tr>
<tr><td>列存
(orientation=column)</td></tr>
</table>

存储大小对比

类型	文件	堆存储	AO表行存	AO表列存	AO表行存压缩	AO表列存压缩
大小	35GB	32GB	34GB	30GB	13GB	6822MB

建压缩表例子

```
create table tmp
with (appendonly=true,
      orientation=row,
      compresstype=zlib,
      COMPRESSLEVEL=5)
as
select * from pg_tables
distributed randomly;
```

数据加载

- 加载方式的性能

  GPFDIST > COPY > 批量INSERT >单条INSERT

- **GPFDIST**优化的参数

参数名	说明
writable_external_table_bufsize	控制主实例向文件服务器发送数据包的大小，默认64kb
gp_external_max_segs	控制访问文件服务器的实例数量，默认64

数据加载

- gpfdist导出控制参数: **writable_external_table_bufsize**

文件大小(MB)	导出耗时(s)	速度(MB/s)	参数值(kb)
45441	201	226.07	512
45441	56	811.45	16384

- gpfdist加载控制参数: gp_external_max_segs

文件大小(MB)	导入耗时(s)	速度(MB/s)	参数值
45441	108	420.75	20
45441	59	770.19	40

集群规模

- 服务器数量:20
- 主备实例数:160
- 网络速率:万兆

其它优化点

GREENPLUM DATABASE 腾讯云大学

- **表设计**
  - 尽量设计仅允许插入的表
- **表分布键**
  - 选择唯一性比较高的单个字段做分布键
- **表分区**
  - 对大的表进行分区，优化不同的分区存储模式
- **索引**
  - 注意索引的膨胀维护

日常维护对性能的提升

GREENPLUM DATABASE

腾讯云大学

统计信息

GREENPLUM DATABASE 腾讯云大学

统计信息作用

- 估算表大小及数据膨胀情况
- 帮助优化器生成最优执行计划

使用analyze收集表统计信息

- 运行命令:analyze tablename;
- 相关参数:default_statistics_target，默认25
- 选择列来收集统计信息

收集统计信息

GREENPLUM DATABASE 腾讯云大学

什么时候运行analyze收集统计信息?

- 加载数据后
- 创建索引后
- 大数据量的DML操作后

统计信息收集的参数设置

gp_autostats_mode	on_no_stats	如果表没有统计数据，则CREATE TABLE, INSERT, COPY 操作会触发统计信息的收集
	on_change	当更新的行数超过gp_autostats_on_change_threshold定义的阈值时才触发统计信息收集，其默认值是2147483647
	none	禁用自动统计信息收集功能

数据膨胀

GREENPLUM DATABASE 腾讯云大学

膨胀原因

Greenplum数据库使用多版本并发控制（MVCC）的存储机制，删除和更新的行仅是逻辑删除，其实际数据仍然存储在表中，只是不可见。

膨胀造成的影响

- 占用磁盘存储空间
- 查询表时扫描更多的文件块，浪费IO资源

检测膨胀

GREENPLUM DATABASE 腾讯云大学

HEAP表

```
test=# select *  from gp_toolkit.gp_bloat_expected_pages
test-# where btdrelid=lower('public.tmp_heap')::regclass ;
 btdrelid | btdrelpages | btdexppages
----------+-------------+-------------
    81384 |         637 |           2
(1 row)
```

AO表

```
test=# select * from gp_toolkit.__gp_aovisimap_compaction_info('public.tmp_ao'::regclass) ;
NOTICE:  gp_appendonly_compaction_threshold = 10
 content | datafile | compaction_possible | hidden_tupcount | total_tupcount | percent_hidden
---------+----------+---------------------+-----------------+----------------+----------------
       1 |        0 | t                   |           50111 |          50111 |         100.00
       1 |        1 | f                   |               0 |          50115 |           0.00
       0 |        0 | t                   |           49889 |          49889 |         100.00
       0 |        1 | f                   |               0 |          49885 |           0.00
(4 rows)
```

膨胀处理

GREENPLUM DATABASE 腾讯云大学

HEAP表

- vacuum
  - vacuum回收不了超过空闲映射空间的过期行
- vacuum不能回收时，用以下方法:
  - vacuum full

    调优参数maintenance_work_mem，默认64MB
  - 强制重分布

    alter table $TB set WITH (REORGANIZE=TRUE) $DK

AO表

- vacuum
  - 相关参数:gp_appendonly_compaction_threshold,默认10

数据倾斜

- 数据倾斜是很多性能问题和内存溢出问题的根本原因
- 集群的数据倾斜

```
gpssh –f allhosts –e ‘df|grep $datadir’
```

- 查看某一个表的倾斜

```
select gp_segment_id,pg_relation_size(oid) size
from gp_dist_random('pg_class')
where oid = 'public.demo'::regclass
order by size desc;
 gp_segment_id  |  size
--------------+---------
             0          | 4423680
             1          |       0
```

计算倾斜

GREENPLUM DATABASE 腾讯云大学

- 计算倾斜在表关联、排序、聚合等操作中容易出现
- 有计算倾斜，但是没有溢出临时文件，则不会影响性能
- 控制溢出文件的参数

gp_workfile_limit_files_per_query

SQL查询分配的内存不足，数据库会创建溢出文件。默认值100000，0表示无限制

gp_workfile_compress_algorithm

设置溢出的临时文件是否压缩

计算倾斜排查过程

GREENPLUM DATABASE 腾讯云大学

```
#gpssh -f allsegs -e 'du -sh /primary/gpseg*/base/17146/pgsql_tmp'|sort -k 2|grep -v du
…
[seg10] 881G /primary/pg_system/gpseg2/base/17146/pgsql_tmp
[seg10] 16M /primary/pg_system/gpseg3/base/17146/pgsql_tmp

#ls –lhSr /primary/pg_system/gpseg2/base/17146/pgsql_tmp
881G pgsql_tmp_slice10_sort_15673_0002.1

#lsof /data2/primary/gpseg45/base/17146/pgsql_tmp/pgsql_tmp_slice10_sort_15673_0002.1
postgres 15673 gpadmin …

#ps –ef|grep 15673
gpadmin … postgres: port 40003, … con2132569 seg2 cmd32 slice10 MPPEXEC SELECT
```

datid oid	datname name	procpid integer	sess_id integer	usesysid oid	usename name	current_query text	w bc
17146	csgbi	60018	2132569	10	gpadmin	SELECT * FROM	f

系统表优化

GREENPLUM DATABASE 腾讯云大学

- **系统表**
  - 系统表也是heap表，随着对象的创建、修改、删除会造成数据膨胀
  - 系统表索引同时也会膨胀，且不能被vacuum回收空间

- **优化内容**
  - 每天定时对系统表进行vacuum
  - 定期监控系统表的索引膨胀情况及reindex
  - 避免元数据数量过多

作业流程优化

- 避免作业拥堵

  记录pg_stat_activity和pg_locks的快照，查到历史时间里存在锁的作业，根据实际情况从业务逻辑上优化

```
time=`date '+%Y-%m-%d %H:%M:%S'`

psql -c "
insert into pg_stat_activity_log
select '$time',* from pg_stat_activity where current_query <> '<IDLE>';
insert into pg_locks_log
select '${time}',*
from pg_locks
"
```

SQL优化

GREENPLUM DATABASE

腾讯云大学

从执行计划中优化

- 看懂执行计划
- 两个重要概念
  - 重分布
  - 广播

```
test=# explain select * from tmp1 a join tmp2 b on b.c=a.c;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Gather Motion 80:1  (slice3; segments: 80)  (cost=1.04..3.15 rows=4 width=104)
   ->  Hash Join  (cost=1.04..3.15 rows=1 width=104)
         Hash Cond: b.c = a.c::text
         ->  Redistribute Motion 80:80  (slice1; segments: 80)  (cost=0.00..2.06 rows=1 width=8)
               Hash Key: b.c
               ->  Seq Scan on tmp2 b  (cost=0.00..2.02 rows=1 width=8)
         ->  Hash  (cost=1.03..1.03 rows=1 width=96)
               ->  Redistribute Motion 80:80  (slice2; segments: 80)  (cost=0.00..1.03 rows=1 width=96)
                     Hash Key: a.c::text
                     ->  Seq Scan on tmp1 a  (cost=0.00..1.01 rows=1 width=96)
 Optimizer status: legacy query optimizer
```

union与union all

- 使用union时会去重，去重会发生重分布，而union all不会去重

模似数据

```
create table t01 as select * from pg_tables
distributed by (tablename);
create table t02 as select * from pg_tables
distributed by (tablename);
create table t03 as select * from pg_tables
distributed by (tablename);
```

使用union all时数据不会重分布

```
test=# explain
test-# insert into t03
test-# select * from t01
test-# union all
test-# select * from t02;
                            QUERY PLAN
-----------------------------------------------------------------
 Insert on t03  (cost=0.00..8.30 rows=108 width=259)
   ->  Append  (cost=0.00..6.15 rows=108 width=259)
         ->  Seq Scan on t01  (cost=0.00..3.07 rows=54 width=259)
         ->  Seq Scan on t02  (cost=0.00..3.08 rows=54 width=259)
 Optimizer: Postgres query optimizer
(5 rows)
```

Union的优化

GREENPLUM DATABASE 腾讯云大学

使用union时数据会发生重分布

```
test=# explain
test-# insert into t03
test-# select * from t01
test-# union
test-# select * from t02;

-----------------------------------------------------------------
 Insert on t03  (cost=12.06..16.36 rows=108 width=259)
   ->  Redistribute Motion 2:2  (slice2; segments: 2)  (cost=12.0
         Hash Key: t01.tablename
         ->  HashAggregate  (cost=12.06..14.21 rows=108 width=259
               Group Key: t01.schemaname, t01.tablename, t01.tabl
               ->  Redistribute Motion 2:2  (slice1; segments: 2)
                     Hash Key: t01.schemaname, t01.tablename, t01
                     ->  Append  (cost=0.00..8.30 rows=108 width=
                           ->  Seq Scan on t01  (cost=0.00..3.07
                           ->  Seq Scan on t02  (cost=0.00..3.08
 Optimizer: Postgres query optimizer
(11 rows)
```

分开插入可避免数据重分布

```
test=# explain
test-# insert into t03
test-# select * from t01 a;
                          QUERY PLAN
-----------------------------------------------------------------
 Insert on t03  (cost=0.00..3.07 rows=54 width=259)
   ->  Seq Scan on t01 a  (cost=0.00..3.07 rows=54 width=259)
 Optimizer: Postgres query optimizer
(3 rows)

Time: 1.246 ms
test=# explain
test-# insert into t03
test-# select * from t02 a
test-# where  not exists(select 1 from t01 b
test(#                   where a.tablename=b.tablename);
                                 QUERY PLAN
-----------------------------------------------------------------
 Insert on t03  (cost=4.41..8.09 rows=4 width=259)
   ->  Hash Anti Join  (cost=4.41..8.09 rows=4 width=259)
         Hash Cond: (a.tablename = b.tablename)
         ->  Seq Scan on t02 a  (cost=0.00..3.08 rows=54 width=259)
         ->  Hash  (cost=3.07..3.07 rows=54 width=64)
               ->  Seq Scan on t01 b  (cost=0.00..3.07 rows=54 width=64)
 Optimizer: Postgres query optimizer
(7 rows)
```

分布键优化

- 多表关联时，尽量使用分布键作为关联条件

```
create table test1(id int,message text )
distributed by(id);
create table test2(id int,city text )
distributed by(id,city);
```

```
test=# explain
test-# select *
test-# from test1 a
test-# join test2 b on a.id=b.id ;
                                                QUERY PLAN
-----------------------------------------------------------------
 Gather Motion 2:1  (slice2; segments: 2)  (cost=1216.00..335049.6
   ->  Hash Join  (cost=1216.00..335049.60 rows=1230080 width=72)
         Hash Cond: (b.id = a.id)
         ->  Redistribute Motion 2:2  (slice1; segments: 2)  (cost
               Hash Key: b.id
               ->  Seq Scan on test2 b  (cost=0.00..596.00 rows=24
         ->  Hash  (cost=596.00..596.00 rows=24800 width=36)
               ->  Seq Scan on test1 a  (cost=0.00..596.00 rows=24
 Optimizer: Postgres query optimizer
(9 rows)

test=#
```

```
create table test1(id int,message text )
distributed by(id);
create table test2(id int,city text )
distributed by(id);
```

```
test=# explain
test-# select *
test-# from test1 a
test-# join test2 b on a.id=b.id ;
                                                QUERY PLAN
-----------------------------------------------------------
 Gather Motion 2:1  (slice1; segments: 2)  (cost=1
   ->  Hash Join  (cost=1216.00..334057.60 rows=123
         Hash Cond: (a.id = b.id)
         ->  Seq Scan on test1 a  (cost=0.00..596.0
         ->  Hash  (cost=596.00..596.00 rows=24800
               ->  Seq Scan on test2 b  (cost=0.00
 Optimizer: Postgres query optimizer
(7 rows)
```

一些SQL优化内容

GREENPLUM DATABASE 腾讯云大学

- 避免出现笛卡尔积
- 避免出现计算倾斜
- 尽量避免向客户端返回大数据量
- 在子查询中尽可能滤掉多的行
- 避免不必要的排序

常见性能问题

GREENPLUM DATABASE

腾讯云大学

用户查询慢

GREENPLUM DATABASE 腾讯云大学

- 注意加limit限制
- 进程是否被锁
- SQL是否可优化
- 使用的表数据是否有倾斜
- 表关联中是否有计算倾斜
- 数据库资源是否繁忙

数据库运行慢

GREENPLUM DATABASE 腾讯云大学

问题案例：

集群正常使用时，突然性能慢，用户体验很卡，正常的简单查询耗时长

原因分析：

- 内存不足，使用了swap交换空间
- CPU负载高
- 磁盘IO繁忙

快速定位：

根据节点服务器占用系统资源最大的进程，回溯查到数据库进程

两段事务锁

GREENPLUM DATABASE 腾讯云大学

问题案例：

- 执行删除表命令时，一直在执行中，连接数据库查询并没有锁。
- 久之不处理后，wal文件堆积

```
drop table test; --pid 7511
select procpid,sess_id,waiting from pg_stat_activity where procpid=7511;
 procpid | sess_id | waiting
---------+---------+---------
   7511 | 1741873 | f
```

原因分析：

Segment节点有残留的事务

处理过程

GREENPLUM DATABASE 腾讯云大学

1、根据sess_id查询各segment的状态，发现seg01上的seg0在等待状态

```
gpssh -f allsegs -e 'ps -ef|grep 1741873'

[seg01] ... con1741873 seg0 cmd13 MPPEXEC UTILITY waiting
```

2、使用PGOPTIONS方式连接到seg01的seg0节点,根据当前sess_id查看segment节点的pid

```
PGOPTIONS="-c gp_session_role=utility" psql -h seg01 -d dbname gpadmin -p port

# select procpid from pg_stat_activity where sess_id=1741873
```

3、根据pid找锁源并清除

残留的在线事务

select pg_terminate_backend(pid);

残留的预备事务

```
select relation=62542114 from pg_locks where pid=7511;

select transaction  from pg_locks where relation=62542114

select gid from pg_prepared_xacts where transaction=98158175 ;

ROLLBACK PREPARED ‘gid';
```

试试看吧！

GREENPLUM DATABASE

腾讯云大学

欢迎下载试用Greenplum

GREENPLUM DATABASE 腾讯云大学

Why GitHub? Team Enterprise Explore Marketplace Pricing Search / Sign in Sign up

Watch 429 Star 4.2k Fork 1.2k

6 Wiki Security Insights

Be notified of new releases

Create your free GitHub account today to subscribe to this repository for new releases and build software alongside 50 million developers.

Sign up

Dismiss

Releases Tags

Latest release

6.11.2

d611e78

Compare

6.11.2

gp-releng released this 19 days ago · 13153 commits to master since this release

Greenplum-db version: 6.11.2

Assets 8

6.11.2-src-full.tar.gz	77.5 MB
6.11.2-src-full.zip	85.2 MB
greenplum-db-6.11.2-rhel6-x86_64.rpm	66.8 MB
greenplum-db-6.11.2-rhel7-x86_64.rpm	65.8 MB
greenplum-db-6.11.2-ubuntu18.04-amd64.deb	58.9 MB
open_source_license_greenplum-database-6.0.0-97773a0-1566481111.txt	130 KB
Source code (zip)	
Source code (tar.gz)	

21 days ago

5.28.2

感谢观看

Greenplum中文社区公众号

Greenplum微信技术讨论群

GREENPLUM DATABASE

腾讯云大学

GREENPLUM DATABASE 腾讯云大学

Q&A