Greenplum 释放表的空间

1 Greenplum 产生垃圾空间说明

Greenplum 支持行储存(HEAP 储存)与列(append-only)储存,对于 AO 存储，虽然是 appendonly，但实际上 GP 是支持 DELETE 和 UPDATE 的，被删除或更新的行，通过 visimap 来标记记录的可见性和是否已删除。AO 存储是块级组织，当一个块内的数据大部分都被删除或更新掉时，扫描它浪费的成本实际上是很高的。而 PostgreSQL 是通过 HOT 技术以及 autovacuum 来避免或减少垃圾的。但是 Greenplum 没有自动回收的 worker 进程，所以需要人为的触发。接下来就分析 AO 表与 HEAP 表的问题以及如何解答，执行空间的释放有 3 中方法分别是:
1、执行 VACUUM 只是简单的回收空间且令其可以再次使用。（当膨胀率大于 gp_appendonly_compaction_threshold 参数时），为共享锁，没有请求排它锁，仍旧可以对表读写。
2、执行 VACUUM FULL 更广泛的处理，包括跨块移动行，以便把表压缩至使用最少的磁盘块数目存储。相对 vacuum 要慢。（不管 gp_appendonly_compaction_threshold 参数的设置，都会回收垃圾空间。），为 DDL(排它锁)锁，需要慎用这个命令，会把 CPU 与 IO 沾满。
3、执行重分布。（不管 gp_appendonly_compaction_threshold 参数，都会回收垃圾空间。），为 DDL 锁。

2 查看表的储存类型

名字	引用	描述
regproc	pg_proc	函数名字
regprocedure	pg_proc	带参数类型的函数
regoper	pg_operator	操作符名
regoperator	pg_operator	带参数类型的操作符
regclass	pg_class	关系名

2.1 执行查看命令

以下命令是开启执行的时间，并查看表的类型

```
stagging=# \timing on
Timing is on.

stagging=# select distinct relstorage from pg_class ;
 relstorage
------------
 h
 a
```

 x
 v
 c
(5 rows)

Time: 6.132 ms

2.2 名词解释

timing 打开 SQL 的执行时间,参数分为 on 与 off
h = 堆表(heap)、索引
a = append only row 存储表
c = append only column 存储表
x = 外部表(external table)
v = 视图

3 AO 表分析

3.1 查看当前数据库中有哪些 AO 表

3.1.1 查看当前数据库的所有 AO 表

以下查看是查看当前数据库下的所有的 AO 表
stagging=# select t2.nspname, t1.relname from pg_class t1, pg_namespace t2 where t1.relnamespace=t2.oid and relstorage in ('c', 'a');

 nspname | relname
---------+--------------------------------
test_ao | ao_table_test
(12 rows)

Time: 6.828 ms

可以看出来 ao_table_test 为 AO 表

3.1.2 查看制定 schema 下的 AO 表

stagging=# select t2.nspname, t1.relname from pg_class t1, pg_namespace t2 where t1.relnamespace=t2.oid and relstorage in ('c', 'a') and t2.nspname = 'main';

```
 nspname |             relname
---------+---------------------------------
test_ao      |          ao_table_test
(12 rows)

Time: 6.828 ms
```

main 是当前数据库下的 schema

3.2 查看 AO 表的膨胀率

表的膨胀率也就是表中执行 DELETE 和 UPDATE 产生的垃圾

3.2.1 执行查看命令

stagging=# select * from
gp_toolkit.__gp_aovisimap_compaction_info('test_ao.ao_table_test'::regclass);
NOTICE: gp_appendonly_compaction_threshold = 10

```
stagging=# select * from gp_toolkit.__gp_aovisimap_compaction_info('test_ao.ao_table_test'::regclass);
NOTICE:  gp_appendonly_compaction_threshold = 10
 content | datafile | compaction_possible | hidden_tupcount | total_tupcount | percent_hidden
---------+----------+---------------------+-----------------+----------------+----------------
      17 |        0 | f                   |               0 |              0 |           0.00
      17 |        1 | t                   |          671375 |        2369294 |          28.34
      17 |        2 | t                   |          620098 |        1843910 |          33.63
      17 |        3 | t                   |         1077821 |        1077821 |         100.00
      17 |        4 | f                   |               0 |              0 |           0.00
      40 |        0 | f                   |               0 |              0 |           0.00
      40 |        1 | t                   |          668184 |        2361452 |          28.30
      40 |        2 | t                   |          617547 |        1852376 |          33.34
      40 |        3 | t                   |         1075721 |        1075721 |         100.00
      40 |        4 | f                   |               0 |              0 |           0.00
      41 |        0 | f                   |               0 |              0 |           0.00
      41 |        1 | t                   |          669732 |        2358973 |          28.39
      41 |        2 | t                   |          616413 |        1849942 |          33.32
      41 |        3 | t                   |         1072828 |        1072828 |         100.00
      41 |        4 | f                   |               0 |              0 |           0.00
      44 |        0 | f                   |               0 |              0 |           0.00
      44 |        1 | t                   |          664703 |        2356460 |          28.21
      44 |        2 | t                   |          616464 |        1849219 |          33.34
      44 |        3 | t                   |         1075293 |        1075293 |         100.00
```

(240 rows)

Time: 127.750 ms

3.2.3 名词解释

test_ao : schema 的名字
ao_table_test:当前 schema 下的表
gp_appendonly_compaction_threshold: AO 的压缩进程，目前设置的是 10

content:对应 gp_configuration.content 表示 greenplum 每个节点的唯一编号。
datafile:这条记录对应的这个表的其中一个数据文件的编号，每个数据文件假设 1GB。
hidden_tupcount:有多少条记录已更新或删除（不可见）。
total_tupcount:总共有多少条记录（包括已更新或删除的记录）。
percent_hidden:不可见记录的占比。如果这个占比大于 gp_appendonly_compaction_threshold 参数，那么执行 vacuum 时，会收缩这个数据文件。
compaction_possible:这个数据文件是否可以被收缩。（通过 gp_appendonly_compaction_threshold 参数和 percent_hidden 值判断）。

在以上中可以看出在 17 节点上的第 1 号文件有 2369294 记录其中有 671375 条记录被更新或删除，其中不可见的比例为 28.34%

3.3 检查系统中膨胀率超过 N 的 AO 表

3.3.1 执行命令

stagging=# select * from (select t2.nspname, t1.relname,
(gp_toolkit.__gp_aovisimap_compaction_info(t1.oid)).* from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('c', 'a')) t where t.percent_hidden > 0.2;

```
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
 nspname |    relname    | content | datafile | compaction_possible | hidden_tupcount | total_tupcount | percent_hidden
---------+---------------+---------+----------+---------------------+-----------------+----------------+----------------
 test_ao | ao_table_test |      22 |        1 | t                   |          663840 |        2359842 |          28.34
 test_ao | ao_table_test |      22 |        2 | t                   |          615118 |        1838946 |          33.63
 test_ao | ao_table_test |      22 |        3 | t                   |         1070435 |        1076952 |         100.00
 test_ao | ao_table_test |      17 |        1 | t                   |          667252 |        2348411 |          28.10
 test_ao | ao_table_test |      17 |        2 | t                   |          617841 |        1835328 |          33.70
 test_ao | ao_table_test |      17 |        3 | t                   |         1075383 |        1069380 |         100.00
 test_ao | ao_table_test |      20 |        1 | t                   |          671554 |        2360304 |          28.01
 test_ao | ao_table_test |      20 |        2 | t                   |          618555 |        1851430 |          33.57
 test_ao | ao_table_test |      20 |        3 | t                   |         1074171 |        1073118 |         100.00
 test_ao | ao_table_test |      23 |        1 | t                   |          663191 |        2369294 |          28.39
 test_ao | ao_table_test |      23 |        2 | t                   |          619699 |        1843910 |          33.32
 test_ao | ao_table_test |      23 |        3 | t                   |         1076952 |        1077821 |         100.00
```

(144 rows)

Time: 864.715 ms

以上命令是查询膨胀率超过千分之 2 的 AO 表

3.3.2 名词解释

nspname: 表示查询的 schema 的名字
relname: 是当前 schema 的表的名字

在以上数据中可以看出在每个节点上的膨胀率也不同

3.4 查看膨胀数据的占用大小

```
stagging=# select pg_size_pretty(pg_relation_size('test_ao.ao_table_test'));
 pg_size_pretty
----------------
 16 GB
(1 row)

Time: 32.806 ms
```

在以上可以看出膨胀率占用了 16G 的空间

3.5 查看表的行数

```
stagging=# select count(*) from test_ao.ao_table_test;
    count
-----------
 140324396
(1 row)

Time: 1842.706 ms
```

3.6 释放膨胀的空间

在以上的数据中可以看出膨胀率大于了 gp_appendonly_compaction_threshold 的值可以直接使用 vacuum 命令进行收缩

```
stagging=# vacuum test_ao.ao_table_test;
VACUUM
Time: 57800.144 ms
```

3.7 查看释放后的占用空间

3.7.1 释放膨胀空间

```
stagging=# select pg_size_pretty(pg_relation_size('test_ao.ao_table_test'));
 pg_size_pretty
----------------
 8859 MB
(1 row)
```

Time: 34.990 ms

以上可以看出已经释放了大部分的空间

3.7.2 再次查看 AO 的膨胀率

stagging=# select * from (select t2.nspname, t1.relname,
(gp_toolkit.__gp_aovisimap_compaction_info(t1.oid)).* from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('c', 'a')) t where t.percent_hidden >
0.01;

```
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
 nspname | relname | content | datafile | compaction_possible | hidden_tupcount | total_tupcount | percent_hidden
---------+---------+---------+----------+---------------------+-----------------+----------------+----------------
(0 rows)

Time: 871.210 ms
```

以上命令是查询膨胀率超过万分之 1 的 AO 表

3.8 再次查看表的行数

stagging=# select count(*) from test_ao.ao_table_test;
  count

 140324396
(1 row)

Time: 1680.919 ms

从以上可以看出与第一次查询出来的行数一直

3.9 使用更改随机的方式释放空间

3.9.1 查看膨胀占用空间

stagging=# select * from (select t2.nspname, t1.relname,
(gp_toolkit.__gp_aovisimap_compaction_info(t1.oid)).* from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('c', 'a')) t where t.percent_hidden > 0.01;

```
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
 nspname |    relname    | content | datafile | compaction_possible | hidden_tupcount | total_tupcount | percent_hidden
---------+---------------+---------+----------+---------------------+-----------------+----------------+----------------
 test_ao | ao_table_test |      22 |        1 | t                   |          285721 |        1190995 |          23.98
 test_ao | ao_table_test |      22 |        4 | t                   |          903268 |        1693879 |          53.47
 test_ao | ao_table_test |      40 |        1 | t                   |          284310 |        1193448 |          23.94
 test_ao | ao_table_test |      40 |        4 | t                   |          904009 |        1697919 |          53.48
 test_ao | ao_table_test |      41 |        1 | t                   |          278898 |        1187631 |          24.02
 test_ao | ao_table_test |      41 |        4 | t                   |          892752 |        1688449 |          53.40
 test_ao | ao_table_test |      42 |        1 | t                   |          285455 |        1192682 |          23.87
 test_ao | ao_table_test |      42 |        4 | t                   |          903550 |        1695349 |          53.55
```

3.9.2 随机改变表的分布键

stagging=# alter table test_ao.ao_table_test set with (reorganize=true) distributed randomly;
ALTER TABLE
Time: 81169.170 ms

3.9.3 查看释放后的空间

stagging=# select * from (select t2.nspname, t1.relname,
(gp_toolkit.__gp_aovisimap_compaction_info(t1.oid)).* from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('c', 'a')) t where t.percent_hidden > 0.01;

```
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
 nspname | relname | content | datafile | compaction_possible | hidden_tupcount | total_tupcount | percent_hidden
---------+---------+---------+----------+---------------------+-----------------+----------------+----------------
(0 rows)

Time: 730.748 ms
```

注意在执行随机分布键是在非业务的时候执行，执行 distribute 会执行排它锁不,要堵塞业务。

3.10 使用多分布键的形式释放空间

3.10.1 执行重新分布命令

stagging=# alter table test_ao.ao_table_test set with (reorganize=true) distributed by (pripid,s_ext_nodenum);
ALTER TABLE
Time: 82621.274 ms

3.10.2 查看数据的膨胀率

stagging=# select * from (select t2.nspname, t1.relname,
(gp_toolkit.__gp_aovisimap_compaction_info(t1.oid)).* from pg_class t1, pg_namespace t2

where t1.relnamespace=t2.oid and relstorage in ('c', 'a')) t where t.percent_hidden > 0.01;

```
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
 nspname | relname | content | datafile | compaction_possible | hidden_tupcount | total_tupcount | percent_hidden
---------+---------+---------+----------+---------------------+-----------------+----------------+----------------
(0 rows)

Time: 936.319 ms
```

注意在执行随机分布键是在非业务的时候执行，执行 distribute 会执行排它锁不,要堵塞业务。

4 AO 表总结

4.1 查看表的行的个数

stagging=# select count(*) from test_ao.ao_table_test;
  count

 140324396
(1 row)

Time: 1764.584 ms

4.2 更新数据的行数与占用大小

4.2.1 更新数据

stagging=# update test_ao.ao_table_test set alttime='2018-10-23 11:54:57.000000' where nodenum='850000';

受影响的行: 5701,7349
时间: 104.007s

4.2.2 查看表的膨胀率

stagging=# select * from (select t2.nspname, t1.relname, (gp_toolkit.__gp_aovisimap_compaction_info(t1.oid)).* from pg_class t1, pg_namespace t2 where t1.relnamespace=t2.oid and relstorage in ('c', 'a')) t where t.percent_hidden > 0.01;

```
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
NOTICE:  gp_appendonly_compaction_threshold = 10
 nspname |    relname    | content | datafile | compaction_possible | hidden_tupcount | total_tupcount | percent_hidden
---------+---------------+---------+----------+---------------------+-----------------+----------------+----------------
 test_ao | ao_table_test |      17 |        1 | t                   |         1171650 |        4115179 |          29.00
 test_ao | ao_table_test |      44 |        1 | t                   |         1187074 |        4102600 |          28.88
 test_ao | ao_table_test |      40 |        1 | t                   |         1193448 |        4117086 |          28.91
 test_ao | ao_table_test |      43 |        1 | t                   |         1193383 |        4111523 |          28.86
 test_ao | ao_table_test |      20 |        1 | t                   |         1188319 |        4088453 |          28.84
```

(48 rows)

Time: 874.505 ms

在以上数据中可以看出 57017349 除以 140324396 大概是 40% ,膨胀率大概是 28.88 左右。

5 AO 表释放空间 SHELL 脚本

$ cat greenplum-inspect-ao.sh

```
#!bin/bash
# 1、把改脚本放到任意目录下
# 2、inspect-ao-sql 文件夹存放的是查询 AO 表的 SQL 与查询膨胀率的 SQL
# 3、log 文件夹则是存放临时生成的 schema 与 table 的文件,还有存放每个 AO 表的膨胀率详细的信息
# 4、释放空间使用的是 vacuum schema.tablename

#当前该脚本的路径
bashpath=$(cd `dirname $0`;pwd)

# 执行查看 AO 表的 SQL 脚本
inspect_ao_sql_ori=$bashpath"/inspect-ao-sql/inspect-ao-ori.sql"
inspect_ao_sql=$bashpath"/inspect-ao-sql/inspect-ao.sql"

# 查看表膨胀率的 SQL 脚本
inspect_ao_expansivity_ori=$bashpath"/inspect-ao-sql/inspect-ao-percent-hidden-ori.sql"
inspect_ao_expansivity=$bashpath"/inspect-ao-sql/inspect-ao-percent-hidden.sql"

# 所产生的日志路径
inspect_ao_log=$bashpath"/log"

#当前的日期
currentDate=`date +%Y%m%d`

#创建生成结果的临时目录
temp_inspect_results=$inspect_ao_log"/"$currentDate"/temp-inspect-results"
```

```
# 查看表膨胀率的详细信息
table_percent_hidden=$inspect_ao_log"/"$currentDate"/table-percent-hidden"

# 数据库名字
gpdatabase='****'

# scheam 名字
scheamname='main'

# gp 服务器 ip
gpip='192.168.***.11'

#gp port
gpport='5432'

# gp user
gpuser='gpadmin'

# gp password
gppassword='gpadmin'

# 需要检查的 schema
schema_inspect='main,ods'

# 删除日志文件并创建新文件
if [ -d $inspect_ao_log ];then
        rm -rf $inspect_ao_log
   fi

mkdir -p    $temp_inspect_results
mkdir -p    $table_percent_hidden

# 导入 GP 密码环境标量
export PGPASSWORD=$gppassword

# 获取数据并处理为需要的格式
array=(${schema_inspect//,/ })
for schema_var in ${array[@]}
do
    cp $inspect_ao_sql_ori $inspect_ao_sql
    sed -i 's/schemaName/'$schema_var'/g' $inspect_ao_sql

    export PGPASSWORD=$gppassword
    psql -d $gpdatabase    -h $gpip -p $gpport -U $gpuser -f    $inspect_ao_sql >>
```

```
$temp_inspect_results/$currentDate-$schema_var".txt"

    sed -i '1,2d' $temp_inspect_results/$currentDate-$schema_var".txt"
    sed -i '$d' $temp_inspect_results/$currentDate-$schema_var".txt"
    sed -i '$d' $temp_inspect_results/$currentDate-$schema_var".txt"

    cat $temp_inspect_results/$currentDate-$schema_var".txt" |awk -F '|' '{print $1,$2}'|awk -F
'\t' '{print $1"."$2}' >> $temp_inspect_results/$currentDate"-tra.txt"

    if [ -f $inspect_ao_sql ];then
        rm -rf $inspect_ao_sql
  fi
done

# 生成带有 schema 与 AO 表的文件
cat $temp_inspect_results/$currentDate"-tra.txt"|awk '{print $1"."$2}'|awk '{sub(/.$/,"")}1' >>
$temp_inspect_results/$currentDate"-finish.txt"

# 遍历带有 schema 与表名的文件
for schema_tablename in `cat $temp_inspect_results/$currentDate"-finish.txt"`
do
  echo $schema_tablename
  cp $inspect_ao_expansivity_ori $inspect_ao_expansivity
  sed -i 's/tablepath/'$schema_tablename'/g' $inspect_ao_expansivity

  export PGPASSWORD=$gppassword
  psql -d $gpdatabase    -h $gpip -p $gpport -U $gpuser -f    $inspect_ao_expansivity    >>
$table_percent_hidden"/"$schema_tablename".txt";
  psql -d $gpdatabase    -h $gpip -p $gpport -U $gpuser -c "vacuum $schema_tablename";

  if [ -f $inspect_ao_expansivity ];then
      rm -rf $inspect_ao_expansivity
  fi
done

# 正确退出脚本
exit 0
```