

Greenplum 数据库巡检报告



作者：小徐

制作时间：20190525

联系方式：xiaoxubigdata@163.com

目录

目录.....	2
1 巡检说明.....	6
2 查看集群硬件相关信息.....	6
2.1 查看集群当前系统时间.....	6
2.2 查看操作系统版本.....	7
2.3 查看内核版本.....	7
2.4 查看内存的详细信息.....	7
2.4.1 查看内存的使用情况.....	7
2.4.2 查看实时的内存使用情况.....	8
2.5 查看集群的 CPU 详细信息.....	8
2.5.1 查看集群中 CPU 的核数.....	8
2.5.2 实时查看 CPU 的使用情况.....	9
2.6 查看集群的磁盘的详细信息.....	10
2.6.1 查看集群磁盘使用空间.....	10
2.6.2 动态查看磁盘的使用情况.....	10
2.7 查看机器内核配置参数.....	10
2.8 查看系统打开文件的最大数.....	11
2.9 查看磁盘的挂在情况.....	12
2.10 查看开机需要启动的项.....	12
2.11 查看系统系统资源限制.....	12
2.12 查看集群的网络的详细信息.....	13
2.12.1 查看集群中网卡的大小.....	13
2.12.2 动态查看网络的吞吐情况.....	13
3 查看集群上运行的任务.....	14
3.1 查看集群上运行的定时任务.....	14
3.1.1 编写查看定时的脚本.....	14
3.1.2 运行定时的脚本.....	14
3.2 查看集群上运行 greenplum 的进程.....	15
4 查看集群的基本信息.....	15
4.1 查看集群的版本信息.....	15
4.2 查看 segment 相关信息.....	15
4.2.1 查看当前 down 的 segment 节点.....	15
4.2.2 查看当前处于 change tracking 的 segment 节点.....	16
4.2.3 查看当前处于 re-syncing 状态的 segment 节点.....	16
4.2.4 查看所有 segment 是否可达,确保 QD(query dispatching)正常.....	16
4.3 查看 standby 与 master 的信息.....	16
4.3.1 查看 standby 的配置.....	16
4.3.2 查看 standby 的运行状态.....	17
4.3.3 查看 master 节点的是否正常.....	17
4.3.4 检查 gp_persistent 表,确保主备 Segment 之间是否有数据不一致的问题.....	18
4.4 查看有问题的 segment primary/mirror 信息.....	18

4.5	显示 mirror 列表.....	19
4.6	显示所有配置参数.....	19
4.7	查看集群中数据库中的详细信息.....	19
4.7.1	查看每个数据库占用的大小.....	19
4.7.2	查看每个 schema 的占用大小.....	20
4.7.3	查看 AO 表的相关信息.....	21
4.7.3.1	查看数据库中的 AO 表.....	21
4.7.3.2	查看数据库中的 AO 表的数量.....	21
4.7.4	查看堆表的相关信息.....	21
4.7.4.1	查看数据库中的堆表.....	21
4.7.4.2	查看堆表的数量.....	21
4.7.5	查看外部表相关信息.....	22
4.7.5.1	查看外部表.....	22
4.7.5.2	查看外部表的数量.....	22
4.7.6	查看视图的相关信息.....	22
4.7.6.1	查看制定 schema 下视图.....	22
4.7.6.2	查看制定 schema 下视图的数量.....	22
4.7.7	查看表的相关信息.....	22
4.7.7.1	查看每个表的大小信息.....	22
4.8	查看索引的相关信息.....	23
4.8.1	查看索引大小超过表总大小 1/2 的系统表大小和索引大小.....	23
4.8.2	查看制定 schema 上表的索引.....	24
4.8.3	索引活跃度监控.....	24
4.8.3.1	IO 活跃度查看.....	24
4.8.3.2	访问活跃度.....	24
4.8.4	查看 Master 与 Segment 上索引不一致的问题.....	25
4.8.5	查看阴的使用次数.....	25
4.9	检查是否使用了 a-z 0-9 _ 以外的字母作为对象名.....	25
4.9.1	查看数据库是否使用了命名规范.....	25
4.9.2	查看表的索引和视图的命名规范.....	26
4.9.3	查看数据库中的类型命名规范.....	26
4.9.4	查看数据库中的储存过程的命名规范.....	26
4.9.5	查看数据库中的表,视图的的命名规范.....	26
4.10	查看集群是否处于 not balanced 状态.....	27
4.10.1	查看当前的连接数.....	27
4.10.2	查看密码有效期不足 30 天的用户.....	27
4.10.3	查看每个用户链接的个数.....	28
4.10.4	查看数据库的连接数.....	28
4.11	查看 ctid 的值.....	28
5	集群巡检过程详细信息.....	29
5.1	检查大小超过 1GB 的表倾斜情况.....	29
5.1.1	查看超过 1GB 倾斜率的表.....	29
5.1.2	查看表在 segment 上占用大小及倾斜率.....	30
5.1.3	查看表在 segment 上占用的行数及百分比.....	30

5.1.4 解决表分布倾斜的情况.....	31
5.1.4.1 分布键说明.....	31
5.1.4.2 修改分布键.....	31
5.1.4.3 对表进行重新创建.....	31
5.2 检查膨胀率过高的表.....	32
5.2.1 查看 schema 下的 AO 表.....	32
5.2.2 查看表的膨胀率.....	33
5.2.3 对表进行释放空间.....	33
5.3 检查元数据不一致问题.....	33
5.3.1 问题描述.....	33
5.3.2 集群检查.....	34
5.3.3 问题处理.....	34
5.4 查看服务器配置参数.....	34
5.4.1 服务器 OS 参数查看.....	34
5.5 查看集群参数详细信息.....	35
5.5.1 查看每个 segment 的内存配置参数.....	35
5.5.1.1 查看分配内存信息.....	35
5.5.1.2 修改内存参数.....	35
5.5.2 查看 shared_buffers(共享缓冲区)的内存.....	36
5.5.2.1 查看系统配置的参数.....	36
5.5.2.2 参数详解.....	36
5.5.2.3 修改参数.....	36
5.5.3 查看 max_connections(最大连接数).....	36
5.5.3.1 查看最大连接数参数.....	36
5.5.3.2 参数详解.....	37
5.5.4 查看 block_size(磁盘块)的大小.....	37
5.5.4.1 查看磁盘块的大小.....	37
5.5.4.2 参数详解.....	37
5.5.5 查看 work_mem 的值.....	37
5.5.5.1 查看集群中 work_mem 的配置大小.....	37
5.5.5.2 参数详解.....	38
5.5.5.3 修改参数.....	38
5.5.6 查看 statement_mem 的值.....	38
5.5.6.1 查看集群中 statement_mem 的值.....	38
5.5.6.2 参数详解.....	38
5.5.6.3 修改参数.....	38
5.5.7 查看 gp_workfile_limit_files_per_query 的值.....	39
5.5.7.1 查看此值的大小.....	39
5.5.7.1 参数详解.....	39
5.5.8 查看 gp_resqueue_priority_cpucore_per_segment 的值.....	39
5.5.8.1 查看此值的大小.....	39
5.5.8.2 参数详解.....	39
5.5.8.3 修改参数.....	40
5.6 备 Master 镜像情况.....	40

5.6.1 概述.....	40
5.6.2 查看备 master 的运行情况.....	40
5.6.3 备 master 需要检测的项.....	40
5.7 segment 镜像情况.....	41
5.7.1 查看集群中的镜像分布情况.....	41
5.8 资源队列情况.....	41
5.8.1 用户与资源队列检查.....	41
5.8.1.1 查看负载管理资源队列的状态和活动.....	41
5.8.1.2 查看当前用户使用的是什么队列.....	42
5.8.1.3 查看队列的活动负载状态.....	43
5.8.1.4 查看负载管理特性的 Greenplum 数据库资源队列的信息.....	44
5.8.1.5 查看队列资源的状态.....	44
5.8.1.6 查看每个资源队列的配置情况.....	45
5.8.1.7 修改资源队列的语句.....	46
5.9 Statistics 状态检查.....	46
5.9.1 查看所有的表.....	46
5.9.2 查看指定 schema 下的表.....	47
5.10 Skew 状态检查.....	47
5.11 Bloat 状态.....	47
5.12 集群系统性能.....	48
5.12.1 DB 性能查看.....	48
5.12.2 SQL 锁检.....	48
5.12.3 集群硬件性能查看.....	49
5.13 查看集群系统元数据.....	50
5.14 查看集群中分布键倾斜率比较高的表.....	50
5.15 使用 gpcheckcat 命令检测集群.....	50
5.15.1 gpcheckcat 检测项说明.....	50
5.15.2 检查 master,segment 的 catalog 一致性.....	50
5.15.3 检查持久化表的 catalog 一致性。.....	51
5.15.4 检查 pg_class 与 pg_attribute 是否不一致.....	51
5.16 表活跃度监控.....	52
5.16.1 IO 活跃度的表.....	52
5.16.2 表的访问活跃度.....	53
5.16.3 热表的查询.....	54
5.16.4 冷表的查询.....	54

1 巡检说明

作为上线的数据库必须经常做一下巡检，就像人的身体要经常做检查一样，敲代码的说不定哪一天就挂了，巡检发现问题，并及时排除问题，让 greenplum 数据库健康运行，它没有问题，我们也放心、、、

2 查看集群硬件相关信息

创建集群所有主机的映射文件,如:cluster_hosts

```
$ cat cluster_hosts
```

```
gpmdw
```

```
gpsdw1
```

```
gpsdw2
```

```
gpsdw3
```

执行 gpssh 命令

```
$ gpssh -f cluster_hosts
```

```
=>
```

使用 greenplum 自带的 gpssh 命令执行集群运行的参数,详细资料请参考:

https://gp-docs-cn.github.io/docs/utility_guide/admin_utilities/gpssh.html

2.1 查看集群当前系统时间

```
=> date +"%Y-%m-%d %H:%M:%S"
```

```
[gpsdw3] 2019-05-29 17:34:27
```

```
[gpmdw] 2019-05-29 17:34:27
```

```
[gpsdw1] 2019-05-29 17:34:27
```

```
[gpsdw2] 2019-05-29 17:34:27
```

主要查看集群时间是否与外网时间是否同步

2.2 查看操作系统版本

=> cat /etc/system-release

[gpsdw3] CentOS Linux release 7.2.1511 (Core)

[gpmdw] CentOS Linux release 7.2.1511 (Core)

[gpsdw1] CentOS Linux release 7.2.1511 (Core)

[gpsdw2] CentOS Linux release 7.2.1511 (Core)

在以上可以看出机器使用的是 centos 7.2 发布版

2.3 查看内核版本

=> uname -a

```
--
=> uname -a
[gpsdw3] Linux gpsdw3 3.10.0-693.21.1.el7.x86_64 #1 SMP Wed Mar 7 19:03:37 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
[gpmdw] Linux gpmdw 3.10.0-693.21.1.el7.x86_64 #1 SMP Wed Mar 7 19:03:37 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
[gpsdw1] Linux gpsdw1 3.10.0-693.21.1.el7.x86_64 #1 SMP Wed Mar 7 19:03:37 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
[gpsdw2] Linux gpsdw2 3.10.0-693.21.1.el7.x86_64 #1 SMP Wed Mar 7 19:03:37 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
=> []
```

在以上可以看出使用的 3.10.0-693.21.1.el7.x86_64 的内核 64 位操作系统

2.4 查看内存的详细信息

2.4.1 查看内存的使用情况

=> free -g

```
=> free -g
[gpmdw3]          total        used        free      shared  buff/cache   available
[gpmdw3] Mem:      377          14           3           9         360        352
[gpmdw3] Swap:      15           0          15
[gpmdw]          total        used        free      shared  buff/cache   available
[gpmdw] Mem:      125          15          93           1          17        108
[gpmdw] Swap:      15           0          15
[gpsdw1]          total        used        free      shared  buff/cache   available
[gpsdw1] Mem:      346          14          35           9         296        321
[gpsdw1] Swap:      15           0          15
[gpsdw2]          total        used        free      shared  buff/cache   available
[gpsdw2] Mem:      377          10          19           7         347        357
[gpsdw2] Swap:      15           0          15
```

由于 gpmdw 是 master 内存小了一点，其他的数据节点都被缓存使用了

2.4.2 查看实时的内存使用情况

以下命令式输出 10 次结果

```
$ vmstat 1 10
```

```
[gpadmin@gpmdw /home/xiaoxu/gp-dump-test]$ vmstat 1 10
procs -----memory----- --swap-- --io-- --system-- -----cpu-----
r  b   swpd   free   buff  cache   si   so    bi    bo    in   cs  us  sy  id  wa  st
1  0     256 91519192    28 24066544    0    0     1     82     0    0  1  1 98  0  0
0  0     256 91519376    28 24066552    0    0     0    124 5911 10166 1  1 99  0  0
0  0     256 91519592    28 24066568    0    0     0   236 4002 6308 1  0 99  0  0
0  0     256 91519104    28 24066628    0    0     0   180 6269 9354 1  1 98  0  0
0  0     256 91518896    28 24066628    0    0     0     0 3763 5717 0  0 99  0  0
1  0     256 91518304    28 24066628    0    0     0    64 4502 7448 0  0 99  0  0
1  0     256 91518232    28 24066652    0    0     0   108 5962 10133 1  1 99  0  0
0  0     256 91518560    28 24066664    0    0     0   188 4363 6323 1  1 99  0  0
0  0     256 91518616    28 24066680    0    0     0   280 4844 7829 1  0 99  0  0
1  0     256 91519040    28 24066668    0    0     0    44 4919 6448 1  0 99  0  0
```

主要关注 buffers/cache 的 free 是否已经快没有了，如果是，那就说明内存使用已经很吃紧了。还有就是看 swap 是否已经开始使用了，如果 swap 开始使用，说明操作系统的内存已经不足。在内存使用过程中，要时刻监控空闲内存的大小，如果发现内存占用过高，尤其是 swap 开始使用时（swap 开始使用，会导致 Greenplum 性能大减），要及时发现并处理。

2.5 查看集群的 CPU 详细信息

2.5.1 查看集群中 CPU 的核数

```
=> lscpu | grep -E "Thread|Core|Socket|Model name|CPU"
```

或使用

```
=> lscpu
```



```
=> lscpu|grep -E "per core|per socket|Socket|Model name"
[gpsdw3] Thread(s) per core: 2
[gpsdw3] Core(s) per socket: 18
[gpsdw3] Socket(s): 2
[gpsdw3] Model name: Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz
[ gpmdw] Thread(s) per core: 1
[ gpmdw] Core(s) per socket: 6
[ gpmdw] Socket(s): 2
[ gpmdw] Model name: Intel(R) Xeon(R) CPU E5-2603 v4 @ 1.70GHz
[gpsdw2] Thread(s) per core: 2
[gpsdw2] Core(s) per socket: 18
[gpsdw2] Socket(s): 2
[gpsdw2] Model name: Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz
[gpsdw1] Thread(s) per core: 2
[gpsdw1] Core(s) per socket: 18
[gpsdw1] Socket(s): 2
[gpsdw1] Model name: Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz
```

有与集群机器多，这里只列出了一台数据节点的详细信息，可以看出该台机器 18 核，2.1GHz 的

2.5.2 实时查看 CPU 的使用情况

以下是查看机器上所有的 CPU 的核,每 1 秒刷新一次

```
$ mpstat -P ALL 1
```

```
[gpadmin@gpmdw /home/xiaoxu/gp-dump-test]$ mpstat -P ALL 1
Linux 3.10.0-693.21.1.el7.x86_64 (gpmdw) 06/11/2019 _x86_64_ (12 CPU)

03:58:31 PM CPU      %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
03:58:32 PM all      0.67    0.00    0.25   0.08    0.00   0.00   0.00   0.00   0.00   99.00
03:58:32 PM 0        0.00    0.00    0.99   0.00    0.00   0.00   0.00   0.00   0.00   99.01
03:58:32 PM 1        1.00    0.00    0.00   0.00    0.00   0.00   0.00   0.00   0.00   99.00
03:58:32 PM 2        1.00    0.00    0.00   0.00    0.00   0.00   0.00   0.00   0.00   99.00
03:58:32 PM 3        0.99    0.00    0.99   0.00    0.00   0.00   0.00   0.00   0.00   98.02
03:58:32 PM 4        2.00    0.00    0.00   0.00    0.00   0.00   0.00   0.00   0.00   98.00
03:58:32 PM 5        1.00    0.00    0.00   0.00    0.00   0.00   0.00   0.00   0.00   99.00
03:58:32 PM 6        1.00    0.00    0.00   0.00    0.00   1.00   0.00   0.00   0.00   98.00
03:58:32 PM 7        0.00    0.00    0.00   0.00    0.00   0.00   0.00   0.00   0.00  100.00
03:58:32 PM 8        0.00    0.00    0.00   0.00    0.00   0.00   0.00   0.00   0.00  100.00
03:58:32 PM 9        0.00    0.00    0.00   0.00    0.00   0.00   0.00   0.00   0.00  100.00
03:58:32 PM 10       1.01    0.00    0.00   0.00    0.00   0.00   0.00   0.00   0.00   98.99
03:58:32 PM 11       0.00    0.00    1.00   0.00    0.00   0.00   0.00   0.00   0.00   99.00
```

%user 在 internal 时间段里，用户态的 CPU 时间(%)，不包含 nice 值为负进程
(usr/total)*100

%nice 在 internal 时间段里，nice 值为负进程的 CPU 时间(%) (nice/total)*100

%sys 在 internal 时间段里，内核时间(%) (system/total)*100

%iowait 在 internal 时间段里，硬盘 IO 等待时间(%) (iowait/total)*100

%irq 在 internal 时间段里，硬中断时间(%) (irq/total)*100

%soft 在 internal 时间段里，软中断时间(%) (softirq/total)*100

%idle 在 internal 时间段里，CPU 除去等待磁盘 IO 操作外的因为任何原因而空闲的时间
闲置时间(%) (idle/total)*100

2.6 查看集群的磁盘的详细信息

2.6.1 查看集群磁盘使用空间

=> df -h

```
=> df -h
[gpsdw3] Filesystem                Size      Used Avail Use% Mounted on
[gpsdw3] /dev/mapper/centos-root    300G    176G    125G   59% /
[gpsdw3] devtmpfs                    189G         0    189G    0% /dev
[gpsdw3] tmpfs                        189G         0    189G    0% /dev/shm
[gpsdw3] tmpfs                        189G     2.5G    187G    2% /run
[gpsdw3] tmpfs                        189G         0    189G    0% /sys/fs/cgroup
[gpsdw3] /dev/sda2                     1014M    137M    878M   14% /boot
[gpsdw3] /dev/sdal                     1022M     9.5M    1013M    1% /boot/efi
[gpsdw3] /dev/mapper/centos-data     8.5T     4.8T     3.7T   57% /greenplum
[gpsdw3] tmpfs                      38G         0     38G    0% /run/user/0
[gpsdw3] tmpfs                      38G         0     38G    0% /run/user/3000
```

有与集群机器多，这里只列出了一台数据节点的详细信息，可以看出该数据节点的数据目录 /greenplum 已经使用 57%，建议不要超过 70%

2.6.2 动态查看磁盘的使用情况

\$ iostat -x 1

```
[gpadmin@gpmdw /home/xiaoxu/gp-dump-test]$ iostat -x 1
Linux 3.10.0-693.21.1.el7.x86_64 (gpmdw)      06/11/2019      _x86_64_      (12 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           1.24    0.05    0.64    0.02    0.00   98.06

Device:            rrqm/s   wrqm/s     r/s     w/s    kB/s    kB/s  avgrq-sz  avgqu-sz   await  r_await  w_await  svctm  %util
sda                 0.00     0.36    0.07   11.97   14.97   982.31    165.58     0.25   21.14     9.28   21.21    0.20   0.24
dm-0                 0.00     0.00    0.01    0.90    1.26   44.82   100.76     0.01   12.78     1.86   12.91    0.16   0.01
dm-1                 0.00     0.00    0.00    0.00    0.00    0.00   14.41     0.00    6.66     0.16   22.20    2.95   0.00
dm-2                 0.00     0.00    0.06   11.43   13.71   937.49   165.56     0.25   21.32    10.65   21.38    0.20   0.23
```

在以上可以详细的看出服务器的详细信息，重点观察最后一个参数%util，它表示磁盘使用时间的占比，当这个数据是 100%时，就说明磁盘已经很忙碌了。然后再看下 rsec/s、wsec/s 等参数，观察磁盘的吞吐。

2.7 查看机器内核配置参数

=> grep "[a-z]" /etc/sysctl.conf

```
=> grep "^[a-z]" /etc/sysctl.conf
[gpsdw3] kernel.shmmax = 1800000000000
[gpsdw3] kernel.shmmni = 8192
[gpsdw3] kernel.shmall = 1800000000000
[gpsdw3] kernel.sem = 1000 10240000 400 10240
[gpsdw3] kernel.sysrq = 1
[gpsdw3] kernel.core_uses_pid = 1
[gpsdw3] kernel.msgmnb = 65536
[gpsdw3] kernel.msgmax = 65536
[gpsdw3] kernel.msgmni = 2048
[gpsdw3] net.ipv4.tcp_syncookies = 1
[gpsdw3] net.ipv4.ip_forward = 0
[gpsdw3] net.ipv4.conf.default.accept_source_route = 0
[gpsdw3] net.ipv4.tcp_tw_recycle = 1
[gpsdw3] net.ipv4.tcp_max_syn_backlog = 4096
[gpsdw3] net.ipv4.conf.all.arp_filter = 1
[gpsdw3] net.ipv4.ip_local_port_range = 1025 65535
[gpsdw3] net.core.netdev_max_backlog = 10000
[gpsdw3] net.core.rmem_max = 2097152
[gpsdw3] net.core.wmem_max = 2097152
[gpsdw3] vm.overcommit_memory = 1
[gpsdw3] vm.swappiness = 1
[gpsdw3] kernel.pid_max = 655360
```

详细的参数含义请查看:

<https://blog.csdn.net/xfg0218/article/details/91536066>

2.8 查看系统打开文件的最大数

```
=> grep -v "^#" /etc/security/limits.conf|grep -v "^$"
```

```
=> grep -v "^#" /etc/security/limits.conf|grep -v "^$"
[gpsdw3] * soft nofile 65536
[gpsdw3] * hard nofile 65536
[gpsdw3] * soft nproc 131072
[gpsdw3] * hard nproc 131072
[ gpmdw] * soft nofile 65536
[ gpmdw] * hard nofile 65536
[ gpmdw] * soft nproc 131072
[ gpmdw] * hard nproc 131072
[gpsdw1] * soft nofile 65536
[gpsdw1] * hard nofile 65536
[gpsdw1] * soft nproc 131072
[gpsdw1] * hard nproc 131072
[gpsdw2] * soft nofile 65536
[gpsdw2] * hard nofile 65536
[gpsdw2] * soft nproc 131072
[gpsdw2] * hard nproc 131072
=> █
```

详细参数详解请查看:

<https://blog.csdn.net/xfg0218/article/details/91554032>

2.9 查看磁盘的挂在情况

=> egrep -v "^#|^\$" /etc/fstab

```
=> egrep -v "^#|^$" /etc/fstab
[gpsdw3] /dev/mapper/centos-root / xfs defaults 0 0
[gpsdw3] UUID=511f1e62-1d0b-4613-88a8-dd6300e4432a /boot xfs defaults 0 0
[gpsdw3] UUID=C552-0989 /boot/efi vfat defaults,uid=0,gid=0,umask=0077,shortname=winnt 0 0
[gpsdw3] /dev/mapper/centos-swap swap swap defaults 0 0
[gpsdw3] /dev/centos/data /greenplum xfs nodev,noatime,inode64,allocsize=16m 0 0
[ gpmdw] /dev/mapper/centos-root / xfs defaults 0 0
[ gpmdw] UUID=a15acf68-07d3-4699-92c2-80f5bce87e10 /boot xfs defaults 0 0
[ gpmdw] UUID=B9E0-D344 /boot/efi vfat defaults,uid=0,gid=0,umask=0077,shortname=winnt 0 0
[ gpmdw] /dev/mapper/centos-swap swap swap defaults 0 0
[ gpmdw] /dev/centos/data /greenplum xfs nodev,noatime,inode64,allocsize=16m 0 0
[gpsdw1] /dev/mapper/centos-root / xfs defaults 0 0
[gpsdw1] UUID=54459ccd-3ccb-43ed-9cac-dd9eab9c0cf6 /boot xfs defaults 0 0
[gpsdw1] UUID=EF8C-E450 /boot/efi vfat defaults,uid=0,gid=0,umask=0077,shortname=winnt 0 0
[gpsdw1] /dev/mapper/centos-swap swap swap defaults 0 0
[gpsdw1] /dev/centos/data /greenplum xfs nodev,noatime,inode64,allocsize=16m 0 0
[gpsdw2] /dev/mapper/centos-root / xfs defaults 0 0
[gpsdw2] UUID=ccfa2089-1cda-4011-97b2-9d862433c08c /boot xfs defaults 0 0
[gpsdw2] UUID=B4BC-4693 /boot/efi vfat defaults,uid=0,gid=0,umask=0077,shortname=winnt 0 0
[gpsdw2] /dev/mapper/centos-swap swap swap defaults 0 0
[gpsdw2] /dev/centos/data /greenplum xfs nodev,noatime,inode64,allocsize=16m 0 0
```

2.10 查看开机需要启动的项

=> cat /etc/rc.local|grep '^[a-z]'

```
=> cat /etc/rc.local|grep '^[a-z]'
[gpsdw3] touch /var/lock/subsys/local
[gpsdw3] ulimit -SHn 102400
[ gpmdw] touch /var/lock/subsys/local
[ gpmdw] ulimit -SHn 102400
[gpsdw1] touch /var/lock/subsys/local
[gpsdw1] ulimit -SHn 102400
[gpsdw2] touch /var/lock/subsys/local
[gpsdw2] ulimit -SHn 102400
=> █
```

在以上可以看出开机需要执行以下操作

touch /var/lock/subsys/local

ulimit -SHn 102400

2.11 查看系统系统资源限制

=> for dir in `ls /etc/security/limits.d`; do echo "/etc/security/limits.d/\$dir : "; grep -v "^#" /etc/security/limits.d/\$dir|grep -v "^\$"; echo ""; done


```

=> for dir in `ls /etc/security/limits.d`; do echo "/etc/security/limits.d/$dir";
[gpsdw3] /etc/security/limits.d/20-nproc.conf :
[gpsdw3] *          soft   nproc   16384
[gpsdw3] root        soft   nproc   unlimited
[gpsdw3]
[ gpmdw] /etc/security/limits.d/20-nproc.conf :
[ gpmdw] *          soft   nproc   16384
[ gpmdw] root        soft   nproc   unlimited
[ gpmdw]
[gpsdw2] /etc/security/limits.d/20-nproc.conf :
[gpsdw2] *          soft   nproc   16384
[gpsdw2] root        soft   nproc   unlimited
[gpsdw2]
[gpsdw1] /etc/security/limits.d/20-nproc.conf :
[gpsdw1] *          soft   nproc   16384
[gpsdw1] root        soft   nproc   unlimited
[gpsdw1]
=> █

```

以上的储存不要超过 70%，否则会影响集群的性能下降

2.12 查看集群的网络的详细信息

2.12.1 查看集群中网卡的大小

```

=> ethtool bond0|grep "Speed"

```

```

[gpsdw3]    Speed: 10000Mb/s
[ gpmdw]    Speed: 4000Mb/s
[gpsdw1]    Speed: 10000Mb/s
[gpsdw2]    Speed: 10000Mb/s

```

bond0：绑定网卡的名字

在以上可以看出 gpmdw 使用了 4000Mb 的网卡，其余的都是 10000Mb 网卡

2.12.2 动态查看网络的吞吐情况

```

$ sar -n DEV 1 100

```

```
[gpadmin@gpmdw /home/xiaoxu/gp-dump-test]$ sar -n DEV 1 100
Linux 3.10.0-693.21.1.el7.x86_64 (gpmdw) 06/11/2019 _x86_64_ (12 CPU)

03:34:34 PM      IFACE  rxpck/s  txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
03:34:35 PM    bond0    206.00   300.00    19.36   141.39     0.00     0.00     3.00
03:34:35 PM        lo    260.00   260.00    33.72    33.72     0.00     0.00     0.00
03:34:35 PM       em1     65.00    78.00     5.34    44.93     0.00     0.00     2.00
03:34:35 PM       em2     14.00    13.00     1.06     2.22     0.00     0.00     0.00
03:34:35 PM       em4     61.00   111.00     5.69    52.88     0.00     0.00     0.00
03:34:35 PM       em3     66.00    98.00     7.28    41.35     0.00     0.00     1.00
```

3 查看集群上运行的任务

3.1 查看集群上运行的定时任务

3.1.1 编写查看定时的脚本

以下脚本请在 root 的用户下运行

```
# cat cluster-crontab.sh
#!/bin/bash

for OS_USER in `cat /etc/passwd | grep -e '/bin/bash' | awk -F":" '{print $1}`
do
echo "用户 $OS_USER " && crontab -l -u $OS_USER
done
```

3.1.2 运行定时的脚本

```
# sh cluster-crontab.sh
用户 root
0-59/10 * * * * /usr/sbin/ntpdate 192.168.201.11
用户 wasd
no crontab for wasd
用户 gpadmin
no crontab for gpadmin
用户 gp
no crontab for gp
用户 opsadmin
no crontab for opsadmin
```

查看每台集群上的定时的任务

3.2 查看集群上运行 greenplum 的进程

脚本下载:

<https://github.com/xfg0218/shellForRoot>

```
# sh cmd.sh all "ps -ef|grep greenplum"
```

```
===== exec ps -ef|grep greenplum for gpmdw start =====
root@gpmdw's password:
gpadmin 345 1697 0 Apr16 ? 01:08:53 /greenplum/soft/greenplum-db-5.11.1/bin/gpmmon -D /greenplum/data/gpmaster/gpseg-1/gpperfmon/conf/gpperfmon.conf -p 5432
gpadmin 1697 1 0 Mar11 ? 00:29:23 /greenplum/soft/greenplum-db-5.11.1/bin/postgres -D /greenplum/data/gpmaster/gpseg-1 -p 5432 --gp_dbid=1 --gp_num_contents_in_clu
ster=48 --silent-mode=true -i -M master --gp_contentid=1 -X 0 -E
root 110949 48832 0 11:31 pts/3 00:00:00 sh cmd.sh all ps -ef|grep greenplum
root 110952 110949 0 11:31 pts/3 00:00:00 ssh gpmdw -C ps -ef|grep greenplum
root 110955 110953 0 11:31 ? 00:00:00 bash -c ps -ef|grep greenplum
root 110962 110955 0 11:31 ? 00:00:00 grep greenplum
gpadmin 270028 1 7 Apr17 ? 3-00:30:17 /greenplum/soft/greenplum-cc-web-4.4.2/bin/gpccws
gpadmin 270405 1 5 Apr17 ? 2-11:22:15 /greenplum/soft/greenplum-cc-web-4.4.2/bin/ccagent -udport 9898 -rpcaddr gpmdw:8899
gpadmin 270750 1 0 Apr17 ? 01:11:37 /greenplum/soft/greenplum-db/.bin/gpsmon -m 0 -t 150 -l /greenplum/data/gpmaster/gpseg-1/gpperfmon/logs -v 0 8888
gpadmin 623354 1 0 May21 ? 00:00:00 python /greenplum/soft/greenplum-db/.bin/gpsh -f seg_host F=gpnetbenchServer && (pkill &F || pkill -f &F || killall -9 &F) > /d
ev/null 2>&1 || true
===== exec ps -ef|grep greenplum for gpmdw end =====

===== exec ps -ef|grep greenplum for gpssdw start =====
root 2128 1 0 2018 ? 00:25:00 /usr/sbin/glusterfd -s gpssdw --volfile-id datapool.gpssdw.greenplum-brick1-data -p /var/run/gluster/vols/datapool/gpssdw-greenplu
m-brick1-data.pid -S /var/run/gluster/98265ea252c1cb765b2f69f7d4a43fc.socket --brick-name /greenplum/brick1/data -l /var/log/glusterfs/bricks/greenplum-brick1-data.log --xlator-o
ption *posix.glusterd-uuid=f72385c4-169e-4b05-52d0-4be907923648 --brick-port 49152 --xlator-option datapool-server.listen-port=49152
gpadmin 346530 1 8 Apr17 ? 3-12:38:03 /greenplum/soft/greenplum-cc-web-4.4.2/bin/ccagent -udport 9898 -rpcaddr gpmdw:8899
gpadmin 349189 1 0 Apr17 ? 04:08:11 /greenplum/soft/greenplum-db/.bin/gpsmon -m 0 -t 150 -l /greenplum/data/gpdatap9/gpseg24/gpperfmon -v 0 8888
gpadmin 438424 1 0 Mar11 ? 00:38:41 /greenplum/soft/greenplum-db-5.11.1/bin/postgres -D /greenplum/data/gpdatap3/gpseg2 -p 40002 --gp_dbid=4 --gp_num_contents_in_clu
ster=48 --silent-mode=true -i -M quiescent --gp_contentid=2
```

在以上可以看出集群链接的数量以及每个 segment 的端口和 gpcc 提供服务的端口等信息，需要仔细的查看一下

4 查看集群的基本信息

4.1 查看集群的版本信息

```
psql -d staging -c "select version()"
```

```
root@gpmdw /home/xiaoxi# psql -d staging -c "select version()"
version
-----
PostgreSQL 8.3.23 (Greenplum Database 5.11.1 build commit:540866674a4dd1ef5f19c2670d020e3363699dfd2) on x86_64-pc-linux-gnu, compiled by GCC gcc (GCC) 6.2.0, 64-bit compiled on Se
p 20 2018 18:12:07
(1 row)
```

在以上可以看出 greenplum 集群使用的是 PostgreSQL 8.3.23，GCC gcc (GCC) 6.2.0 版本

4.2 查看 segment 相关信息

4.2.1 查看当前 down 的 segment 节点

```
$ psql -d staging -c "select * from gp_segment_configuration where status='d';"
```

```

dbid | content | role | preferred_role | mode | status | port | hostname | address |
replication_port
-----+-----+-----+-----+-----+-----+-----+-----+-----
(0 rows)

```

如果没有数据表示没有宕机的 segment

或使用以下 SQL 语句

```

SELECT * FROM gp_segment_configuration
WHERE status <> 'u';

```

4.2.2 查看当前处于 change tracking 的 segment 节点

```

SELECT * FROM gp_segment_configuration WHERE mode = 'c';

```

如果有记录返回，表示有处于 change tracking 的 segment。

4.2.3 查看当前处于 re-syncing 状态的 segment 节点

```

SELECT * FROM gp_segment_configuration WHERE mode = 'r';

```

如果有记录返回，表示有处于 re-syncing 的 segment

4.2.4 查看所有 segment 是否可达,确保 QD(query dispatching) 正常

```

SELECT gp_segment_id, count(*) FROM gp_dist_random('pg_class') GROUP BY 1;

```

正常情况下，每个节点返回一条记录，如果执行失败，表示有不可达的 segment，执行 SQL 是 QD 阶段会失败。

4.3 查看 standby 与 master 的信息

4.3.1 查看 standby 的配置

```

$ gpstate -f

```



```
[gpadmin@gpdev152 ~]$ gpstate -f
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-Starting gpstate with args: -f
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-local Greenplum Version: 'postgres (Greenplum Database) 5.11.1 bu:
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-master Greenplum Version: 'PostgreSQL 8.3.23 (Greenplum Database)
dfdd2 on x86_64-pc-linux-gnu, compiled by GCC gcc (GCC) 6.2.0, 64-bit compiled on Sep 20 2018 18:12:07'
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-Obtaining Segment details from master...
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-Standby master details
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-----
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:- Standby address          = gpdev153
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:- Standby data directory    = /data/gpmaster/gpseg-1
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:- Standby port              = 5432
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:- Standby PID               = 43194
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:- Standby status            = Standby host passive
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-----
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-pg_stat_replication
20190530:14:04:36:048941 gpstate:gpdev152:gpadmin-[INFO]:-----
20190530:14:04:37:048941 gpstate:gpdev152:gpadmin-[INFO]:-WAL Sender State: streaming
20190530:14:04:37:048941 gpstate:gpdev152:gpadmin-[INFO]:-Sync state: sync
20190530:14:04:37:048941 gpstate:gpdev152:gpadmin-[INFO]:-Sent Location: 2/5A4B6820
20190530:14:04:37:048941 gpstate:gpdev152:gpadmin-[INFO]:-Flush Location: 2/5A4B6820
20190530:14:04:37:048941 gpstate:gpdev152:gpadmin-[INFO]:-Replay Location: 2/5A4AD280
20190530:14:04:37:048941 gpstate:gpdev152:gpadmin-[INFO]:-----
```

在以上可以看出备用 master 的信息是 gpdev153 上数据目录在 /data/gpmaster/gpseg-1

4.3.2 查看 standby 的运行状态

```
SELECT procpid, state FROM pg_stat_replication;
```

```
1 SELECT procpid, state FROM pg_stat_replication;
```

信息	结果1
procpid	state
58117	streaming

如果 state 不是 'STREAMING', 或者没有记录返回, 那么说明 master standby 节点异常。

或使用 `select * from pg_stat_replication` 查看 standby 的详细信息

4.3.3 查看 master 节点的是否正常

```
SELECT count(*) FROM gp_segment_configuration;
```

QUERY 正常返回, 表示 master 节点正常。

4.3.4 检查 gp_persistent 表,确保主备 Segment 之间是否有数据不一致的问题

如果在有 standby 的情况下执行以下语句有结果说明有不同步的情况

```
SELECT p.tablespace_oid,p.relfilenode_oid,p.segment_file_num,
case  when p.persistent_state=0 then ' free'
when p.persistent_state=1 then 'create pending'
when p.persistent_state=2 then 'created'
when p.persistent_state=3 then 'drop pending'
when p.persistent_state=4 then 'abort create'
when p.persistent_state=5 then 'JIT create pending'
else 'unknown state:' || p.persistent_state end as persistent_state,
case when p.mirror_existence_state=0 then 'mirror free'
when p.mirror_existence_state=1 then 'not mirrored'
when p.mirror_existence_state=2 then 'mirror create pending'
when p.mirror_existence_state=3 then 'mirror created'
when p.mirror_existence_state=4 then 'mirror down before create'
when p.mirror_existence_state=5 then 'mirror down during create'
when p.mirror_existence_state=6 then 'mirror drop pending'
when p.mirror_existence_state=7 then 'mirror only drop remains'
else 'unknown state:' || p.mirror_existence_state end as mirror_existence_state
FROM gp_persistent_relation_node p WHERE (p.persistent_state not in (0,2)
or p.mirror_existence_state not in (0,1,3))
and p.database_oid in(
SELECT oid FROM pg_database WHERE datname=current_database()
)
```

4.4 查看有问题的 segment primary/mirror 信息

\$ gpstate -e

```
[gpadmin@gpmdw /home/xiaoxu]$ gpstate -e
20190530:14:07:39:142486 gpstate:gpmdw:gpadmin-[INFO]:--Starting gpstate with args: -e
20190530:14:07:39:142486 gpstate:gpmdw:gpadmin-[INFO]:--local Greenplum Version: 'postgres (Greenplum Database)
20190530:14:07:39:142486 gpstate:gpmdw:gpadmin-[INFO]:--master Greenplum Version: 'PostgreSQL 8.3.23 (Greenplum
2) on x86_64-pc-linux-gnu, compiled by GCC gcc (GCC) 6.2.0, 64-bit compiled on Sep 20 2018 18:12:07'
20190530:14:07:39:142486 gpstate:gpmdw:gpadmin-[INFO]:--Obtaining Segment details from master...
20190530:14:07:39:142486 gpstate:gpmdw:gpadmin-[INFO]:--Gathering data from segments...
.....
20190530:14:07:54:142486 gpstate:gpmdw:gpadmin-[INFO]:-----
20190530:14:07:54:142486 gpstate:gpmdw:gpadmin-[INFO]:--Segment Mirroring Status Report
20190530:14:07:54:142486 gpstate:gpmdw:gpadmin-[INFO]:-----
20190530:14:07:54:142486 gpstate:gpmdw:gpadmin-[INFO]:--All segments are running normally
```

当看到 All segments are running normally 时表示所有的节点都没有问题

4.5 显示 mirror 列表

```
$ gpstate -m
```

```
[gpadmin@gpmdw /home/xiaoxu]$ gpstate -m
20190530:14:09:36:142842 gpstate:gpmdw:gpadmin-[INFO]:-Starting gpstate with args: -m
20190530:14:09:36:142842 gpstate:gpmdw:gpadmin-[INFO]:-local Greenplum Version: 'postgres (Greenplum Database) 5.11.1 build commit:54f
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:-master Greenplum Version: 'PostgreSQL 8.3.23 (Greenplum Database 5.11.1 build
2) on x86_64-pc-linux-gnu, compiled by GCC gcc (GCC) 6.2.0, 64-bit compiled on Sep 20 2018 18:12:07'
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:-Obtaining Segment details from master...
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:-----
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:--Current GPDB mirror list and status
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:--Type = Group
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:-----
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:- Mirror      Datadir      Port      Status      Data Status
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:- gpdw2      /greenplum/data/gpdataml/gpseg0  50000     Passive     Synchronized
20190530:14:09:37:142842 gpstate:gpmdw:gpadmin-[INFO]:- gpdw2      /greenplum/data/gpdatam2/gpseg1  50001     Passive     Synchronized
```

在以上可以所有的 Mirror 所有的配置信息

4.6 显示所有配置参数

```
$ psql -d staging -c "show all;"
```

```
[gpadmin@gpmdw /home/xiaoxu]$ psql -d staging -c "show all;"
name                                | description                                | setting
-----+-----+-----
add_missing_from                    |                                             | off
ces to FROM clauses.
application_name                     |                                             | psql
in statistics and logs.
archive_mode                         |                                             | off
ive_command.
array_nulls                         |                                             | on
authentication_timeout              |                                             | 1min
e client authentication.
autovacuum_max_workers               |                                             | 3
y running autovacuum worker processes.
autovacuum_max_workers               |                                             | 3
y running autovacuum worker processes.
backslash_quote                     |                                             | safe_encoding
iterals.
block_size                           |                                             | 32768
bonjour_name                         |                                             |
bytea_output                         |                                             | escape
check_function_bodies                |                                             | on
```

以上全部列出了集群中配置信息，请仔细查看选项

4.7 查看集群中数据库中的详细信息

4.7.1 查看每个数据库占用的大小

```
SELECT d.datname as "Name",
pg_catalog.pg_encoding_to_char(d.encoding) as "Encoding",
```

```

CASE WHEN pg_catalog.has_database_privilege(d.datname, 'CONNECT')
THEN pg_catalog.pg_size_pretty(pg_catalog.pg_database_size(d.datname))
ELSE 'No Access' END as "Size",
t.spcname as "Tablespace",
pg_catalog.shobj_description(d.oid, 'pg_database') as "Description"
FROM pg_catalog.pg_database d
JOIN pg_catalog.pg_tablespace t on d.dattablespace = t.oid
ORDER BY 1;

```

查询创建工具

查询编辑器

```

2 pg_catalog.pg_encoding_to_char(d.encoding) as "Encoding",
3 CASE WHEN pg_catalog.has_database_privilege(d.datname, 'CONNECT')
4 THEN pg_catalog.pg_size_pretty(pg_catalog.pg_database_size(d.datname))
5 ELSE 'No Access' END as "Size",
6 t.spcname as "Tablespace",
7 pg_catalog.shobj_description(d.oid, 'pg_database') as "Description"
8 FROM pg_catalog.pg_database d
9 JOIN pg_catalog.pg_tablespace t on d.dattablespace = t.oid
10 ORDER BY 1;

```

信息

结果1

Name	Encoding	Size	Tablespace	Description
postgres	UTF8	6601 GB	pg_default	(Null)
gpperfmon	UTF8	12 GB	pg_default	(Null)
postgres	UTF8	525 MB	pg_default	(Null)
staging	UTF8	65 GB	pg_default	(Null)
template0	UTF8	505 MB	pg_default	(Null)
template1	UTF8	510 MB	pg_default	default template database

4.7.2 查看每个 schema 的占用大小

```

select pg_size_pretty(cast(sum(pg_relation_size( schemaname || '.' || tablename)) as bigint)),
schemaname from pg_tables t inner join pg_namespace d on t.schemaname=d.nspname
group by schemaname ORDER BY pg_size_pretty desc;

```

```

1 select pg_size_pretty(cast(sum(pg_relation_size( schemaname || '.' || tablename)) as bigint)), schemaname
2 from pg_tables t inner join pg_namespace d on t.schemaname=d.nspname
3 group by schemaname ORDER BY pg_size_pretty desc;

```

pg_size_pretty	schemaname
9594 MB	ods_spider
839 GB	datafix
792 GB	main
781 GB	ods
73 MB	public
738 GB	produce_check
4608 kB	pg_bitmapindex
27 GB	dim
260 GB	pg_catalog

该 SQL 比较耗时，会查询每个表的大小进行合并，可以详细的查看出数据库下每个 schema 的大小

4.7.3 查看 AO 表的相关信息

4.7.3.1 查看数据库中的 AO 表

```

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

```

4.7.3.2 查看数据库中的 AO 表的数量

```

select count(*) from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('c', 'a');

```

4.7.4 查看堆表的相关信息

4.7.4.1 查看数据库中的堆表

```

select t2.nspname, t1.relname from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('h') and relkind='r';

```

4.7.4.2 查看堆表的数量

```

select count(*) from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('h') and relkind='r';

```

4.7.5 查看外部表相关信息

4.7.5.1 查看外部表

```
select t2.nspname, t1.relname from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('x') and relkind='r';
```

4.7.5.2 查看外部表的数量

```
select count(*) from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('x') and relkind='r';
```

4.7.6 查看视图的相关信息

4.7.6.1 查看制定 schema 下视图

```
select t2.nspname, t1.relname from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('v')
and nspname in ('schema1','schema2',[schema3,schema4]);
```

4.7.6.2 查看制定 schema 下视图的数量

```
select count(*) from pg_class t1, pg_namespace t2
where t1.relnamespace=t2.oid and relstorage in ('v')
and nspname in ('schema1','schema2',[schema3,schema4]);
```

4.7.7 查看表的相关信息

4.7.7.1 查看每个表的大小信息

以下 SQL 会比较耗时，请在查询时注意时间的分配

```
SELECT t2.nspname, relname as name, sotdsize/1024/1024 as size_MB, sotdtoastsize as toast,
sotdadditionalsize as other
FROM gp_toolkit.gp_size_of_table_disk as sotd, pg_class, pg_namespace t2
WHERE sotd.sotdoid = pg_class.oid
and t2.nspname in ('schema1'[schema2,schema3])
```


4.8 查看索引的相关信息

4.8.1 查看索引大小超过表总大小 1/2 的系统表大小和索引大小

```
select indrelid::regclass tab,
pg_size_pretty(avg(pg_total_relation_size(indrelid))::bigint) all_size,
case when avg(pg_relation_size(indrelid))+ sum(pg_total_relation_size(indexrelid)) = avg
(pg_total_relation_size(indrelid)) then 't' else 'f' end,
pg_size_pretty(avg(pg_relation_size(indrelid))::bigint) tab_size,
pg_size_pretty(sum(pg_total_relation_size(indexrelid))::bigint) all_idx_size
from pg_index
where indrelid in (select oid from pg_class where relnamespace in (select oid from
pg_namespace where
nspname='pg_catalog'))
group by indrelid::regclass
having sum(pg_total_relation_size(indexrelid))::bigint >=
avg(pg_total_relation_size(indrelid))::bigint * 0.5
order by sum(pg_total_relation_size(indexrelid)) desc;
```

```
1  select indrelid::regclass tab,
2  pg_size_pretty(avg(pg_total_relation_size(indrelid))::bigint) all_size,
3  case when avg(pg_relation_size(indrelid))+ sum(pg_total_relation_size(indrelid)) = avg
4  (pg_total_relation_size(indrelid)) then 't' else 'f' end,
5  pg_size_pretty(avg(pg_relation_size(indrelid))::bigint) tab_size,
6  pg_size_pretty(sum(pg_total_relation_size(indexrelid))::bigint) all_idx_size,
7  from pg_index
8  where indrelid in (select oid from pg_class where relnamespace in (select
9  nspname='pg_catalog'))
10 group by indrelid::regclass
11 having sum(pg_total_relation_size(indexrelid))::bigint >= avg(pg_total_relation_size(indrelid))::bigint * 0.5
12 order by sum(pg_total_relation_size(indexrelid)) desc;
```

tab	all_size	case	tab_size	all_idx_size
	39 GB	t	18 GB	22 GB
	543 MB	t	213 MB	330 MB
	55 MB	t	15 MB	40 MB
	18 MB	f	4736 kB	12 MB
	15 MB	f	1568 kB	12 MB
	10 MB	t	832 kB	9472 kB
	11 MB	t	1568 kB	9408 kB

在以上可以详细的看到索引大小超过表总大小 1/2 的系统表大小和索引大小

4.8.2 查看制定 schema 上表的索引

```
select * from pg_stat_all_indexes where schemaname in ('schema1',[schema2,schema3]);
```

4.8.3 索引活跃度监控

详细的参数请查看:

<https://www.postgresql.org/docs/10/monitoring-stats.html>

4.8.3.1 IO 活跃度查看

```
select * from pg_catalog.pg_statio_all_indexes where schemaname in ('ods','main');
```

参数	类型	描述
relid	oid	此索引的表的 OID
indexrelid	oid	该指数的 OID
schemaname	name	此索引所在的架构的名称
relname	name	此索引的表的名称
indexrelname	name	该索引的名称
idx_blks_read	bigint	从此索引读取的磁盘块数
idx_blks_hit	bigint	此索引中的缓冲区命中数

1、使用较少、较多的索引 (idx_blks_read + idx_blks_hit)

2、消耗物理 IO 较多、较少的索引 (idx_blks_read)

4.8.3.2 访问活跃度

```
select * from pg_catalog.pg_stat_all_indexes where schemaname in ('ods','main');
```

参数	类型	描述
relid	oid	此索引的表的 OID
indexrelid	oid	该指数的 OID
schemaname	name	此索引所在的架构的名称
relname	name	此索引的表的名称

indexrelname	name	该索引的名称
idx_scan	bigint	在此索引上启动的索引扫描数
idx_tup_read	bigint	扫描在此索引上返回的索引条目数
idx_tup_fetch	bigint	使用此索引通过简单索引扫描获取的活动表行数

- 1、使用较少、较多的索引（idx_scan）
- 2、扫描较少、较多 index item 的索引（idx_tup_read）
- 3、从索引读取 HEAP TUPLE 较多、较少的索引（idx_tup_fetch）

4.8.4 查看 Master 与 Segment 上索引不一致的问题

以下语句只要有数据输出表示有的表索引不一致，请进一步查看

```
select distinct n.nspname,c.relname from gp_dist_random('pg_class') r ,pg_class
c,pg_namespace n
where r.oid=c.oid and r.relhasindex<>c.relhasindex
and c.relnamespace = n.oid limit 10
```

4.8.5 查看索引的使用次数

```
select
    relname, indexrelname, idx_scan, idx_tup_read, idx_tup_fetch
from
    pg_stat_user_indexes
order by
    idx_scan asc, idx_tup_read asc, idx_tup_fetch asc;
```

4.9 检查是否使用了 a-z 0-9 _ 以外的字母作为对象名

4.9.1 查看数据库是否使用了命名规范

```
select distinct datname from (select datname,regexp_split_to_table(datname,$$$)
word from pg_database) t where (not (ascii(word) >=97 and ascii(word) <=122)) and (not
(ascii(word) >=48 and
ascii(word) <=57)) and ascii(word)<>95
```

4.9.2 查看表的索引和视图的命名规范

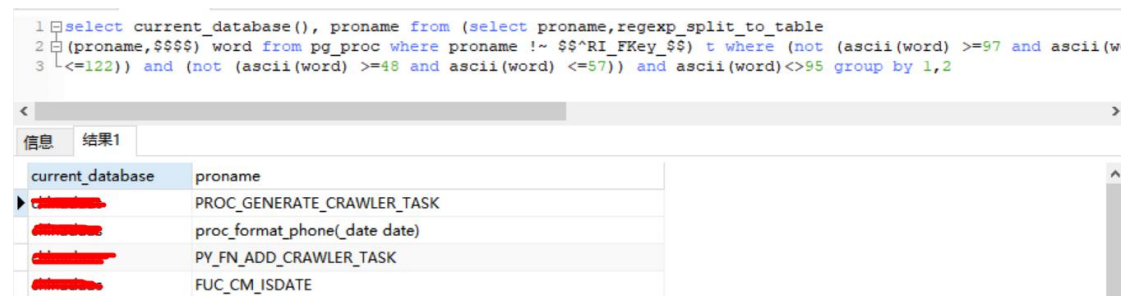
```
select current_database(),relname,relkind from (select relname,relkind,
regexp_split_to_table(relname,$$$$) word from pg_class) t
where (not (ascii(word) >=97 and ascii(word) <=122))
and (not (ascii(word) >=48 and ascii(word) <=57)) and ascii(word)<>95 group by 1,2,3
```

4.9.3 查看数据库中的类型命名规范

```
select current_database(), typename from (select typename,regexp_split_to_table
(typename,$$$$) word from pg_type) t
where (not (ascii(word) >=97 and ascii(word) <=122)) and (not (ascii(word)
>=48 and ascii(word) <=57)) and ascii(word)<>95 group by 1,2
```

4.9.4 查看数据库中的储存过程的命名规范

```
select current_database(), proname from (select proname,regexp_split_to_table
(proname,$$$$) word from pg_proc where proname !~ $$^RI_FKey_$$) t where (not
(ascii(word) >=97 and ascii(word)
<=122)) and (not (ascii(word) >=48 and ascii(word) <=57)) and ascii(word)<>95 group by 1,2
```



current_database	proname
postgres	PROC_GENERATE_CRAWLER_TASK
postgres	proc_format_phone(date date)
postgres	PY_FN_ADD_CRAWLER_TASK
postgres	FUC_CM_ISDATE

在以上可以看出储存过程使用了不规范命名,在 database 和 schema 和 table 以及 FUNCTION 都不建议使用大写, 如果大写使用时需要加双引使用

4.9.5 查看数据库中的表,视图的的命名规范

```
select current_database(),nspname,relname,attname from (select nspname,relname,
attname,regexp_split_to_table(attname,$$$$) word from pg_class a,pg_attribute
b,pg_namespace c where a.oid=b.
attrelid and a.relnamespace=c.oid ) t where (not (ascii(word) >=97 and ascii(word) <=122)) and
(not (ascii
```

(word) >=48 and ascii(word) <=57)) and ascii(word)<>95 group by 1,2,3,4

```
1 select current_database(),nspname,relname,attname from (select nspname,relname,
2 attname,regexp_split_to_table(attname,$$$) word from pg_class a,pg_attribute b,pg_namespace c where a.oi
3 attrelid and a.relnamespace=c.oid ) t where (not (ascii(word) >=97 and ascii(word) <=122)) and (not (asci
4 (word) >=48 and ascii(word) <=57)) and ascii(word)<>95 group by 1,2,3,4
```

<

信息 结果1

current_database	nspname	relname	attname
.....	ods	t_ent_zl_rctinfo_20190527pg.dropped.138.....
.....	ods	t_ju_lesscreditimp_pg.dropped.33.....
.....	ods	t_ent_trademarkpic_20190513pg.dropped.74.....

出现.....pg.dropped.138.....问题请查看:

https://www.postgresql.org/message-id/CAFj8pRDUSTNoX8zJSMLy%3Dr45z_Tq-OUwd5S7_HG0wtxHZN-yAg%40mail.gmail.com

4.10 查看集群是否处于 not balanced 状态

4.10.1 查看当前的连接数

```
select * from pg_stat_activity;
```

或使用以下 SQL

```
select current_query,count(*) from pg_stat_activity group by current_query ORDER by count desc;
```

4.10.2 查看密码有效期不足 30 天的用户

```
select rolname,rolvaliduntil from pg_authid
where rolvaliduntil-now()<'30 days' order by rolvaliduntil;
```

```
1 select rolname,rolvaliduntil from pg_authid
2 where rolvaliduntil-now()<'30 days' order by rolvaliduntil;
```

信息	结果1
rolname	rolvaliduntil
(Null)	(Null)

4.10.3 查看每个用户链接的个数

```
select username,count(*) from pg_stat_activity group by 1;
```

```
2
3  select username,count(*) from pg_stat_activity group by 1;
4
```

信息	结果1	结果2
username	count	
gpadmin	125	
gpadmin	8	
spider	10	

4.10.4 查看数据库的连接数

```
select datname, count(*) from pg_stat_activity group by 1;
```

4.11 查看 ctid 的值

先查看当前最大的 ctid 的值

```
select max(ctid) from pg_class;
-- (19122,123)
```

创建一张新表

```
create table xiaoxu_a(id int);
create table xiaoxu_b(id int);
```

再次查看 ctid 的值

```
select ctid,relname from pg_class where relname like 'xiaoxu%';
```

```
9  select ctid,relname from pg_class where relname like 'xiaoxu%';
10
```

信息	结果1
ctid	relname
(19122,124)	xiaoxu_a
▶ (19122,125)	xiaoxu_b

在以上可以看出 `ctid` 是递增的，说明当前的数据库没有执行 `vacuum`，可以查找出膨胀比较大的表进行 `vacuum table` 或者 `vacuum pg_class, vacuum pg_class` 需要谨慎，首先需要查看表的个数 `select count(*) from pg_class;` 在进行操作。

5 集群巡检过程详细信息

5.1 检查大小超过 1GB 的表倾斜情况

问题描述：GPDB 为分布式数据库，所有表数据应该均匀分布到所有 `segment` 实例中存储，才能发挥数据库最佳的性能。数据倾斜是指表的数据没有均匀分布到所有 `segment` 实例中，部分实例存储数据量大，其他实例存储数据量小，甚至没有数据。数据查询过程中出现部分实例繁忙，其他实例空闲，导致数据库性能不均衡。

建议在创建表时使用唯一键作为分布键，例如使用 `uuid` 作为分布键做到表分布均匀，代码请查看：<https://blog.csdn.net/xfg0218/article/details/90699970>

5.1.1 查看超过 1GB 倾斜率的表

```
SELECT
'select ''' || schemaname || '.' || tablename || ''' tablename' ||
E' ,pg_size_pretty(sum(pg_relation_size(\' ' || schemaname || '.' ||
tablename || E'\'))::bigint) as
sum_relation_size' ||
E' ,pg_size_pretty(max(pg_relation_size(\' ' || schemaname || '.' ||
tablename || E'\')) as max_relation_size'
||
E' ,pg_size_pretty(min(pg_relation_size(\' ' || schemaname || '.' ||
tablename || E'\')) as min_relation_size' ||
E' ,max(pg_relation_size(\' ' || schemaname || '.' || tablename ||
E'\'))::numeric(100,1)/min(pg_relation_size
(\' ' || schemaname || '.' || tablename || E'\')) skew_info ' || E' from
gp_dist_random(\' gp_id\') ' || E' where
pg_relation_size(\' ' || schemaname || '.' || tablename || E'\') <> 0;' as
question_table_sql
FROM
pg_tables
WHERE
pg_relation_size(tablename) > 1.0 * 1024 * 1024 * 1024
and schemaname not in ('information_schema','pg_catalog')
```

```
ORDER BY
pg_relation_size(tablename)
DESC;
```

5.1.2 查看表在 segment 上占用大小及倾斜率

运行以上的 SQL 会把超过 1GB 倾斜的表的 SQL 列出来, 例如以下 SQL:

```
select 'datafix.lyj_main_ent'
tabname ,pg_size_pretty(sum(pg_relation_size('datafix.lyj_main_ent')):
:bigint) as
sum_relation_size ,pg_size_pretty(max(pg_relation_size('datafix.lyj_m
ain_ent')) as
max_relation_size ,pg_size_pretty(min(pg_relation_size('datafix.lyj_m
ain_ent')) as
min_relation_size ,max(pg_relation_size('datafix.lyj_main_ent'))::num
eric(100,1)/min(pg_relation_size
('datafix.lyj_main_ent')) skew_info from gp_dist_random('gp_id')
where
pg_relation_size('datafix.lyj_main_ent' )>>0;
```

```
1 select 'datafix.lyj_main_ent' tabname ,pg_size_pretty(sum(pg_relation_size('datafix.lyj_main_ent'))::bi
2 sum_relation_size ,pg_size_pretty(max(pg_relation_size('datafix.lyj_main_ent')) as max_relation_size ,
3 ('datafix.lyj_main_ent')) skew_info from gp_dist_random('gp_id') where
4 pg_relation_size('datafix.lyj_main_ent' )>>0;
```

tabname	sum_relation_size	max_relation_size	min_relation_size	skew_info
datafix.lyj_main_ent	41 GB	872 MB	868 MB	1.00494002197486

在以上可以看出列出了当前表的详细信息,其中字段的详细信息为:

tabname: 当前表的名字

sum_relation_size: 表的大小

max_relation_size: segment 上最大的占用空间

min_relation_size: segment 上最小的占用空间

skew_info: 倾斜率

5.1.3 查看表在 segment 上占用的行数及百分比

```
SELECT 'Example Table' AS "Table Name",
max(c) AS "Max Seg Rows", min(c) AS "Min Seg Rows",
(max(c)-min(c))*100.0/max(c) AS "Percentage Difference Between Max & Min"
FROM (SELECT count(*) c, gp_segment_id FROM datafix.lyj_main_ent GROUP BY 2) AS a;
```

```
1 SELECT 'Example Table' AS "Table Name",
2 max(c) AS "Max Seg Rows", min(c) AS "Min Seg Rows",
3 (max(c)-min(c))*100.0/max(c) AS "Percentage Difference Between Max & Min"
4 FROM (SELECT count(*) c, gp_segment_id FROM datafix.lyj_main_ent GROUP BY 2) AS a;
5
```

信息	结果1		
Table Name	Max Seg Rows	Min Seg Rows	Percentage Difference Be
▶ Example Table	1240383	1232367	0.646252004421215

Max Seg Rows：单个 segment 实例存储占用空间（最大）

Min Seg Rows：单个 segment 实例存储占用空间（最小）

Percentage Difference Between Max & Min :最大值和最小值的百分比差异

5.1.4 解决表分布倾斜的情况

5.1.4.1 分布键说明

对以上查询出来的倾斜比较大的表可以使用修改表分布键的形式是数据分布均匀, 常见的分布方式有:

- 1、指定一个字段作为分布键:distributed by(filed1)
- 2、指定多个字段作为分布键:distributed by(filed1,filed2)
- 3、随机字段分布键:distributed randomly;

建议使用制定一个字段作为分布键, 因为多个字段和随机字段做分布键都可能影响性能的下降,详细创建表的方式请查看:

<https://blog.csdn.net/xfg0218/article/details/86473234>

5.1.4.2 修改分布键

修改一个或多个字段作为分布键

```
alter table tablename set with (reorganize=true) distributed by
(filed1[filed2,filed3]) ;
```

修改随机字段作为分布键

```
alter table tablename set with (reorganize=true) distributed randomly ;
```

5.1.4.3 对表进行重新创建

在原表的基础上创建一张压缩表

```
create table newtablename with (appendonly = true, compresstype = zlib,
```

```
compresslevel = 7
,orientation=column, checksum = false,blocksize = 2097152) as
select * from tablename
Distributed by (filed1[filed2]);
```

获取原始表的权限

```
select 'grant ' || privilege_type || ' on tablename to ' || grantee || ';' from
information_schema.table_privileges
where table_schema='schema' and table_name='tablename'
group by grantee, privilege_type;
```

重命名原表的名字

```
alter table tablename rename to newtablename;
```

5.2 检查膨胀率过高的表

Greenplum 支持行储存(HEAP 储存)与列(append-only)储存,对于 AO 储存,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 BITMAP 来标记删除与修改。AO 储存是块级组织,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。

详细介绍请查看:

<https://blog.csdn.net/xfg0218/article/details/83031550>

shell 脚本请下载:

<https://github.com/xfg0218/greenplum--summarize/tree/master/201905/greenplum-inspect-ao>

5.2.1 查看 schema 下的 AO 表

```
SELECT ns.nspname,pc.relname FROM pg_class pc,pg_namespace ns
WHERE pc.relnamespace = ns.oid AND relstorage IN ('c', 'a')
AND ns.nspname = 'dim';
```


5.2.2 查看表的膨胀率

```
select percent_hidden
from gp_toolkit.__gp_aovisimap_compaction_info('dim.t_dex_app_codelist_mapping'::regclass)
ORDER BY percent_hidden desc;
```



5.2.3 对表进行释放空间

使用 `vacuum tablename` 即可清理表空间

```
VACUUM dim.t_dex_app_codelist_mapping;
```

5.3 检查元数据不一致问题

5.3.1 问题描述

GPDB 元数据就是系统表所存的数据，也叫数据字典。GP 可以看做一组 PostgreSQL 实例组成的阵列，其每一个实例都存在系统表，这些实例的大部分系统表存储的数据应该都是相同的；并且在每个实例中，不同的系统表之间也应存在这严格的外键约束。

元数据不一致就是，以上的情况出现了错误，同一张系统表在不同实例中存储的数据存在缺失、多余；或者同一个实例中，不同的系统表外键约束出现错误；该错误原因大都是由于系统宕机，实例故障引起。

该系统存在"Found a total of 130554 issue(s)", 其中大部分异常信息如下：

```
"No pg_class entry for gp_distribution_policy {'localoid': 12163942} on
master (datacenter-mdw:5432)"
```

5.3.2 集群检查

```
$ psql -d staging -c " select count(0) from gp_distribution_policy where localoid not in (select  
oid from pg_class);"
```

```
count  
-----  
0  
(1 row)
```

Found a total of 130554 issue(s)

No pg_class entry for gp_distribution_policy {'localoid': 12163942} on
master (gpmdw:5432)

通过日志可以看出该系统存在"Found a total of 130554 issue(s)", 其中大部分异常信息如下:
"No pg_class entry for gp_distribution_policy {'localoid': 12163942} on master
(datacenter-mdw:5432)", 经过分析检查确认, 为 master 实例 gp_distribution_policy 和
pg_class 的外键约束出现大量错误数据

5.3.3 问题处理

目前暂不影响使用,但是存在安全隐患,元数据异常很可能会导致部分表无法正常访问,
严重可能导致实例宕机、数据库启动失败、segment 实例恢复失败等情况。

该问题需要每一项错误进行逐一排查,清理失效的元数据,修复错误;例如本次 master
外键故障,可以删除 gp_distribution_policy 表中多余的数据。另外,元数据修复需要谨慎
处理,处理前需要多次测试,或者由专业工程师操作。

5.4 查看服务器配置参数

数据库参数 gp_vmem_protect_limit 控制每个数据库 segment 实例可使用的内存数
量,单位 MB

5.4.1 服务器 OS 参数查看

查看主机映射文件

```
$ cat seg_host  
gpmdw  
gpsdw1  
gpsdw2
```

gpsdw3

使用 gpcheck -f seg_host

```
20190531:15:50:30:424288 gpcheck:gpmdw:gpadmin-[INFO]:--dedupe hostnames
20190531:15:50:30:424288 gpcheck:gpmdw:gpadmin-[INFO]:--Detected platform: Generic Linux Cluster
20190531:15:50:30:424288 gpcheck:gpmdw:gpadmin-[INFO]:--generate data on servers
20190531:15:50:30:424288 gpcheck:gpmdw:gpadmin-[INFO]:--copy data files from servers
20190531:15:50:31:424288 gpcheck:gpmdw:gpadmin-[INFO]:--delete remote tmp files
20190531:15:50:31:424288 gpcheck:gpmdw:gpadmin-[INFO]:--Using gpcheck config file: /greenplum/soft/greenplum-db/.etc/gpcheck.cnf
20190531:15:50:31:424288 gpcheck:gpmdw:gpadmin-[ERROR]:-GPCHECK_ERROR host(None): utility will not check all settings when run as non-root user
20190531:15:50:31:424288 gpcheck:gpmdw:gpadmin-[ERROR]:-GPCHECK_ERROR host(gpsdw3): on device (sr0) IO scheduler 'cfq' does not match expected value 'deadline'
20190531:15:50:31:424288 gpcheck:gpmdw:gpadmin-[ERROR]:-GPCHECK_ERROR host(gpsdw3): /etc/sysctl.conf value for key 'vm.overcommit_memory' has value '1' and expects '2'
20190531:15:50:31:424288 gpcheck:gpmdw:gpadmin-[ERROR]:-GPCHECK_ERROR host(gpsdw3): /etc/sysctl.conf value for key 'kernel.sem' has value '1000 10240000 400 10240' and expects '50
0 1024000 200 4096'
```

当出现以上的 ERROR 信息时请排查 OS 系统参数

5.5 查看集群参数详细信息

5.5.1 查看每个 segment 的内存配置参数

5.5.1.1 查看分配内存信息

gpconfig -s gp_vmem_protect_limit

```
[gpadmin@gpmdw /home/xiaoxu]$ gpconfig -s gp_vmem_protect_limit
Values on all segments are consistent
GUC          : gp_vmem_protect_limit
Master value: 8192
Segment value: 8192
```

在以上可以看出 segment 使用了系统默认的内存配置 8192MB, 改参数按照机器的内存大小可以适当的调大, 详见计算如下:

- 1、计算公式可参考如下: $(\text{mem} + \text{swap}) * 0.9 / \text{单个节点 segment 数量}$
- 2、例如 master 节点上有 252G 的内存, segment 个数为 2 个, 分配最高的内存为:
 $252 * 0.9 / 2 \approx 110\text{GB}$ (112640 MB)
- 3、例如数据节点上有 252G 的内存, segment 个数为 12 个, 分配最高的内存为:
 $252 * 0.9 / 12 \approx 18\text{GB}$ (18432MB)

5.5.1.2 修改内存参数

登录到 master 节点上执行以下命令即可

gpconfig -c gp_vmem_protect_limit -m 112640 -v 18432

- c : 改变参数的名称
- m : 修改主备 master 的内存的大小一般的和 -v 一块使用
- v : 此值用于所有的 segments, mirrors 和 master 的修改

5.5.2 查看 shared_buffers(共享缓冲区)的内存

5.5.2.1 查看系统配置的参数

```
$ gpconfig -s shared_buffers
```

```
[gpadmin@gpmdw /home/xiaoxu]$ gpconfig -s shared_buffers
Values on all segments are consistent
GUC          : shared_buffers
Master value: 125MB
Segment value: 125MB
```

5.5.2.2 参数详解

只能配置 segment 节点，用作磁盘读写的内存缓冲区,开始可以设置一个较小的值，比如总内存的 15%，然后逐渐增加，过程中监控性能提升和 swap 的情况。以上的缓冲区的参数为 125MB，此值不易设置过大，过大或导致以下错误

[WARNING]:-FATAL: DTM initialization: failure during startup recovery, retry failed, check segment status (cdbtm.c:1603)，详细的配置请查看

http://gpdb.docs.pivotal.io/4390/guc_config-shared_buffers.html

5.5.2.3 修改参数

修改配置

```
gpconfig -c shared_buffers -v 1024MB
```

```
gpconfig -r shared_buffers -v 1024MB
```

5.5.3 查看 max_connections(最大连接数)

5.5.3.1 查看最大连接数参数

```
$ gpconfig -s max_connections
```

```
[gpadmin@gpmdw /home/xiaoxu]$ gpconfig -s max_connections
Values on all segments are consistent
GUC          : max_connections
Master value: 500
Segment value: 1000
```

5.5.3.2 参数详解

此参数为客户端链接数据库的连接数，按照个人数据库需求配置，参数详解请查看：
https://gpdb.docs.pivotal.io/4380/guc_config-max_connections.html

5.5.4 查看 block_size(磁盘块)的大小

5.5.4.1 查看磁盘块的大小

```
$ gpconfig -s block_size
```

```
[gpadmin@gpmdw /home/xiaoxu]$ gpconfig -s block_size
Values on all segments are consistent
GUC          : block_size
Master value: 32768
Segment value: 32768
```

5.5.4.2 参数详解

此参数表示表中的数据以默认的参数 32768 KB 作为一个文件，参数的范围 8192KB - 2MB，范围在 8192 - 2097152，值必须是 8192 的倍数，使用时在 `blocksize = 2097152` 即可

5.5.5 查看 work_mem 的值

5.5.5.1 查看集群中 work_mem 的配置大小

```
$ gpconfig -s work_mem
```

```
[gpadmin@gpmdw /home/xiaoxu]$ gpconfig -s work_mem
Values on all segments are consistent
GUC          : work_mem
Master value: 32MB
Segment value: 32MB
```

5.5.5.2 参数详解

`work_mem` 在 segment 用作 sort,hash 操作的内存大小当 PostgreSQL 对大表进行排序时，数据库会按照此参数指定大小进行分片排序，将中间结果存放在临时文件中，这些中间结果的临时文件最终会再次合并排序，所以增加此参数可以减少临时文件个数进而提升排序效率。当然如果设置过大，会导致 swap 的发生，所以设置此参数时仍需谨慎。刚开始可设置总内存的 5%

5.5.5.3 修改参数

修改系统配置文件,重启集群使之生效

```
gpconfig -c work_mem -v 128MB
```

或在客户端 session 设置此参数

```
SET work_mem TO '64MB'
```

销毁 session 参数为:

```
reset work_mem;
```

5.5.6 查看 statement_mem 的值

5.5.6.1 查看集群中 statement_mem 的值

```
$ gpconfig -s statement_mem
```

```
[gpadmin@gpmdw /home/xiaoxu]$ gpconfig -s statement_mem
Values on all segments are consistent
GUC          : statement_mem
Master value: 125MB
Segment value: 125MB
```

5.5.6.2 参数详解

设置每个查询在 segment 主机中可用的内存，该参数设置的值不能超过 `max_statement_mem` 设置的值，如果配置了资源队列，则不能超过资源队列设置的值。

5.5.6.3 修改参数

修改配置后重启生效

```
gpconfig -c statement_mem -v 256MB
```

5.5.7 查看 gp_workfile_limit_files_per_query 的值

5.5.7.1 查看此值的大小

```
$ gpconfig -s gp_workfile_limit_files_per_query
```

```
[gpadmin@gpmdw /home/xiaoxu]$ gpconfig -s gp_workfile_limit_files_per_query
Values on all segments are consistent
GUC          : gp_workfile_limit_files_per_query
Master value: 100000
Segment value: 100000
```

5.5.7.1 参数详解

SQL 查询分配的内存不足，Greenplum 数据库会创建溢出文件（也叫工作文件）。在默认情况下，一个 SQL 查询最多可以创建 100000 个溢出文件，这足以满足大多数查询。该参数决定了一个查询最多可以创建多少个溢出文件。0 意味着没有限制。限制溢出文件数据可以防止失控查询破坏整个系统。

如果数据节点的内存是 512G 的内存,表的压缩快的大小(block_size)是 2M 的话，计算为: $512G + 2 * 1000000 / 1024 \approx 707 G$ 的空间，一般的表都是可以的，一般的此值不需要修改

5.5.8 查看 gp_resqueue_priority_cpucore_per_segment 的值

5.5.8.1 查看此值的大小

```
$ gpconfig -s gp_resqueue_priority_cpucore_per_segment
```

```
[gpadmin@gpmdw /home/xiaoxu]$ gpconfig -s gp_resqueue_priority_cpucore_per_segment
Values on all segments are consistent
GUC          : gp_resqueue_priority_cpucore_per_segment
Master value: 4
Segment value: 4
```

5.5.8.2 参数详解

每个 segment 分配的分配的 cpu 的个数，例如:在一个 20 核的机器上有 4 个 segment,则每个 segment 有 5 个核，而对于 master 节点则是 20 个核，master 节点上不运行 segment 的信息，

因此 master 反映了 cpu 的使用情况

5.5.8.3 修改参数

按照不同集群的核数以及 segment 修改此参数即可，下面的实例是修改成 8 核

```
gpconfig -c gp_resqueue_priority_cpucore_per_segment -v 8
```

5.6 备 Master 镜像情况

5.6.1 概述

作为重要的数据库，备 master 是必不可少的，经常查看备 master 健康情况能有效的替换主 master 节点。

5.6.2 查看备 master 的运行情况

```
$ gpstate -f
```

```
[gpadmin@gpdev152 ~]$ gpstate -f
20190531:18:41:45:407725 gpstate:gpdev152:gpadmin-[INFO]:--Starting gpstate with args: -f
20190531:18:41:45:407725 gpstate:gpdev152:gpadmin-[INFO]:--local Greenplum Version: 'postgres (Greenplum Database) 5.11.1 build commi
20190531:18:41:45:407725 gpstate:gpdev152:gpadmin-[INFO]:--master Greenplum Version: 'PostgreSQL 8.3.23 (Greenplum Database 5.11.1 bu
dfd2) on x86_64-pc-linux-gnu, compiled by GCC gcc (GCC) 6.2.0, 64-bit compiled on Sep 20 2018 18:12:07'
20190531:18:41:45:407725 gpstate:gpdev152:gpadmin-[INFO]:--Obtaining Segment details from master...
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:--Standby master details
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-----
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-- Standby address          = gpdev153
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-- Standby data directory    = /data/gpmaster/gpseg-1
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-- Standby port              = 5432
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-- Standby PID                = 43194
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-- Standby status           = Standby host passive
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-----
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:--pg_stat_replication
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-----
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:--WAL Sender State: streaming
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:--Sync state: sync
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:--Sent Location: 2/6EDC7DF0
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:--Flush Location: 2/6EDC7DF0
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:--Replay Location: 2/6EDC7270
20190531:18:41:46:407725 gpstate:gpdev152:gpadmin-[INFO]:-----
```

在以上可以看出备用 master 的信息是 gpdev153 上数据目录在/data/gpmaster/gpseg-1

5.6.3 备 master 需要检测的项

选项	内容
Master 是否实际切换过	没有
是否进行过切换演练	没有

浮动 IP	无
集群 HA	没有

5.7 segment 镜像情况

5.7.1 查看集群中的镜像分布情况

\$ gpstate -m

```

[[gpadmin@gpmdw /home/xiaoxu]$ gpstate -m
20190531:18:47:02:473050 gpstate:gpmdw:gpadmin-[INFO]:-Starting gpstate with args: -m
20190531:18:47:02:473050 gpstate:gpmdw:gpadmin-[INFO]:-local Greenplum Version: 'postgres (Greenplum Database) 5.11.1 build commit:5408
20190531:18:47:02:473050 gpstate:gpmdw:gpadmin-[INFO]:-master Greenplum Version: 'PostgreSQL 8.3.23 (Greenplum Database 5.11.1 build co
2) on x86_64-pc-linux-gnu, compiled by GCC gcc (GCC) 6.2.0, 64-bit compiled on Sep 20 2018 18:12:07'
20190531:18:47:02:473050 gpstate:gpmdw:gpadmin-[INFO]:-Obtaining Segment details from master...
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:-----
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:--Current GPDB mirror list and status
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:--Type = Group
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:-----
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- Mirror   Datadir   Port   Status   Data Status
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw2   /greenplum/data/gpdataml/gpseg0   50000   Passive   Synchronized
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw2   /greenplum/data/gpdatam2/gpseg1   50001   Passive   Synchronized
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw2   /greenplum/data/gpdatam3/gpseg2   50002   Passive   Synchronized
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw2   /greenplum/data/gpdatam4/gpseg3   50003   Passive   Synchronized
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw2   /greenplum/data/gpdatam5/gpseg4   50004   Passive   Synchronized
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw2   /greenplum/data/gpdatam6/gpseg5   50005   Passive   Synchronized
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw2   /greenplum/data/gpdatam7/gpseg6   50006   Passive   Synchronized
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw2   /greenplum/data/gpdatam8/gpseg7   50007   Passive   Synchronized
20190531:18:47:03:473050 gpstate:gpmdw:gpadmin-[INFO]:- gpsdw3   /greenplum/data/gpdataml/gpseg8   50000   Passive   Synchronized

```

主要查看是否有 ERROR 信息以及 Data Status 的状态是否是同步状态

5.8 资源队列情况

5.8.1 用户与资源队列检查

5.8.1.1 查看负载管理资源队列的状态和活动

该视图允许管理员查看到一个负载管理资源队列的状态和活动。它显示在系统中一个特定的资源队列有多少查询正在等待执行以及有多少查询当前是活动的。

```
select * from gp_toolkit.gp_resqueue_status;
```

1 select * from gp_toolkit.gp_resqueue_status;

2

信息

结果1

queueid	rsqname	rsqcountlimit	rsqcountvalue	rsqcostlimit	rsqcostvalue	rsqmemorylimit	rs
2156929	test_queue	50	0	-1	0	1099510000000	
6055	pg_default	20	0	-1	0	-1	

参数详解如下：

列名	描述
queueid	资源队列的 ID。
rsqname	资源队列名。
rsqcountlimit	一个资源队列的活动查询数的阈值。如果值为-1 则意味着没有限制。
rsqcountvalue	资源队列中当前正在被使用的活动查询槽的数量。
rsqcostlimit	资源队列的查询代价阈值。如果值为-1 则意味着没有限制。
rsqcostvalue	当前在资源队列中所有语句的总代价。
rsqmemorylimit	资源队列的内存限制。
rsqmemoryvalue	当前资源队列中所有语句使用的总内存。
rsqwaiters	当前在资源队列中处于等待状态的语句数目。
rsqholders	资源队列中当前正在系统上运行的语句数目。

5.8.1.2 查看当前用户使用的是什么队列

该视图显示与角色相关的资源队列。该视图能够被所有用户访问。

```
select * from gp_toolkit.gp_resq_role;
```

```
1 select * from gp_toolkit.gp_resq_role;
```

信息 结果1	
rrrolname	rrrsqname
pg_default	pg_default
pg_default	pg_default
gpcc_basic	pg_default
gpcc_operator	pg_default
gpcc_operator_basic	pg_default
spider	pg_default
fresh_ent	pg_default

列名	描述
rrrolname	角色（用户）名。
rrrsqname	指定给角色的资源队列名。如果一个角色没有被明确地指定一个资源队列，那么它将会在默认资源队列 <i>pg_default</i> 中。

5.8.1.3 查看队列的活动负载状态

对于那些有活动负载的资源队列，该视图为每一个通过资源队列提交的活动语句显示一行。该视图能够被所有用户访问。

```
select * from gp_toolkit.gp_resq_activity;
```

```
1 select * from gp_toolkit.gp_resq_activity;
```

信息 结果1					
resqprocpid	resqrole	resqoid	resqname	resqstart	resqstatus
(Null)	(Null)	(Null)	(Null)	(Null)	(Null)

列名	描述
resqprocpid	指定给该语句的进程 ID（在 Master 上）。
resqrole	用户名。
resqoid	资源队列对象 ID。
resqname	资源队列名。
resqstart	语句被发送到系统的时间。
resqstatus	语句的状态：正在执行、等待或者取消。

5.8.1.4 查看负载管理特性的 Greenplum 数据库资源队列的信息

```
select * from pg_catalog.pg_resqueue;
```

名称	类型	描述
rsqname	name	资源队列名。
rsqcountlimit	real	资源队列的活动查询阈值。
rsqcostlimit	real	资源队列的查询代价阈值。
rsqovercommit	boolean	当系统是空闲时，允许超过代价阈值的查询运行。
rsqignorecostlimit	real	查询被认为是一个“小查询”的查询代价限制。代价低于该限制的查询将不会被入队列而是立即被执行。

5.8.1.5 查看队列资源的状态

```
SELECT * FROM pg_resqueue_status;
```

参数	描述
rsqname	资源队列的名称。

rsqcountlimit	资源队列的活动查询阈值。值-1 表示没有限制。
rsqcountvalue	当前在资源队列中使用的活动查询槽的数量。
rsqcostlimit	资源队列的查询开销阈值。值-1 表示没有限制。
rsqcostvalue	当前在资源队列中的所有语句的总成本。
rsqwaiters	当前在资源队列中等待的语句数。
rsqholders	当前在此资源队列中在系统上运行的语句数。

5.8.1.6 查看每个资源队列的配置情况

```
select * from pg_resqueue_attributes;
```

1 select * from pg_resqueue_attributes;			
信息	结果1		
rsqname	resname	ressetting	restypid
pg_default	active_statements	20	1
pg_default	max_cost	-1	2
pg_default	min_cost	0	3
pg_default	cost_overcommit	0	4
pg_default	priority	max	5
pg_default	memory_limit	-1	6
test_queue	active_statements	50	1
test_queue	max_cost	-1	2
test_queue	min_cost	0	3
test_queue	cost_overcommit	0	4
test_queue	priority	max	5
test_queue	memory_limit	-1	6

column	type	references	说明
--------	------	------------	----

rsqname	name	pg_resqueue.rsqname	资源队列的名字
resname	text		资源队列属性的名称
resetting	text		资源队列当前属性的名字
restypid	integer		系统分配资源队列的类型 ID

5.8.1.7 修改资源队列的语句

ALTER RESOURCE QUEUE pg_default WITH (priority=max);

pg_default：资源队列名称

priority=max：需要修改的值

5.9 Statistics 状态检查

该视图显示那些没有统计信息的表，因此可能需要在表上执行 ANALYZE tablename 命令。

5.9.1 查看所有的表

select * from gp_toolkit.gp_stats_missing where smisize='f';

```

1  select * from gp_toolkit.gp_stats_missing where smisize='f';
2
3

```

信息	结果1	
smischema	smitable	smisize smicols smirecs
riskbell	master_changed_baseinfo	f 4 0
ods	t_ent_mab_pawn_20180824	f 10 0
ods	t_ent_inspect_20181115	f 8 0
main	t_ent_copyrightinfo_tmp4ae_update	f 18 0
ods	t_ent_person_staff_all_tmp_relation_neo4j	f 2 2

5.9.2 查看指定 schema 下的表

select * from gp_toolkit.gp_stats_missing where smisize='f' and smischema in ('riskbell','main');

1
2

select * from gp_toolkit.gp_stats_missing where smisize='f' and smischema in ('riskbell','main');

信息	结果1			
smischema	smitable	smisize	smicols	smirecs
main	t_ent_job_employ_entinfo_tmp4record_id_hdfs	f	21	0
riskbell	master_changed_baseinfo	f	4	0
main	t_ent_copyrightinfo_tmp4ae_update	f	18	0
main	t_ent_formated_phone	f	6	0
main	t_ent_copyrightorg_tmp4insert	f	19	0
riskbell	publish_pushdate	f	3	0

字解释如下:

列名	描述
smischema	方案名。
smitable	表名。
smisize	这些表有统计信息吗？如果这些表没有行数量统计以及行大小统计记录在系统表中，取值为 false (简写 f)，这也表明该表需要被分析。如果表没有包含任何的函数时，值也为 false 。例如，分区表的父表总是空的，同时也总是返回一个 false 。
smicols	表中的列数。
smirecs	表中的行数。

5.10 Skew 状态检查

Skew 阶段主要查看表的倾斜的表，过高的表倾斜由于查询会落到一个 segment 上，影响了集群的性能,建议分布数据做到分布均匀。请看 <[5.1 检查大小超过 1GB 的表倾斜情况](#)> 详解

5.11 Bloat 状态

Bloat 阶段主要查看表的膨胀率，过高的表的膨胀率会影响表的查询以及增删改查性能，原

因是表中存在的不可见的文件过大,建议定期的清理。请看 <[5.2 检查膨胀率过高的表](#)> 详解

5.12 集群系统性能

5.12.1 DB 性能查看

以下主要对集群的性能测试,建议在安装数据库时检测一次,并把检测的结果保存,以便于下次测试结果相比较。性能测试的案例请查看:

<https://blog.csdn.net/xfg0218/article/details/82785187>

5.12.2 SQL 锁检

```
select b.query_start,a.* from
gp_toolkit.gp_locks_on_relation a, pg_stat_activity b where a.lorpid=b.procpid
and a.lorrelname not like 'pg_%' and a.lorrelname not like 'gp_%' order by 1
desc;
```

```
1 select b.query_start,a.* from
2 gp_toolkit.gp_locks_on_relation a, pg_stat_activity b where a.lorpid=b.procpid
3 and a.lorrelname not like 'pg_%' and a.lorrelname not like 'gp_%' order by 1
4 desc;
```

query_start	lorlocktype	lordatabase	lorrelname	lorrelation	lortransaction	lorpid	lormode	lorgranted	lorcurrentquery
2019-06-03 11:14:relation		25270	t_ent_formated_contact	9746861	(Null)	557492	AccessShareL t		select * from main_t

以上 SQL 显示当前所有表上持有锁,以及查询关联的锁的相关联的会话信息。

列	描述
lorlocktype	能够加锁对象的类型: relation、extend、page、tuple、transactionid、object、userlock、resource queue 以及 advisory
lordatabase	对象存在的数据库对象 ID, 如果对象为一个共享对象则该值为 0。
lorrelname	关系名。
lorrelation	关系对象 ID。
lortransaction	锁所影响的事务 ID 。
lorpid	持有或者等待该锁的服务器端进程的进程 ID 。如果该锁被一个预备事务持有则为 NULL。

lormode	由该进程持有或者要求的锁模式名。
lorgranted	显示是否该锁被授予（true）或者未被授予（false）。
lorcurrentquery	会话中的当前查询。

5.12.3 集群硬件性能查看

1、请使用 gpcheckperf 命令检测集群中磁盘的读写网卡的信息等，详细的过程请查看：

<https://blog.csdn.net/xfg0218/article/details/90711569>

2、在测试期间请使用以下命令收集服务器的指标信息并保存

```
$ cat cluster_hosts
```

```
gpmdw
```

```
gpsdw1
```

```
gpsdw2
```

```
gpsdw3
```

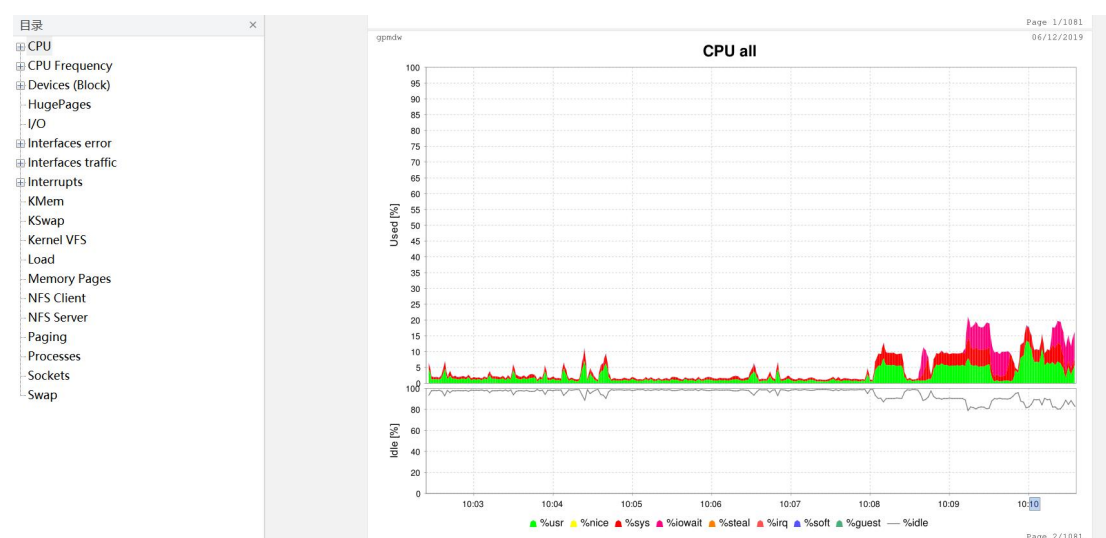
```
nohup gpssh -f cluster_hosts sar -A 1 20000 >>
```

```
/home/gpadmin/cluster-target.txt &
```

详细的查看指标的信息请查看：

<https://blog.csdn.net/xfg0218/article/details/91490999>

3、根据文件 cluster-target.txt 过滤出需要查看的机器的信息在 kSar 上查看即可, 效果如下所示：



5.13 查看集群系统元数据

```
nohup gpcheckcat -p 5432 staging>> greenplum-metadata.log &
```

```
$ head -n 20 greenplum-metadata.log
```

```
Truncated batch size to number of primaries: 49
```

```
Connected as user 'gpadmin' to database 'ch****', port '5432', gpdb  
version '5.11'
```

```
-----  
Batch size: 49
```

```
Found and dropped 8 unbound temporary schemas
```

```
Performing test 'unique_index_violation'
```

```
Total runtime for test 'unique_index_violation': 0:12:33.53
```

```
Performing test 'duplicate'
```

```
Total runtime for test 'duplicate': 0:00:48.59
```

```
Performing test 'missing_extraneous'
```

```
Total runtime for test 'missing_extraneous': 0:02:28.29
```

```
*****
```

5.14 查看集群中分布键倾斜率比较高的表

表的分布及均匀会造成查询数据时性能会大大下降, 定期清理倾斜率高的表会提高集群的整体性能, 详见 shell 脚本请查看:

<https://github.com/xfg0218/greenplum--summarize/tree/master/201906/greenplum-table-percentage>

5.15 使用 gpcheckcat 命令检测集群

5.15.1 gpcheckcat 检测项说明

gpcheckcat 详细的参数说明请查看:

<https://blog.csdn.net/xfg0218/article/details/90910764>

5.15.2 检查 master,segment 的 catalog 一致性

对每一个数据执行以下命令,默认的是

```
$ gpcheckcat -O
```

```
*****
```

```
SUMMARY REPORT
```

```
=====
```

```
Completed 11 test(s) on database 'template1' at 2019-06-05 17:40:32 with elapsed time 0:00:38
```

```
Found no catalog issue
```

```
real 0m20.840s
```

```
user 0m0.370s
```

```
sys 0m0.180s
```

在以上可以看出没有找到问题

5.15.3 检查持久化表的 catalog 一致性。

对每一个数据执行以下命令

```
$ time gpcheckcat -R persistent
```

```
*****
```

```
SUMMARY REPORT
```

```
=====
```

```
Completed 1 test(s) on database 'template1' at 2019-06-05 17:44:09 with elapsed time 0:00:11
```

```
Found no catalog issue
```

```
real 0m12.840s
```

```
user 0m0.370s
```

```
sys 0m0.180s
```

如果有输出，说明有不一致的持久化表的 catalog。

5.15.4 检查 pg_class 与 pg_attribute 是否不一致

对每一个数据执行以下命令

```
$ gpcheckcat -R pgclass
```

```
*****
```

```
SUMMARY REPORT
```

```
=====
```

```
Completed 1 test(s) on database 'template1' at 2019-06-05 17:51:46 with elapsed time 0:00:00
```

```
Found no catalog issue
```

real 0m1.804s
user 0m0.215s
sys 0m0.071s

如果有输出，说明 pg_class 与 pg_attribute 不一致。

1 select username, datname, query_start, current_query from pg_stat_activity where waiting;

信息结果1

username	datname	query_start	current_query
gpadmin	chind	2019-06-05 18:37:10.3187	select ods.s_...('S20190605183327')
gpadmin	chind	2019-06-05 18:36:47.8852	select ods.s_...('S20190605183327')

5.16 表活跃度监控

5.16.1 IO 活跃度的表

select * from pg_catalog.pg_statio_all_tables;

参数	类型	描述
relid	oid	表的 OID
schemaname	name	此表所在的架构的名称
relname	name	该表的名称
heap_blks_read	bigint	从此表读取的磁盘块数
heap_blks_hit	bigint	此表中的缓冲区命中数
idx_blks_read	bigint	从此表上的所有索引读取的磁盘块数
idx_blks_hit	bigint	此表中所有索引中的缓冲区命中数
toast_blks_read	bigint	从此表的 TOAST 表中读取的磁盘块数(如果有)
toast_blks_hit	bigint	此表的 TOAST 表中的缓冲区命中数(如果有)
tidx_blks_read	bigint	从此表的 TOAST 表索引读取的磁盘块数(如果有)
tidx_blks_hit	bigint	此表的 TOAST 表索引中的缓冲区命中数(如果有)

1、使用(索引+表)较少、较多的表

(heap_blks_read+heap_blks_hit+toast_blks_read+toast_blks_hit+idx_blks_read+idx_blks_hit+tidx_blks_read+tidx_blks_hit)

2、消耗 IO 较多的(索引+表)表(heap_blks_read+toast_blks_read+idx_blks_read+tidx_blks_read)

3、使用(索引)较少、较多的表 (idx_blks_read+idx_blks_hit+tidx_blks_read+tidx_blks_hit)

4、消耗 IO 较多的(索引)表 (idx_blks_read+tidx_blks_read)

5、使用(表)较少、较多的表 (heap_blks_read+heap_blks_hit+toast_blks_read+toast_blks_hit)

6、消耗 IO 较多的(表)表 (heap_blks_read+toast_blks_read)

5.16.2 表的访问活跃度

```
select * from pg_catalog.pg_stat_all_tables;
```

参数	类型	描述
relid	oid	表的 OID
schemaname	name	此表所在的架构的名称
relname	name	该表的名称
seq_scan	bigint	在此表上启动的顺序扫描数
seq_tup_read	bigint	顺序扫描获取的实时行数
idx_scan	bigint	在此表上启动的索引扫描数
idx_tup_fetch	bigint	索引扫描获取的实时行数
n_tup_ins	bigint	插入的行数
n_tup_upd	bigint	更新的行数（包括 HOT 更新的行）
n_tup_del	bigint	删除的行数
n_tup_hot_upd	bigint	HOT 更新的行数（即，不需要单独的索引更新）
n_live_tup	bigint	估计的实时行数
n_dead_tup	bigint	估计死行数
n_mod_since_analyze	bigint	自上次分析此表以来修改的行数估计值
last_vacuum	timestamp with time zone	上次手动清理此表（不计 VACUUM FULL）
last_autovacuum	timestamp with time zone	上次由 autovacuum 守护程序对此表进行清理的时间
last_analyze	timestamp with time zone	上次手动分析此表的时间
last_autoanalyze	timestamp with time zone	上次由 autovacuum 守护程序分析此表的时间

vacuum_count	bigint	此表已手动清理的次数（不计 VACUUM FULL）
autovacuum_count	bigint	autovacuum 守护程序对此表进行清理的次数
analyze_count	bigint	手动分析此表的次数
autoanalyze_count	bigint	autovacuum 守护程序分析此表的次数

5.16.3 热表的查询

```
select current_database(),* from pg_stat_all_table order by seq_scan+ idx_scan desc limit 10;
```

5.16.4 冷表的查询

```
select  current_database(),* from pg_stat_all_tables order by seq_scan _idx_scan limit 10;
```