GreenPlum集群配置

## 安装服务器配置

|  |  |
| --- | --- |
| Operating System | 根据GreenPlum版本号参考官方文档  CentOS 64-bit 6.x or 7.x、Red Hat Enterprise Linux (RHEL) 64-bit 6.x or 7.x |
| File Systems | xfs required for data storage on SUSE Linux and Red Hat (ext3 supported for root file system) |
| Minimum CPU | Pentium Pro compatible (P3/Athlon and above) |
| Minimum Memory | 16 GB RAM per server |
| Disk Requirements | 150MB per host for Greenplum installation  Approximately 300MB per segment instance for meta data  Appropriate free space for data with disks at no more than 70% capacity High-speed, local storage |
| Network Requirements | 10 Gigabit Ethernet within the array  Dedicated, non-blocking switch |
| Software and Utilities | bash shell  GNU tars  GNU zip  GNU sed (used by Greenplum Database gpinitsystem) |

说明：大数据情况下的OLAP分析，服务器的磁盘IO、网络IO、CPU处理能力对于性能影响明显。且由水桶效应，以集群中最差的一个segment处理能力为基准。

## Linux系统OS参数配置

1. 禁用SELINUX、禁用防火墙
2. 编辑/etc/hosts，集群中所有服务器的IP和hostname
3. 编辑 /etc/security/limits.conf文件

\* soft nofile 65536

\* hard nofile 65536

\* soft nproc 131072

\* hard nproc 131072

1. 编辑/etc/sysctl.conf文件，以下值为greenplum默认linux内核参数配置

kernel.shmmax = 500000000

--maximum size of a single shared memory segment，单个share memory segment最大值，该参数设置与postgresql.conf配置中的shared\_buffers相关,具体配置值参考后续章节，官方默认配置大小：500000000/1024/1024=476.8MB

kernel.shmmni = 4096

*--*maximum amount *of* share memory segment on the host*，服务器*share memory segment个数最大值，官方默认配置4096个段

kernel.shmall = 4000000000

--maximum amount of shared memory on the host，服务器共享内存最大值，官方默认配置4000000\*PAGESIZE(4Kb)/1024/1024/1024= 15.258GB，ps：感觉原来的配置是算好pagesize之后的值，所以多了三个0，实际配置时应该根据系统pagesize设置

kernel.sem = 250 512000 100 2048

--第一列，表示每个信号集中的最大信号量数目；第二列，表示系统范围内的最大信号量总数目；第三列，表示每个信号发生时的最大系统操作数目；第四列，表示系统范围内的最大信号集总数目，计算方式250\*2048=512000。官方默认配置较小，可修改为kernel.sem = 3200 10240000 500 3200

kernel.sysrq = 1

--激活sysctem request key

kernel.core\_uses\_pid = 1

--激活core dump文件名添加pid，主要是排查问题

kernel.msgmnb = 65536

--一个消息队列中最大的字节数

kernel.msgmax = 65536

--从一个进程发送到另一个进程的消息最大长度。进程间的消息传递是在内核的内存中进行的。不会交换到硬盘上。所以如果增加该值，则将增加操作系统所使用的内存数量

kernel.msgmni = 2048

--指定消息队列标识的最大数目。

net.ipv4.tcp\_syncookies = 1

--开启SYN Cookies。在TCP服务器收到TCP SYN包并返回TCP SYN+ACK包时，不分配一个专门的数据区，而是根据这个SYN包计算出一个cookie值。在收到TCP ACK包时，TCP服务器在根据那个cookie值检查这个TCP ACK包的合法性。如果合法，再分配专门的数据区进行处理未来的TCP连接。当出现SYN等待队列溢出时，启用cookies来处理，可防范少量SYN攻击，默认为0，表示关闭。

net.ipv4.conf.default.accept\_source\_route = 0

--是否接受含有源路由信息的ip包。参数值为布尔值，1表示接受，0表示不接受。在充当网关的linux主机上缺省值为1，在一般的linux主机上缺省值为0。从安全性角度出发，建议关闭该功能

net.ipv4.tcp\_tw\_recycle = 1

--开启TCP连接复用功能，允许将time\_wait sockets重新用于新的TCP连接（主要针对time\_wait连接）

net.ipv4.tcp\_max\_syn\_backlog = 4096

--对于那些依然还未获得客户端确认的连接请求，需要保存在队列中最大数目

net.ipv4.conf.all.arp\_filter = 1

net.ipv4.ip\_local\_port\_range = 10000 65535

--segment实例监听端口号，不选择该区间，避免与系统其它应用冲突

net.core.netdev\_max\_backlog = 10000

--每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许送到队列的数据包的最大数目

net.core.rmem\_max = 2097152

-- 接收套接字缓冲区大小的最大值

net.core.wmem\_max = 2097152

--发送套接字缓冲区大小的最大值

vm.overcommit\_memory = 2

--0 、默认设置。内核执行启发式内存过量使用处理，方法是估算可用内存量，并拒绝明显无效的请求。

1 、内核执行无内存过量使用处理。使用这个设置会增大内存超载的可能性，但也可以增强大量使用内存任务的性能。

2 、内存拒绝等于或者大于总可用 swap 大小以及 overcommit\_ratio 指定的物理 RAM 比例的内存请求。设置为2时需设置合理的overcommit\_ratio参数

vm.overcommit\_ratio

--物理内存的比例，默认50

可用内存计算公式：（Physical-RAM-Size）\*ratio / 100 +(Swap-Size)

1. 磁盘I/O配置

设置磁盘的预读大小：

--/sbin/blockdev --getra devname 获取devname的预读大小

--/sbin/blockdev --setra bytes devname 设置devname的预读大小

官网推荐配置bytes = 16384

官方推荐XFS文件系统，mount配置如下：

--rw,nodev,noatime,nobarrier,inode64

磁盘I/O调度策略

--cfq：完全公平排队I/O调度程序

试图均匀地分布对I/O带宽的访问,避免进程被饿死并实现较低的延迟,默认的I/O调度器

--deadline：截止时间调度程序

确保了在一个截止时间内服务请求,这个截止时间是可调整的,而默认读期限短于写期限.这样就防止了写操作因为不能被读取而饿死的现象，Deadline对数据库环境是最好的选择

--as：预料I/O调度程序

本质上与Deadline一样,但在最后一次读操作后,要等待6ms,才能继续进行对其它I/O请求进行调度

官方推荐Deadline策略

查看支持调度策略：dmesg | grep -i scheduler

临时设置调度策略：echo deadline > /sys/block/sda/queue/scheduler

永久更改调度策略：修改内核引导参数,加入elevator=调度程序名

RHEL 6.x or CentOS 6.x可执行：

编辑/boot/grub/grub.conf文件

RHEL 7.x or CentOS 7.x可执行：

grubby --update-kernel=ALL --args="elevator=deadline"

Reboot重启生效

禁用Transparent Huge Pages，一般默认禁用，修改方法与调度策略类似，在内核引导参数添加transparent\_hugepage=never

禁用RemoveIPC=no，编辑/etc/systemd/logind.conf，执行service systemd-logind restart生效

## GreenPlum集群配置

集群参数修改命令：gpconfig -c configname -v value -m mv/master有特殊设置的

1、内存参数设置：

shared\_buffers

--单个segment实例分配的共享内存大小，与SHMMAX 和SHMALL参数相关。

SHMMAX >shared\_buffers(默认128MB) + other\_seg\_shmem(默认数据节点111MB、主节点79MB)

SHMALL > (num\_instances\_per\_host \* ( shared\_buffers + other\_seg\_shmem )) + other\_app\_shared\_mem

postgresql对数据操作时都要先将数据从磁盘读取到内存中，然后进行更新，最后再将数据写回磁盘。shared\_buffers的功能就是用于存放从磁盘读取的数据。

该参数为system、local参数，需在每一个segment修改，并重启GP服务生效

temp\_buffers

--临时缓冲区，用于数据库会话访问临时表数据。默认1024，但是使用时会额外增加到8192。

该参数为master、session参数，只用在主节点参数，或者会话第一次使用临时表之前设置，可以设置很大的值，根据实际使用分配内存。

work\_mem

--工作内存或者操作内存。其负责内部的sort和hash操作，合适的work\_mem大小能够保证这些操作在内存中进行。定义太小的话，sort或者hash操作将需要与硬盘进行swap，这样会极大的降低系统的性能；太大的话致使在能够在内存中完成的操作数量减少，其他的部分需要与磁盘进行swap操作，增加IO降低性能。

实际计算时符合如下公式：

max\_connections\*work\_mem+shared\_buffers+temp\_buffers+maintenance\_work\_mem+操作系统所需内存< RAM

maintenance\_work\_mem

维护工作内存，主要是针对数据库的维护操作或者语句。尽量的将这些操作在内存中进行。主要针对VACUUM，CREATE INDEX，REINDEX等操作。在对整个数据库进行VACUUM或者较大的index进行重建时，适当的调整该参数非常必要。

2、队列参数配置

gp\_vmem\_protect\_limit

单个segment实例所有进程可分配的最大内存，只有在使用资源队列的时候该参数有效，计算公式如下：

gp\_vmem\_protect\_limit = ((SWAP + RAM) – (7.5GB + 0.05 \* RAM)) / 1.7

/ acting\_primary\_segments

以124GB+4GB，4主4备份，假设最大可能活跃节点为6

gp\_vmem\_protect\_limit = ((124+4)-(7.5+0.05\*124))/1.7/6 = 11.2GB

实际配置时可适当高于该值，现在是保证的是备用节点启用时候的状态。

同理可根据该参数计算内核参数：

vm.overcommit\_ratio = (RAM - (0.026 \* gp\_vmem)) / RAM

vm.overcommit\_ratio = (124-0.026\*11.2\*6)/124 = 0.98, 默认50，推荐设置99

该参数为system、local参数，需在每一个segment修改，并重启GP服务生效

gp\_resqueue\_priority\_cpucores\_per\_segment

每个segment实例分配的CPU单位，例如：10核服务器安装了4个segment实例则配置为2.5，只有master则配置为10。默认配置4，错误的配置会导致CPU利用率不高，或执行计划不正常执行，在配置多个资源队列之间的调度非常有效。

该参数为system、local参数，需在每一个segment修改，并重启GP服务生效

gp\_resqueue\_memory\_policy

资源队列内存管理策略，默认eager\_free。

eager\_free：物尽其用，利用并不是所有的查询都同时执行，尽可能利用内存空间。

auto：启用该策略时，内存参数由[statement\_mem](http://gpdb.docs.pivotal.io/590/ref_guide/config_params/guc-list.html" \l "statement_mem)和资源队列的memory\_limit控制管理。

gp\_resqueue\_priority

启用资源队列优先级

statement\_mem

设置单个查询的segment内存，该值不能超过max\_statement\_mem，配置资源队列的memory\_limit，当[gp\_resqueue\_memory\_policy](http://gpdb.docs.pivotal.io/590/ref_guide/config_params/guc-list.html" \l "gp_resqueue_memory_policy) =auto时该参数与memory\_limit共同控制内存管理，溢出时分配到文件，修改该参数先修改max\_statement\_mem。

计算公式如下：

( gp\_vmem\_protect\_limitGB \* .9 ) / max\_expected\_concurrent\_queries

默认资源队列最大并发数20，以124GB+4GB配置为例：

statement\_mem = 11.2GB\*0.9/20=0.5GB, 其中0.1用于并发连接

Master、session级别参数

max\_statement\_mem

单个查询segment分配的最大内存，避免statement\_mem设置过大引起的OOM错误，默认2000MB

3、其他

effective\_cache\_size

估值参数，该值设置越大，越利于使用索引，不会影响其他内存分配，默认512MB。

## 推荐配置

服务器配置128GB RAM + 16GB SWAP，32核，CPU+

万兆网络配置

最小集群1 Master + 3 Slave（Slave节点最好偶数）

每台Slave服务器Segment实例推荐配置4~8个其中Primary、Mirror各一半，集群配置segment节点时，建议将一台Slave上的备份节点分配到不同的机器上。

已1Master + 4 Slave，每个Slave配置8个Segment实例为例：

Linux内核参数推荐配置如下：

etc/security/limits.conf

soft nofile 65536

hard nofile 65536

soft nproc 131072

hard nproc 131072

 /etc/security/limits.conf

kernel.shmmax = 1030792151

#kernel.shmmax = 500000000

kernel.shmmni = 4096

kernel.shmall = 30198988

#kernel.shmall = 4000000000

kernel.sem = 250 512000 100 2048

kernel.sysrq = 1

kernel.core\_uses\_pid = 1

kernel.msgmnb = 65536

kernel.msgmax = 65536

kernel.msgmni = 2048

net.ipv4.tcp\_syncookies = 1

net.ipv4.conf.default.accept\_source\_route = 0

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_max\_syn\_backlog = 4096

net.ipv4.conf.all.arp\_filter = 1

net.ipv4.ip\_local\_port\_range = 10000 65535

net.core.netdev\_max\_backlog = 10000

net.core.rmem\_max = 2097152

net.core.wmem\_max = 2097152

vm.overcommit\_memory = 2

vm.overcommit\_ratio = 98

磁盘I/O配置：

预读大小设置：--/sbin/blockdev --setra 16384 devname

调度策略：deadline

硬盘选择：SAS接口磁盘，XFS文件系统，mount配置：

rw,nodev,noatime,nobarrier,inode64

GreenPlum集群推荐配置如下：

shared\_buffers