PostgreSQL 数据库巡检模板和脚本

简介：

https://github.com/digoal/pgsql_admin_script

```
#!/bin/bash

# 已在 CentOS 6.x 上进行测试
# author: digoal
# 2015-10
# 用法 . ./generate_report.sh >/tmp/report.log 2>&1
# 生成目录 grep -E "^----->>>|^\|" /tmp/report.log | sed 's/^----->>>---->>>/ /' | sed '1 i\ \ 目录\n\n' | sed '$ a\ \n\n\ \ 正文\n\n'

# 请将以下变量修改为与当前环境一致，并且确保使用这个配置连接任何数据库都不需要输入密码
export PGDATA=/data01/pg_root_1921
export PGHOST=127.0.0.1
export PGPORT=1921
export PGDATABASE=postgres
export PGUSER=postgres
export PGHOME=/opt/pgsql

export DATE=`date +"%Y%m%d%H%M"`
```

```
export
LD_LIBRARY_PATH=$PGHOME/lib:/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib:$LD_LIBRARY_PATH
export PATH=$PGHOME/bin:$PATH:.


# 记住当前目录
PWD=`pwd`

# 获取 postgresql 日志目录
pg_log_dir=`grep '^\ *[a-z]' $PGDATA/postgresql.conf|awk -F "#" '{print $1}'|grep log_directory|awk -F "=" '{print $2}'`

# 检查是否 standby
is_standby=`psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE -q -A -t -c 'select pg_is_in_recovery()'`


echo "      ----- PostgreSQL 巡检报告 -----   "
echo "      ===== $DATE          =====   "

if [ $is_standby == 't' ]; then
echo "      ===== 这是 standby 节点      =====   "
else
echo "      ===== 这是 primary 节点      =====   "
fi
echo ""
```

```
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "|                                操作系统信息                                |"
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""

echo "----->>>---->>>   主机名: "
hostname -s
echo ""
echo "----->>>---->>>   主机网络信息: "
ifconfig
echo ""
echo "----->>>---->>>   操作系统内核: "
uname -a
echo ""
echo "----->>>---->>>   内存(MB): "
free -m
echo ""
echo "----->>>---->>>   CPU: "
lscpu
echo ""
echo "----->>>---->>>   块设备: "
lsblk
```

```
echo ""
echo "----->>>---->>>   拓扑: "
lstopo-no-graphics
echo ""
echo "----->>>---->>>   进程树: "
pstree -a -A -c -l -n -p -u -U -Z
echo ""
echo "----->>>---->>>   操作系统配置: "
echo "----->>>---->>>   /etc/sysctl.conf "
grep "^[a-z]" /etc/sysctl.conf
echo ""
echo "----->>>---->>>   /etc/security/limits.conf "
grep -v "^#" /etc/security/limits.conf|grep -v "^$"
echo ""
echo "----->>>---->>>   /etc/security/limits.d/*.conf "
grep -v "^#" /etc/security/limits.d/*.conf|grep -v "^$"
echo ""
echo "----->>>---->>>   /etc/sysconfig/iptables "
cat /etc/sysconfig/iptables
echo ""
echo "----->>>---->>>   sysctl -a  信息: "
sysctl -a
echo ""
echo "----->>>---->>>   硬盘 SMART 信息(需要 root): "
for i in `smartctl --scan|awk '{print $1}'`; do echo -e "\n\nDEVICE $i"; smartctl
-x $i; done
echo ""
echo "----->>>---->>>   /var/log/boot.log "
```

```
cat /var/log/boot.log
echo ""
echo "----->>>---->>>    /var/log/cron(需要 root) "
cat /var/log/cron
echo ""
echo "----->>>---->>>    /var/log/dmesg "
cat /var/log/dmesg
echo ""
echo "----->>>---->>>    /var/log/messages(需要 root) "
tail -n 500 /var/log/messages
echo ""
echo "----->>>---->>>    /var/log/secure(需要 root) "
cat /var/log/secure
echo ""
echo "----->>>---->>>    /var/log/wtmp "
who -a /var/log/wtmp
echo -e "\n"


echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "|                                数据库信息                                |"
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""
```

```
echo "----->>>---->>>  数据库版本: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select version()'

echo "----->>>---->>>  用户已安装的插件版本: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),* from pg_extension'
done

echo "----->>>---->>>  用户使用了多少种数据类型: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
```

```
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),b.typname,count(*) from pg_attribute a,pg_type b where a.atttypid=b.oid and a.attrelid in (select oid from pg_class where relnamespace not in (select oid from pg_namespace where nspname ~
$$
^pg_
$$
  or nspname=
$$
information_schema
$$
)) group by 1,2 order by 3 desc'
done

echo "----->>>---->>>  用户创建了多少对象: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
```

```
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),rolname,nspname,relkind,count(*) from pg_class a,pg_authid b,pg_namespace c where a.relnamespace=c.oid and a.relowner=b.oid and nspname !~
$$
^pg_
$$
 and nspname<>
$$
information_schema
$$
 group by 1,2,3,4 order by 5 desc'
done

echo "----->>>---->>>  用户对象占用空间的柱状图: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
```

```
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),buk this_buk_no,cnt rels_in_this_buk,pg_size_pretty(min) buk_min,pg_size_pretty(max) buk_max from( select row_number() over (partition by buk order by tsize),tsize,buk,min(tsize) over (partition by buk),max(tsize) over (partition by buk),count(*) over (partition by buk) cnt from ( select pg_relation_size(a.oid) tsize, width_bucket(pg_relation_size(a.oid),tmin,tmax,10) buk from (select min(pg_relation_size(a.oid)) tmin,max(pg_relation_size(a.oid)) tmax from pg_class a,pg_namespace c where a.relnamespace=c.oid and nspname !~
$$
^pg_
$$
  and nspname<>
$$
information_schema
$$
) t, pg_class a,pg_namespace c where a.relnamespace=c.oid and nspname !~
$$
^pg_
$$
  and nspname<>
$$
information_schema
$$
  ) t)t where row_number=1;'
done

echo "----->>>---->>>   当前用户的操作系统定时任务: "
```

```
echo "I am `whoami`"
crontab -l
echo "建议: "
echo "       仔细检查定时任务的必要性，以及定时任务的成功与否的评判标准，以及监控措施. "
echo "       请以启动数据库的 OS 用户执行本脚本. "
echo -e "\n"

common() {
#  进入 pg_log 工作目录
cd $PGDATA
eval cd $pg_log_dir

echo "----->>>---->>>    获取 pg_hba.conf md5 值: "
md5sum $PGDATA/pg_hba.conf
echo "建议: "
echo "       主备 md5 值一致(判断主备配置文件是否内容一致的一种手段，或者使用 diff)."
echo -e "\n"

echo "----->>>---->>>    获取 pg_hba.conf 配置: "
grep '^\ *[a-z]' $PGDATA/pg_hba.conf
echo "建议: "
echo "       主备配置尽量保持一致，注意 trust 和 password 认证方法的危害
```

```
(password 方法 验证时网络传输密码明文，建议改为 md5)，建议除了 unix socket 可以使用 trust 以外，其他都使用 md5 或者 LDAP 认证方法."
echo "      建议先设置白名单(超级用户允许的来源 IP，可以访问的数据库)，再设置黑名单(不允许超级用户登陆，reject)，再设置白名单(普通应用)，参考 pg_hba.conf 中的描述. "
echo -e "\n"

echo "----->>>---->>>   获取 postgresql.conf md5 值: "
md5sum $PGDATA/postgresql.conf
echo "建议: "
echo "      主备 md5 值一致(判断主备配置文件是否内容一致的一种手段，或者使用 diff)."
echo -e "\n"

echo "----->>>---->>>   获取 postgresql.conf 配置: "
grep '^\ *[a-z]' $PGDATA/postgresql.conf|awk -F "#" '{print $1}'
echo "建议: "
echo "      主备配置尽量保持一致，配置合理的参数值."
echo -e "      建议修改的参数列表如下  ( 假设操作系统内存为 128GB，数据库独占操作系统，数据库版本 9.4.x )  :
echo ""
listen_addresses = '0.0.0.0'                  # 监听所有 IPV4 地址
port = 1921                                   # 监听非默认端口
max_connections = 4000                        # 最大允许连接数
```

superuser_reserved_connections = 20 # 为超级用户保留的连接

unix_socket_directories = '.' # unix socket 文件目录最好放在 $PGDATA 中，确保安全

unix_socket_permissions = 0700 # 确保权限安全

tcp_keepalives_idle = 30 # 间歇性发送 TCP 心跳包，防止连接被网络设备中断.

tcp_keepalives_interval = 10

tcp_keepalives_count = 10

shared_buffers = 16GB # 数据库自己管理的共享内存大小

huge_pages = try # 尽量使用大页，需要操作系统支持，配置 vm.nr_hugepages*2MB 大于 shared_buffers.

maintenance_work_mem = 512MB # 可以加速创建索引，回收垃圾(假设没有设置 autovacuum_work_mem)

autovacuum_work_mem = 512MB # 可以加速回收垃圾

shared_preload_libraries = 'auth_delay,passwordcheck,pg_stat_statements,auto_explain' # 建议防止暴力破解，密码复杂度检测，开启 pg_stat_statements，开启 auto_explain，参考 http://blog.163.com/digoal@126/blog/static/163877040201498529415 86

bgwriter_delay = 10ms # bgwriter process 间隔多久调用 write 接口(注意不是 fsync)将 shared buffer 中的 dirty page 写到文件系统.

bgwriter_lru_maxpages = 1000 # 一个周期最多写多少脏页

max_worker_processes = 20 # 如果要使用 worker process,

最多可以允许 fork 多少个 worker 进程.

wal_level = logical # 如果将来打算使用 logical 复制，最后先配置好，不需要停机再改.

synchronous_commit = off # 如果磁盘的 IOPS 能力一般，建议使用异步提交来提高性能，但是数据库 crash 或操作系统 crash 时，最多可能丢失 2*wal_writer_delay 时间段产生的事务日志(在 wal buffer 中).

wal_sync_method = open_datasync # 使用 pg_test_fsync 测试 wal 所在磁盘的 fsync 接口，使用性能好的.

wal_buffers = 16MB

wal_writer_delay = 10ms

checkpoint_segments = 1024 # 等于 shared_buffers 除以单个 wal segment 的大小.

checkpoint_timeout = 30min

checkpoint_completion_target = 0.2

archive_mode = on # 最好先开启，否则需要重启数据库来修改

archive_command = '/bin/date' # 最好先开启，否则需要重启数据库来修改，将来修改为正确的命令例如，test ! -f /home/postgres/archivedir/pg_root/%f && cp %p /home/postgres/archivedir/pg_root/%f

max_wal_senders = 32 # 最多允许多少个 wal sender 进程.

wal_keep_segments = 2048 # 在 pg_xlog 目录中保留的 WAL 文件数，根据流复制业务的延迟情况和 pg_xlog 目录大小来预估.

max_replication_slots = 32 # 最多允许多少个复制插槽

hot_standby = on

max_standby_archive_delay = 300s # 如果备库要被用于只读，有大的查询的情况下，如果遇到 conflicts，可以考虑调整这个值来避免 conflict 造成 cancel query.

max_standby_streaming_delay = 300s # 如果备库要被用于只读，有大的查询的情况下，如果遇到 conflicts，可以考虑调整这个值来避免 conflict 造成 cancel query.

wal_receiver_status_interval = 1s

hot_standby_feedback = on

random_page_cost = 2 # 根据 IO 能力调整

effective_cache_size = 100GB # 调整为与内存一样大，或者略小(减去 shared_buffer). 用来评估 OS PAGE CACHE 可以用到的内存大小.

log_destination = 'csvlog'

logging_collector = on

log_truncate_on_rotation = on

log_rotation_size = 10MB

log_min_duration_statement = 1s

log_checkpoints = on

log_connections = on

log_disconnections = on

log_error_verbosity = verbose # 在日志中输出代码位置

log_lock_waits = on

log_statement = 'ddl'

autovacuum = on

log_autovacuum_min_duration = 0

autovacuum_max_workers = 10

autovacuum_naptime = 30s # 快速唤醒，防止膨胀

autovacuum_vacuum_scale_factor = 0.02 # 当垃圾超过比例时，启动垃圾回收工作进程

autovacuum_analyze_scale_factor = 0.1

auth_delay.milliseconds = 5000 # 认证失败，延迟多少毫秒反馈

auto_explain.log_min_duration = 5000 # 记录超过多少毫秒的 SQL 当时的执行计划

auto_explain.log_analyze = true

auto_explain.log_verbose = true

auto_explain.log_buffers = true

auto_explain.log_nested_statements = true

pg_stat_statements.track_utility=off

建议的操作系统配置(根据实际情况修改) :

vi /etc/sysctl.conf

add by digoal.zhou

net.ipv4.ip_forward = 0

net.ipv4.conf.default.rp_filter = 1

net.ipv4.conf.default.accept_source_route = 0

kernel.sysrq = 0

kernel.core_uses_pid = 1

net.ipv4.tcp_syncookies = 1

net.bridge.bridge-nf-call-ip6tables = 0

net.bridge.bridge-nf-call-iptables = 0

net.bridge.bridge-nf-call-arptables = 0

kernel.msgmnb = 65536

kernel.msgmax = 65536

kernel.shmmax = 68719476736

kernel.shmall = 4294967296

kernel.shmmni = 4096

kernel.sem = 50100 64128000 50100 1280

fs.file-max = 76724600

net.ipv4.ip_local_port_range = 9000 65000

net.core.rmem_default = 1048576

net.core.rmem_max = 4194304

net.core.wmem_default = 262144

net.core.wmem_max = 1048576

net.ipv4.tcp_tw_recycle = 1

net.ipv4.tcp_max_syn_backlog = 4096

net.core.netdev_max_backlog = 10000

net.ipv4.netfilter.ip_conntrack_max = 655360

fs.aio-max-nr = 1048576

net.ipv4.tcp_timestamps = 0

vm.overcommit_memory = 0

net.ipv4.tcp_tw_recycle=1

net.ipv4.tcp_timestamps=1

net.ipv4.tcp_keepalive_time = 72

net.ipv4.tcp_keepalive_probes = 9

net.ipv4.tcp_keepalive_intvl = 7

vm.zone_reclaim_mode=1

vm.dirty_background_ratio = 10

vm.dirty_background_bytes = 1024000000

vm.dirty_ratio = 60

vm.dirty_bytes = 0

vm.dirty_writeback_centisecs = 500

vm.dirty_expire_centisecs = 3000

vm.swappiness=0

net.ipv4.tcp_syncookies = 0

```
net.ipv4.tcp_tw_reuse = 0
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_fin_timeout = 30
vm.nr_hugepages=102400

vi /etc/security/limits.conf
* soft        nofile    1024000
* hard        nofile    1024000
* soft        nproc     unlimited
* hard        nproc     unlimited
* soft        core      unlimited
* hard        core      unlimited
* soft        memlock unlimited
* hard        memlock unlimited

rm -f /etc/security/limits.d/90-nproc.conf
\n "

echo "----->>>--->>>   用户或数据库级别定制参数: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select * from pg_db_role_setting'
echo "建议: "
echo "       定制参数需要关注，优先级高于数据库的启动参数和配置文件中的参数，特别是排错时需要关注. "
echo -e "\n"

echo
```

```
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "|                                数据库错误日志分析                                |"
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""

echo "----->>>--->>>   获取错误日志信息: "
cat *.csv | grep -E "^[0-9]" | grep -E "WARNING|ERROR|FATAL|PANIC" | awk
-F "," '{print $12" , "$13" , "$14}'|sort|uniq -c|sort -rn
echo "建议: "
echo "      参考 https://images1.tqwba.com/20201125/2g3gdpmebik.html ."
echo -e "\n"

echo "----->>>--->>>   获取连接请求情况: "
find . -name "*.csv" -type f -mtime -28 -exec grep "connection authorized" {}
+|awk -F "," '{print $2,$3,$5}'|sed 's/\:[0-9]*//g'|sort|uniq -c|sort -n -r
echo "建议: "
echo "      连接请求非常多时，请考虑应用层使用连接池，或者使用 pgbouncer
连接池. "
echo -e "\n"

echo "----->>>--->>>   获取认证失败情况: "
find . -name "*.csv" -type f -mtime -28 -exec grep "password authentication
failed" {} +|awk -F "," '{print $2,$3,$5}'|sed 's/\:[0-9]*//g'|sort|uniq -c|sort -n -r
```

```
echo "建议: "
echo "　　认证失败次数很多时，可能是有用户在暴力破解，建议使用 auth_delay 插件防止暴力破解. "
echo -e "\n"


echo "|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++|"
echo "|                                数据库慢 SQL 日志分析                                |"
echo "|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++|"
echo ""

echo "----->>>---->>>  慢查询统计: "
cat *.csv|awk -F "," '{print $1" "$2" "$3" "$8" "$14}' |grep "duration:"|grep -v "plan:"|awk '{print $1" "$4" "$5" "$6}'|sort|uniq -c|sort -rn
echo "建议: "
echo "　　输出格式(条数,日期,用户,数据库,QUERY,耗时 ms). "
echo "　　慢查询反映执行时间超过 log_min_duration_statement 的 SQL，可以根据实际情况分析数据库或 SQL 语句是否有优化空间. "
echo ""
echo "----->>>---->>>  慢查询分布头 10 条的执行时间, ms: "
cat *.csv|awk -F "," '{print $1" "$2" "$3" "$8" "$14}' |grep "duration:"|grep -v "plan:"|awk '{print $1" "$4" "$5" "$6" "$7" "$8}'|sort -k 6 -n|head -n 10
```

```
echo ""
echo "----->>>---->>>   慢查询分布尾 10 条的执行时间, ms: "
cat *.csv|awk -F "," '{print $1" "$2" "$3" "$8" "$14}' |grep "duration:"|grep -v "plan:"|awk '{print $1" "$4" "$5" "$6" "$7" "$8}'|sort -k 6 -n|tail -n 10
echo -e "\n"

echo "----->>>---->>>   auto_explain 分析统计: "
cat *.csv|awk -F "," '{print $1" "$2" "$3" "$8" "$14}' |grep "plan:"|grep "duration:"|awk '{print $1" "$4" "$5" "$6}'|sort|uniq -c|sort -rn
echo "建议: "
echo "       输出格式(条数,日期,用户,数据库,QUERY). "
echo "       慢查询反映执行时间超过 auto_explain.log_min_duration 的 SQL, 可以根据实际情况分析数据库或 SQL 语句是否有优化空间, 分析 csvlog 中 auto_explain 的输出可以了解语句超时时的执行计划详情. "
echo -e "\n"

echo "|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++|"
echo "|                                数据库空间使用分析                                |"
echo "|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++|"
echo ""

echo "----->>>---->>>   输出文件系统剩余空间: "
```

```
df -m
echo "建议: "
echo "      注意预留足够的空间给数据库. "
echo -e "\n"

echo "----->>>---->>>   输出表空间对应目录: "
echo $PGDATA
ls -la $PGDATA/pg_tblspc/
echo "建议: "
echo "      注意表空间如果不是软链接, 注意是否刻意所为, 正常情况下应该是软链接. "
echo -e "\n"

echo "----->>>---->>>   输出表空间使用情况: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select spcname,pg_tablespace_location(oid),pg_size_pretty(pg_tablespace_size(oid)) from pg_tablespace order by pg_tablespace_size(oid) desc'
echo "建议: "
echo "      注意检查表空间所在文件系统的剩余空间, (默认表空间在$PGDATA/base 目录下), IOPS 分配是否均匀, OS 的 sysstat 包可以观察 IO 使用率. "
echo -e "\n"

echo "----->>>---->>>   输出数据库使用情况: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
```

```
--pset=pager=off -q -c 'select datname,pg_size_pretty(pg_database_size(oid)) from pg_database order by pg_database_size(oid) desc'
echo "建议: "
echo "       注意检查数据库的大小, 是否需要清理历史数据. "
echo -e "\n"

echo "----->>>--->>>   TOP 10 size 对象: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),b.nspname,c.relname,c.relkind,pg_size_pretty(pg_relation_size(c.oid)),a.seq_scan,a.seq_tup_read,a.idx_scan,a.idx_tup_fetch,a.n_tup_ins,a.n_tup_upd,a.n_tup_del,a.n_tup_hot_upd,a.n_live_tup,a.n_dead_tup from pg_stat_all_tables a, pg_class c,pg_namespace b where c.relnamespace=b.oid and c.relkind=
$$
r
$$
```

```
 and a.relid=c.oid order by pg_relation_size(c.oid) desc limit 10'
done
echo "建议: "
echo "      经验值: 单表超过 8GB，并且这个表需要频繁更新 或 删除+插入的话，建议对表根据业务逻辑进行合理拆分后获得更好的性能，以及便于对膨胀索引进行维护；如果是只读的表，建议适当结合 SQL 语句进行优化. "
echo -e "\n"


echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "|                                数据库连接分析                                |"
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""

echo "----->>>--->>>   当前活跃度: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select now(),state,count(*) from pg_stat_activity group by 1,2'
echo "建议: "
echo "      如果 active 状态很多，说明数据库比较繁忙. 如果 idle in transaction 很多，说明业务逻辑设计可能有问题. 如果 idle 很多，可能使用了连接池，并且可能没有自动回收连接到连接池的最小连接数. "
```

```
echo -e "\n"

echo "----->>>---->>>  总剩余连接数: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select max_conn,used,res_for_super,max_conn-used-res_for_super res_for_normal from (select count(*) used from pg_stat_activity) t1,(select setting::int res_for_super from pg_settings where name=
$$
superuser_reserved_connections
$$
) t2,(select setting::int max_conn from pg_settings where name=
$$
max_connections
$$
) t3'
echo "建议: "
echo "      给超级用户和普通用户设置足够的连接，以免不能登录数据库. "
echo -e "\n"

echo "----->>>---->>>  用户连接数限制: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select a.rolname,a.rolconnlimit,b.connects from pg_authid a,(select usename,count(*) connects from pg_stat_activity group by usename) b where a.rolname=b.usename order by b.connects desc'
echo "建议: "
echo "      给用户设置足够的连接数, alter role ... CONNECTION LIMIT . "
echo -e "\n"
```

```
echo "----->>>---->>>  数据库连接限制: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select a.datname, a.datconnlimit, b.connects from pg_database a,(select datname,count(*) connects from pg_stat_activity group by datname) b where a.datname=b.datname order by b.connects desc'
echo "建议: "
echo "    给数据库设置足够的连接数, alter database ... CONNECTION LIMIT . "
echo -e "\n"


echo "|+++++++++++++++++++++++++++++++++++++++++++++++++++++++++|"
echo "|                     数据库性能分析                      |"
echo "|+++++++++++++++++++++++++++++++++++++++++++++++++++++++++|"
echo ""

echo "----->>>---->>>  TOP 5 SQL : total_cpu_time "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -x -c 'select c.rolname,b.datname,a.total_time/a.calls per_call_time,a.* from pg_stat_statements a,pg_database b,pg_authid c where a.userid=c.oid and a.dbid=b.oid order by a.total_time desc limit 5'
echo "建议: "
echo "    检查 SQL 是否有优化空间, 配合 auto_explain 插件在 csvlog 中观察
```

```
LONG SQL 的执行计划是否正确. "

echo -e "\n"


echo "----->>>---->>>   索引数超过 4 并且 SIZE 大于 10MB 的表: "

for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(), t2.nspname, t1.relname, pg_size_pretty(pg_relation_size(t1.oid)), t3.idx_cnt from pg_class t1, pg_namespace t2, (select indrelid,count(*) idx_cnt from pg_index group by 1 having count(*)>4) t3 where t1.oid=t3.indrelid and t1.relnamespace=t2.oid and pg_relation_size(t1.oid)/1024/1024.0>10 order by t3.idx_cnt desc'
done

echo "建议: "

echo "      索引数量太多, 影响表的增删改性能, 建议检查是否有不需要的索引. "

echo -e "\n"
```

```
echo "----->>>---->>>  上次巡检以来未使用或使用较少的索引: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select
current_database(),t2.schemaname,t2.relname,t2.indexrelname,t2.idx_scan,t2.idx_tup_read,t2.idx_tup_fetch,pg_size_pretty(pg_relation_size(indexrelid))
from pg_stat_all_tables t1,pg_stat_all_indexes t2 where t1.relid=t2.relid and t2.idx_scan<10 and t2.schemaname not in (
$$
pg_toast
$$
,
$$
pg_catalog
$$
) and indexrelid not in (select conindid from pg_constraint where contype in (
$$
p
```

```
$$
,
$$
u
$$
,
$$
f
$$
)) and pg_relation_size(indexrelid)>65536 order by pg_relation_size(indexrelid)
desc'
done
echo "建议: "
echo "      建议和应用开发人员确认后, 删除不需要的索引. "
echo -e "\n"

echo "----->>>---->>>  数据库统计信息, 回滚比例, 命中比例, 数据块读写时
间, 死锁, 复制冲突: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
--pset=pager=off -q -c 'select
datname,round(100*(xact_rollback::numeric/(case when xact_commit > 0
then xact_commit else 1 end + xact_rollback)),2)||
$$
 %
$$
 rollback_ratio, round(100*(blks_hit::numeric/(case when blks_read>0 then
blks_read else 1 end + blks_hit)),2)||
$$
```

%

$$

hit_ratio, blk_read_time, blk_write_time, conflicts, deadlocks from pg_stat_database'

echo "建议: "

echo "　　回滚比例大说明业务逻辑可能有问题，命中率小说明 shared_buffer 要加大，数据块读写时间长说明块设备的 IO 性能要提升，死锁次数多说明业务逻辑有问题，复制冲突次数多说明备库可能在跑 LONG SQL. "

echo -e "\n"

echo "----->>>---->>>　检查点, bgwriter 统计信息: "

psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -x -c 'select * from pg_stat_bgwriter'

echo "建议: "

echo "　　checkpoint_write_time 多说明检查点持续时间长，检查点过程中产生了较多的脏页. "

echo "　　checkpoint_sync_time 代表检查点开始时的 shared buffer 中的脏页被同步到磁盘的时间，如果时间过长，并且数据库在检查点时性能较差，考虑一下提升块设备的 IOPS 能力. "

echo "　　buffers_backend_fsync 太多说明需要加大 shared buffer 或者 减小 bgwriter_delay 参数. "

echo -e "\n"

echo

```
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "|                                数据库垃圾分析                                |"
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""

echo "----->>>---->>>   表引膨胀检查: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
--pset=pager=off -t -A -q -c 'select datname from pg_database where
datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -x
-c 'SELECT
  current_database() AS db, schemaname, tablename, reltuples::bigint AS
tups, relpages::bigint AS pages, otta,
  ROUND(CASE WHEN otta=0 OR sml.relpages=0 OR sml.relpages=otta
THEN 0.0 ELSE sml.relpages/otta::numeric END,1) AS tbloat,
  CASE WHEN relpages < otta THEN 0 ELSE relpages::bigint - otta END AS
wastedpages,
```

```
  CASE WHEN relpages < otta THEN 0 ELSE bs*(sml.relpages-otta)::bigint
END AS wastedbytes,
  CASE WHEN relpages < otta THEN
$$
0 bytes
$$
::text ELSE (bs*(relpages-otta))::bigint ||
$$
 bytes
$$
 END AS wastedsize,
  iname, ituples::bigint AS itups, ipages::bigint AS ipages, iotta,
  ROUND(CASE WHEN iotta=0 OR ipages=0 OR ipages=iotta THEN 0.0
ELSE ipages/iotta::numeric END,1) AS ibloat,
  CASE WHEN ipages < iotta THEN 0 ELSE ipages::bigint - iotta END AS
wastedipages,
  CASE WHEN ipages < iotta THEN 0 ELSE bs*(ipages-iotta) END AS
wastedibytes,
  CASE WHEN ipages < iotta THEN
$$
0 bytes
$$
 ELSE (bs*(ipages-iotta))::bigint ||
$$
 bytes
$$
 END AS wastedisize,
  CASE WHEN relpages < otta THEN
    CASE WHEN ipages < iotta THEN 0 ELSE bs*(ipages-iotta::bigint) END
    ELSE CASE WHEN ipages < iotta THEN bs*(relpages-otta::bigint)
```

```
        ELSE bs*(relpages-otta::bigint + ipages-iotta::bigint) END
    END AS totalwastedbytes
FROM (
    SELECT
        nn.nspname AS schemaname,
        cc.relname AS tablename,
        COALESCE(cc.reltuples,0) AS reltuples,
        COALESCE(cc.relpages,0) AS relpages,
        COALESCE(bs,0) AS bs,
        COALESCE(CEIL((cc.reltuples*((datahdr+ma-
            (CASE WHEN datahdr%ma=0 THEN ma ELSE datahdr%ma
END))+nullhdr2+4))/(bs-20::float)),0) AS otta,
        COALESCE(c2.relname,
$$
?
$$
) AS iname, COALESCE(c2.reltuples,0) AS ituples, COALESCE(c2.relpages,0)
AS ipages,
        COALESCE(CEIL((c2.reltuples*(datahdr-12))/(bs-20::float)),0) AS iotta --
very rough approximation, assumes all cols
    FROM
         pg_class cc
    JOIN pg_namespace nn ON cc.relnamespace = nn.oid AND nn.nspname
<>
$$
information_schema
$$

    LEFT JOIN
    (
```

```
    SELECT
      ma,bs,foo.nspname,foo.relname,
      (datawidth+(hdr+ma-(case when hdr%ma=0 THEN ma ELSE hdr%ma
END)))::numeric AS datahdr,
      (maxfracsum*(nullhdr+ma-(case when nullhdr%ma=0 THEN ma ELSE
nullhdr%ma END))) AS nullhdr2
    FROM (
      SELECT
        ns.nspname, tbl.relname, hdr, ma, bs,
        SUM((1-coalesce(null_frac,0))*coalesce(avg_width,      2048))      AS
datawidth,
        MAX(coalesce(null_frac,0)) AS maxfracsum,
        hdr+(
          SELECT 1+count(*)/8
          FROM pg_stats s2
          WHERE null_frac<>0 AND s2.schemaname = ns.nspname AND
s2.tablename = tbl.relname
        ) AS nullhdr
      FROM pg_attribute att
      JOIN pg_class tbl ON att.attrelid = tbl.oid
      JOIN pg_namespace ns ON ns.oid = tbl.relnamespace
      LEFT JOIN pg_stats s ON s.schemaname=ns.nspname
      AND s.tablename = tbl.relname
      AND s.inherited=false
      AND s.attname=att.attname,
      (
        SELECT
          (SELECT current_setting(
$$
block_size
```

$$
)::numeric) AS bs,
CASE WHEN SUBSTRING(SPLIT_PART(v,
$$

$$
, 2) FROM
$$
#"[0-9]+.[0-9]+#"%
$$
for
$$
#
$$
)
IN (
$$
8.0
$$
,
$$
8.1
$$
,
$$
8.2
$$
) THEN 27 ELSE 23 END AS hdr,
CASE WHEN v ~
$$

```
mingw32
$$
 OR v ~
$$
64-bit
$$
 THEN 8 ELSE 4 END AS ma
            FROM (SELECT version() AS v) AS foo
         ) AS constants
         WHERE att.attnum > 0 AND tbl.relkind=
$$
r
$$


         GROUP BY 1,2,3,4,5
       ) AS foo
   ) AS rs
   ON cc.relname = rs.relname AND nn.nspname = rs.nspname
   LEFT JOIN pg_index i ON indrelid = cc.oid
   LEFT JOIN pg_class c2 ON c2.oid = i.indexrelid
) AS sml order by wastedbytes desc limit 5'
done
echo "建议: "
echo "      根据浪费的字节数, 设置合适的 autovacuum_vacuum_scale_factor, 大表如果频繁的有更新或删除和插入操作, 建议设置较小的 autovacuum_vacuum_scale_factor 来降低浪费空间. "
echo "      同时还需要打开 autovacuum, 根据服务器的内存大小, CPU 核数, 设置足够大的 autovacuum_work_mem 或 autovacuum_max_workers 或
```

```
maintenance_work_mem, 以及足够小的 autovacuum_naptime . "
echo "       同时还需要分析是否对大数据库使用了逻辑备份 pg_dump, 系统中是否经常有长 SQL, 长事务. 这些都有可能导致膨胀. "
echo "       使用 pg_reorg 或者 vacuum full 可以回收膨胀的空间. "
echo "                                参          考          : http://blog.163.com/digoal@126/blog/static/16387704020153291156362 87/ "
echo -e "\n"


echo "----->>>---->>>   索引膨胀检查: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -x -c 'SELECT
  current_database() AS db, schemaname, tablename, reltuples::bigint AS tups, relpages::bigint AS pages, otta,
  ROUND(CASE WHEN otta=0 OR sml.relpages=0 OR sml.relpages=otta
```

```
THEN 0.0 ELSE sml.relpages/otta::numeric END,1) AS tbloat,
   CASE WHEN relpages < otta THEN 0 ELSE relpages::bigint - otta END AS
wastedpages,
   CASE WHEN relpages < otta THEN 0 ELSE bs*(sml.relpages-otta)::bigint
END AS wastedbytes,
   CASE WHEN relpages < otta THEN
$$
0 bytes
$$
::text ELSE (bs*(relpages-otta))::bigint ||
$$
 bytes
$$
 END AS wastedsize,
   iname, ituples::bigint AS itups, ipages::bigint AS ipages, iotta,
   ROUND(CASE WHEN iotta=0 OR ipages=0 OR ipages=iotta THEN 0.0
ELSE ipages/iotta::numeric END,1) AS ibloat,
   CASE WHEN ipages < iotta THEN 0 ELSE ipages::bigint - iotta END AS
wastedipages,
   CASE WHEN ipages < iotta THEN 0 ELSE bs*(ipages-iotta) END AS
wastedibytes,
   CASE WHEN ipages < iotta THEN
$$
0 bytes
$$
 ELSE (bs*(ipages-iotta))::bigint ||
$$
 bytes
$$
 END AS wastedisize,
```

```
  CASE WHEN relpages < otta THEN
    CASE WHEN ipages < iotta THEN 0 ELSE bs*(ipages-iotta::bigint) END
    ELSE CASE WHEN ipages < iotta THEN bs*(relpages-otta::bigint)
      ELSE bs*(relpages-otta::bigint + ipages-iotta::bigint) END
  END AS totalwastedbytes
FROM (
  SELECT
    nn.nspname AS schemaname,
    cc.relname AS tablename,
    COALESCE(cc.reltuples,0) AS reltuples,
    COALESCE(cc.relpages,0) AS relpages,
    COALESCE(bs,0) AS bs,
    COALESCE(CEIL((cc.reltuples*((datahdr+ma-
      (CASE WHEN datahdr%ma=0 THEN ma ELSE datahdr%ma
END))+nullhdr2+4))/(bs-20::float)),0) AS otta,
    COALESCE(c2.relname,
$$
?
$$
) AS iname, COALESCE(c2.reltuples,0) AS ituples, COALESCE(c2.relpages,0)
AS ipages,
    COALESCE(CEIL((c2.reltuples*(datahdr-12))/(bs-20::float)),0) AS iotta --
very rough approximation, assumes all cols
  FROM
      pg_class cc
  JOIN pg_namespace nn ON cc.relnamespace = nn.oid AND nn.nspname
<>
$$
information_schema
$$
```

```
  LEFT JOIN
  (
    SELECT
      ma,bs,foo.nspname,foo.relname,
      (datawidth+(hdr+ma-(case when hdr%ma=0 THEN ma ELSE hdr%ma END)))::numeric AS datahdr,
      (maxfracsum*(nullhdr+ma-(case when nullhdr%ma=0 THEN ma ELSE nullhdr%ma END))) AS nullhdr2
    FROM (
      SELECT
        ns.nspname, tbl.relname, hdr, ma, bs,
        SUM((1-coalesce(null_frac,0))*coalesce(avg_width, 2048)) AS datawidth,
        MAX(coalesce(null_frac,0)) AS maxfracsum,
        hdr+(
          SELECT 1+count(*)/8
          FROM pg_stats s2
          WHERE null_frac<>0 AND s2.schemaname = ns.nspname AND s2.tablename = tbl.relname
        ) AS nullhdr
      FROM pg_attribute att
      JOIN pg_class tbl ON att.attrelid = tbl.oid
      JOIN pg_namespace ns ON ns.oid = tbl.relnamespace
      LEFT JOIN pg_stats s ON s.schemaname=ns.nspname
      AND s.tablename = tbl.relname
      AND s.inherited=false
      AND s.attname=att.attname,
      (
        SELECT
```

```
                (SELECT current_setting(
$$
block_size
$$
)::numeric) AS bs,
                    CASE WHEN SUBSTRING(SPLIT_PART(v,
$$


$$
, 2) FROM
$$
#"[0-9]+.[0-9]+#"%
$$
  for
$$
#
$$
)
                        IN (
$$
8.0
$$
,
$$
8.1
$$
,
$$
8.2
$$
```

```
) THEN 27 ELSE 23 END AS hdr,
                CASE WHEN v ~
$$
mingw32
$$
  OR v ~
$$
64-bit
$$
  THEN 8 ELSE 4 END AS ma
             FROM (SELECT version() AS v) AS foo
          ) AS constants
          WHERE att.attnum > 0 AND tbl.relkind=
$$
r
$$


          GROUP BY 1,2,3,4,5
       ) AS foo
    ) AS rs
    ON cc.relname = rs.relname AND nn.nspname = rs.nspname
    LEFT JOIN pg_index i ON indrelid = cc.oid
    LEFT JOIN pg_class c2 ON c2.oid = i.indexrelid
) AS sml order by wastedibytes desc limit 5'
done
echo "建议: "
echo "        如果索引膨胀太大，会影响性能，建议重建索引, create index
CONCURRENTLY ... . "
echo -e "\n"
```

```
echo "----->>>---->>>  垃圾数据: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),schemaname,relname,n_dead_tup from pg_stat_all_tables where n_live_tup>0 and n_dead_tup/n_live_tup>0.2 and schemaname not in (
$$
pg_toast
$$
,
$$
pg_catalog
$$
) order by n_dead_tup desc limit 5'
done
echo "建议: "
echo "      通常垃圾过多，可能是因为无法回收垃圾，或者回收垃圾的进程繁忙
```

或没有及时唤醒，或者没有开启 autovacuum，或在短时间内产生了大量的垃圾 . "

echo "　　可以等待 autovacuum 进行处理，或者手工执行 vacuum table . "

echo -e "\n"

echo "----->>>---->>>　未引用的大对象: "

```
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
vacuumlo -n -h $PGHOST -p $PGPORT -U $PGUSER $db -w
echo ""
done
```

echo "建议: "

echo "　　如果大对象没有被引用时，建议删除，否则就类似于内存泄露，使用 vacuumlo 可以删除未被引用的大对象，例如: vacuumlo -l 1000 -h $PGHOST -p $PGPORT -U $PGUSER $db -w . "

echo "　　应用开发时，注意及时删除不需要使用的大对象，使用 lo_unlink 或驱动对应的 API . "

```
echo "      参考 https://images1.tqwba.com/20201125/gd32pfueri5.html "
echo -e "\n"


echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "|                                数据库年龄分析                                |"
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""

echo "----->>>---->>>   数据库年龄: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
--pset=pager=off -q -c 'select
datname,age(datfrozenxid),2^31-age(datfrozenxid) age_remain from
pg_database order by age(datfrozenxid) desc'
echo "建议: "
echo "      数据库的年龄正常情况下应该小于 vacuum_freeze_table_age，如果
剩余年龄小于 5 亿，建议人为干预，将 LONG SQL 或事务杀掉后，执行 vacuum
freeze . "
echo -e "\n"

echo "----->>>---->>>   表年龄: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER --pset=pager=off -d
$PGDATABASE -t -A -q -c 'select datname from pg_database where
```

```
datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),rolname,nspname,relkind,relname,age(relfrozenxid),2^31-age(relfrozenxid) age_remain from pg_authid t1 join pg_class t2 on t1.oid=t2.relowner join pg_namespace t3 on t2.relnamespace=t3.oid where t2.relkind in (
$$
t
$$
,
$$
r
$$
) order by age(relfrozenxid) desc limit 5'
done
echo "建议: "
echo "　　表的年龄正常情况下应该小于 vacuum_freeze_table_age，如果剩余年龄小于 5 亿，建议人为干预，将 LONG SQL 或事务杀掉后，执行 vacuum freeze . "
```

```
echo -e "\n"

echo "----->>>---->>>  长事务, 2PC: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -x -c 'select datname,usename,query,xact_start,now()-xact_start xact_duration,query_start,now()-query_start query_duration,state from pg_stat_activity where state<>
$$
idle
$$
 and (backend_xid is not null or backend_xmin is not null) and now()-xact_start > interval
$$
30 min
$$
 order by xact_start'
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -x -c 'select name,statement,prepare_time,now()-prepare_time,parameter_types,from_sql from pg_prepared_statements where now()-prepare_time > interval
$$
30 min
$$
 order by prepare_time'
echo "建议: "
echo "      长事务过程中产生的垃圾，无法回收，建议不要在数据库中运行 LONG SQL，或者错开 DML 高峰时间去运行 LONG SQL. 2PC 事务一定要记得
```

```
尽快结束掉，否则可能会导致数据库膨胀. "
echo " 参 考 :
http://blog.163.com/digoal@126/blog/static/16387704020153291156362 87/
"
echo -e "\n"


echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "| 数据库 XLOG，流复制状态分析 |"
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""

echo "----->>>---->>> 是否开启归档，自动垃圾回收: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
--pset=pager=off -q -c 'select name,setting from pg_settings where name in (
$$
archive_mode
$$
,
$$
autovacuum
$$
,
$$
```

```
archive_command
$$
)'
echo "建议: "
echo "       建议开启自动垃圾回收，开启归档. "
echo -e "\n"

echo "----->>>--->>>   归档统计信息: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select pg_xlogfile_name(pg_current_xlog_location()) now_xlog, * from pg_stat_archiver'
echo "建议: "
echo "       如果当前的 XLOG 文件和最后一个归档失败的 XLOG 文件之间相差很多个文件，建议尽快排查归档失败的原因，以便修复，否则 pg_xlog 目录可能会撑爆. "
echo -e "\n"

echo "----->>>--->>>   流复制统计信息: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -x -c 'select pg_xlog_location_diff(pg_current_xlog_location(),flush_location), * from pg_stat_replication'
echo "建议: "
echo "       关注流复制的延迟，如果延迟非常大，建议排查网络带宽，以及本地读 xlog 的性能，远程写 xlog 的性能. "
echo -e "\n"
```

echo "----->>>---->>>　流复制插槽: "

psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select pg_xlog_location_diff(pg_current_xlog_location(),restart_lsn), * from pg_replication_slots'

echo "建议: "

echo "　　如果 restart_lsn 和当前 XLOG 相差非常大的字节数，需要排查 slot 的订阅者是否能正常接收 XLOG，或者订阅者是否正常. 长时间不将 slot 的数据取走, pg_xlog 目录可能会撑爆. "

echo -e "\n"

echo "|+++|"

echo "|　　　　　　　　　数据库安全或潜在风险分析　　　　　　　　　|"

echo "|+++|"

echo ""

echo "----->>>---->>>　密码泄露检查: "

echo "　　检查 ~/.psql_history :　"

grep -i "password" ~/.psql_history|grep -i -E "role|group|user"

echo ""

echo "　　检查 *.csv :　"

```
cat *.csv | grep -E "^[0-9]" | grep -i -r -E "role|group|user" |grep -i "password"|grep -i -E "create|alter"
echo ""
echo "      检查 $PGDATA/recovery.* :   "
grep -i "password" ../recovery.*
echo ""
echo "      检查 pg_stat_statements :   "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -c 'select query from pg_stat_statements where (query ~*
$$
group
$$
 or query ~*
$$
user
$$
 or query ~*
$$
role
$$
) and query ~*
$$
password
$$
'
echo "      检查 pg_authid :   "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select * from pg_authid where rolpassword !~
$$
```

```
^md5
$$
 or length(rolpassword)<>35'
echo "       检查 pg_user_mappings, pg_views :    "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -c 'select current_database(),* from pg_user_mappings where umoptions::text ~*
$$
password
$$
'
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -c 'select current_database(),* from pg_views where definition ~*
$$
password
$$
 and definition ~*
```

```
$$
dblink
$$
'
done
echo "建议: "
echo "      如果以上输出显示密码已泄露，尽快修改，并通过参数避免密码又被记录到以上文件中 (psql -n) (set log_statement='none'; set log_min_duration_statement=-1; set log_duration=off; set pg_stat_statements.track_utility=off;) . "
echo "      明文密码不安全，建议使用 create|alter role ... encrypted password. "
echo "      在 fdw, dblink based view 中不建议使用密码明文. "
echo "      在 recovery.*的配置中不要使用密码，不安全，可以使用.pgpass 配置密码 . "
echo -e "\n"

echo "----->>>--->>>  简单密码检查: "
echo "      1. 检查已有密码是否简单，从 crackdb 库提取密码字典，挨个检查 : "
echo "       检查 md5('$pwd'||'$username')是否与 pg_authid.rolpassword 匹配 :  "
echo "      匹配则说明用户使用了简单密码 :  "
echo ""
echo "                  2. 事前检查参考
```

```
http://blog.163.com/digoal@126/blog/static/163877040201498529415 86"
echo -e "\n"

echo "----->>>---->>>  用户密码到期时间: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c 'select rolname,rolvaliduntil from pg_authid order by rolvaliduntil'
echo "建议: "
echo "      到期后，用户将无法登陆，记得修改密码，同时将密码到期时间延长到某个时间或无限时间, alter role ... VALID UNTIL 'timestamp' . "
echo -e "\n"

echo "----->>>---->>>  SQL 注入风险分析: "
cat *.csv | grep -E "^[0-9]" | grep exec_simple_query |awk -F "," '{print $2" "$3" "$5" "$NF}'|sed 's/\:[0-9]*//g'|sort|uniq -c|sort -n -r
echo "建议: "
echo "      调用 exec_simple_query 有风险，允许多个 SQL 封装在一个接口中调用，建议程序使用绑定变量规避 SQL 注入风险，或者程序端使用 SQL 注入过滤插件. "
echo -e "\n"

echo "----->>>---->>>  普通用户对象上的规则安全检查: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
```

```
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -c 'select current_database(),a.schemaname,a.tablename,a.rulename,a.definition from pg_rules a,pg_namespace b,pg_class c,pg_authid d where a.schemaname=b.nspname and a.tablename=c.relname and d.oid=c.relowner and not d.rolsuper union all select current_database(),a.schemaname,a.viewname,a.viewowner,a.definition from pg_views a,pg_namespace b,pg_class c,pg_authid d where a.schemaname=b.nspname and a.viewname=c.relname and d.oid=c.relowner and not d.rolsuper'
done
echo "建议: "
echo "　　防止普通用户在规则中设陷阱，注意有危险的 security invoker 的函数调用，超级用户可能因为规则触发后误调用这些危险函数(以 invoker 角色). "
echo "　　参　　考　　http://blog.163.com/digoal@126/blog/static/16387704020155131217736/ "
echo -e "\n"

echo "----->>>---->>>　普通用户自定义函数安全检查: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
```

```
--pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -c 'select current_database(),b.rolname,c.nspname,a.proname from pg_proc a,pg_authid b,pg_namespace c where a.proowner=b.oid and a.pronamespace=c.oid and not b.rolsuper and not a.prosecdef'
done
echo "建议: "
echo "　　防止普通用户在函数中设陷阱，注意有危险的 security invoker 的函数调用，超级用户可能因为触发器触发后误调用这些危险函数(以 invoker 角色). "
echo "　　参　　考　　http://blog.163.com/digoal@126/blog/static/16387704020155131217736/ "
echo -e "\n"

echo "----->>>---->>>　unlogged table 和 哈希索引: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
```

```
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c
'select current_database(),t3.rolname,t2.nspname,t1.relname from pg_class
t1,pg_namespace t2,pg_authid t3 where t1.relnamespace=t2.oid and
t1.relowner=t3.oid and t1.relpersistence=
$$
u
$$
'
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c
'select current_database(),pg_get_indexdef(oid) from pg_class where relkind=
$$
i
$$
  and pg_get_indexdef(oid) ~
$$
USING hash
$$
'
done
echo "建议: "
```

```
echo "      unlogged table 和 hash index 不记录 XLOG，无法使用流复制或者 log shipping 的方式复制到 standby 节点，如果在 standby 节点执行某些 SQL，可能导致报错或查不到数据. "
echo "      在数据库 CRASH 后无法修复 unlogged table 和 hash index，不建议使用. "
echo "      PITR 对 unlogged table 和 hash index 也不起作用. "
echo -e "\n"

echo "----->>>---->>>  剩余可使用次数不足 1000 万次的序列检查: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off <<EOF
create or replace function f(OUT v_datname name, OUT v_role name, OUT v_nspname name, OUT v_relname name, OUT v_times_remain int8) returns setof record as \$\$
declare
```

```
begin
  v_datname := current_database();
  for v_role,v_nspname,v_relname in select rolname,nspname,relname from pg_authid t1 , pg_class t2 , pg_namespace t3 where t1.oid=t2.relowner and t2.relnamespace=t3.oid and t2.relkind='S'
  LOOP
    execute 'select (max_value-last_value)/increment_by from "'||v_nspname||'"."'||v_relname||'" where not is_cycled' into v_times_remain;
    return next;
  end loop;
end;
\$\$ language plpgsql;

select * from f() where v_times_remain is not null and v_times_remain < 10240000 order by v_times_remain limit 10;
EOF
done
echo "建议: "
echo "      序列剩余使用次数到了之后，将无法使用，报错，请开发人员关注. "
echo -e "\n"

echo "----->>>---->>>  触发器，事件触发器: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
```

```
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),relname,tgname,proname,tgenabled from pg_trigger t1,pg_class t2,pg_proc t3 where t1.tgfoid=t3.oid and t1.tgrelid=t2.oid'
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select
current_database(),rolname,proname,evtname,evtevent,evtenabled,evttags
from pg_event_trigger t1,pg_proc t2,pg_authid t3 where t1.evtfoid=t2.oid and t1.evtowner=t3.oid'
done
echo "建议: "
echo "      请管理员注意触发器和事件触发器的必要性. "
echo -e "\n"

echo "----->>>---->>>  检查是否使用了 a-z 0-9 _ 以外的字母作为对象名: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -c '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'
```

```
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -t -A -q -c 'select datname from pg_database where datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),relname,relkind from (select relname,relkind,regexp_split_to_table(relname,
$$

$$
) word from pg_class) t where (not (ascii(word) >=97 and ascii(word) <=122)) and (not (ascii(word) >=48 and ascii(word) <=57)) and ascii(word)<>95 group by 1,2,3'
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(), typname from (select typname,regexp_split_to_table(typname,
$$

$$
) 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'
```

```
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(), proname from (select proname,regexp_split_to_table(proname,
$$

$$
) word from pg_proc where proname !~
$$
^RI_FKey_
$$
) t where (not (ascii(word) >=97 and ascii(word) <=122)) and (not (ascii(word) >=48 and ascii(word) <=57)) and ascii(word)<>95 group by 1,2'
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c 'select current_database(),nspname,relname,attname from (select nspname,relname,attname,regexp_split_to_table(attname,
$$

$$
) 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'
done
echo "建议: "
echo "      建议任何 identify 都只使用  a-z, 0-9, _ (例如表名，列名，视图名，函数名，类型名，数据库名, schema 名，物化视图名等等). "
echo -e "\n"
```

```
echo "----->>>---->>>  锁等待: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE -x --pset=pager=off <<EOF
create or replace function f_lock_level(i_mode text) returns int as \$\$
declare
begin
   case i_mode
      when 'INVALID' then return 0;
      when 'AccessShareLock' then return 1;
      when 'RowShareLock' then return 2;
      when 'RowExclusiveLock' then return 3;
      when 'ShareUpdateExclusiveLock' then return 4;
      when 'ShareLock' then return 5;
      when 'ShareRowExclusiveLock' then return 6;
      when 'ExclusiveLock' then return 7;
      when 'AccessExclusiveLock' then return 8;
      else return 0;
   end case;
end;
\$\$ language plpgsql strict;

with t_wait as
(select
a.mode,a.locktype,a.database,a.relation,a.page,a.tuple,a.classid,a.objid,a.objsubid,a.pid,a.virtualtransaction,a.virtualxid,a,transactionid,b.query,b.xact_start,b.query_start,b.usename,b.datname from pg_locks a,pg_stat_activity b where a.pid=b.pid and not a.granted),
t_run as
(select
```

```
a.mode,a.locktype,a.database,a.relation,a.page,a.tuple,a.classid,a.objid,a.obj
subid,a.pid,a.virtualtransaction,a.virtualxid,a,transactionid,b.query,b.xact_sta
rt,b.query_start,b.usename,b.datname from pg_locks a,pg_stat_activity b
where a.pid=b.pid and a.granted)
select r.locktype,r.mode r_mode,r.usename r_user,r.datname
r_db,r.relation::regclass,r.pid r_pid,r.xact_start r_xact_start,r.query_start
r_query_start,now()-r.query_start r_locktime,r.query r_query,
w.mode w_mode,w.pid w_pid,w.xact_start w_xact_start,w.query_start
w_query_start,now()-w.query_start w_locktime,w.query w_query
from t_wait w,t_run r where
   r.locktype is not distinct from w.locktype and
   r.database is not distinct from w.database and
   r.relation is not distinct from w.relation and
   r.page is not distinct from w.page and
   r.tuple is not distinct from w.tuple and
   r.classid is not distinct from w.classid and
   r.objid is not distinct from w.objid and
   r.objsubid is not distinct from w.objsubid and
   r.transactionid is not distinct from w.transactionid and
   r.pid <> w.pid
   order by f_lock_level(w.mode)+f_lock_level(r.mode) desc,r.xact_start;
EOF
echo "建议: "
echo "      锁等待状态，反映业务逻辑的问题或者 SQL 性能有问题，建议深入
排查持锁的 SQL. "
echo -e "\n"

echo "----->>>---->>>   继承关系检查: "
```

```
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
--pset=pager=off -t -A -q -c 'select datname from pg_database where
datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -q -c
'select inhrelid::regclass,inhparent::regclass,inhseqno from pg_inherits order
by 2,3'
done
echo "建议: "
echo "        如果使用继承来实现分区表，注意分区表的触发器中逻辑是否正常，
对于时间模式的分区表是否需要及时加分区，修改触发器函数 . "
echo "        建议继承表的权限统一，如果权限不一致，可能导致某些用户查询时
权限不足. "
echo -e "\n"


echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "|                          重置统计信息                          |"
```

```
echo
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""

echo "----->>>---->>>  重置统计信息: "
for db in `psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
--pset=pager=off -t -A -q -c 'select datname from pg_database where
datname not in (
$$
template0
$$
,
$$
template1
$$
)'`
do
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $db --pset=pager=off -c
'select pg_stat_reset()'
done
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
--pset=pager=off -c 'select pg_stat_reset_shared(
$$
bgwriter
$$
)'
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE
--pset=pager=off -c 'select pg_stat_reset_shared(
```

```
$$
archiver
$$
)'

echo "----->>>---->>>  重置 pg_stat_statements 统计信息: "
psql -h $PGHOST -p $PGPORT -U $PGUSER -d $PGDATABASE --pset=pager=off -q -A -c 'select pg_stat_statements_reset()'

}  # common function end


primary() {
echo "----->>>---->>>  获取 recovery.done md5 值: "
md5sum $PGDATA/recovery.done
echo "建议: "
echo "      主备 md5 值一致(判断主备配置文件是否内容一致的一种手段，或者使用 diff)."
echo -e "\n"

echo "----->>>---->>>  获取 recovery.done 配置: "
grep '^\ *[a-z]' $PGDATA/recovery.done|awk -F "#" '{print $1}'
echo "建议: "
echo "      在 primary_conninfo 中不要配置密码，容易泄露．建议为流复制用户创建 replication 角色的用户，并且配置 pg_hba.conf 只允许需要的来源 IP 连接．"
echo -e "\n"
```

```
}  # primary function end


standby() {
echo "----->>>---->>>  获取 recovery.conf md5 值: "
md5sum $PGDATA/recovery.conf
echo "建议: "
echo "    主备 md5 值一致(判断主备配置文件是否内容一致的一种手段，或者
使用 diff)."
echo -e "\n"

echo "----->>>---->>>  获取 recovery.conf 配置: "
grep '^\ *[a-z]' $PGDATA/recovery.conf|awk -F "#" '{print $1}'
echo "建议: "
echo "    在 primary_conninfo 中不要配置密码，容易泄露. 建议为流复制用户
创建 replication 角色的用户，并且配置 pg_hba.conf 只允许需要的来源 IP 连接.
"
echo -e "\n"
}  # standby function end


adds() {
echo
"|+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo "|                    附加信息                    |"
echo
```

```
"|++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+|"
echo ""

echo "----->>>---->>>   附件 1 : `date -d '-1 day' +"%Y-%m-%d"` 操作系统
sysstat 收集的统计信息 "
sar -A -f /var/log/sa/sa`date -d '-1 day' +%d`
echo -e "\n"

echo "----->>>---->>>   其他建议: "
echo "      其他建议的巡检项: "
echo "            HA 状态是否正常，例如检查 HA 程序，检查心跳表的延迟. "
echo "            sar io, load, ..... "
echo "      巡检结束后，清理 csv 日志 "
}   # adds function end


if [ $is_standby == 't' ]; then
standby
else
primary
fi

common
adds
cd $pwd
return 0
```

备注

csv 日志分析需要优化

某些操作需要 root