GBase 8s

管理员手册

GBase 8s 管理员手册，南大通用数据技术股份有限公司

版权声明

免责声明

通讯方式

南大通用数据技术股份有限公司

天津市高新区华苑产业园区开华道22号普天创新园东塔20-23层(300384)

电话：400-013-9696　　邮箱：info@gbase.cn

商标声明

目　录

1 数据库管理

数据库管理员可以通过 gha_ctl 工具（详见《GBase 8s V8.8_工具参考手册》中 gha_ctl 章节），对 GBase 8s 数据库进行启动、停止等操作。

1.1 启动

语法格式

启动 GBase 8s 数据库集群的语法格式为：

```
gha_ctl start all -l dcslist [-c cluster]
gha_ctl start datanode | server -l dcslist [-c cluster] [-t timeout]
```

参数说明

- all | datanode | server：必选字段。指定启动全部节点或某类型节点。
- -l：必选字段。指定 DCS 地址，格式为 http://<dcs_IP>:<port>。一般情况下，可以只列出一个节点地址，其他节点会自动同步消息。为了保证高可用，也可以列出所有节点地址。
- -c 参数：可选字段。指定数据库集群名称。缺省默认值 gbase。
- -t 参数：可选字段。指定等待数据库启动的超时时间。缺省默认值为 30min。

示例

名为 gbase 的集群中一个 DCS 节点 IP 为 10.0.7.16，可使用以下命令启动数据库集群：

```
[gbase@gbase8s ~]$ gha_ctl start all -l http://10.0.7.16:2379 -c gbase
```

返回如下信息，则说明启动操作成功。

```
{
    "ret":0,
    "msg":"Success"
}
```

1.2 停止

语法格式

停止 GBase 8s 数据库集群的语法格式为：

```
gha_ctl stop all -l dcslist [-c cluster]
gha_ctl stop datanode | server -l dcslist [-c cluster] [-t timeout]
```

参数说明

- all | datanode | server：必选字段。指定停止全部节点或某类型节点。
- -l：必选字段。指定 DCS 地址，格式为 http://<dcs_IP>:<port>。一般情况下，可以只列出一个节点地址，其他节点会自动同步消息。为了保证高可用，也可以列出所有节点地址。
- -c 参数：可选字段。指定数据库集群名称。缺省默认值 gbase。
- -t 参数：可选字段。指定等待数据库停止的超时时间。缺省默认值为 30min。

示例

名为 gbase 的集群中一个 DCS 节点 IP 为 10.0.7.16，可使用以下命令停止数据库集群：

```
[gbase@gbase8s ~]$ gha_ctl stop all -l http://10.0.7.16:2379 -c gbase
```

返回如下信息，则说明停止操作成功。

```
{
    "ret":0,
    "msg":"Success"
}
```

2 状态查询

GBase 8s 数据库管理员可使用 gha_ctl monitor 命令，来查询数据库集群的运行状态。

语法格式

```
gha_ctl monitor  all | datanode | server| dcs [-H] -l dcslist [-c cluster]
```

其中参数说明：

- dcslist、[-c cluster]参数与上述相同；
- all | datanode | server|dcs：必选字段。指定查看哪类集群节点或全部。
- -H：可选字段。指定返回信息是否以表格形式显示。缺省默认为文件命令行形式。

示例

名为 gbase 的集群中一个 DCS 节点的 IP 为 10.0.7.16，则具体操作命令如下：

```
[gbase@gbase8s ~]$ gha_ctl monitor all -l http://10.0.7.16:2379
```

返回运行中为成功：

```
{
    "cluster": "gbase8s",
    "version": "3.0.0B51",
    "server": [
        {
            "name": "gha_server1",
            "host": "10.0.7.16",
            "port": "20001",
            "state": "running",
            "isLeader": true
        }
    ],
    "datanode": {
        "dn1": [
            {
                "name": "dn1_1",
                "host": "10.0.7.16",
                "port": "20008",
                "work_dir": "/home/gbase/data/dn1/dn1_1",
```

```
                "agentPort": "8005",
                "state": "running",
                "role": "primary"
                "agentHost": "10.0.7.16"
            }
        ]
    },
    "dcs": {
        "clusterState": "healthy",
        "members": [
            {
                "url": "http://10.0.7.16:2379",
                "id": "84e2852165d202b0",
                "name": "node_0",
                "isLeader": true,
                "state": "healthy"
            }
        ]
    }
}
```

3 实例主备切换

操作场景

GBase 8s 在运行过程中，数据库管理员可能需要手工对数据库节点做主备切换。例如发现数据库节点主备宕机后，需要恢复原有的主备角色，或怀疑硬件故障需要手动进行主备切换。级联备机不能直接转换为主机，只能先通过 switchover 成为备机，然后再成为主机。

说明

- 主备切换为维护操作。需确保 GBase 8s 状态正常，且在所有业务结束后，进行切换操作。
- 在开启极致 RTO 时，不支持级联备机。由于在此情况下，备机不支持连接而无法同步数据。
- 对于同一数据库，如果前一次主备切换操作尚未完成，则不能再次执行切换。
- 假如在业务期间发起 switchover，可能由于主机线程无法停止，而导致显示切换超时，但实际后台仍然在运行。只需等主机线程停止后，再执行 switchover 即可完成。例如当主机删除一个大的分区表时，可能无法即时响应 switchover 发起的信号。

语法格式

```
gha_ctl switchover datanode group_name leader_node_name  [standby_node_name] -l dcslist [-c cluster]
```

示例

将数据库 DN 备节点切换为主节点。需执行以下步骤：

步骤 1 查询数据库状态，执行命令：

```
[gbase@gbase8s ~]$ gs_om -t status --detail
```

返回如下信息，此时 10.0.7.7 为主数据节点，10.0.7.8 为备数据节点。

```
.....
[  Datanode State(group id: 2)   ]
```

```
    node       node_ip         port      instance
state
------------------------------------------------------------------------------
--------------------
1  gbase8s_7_7 10.0.7.7        15432      6001 /home/gbase/data/dn1/dn1_1   P
Primary Normal
2  gbase8s_7_8 10.0.7.8        15432      6002 /home/gbase/data/dn1/dn1_2   S
Standby Normal
```

步骤 2 登录备节点，执行主备切换命令。另外，在 switchover 级联备机后，级联备机成为备机，而备机降为级联备。

```
[gbase@gbase8s_7_8 ~]$ gha_ctl switchover datanode dn1 dn1_1 -c gbase8s -l
http://10.0.7.7:2379
```

返回如下信息后，主备切换成功。

```
{
    "ret":0,
    "msg":"Success"
}
```

再次执行步骤 1，查询主备状态：10.0.7.7 为备节点，10.0.7.8 为主节点。

错误排查

如果 switchover 过程中出错，请根据日志文件中的信息排查错误，详见 9 日志参考。

异常处理

异常判断标准如下：

- 业务压力下，主备实例切换时间长，这种情况不需要处理。
- 其他备机正在 build 的情况下，主机需要发送日志到备机后，才能降备，导致主备切换时间长。这种情况不需要处理，但应尽量避免 build 过程中进行主备切换。
- 切换过程中，因网络故障、磁盘满等原因造成主备实例连接断开，出现双主现象时，此时请参考如下步骤修复。

出现双主状态后，请按如下步骤操作，恢复为正常的主备状态。否则可能会造成数据丢失。

(1) 执行以下命令，查询数据库当前的实例状态。

```
gs_om -t status --detail
```

若查询结果显示两个实例的状态都为 Primary，这种状态为异常状态。

(2) 在降为备机的节点上，执行如下命令关闭服务。

```
gha_ctl stop datanode group_name node_name -l dcslist [-c cluster]
```

(3) 执行以下命令，启动备节点。

```
gha_ctl start datanode group_name node_name -c cluster_name -l dcslist [-c cluster]
```

(4) 查看数据库状态，确认实例状态恢复。

4 例行维护

GBase 8s 数据库允许管理员用户进行例行维护的操作，以保证数据库中的数据安全，避免发生丢失数据、非法访问数据等事故。

4.1 日维护检查项

4.1.1 检查数据库状态

通过 GBase 8s 提供的工具查询数据库和实例状态，确认数据库和实例都处于正常的运行状态，可以对外提供数据服务。

- 检查数据库状态

```
gs_check -U admin_user -i CheckClusterState
```

参数说明

- -U：指定以哪个管理员用户身份进行检查操作。
- -i：指定健康检查项的名称

执行返回检查状态的信息。

例如：

```
[gbase@gbase8s ~]$ gs_check -U gbase -i CheckClusterState
Parsing the check items config file successfully
Distribute the context file to remote hosts successfully
Start to health check for the cluster. Total Items:1 Nodes:1

Checking...                [=========================] 1/1
Start to analysis the check result
CheckClusterState.....................OK
The item run on 1 nodes.  success: 1

Analysis the check result successfully
Success.         All check items run completed. Total:1   Success:1
For more information please refer to
/home/gbase/gbase_db/om_f5c5a0af/script/gspylib/inspection/output/CheckRepor
t_202212063481010006.tar.gz
```

- 检查参数

以管理员身份登录数据库，使用 show 命令查看指定参数的值。

```
SHOW parameter_name;
```

例如，查看 listen_addresses 参数值：

```
[gbase@gbase8s ~]$ gsql -d postgres -p 5432
gsql ((single_node GBase8sV8.8 3.0.0B51 build 0b9407bc) compiled at 2022-11-18
16:09:02 commit 0 last mr 874 )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# show listen_addresses;
    listen_addresses
-----------------------
 localhost,10.0.7.16
(1 row)
```

- 修改参数

通过 gs_guc 命令修改参数值。参数修改生效方式不同，请参看《GBase 8s V8.8_开发者指南》“重设参数”章节。

```
gs_guc reload -Z node_type -D datadir -c "paraname_name=value"
```

例如，修改 listen_addresses 参数：

```
[gbase@gbase ~]$ gs_guc reload -Z datanode -D /home/gbase/data/dn1/dn1_1/ -c
"listen_addresses='localhost,10.0.7.16,10.0.7.17'"
The gs_guc run with the following arguments: [gs_guc -Z datanode -D
/home/gbase/data/dn1/dn1_1/ -c
listen_addresses='localhost,10.0.7.16,10.0.7.17' reload ].
expected instance path: [/home/gbase/data/dn1/dn1_1/postgresql.conf]
gs_guc reload: listen_addresses='localhost,10.0.7.16,10.0.7.17':
[/home/gbase/data/dn1/dn1_1/postgresql.conf]
server signaled

Total instances: 1.
Success to perform gs_guc!
```

4.1.2 检查锁信息

锁机制是 GBase 8s 数据库保证数据一致性的重要手段，检查相关信息可以检查数据库的事务和运行状况。

- 查询数据库中的锁信息

  以管理员身份登录数据库，执行如下命令：

```
SELECT * FROM pg_locks;
```

- 查询等待锁的线程状态信息

  以管理员身份登录数据库，执行如下命令：

```
SELECT * FROM pg_thread_wait_status WHERE wait_status = 'acquire lock';
```

- 结束系统进程

  查找正在运行的系统进程，并使用 kill 命令结束特定进程。根据实际情况指定 pid 号。

```
ps ux
kill -9 pid
```

4.1.3 统计事件数据

SQL 语句长时间运行会占用大量系统资源。用户可以通过查看事件发生的时间以及占用内存大小，了解当前数据库运行状态。

- 查询事件时间信息

  查询事件时间，如线程启动时间、事务启动时间、SQL 启动时间、状态变更时间等。

  以管理员身份登录数据库，执行如下命令：

```
SELECT backend_start,xact_start,query_start,state_change FROM
pg_stat_activity;
```

- 查询当前服务器的会话计数信息

  以管理员身份登录数据库，执行如下命令：

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

- 查询系统级统计信息

  查询当前使用内存最多的 TOP10 会话信息。可根据实际需要修改限制条数。

  以管理员身份登录数据库，执行如下命令：

```
SELECT * FROM pv_session_memory_detail() ORDER BY usedsize desc limit 10;
```

4.1.4 对象检查

表、索引、分区、约束等对象是数据库的核心存储对象。这些对象的核心信息监控和维护是 DBA 重要的日常工作。

- 查看表的详细信息

  以管理员身份登录数据库，执行如下命令：

```
\d+ table_name;
```

- 查询表统计信息

  以管理员身份登录数据库，执行如下命令：

```
SELECT * FROM pg_statistic;
```

- 查看索引的详细信息

  以管理员身份登录数据库，执行如下命令：

```
\d+ index_name;
```

- 查询分区表信息

  以管理员身份登录数据库，执行如下命令：

```
SELECT * FROM pg_partition;
```

- 收集统计信息

  - 使用 ANALYZE 语句，收集数据库相关的统计信息。

  登录数据库，并执行命令：

```
ANALYZE;
```

  - 使用 VACUUM 语句，回收空间并更新统计信息。

  登录数据库，并执行命令：

```
VACUUM;
```

- 查询约束信息

  以管理员身份登录数据库，执行以下命令：

```
SELECT * FROM pg_constraint;
```

4.1.5 SQL 报告检查

使用 EXPLAIN 语句，查看执行计划。例如：

```
gbase=# EXPLAIN SELECT * FROM pg_constraint;
                            QUERY PLAN
------------------------------------------------------------------
 Seq Scan on pg_constraint  (cost=0.00..1.03 rows=3 width=1157)
(1 row)
```

4.1.6 备份

数据备份重于一切，日常应检查备份执行情况、备份有效性，以确保备份能够保障数据安全。备份安全加密也应兼顾。

- 将数据库导出备份。语法格式：

```
gs_dump dbname -f filename [-F c|d|t|p]-U user_name -W password
```

参数说明：

- dbname：指定待导出备份的数据库的名称。
- -f：指定导出文件的名称。
- -F：指定导出文件的格式。支持 custom、directory、tar、plain text。缺省默认为纯文本格式。
- -U：指定以哪个管理员身份连接。
- -W：指定管理员的登录密码。

- 将模式导出备份。语法格式：

```
gs_dump dbname -p port -n schema_name -f filename
```

参数说明：

- -n：指定待导出备份的模式的名称。
- 其他参数同上。

- 将表导出备份。语法格式：

```
gs_dump dbname -p port -t table_name -f filename
```

参数说明：

- -t：指定待导出备份的表的名称。
- 其他参数同上。

4.1.7 基本信息检查

数据库基本信息包括版本、组件、容量等信息，定期检查数据库信息并登记在案是数据库生命周期管理的重要内容之一。

- 版本信息

```
SELECT version();
```

- 容量检查

```
SELECT pg_table_size('table_name');
SELECT pg_database_size('database_name');
```

4.2 检查操作系统参数

4.2.1 检查办法

通过 GBase 8s 提供的 gs_checkos 工具，可以完成操作系统状态的检查。

前提条件

- 当前的硬件和网络环境正常。
- 各主机间 root 互信状态正常。
- 只能使用 root 用户执行 gs_checkos 命令。

操作步骤

步骤 1 以 root 用户身份登录任意一台服务器。

步骤 2 执行如下命令，对 GBase 8s 节点服务器的 OS 参数进行检查。

```
[root@gbase8s ~]$ gs_checkos -i A
```

检查节点服务器 OS 参数的目的：保证 GBase 8s 正常通过预安装环节，并且在安装成功后可以持续安全高效地运行。详细检查项目，参见《GBase 8s V8.8_工具参考手册》中“gs_checkos”章节。

示例

- 以参数"A"为例。

```
[root@gbase8s ~]$ gs_checkos -i A
```

执行返回如下信息：

```
Checking items:
    A1. [ OS version status ]                                   : Normal
    A2. [ Kernel version status ]                               : Normal
    A3. [ Unicode status ]                                      : Normal
    A4. [ Time zone status ]                                    : Normal
    A5. [ Swap memory status ]                                  : Normal
```

```
    A6. [ System control parameters status ]                    : Normal
    A7. [ File system configuration status ]                    : Normal
    A8. [ Disk configuration status ]                           : Normal
    A9. [ Pre-read block size status ]                          : Normal
    A10.[ IO scheduler status ]                                 : Normal
    A11.[ Network card configuration status ]                   : Normal
    A12.[ Time consistency status ]                             : Normal
    A13.[ Firewall service status ]                             : Normal
    A14.[ THP service status ]                                  : Normal
Total numbers:14. Abnormal numbers:0. Warning numbers:0.
Do checking operation finished. Result: Normal.
```

- 以参数"B"为例。

```
[root@gbase8s ~]$ gs_checkos -i B
```

执行返回如下信息：

```
Setting items:
    B1. [ Set system control parameters ]                       : Normal
    B2. [ Set file system configuration value ]                 : Normal
    B3. [ Set pre-read block size value ]                       : Normal
    B4. [ Set IO scheduler value ]                              : Normal
    B5. [ Set network card configuration value ]                : Normal
    B6. [ Set THP service ]                                     : Normal
    B7. [ Set RemoveIPC value ]                                 : Normal
    B8. [ Set Session Process ]                                 : Normal
Total numbers:8. Abnormal numbers:0. Warning numbers:0.
```

4.2.2 异常处理

使用 gs_checkos 检查 GBase 8s 状态时，可以使用--detail 参数，来查看详细的错误信息。

返回检查状态：

- Abnormal 为必须处理项，影响 GBase 8s 的安装。
- Warning 可以不处理，不会影响 GBase 8s 的安装。
- Normal 表示正常，支持安装。

返回检查项：

- 如果操作系统版本（A1）检查项检查结果为 Abnormal，需要将不属于混编范围的操作

系统版本替换为混编范围内的操作系统版本。

- 如果内核版本（A2）检查项检查结果为 Warning，则表示 GBase 8s 内平台的内核版本不一致。

- 如果 Unicode 状态（A3）检查项检查结果为 Abnormal，需要将各主机的字符集设置为相同的字符集。编辑/etc/profile 文件实现。

```
vim /etc/profile
```

在打开的配置文件中，添加如下内容：

```
export LANG=unicode
```

其中 unicode 为 Unicode 编码方式，根据实际情况指定。输入“:wq”保存并退出。

- 如果时区状态（A4）检查项检查结果为 Abnormal，需要将各主机的时区设置为相同时区。可以将/usr/share/zoneinfo/目录下的时区文件拷贝为/etc/localtime 文件。

```
cp /usr/share/zoneinfo/zonefile /etc/localtime
```

其中 zonefile 为需要使用的时区文件名，根据实际情况指定。

- 如果交换内存状态（A5）检查项检查结果为 Abnormal，可能是因为 swap 空间大于 mem 空间。可减小 Swap 解决或者增大 Mem 空间解决。

- 如果系统控制参数（A6）检查项检查结果为 Abnormal，可使用以下两种方法进行设置。

  - 第一种：使用如下命令进行设置。

  ```
  gs_checkos -i B1
  ```

  - 第二种：根据错误提示信息，在/etc/sysctl.conf 文件中进行设置。

  ```
  vim /etc/sysctl.conf
  ```

  修改配置文件，输入“:wq”保存并退出。

  并执行如下命令，使配置修改后生效。

  ```
  sysctl -p
  ```

- 如果文件系统配置状态（A7）检查项检查结果为 Abnormal，可以使用如下命令进行设置。

```
gs_checkos -i B2
```

- 如果磁盘配置状态（A8）检查项检查结果为 Abnormal，需修改磁盘挂载格式为：“rw,noatime,inode64,allocsize=16m”。

  使用 linux 的 man mount 命令挂载 XFS 选项：

```
rw, noatime, inode64, allocsize=16m
```

  也可以在/etc/fstab 文件中设定 XFS 选项。如下示例：

```
/dev/data /data xfs rw, noatime, inode64, allocsize=16m 0 0
```

- 如果预读块大小（A9）检查项检查结果为 Abnormal，可使用如下命令进行设置。

```
gs_checkos -i B3
```

- 如果 IO 调度状态（A10）检查项检查结果为 Abnormal，可使用如下命令进行设置。

```
gs_checkos -i B4
```

- 如果网卡配置状态（A11）检查项检查结果为 Warning，可使用如下命令进行设置。

```
gs_checkos -i B5
```

- 如果时间一致性（A12）检查项检查结果为 Abnormal，需检查是否安装 ntp 服务，以及 ntp 服务是否启动；并与 ntp 时钟源同步。

- 如果防火墙状态（A13）检查项检查结果为 Abnormal，需关闭防火墙服务。使用如下命令进行设置。以 CentOS7、openEuler 操作系统为例：

```
systemctl stop firewalld
systemctl disable firewalld
```

- 如果 THP 服务（A14）检查项检查结果为 Abnormal，可使用如下命令进行设置。

```
gs_checkos -i B6
```

4.3 检查 GBase 8s 健康状态

4.3.1 检查办法

通过 GBase 8s 提供的 gs_check 工具，可以检查 GBase 8s 健康状态。

注意事项

- 扩容新节点检查只能在 root 用户下执行，其他场景都必须在 gbase 用户下执行。
- 必须指定-i 或-e 参数，-i 会检查指定的单项，-e 会检查对应场景配置中的多项。
- 如果-i 参数中不包含 root 类检查项或-e 场景配置列表中没有 root 类检查项，则不需要交互输入 root 权限的用户及其密码。
- 可使用--skip-root-items 跳过检查项中包含的 root 类检查，以免需要输入 root 权限用户及密码。
- 检查扩容新节点与现有节点之间的一致性，在现有节点执行 gs_check 命令指定-- hosts 参数进行检查，其中 hosts 文件中需要写入新节点 IP。

操作步骤

方式 1

步骤 1 以管理员用户 gbase 登录数据库主节点。

步骤 2 检查 GBase 8s 数据库状态，执行如下命令：

```
gs_check -i CheckClusterState
```

其中参数说明：

- -i：该参数用于指定检查项。注意区分大小写。格式为

  ```
  -i CheckClusterState/CheckCPU/CheckClusterState/CheckCPU
  ```

  取值范围为所有支持的检查项名称，详见《GBase 8s V8.8_工具参考手册》中“gs_checkos”章节。用户可根据实际需求，指定检查项。

方式 2

步骤 1 以管理员用户 gbase 登录数据库主节点。

步骤 2 对 GBase 8s 数据库进行健康检查，执行如下命令：

```
gs_check -e inspect
```

其中参数说明：

- -e：该参数用于指定场景名。注意区分大小写。格式为

```
-e inspect/upgrade
```

取值范围为所有支持的巡检场景名称。默认场景包括：inspect（例行巡检）、upgrade（升级前巡检）、binary_upgrade（就地升级前巡检）、health（健康检查巡检）、slow_node（节点）、longtime（耗时长巡检）、install（安装）。用户可根据实际需求，指定巡检场景。

GBase 8s 巡检的主要作用：在数据库运行过程中，检查整个 GBase 8s 数据库集群状态是否正常，或者重大操作前（升级、扩容），确保 GBase 8s 满足操作所需的环境条件和状态条件。详细的巡检项目和场景，请参见《GBase 8s V8.8_工具参考手册》中“服务端工具 > gs_checkos > GBase 8s 状态检查表”。

示例

执行单项检查结果：

```
[gbase@gbase8s ~]$ gs_check -i CheckCPU
Parsing the check items config file successfully
Distribute the context file to remote hosts successfully
Start to health check for the cluster. Total Items:1 Nodes:1

Checking...                [=========================] 1/1
Start to analysis the check result
CheckCPU.....................................OK
The item run on 1 nodes.  success: 1

Analysis the check result successfully
Success.         All check items run completed. Total:1   Success:1
For more information please refer to
/home/gbase/gbase_db/om_f5c5a0af/script/gspylib/inspection/output/CheckRepor
t_202212065188911009.tar.gz
```

本地执行结果：

```
[gbase@gbase8s ~]$ gs_check -i CheckCPU -L
2022-08-18 15:17:42 [NAM] CheckCPU
2022-08-18 15:17:42 [STD] 检查主机 CPU 占用率，如果 idle 大于 30%,或者 iowait 小
于 30%.则检查项通过，否则检查项不通过
2022-08-18 15:17:42 [RST] OK
```

```
2022-08-18 15:17:42 [RAW]
Linux 3.10.0-1127.el7.x86_64 (gbase8s_hostname)      08/18/22        _x86_64_
(2 CPU)

15:17:37
CPU      %user      %nice    %system    %iowait     %steal      %idle
15:17:38        all      14.57       0.00       3.52       0.00       0.00
81.91
15:17:39        all       5.53       0.00       1.51       0.00       0.00
92.96
15:17:40        all      19.00       0.00       4.50       0.00       0.00
76.50
15:17:41        all      49.24       0.00      11.17       0.00       0.00
39.59
15:17:42        all       6.47       0.00       2.99       0.50       0.00
90.05
Average:        all      18.88       0.00       4.72       0.10       0.00
76.31
```

执行场景检查结果：

```
[gbase@gbase8s ~]$ gs_check -e inspect
Parsing the check items config file successfully
The below items require root privileges to execute:[CheckBlockdev
CheckIOrequestqueue CheckIOConfigure CheckMTU CheckRXTX CheckMultiQueue
CheckFirewall CheckSshdService CheckSshdConfig CheckCrondService
CheckMaxProcMemory CheckBootItems CheckFilehandle CheckNICModel
CheckDropCache]
Please enter root privileges user[root]:root
Please enter password for user[root]:
Check root password connection successfully
Distribute the context file to remote hosts successfully
Start to health check for the cluster. Total Items:57 Nodes:1

Checking...                [=========================] 57/57
Start to analysis the check result
CheckClusterState.......................OK
The item run on 1 nodes.  success: 1

CheckDBParams...........................OK
The item run on 1 nodes.  success: 1
```

```
CheckDebugSwitch.........................OK
The item run on 1 nodes.  success: 1

CheckDirPermissions......................NG
The item run on 1 nodes.  ng: 1
The ng[gbase8s_5_124] value:
AppPath directory(/home/gbase/gbase_db/app) permissions 700           : Normal
Tmp directory(/home/gbase/gbase_db/tmp) permissions 700               : Normal
Log directory(/home/gbase/gbase_db/log) permissions 700               : Normal
ToolPath directory(/home/gbase/gbase_db/om) permissions 775           :
Abnormal reason: Directory permission can not exceed 750.
DN directory(/home/gbase/data/dn1/dn1_1) permissions 700              : Normal
DN Xlog directory(/home/gbase/data/dn1/dn1_1/pg_xlog) permissions 700 : Normal


CheckReadonlyMode........................OK
The item run on 1 nodes.  success: 1

CheckEnvProfile..........................OK
The item run on 1 nodes.  success: 1  (consistent)
The success on all nodes value:
GAUSSHOME        /home/gbase/gbase_db/app
LD_LIBRARY_PATH  /home/gbase/gbase_db/app/lib
PATH             /home/gbase/gbase_db/app/bin


CheckBlockdev............................OK
The item run on 1 nodes.  success: 1

CheckCurConnCount........................OK
The item run on 1 nodes.  success: 1

CheckCursorNum...........................OK
The item run on 1 nodes.  success: 1

CheckPgxcgroup...........................OK
The item run on 1 nodes.  success: 1

CheckDiskFormat..........................OK
The item run on 1 nodes.  success: 1
```

```
CheckSpaceUsage.............................OK
The item run on 1 nodes.  success: 1

CheckInodeUsage.............................OK
The item run on 1 nodes.  success: 1

CheckSwapMemory.............................NG
The item run on 1 nodes.  ng: 1
The ng[gbase8s_5_124] value:
SwapMemory(8455712768) must be 0.
MemTotal: 8201502720.

CheckLogicalBlock...........................OK
The item run on 1 nodes.  success: 1

CheckIOrequestqueue.........................OK
The item run on 1 nodes.  success: 1

CheckMaxAsyIOrequests.....................WARNING
The item run on 1 nodes.  warning: 1
The warning[gbase8s_5_124] value:
Asy IO requests 65536  expectedScheduler 104857600.


CheckIOConfigure............................OK
The item run on 1 nodes.  success: 1

CheckMTU....................................OK
The item run on 1 nodes.  success: 1  (consistent)
The success on all nodes value:
1500

CheckPing...................................OK
The item run on 1 nodes.  success: 1

CheckRXTX...................................OK
The item run on 1 nodes.  success: 1

CheckNetWorkDrop............................OK
The item run on 1 nodes.  success: 1
```

```
CheckMultiQueue.........................ERROR
The item run on 1 nodes.  error: 1
The error[gbase8s_5_124] value:
string indices must be integers

CheckEncoding...........................OK
The item run on 1 nodes.  success: 1  (consistent)
The success on all nodes value:
LANG=en_US.UTF-8

CheckFirewall...........................OK
The item run on 1 nodes.  success: 1

CheckKernelVer..........................OK
The item run on 1 nodes.  success: 1  (consistent)
The success on all nodes value:
3.10.0-1127.el7.x86_64

CheckMaxHandle..........................NG
The item run on 1 nodes.  ng: 1
The ng[gbase8s_5_124] value:
Max open files: 1000000
The value of max open files is 1024 on pid 5436. it must not be less than 1000000.
The value of max open files is 1024 on pid 5508. it must not be less than 1000000.
The value of max open files is 1024 on pid 5579. it must not be less than 1000000.


CheckNTPD...............................NG
gbase8s_5_124: NTPD service is not running, 2022-12-06 14:28:06

CheckOSVer..............................OK
gbase8s_5_124: The current OS is centos 7.8 64bit.


CheckSysParams..........................NG
The item run on 1 nodes.  ng: 1
The ng[gbase8s_5_124] value:
Abnormal reason: variable 'net.ipv4.tcp_retries2' RealValue '12' ExpectedValue
'80'.
Warning reason: variable 'net.ipv4.tcp_retries1' RealValue '3' ExpectedValue
'5'.
```

```
Warning reason: variable 'net.ipv4.tcp_syn_retries' RealValue '6' ExpectedValue
'5'.


CheckTHP.....................................OK
The item run on 1 nodes.  success: 1

CheckTimeZone................................OK
The item run on 1 nodes.  success: 1  (consistent)
The success on all nodes value:
+0800

CheckCPU.....................................OK
The item run on 1 nodes.  success: 1

CheckSshdService.............................OK
The item run on 1 nodes.  success: 1

CheckSshdConfig.........................WARNING
The item run on 1 nodes.  warning: 1
The warning[gbase8s_5_124] value:

Warning reason: UseDNS parameter is not set; expected: no

CheckCrondService............................OK
The item run on 1 nodes.  success: 1

CheckStack...................................OK
The item run on 1 nodes.  success: 1  (consistent)
The success on all nodes value:
8192

CheckSysPortRange............................OK
The item run on 1 nodes.  success: 1

CheckMemInfo.................................OK
The item run on 1 nodes.  success: 1  (consistent)
The success on all nodes value:
totalMem: 7.638244628906256

CheckHyperThread.............................NG
```

```
The item run on 1 nodes.  ng: 1
The ng[gbase8s_5_124] value:
Hyper-threading is down.

CheckTableSpace.............................OK
The item run on 1 nodes.  success: 1

CheckSysadminUser...........................NG
The item run on 1 nodes.  ng: 1
The ng[gbase8s_5_124] value:
There are sysadmin users except gbase:
gha

CheckGUCConsistent..........................OK
All DN instance guc value is consistent.

CheckMaxProcMemory..........................NG
The item run on 1 nodes.  ng: 1
The ng[gbase8s_5_124] value:
parameter max_process_memory setting should not be bigger than
recommended(kb):3203712:
RecommendedMaxMem is 3203712


CheckBootItems..............................OK
The item run on 1 nodes.  success: 1

CheckHashIndex..............................OK
The item run on 1 nodes.  success: 1

CheckPgxcRedistb............................OK
The item run on 1 nodes.  success: 1

CheckNodeGroupName..........................OK
The item run on 1 nodes.  success: 1

CheckTDDate.................................OK
The item run on 1 nodes.  success: 1

CheckDilateSysTab...........................OK
The item run on 1 nodes.  success: 1
```

```
CheckKeyProAdj.............................OK
The item run on 1 nodes.  success: 1

CheckProStartTime..........................OK
Basically ,all the gaussdb process start at the same time

CheckFilehandle............................OK
The item run on 1 nodes.  success: 1

CheckRouting...............................OK
The item run on 1 nodes.  success: 1

CheckNICModel..............................OK
The item run on 1 nodes.  success: 1  (consistent)
The success on all nodes value:
version: 1.4.16.0-k-NAPI
model: VMware VMXNET3 Ethernet Controller


CheckDropCache.........................WARNING
The item run on 1 nodes.  warning: 1
The warning[gbase8s_5_124] value:
No DropCache process is running

CheckMpprcFile.............................NG
The item run on 1 nodes.  ng: 1
The ng[gbase8s_5_124] value:
There is no mpprc file

Analysis the check result successfully
Failed. All check items run completed. Total:57   Success:43   Warning:4   NG:9
Error:1
For more information please refer to
/home/gbase/gbase_db/om_f5c5a0af/script/gspylib/inspection/output/CheckRepor
t_inspect_202208185514623739.tar.gz
```

4.3.2 异常处理

如检查发现 GBase 8s 状态异常，可参考以下处理方法进行修复。

表 4-1 检查 GBase 8s 运行状态

检查项	检查内容	异常状态	处理方法
CheckClusterState	检查 GBase 8s 集群状态	集群未启动或实例未启	启动 GBase 8s 及实例。 `gs_om -t start`
		集群状态异常或实例异常	检查各主机、实例状态，根据状态信息排查。 `gs_check -i CheckClusterState`
CheckDBParams	检查数据库参数	数据库参数错误	通过 gs_guc 工具修改数据库参数为指定值。
CheckDebugS witch	检查调试日志	日志级别不正确	使用 gs_guc 工具将 log_min_messages 改为指定内容。
CheckDirPermissions	检查目录权限	路径权限错误	修改对应目录权限为指定数值(750/700) `chmod 750 DIR`

CheckReadonlyMode	检查只读模式	只读模式被打开	确认数据库节点所在磁盘使用率未超阈值(默认 60%)且未在执行其他运维操作。
			`gs_check -i CheckDataDiskUsage`
			`ps ux`
			使用 gs_guc 工具关闭 GBase 8s 只读模式。
			`gs_guc reload -N all -I all -c 'default_transaction_read_only = off'`
			`gs_guc reload -N all -I all -c 'default_transaction_read_only = off'`

CheckEnvProfile	检查环境变量	环境变量不一	重新执行前置更新环境变量信息。

检查项	检查内容	异常状态	处理方法
		致	
CheckBlockdev	检查磁盘预读块	磁盘预读块大小不为 16384	使用 gs_checkos 设置预读块大小为 16384KB，并写入自启动文件。 `gs_checkos -i B3`
CheckCursorNum	检查游标数	检查游标数失败	检查数据库能否正常连接，GBase 8s 状态是否正常。
CheckPgxcgroup	检查重分布状态	有未完成重分布的 pgxc_group 表	继续完成扩容或缩容的数据重分布操作。 `gs_expand、gs_shrink`
CheckDiskFormat	检查磁盘配置	各节点磁盘配置不一致	将各节点的磁盘规格改为相同。
CheckSpaceUsage	检查磁盘空间使用率	磁盘可用空间不足	清理或扩展对应目录所在的磁盘。
CheckInodeUsage	检查磁盘索引使用率	磁盘可用索引不足	清理或扩展对应目录所在的磁盘。
CheckSwapMeory	检查交换内存	交换内存大于物理内存	将交换内存调小或关闭。
CheckLogicalBlock	检查磁盘逻辑块	磁盘逻辑块大小不为 512	使用 gs_checkos 修改磁盘逻辑块大小为 512KB，并写入开机自启动文件。 `gs_checkos -i B4`
CheckIOrequestqueue	检查 IO 请求	IO 请求值不为 32768	使用 gs_checkos 设置 IO 请求值为 32768，并写入开机自启动文件。 `gs_checkos -i B4`
CheckCurConn	检查当前连接	当前连接数超过最大连接数	断开未使用的数据库主节点连接。

检查项	检查内容	异常状态	处理方法
Count	数	的 90%	
CheckMaxAsyIOrequests	检查最大异步请求	最大异步请求值小于 104857600 或当前节点数据库实例数）*1048576	使用 gs_checkos 设置最大异步请求值为 104857600 和当前节点数据库实例数*1048576 中的最大值。 `gs_checkos -i B4`
CheckMTU	检查 MTU 值	MTU 值不一致	设置各节点的 MTU 一致为 1500 或 8192。 `ifconfig eth* MTU 1500`
CheckIOConfigure	检查 IO 配置	IO 配置不是 deadline	使用 gs_checkos 设置 IO 配置为 deadline，并写入开机自启动文件。 `gs_checkos -i B4`
CheckRXTX	检查 RXTX 值	网卡 RX/TX 值不是 4096	使用 checkos 设置 GBase 8s 使用的物理网卡 RX/TX 值为 4096 `gs_checkos -i B5`
CheckPing	检查网络通畅	存在 GBase 8s IP 无法 ping 通	检查异常 ip 间网络设置和状态、防火墙状态。
CheckNetWorkDrop	检查网络丢包率	网络通信丢包率高于 1%	检查对应 IP 间网络负载、状态。
CheckMultiQueue	检查网卡多队列	未开启网卡多队列并未将网卡中断绑定到不同 CPU 核心	开启网卡多队列并将网卡队列中断绑定到不同的 CPU 核心。
CheckEncoding	检查编码格式	各节点编码格式不一致	在/etc/profile 中写入一致的编码信息。

检查项	检查内容	异常状态	处理方法
			echo "export LANG=XXX" >> /etc/profile
CheckFirewall	检查防火墙	防火墙未关闭	关闭防火墙服务。 redHat(CentOS)7.x： systemctl stop firewalld redHat(CentOS)6.x： service iptables down SuSE：SuSE firewall2 down
CheckKernelVer	检查内核版本	内核版本不一致	使用 gs_replace 替换不一致的节点。 gs_replace
CheckMaxHandle	检查最大文件句柄数	最大文件句柄数小于1000000	设置 91-nofile.conf/90-nofile.conf 最大文件句柄数软硬限制为1000000。 gs_checkos -i B2
CheckNTPD	检查时间同步服务	NTPD 服务未开启或时间误差超过一分钟	开启 NTPD 服务并设置时钟一致。
CheckOSVer	检查操作系统版本	操作系统不被支持或不在同一混搭列表	使用 gs_replace 将异常节点替换为受支持系统 或在同一混搭列表内的系统的节点。 gs_replace
CheckSysParams	检查操作系统参数	操作系统参数设置不满足要求	使用 gs_checkos 进行参数设置 gs_checkos -i B1 或手动设置 vim /etc/sysctl.conf

检查项	检查内容	异常状态	处理方法
CheckTHP	检查 THP 服务	THP 服务未开启	使用 gs_checkos 设置 THP 服务。 `gs_checkos -i B6`
CheckTimeZone	检查时区	时区不一致	设置各节点为同一时区。cp /usr/share/zoneinfo/$主时区/$次时区 /etc/localtime
CheckCPU	检查 CPU	CPU 占用过高或 IO 等待过高	进行 CPU 配置升级或磁盘性能升级。
CheckSshdService	检查 SSHD 服务	未开启 SSHD 服务	启动SSHD服务并写入开机自启动文件。 `server sshd startecho "server sshd start" >> initFile`

CheckSshdConfig	检查 SSHD 配置	SSHD 服务配置错误	设置 SSHD 服务。
			`PasswordAuthentication=no; MaxStartups=1000; UseDNS=yes; ClientAliveInterval=10800/ClientAliveInterval=0`
			并重启服务
			`server sshd start`

CheckCrondService	检查 Crond 服务	Crond 服务未启动	安装 Crond 服务并启用。
CheckStack	检查堆栈大小	堆栈大小小于 3072	使用 gs_checkos 设置为 3072 并重启堆栈值过小进程。 `gs_checkos -i B2`
CheckSysPortRange	检查系统端口设置	系统 ip 端口不在预期范围内	设置系统 ip 端口范围参数到 26000-65535 之中；设置 GBase 8s

检查项	检查内容	异常状态	处理方法
		或 GBase 8s 端口在系统 ip 端口内	端口在系统 IP 端口范围外。 `vim /etc/sysctl.conf`
CheckMemInfo	检查内存信息	各节点内存大小不一致	使用相同规格的物理内存。
CheckHyperThread	检查超线程	未开启 CPU 超线程	开启 CPU 超线程。
CheckTableSpace	检查表空间	表空间路径和 GBase 8s 路径存在嵌套或表空间路径相互存在嵌套	将表空间数据迁移到路径合法的表空间中。

4.4 检查数据库性能

检查办法

通过 GBase 8s 提供的性能统计工具 gs_checkperf，可以检查硬件性能。

前提条件

- GBase 8s 运行状态正常。
- 运行在数据库之上的业务运行正常。

操作步骤

步骤 1 以管理员用户 gbase，登录数据库主节点。

步骤 2 执行如下命令，对 GBase 8s 数据库进行性能检查。

```
gs_checkperf
```

具体的性能统计项目请参见《GBase 8s V8.8_工具参考手册》中“服务端工具 > gs_checkperf > 性能检查项”。

示例

以简要格式在屏幕上显示性能统计结果，执行如下命令：

```
gs_checkperf -i pmk -U gbase
```

返回如下信息：

```
Cluster statistics information:
    Host CPU busy time ratio                    :    1.43        %
    MPPDB CPU time % in busy time               :    1.88        %
    Shared Buffer Hit ratio                     :    99.96       %
    In-memory sort ratio                        :    100.00      %
    Physical Reads                              :    4
    Physical Writes                             :    25
    DB size                                     :    70          MB
    Total Physical writes                       :    25
    Active SQL count                            :    2
    Session count                               :    3
```

异常处理

使用 gs_checkperf 工具，检查 GBase 8s 性能状态。如果检查发现结果异常，可以根据以下内容进行修复。

表 4-2 检查 GBase 8s 级别性能状态

异常状态	处理方法
主机 CPU 占有率高	(1) 更换或增加高性能的 CPU (2) 使用 top 命令，查看系统哪些进程的 CPU 占有率高，并使用 kill 命令关闭无用进程。
GBase 8s CPU 占有率高	(1) 更换和增加高性能的 CPU。 (2) 使用 top 命令查看数据库哪些进程的 CPU 占有率高，然后使用 kill 命令关闭没有使用的进程。 (3) 使用 gs_expand 工具扩容，增加新的主机，以均衡 CPU 占有率。

异常状态	处理方法
共享内存命中率低	(1) 扩大内存。 (2) 使用如下命令查看操作系统配置文件/etc/sysctl.conf，调大共享内存 kernel.shmmax 值。 `vim /etc/sysctl.conf`
内存中排序比率低	扩大内存。
I/O、磁盘使用率高	(1) 更换高性能的磁盘。 (2) 调整数据布局，尽量将 I/O 请求较合理的分配到所有物理磁盘中。 (3) 全库进行 VACUUM FULL 操作。 `vacuum full;` (4) 进行磁盘整理。 (5) 降低并发数。
事务统计	查询 pg_stat_activity 系统表，将不必要的连接断开。登录数据库后查询： `\d+ pg_stat_activity;`

表 4-3 检查节点级别性能状态

异常状态	处理方法
CPU 占有率高	(1) 更换和增加高性能的 CPU。 (2) 使用 top 命令查看系统哪些进程的 CPU 占有率高，然后使用 kill 命令关闭没有使用的进程。
内存使用率过高情况	扩大或清理内存。
I/O 使用率过高情况	(1) 更换高性能的磁盘。 (2) 进行磁盘清理。 (3) 尽可能用内存的读写代替直接磁盘 I/O，使频繁访

	问的文件或数据放入内存中进行操作处理。

表 4-4 会话/进程级别性能状态

异常状态	处理方法
CPU、内存、I/O 使用率过高情况	查看哪个进程占用 CPU/内存高或 I/O 使用率高，若是无用的进程，则 kill 掉，否则排查具体原因。例如 SQL 执行占用内存大，查看是否 SQL 语句需要优化。

表 4-5 SSD 性能状态

异常状态	处理方法
SSD 读写性能故障	使用以下命令查看 SSD 是否有故障，排查具体故障原因。 `gs_checkperf -i SSD -U gbase`

4.5 检查和清理日志

日志是检查系统运行及故障定位的关键手段。建议按月度例行查看操作系统日志及数据库的运行日志。同时，随着时间的推移，日志的增加会占用较多的磁盘空间。建议按月度清理数据库的运行日志。

4.5.1 检查操作系统日志

建议按月检查操作系统日志，排除操作系统运行异常隐患。以超级用户（如 root），查看操作系统日志文件。

```
[root@gbase8s ~]$ vim /var/log/messages
```

关注其中近一个月出现的 kernel、error、fatal 等字样，根据系统报警信息进行处理。

4.5.2 检查数据库运行日志

数据库运行时，某些操作在执行过程中可能会出现错误，数据库依然能够运行。但是此时数据库中的数据可能已经发生不一致的情况。建议按月检查 GBase 8s 运行日志，及时发现隐患。

前提条件

- 收集日志的主机网络通畅且未宕机，数据库安装用户互信正常。
- 日志收集工具依赖操作系统工具如 gstack 等。如果未安装该工具，则提示错误后，跳过该收集项。

操作步骤

步骤 1 以管理员用户 gbase，登录数据库主节点。

步骤 2 使用 gs_collector 命令收集数据库日志，格式：

```
gs_collector --begin-time="begin_time" --end-time="end_time" [-h] [-f] [-o] [-l]
[--keyword=KEYWORD]
```

其中参数说明：

- begin_time 参数：指定日志的开始时间
- end_time 参数：指定为日志的结束时间
- -h 参数：指定要连接到主机的名称
- -f 参数：指定列出要连接到所有主机名称的文件
- -o 参数：指定收集日志文件的存储路径
- -l 参数：指定读取日志文件的路径
- --keyword=KEYWORD 参数：指定存储日志文件的密码

步骤 3 根据步骤 2 的界面输出提示，进入相应的日志收集目录，解压收集的日志，并检查数据库日志。

示例

- 以--begin-time 与--end-time 为参数执行 gs_collector 命令。

```
[gbase@gbase8s ~]$ gs_collector --begin-time="20160616 01:01"
--end-time="20160616 23:59"
```

当显示类似如下信息表示日志已经归档。

```
......
Successfully collected files.
All results are stored in
/home/gbase/gbase8s/log/collector_20220818_165907.tar.gz.
```

- 以--begin-time，--end-time 与-h 为参数执行 gs_collector 命令。

```
[gbase@gbase8s ~]$ gs_collector --begin-time="20160616 01:01"
--end-time="20160616 23:59" -h host_name
```

当显示类似如下信息表示日志已经归档。

```
......
Successfully collected files.
All results are stored in
/home/gbase/gbase8s/log/collector_20220818_173304.tar.gz.
```

- 以--begin-time，--end-time 与-f 为参数执行 gs_collector 命令。

```
[gbase@gbase8s ~]$ gs_collector --begin-time="20160616 01:01"
--end-time="20160616 23:59" -f /opt/software/gbase/output
```

当显示类似如下信息表示日志已经归档。

```
......
Successfully collected files.
All results are stored in
/home/gbase/gbase8s/log/collector_20220818_173901.tar.gz.
```

- 以--begin-time，--end-time 与--keyword 为参数执行 gs_collector 命令。

```
[gbase@gbase8s ~]$ gs_collector --begin-time="20160616 01:01"
--end-time="20160616 23:59" --keyword="os"
```

当显示类似如下信息表示日志已经归档。

```
......
Successfully collected files.
All results are stored in
/home/gbase/gbase8s/log/collector_20220818_174000.tar.gz.
```

- 以--begin-time，--end-time 与-o 为参数执行 gs_collector 命令。

```
[gbase@gbase8s ~]$ gs_collector --begin-time="20160616 01:01"
--end-time="20160616 23:59" -o /home/gbase/gbase8s/log/output
```

当显示类似如下信息表示日志已经归档。

```
Successfully collected files.
All results are stored in
/home/gbase/gbase8s/log/output/collector_20220818_175622.tar.gz.
```

- 以--begin-time，--end-time 与-l 为参数（文件名必须以.log 为后缀）执行 gs_collector 命令。

```
[gbase@gbase8s ~]$ gs_collector --begin-time="20160616 01:01"
--end-time="20160616 23:59" -l /home/gbase/gbase8s/log/logfile.log
```

当显示类似如下信息表示日志已经归档。

```
Successfully collected files.
All results are stored in
/home/gbase/gbase8s/log/collector_20220818_175871.tar.gz.
```

4.5.3 清理运行日志

数据库运行过程中会产生大量运行日志，占用大量的磁盘空间，建议清理过期日志文件，只保留一个月的日志。

操作步骤

步骤 1 以管理员用户 gbase 登录数据库主节点。

步骤 2 清理日志。

(1) 将超过 1 个月的日志备份到其他磁盘。

(2) 进入日志存放目录。

```
cd $GAUSSLOG/pg_log/
```

(3) 进入相应的子目录，使用如下方式删除 1 个月之前产生的日志。

```
rm 日志名称
```

日志文件的命名格式为“postgresql-年-月-日_HHMMSS”。

例如：删除 DN1 节点下的日志文件 postgresql-2022-08-18_000000.log，命令如下：

```
[gbase@gbase8s ~]$ cd $GAUSSLOG/pg_log/dn1_1/
[gbase@gbase8s dn1_1]$ ll
-rw-------. 1 gbase gbase  44434 Nov 29 23:56 postgresql-2022-11-29_151607.log
-rw-------. 1 gbase gbase  74443 Nov 30 23:56 postgresql-2022-11-30_000000.log
```

```
-rw-------. 1 gbase gbase  74013 Dec  1 23:56 postgresql-2022-12-01_000000.log
-rw-------. 1 gbase gbase  74985 Dec  2 23:57 postgresql-2022-12-02_000000.log
-rw-------. 1 gbase gbase  76036 Dec  3 23:57 postgresql-2022-12-03_000000.log
-rw-------. 1 gbase gbase  75074 Dec  4 23:57 postgresql-2022-12-04_000000.log
-rw-------. 1 gbase gbase  73528 Dec  5 23:57 postgresql-2022-12-05_000000.log
-rw-------. 1 gbase gbase 159550 Dec  6 15:30 postgresql-2022-12-06_000000.log
[gbase@gbase8s dn1_1]$ rm postgresql-2022-11-29_151607.log
rm: remove regular file ‘postgresql-2022-11-29_151607.log’ ? y
```

4.6 检查时间一致性

数据库事务一致性通过逻辑时钟保证，与操作系统时间无关。但是系统时间不一致，会导致诸多潜在问题，主要是后台运维和监控功能异常。因此在月度检查时，建议检查各个节点的时间一致性。

操作步骤

步骤 1 以管理员用户 gbase，登录数据库主节点。

步骤 2 创建记录 GBase 8s 各节点的配置文件。用户可随意指定 mpphosts 文件所在目录，建议放在/tmp 下）。

```
vim /tmp/mpphosts
```

打开配置文件，按键“i”进入编辑模式，增加各节点的主机名称。

```
10.0.7.16
10.0.7.17
10.0.7.18
```

按键“:wq!”保存并退出。

步骤 3 执行如下命令，输出各节点上的时间到“/tmp/sys_ctl-os1.log”文件中。

```
for ihost in `cat /tmp/mpphosts`; do ssh -n -q $ihost "hostname;date"; done >
/tmp/sys_ctl-os1.log
```

步骤 4 根据输出确认各个节点的时间一致性，节点之间时间差异不能超过 30 秒。

```
cat /tmp/sys_ctl-os1.log
```

返回如下信息：

```
10.0.7.16
```

```
Thu Aus 9 16:46:38 CST 2022
10.0.7.17
Thu Aus 9 16:46:38 CST 2022
10.0.7.18
Thu Aus 9 16:46:39 CST 2022
```

4.7 检查应用连接数

如果应用程序与数据库的连接数超过最大值，则新的连接无法建立。建议每天检查连接数，及时释放空闲的连接或者增加最大连接数。

操作步骤

步骤 1 以管理员用户 gbase，登录数据库主节点。

步骤 2 连接数据库，并执行如下 SQL 语句查看连接数。

```
SELECT count(*) FROM (SELECT pg_stat_get_backend_idset() AS backendid) AS s;
```

返回如下信息：

```
 count
-------
    22
(1 row)
```

其中 22 表示当前有 22 个应用连接到数据库。

步骤 3 查看现有最大连接数。

```
SHOW max_connections;
```

返回如下信息：

```
 max_connections
-----------------
 3000
(1 row)
```

其中 3000 表示当前最大连接数。

异常处理

如果显示的连接数接近数据库的最大连接数 max_connections，则需要考虑清理现有连

接数或者增加新的连接数。

步骤 1 查看空闲（state 字段为”idle”）且长时间没有更新过的连接信息，使用如下命令。

```
SELECT * FROM pg_stat_activity where state='idle' order by state_change;
```

返回类似如下的信息：

```
 datid | datname  |       pid       |    sessionid    | usesysid | usename |
application_name    | client_addr | client_hostname | client_p
ort |         backend_start         | xact_start |          query_start
|         state_change          | waiting | enqueue | state |
 resource_pool | query_id |                                  query
| connection_info | unique_sql_id | trace_id

-------+----------+-----------------+-----------------+----------+---------+
-----------------------+-------------+-----------------+---------
----+-------------------------------+-----------+-------------------------
-----+-------------------------------+---------+---------+-------+
--------------+----------+-----------------------------------------------------
--------------------+----------------+--------------+---------
-
 16935 | postgres | 140658935854848 | 140658935854848 |       10 | gbase   |
statement flush thread |             |                 |
    | 2022-08-19 16:06:35.807855+08 |            |
| 2022-08-19 16:06:35.807891+08 | f       |         | idle  |
 default_pool  |        0 |
|                 |             0 |
......
```

步骤 2 释放空闲的连接数。

查看每个连接，并与其连接使用者确认是否可以断开连接。执行如下 SQL 语句释放连接。其中，pid 为上一步查询中空闲连接所对应的 pid 字段值。例如：

```
SELECT pg_terminate_backend(140658560005888);
```

返回如下信息：

```
 pg_terminate_backend
----------------------
 t
(1 row)
```

如果没有可释放的连接，请跳过释放连接操作，直接执行步骤 3 和步骤 4。

步骤 3 退出数据库登录状态。调整最大连接数，语法格式：

```
gs_guc set [-N NODE-NAME | all] {[-I INSTANCE-NAME | all] | [-D DATADIR]}
[--lcname=LCNAME] [--ignore-node=NODES] {-c "parameter = value" -c "parameter
= value" ...}
```

请根据实际环境和需要指定。例如，将 DN 所有节点的最大连接数设置为 5000：

```
[gbase@gbase8s ~] gs_guc set -N all -I all -c "max_connections=5000"
The gs_guc run with the following arguments: [gs_guc -N all -I all -c
max_connections=5000 set ].
Begin to perform the total nodes: 2.
Popen count is 2, Popen success count is 2, Popen failure count is 0.
Begin to perform gs_guc for datanodes.
Command count is 2, Command success count is 2, Command failure count is 0.

Total instances: 2. Failed instances: 0.
ALL: Success to perform gs_guc!
```

步骤 4 重启数据库服务使新的设置生效。

```
gha_ctl stop all -l dcslist
gha_ctl start all -l dcslist
```

说明

重启 GBase 8s 操作，会导致用户执行操作中断。请在操作之前规划好合适的时间窗口。

4.8 例行维护表

为了保证数据库的有效运行，数据库必须在插入/删除操作后，定期执行 VACUUM FULL 和 ANALYZE 更新统计信息，以便获得更优的性能。

相关概念

使用VACUUM、VACUUM FULL和ANALYZE命令，定期维护表，主要原因如下：

- VACUUM FULL 可回收已更新或已删除的数据所占据的磁盘空间，同时将小数据文件合并。
- VACUUM 对每个表维护了一个可视化映射，来跟踪包含对别的活动事务可见的数组的

页。一个普通的索引扫描首先通过可视化映射来获取对应的数组，来检查是否对当前事务可见。若无法获取，再通过堆数组抓取的方式来检查。因此更新表的可视化映射，可加速唯一索引扫描。

- VACUUM 可避免执行的事务数超过数据库阈值时，事务 ID 重叠造成的原有数据丢失。
- ANALYZE 可收集与数据库中表内容相关的统计信息。统计结果存储在系统表 PG_STATISTIC 中。查询优化器会使用这些统计数据，生成最有效的执行计划。

操作步骤

步骤 1 使用 VACUUM 或 VACUUM FULL 命令，进行磁盘空间回收。

- VACUUM

对表执行 VACUUM 操作。可以与数据库操作命令并行运行。执行期间，可正常使用的语句：SELECT、INSERT、UPDATE 和 DELETE。不可正常使用的语句：ALTER TABLE。

例如，对普通表执行 VACUUM 操作：

```
postgres=# VACUUM customer;
VACUUM
```

例如，对表分区执行 VACUUM 操作：

```
postgres=# VACUUM customer_par PARTITION (P1);
VACUUM
```

- VACUUM FULL

需要向正在执行的表增加排他锁，且需要停止其他所有数据库操作。

例如，对普通表执行 VACUUM FULL 操作：

```
postgres=# VACUUM FULL customer;
VACUUM
```

例如，对表分区执行 VACUUM 操作：

```
postgres=# VACUUM FULL customer_par PARTITION (P1);
VACUUM
```

步骤 2 使用 ANALYZE 语句更新统计信息。例如：

```
postgres=# ANALYZE customer;
ANALYZE
```

步骤 3 使用 ANALYZE VERBOSE 语句更新统计信息，并输出表的相关信息。例如：

```
postgres=# ANALYZE VERBOSE customer;
ANALYZE
```

也可以同时执行 VACUUM ANALYZE 命令进行查询优化。例如：

```
postgres=# VACUUM ANALYZE customer;
VACUUM
```

说明

VACUUM 和 ANALYZE 会导致 I/O 流量的大幅增加，这可能会影响其他活动会话的性能。因此，建议通过“vacuum_cost_delay”参数设置清理延迟，参见《GBase 8s V8.8__开发者指南》中“GUC 参数说明 > 资源消耗 > 基于开销的清理延迟”。

步骤 4 删除表

```
postgres=# DROP TABLE customer;
postgres=# DROP TABLE customer_par;
postgres=# DROP TABLE part;
```

维护建议

- 定期对部分大表做 VACUUM FULL。
- 在性能下降后为全库做 VACUUM FULL。
- 建议每月做一次 VACUUM FULL。
- 定期对系统表做 VACUUM FULL，主要是 PG_ATTRIBUTE。
- 启用系统自动清理线程（AUTOVACUUM）自动执行 VACUUM 和 ANALYZE，回收被标识为删除状态的记录空间，并更新表的统计数据。

4.9 例行重建索引

背景信息

数据库经过多次删除操作后，索引页面上的索引键将被删除，造成索引膨胀。例行重建

索引，可有效的提高查询效率。

数据库支持的索引类型为 B-tree 索引，例行重建索引可有效的提高查询效率。

- 如果数据发生大量删除后，索引页面上的索引键将被删除，导致索引页面数量的减少，造成索引膨胀。重建索引可回收浪费的空间。
- 新建的索引中逻辑结构相邻的页面，通常在物理结构中也是相邻的，所以一个新建的索引比更新了多次的索引访问速度要快。

重建索引

重建索引有以下两种方式。选其中之一即可。

- 先运行 DROP　INDEX 语句删除索引，再运行 CREATE　INDEX 语句创建索引。

在删除索引过程中，会在父表上增加一个临时排他锁，阻止相关读写操作。在创建索引过程中，会锁住写操作但是不会锁住读操作，此时读操作只能使用顺序扫描。

- 使用 REINDEX 语句重建索引。
  - 使用 REINDEX TABLE 语句重建索引，会在重建过程中增加排他锁，阻止相关读写操作。
  - 使用 REINDEX INTERNAL TABLE 语句重建 desc 表(包括列存表的 cudesc 表）的索引，会在重建过程中增加排他锁，阻止相关读写操作。

操作步骤

重建索引有以下两种方式。根据实际情况选一即可。

以导入表“areaS”上的“area_id”字段存在普通索引“areaS_idx”为例。

方式一 删除索引再创建

先运行 DROP INDEX 语句删除索引，再运行 CREATE INDEX 语句创建索引。

步骤 1 删除索引。

```
postgres=# DROP INDEX areaS_idx;
DROP INDEX
```

步骤 2 创建索引。

```
postgres=# CREATE INDEX areaS_idx ON areaS (area_id);
CREATE INDEX
```

方式二 使用 REINDEX 重建

- 使用 REINDEX TABLE 语句重建索引。

```
postgres=# REINDEX TABLE areaS;
REINDEX
```

- 使用 REINDEX INTERNAL TABLE 重建 desc 表（包括列存表的 cudesc 表）的索引。

```
postgres=# REINDEX INTERNAL TABLE areaS;
REINDEX
```

说明

在重建索引前，用户可以通过临时增大 maintenance_work_mem 和 psort_work_mem 的取值，来加快索引的重建。

4.10 导出并查看 WDR 诊断报告

访问 WDR 快照数据需要 sysadmin 或 monadmin 权限，因此需要使用 root 账号或其他拥有权限的账号来生成 WDR 诊断报告。

操作步骤

步骤 1 新建报告文件，执行如下命令：

```
[root@gbase8s ~]# touch /home/gbase/file/wdrTestNode.html
```

步骤 2 以管理员用户，连接数据库 postgres。例如：

```
[root@gbase8s ~]# su gbase
[gbase@gbase8s ~]$ gsql -d postgres -p 5432 -r
```

步骤 3 选择 snapshot.snapshot 表中两个不同的 snapshot，且这两个 snapshot 时间范围内没有发生数据库重启，则可以利用这两个 snapshot 生成该范围内的报告。

```
select * from snapshot.snapshot order by start_ts desc limit 10;
```

步骤 4 在本地生成 HTML 格式的 WDR 报告，执行如下命令。

```
\a
\t
\o filename
```

其中，\a 表示不显示表行列符号，\t 表示不显示列名，\o 用于指定输出文件。

```
select generate_wdr_report(begin_snap_id Oid, end_snap_id Oid, int report_type,
int report_scope, int node_name );
```

参数说明

表 4-6 参数说明

参数	说明	取值范围
begin_snap_id	要查看的某段时间性能的开始的 snapshot 的 id（表 snapshot.snaoshot 中的 snapshot_id）	——
end_snap_id	结束 snapshot 的 id，默认 end_snap_id 大于 begin_snap_id （表 snapshot.snaoshot 中的 snapshot_id）	——
report_type	指定生成 report 的类型。	● summary ● detail ● all，即同时包含 summary 和 detail。
report_scope	指定生成 report 的范围。	cluster：集群 node：集群中某个节点。
node_name	● 在 report_scope 指定为 single node 时，需要把该参数指定为对应节点的名称。 ● 在 report_scope 为 cluster 时，该值可以指定为省略或者为 NULL。	——

执行如下命令关闭输出选项及格式化输出命令。

```
\o \a \t
```

在/home/om/下根据需要查看 WDR 报告内容。

表 4-7 WDR 报表主要内容

项目	描述
Database Stat（集群范围）	数据库维度性能统计信息：事务，读写，行活动，写冲突，死锁等。
Load Profile（集群范围）	集群维度的性能统计信息：CPU 时间，DB 时间，逻辑读/物理读，IO 性能，登入登出，负载强度，负载性能表现等。
Instance Efficiency Percentages（集群/节点范围）	集群级或者节点缓冲命中率。
IO Profile（集群/节点范围）	集群或者节点维度的 IO 的使用情况。
Top 10 Events by Total Wait Time（节点范围）	最消耗时间的事件。
Wait Classes by Total Wait Time（节点范围）	最消耗时间的等待时间分类。
Host CPU（节点范围）	主机 CPU 消耗。
Memory Statistics（节点范围）	内核内存使用分布。
Time Model（节点范围）	节点范围的语句的时间分布信息。
Wait Events（节点范围）	节点级别的等待事件的统计信息。
Cache IO Stats (集群/节点范围)	用户的表、索引的 IO 的统计信息。
Utility status （节点范围）	复制槽和后台 checkpoint 的状态信息。
Object stats（集群/节点范围）	表、索引维度的性能统计信息。
Configuration settings（节点范围）	节点配置。
SQL Statistics（集群/节点范围）	SQL 语句各个维度性能统计：端到端时间，行活动，缓存命中，CPU 消耗，时间消耗细分。
SQL Detail（集群/节点范围）	SQL 语句文本详情。

示例

示例一：生成集群级别的报告。

```
select generate_wdr_report(1, 2, 'all', 'cluster',null);
```

示例二：生成某个节点的报告。

```
select generate_wdr_report(1, 2, 'all', 'node', pgxc_node_str()::cstring);
```

4.11 数据安全维护建议

为保证 GBase 8s 数据库中的数据安全，避免丢失数据，非法访问数据等事故发生，请仔细阅读以下内容。

避免数据被丢失

建议用户规划周期性的物理备份，且对备份文件进行可靠的保存。在系统发生严重错误的情况下，可以利用备份文件，将系统恢复到备份前的状态。

避免数据被非法访问

建议对数据库用户进行权限分级管理。数据库管理员根据业务需要，建立用户并赋予权限，保证各用户对数据库的合理访问。

对于 GBase 8s 的服务端和客户端（或基于客户端库开发的应用程序），最好也部署在可信任的内网中。如果服务端和客户端一定要部署在非信任的网络中，需要在服务启动前，打开 SSL 加密，保证数据在非信任网络上的传输安全。需要注意的是，打开 SSL 加密会降低数据库的性能。

避免系统日志泄露个人数据

- 将调试日志发给他人进行分析前，请删除个人数据。

  说明

  因为日志级别（log_min_messages）设置为 DEBUGx（x 为 DEBUG 级别，取值范围为 1~5）时，调试日志中记录的信息可能包含用户的个人数据。

- 将系统日志发给其他人进行分析前，请删除个人数据。因为在默认配置下，当 SQL 语句执行错误时，日志中会记录出错的 SQL 语句，而这些 SQL 语句中可能包含用户个人数据。

- 将 log_min_error_statement 参数的值设置为 PANIC，可以避免将出错的 SQL 语句记录在系统日志中。若禁用该功能，当出现故障时，很难定位故障原因。

4.12 慢 SQL 诊断

背景信息

在 SQL 语句执行性能不符合预期时，可以查看 SQL 语句执行信息，便于事后分析 SQL 语句执行时的行为，从而诊断 SQL 语句执行出现的相关问题。

前提条件

- 数据库实例运行正常。
- 查询 SQL 语句信息，需要正确设置 GUC 参数 track_stmt_stat_level。
- 只能用系统管理员和监控管理员权限进行操作。

应用场景

- 查看数据库实例中慢 SQL 语句执行信息，语法格式：

```
select * from dbe_perf.get_global_slow_sql_by_timestamp(start_timestamp, end_timestamp);
```

- 查看数据库实例中 SQL 语句执行信息，语法格式：

```
select * from dbe_perf.get_global_full_sql_by_timestamp(start_timestamp, end_timestamp);
```

例如：

```
postgres=# select * from DBE_PERF.get_global_full_sql_by_timestamp('2020-12-01 09:25:22', '2020-12-31 23:54:41');
-[ RECORD 1 ]-------+-----------------------------------------------------------------------------------------------------------
node_name           | dn_6001_6002_6003
db_name             | postgres
schema_name         | "$user",public
origin_node         | 1938253334
user_name           | user_dj
```

```
application_name     | gsql
client_addr          |
client_port          | -1
unique_query_id      | 3671179229
debug_query_id       | 72339069014839210
query                | select name, setting from pg_settings where name in (?)
start_time           | 2020-12-19 16:19:51.216818+08
finish_time          | 2020-12-19 16:19:51.224513+08
slow_sql_threshold   | 1800000000
transaction_id       | 0
thread_id            | 139884662093568
session_id           | 139884662093568
n_soft_parse         | 0
n_hard_parse         | 1
query_plan           | Datanode Name: dn_6001_6002_6003
                     | Function Scan on pg_show_all_settings a
(cost=0.00..12.50 rows=5 width=64)
                     |   Filter: (name = '***'::text)
...
```

- 查看当前主节点 SQL 语句执行信息，语法格式：

```
select * from statement_history;
```

例如：

```
postgres=# select * from statement_history;
-[ RECORD
1 ]--------+-----------------------------------------------------------------
-------------------------------------------
-------------------------------------------------------------------
db_name              | postgres
schema_name          | "$user",public
origin_node          | 1938253334
user_name            | user_dj
application_name     | gsql
client_addr          |
client_port          | -1
unique_query_id      | 3671179229
debug_query_id       | 72339069014839210
query                | select name, setting from pg_settings where name in (?)
start_time           | 2020-12-19 16:19:51.216818+08
finish_time          | 2020-12-19 16:19:51.224513+08
```

```
slow_sql_threshold   | 1800000000
transaction_id       | 0
thread_id            | 139884662093568
session_id           | 139884662093568
n_soft_parse         | 0
n_hard_parse         | 1
query_plan           | Datanode Name: dn_6001_6002_6003
                     | Function Scan on pg_show_all_settings a
(cost=0.00..12.50 rows=5 width=64)
                     |   Filter: (name = '***'::text)
...
```

- 查看当前备节点 SQL 语句执行信息

```
select * from dbe_perf.standby_statement_history(is_only_slow, start_timestamp,
end_timestamp);
```

例如：

```
postgres=# select * from dbe_perf.standby_statement_history(true, '2022-08-01
09:25:22', '2022-08-31 23:54:41');
db_name              | postgres
schema_name          | "$user",public
origin_node          | 0
user_name            | user_dj
application_name     | gsql
client_addr          |
client_port          | -1
unique_query_id      | 1660376009
debug_query_id       | 281474976710740
query                | select name, setting from pg_settings where name in (?)
start_time           | 2022-08-19 16:19:51.216818+08
finish_time          | 2022-08-19 16:19:51.224513+08
slow_sql_threshold   | 1800000000
transaction_id       | 0
thread_id            | 140058747205376
session_id           | 140058747205376
n_soft_parse         | 0
n_hard_parse         | 1
query_plan           | Datanode Name: sgnode
                     | Function Scan on pg_show_all_settings a
(cost=0.00..12.50 rows=5 width=64)
                     |   Filter: (name = '***'::text)
```

...

5 备份与恢复

5.1 概述

数据备份是保护数据安全的重要手段之一。为了更好地保护数据安全，GBase 8s 数据库提供三种备份恢复类型、多种备份恢复方案，能够在备份和恢复过程中保障数据的可靠性。

备份与恢复类型可分为逻辑备份与恢复、物理备份与恢复、闪回恢复。

- 逻辑备份与恢复：通过逻辑导出对数据进行备份，逻辑备份只能基于备份时刻进行数据转储，所以恢复时也只能恢复到备份时保存的数据。对于故障点和备份点之间的数据，逻辑备份无能为力，逻辑备份适合备份那些很少变化的数据，当这些数据因误操作被损坏时，可以通过逻辑备份进行快速恢复。如果通过逻辑备份进行全库恢复，通常需要重建数据库，导入备份数据来完成，对于可用性要求很高的数据库，这种恢复时间太长，通常不被采用。由于逻辑备份具有平台无关性，所以更为常见的是，逻辑备份被作为一个数据迁移及移动的主要手段。
- 物理备份与恢复：通过物理文件拷贝的方式对数据库进行备份，以磁盘块为基本单位将数据从主机复制到备机。通过备份的数据文件及归档日志等文件，数据库可以进行完全恢复。物理备份速度快，一般被用作对数据进行备份和恢复，用于全量备份的场景。通过合理规划，可以低成本进行备份与恢复。
- 闪回恢复：利用回收站的闪回恢复删除的表。数据库的回收站功能类似于 windows 系统的回收站，将删除的表信息保存到回收站中。利用 MVCC 机制闪回恢复到指定时间点或者 CSN 点。

以下为三类数据备份恢复类型对比。当异常发生时，便于管理员及时制定恢复方案。

表 5-1 三种备份恢复类型对比

备份类型	应用场景	支持介质	工具名称	恢复时间	优缺点
逻辑备份	适合于数据量小的	磁盘	gs_cump	纯文本格式数据恢	导出数据库相关信息的工具，用户可以自定义导出一个数据库或其中的

备份类型	应用场景	支持介质	工具名称	恢复时间	优缺点
与恢复	场景。可以备份单表和多表，单database 和所有database。 备份后的数据需要使用gsql或者gs_restore工具恢复。数据量大时，恢复需要较长时间。	SSD		复时间长。归档格式数据恢复时间中等。	对象（模式、表、视图等）。 支持导出的数据库可以是默认数据库 postgres，也可以是自定义数据库。 导出的格式可选择纯文本格式或者归档格式。纯文本格式的数据只能通过 gsql 进行恢复，恢复时间较长。归档格式的数据只能通过 gs_restore 进行恢复， 恢复时间较纯文本格式短。
			gs_dumpall	数据恢复时间长。	导出所有数据库相关信息工具，它可以导出GBase 8s数据库的所有数据，包括默认数据库 postgres 的数据、自定义数据库的数据、以及 GBase 8s 所有数据库公共的全局对象。 只能导出纯文本格式的数据，导出的数据只能通过 gsql 进行恢复，恢复时间较长。
物理备份与恢复	适用于数据量大的场景，主要用于全量数据备份恢复，也可对整个数据库中的WAL 归档日志和运		gs_backup	数据量小数据恢复时间快。	导出数据库相关信息的 OM 工具，可以导出数据库参数文件和二进制文件。支持备份、恢复重要数据、显示帮助信息和版本号信息。 在进行备份时，可以选择备份内容的类型，在进行还原时，需要保证各节点备份目录中存在备份文件。在集群恢复时，通过静态配置文件中的集群信息进行恢复。只恢复参数文件恢复时间较短。

备份类型	应用场景	支持介质	工具名称	恢复时间	优缺点
	行日志进行备份。		gs_basebackup	恢复时可以直接拷贝替换原有的文件，或者直接在备份的库上启动数据库，恢复时间快。	对服务器数据库文件的二进制进行全量拷贝，只能对数据库某一个时间点的时间作备份。结合 PITR 恢复，可恢复全量备份时间点后的某一时间点。
			gs_probackup	恢复时可以直接恢复到某个备份点，在备份的库上启动数据库，恢复时间快。	gs_probackup 用于管理 GBase 8s 数据库备份和恢复，可对实例进行定期备份。可用于备份单机数据库或者集群主节点数据库，为物理备份。 可备份外部目录的内容，如脚本文件、配置文件、日志文件、dump 文件等。 支持增量备份、定期备份和远程备份。增量备份时间相对于全量备份时间比较短，只需要备份修改的文件。当前默认备份是数据目录，如果表空间不在数据目录，需要手动指定备份的表空间目录进行备份。当前只支持在主机上执行备份。
闪回恢复	适用于误删除表的场景。需要将表中的数据恢复到指定时		无	可以将表的状态恢复到指定时间点或者是表结构删除前	闪回技术能够有选择性的高效撤销一个已提交事务的影响，从人为错误中恢复。在采用闪回技术之前，只能通过备份恢复、PITR 等手段找回已提交的数据库修改，恢复时长需要数分钟甚至数小时。采用闪回技术后，

备份类型	应用场景	支持介质	工具名称	恢复时间	优缺点
	间点或者CSN。			的状态，恢复时间快。	恢复已提交的数据库修改前的数据，只需要秒级，而且恢复时间和数据库大小无关。 闪回支持两种恢复模式： ● 基于 MVCC 多版本的数据恢复：适用于误删除、误更新、误插入数据的查询和恢复，用户通过配置旧版本保留时间，并执行相应的查询或恢复命令，查询或恢复到指定的时间点或 CSN 点。 ● 基于类似 windows 系统回收站的恢复：适用于误 DROP、误 TRUNCATE 的表的恢复。用户通过配置回收站开关，并执行相应的恢复命令，可以将误 DROP、误 TRUNCATE 的表找回。

当需要进行备份恢复操作时，主要从以下方面考虑数据备份方案。

- 备份对业务的影响在可接受范围。
- 数据库恢复效率。为尽量减小数据库故障的影响，要使恢复时间减到最少，从而使恢复的效率达到最高。
- 数据可恢复程度。当数据库失效后，要尽量减少数据损失。
- 数据库恢复成本。在现网选择备份策略时参考的因素比较多，如备份对象、数据大小、网络配置等，下表列出了可用的备份策略和每个备份策略的适用场景。

表 5-2 备份策略典型场景

备份策略	关键性能因素	典型数据量	性能规格
数据库实例备份	● 数据大小 ● 网络配置	数据：PB 级 对象：约 100 万个	备份： ● 每个主机 80 Mbit/s（NBU/EISOO+磁盘） ● 约 90%磁盘 I/O 速率（SSD/HDD）
表备份	● 表所在模式 ● 网络配置(NBU)	数据：10 TB 级	备份：基于查询性能速度+I/O 速度 说明 多表备份时，备份耗时计算方式： 总时间 = 表数量*起步时间+数据总量/数据备份速度 其中 ● 磁盘起步时间为 5s 左右，NBU 起步时间比 DISK 长（取决于 NBU 部署方案）。 ● 数据备份速度为单节点 50MB/s 左右（基于 1GB 大小的表，物理机备份到本地磁盘得出此速率）。 表越小，备份性能更低。

5.2 物理备份恢复

5.2.1 gs_basebackup

背景信息

GBase 8s 部署成功后，在数据库运行的过程中，会遇到各种问题及异常状态。GBase 8s 提供了 gs_basebackup 工具做基础的物理备份。gs_basebackup 的实现目标是对服务器数据库文件的二进制进行拷贝，其实现原理使用了复制协议。远程执行 gs_basebackup 时，需要使用系统管理员账户。gs_basebackup 当前支持热备份模式和压缩格式备份。

说明

- gs_basebackup 仅支持主机和备机的全量备份，不支持增量。

- gs_basebackup 当前支持热备份模式和压缩格式备份。
- gs_basebackup 在备份包含绝对路径的表空间时，如果在同一台机器上进行备份，可以通过 tablespace-mapping 重定向表空间路径，或使用归档模式进行备份。
- 若打开增量检测点功能且打开双写，gs_basebackup 也会备份双写文件。
- 若 pg_xlog 目录为软链接，备份时将不会建立软链接，会直接将数据备份到目的路径的 pg_xlog 目录下。
- 备份过程中收回用户备份权限，可能导致备份失败或者备份数据不可用。
- 如果因为网络临时故障等原因导致 Server 端无应答，gs_basebackup 将在最长等待 120 秒后退出。

前提条件

- 可以正常连接 GBase 8s 数据库。
- 备份过程中用户权限没有被回收。
- pg_hba.conf 中需要配置允许复制链接，且该连接必须由一个系统管理员建立。
- 如果 xlog 传输模式为 stream 模式，需配置 max_wal_senders 的数量，至少有一个可用。
- 如果 xlog 传输模式为 fetch 模式，有必要把 wal_keep_segments 参数设置得足够高，这样在备份末尾之前日志不会被移除。
- 在进行还原时，需要保证各节点备份目录中存在备份文件，若备份文件丢失，则需要从其他节点进行拷贝。

语法

```
gs_basebackup -D directory [ARGS]
```

参数说明

gs_basebackup 参数可以分为如下几类：

- -D directory：备份文件输出的目录，必选项。

常用参数

- -c，--checkpoint=fast|spread ： 设置检查点模式为 fast 或者 spread（默认）。
- -l，--label=LABEL ： 为备份设置标签。
- -P，--progress ： 启用进展报告。
- -v， --verbose ： 启用冗长模式。
- -V， --version ： 打印版本后退出。
- -?，--help ： 显示 gs_basebackup 命令行参数。
- -T， –tablespace-mapping=olddir=newdir ： 在备份期间将目录 olddir 中的表空间重定位到 newdir 中。

为使之有效，olddir 必须正好匹配表空间所在的路径（但如果备份中没有包含 olddir 中的表空间也不是错误）。olddir 和 newdir 必须是绝对路径。如果一个路径凑巧包含了一个=符号，可用反斜线对它转义。对于多个表空间可以多次使用这个选项。

- -F， –format=plain|tar ： 设置输出格式为 plain(默认)或者 tar。

没有设置该参数的情况下，默认– format=plain。plain 格式把输出写成平面文件，使用和当前数据目录和表空间相同的布局。当集簇没有额外表空间时，整个数据库将被放在目标目录 中。如果集簇包含额外的表空间，主数据目录将被放置在目标目录中，但是所有其他表空间将被放在它们位于服务器上的相同的绝对路径中。tar 模式将输出写成目标目录中的 tar 文件。主数据目录将被写入到一个名为 base.tar 的文件中，并且其他表空间将被以其 OID 命名。生成的 tar 包，需要用 gs_tar 命令解压。

- -X， –xlog-method=fetch|stream ： 设置 xlog 传输方式。

没有设置该参数的情况下，默认–xlog-method=stream。在备份中包括所需的预写式日志文件（WAL 文件）。这包括所有在备份期间产生的预写式日志。fetch 方式在备份末尾收集预写式日志文件。因此，有必要把 wal_keep_segments 参数设置得足够高，这样在备份末尾之前日志不会被移除。如果在要传输日志时它已经被轮转，备份将失败并且是不可用的。stream 方式在备份被创建时流传送预写式日志。这将开启一个到服务器的第二连接并且在运行备份时并行开始流传输预写式日志。因此，它将使用最多两个由 max_wal_senders 参数配

置的连接。只要客户端能保持接收预写式日志，使用这种模式不需要在主控机上保存额外的预写式日志。

- -x， –xlog ： 使用这个选项等效于和方法 fetch 一起使用-X。
- -Z –compress=level：启用对 tar 文件输出的 gzip 压缩，并且制定压缩级别（0 到 9，0 是不压缩，9 是最佳压缩）。只有使用 tar 格式时压缩才可用，并且会在所有 tar 文件名后面自动加上后缀.gz。
- -z ： 启用对 tar 文件输出的 gzip 压缩，使用默认的压缩级别。只有使用 tar 格式时压缩才可用，并且会在所有 tar 文件名后面自动加上后缀.gz。
- -t， –rw-timeout ： 设置备份期间 checkpoint 的时间限制，默认限制时间为 120s。当数据库全量 checkpoint 耗时较长时，可以适当增大 rw-timeout 限制时间。

连接参数

- -h, --host=HOSTNAME ： 指定正在运行服务器的主机名或者 Unix 域套接字的路径。
- -p，--port=PORT ： 指定数据库服务器的端口号。可以通过 port 参数修改默认端口号。
- -U，--username=USERNAME ： 指定连接数据库的用户。
- -s, --status-interval=INTERVAL ： 发送到服务器的状态包的时间（以秒为单位）。
- -w,--no-password ： 不出现输入密码提示。
- -W, --password ：当使用-U 参数连接本地数据库或者连接远端数据库时，可通过指定该选项出现输入密码提示。

示例

```
[gbase@gbase8s ~] gs_basebackup -D /home/gbase/data/backup -h 10.0.7.16 -p 5432
INFO:  The starting position of the xlog copy of the full build is: 0/6000028.
The slot minimum LSN is: 0/0.
[2022-08-22 16:26:39]:begin build tablespace list
[2022-08-22 16:26:39]:finish build tablespace list
[2022-08-22 16:26:39]:begin get xlog by xlogstream
[2022-08-22 16:26:39]: check identify system success
[2022-08-22 16:26:39]: send START_REPLICATION 0/6000000 success
```

```
[2022-08-22 16:26:39]: keepalive message is received
[2022-08-22 16:26:39]: keepalive message is received
[2022-08-22 16:26:44]:gs_basebackup: base backup successfully
```

从备份文件恢复数据

当数据库发生故障时需要从备份文件进行恢复。因为 gs_basebackup 是对数据库按二进制进行备份，因此恢复时可以直接拷贝替换原有的文件，或者直接在备份的库上启动数据库。

说明

- 若当前数据库实例正在运行，直接从备份文件启动数据库可能会存在端口冲突，这时需要修改配置文件的 port 参数，或者在启动数据库时指定一下端口。
- 若当前备份文件为主备数据库，可能需要修改一下主备之间的复制连接。即配置文件中的 postgre.conf 中的 replconninfo1、 replconninfo2 等。
- 若配置文件 postgresql.conf 的参数 data_directory 打开且有配置，当使用备份目录启动数据库时候， data_directory 和备份目录不同会导致启动失败。可以修改 data_directory 的值为新的数据目录，或者注释掉该参数。

若要在原库的地方恢复数据库，参考步骤如下：

步骤 1 停止数据库服务器。

步骤 2 将原数据库和所有表空间复制到另外一个位置以备份。

步骤 3 清理原数据库中的所有或部分文件。

步骤 4 使用数据库系统用户权限从备份中还原需要的数据库文件。

步骤 5 若数据库中存在链接文件，需要修改使其链接到正确的文件。

步骤 6 重启数据库服务器，并检查数据库内容，确保数据库已经恢复到所需的状态。

说明

- 暂不支持备份文件增量恢复。
- 恢复后需要检查数据库中的链接文件是否链接到正确的文件。

5.2.2 PITR 恢复

背景信息

当数据库崩溃或希望回退到数据库之前的某一状态时，GBase 8s 的即时恢复功能（Point-In-Time Recovery，简称 PITR）可以支持恢复到备份归档数据之后的任意时间点。

说明

- PITR 仅支持恢复到物理备份数据之后的某一时间点。
- 仅主节点可以进行 PITR 恢复，备机需要进行全量 build 达成与主机数据同步。

前提条件

- 基于经过物理备份的全量数据文件。
- 基于已归档的 WAL 日志文件。

PITR 恢复流程

步骤 1 将物理备份的文件替换目标数据库目录。

步骤 2 删除数据库目录下 pg_xlog/中的所有文件。

步骤 3 将归档的 WAL 日志文件复制到 pg_xlog 文件中（此步骤可以省略，通过配置 recovery.conf 恢复命令文件中的 restore_command 项替代）。

步骤 4 在数据库目录下创建恢复命令文件 recovery.conf，指定数据库恢复的程度。

步骤 5 启动数据库。

步骤 6 连接数据库，查看是否恢复到希望预期的状态。

步骤 7 若已经恢复到预期状态，通过 pg_xlog_replay_resume()指令使主节点对外提供服务。

文件配置

recovery.conf 文件配置，参考如下：

归档恢复配置

- restore_command = string

这个 SHELL 命令是获取 WAL 文件系列中已归档的 WAL 文件。字符串中的任何一个%f 是用归档检索中的文件名替换，并且%p 是用服务器上的复制目的地的路径名替换。 任意一个%r 是用包含最新可用重启点的文件名替换。

示例：

```
restore_command = ’cp /mnt/server/archivedir/%f %p’
```

- archive_cleanup_command = string

这个选项参数声明一个 shell 命令。在每次重启时会执行这个 shell 命令。archive_cleanup_command 为清理备库不需要的归档 WAL 文件提供一个机制。 任何一个%r 由包含最新可用重启点的文件名代替。这是最早的文件， 因此必须保留以允许恢复能够重新启动，因此所有早于%r 的文件可以安全的移除。

示例：

```
archive_cleanup_command = 'pg_archivecleanup /mnt/server/archivedir %r'
```

需要注意的是，如果多个备服务器从相同的归档路径恢复时， 需要确保在任何一个备服务器在需要之前，不能删除 WAL 文件。

- recovery_end_command = string

这个参数是可选的，用于声明一个只在恢复完成时执行的 SHELL 命令。

recovery_end_command 是为以后的复制或恢复提供一个清理机制。

恢复目标设置

- recovery_target_name = string

此参数声明命名还原到一个使用 pg_create_restore_point()创建的还原点。

示例:

```
recovery_target_name = 'restore_point_1'
```

- recovery_target_time = timestamp

此参数声明命名还原到一个指定时间戳。示例：

```
recovery_target_time = '2020-01-01 12:00:00'
```

recovery_target_xid = string 这个参数声明还原到一个事务 ID。示例：

```
recovery_target_xid = '3000'
```

- recovery_target_lsn = string

这个参数声明还原到日志的指定 LSN 点。示例：

```
recovery_target_lsn = '0/0FFFFFF'
```

- recovery_target_inclusive = boolean

声明是否在指定恢复目标(true)之后停止，或在这(false)之前停止。该声明仅支持恢复目标为 recovery_target_time，recovery_target_xid 和 recovery_target_lsn 的配置。

示例：

```
recovery_target_inclusive = true
```

说明

recovery_target_name，recovery_target_time，recovery_target_xid，recovery_target_lsn 这四个配置项仅同时支持一项。如果不配置任何恢复目标，或配置目标不存在，则默认恢复到最新的 WAL 日志点。

5.2.3 gs_probackup

背景信息

gs_probackup 是一个用于管理 GBase 8s 数据库备份和恢复的工具。它对 GBase 8s 实例进行定期备份，以便在数据库出现故障时能够恢复服务器。

- 可用于备份单机数据库，也可对主机或者主节点数据库备机进行备份，为物理备份。
- 可备份外部目录的内容，如脚本文件、配置文件、日志文件、dump 文件等。
- 支持增量备份、定期备份和远程备份。
- 可设置备份的留存策略。

前提条件

- 可以正常连接 GBase 8s 数据库。
- 若要使用 PTRACK 增量备份，需在 postgresql.conf 中手动添加参数“enable_cbm_tracking = on”。
- 为了防止 xlog 在传输结束前被清理，请适当调高 postgresql.conf 文件中 wal_keep_segments 的值。

限制说明

- 备份必须由运行数据库服务器的用户执行。
- 备份和恢复的数据库服务器的主版本号必须相同。
- 如果要通过 ssh 在远程模式下备份数据库，需要在本地和远程主机安装相同主版本的数据库，并通过 ssh-copy-id remote_user@remote_host 命令设置本地主机备份用户和远程主机数据库用户的无密码 ssh 连接。
- 远程模式下只能执行 add-instance、backup、restore 子命令。
- 使用 restore 子命令前，应先停止 gaussdb 进程。
- 当存在用户自定义表空间时，备份的时候要加上 --external-dirs 参数，否则，该表空间不会被备份。
- 当备份的规模比较大时，为了防止备份过程中 timeout 发生，请适当调整 postgresql.conf 文件的参数 session_timeout、wal_sender_timeout。并且在备份的命令行参数中适当调整参数--rw-timeout 的值。
- 恢复时，使用-T 选项把备份中的外部目录重定向到新目录时，请同时指定参数--external-mapping。
- 增量备份恢复后，之前创建的逻辑复制槽不可用，需删除重建。
- 当使用远程备份时，请确保远程机器和备份机器的时钟同步，以防止使用--recovery-target-time 恢复的场合,启动 gaussdb 时有可能会失败。

- 当远程备份有效时(remote-proto=ssh)，请确保-h 和--remote-host 指定的是同一台机器。当远程备份无效时，如果指定了-h 选项，请确保-h 指定的是本机地址或本机主机名。
- 当前暂不支持备份逻辑复制槽。

语法

- gs_probackup 子命令帮助说明查询。

```
gs_probackup help [command]
```

- gs_probackup 查看工具版本。

```
gs_probackup version
```

- 初始化备份路径 backup-path 中的备份目录，该目录将存储已备份的内容。如果备份路径 backup-path 已存在，则 backup-path 必须为空目录。

```
gs_probackup init -B backup-path [--help]
```

- 在备份路径 backup-path 内初始化一个新的备份实例，并生成 pg_probackup.conf 配置文件，该文件保存了指定数据目录 pgdata-path 的 gs_probackup 设置。

```
gs_probackup add-instance -B backup-path -D pgdata-path
--instance=instance_name
                [-E external-directories-paths]
                [--remote-proto=protocol] [--remote-host=destination]
                [--remote-path=path] [--remote-user=username]
                [--remote-port=port] [--ssh-options=ssh_options]
                [--remote-libpath=libpath]
                [--help]
```

- 在备份路径 backup-path 内删除指定实例相关的备份内容。

```
gs_probackup del-instance -B backup-path --instance=instance_name
                [--help]
```

- 将指定的连接、压缩、日志等相关设置添加到 pg_probackup.conf 配置文件中，或修改已设置的值。不推荐手动编辑 pg_probackup.conf 配置文件。

```
gs_probackup set-config -B backup-path --instance=instance_name
                [-D pgdata-path] [-E external-directories-paths]
                [--archive-timeout=timeout]
                [--retention-redundancy=retention-redundancy]
```

```
                [--retention-window=retention-window]
                [--wal-depth=wal-depth]
                [--compress-algorithm=compress-algorithm]
                [--compress-level=compress-level]
                [-d dbname] [-h host] [-p port] [-U username]
                [--log-level-console=log-level-console]
                [--log-level-file=log-level-file]
                [--log-filename=log-filename]
                [--error-log-filename=error-log-filename]
                [--log-directory=log-directory]
                [--log-rotation-size=log-rotation-size]
                [--log-rotation-age=log-rotation-age]
                [--remote-proto=protocol] [--remote-host=destination]
                [--remote-path=path] [--remote-user=username]
                [--remote-port=port] [--ssh-options=ssh_options]
                [--remote-libpath=libpath]
                [--help]
```

- 将备份相关设置添加到 backup.control 配置文件中，或修改已设置的值。

```
gs_probackup set-backup -B backup-path --instance=instance_name -i backup-id
[--note=text] [pinning_options] [--help]
```

- 显示位于备份目录中的 pg_probackup.conf 配置文件的内容。可以通过指定-- format=json 选项，以 json 格式显示。默认情况下，显示为纯文本格式。

```
gs_probackup show-config -B backup-path --instance=instance_name
[--format=plain|json] [--help]
```

- 显示备份目录的内容。如果指定了 instance_name 和 backup_id，则显示该备份的详细信息。可以通过指定--format=json 选项，以 json 格式显示。默认情况下，备份目录的内容显示为纯文本格式。

```
gs_probackup show -B backup-path [--instance=instance_name] [-i backup-id]
[--archive] [--format=plain|json] [--help]
```

- 创建指定实例的备份。

```
gs_probackup backup -B backup-path --instance=instance_name -b backup-mode
                [-D pgdata-path] [-C] [-S slot-name] [--temp-slot]
                [--backup-pg-log] [-j threads_num] [--progress]
                [--no-validate] [--skip-block-validation]
                [-E external-directories-paths]
```

```
                [--no-sync] [--note=text]
                [--archive-timeout=timeout]
                [--log-level-console=log-level-console]
                [--log-level-file=log-level-file]
                [--log-filename=log-filename]
                [--error-log-filename=error-log-filename]
                [--log-directory=log-directory]
                [--log-rotation-size=log-rotation-size]
                [--log-rotation-age=log-rotation-age]
                [--delete-expired] [--delete-wal] [--merge-expired]
                [--retention-redundancy=retention-redundancy]
                [--retention-window=retention-window]
                [--wal-depth=wal-depth] [--dry-run]
                [--compress-algorithm=compress-algorithm]
                [--compress-level=compress-level]
                [--compress]
                [-d dbname] [-h host] [-p port] [-U username] [-w] [-W password]
                [-t rwtimeout]
                [--remote-proto=protocol] [--remote-host=destination]
                [--remote-path=path] [--remote-user=username]
                [--remote-port=port] [--ssh-options=ssh_options]
                [--remote-libpath=libpath]
                [--ttl=interval] [--expire-time=time]
                [--help]
```

- 从备份目录 backup-path 中的备份副本恢复指定实例。如果指定了恢复目标选项，gs_probackup 将查找最近的备份并将其还原到指定的恢复目标。否则，使用最近一次备份。

```
gs_probackup restore -B backup-path --instance=instance_name
                [-D pgdata-path] [-i backup-id] [-j threads_num] [--progress]
                [--force] [--no-sync] [--no-validate]
[--skip-block-validation]
                [--external-mapping=OLDDIR=NEWDIR] [-T OLDDIR=NEWDIR]
                [--skip-external-dirs] [-I incremental_mode]
                [--recovery-target-time=time|--recovery-target-xid=xid

|--recovery-target-lsn=lsn|--recovery-target-name=target-name]
                [--recovery-target-inclusive=boolean]
                [--remote-proto=protocol] [--remote-host=destination]
                [--remote-path=path] [--remote-user=username]
```

```
                [--remote-port=port] [--ssh-options=ssh_options]
                [--remote-libpath=libpath]
                [--log-level-console=log-level-console]
                [--log-level-file=log-level-file]
                [--log-filename=log-filename]
                [--error-log-filename=error-log-filename]
                [--log-directory=log-directory]
                [--log-rotation-size=log-rotation-size]
                [--log-rotation-age=log-rotation-age]
                [--help]
```

- 将指定的增量备份与其父完全备份之间的所有增量备份合并到父完全备份。父完全备份将接收所有合并的数据，而已合并的增量备份将作为冗余被删除。

```
gs_probackup merge -B backup-path --instance=instance_name -i backup-id
                [-j threads_num] [--progress]
                [--log-level-console=log-level-console]
                [--log-level-file=log-level-file]
                [--log-filename=log-filename]
                [--error-log-filename=error-log-filename]
                [--log-directory=log-directory]
                [--log-rotation-size=log-rotation-size]
                [--log-rotation-age=log-rotation-age]
                [--help]
```

- 删除指定备份，或删除不满足当前保留策略的备份。

```
gs_probackup delete -B backup-path --instance=instance_name
                [-i backup-id | --delete-expired | --merge-expired |
--status=backup_status]
                [--delete-wal] [-j threads_num] [--progress]
                [--retention-redundancy=retention-redundancy]
                [--retention-window=retention-window]
                [--wal-depth=wal-depth] [--dry-run]
                [--log-level-console=log-level-console]
                [--log-level-file=log-level-file]
                [--log-filename=log-filename]
                [--error-log-filename=error-log-filename]
                [--log-directory=log-directory]
                [--log-rotation-size=log-rotation-size]
                [--log-rotation-age=log-rotation-age]
                [--help]
```

- 验证恢复数据库所需的所有文件是否存在且未损坏。如果未指定 instance_name，gs_probackup 将验证备份目录中的所有可用备份。如果指定 instance_name 而不指定任何附加选项，gs_probackup 将验证此备份实例的所有可用备份。如果指定了 instance_name 并且指定 backup-id 或恢复目标相关选项， gs_probackup 将检查是否可以使用这些选项恢复数据库。

```
gs_probackup validate -B backup-path
                [--instance=instance_name] [-i backup-id]
                [-j threads-num] [--progress] [--skip-block-validation]
                [--recovery-target-time=time|--recovery-target-xid=xid

|--recovery-target-lsn=lsn|--recovery-target-name=target-name]
                [--recovery-target-inclusive=boolean]
                [--log-level-console=log-level-console]
                [--log-level-file=log-level-file]
                [--log-filename=log-filename]
                [--error-log-filename=error-log-filename]
                [--log-directory=log-directory]
                [--log-rotation-size=log-rotation-size]
                [--log-rotation-age=log-rotation-age]
                [--help]
```

参数说明

通用参数

- command ：gs_probackup 除 version 和 help 以外的子命令：init、add-instance、del- instance、set-config、set-backup、show-config、show、backup、restore、merge、delete、validate。
- -?, --help ： 显示 gs_probackup 命令行参数的帮助信息，然后退出。子命令中只能使用 --help，不能使用-?。
- -V, --version ： 打印 gs_probackup 版本，然后退出。
- -B backup-path, --backup-path=backup-path ： 备份的路径。系统环境变量：$BACKUP_PATH
- -D pgdata-path, --pgdata=pgdata-path ： 数据目录的路径。系统环境变量：$PGDATA

- --instance=instance_name ： 实例名。

恢复相关参数

- -I, --incremental-mode=none|checksum|lsn ：若 PGDATA 中可用的有效页没有修改，则重新使用它们。默认值：none
- --external-mapping=OLDDIR=NEWDIR ： 在恢复时，将包含在备份中的外部目录从 OLDDIR 重新定位到 NEWDIR 目录。OLDDIR 和 NEWDIR 都必须是绝对路径。如果路径中包含“=”，则使用反斜杠转义。此选项可为多个目录多次指定。
- -T *OLDDIR=NEWDIR*, --tablespace-mapping=*OLDDIR=NEWDIR* ： 在恢复时，将表空间从 OLDDIR 重新定位到 NEWDIR 目录。OLDDIR 和 NEWDIR 必 须都是绝对路径。如果路径中包含“=”，则使用反斜杠转义。多个表空间可以多次指定此选项。此选项必须和--external-mapping 一起使用。
- --skip-external-dirs ： 跳过备份中包含的使用--external-dirs 选项指定的外部目录。这些目录的内容将不会被恢复。
- --skip-block-validation ：跳过块级校验，以加快验证速度。在恢复之前的自动验证期间，将仅做文件级别的校验。
- --no-validate ： 跳过备份验证。
- --force ： 允许忽略备份的无效状态。如果出于某种原因需要从损坏的或无效的备份中恢复数据，可以使用此标志。请谨慎使用。

恢复目标相关参数(recovery_options)

说明

当前不支持配置连续的 WAL 归档的 PITR，因而使用这些参数会有一定限制，具体如下描述。如果需要使用持续归档的 WAL 日志进行 PITR 恢复，请按照下面描述的步骤：

1. 将物理备份的文件替换目标数据库目录。
2. 删除数据库目录下pg_xlog/中的所有文件。

3. 将归档的WAL日志文件复制到pg_xlog文件中（此步骤可以省略，通过配置recovery.conf恢复命令文件中的restore_command项替代）。

4. 在数据库目录下创建恢复命令文件recovery.conf，指定数据库恢复的程度。

5. 启动数据库。

6. 连接数据库，查看是否恢复到希望预期的状态。若已经恢复到预期状态，通过pg_xlog_replay_resume()指令使主节点对外提供服务。

- --recovery-target-lsn=lsn ：指定要恢复到的 lsn，当前只能指定备份的 stop lsn。
- --recovery-target-name=target-name ：指定要将数据恢复到的已命名的保存点，保存点可以通过查看备份中 recovery- name 字段得到。
- --recovery-target-time=time ：指定要恢复到的时间，当前只能指定备份中的 recovery-time。
- --recovery-target-xid=xid ：指定要恢复到的事务 ID，当前只能指定备份中的 recovery-xid。
- --recovery-target-inclusive=boolean ：当该参数指定为 true 时，恢复目标将包括指定的内容。 当该参数指定为 false 时，恢复目标将不包括指定的内容。该参数必须和 --recovery-target-name、--recovery-target-time、--recovery- target-lsn 或--recovery-target-xid 一起使用。

留存相关参数(retention_options)

说明

可以和 backup 和 delete 命令一起使用这些参数。

- --retention-redundancy=retention-redundancy ：指定在数据目录中留存的完整备份数。必须为正整数。0 表示禁用此设置。默认值：0
- --retention-window=retention-window ：指定留存的天数。必须为正整数。0 表示禁用此设置。默认值：0

- --wal-depth=wal-depth ： 每个时间轴上必须留存的执行 PITR 能力的最新有效备份数。必须为正整数。0 表示禁用此设置。默认值：0
- --delete-wal ： 从任何现有的备份中删除不需要的 WAL 文件。
- --delete-expired ： 删除不符合 pg_probackup.conf 配置文件中定义的留存策略的备份。
- --merge-expired ： 将满足留存策略要求的最旧的增量备份与其已过期的父备份合并。
- --dry-run ： 显示所有可用备份的当前状态，不删除或合并过期备份。

固定备份相关参数(pinning_options)

说明

如果要将某些备份从已建立的留存策略中排除，可以和 backup 和 set-backup 命令一起使用这些参数。

- --ttl=interval ： 指定从恢复时间开始计算，备份要固定的时间量。必须为正整数。0 表示取消备份固定。支持的单位：ms, s, min, h, d（默认为 s）。例如：--ttl=30d。
- --expire-time=time ： 指定备份固定失效的时间戳。必须是 ISO-8601 标准的时间戳。例如：--expire-time='2020-01-01 00:00:00+03'

日志相关参数(logging_options)

日志级别：verbose、log、info、warning、error 和 off。

- --log-level-console=log-level-console ：设置要发送到控制台的日志级别。每个级别都包含其后的所有级别。级别越高，发送的消息越少。指定 off 级别表示禁用控制台日志记录。默认值：info
- --log-level-file=log-level-file ： 设置要发送到日志文件的日志级别。每个级别都包含其后的所有级别。级别越高，发送的消息越少。指定 off 级别表示禁用日志文件记录。默认值：off
- --log-filename=log-filename ：指定要创建的日志文件的文件名。文件名可以使用 strftime 模式，因此可以使用%-escapes 指定随时间变化的文件名。

例如，如果指定了“pg_probackup-%u.log”模式，则 pg_probackup 为每周的每一天生成单独的日志文件，其中%u 替换为相应的十进制数字，即 pg_probackup-1.log 表示星期一；pg_probackup-2.log 表示星期二，以此类推。

如果指定了--log-level-file 参数启用日志文件记录，则该参数有效。默认值：“pg_probackup.log”

- --error-log-filename=error-log-filename ：指定仅用于 error 日志的日志文件名。指定方式与--log-filename 参数相同。此参数用于故障排除和监视。
- --log-directory=log-directory ：指定创建日志文件的目录。必须是绝对路径。此目录会在写入第一条日志时创建。默认值：$BACKUP_PATH/log
- --log-rotation-size=log-rotation-size ：指定单个日志文件的最大大小。如果达到此值，则启动 gs_probackup 命令后，日志文件将循环，但 help 和 version 命令除外。0 表示禁用基于文件大小的循环。支持的单位：KB、MB、GB、TB（默认为 KB）。默认值：0
- --log-rotation-age=log-rotation-age ：单个日志文件的最大生命周期。如果达到此值，则启动 gs_probackup 命令后，日志文件将循环，但 help 和 version 命令除外。$BACKUP_PATH/log/log_rotation 目录下保存最后一次创建日志文件的时间。0 表示禁用基于时间的循环。支持的单位：ms, s, min, h, d（默认为 min）。默认值：0

连接相关参数(connection_options)

说明

可以和 backup 命令一起使用这些参数。

- -d *dbname*, --pgdatabase=*dbname* ：指定要连接的数据库名称。该连接仅用于管理备份进程，因此您可以连接到任何现有的数据库。如果命令行、PGDATABASE 环境变量或 pg_probackup.conf 配置文件中没有指定此参数，则 gs_probackup 会尝试从 PGUSER 环境变量中获取该值。 如果未设置 PGUSER 变量，则从当前用户名获取。系统环境变量：$PGDATABASE。
- -h *hostname*, --pghost=*hostname* ：指定运行服务器的系统的主机名。如果该值以斜杠开

头，则被用作到 Unix 域套接字的路径。系统环境变量：$PGHOST。默认值：local socket

- -p *port*, --pgport=*port* ： 指定服务器正在侦听连接的 TCP 端口或本地 Unix 域套接字文件扩展名。系统环境变量：$PGPORT。默认值：5432
- -U *username*, --pguser=*username* ：指定所连接主机的用户名。系统环境变量：$PGUSER
- -w, --no-password ： 不出现输入密码提示。如果主机要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。该选项在批量工作和不存在用户输入密码的脚本中很有帮助。
- -W password, --password=*password* ： 指定用户连接的密码。如果主机的认证策略是 trust，则不会对系统管理员进行密码验证，即无需输入-W 选项；如果没有-W 选项，并且不是系统管理员，则会提示用户输入密码。

压缩相关参数(compression_options)

说明

可以和 backup 命令一起使用这些参数。

- --compress-algorithm=*compress-algorithm* ： 指定用于压缩数据文件的算法。取值包括 zlib、pglz 和 none。如果设置为 zlib 或 pglz，此选项将启用压缩。默认情况下，压缩功能处于关闭状态。默认值：none
- --compress-level=*compress-level* ： 指定压缩级别。取值范围：0~9。0 表示无压缩。1 表示压缩比最小，处理速度最快。9 表示压缩比最大，处理速度最慢。可与 --compress-algorithm 选项一起使用。默认值：1。
- --compress ： 以--compress-algorithm=zlib 和--compress-level=1 进行压缩。

远程模式相关参数(remote_options)

说明

通过 SSH 远程运行 gs_probackup 操作的相关参数。可以和 add-instance、set-config、backup、restore 命令一起使用这些参数。

- --remote-proto=*protocol* ：指定用于远程操作的协议。目前只支持 SSH 协议。取值包括：
  - ssh：通过 SSH 启用远程备份模式。这是默认值。
  - none：显式禁用远程模式。如果指定了--remote-host 参数，可以省略此参数。
- --remote-host=*destination* ：指定要连接的远程主机的 IP 地址或主机名。
- --remote-port=*port* ：指定要连接的远程主机的端口号。默认值：22
- --remote-user=username ：指定 SSH 连接的远程主机用户。如果省略此参数，则使用当前发起 SSH 连接的用户。默认值：当前用户
- --remote-path=*path* ：指定 gs_probackup 在远程系统的安装目录。默认值：当前路径
- --remote-libpath=*libpath* ：指定 gs_probackup 在远程系统安装的 lib 库目录。
- --ssh-options=*ssh_options* ：指定 SSH 命令行参数的字符串。例如：

```
--ssh-options='-c cipher_spec -F configfile'
```

说明

如果因为网络临时故障等原因导致 server 端无应答,gs_probackup 将在等待 archive-timeout（默认 300 秒）后退出。如果备机 lsn 与主机有差别时，数据库会不停地刷以下 log 信息，此时应重新 build 备机。

```
LOG: walsender thread shut down LOG: walsender thread started
LOG: received wal replication command: IDENTIFY_VERSION LOG: received wal
replication command: IDENTIFY_MODE LOG: received wal replication command:
IDENTIFY_SYSTEM
LOG: received wal replication command: IDENTIFY_CONSISTENCE 0/D0002D8 LOG:
remote request lsn/crc: [xxxxx] local max lsn/crc: [xxxxx]
```

备份流程

步骤 1 初始化备份目录。在指定的目录下创建 backups/和 wal/子目录，分别用于存放备份文件和 WAL 文件。

```
gs_probackup init -B backup_dir
```

步骤 2 添加一个新的备份实例。gs_probackup 可以在同一个备份目录下存放多个数据

库实例的备份。

```
gs_probackup add-instance -B backup_dir -D data_dir --instance instance_name
```

步骤 3 创建指定实例的备份。在进行增量备份之前，必须至少创建一次全量备份。

```
gs_probackup backup -B backup_dir --instance instance_name -b backup_mode
```

步骤 4 从指定实例的备份中恢复数据。

```
gs_probackup restore -B backup_dir --instance instance_name -D pgdata-path -i backup_id
```

故障处理

故障问题及其原因和解决方案，参见表 5-3。

表 5-3 故障处理

问题描述	原因和解决方案

`ERROR: query failed: ERROR: canceling statement due to conflict with recovery`
(错误：查询失败：由于与恢复操作冲突，正在取消语句命令)
	原因：在备机上执行的操作正在访问存储行，主机上更改或者删除了对应的 行，并将 xlog 在备机上重放，迫使备机上操作取消。
	解决方案：
	1. 适当增加如下配置参数的值
	`max_standby_archive_delay`
	`max_standby_streaming_delay`
	2. 增加如下配置
	`hot_standby_feedback = on`

5.3 逻辑备份恢复

5.3.1 gs_dump

背景信息

gs_dump 是 GBase 8s 用于导出数据库相关信息的工具，用户可以自定义导出一个数据库或其中的对象（模式、表、视图等），回收站对象除外。支持导出的数据库可以是默认数

据库 postgres，也可以是自定义数据库。

gs_dump 工具由管理员用户 gbase 执行。gs_dump 工具在进行数据导出时，其他用户可以访问 GBase 8s 数据库（读或写）。

gs_dump 工具支持导出完整一致的数据。例如，T1 时刻启动 gs_dump 导出 A 数据库，那么导出数据结果将会是 T1 时刻 A 数据库的数据状态，T1 时刻之后对 A 数据库的修改不会被导出。

gs_dump 时生成列不会被转储。

gs_dump 支持导出兼容 v1 版本数据库的文本格式文件。gs_dump 支持将数据库信息导出至纯文本格式的 SQL 脚本文件或其他归档文件中。

- 纯文本格式的 SQL 脚本文件：包含将数据库恢复为其保存时的状态所需的 SQL 语 句。通过 gsql 运行该 SQL 脚本文件，可以恢复数据库。即使在其他主机和其他数据库产品上，只要对 SQL 脚本文件稍作修改，也可以用来重建数据库。
- 归档格式文件：包含将数据库恢复为其保存时的状态所需的数据，可以是 tar 格式、目录归档格式或自定义归档格式，详见表 6-4。该导出结果必须与 gs_restore 配合使用来恢复数据库，gs_restore 工具在导入时，系统允许用户选择需要导入的内容，甚至可以在导入之前对等待导入的内容进行排序。

主要功能

gs_dump 可以创建四种不同的导出文件格式，通过[-F 或者--format=]选项指定，具体如下表所示。

表 5-4 导出文件格式

格式名称	-F 的参数值	说明	建议	对应导入工具
纯文本格式	p	纯文本脚本文件包含 SQL 语句和命令。命令可	小型数据库，一般推荐纯文本	使用 gsql 工具恢复数据库对象前，可根据需要使用

		以由 gsql 命令行终端程序执行，用于重新创建数据库对象并加载表数据。	格式。	文本编辑器编辑纯文本导出文件。
自定义归档格式	c	一种二进制文件。支持从导出文件中恢复所有或所选数据库对象。	中型或大型数据库，推荐自定义归档格式。	使用 gs_restore 可以选择要从自定义归档导出文件中导入相应的数据库对象。
目录归档格式	d	该格式会创建一个目录，该目录包含两类文件，一类是目录文件，另一类是每个表和 blob 对象对应的数据文件。	——	
tar 归档格式	t	tar 归档文件支持从导出文件中恢复所有或所选数据库对象。tar 归档格式不支持压缩且对于单独表大小应小于 8GB。	——	

说明

可以使用 gs_dump 程序将文件压缩为目录归档或自定义归档导出文件，减少导出文件的大小。生成目录归档或自定义归档导出文件时，默认进行中等级别的压缩。gs_dump 程序无法压缩已归档导出文件。

注意事项

- 禁止修改导出的文件和内容，否则可能无法恢复成功。
- 为了保证数据一致性和完整性，gs_dump 会对需要转储的表设置共享锁。如果表在别的事务中设置了共享锁，gs_dump 会等待锁释放后锁定表。如果无法在指定时间内锁定某个表，转储会失败。用户可以通过指定--lock-wait-timeout 选项，自定义等待锁超时时间。
- 不支持加密导出存储过程和函数。

语法

```
gs_dump  [OPTION]... [DBNAME]
```

说明

“DBNAME”前面不需要加短或长选项。“DBNAME”指定要连接的数据库。例如：不需要-d，直接指定“DBNAME”。

```
gs_dump -p port_number postgres -f dump1.sql
```

或者

```
export PGDATABASE=postgres
gs_dump -p port_number -f dump1.sql
```

环境变量：PGDATABASE

参数说明

通用参数

- -f, --file=FILENAME ：将输出发送至指定文件或目录。如果省略该参数，则使用标准输出。如果输出格式为(-F c/-F d/-F t)时，必须指定-f 参数。如果-f 的参数值含有目录，要求当前用户对该目录具有读写权限，并且不能指定已有目录。
- -F, --format=c|d|t|p ：选择输出格式。格式如下：
  - p|plain：输出一个文本 SQL 脚本文件（默认）。
  - c|custom：输出一个自定义格式的归档，并且以目录形式输出，作为 gs_restore 输入信息。该格式是最灵活的输出格式，因为能手动选择，而且能在恢复过程中将归档项重新排序。该格式默认状态下会被压缩。
  - d|directory：该格式会创建一个目录，该目录包含两类文件，一类是目录文件，另一类是每个表和 blob 对象对应的数据文件。
  - t|tar：输出一个 tar 格式的归档形式，作为 gs_restore 输入信息。tar 格式与目 录格式兼容；tar 格式归档形式在提取过程中会生成一个有效的目录格式归档形式。但是，tar 格式不支持压缩且对于单独表有 8GB 的大小限制。此外，表数据项的相应

排序在恢复过程中不能更改。

输出一个 tar 格式的归档形式，也可以作为 gsql 输入信息。

- -v, --verbose ：指定 verbose 模式。该选项将导致 gs_dump 向转储文件输出详细的对象注解和启动/停止次数，向标准错误流输出处理信息。
- -V, --version ：打印 gs_dump 版本，然后退出。
- -Z, --compress=0-9 ：指定使用的压缩比级别。取值范围：0~9。0 表示无压缩。1 表示压缩比最小，处理速度最快。9 表示压缩比最大，处理速度最慢。

  针对自定义归档格式，该选项指定单个表数据片段的压缩，默认方式是以中等级别进行压缩。tar 归档格式和纯文本格式目前不支持压缩。
- --lock-wait-timeout=TIMEOUT ：请勿在转储刚开始时一直等待以获取共享表锁。如果无法在指定时间内锁定某个表，就选择失败。可以以任何符合 SET statement_timeout 的格式指定超时时间。
- -?, --help ：显示 gs_dump 命令行参数帮助，然后退出。转储参数：
- -a, --data-only ：只输出数据，不输出模式(数据定义)。转储表数据、大对象和序列值。
- -b, --blobs ：该参数为扩展预留接口，不建议使用。
- -c, --clean ：在将创建数据库对象的指令输出到备份文件之前，先将清理（删除）数据库对象的指令输出到备份文件中。如果目标数据库中没有任何对象，gs_restore 工具可能会输出一些提示性的错误信息。该选项只对文本格式有意义。针对归档格式，可以调用 gs_restore 时指定选项。
- -C, --create ：备份文件以创建数据库和连接到创建的数据库的命令开始。（如果命令脚本是这种方式执行，可以先指定任意数据库用于执行创建数据库的命令，数据不会恢复到指定的数据库中，而是恢复到创建的数据库中。

  该选项只对文本格式有意义。针对归档格式，可以在调用 gs_restore 时指定选项。
- -E, --encoding=ENCODING ：以指定的字符集编码创建转储。默认情况下，以数据库

编码创建转储。得到相同结果的另一个办法是将环境变量“PGCLIENTENCODING”设置为所需的转储编码。

- -n, --schema=SCHEMA ：只转储与模式名称匹配的模式，此选项包括模式本身和所有它包含的对象。如果该选项没有指定，所有在目标数据库中的非系统模式将会被转储。写入多个-n 选项来选择多个模式。此外，根据 gsql 的\d 命令所使用的相同规则，模式参数可被 理解成一个 pattern，所以多个模式也可以通过在该 pattern 中写入通配符来选择。使用通配符时，注意给 pattern 打引号，防止 shell 扩展通配符。

  说明

  - 当-n 已指定时，gs_dump 不会转储已选模式所附着的任何其他数据库对象。因此，无法保证某个指定模式的转储结果能够自行成功地储存到一个空数据库中。
  - 当-n 指定时，非模式对象不会被转储。
  - 转储支持多个模式的转储。多次输入-n schemaname 转储多个模式。例如：

```
gs_dump -h 10.0.7.16 -p 5432 postgres -f /home/gbase/data/backup/bkp_sh12.sql
-n sch1 -n sch2
```

在上面这个例子中，sch1 和 sch2 会被转储。

- -N, --exclude-schema=SCHEMA ：不转储任何与模式 pattern 匹配的模式。pattern 将参照针对-n 的相同规则来理解。可以通过输入多次-N，不转储与任何 pattern 匹配的模式。当同时输入-n 和-N 时，会转储与至少一个-n 选项匹配、与-N 选项不匹配的模式。如果有-N 没有-n，则不转储常规转储中与-N 匹配的模式。

  转储过程支持排除多个模式。

  在转储过程中，输入-N exclude schema name 排除多个模式。例如：

```
gs_dump -h 10.0.7.16 -p 5432 postgres -f /home/gbase/data/backup/bkp_sh12.sql
-N sch1 -N sch2
```

在上面这个例子中，sch1 和 sch2 在转储过程中会被排除。

- -o, --oids ：转储每个表的对象标识符（OIDs），作为表的一部分数据。该选项用于应用以某种方式参照了 OID 列的情况。如果不是以上这种情况，请勿使用该选项。

- -O, --no-owner ：不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_dump 会发出 ALTER OWNER 或 SET SESSION AUTHORIZATION 语句设置所创建的数据库对象的归属。如果脚本正在运行，该语句不会执行成功，除非是由系统管理员触发（或是拥有脚本中所有对象的同一个用户）。通过指定-O，编写一个任何用户都能存储的脚本，且该脚本会授予该用户拥有所有对象的权限。

  该选项只对文本格式有意义。针对归档格式，可以在调用 gs_restore 时指定选项。

- -s, --schema-only ： 只转储对象定义（模式），而非数据。

- -S, --sysadmin=NAME ： 该参数为扩展预留接口，不建议使用。

- -t, --table=TABLE ： 指定转储的表（或视图、或序列、或外表）对象列表，可以使用多个-t 选项来选择多个表，也可以使用通配符指定多个表对象。

  当使用通配符指定多个表对象时，注意给 pattern 打引号，防止 shell 扩展通配符。

  当使用-t 时，-n 和-N 没有任何效应，这是因为由-t 选择的表的转储不受那些选项的影响。

  说明

  - -t 参数选项个数必须小于等于 100。
  - 如果-t 参数选项个数大于 100，建议使用参数--include-table-file 来替换。
  - 当-t 已指定时，gs_dump 不会转储已选表所附着的任何其他数据库对象。因此，无法保证某个指定表的转储结果能够自行成功地储存到一个空数据库中。
  - -t tablename 只转储在默认搜索路径中可见的表。-t '*.tablename'转储数据库下所有模式下的 tablename 表。-t schema.table 转储特定模式中的表。
  - -t tablename 不会导出表上的触发器信息。

  例如：

```
gs_dump -h 10.0.7.16 -p 5432 postgres -f backup/bkp_sh12.sql -t
schema1.table1 -t schema2.table2
```

  在上面这个例子中，schema1.table1 和 schema2.table2 会被转储。

- --include-table-file=FILENAME ： 指定需要 dump 的表文件。
- -T, --exclude-table=TABLE ： 不转储的表（或视图、或序列、或外表）对象列表，可以使用多个-T 选项来选择多个表，也可以使用通配符指定多个表对象。

  当同时输入-t 和-T 时，会转储在-t 列表中，而不在-T 列表中的表对象。例如：

```
gs_dump -h 10.0.7.16 -p 5432 postgres -f backup/bkp_sh12.sql -T table1 -T table2
```

  在上面这个例子中，table1 和 table2 在转储过程中会被排除。

- --exclude-table-file=FILENAME ： 指定不需要 dump 的表文件。

  说明

  同--include-table-file，其内容格式如下：

```
schema1.table1 schema2.table2......
```

- -x, --no-privileges|--no-acl ： 防止转储访问权限（授权/撤销命令）。
- -q, --target ： 指定导出兼容其他版本数据库的文本文件，目前支持 v1 和 v5 参数。v1 参数用于导出 v5 数据库的数据为兼容 v1 的文本文件。v5 参数用于导出 v5 数据库的数据为 v5 格式的文本文件，减少了导入 v5 时的可能的报错情况。

  在使用 v1 参数时，建议和--exclude-guc="enable_cluster_resize"， --exclude- function，--exclude-with 等选项共用，否则导入到 v1 时可能报错。

- --exclude-guc ： 导出的文本文件中，不包括相关 guc 参数的 set 命令，目前只支持 enable_cluster_resize。
- --exclude-function ： 不导出函数和存储过程。
- --exclude-with ： 导出的表定义，末尾不添加 WITH(orientation=row，compression=on）这样的描述。
- --binary-upgrade ： 该参数为扩展预留接口，不建议使用。
- --binary-upgrade-usermap="USER1=USER2" ： 该参数为扩展预留接口，不建议使用。
- --column-inserts|--attribute-inserts ： 以 INSERT 命令带列名（INSERT INTO 表（列、…）

值…）方式导出数据。这会导致恢复缓慢。但是由于该选项会针对每行生成一个独立分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。

- --disable-dollar-quoting ： 该选项将禁止在函数体前使用美元符号$，并强制使用 SQL 标准字符串语法对其进行引用。
- --disable-triggers ： 该参数为扩展预留接口，不建议使用。
- --exclude-table-data=TABLE ： 指定不转储任何匹配表 pattern 的表这方面的数据。依照针对-t 的相同规则理解该 pattern。

  可多次输入--exclude-table-data 来排除匹配任何 pattern 的表。当用户需要特定表的定义但不需要其中的数据时，这个选项很有帮助。排除数据库中所有表的数据，参见 --schema-only。
- --inserts ： 发出 INSERT 命令（而非 COPY 命令）转储数据。

  这会导致恢复缓慢。但是由于该选项会针对每行生成一个独立分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。注意：如果重排列顺序，可能会导致整个恢复失败。列顺序改变时，--column-inserts 选项不受影响，虽然会更慢。
- --no-publications ： 不转储发布。
- --no-security-labels ： 该参数为扩展预留接口，不建议使用。
- --no-subscriptions ： 不转储订阅。
- --no-tablespaces ： 不输出选择表空间的命令。

  使用该选项，无论默认表空间是哪个，在恢复过程中所有对象都会被创建。该选项只对文本格式有意义。针对归档格式，可以在调用 gs_restore 时指定选项。
- --no-unlogged-table-data ： 该参数为扩展预留接口，不建议使用。
- --non-lock-table ： 该参数为扩展预留接口，不建议使用。
- --include-alter-table ： 转储表删除列。该选项会记录列的删除。

- --quote-all-identifiers ：强制对所有标识符加引号。为了向后续版本迁移，且其中可能涉及引入额外关键词，在转储相应数据库时该选项会有帮助。
- --section=SECTION ：指定已转储的名称区段（pre-data、data 和 post-data）。
- --serializable-deferrable ：转储过程中使用可串行化事务，以确保所使用的快照与之后的数据库状态一致；要实现该操作需要在无异常状况的事务流中等待某个点，因为这样才能保证转储成功，避免引起其他事务出现 serialization_failure 要重新再做。

  但是该选项对于灾难恢复没有益处。对于在原始数据库进行升级的时候，加载一个数据库的拷贝作为报告或其他只读加载共享的转储是有帮助的。没有这个选项，转储会反映一个与任何事务最终提交的序列化执行不一致的状态。

  如果当 gs_dump 启动时，读写事务仍处于非活动状态，即便使用该选项也不会对其产生影响。如果读写事务处于活动状态，转储的开始时间可能会延迟一段不确定的时间。
- --use-set-session-authorization ：输出符合 SQL 标准的 SET SESSION AUTHORIZATION 命令而不是 ALTER OWNER 命令来确定对象所有权。这样令转储更加符合标准，但是如果转储文件中的对象的历史有些问题，那么可能不能正确恢复。并且，使用 SET SESSION AUTHORIZATION 的转储需要数据库系统管理员的权限才能转储成功，而 ALTER OWNER 需要的权限则低得多。
- --with-encryption=AES128 ：指定转储数据需用 AES128 进行加密。
- --with-key=KEY AES128 密钥规则如下：
  - 密钥长度为 8~16 个字符。
  - 至少包含大写字母（A-Z）、小写字母（a-z）、数字（0-9）、非字母数字字符（限定为~!@#$%^&*()-_=+\|[{}];:,<.>/?）四类字符中的三类字符。

  说明

  - 使用 gs_dump 工具进行加密导出时，仅支持 plain 格式导出。通过-F plain 导出的数据，需要通过 gsql 工具进行导入，且如果以加密方式导入，在通过 gsql 导入时，需要指定-- with-key 参数。

- 不支持加密导出存储过程和函数。

- --with-salt=RANDVALUES gs_dumpall ： 使用此参数传递随机值。
- --include-extensions ： 在转储中包含扩展。
- --include-depend-objs ： 备份结果包含依赖于指定对象的对象信息。该参数需要同-t/--include-table-file 参数关联使用才会生效。
- --exclude-self ：备份结果不包含指定对象自身的信息。该参数需要同-t/--include-table-file 参数关联使用才会生效。
- --pipeline ： 使用管道传输密码，禁止在终端使用。
- --dont-overwrite-file ： 文本、tar 以及自定义格式情况下会重写现有文件。这对目录格式不适用。例如：

  设想这样一种情景，即当前目录下 backup.sql 已存在。如果在输入命令中输入-f backup.sql 选项时，当前目录恰好也生成 backup.sql，文件就会被重写。

  如果备份文件已存在，且输入--dont-overwrite-file 选项，则会报告附带转储文件已经存在信息的错误。

```
gs_dump -p 5432 postgres -f backup.sql -F plain --dont-overwrite-file
```

说明

- -s/--schema-only 和-a/--data-only 不能同时使用。
- -c/--clean 和-a/--data-only 不能同时使用。
- --inserts/--column-inserts 和-o/--oids 不能同时使用，因为 INSERT 命令不能设置 OIDS。
- --role 和--rolepassword 必须一起使用。
- --binary-upgrade-usermap 和--binary-upgrade 必须一起使用。
- --include-depend-objs/--exclude-self 需要同-t/--include-table-file 参数关联使用才会生效

- --exclude-self 必须同--include-depend-objs 一起使用。

连接参数

- -h, --host=HOSTNAME ： 指定主机名称。如果数值以斜杠开头，则被用作到 Unix 域套接字的路径。缺省从 PGHOST 环境变量中获取（如果已设置），否则，尝试一个 Unix 域套接字连接。该参数只针对 GBase 8s 外，对内本机只能用 127.0.0.1。
- -p, --port=PORT ： 指定主机端口。在开启线程池情况下，建议使用 pooler port，即主机端口+1。环境变量：PGPORT
- -U, --username=NAME ： 指定所连接主机的用户名。不指定连接主机的用户名时，用户默认系统管理员。环境变量：PGUSER
- -w, --no-password ： 不出现输入密码提示。如果主机要求密码认证并且密码没有通过其它形式给出， 则连接尝试将会失败。 该选项在批量工作和不存在用户输入密码的脚本中很有帮助。
- -W, --password=PASSWORD ： 指定用户连接的密码。如果主机的认证策略是 trust，则不会对系统管理员进行密码验证，即无需输入-W 选项；如果没有-W 选项，并且不是系统管理员，“Dump Restore 工具”会提示用户输入密码。
- --role=ROLENAME ： 指定创建转储使用的角色名。选择该选项，会使 gs_dump 连接数据库后，发起一个 SET ROLE 角色名命令。当所授权用户（由-U 指定）没有 gs_dump 要求的权限时，该选项会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允许直接以超系统管理员身份登录，而使用该选项能够在不违反该规定的情况下完成转储。
- --rolepassword=ROLEPASSWORD ： 指定角色名的密码。

说明

如果 GBase 8s 有任何本地数据要添加到 template1 数据库，请谨慎将 gs_dump 的输出恢复到一个真正的空数据库中，否则可能会因为被添加对象的定义被复制，出现错误。要创建一个无本地添加的空数据库，需从 template0 而非 template1 复制，例如：

```
postgres=# CREATE DATABASE foo WITH TEMPLATE template0;
```

tar 归档形式的文件大小不得超过 8GB（tar 文件格式的固有限制）。tar 文档整体大小和任何其他输出格式没有限制，操作系统可能对此有要求。

由 gs_dump 生成的转储文件不包含优化程序用来做执行计划决定的统计数据。因此，最好从某转储文件恢复之后运行 ANALYZE 以确保最佳效果。转储文件不包含任何 ALTER DATABASE…SET 命令，这些设置由 gs_dumpall 转储，还有数据库用户和其他完成安装设置。

示例

使用 gs_dump 转储数据库为 SQL 文本文件或其它格式的操作，如下所示。导出操作时，请确保指定目录存在，并且当前的操作系统用户对其具有读写权限。

示例 1

执行 gs_dump，导出 postgres 数据库全量信息，导出的 MPPDB_backup.sql 文件格式为纯文本格式。

```
gs_dump postgres -U gbase -W gbase,123 -f
/home/gbase/data/backup/MPPDB_backup.sql -p 5432 -F p
gs_dump[port='5432'][postgres][2018-06-27 09:49:17]: The total objects number
is 356. gs_dump[port='5432'][postgres][2018-06-27 09:49:17]: [100.00%] 356
objects have been dumped. gs_dump[port='5432'][postgres][2018-06-27 09:49:17]:
dump database postgres successfully gs_dump[port='5432'][postgres][2018-06-27
09:49:17]: total time: 1274 ms
```

使用 gsql 程序从纯文本导出文件中导入数据。

示例 2

执行 gs_dump，导出 postgres 数据库全量信息，导出的 MPPDB_backup.tar 文件格式为 tar 格式。

```
gs_dump postgres -U gbase -W gbase,123 -f
/home/gbase/data/backup/MPPDB_backup.tar -p 5432 -F t
gs_dump[port='5432'][postgres][2018-06-27 10:02:24]: The total objects number
is 1369. gs_dump[port='5432'][postgres][2018-06-27 10:02:53]: [100.00%] 1369
objects have been dumped. gs_dump[port='5432'][postgres][2018-06-27 10:02:53]:
```

```
dump database postgres successfully gs_dump[port='5432'][postgres][2018-06-27
10:02:53]: total time: 50086 ms
```

示例 3

执行 gs_dump，导出 postgres 数据库全量信息，导出的 MPPDB_backup.dmp 文件格式为自定义归档格式。

```
gs_dump postgres -U gbase -W gbase,123 -f
/home/gbase/data/backup/MPPDB_backup.dmp -p 5432 -F c
gs_dump[port='5432'][postgres][2018-06-27 10:05:40]: The total objects number
is 1369. gs_dump[port='5432'][postgres][2018-06-27 10:06:03]: [100.00%] 1369
objects have been dumped. gs_dump[port='5432'][postgres][2018-06-27 10:06:03]:
dump database postgres successfully gs_dump[port='5432'][postgres][2018-06-27
10:06:03]: total time: 36620 ms
```

示例 4

执行 gs_dump，导出 postgres 数据库全量信息，导出的 MPPDB_backup 文件格式为目录格式。

```
gs_dump postgres -U gbase -W gbase,123 -f /home/gbase/data/backup/MPPDB_backup
-p 5432 -F d
gs_dump[port='5432'][postgres][2018-06-27 10:16:04]: The total objects number
is 1369. gs_dump[port='5432'][postgres][2018-06-27 10:16:23]: [100.00%] 1369
objects have been dumped. gs_dump[port='5432'][postgres][2018-06-27 10:16:23]:
dump database postgres successfully gs_dump[port='5432'][postgres][2018-06-27
10:16:23]: total time: 33977 ms
```

示例 5

执行 gs_dump，导出 postgres 数据库信息，但不导出/home/MPPDB_temp.sql 中指定的表信息。导出的 MPPDB_backup.sql 文件格式为纯文本格式。

```
gs_dump postgres -U gbase -W gbase,123
--exclude-table-file=/home/gbase/data/MPPDB_temp.sql -f
/home/gbase/data/backup/MPPDB_backup.sql -p 5432
gs_dump[port='5432'][postgres][2018-06-27 10:37:01]: The total objects number
is 1367. gs_dump[port='5432'][postgres][2018-06-27 10:37:22]: [100.00%] 1367
objects have been dumped. gs_dump[port='5432'][postgres][2018-06-27 10:37:22]:
dump database postgres successfully gs_dump[port='5432'][postgres][2018-06-27
10:37:22]: total time: 37017 ms
```

示例 6

执行 gs_dump，仅导出依赖于指定表 testtable 的视图信息。然后创建新的 testtable 表，再恢复依赖其上的视图。

- 备份仅依赖于 testtable 的视图。

```
gs_dump -s -p 5432 postgres -t PUBLIC.testtable --include-depend-objs
--exclude-self -f backup/ MPPDB_backup.sql -F p
gs_dump[port='5432'][postgres][2018-06-15 14:12:54]: The total objects number
is 331. gs_dump[port='5432'][postgres][2018-06-15 14:12:54]: [100.00%] 331
objects have been dumped. gs_dump[port='5432'][postgres][2018-06-15 14:12:54]:
dump database postgres successfully gs_dump[port='5432'][postgres][2018-06-15
14:12:54]: total time: 327 ms
```

- 修改 testtable 名称。

```
gsql -p 5432 postgres -r -c "ALTER TABLE PUBLIC.testtable RENAME TO
testtable_bak;"
```

- 创建新的 testtable 表。

```
postgres=# CREATE TABLE PUBLIC.testtable(a int, b int, c int);
```

- 还原依赖于 testtable 的视图。

```
gsql -p 5432 postgres -r -f backup/MPPDB_backup.sql
```

5.3.2 gs_dumpall

背景信息

gs_dumpall 是 GBase 8s 用于导出所有数据库相关信息工具，它可以导出 GBase 8s 数据库的所有数据，包括默认数据库 postgres 的数据、自定义数据库的数据、以及 GBase 8s 所有数据库公共的全局对象。

gs_dumpall 工具由管理员用户 gbase 执行。gs_dumpall 工具在进行数据导出时，其他用户可以访问 GBase 8s 数据库（读或写）。

gs_dumpall 工具支持导出完整一致的数据。例如，T1 时刻启动 gs_dumpall 导出 GBase 8s 数据库，那么导出数据结果将会是 T1 时刻该 GBase 8s 数据库的数据状态，T1 时刻之后对 GBase 8s 的修改不会被导出。

gs_dumpall 在导出 GBase 8s 所有数据库时分为两部分：

- gs_dumpall 自身对所有数据库公共的全局对象进行导出，包括有关数据库用户和组，表空间以及属性（例如，适用于数据库整体的访问权限）信息。
- gs_dumpall 通过调用 gs_dump 来完成 GBase 8s 中各数据库的 SQL 脚本文件导出，该脚本文件包含将数据库恢复为其保存时的状态所需要的全部 SQL 语句。

以上两部分导出的结果为纯文本格式的 SQL 脚本文件，使用 gsql 运行该脚本文件可以恢复 GBase 8s 数据库。

注意事项

- 禁止修改导出的文件和内容，否则可能无法恢复成功。
- 为了保证数据一致性和完整性，gs_dumpall 会对需要转储的表设置共享锁。如果某张表在别的事务中设置了共享锁，gs_dumpall 会等待此表的锁释放后锁定此 表。如果无法在指定时间内锁定某张表，转储会失败。用户可以通过指定--lock- wait-timeout 选项，自定义等待锁超时时间。
- 由于 gs_dumpall 读取所有数据库中的表，因此必须以 GBase 8s 管理员身份进行连接，才能导出完整文件。在使用 gsql 执行脚本文件导入时，同样需要管理员权限，以便添加用户和组以及创建数据库。

语法

```
gs_dumpall [OPTION]...
```

参数说明

通用参数：

- -f, --filename=FILENAME ：将输出发送至指定文件。如果这里省略，则使用标准输出。
- -v, --verbose ：指定 verbose 模式。该选项将导致 gs_dumpall 向转储文件输出详细的对象注解和启动/停止次数，向标准错误流输出处理信息。

- -V, --version ： 打印 gs_dumpall 版本，然后退出。
- --lock-wait-timeout=TIMEOUT ： 请勿在转储刚开始时一直等待以获取共享表锁。如果无法在指定时间内锁定某个表，就选择失败。可以以任何符合 SET statement_timeout 的格式指定超时时间。
- -?, --help ： 显示 gs_dumpall 命令行参数帮助，然后退出。

转储参数：

- -a, --data-only ： 只转储数据，不转储模式（数据定义）。
- -c, --clean ： 在重新创建数据库之前，执行 SQL 语句清理（删除）这些数据库。针对角色和表空间的转储命令已添加。
- -g, --globals-only ： 只转储全局对象（角色和表空间），无数据库。
- -o, --oids ： 转储每个表的对象标识符（OIDs），作为表的一部分数据。该选项用于应用以某种方式参照了 OID 列的情况。如果不是以上这种情况，请勿使用该选项。
- -O, --no-owner ：不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_dumpall 会发出 ALTER OWNER 或 SET SESSION AUTHORIZATION 语句设置所创建的模式元素的所属。如果脚本正在运行，该语句不会执行成功，除非是由系统管理员触发（或是拥有脚本中所有对象的同一个用户）。通过指定-O，编写一个任何用户都能存储的脚本，且该脚本会授予该用户拥有所有对象的权限。
- -r, --roles-only ： 只转储角色，不转储数据库或表空间。
- -s, --schema-only ： 只转储对象定义（模式），而非数据。
- -S, --sysadmin=NAME ： 在转储过程中使用的系统管理员名称。
- -t, --tablespaces-only ： 只转储表空间，不转储数据库或角色。
- -x, --no-privileges ： 防止转储访问权限（授权/撤销命令）。
- --column-inserts|--attribute-inserts ： 以 INSERT 命令带列名（INSERT INTO 表（列、…）值…）方式导出数据。这会导致恢复缓慢。但是由于该选项会针对每行生成一个独立

分开的命令，所以在重新加载某行时出现的错误只会导致那行丢失，而非整个表内容。

- --disable-dollar-quoting ：该选项将禁止在函数体前使用美元符号$，并强制使用 SQL 标准字符串语法对其进行引用。
- --disable-triggers ： 该参数为扩展预留接口，不建议使用。
- --inserts ：发出 INSERT 命令（而非 COPY 命令）转储数据。这会导致恢复缓慢。注意：如果 重排列顺序，可能会导致恢复整个失败。--column-inserts 选项更加安全，虽然可能更慢些。
- --no-publications ： 不转储发布。
- --no-security-labels ： 该参数为扩展预留接口，不建议使用。
- --no-subscriptions ： 不转储订阅。
- --no-tablespaces ：请勿输出创建表空间的命令，也请勿针对对象选择表空间。使用该选项，无论默认表空间是哪个，在恢复过程中所有对象都会被创建。
- --no-unlogged-table-data ： 该参数为扩展预留接口，不建议使用。
- --include-alter-table ： 导出表中已删除的列信息。
- --quote-all-identifiers ： 强制对所有标识符加引号。为了向后续版本迁移，且其中可能涉及引入额外关键词，在转储相应数据库时该选项会有帮助。
- --dont-overwrite-file ： 不重写当前文件。
- --use-set-session-authorization ：输出符合 SQL 标准的 SET SESSION AUTHORIZATION 命令而不是 ALTER OWNER 命令来确定对象所有权。这样令转储更加符合标准，但是如果转储文件中的对象的历史有些问题，那么可能不能正确恢复。并且，使用 SET SESSION AUTHORIZATION 的转储需要数据库系统管理员的权限才能转储成功，而 ALTER OWNER 需要的权限则低得多。
- --with-encryption=AES128 ： 指定转储数据需用 AES128 进行加密。
- --with-key=KEY　AES128 密钥规则如下：

- 密钥长度为 8~16 个字符。
- 至少包含大写字母（A-Z），小写字母（a-z），数字（0-9），非字母数字字符（限定为~!@#$%^&*()-_=+\|[{}];:,<.>/?）四类字符中的三类字符。

- --include-extensions ： 如果 include-extensions 参数被设置，将备份所有的 CREATE EXTENSION 语句。
- --include-templatedb ： 转储过程中包含模板库。
- --binary-upgrade ： 该参数为扩展预留接口，不建议使用。
- --binary-upgrade-usermap="USER1=USER2" ： 该参数为扩展预留接口，不建议使用。
- --non-lock-table ： 该参数仅供 OM 工具使用。
- --tablespaces-postfix ： 该参数为扩展预留接口，不建议使用。
- --parallel-jobs ： 指定备份进程并发数，取值范围为 1~1000。
- --pipeline ： 使用管道传输密码，禁止在终端使用。

说明

- -g/--globals-only 和-r/--roles-only 不能同时使用。
- -g/--globals-only 和-t/--tablespaces-only 不能同时使用。
- -r/--roles-only 和-t/--tablespaces-only 不能同时使用。
- -s/--schema-only 和-a/--data-only 不能同时使用。
- -r/--roles-only 和-a/--data-only 不能同时使用。
- -t/--tablespaces-only 和-a/--data-only 不能同时使用。
- -g/--globals-only 和-a/--data-only 不能同时使用。
- --tablespaces-postfix 和--binary-upgrade 必须一起使用。
- --binary-upgrade-usermap 和--binary-upgrade 必须一起使用。
- --parallel-jobs 和-f/--file 必须一起使用。

连接参数：

- -h, --host ：指定主机的名称。如果取值是以斜线开头，它将用作 Unix 域套接字的目录。默认值取自 PGHOST 环境变量；如果没有设置，将启动某个 Unix 域套接字建立连接。该参数只针对 GBase 8s 外，对 GBase 8s 内本机只能用 127.0.0.1。环境变量：PGHOST
- -l, --database ：指定所连接的转储全局对象的数据库名称，并去寻找还有其他哪些数据库需要被转储。如果没有指定，会使用 postgres 数据库，如果 postgres 数据库不存在，会使用 template1。
- -p, --port ：指定服务器所侦听的 TCP 端口或本地 Unix 域套接字后缀，以确保连接。默认值设置为 PGPORT 环境变量。在开启线程池情况下，建议使用 pooler port，即侦听端口+1。环境变量：PGPORT
- -U, --username ：所连接的用户名。环境变量：PGUSER
- -w, --no-password ：不出现输入密码提示。如果服务器要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。 该选项在批量工作和不存在用户输入密码的脚本中很有帮助。
- -W, --password ：指定用户连接的密码。如果主机的认证策略是 trust，则不会对系统管理员进行密码验证，即无需输入-W 选项；如果没有-W 选项，并且不是系统管理员，“Dump Restore 工具”会提示用户输入密码。
- --role ：指定创建转储使用的角色名。选择该选项，会使 gs_dumpall 连接数据库后，发起一个 SET ROLE 角色名命令。当所授权用户（由-U 指定）没有 gs_dumpall 要求的权限时，该选项会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允许直接以系统管理员身份登录，而使用该选项能够在不违反该规定的情况下完成转储。
- --rolepassword ：指定具体角色用户的角色密码。

说明

- 由于 gs_dumpall 内部调用 gs_dump，所以一些诊断信息参见 gs_dump。一旦恢复，最好在每个数据库上运行 ANALYZE，优化程序提供有用的统计数据。

- gs_dumpall 恢复前需要所有必要的表空间目录才能退出；否则，对于处在非默认位置的数据库，数据库创建会失败。

示例

使用 gs_dumpall 一次导出 GBase 8s 的所有数据库。

说明

gs_dumpall 仅支持纯文本格式导出。只能使用 gsql 恢复 gs_dumpall 导出的转储内容。

```
gs_dumpall -f backup/bkp2.sql -p 5432
gs_dump[port='5432'][dbname='postgres'][2018-06-27 09:55:09]: The total
objects number is 2371. gs_dump[port='5432'][dbname='postgres'][2018-06-27
09:55:35]: [100.00%] 2371 objects have been dumped.
gs_dump[port='5432'][dbname='postgres'][2018-06-27 09:55:46]: dump database
dbname='postgres' successfully
gs_dump[port='5432'][dbname='postgres'][2018-06-27 09:55:46]: total time:
55567 ms gs_dumpall[port='5432'][2018-06-27 09:55:46]: dumpall operation
successful gs_dumpall[port='5432'][2018-06-27 09:55:46]: total time: 56088 ms
```

5.3.3 gs_restore

背景信息

gs_restore 是 GBase 8s 提供的针对 gs_dump 导出数据的导入工具。通过此工具可由 gs_dump 生成的导出文件进行导入。

gs_restore 工具由管理员用户 gbase 执行。主要功能包含：

- 导入到数据库

如果连接参数中指定了数据库，则数据将被导入到指定的数据库中。其中，并行导入必须指定连接的密码。导入时生成列会自动更新，并像普通列一样保存。

- 导入到脚本文件

如果未指定导入数据库，则创建包含重建数据库所必须的 SQL 语句脚本并写入到文件或者标准输出。等效于直接使用 gs_dump 导出为纯文本格式。

语法

```
gs_restore [OPTION]... FILE
```

说明

- FILE 没有短选项或长选项。用来指定归档文件所处的位置。
- 作为前提条件，需输入 dbname 或-l 选项。不允许用户同时输入 dbname 和-l 选项。
- gs_restore 默认是以追加的方式进行数据导入。为避免多次导入造成数据异常，在进行导入时，建议使用“-c”参数，在重新创建数据库对象前，清理（删除）已存在于将要还原的数据库中的数据库对象。
- 日志打印无开关，若需隐藏日志，请将日志重定向到日志文件。若恢复表数据时，数据量很大，会分批恢复，因此会多次出现“表数据已完成导入”的日志。

参数说明

- -d, --dbname=NAME ： 连接数据库 dbname 并直接导入到该数据库中。
- -f, --file=FILENAME ： 指定生成脚本的输出文件，或使用-l 时列表的输出文件。默认是标准输出。

  说明

  - -f 不能同-d 一起使用。
- -F, --format=c|d|t ： 指定归档格式。由于 gs_restore 会自动决定格式，因此不需要指定格式。取值范围：
  - – c/custom：该归档形式为 gs_dump 的自定义格式。
  - – d/directory：该归档形式是一个目录归档形式。
  - – t/tar：该归档形式是一个 tar 归档形式。
- -l, --list ： 列出归档形式内容。这一操作的输出可用作-L 选项的输入。注意如果像-n 或-t 的过滤选项与-l 使用，过滤选项将会限制列举的项目（即归档形式内容）。
- -v, --verbose ： 指定 verbose 模式。
- -V, --version ： 打印 gs_restore 版本，然后退出。

- -?, --help ： 显示 gs_restore 命令行参数帮助，然后退出。

导入参数：

- -a, --data-only ： 只导入数据，不导入模式（数据定义）。gs_restore 的导入是以追加方式进行的。
- -c, --clean ： 在重新创建数据库对象前，清理（删除）已存在于将要还原的数据库中的数据库对象。
- -C, --create ： 导入数据库之前会先使用 CREATE DATABASE 创建数据库。（指定该选项后，-d 指定的数据库仅用以执行 CREATE DATABASE 命令，所有数据依然会导入到创建的数据库中。）
- -e, --exit-on-error ： 当发送 SQL 语句到数据库时如果出现错误，请退出。默认状态下会继续，且在导入后会显示一系列错误信息。
- -I, --index=NAME ： 只导入已列举的 index 的定义。允许导入多个 index。如果多次输入-I index 导入多个 index。例如：

```
gs_restore -h 10.0.7.16 -p 5432 -d postgres -I Index1 -I Index2
/home/gbase/data/backup/MPPDB_backup.tar
```

在上面这个例子中，Index1 和 Index2 会被导入。

- -j, --jobs=NUM ： 运行 gs_restore 最耗时的部分（如加载数据、创建 index 或创建约束）使用并发任务。该选项能大幅缩短导入时间，即将一个大型数据库导入到某一多处理器的服务器上。

  每个任务可能是一个进程或一个线程，这由操作系统决定；每个任务与服务器进行单独连接。

  该选项的最优值取决于服务器的硬件设置、客户端以及网络。还包括这些因素， 如 CPU 核数量、硬盘设置。建议是从增加服务器上的 CPU 核数量入手，更大的值

  （服务器上 CPU 核数量）在很多情况下也能导致数据文件更快的被导入。当然， 过高的值会由于超负荷反而导致性能降低。

该选项只支持自定义归档格式。输入文件必须是常规文件（不能是像 pipe 的文件）。如果是通过脚本文件，而非直接连接数据库服务器，该选项可忽略。而且，多任务不能与--single-transaction 选项一起使用。

- -L, --use-list=FILENAME：只导入列举在 list-file 中的那些归档形式元素，导入顺序以它们在文件中的顺序为准。注意如果像-n 或-t 的过滤选项与-L 使用，它们将会进一步限制导入的项目。

  一般情况下，list-file 是通过编辑前面提到的某个-l 参数的输出创建的。文件行的 位置可更改或直接删除，也可使用分号（;）在行的开始注出。

- -n, --schema=NAME ：只导入已列举的模式中的对象。

  该选项可与-t 选项一起用以导入某个指定的表。多次输入-n schemaname 可以导入多个模式。例如：

```
gs_restore -h 10.0.7.16 -p 5432 -d postgres -n sch1 -n sch2
/home/gbase/data/backup/MPPDB_backup.tar
```

  在上面这个例子中，sch1 和 sch2 会被导入。

- -O, --no-owner ：不输出设置对象的归属这样的命令，以匹配原始数据库。默认情况下，gs_restore 会发出 ALTER OWNER 或 SET SESSION AUTHORIZATION 语句设置所创建的模式元素的所属。除非是由系统管理员（或是拥有脚本中所有对象的同一个用户）进行数据库首次连接的操作，否则语句会失败。使用-O 选项，任何用户名都可用于首次连接，且该用户拥有所有已创建的对象。

- -P, --function=NAME(args) ：只导入已列举的函数。请按照函数所在转储文件中的目录，准确拼写函数名称和参数。

  当-P 单独使用时，表示导入文件中所有'function-name(args)'函数；当-P 同-n 一起使用时，表示导入指定模式下的'function-name(args)'函数；多次输入-P，而仅指定一次-n，表示所有导入的函数默认都是位于-n 模式下的。

  可以多次输入-n schema-name -P 'function-name(args)'同时导入多个指定模式下的函数。

  例如：

```
gs_restore -h 10.0.7.16 -p 5432 -d postgres -n test1 -P 'Func1(integer)' -n test2
-P 'Func2(integer)' /home/gbase/data/backup/MPPDB_backup.tar
```

在上面这个例子中，test1 模式下的函数 Func1(i integer)和 test2 模式下的函数 Func2(j integer)会被一起导入。

- -s, --schema-only ：只导入模式（数据定义），不导入数据（表内容）。当前的序列值也不会导入。
- -S, --sysadmin=NAME ：该参数为扩展预留接口，不建议使用。
- -t, --table=NAME ：只导入已列举的表定义、数据或定义和数据。该选项与-n 选项同时使用时，用来指定某个模式下的表对象。-n 参数不输入时，默认为 PUBLIC 模式。多次输入-n <schemaname> -t <tablename>可以导入指定模式下的多个表。

例如：

导入 PUBLIC 模式下的 table1

```
gs_restore -h 10.0.7.16 -p 5432 -d postgres -t table1
/home/gbase/data/backup/MPPDB_backup.tar
```

导入 test1 模式下的 test1 和 test2 模式下 test2

```
gs_restore -h 10.0.7.16 -p 5432 -d postgres -n test1 -t test1 -n test2 -t test2
/home/gbase/data/backup/MPPDB_backup.tar
```

导入 PUBLIC 模式下的 table1 和 test1 模式下 test1

```
gs_restore -h 10.0.7.16 -p 5432 -d postgres -n PUBLIC -t table1 -n test1 -t table1
/home/gbase/data/backup/MPPDB_backup.tar
```

说明

- -t 不支持 schema_name.table_name 的输入格式。

- -T, --trigger=NAME ：该参数为扩展预留接口。
- -x, --no-privileges/--no-acl ：防止导入访问权限（GRANT/REVOKE 命令）。
- -1, --single-transaction ：执行导入作为一个单独事务 (即把命令包围在 BEGIN/COMMIT 中）。

该选项确保要么所有命令成功完成，要么没有改变应用。该选项意为--exit-on- error。

- --disable-triggers ：该参数为扩展预留接口，不建议使用。
- --no-data-for-failed-tables ：默认状态下，即使创建表的命令失败（如表已经存在），表数据仍会被导入。使用该选项，像这种表的数据会被跳过。如果目标数据库已包含想要的表内容，这种行为会有帮助。

  该选项只有在直接导入到某数据库中时有效，不针对生成 SQL 脚本文件输出。
- --no-publications ：不导入发布。
- --no-security-labels ：该参数为扩展预留接口，不建议使用。
- --no-subscriptions ：不导入订阅。
- --no-tablespaces ：不输出选择表空间的命令。使用该选项，无论默认表空间是哪个，在导入过程中所有对象都会被创建。
- --section=SECTION ：导入已列举的区段（如 pre-data、data 或 post-data）。
- --use-set-session-authorization ：该选项用来进行文本格式的备份。

  输出 SET SESSION AUTHORIZATION 命令，而非 ALTER OWNER 命令，用以决定

  对象归属。该选项使转储更加兼容标准，但通过参考转储中对象的记录，导入过程可能会有问题。使用 SET SESSION AUTHORIZATION 的转储要求必须是系统管理员，同时在导入前还需参考“SET SESSION AUTHORIZATION”，手工对导出文件的密码进行修改验证，只有这样才能进行正确的导入操作，相比之下， ALTER OWNER 对权限要求较低。
- --pipeline ：使用管道传输密码，禁止在终端使用。

注意：

如果安装过程中有任何本地数据要添加到 template1 数据库，请谨慎将 gs_restore 的输出载入到一个真正的空数据库中；否则可能会因为被添加对象的定义被复制，而出现错误。要创建一个无本地添加的空数据库，需从 template0 而非 template1 复制，例如：

```
postgres=# CREATE DATABASE foo WITH TEMPLATE template0;
```

gs_restore 不能选择性地导入大对象；例如只能导入那些指定表的对象。如果某个归档形式包含大对象，那所有大对象都会被导入或一个都不会被导入，如果它们通过-L、-t 或其他选项被排除。

说明

- -d/--dbname 和 -f/--file 不能同时使用；
- -s/--schema-only 和 -a/--data-only 不能同时使用；
- -c/--clean 和 -a/--data-only 不能同时使用；
- 使用--single-transaction 时，-j/--jobs 必须为单任务；
- --role 和 --rolepassword 必须一起使用。

连接参数：

- -h, --host=HOSTNAME ： 指定的主机名称。如果取值是以斜线开头，他将用作 Unix 域套接字的目录。默认值取自 PGHOST 环境变量；如果没有设置，将启动某个 Unix 域套接字建立连接。该参数只针对 GBase 8s 外，对 GBase 8s 内本机只能用 127.0.0.1。
- -p, --port=PORT ：指定服务器所侦听的 TCP 端口或本地 Unix 域套接字后缀，以确保连接。默认值设置为 PGPORT 环境变量。

  在开启线程池情况下，建议使用 pooler port，即侦听端口+1。
- -U, --username=NAME ： 所连接的用户名。
- -w, --no-password ： 不出现输入密码提示。如果服务器要求密码认证并且密码没有通过其它形式给出，则连接尝试将会失败。该选项在批量工作和不存在用户输入密码的脚本中很有帮助。
- -W, --password=PASSWORD ： 指定用户连接的密码。如果主机的认证策略是 trust，则不会对系统管理员进行密码验证，即无需输入-W 参数；如果没有-W 参数，并且不是系统管理员，“gs_restore”会提示用户输入密码。

- --role=ROLENAME ： 指定导入操作使用的角色名。选择该参数，会使 gs_restore 连接数据库后，发起一个 SET ROLE 角色名命令。当所授权用户（由-U 指定）没有 gs_restore 要求的权限时，该参数会起到作用，即切换到具备相应权限的角色。某些安装操作规定不允许直接以初始用户身份登录，而使用该参数能够在不违反该规定的情况下完成导入。
- --rolepassword=ROLEPASSWORD ： 指定具体角色用户的角色密码。

示例

示例 1

执行 gsql 程序，使用如下选项导入由 gs_dump/gs_dumpall 生成导出文件夹（纯文本格式）的 MPPDB_backup.sql 文件到 postgres 数据库。

```
gsql -d postgres -p 5432 -W gbase,123 -f
/home/gbase/data/backup/MPPDB_backup.sql
......
total time: 30476  ms
```

gs_restore 用来导入由 gs_dump 生成的导出文件。

示例 2

执行 gs_restore，将导出的 MPPDB_backup.dmp 文件（自定义归档格式）导入到 postgres 数据库。

```
gs_restore -W gbase,123 /home/gbase/data/backup/MPPDB_backup.dmp -p 5432 -d
postgres
gs_restore: restore operation successful
gs_restore: total time: 13053 ms
```

示例 3

执行 gs_restore，将导出的 MPPDB_backup.tar 文件（tar 格式）导入到 postgres 数据库。

```
gs_restore /home/gbase/data/backup/MPPDB_backup.tar -p 5432 -d postgres
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21203 ms
```

示例 4

执行 gs_restore，将导出的 MPPDB_backup 文件（目录格式）导入到 postgres 数据库。

```
gs_restore /home/gbase/data/backup/MPPDB_backup -p 5432 -d postgres
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21003 ms
```

示例 5

执行 gs_restore，使用自定义归档格式的 MPPDB_backup.dmp 文件来进行如下导入操作。导入 PUBLIC 模式下所有对象的定义和数据。在导入时会先删除已经存在的对象，如果原对象存在跨模式的依赖则需手工强制干预。

```
gs_restore /home/gbase/data/backup/MPPDB_backup.dmp -p 5432 -d postgres -e -c
-n PUBLIC
gs_restore: [archiver (db)] Error while PROCESSING TOC:
gs_restore: [archiver (db)] Error from TOC entry 313; 1259 337399 TABLE table1
gaussdba
gs_restore: [archiver (db)] could not execute query: ERROR: cannot drop table
table1 because other objects depend on it
DETAIL: view t1.v1 depends on table table1
HINT: Use DROP ... CASCADE to drop the dependent objects too.
Command was: DROP TABLE public.table1;
```

手工删除依赖，导入完成后再重新创建。

```
gs_restore /home/gbase/data/backup/MPPDB_backup.dmp -p 5432 -d postgres -e -c
-n PUBLIC
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 2203 ms
```

示例 6

执行 gs_restore，使用自定义归档格式的 MPPDB_backup.dmp 文件来进行如下导入操作。只导入 PUBLIC 模式下表 table1 的定义。

```
gs_restore /home/gbase/data/backup/MPPDB_backup.dmp -p 15400 -d postgres -e -c
-s -n PUBLIC -t table1
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 21000 ms
```

示例 7

执行 gs_restore，使用自定义归档格式的 MPPDB_backup.dmp 文件来进行如下导入操作。

只导入 PUBLIC 模式下表 table1 的数据。

```
gs_restore /home/gbase/data/backup/MPPDB_backup.dmp -p 15400 -d postgres -e -a
-n PUBLIC -t table1
gs_restore[2017-07-21 19:16:26]: restore operation successful
gs_restore[2017-07-21 19:16:26]: total time: 20203 ms
```

5.3.4 gs_backup

背景信息

GBase 8s 部署成功后，在数据库运行的过程中，会遇到各种问题及异常状态。GBase 8s 提供了 gs_backup 工具帮助 GBase 8s 备份、恢复重要数据、显示帮助信息和版本号信息。

前提条件

- 可以正常连接 GBase 8s 数据库。
- 在进行还原时，需要保证各节点备份目录中存在备份文件。
- 需以管理员用户 gbase 执行 gs_backup 命令。

语法

- 备份数据库主机

```
gs_backup -t backup --backup-dir=BACKUPDIR [-h HOSTNAME] [--parameter] [--binary]
[--all] [-l LOGFILE]
```

- 恢复数据库主机

```
gs_backup -t restore --backup-dir=BACKUPDIR [-h HOSTNAME] [--parameter]
[--binary] [--all] [-l
LOGFILE] [--force]
```

- 显示帮助信息

```
gs_backup -? | --help
```

- 显示版本号信息

```
gs_backup -V | --version
```

参数说明

gs_backup 参数可以分为如下几类：

备份数据库主机参数：

- -h ： 指定存储备份文件的主机名称。取值范围：主机名称。如果不指定主机名称，则备份当前集群。
- --backup-dir=BACKUPDIR ： 备份文件保存路径。
- --parameter ： 备份参数文件，不指定--parameter、--binary、--all 参数时默认只备份参数文件。
- --binary ： 备份 app 目录下的二进制文件。
- --all ： 备份 app 目录下的二进制文件、pg_hba.conf 和 postgsql.conf 文件。
- -l ： 指定日志文件及存放路径。默认值：$GAUSSLOG/om/gs_backup-YYYY-MM-DD_hhmmss.log

恢复数据库主机参数：

- -h ： 指定需要恢复主机的名称。取值范围：主机名称。如果不指定主机，则恢复对应的备份节点。
- --backup-dir=BACKUPDIR ： 恢复文件提取路径。
- --parameter ： 恢复参数文件，不指定--parameter、--binary、--all 参数时默认只恢复参数文件。
- --binary ： 恢复二进制文件。
- --all ： 恢复二进制和参数文件。
- -l ： 指定日志文件及存放路径。默认值：$GAUSSLOG/om/gs_backup-YYYY-MM-DD_hhmmss.log
- --force ：节点的静态文件丢失后强行 restore，仅限--all 或者--binary 一起使用时才生效。

其他参数：

- -?, --help ： 显示帮助信息。

- -V, --version ： 显示版本号信息。
- -t ： 指定操作类型。取值范围：backup 或者 restore。

示例

- 使用 gs_backup 脚本备份数据库主机。

```
gs_backup -t backup --backup-dir=/opt/software/gbase/backup_dir -h plat1
--parameter
Backing up GBase 8s.
Parsing configuration files.
Successfully parsed the configuration file. Performing remote backup.
Remote backup succeeded. Successfully backed up GBase 8s.
```

- 使用 gs_backup 脚本恢复数据库主机。

```
gs_backup -t restore --backup-dir=/opt/software/gbase/backup_dir -h plat1
--parameter
Restoring GBase 8s.
Parsing the configuration file. Successfully parsed configuration files.
Performing remote restoration.
Remote restoration succeeded. Successfully restored GBase 8s.
```

5.4 闪回恢复

闪回恢复功能是数据库恢复技术的一环，可以有选择性的撤销一个已提交事务的影响，将数据从人为不正确的操作中进行恢复。在采用闪回技术之前，只能通过备份恢复、PITR 等手段找回已提交的数据库修改，恢复时长需要数分钟甚至数小时。采用闪回技术后，恢复已提交的数据库修改前的数据，只需要秒级，而且恢复时间和数据库大小无关。

说明

ASTORE 引擎暂不支持闪回功能。

5.4.1 闪回查询

背景信息

闪回查询可以查询过去某个时间点表的某个 snapshot 数据，这一特性可用于查看和逻辑重建意外删除或更改的受损数据。闪回查询基于 MVCC 多版本机制，通过检索查询旧版本，

获取指定老版本数据。

语法

```
{[ ONLY ] table_name [ * ] [ partition_clause ] [ [ AS ] alias [ ( column_alias
[, ...] ) ] ]
[ TABLESAMPLE sampling_method ( argument [, ...] ) [ REPEATABLE ( seed ) ] ]
[TIMECAPSULE { TIMESTAMP | CSN } expression ]
|( select ) [ AS ] alias [ ( column_alias [, ...] ) ]
|with_query_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
|function_name ( [ argument [, ...] ] ) [ AS ] alias [ ( column_alias [, ...]
| column_definition [, ...] ) ]
|function_name ( [ argument [, ...] ] ) AS ( column definition [, ...] )
|from_item [ NATURAL ] join_type from_item [ ON join_condition | USING
( join_column [, ...] ) ]}
```

语法树中“TIMECAPSULE {TIMESTAMP | CSN} expression”为闪回功能新增表达方式，其中 TIMECAPSULE 表示使用闪回功能，TIMESTAMP 以及 CSN 表示闪回功能使用具体时间点信息或使用 CSN（commit sequence number）信息。

参数说明

- TIMESTAMP

指要查询某个表在 TIMESTAMP 这个时间点上的数据，TIMESTAMP 指一个具体的历史时间。

- CSN

指要查询整个数据库逻辑提交序下某个 CSN 点的数据，CSN 指一个具体逻辑提交时间点，数据库中的 CSN 为写一致性点，每个 CSN 代表整个数据库的一个一致性点，查询某个 CSN 下的数据代表 SQL 查询数据库在该一致性点的相关数据。

示例

示例 1

```
SELECT * FROM t1 TIMECAPSULE TIMESTAMP to_timestamp ('2020-02-11
10:13:22.724718', 'YYYY-MM-DD HH24:MI:SS.FF');
```

示例 2

```
SELECT * FROM t1 TIMECAPSULE CSN 9617;
```

示例 3

```
SELECT * FROM t1 AS t TIMECAPSULE TIMESTAMP to_timestamp ('2020-02-11
10:13:22.724718','YYYY-MM-DD HH24:MI:SS.FF');
```

示例 4

```
SELECT * FROM t1 AS t TIMECAPSULE CSN 9617;
```

5.4.2 闪回表

背景信息

闪回表可以将表恢复至特定时间点，当逻辑损坏仅限于一个或一组表，而不是整个数据库时，此特性可以快速恢复表的数据。闪回表基于 MVCC 多版本机制，通过删除指定时间点和该时间点之后的增量数据，并找回指定时间点和当前时间点删除的数据，实现表级数据还原。

前提条件

undo_retention_time 参数用于设置 undo 旧版本的保留时间。

语法

```
TIMECAPSULE TABLE   table_name TO { TIMESTAMP |   CSN }   expression
```

示例

```
TIMECAPSULE TABLE t1 TO TIMESTAMP to_timestamp ('2020-02-11 10:13:22.724718',
'YYYY-MM-DD HH24:MI:SS.FF');
TIMECAPSULE TABLE t1 TO CSN 9617;
```

5.4.3 闪回 DROP/TRUNCATE

背景信息

闪回 DROP：可以恢复意外删除的表，从回收站（recyclebin）中恢复被删除的表及其附属结构如索引、表约束等。闪回 drop 是基于回收站机制，通过还原回收站中记录的表的物理文件，实现已 drop 表的恢复。

闪回 TRUNCATE：可以恢复误操作或意外被进行 truncate 的表，从回收站中恢复 被 truncate 的表及索引的物理数据。闪回 truncate 基于回收站机制，通过还原回收 站中记录的表的物理文件，实现已 truncate 表的恢复。

前提条件

- 开启 enable_recyclebin 参数，启用回收站。
- recyclebin_retention_time 参数用于设置回收站对象保留时间，超过该时间的回收 站对象将被自动清理。

相关语法

- 删除表

```
DROP TABLE table_name [PURGE]
```

- 清理回收站对象

```
PURGE { TABLE { table_name }
| INDEX { index_name }
| RECYCLEBIN
}
```

- 闪回被删除的表

```
TIMECAPSULE TABLE { table_name } TO BEFORE DROP [RENAME TO new_tablename]
```

- 截断表

```
TRUNCATE TABLE { table_name } [ PURGE ]
```

- 闪回截断的表

```
TIMECAPSULE TABLE { table_name } TO BEFORE TRUNCATE
```

参数说明

- DROP/TRUNCATE TABLE table_name PURGE ：默认将表数据放入回收站中，PURGE 直接清理。
- PURGE RECYCLEBIN ： 表示清理回收站对象。
- TO BEFORE DROP ： 使用这个子句检索回收站中已删除的表及其子对象。

可以指定原始用户指定的表的名称，或对象删除时数据库分配的系统生成名称。

回收站中系统生成的对象名称是唯一的。因此，如果指定系统生成名称，那么数据库检索指定的对象。使用“select * from gs_recyclebin;”语句查看回收站中的内容。

如果指定了用户指定的名称，且如果回收站中包含多个该名称的对象，然后数据库检索回收站中最近移动的对象。如果想要检索更早版本的表，你可以这样做：

- 指定你想要检索的表的系统生成名称。
- 执行 TIMECAPSULE TABLE ... TO BEFORE DROP 语句，直到你要检索的表。
- 恢复 DROP 表时，只恢复基表名，其他子对象名均保持回收站对象名。用户可根据需要，执行 DDL 命令手工调整子对象名。
- 回收站对象不支持 DML、DCL、DDL 等写操作，不支持 DQL 查询操作（后续支持）。
- 闪回点和当前点之间，执行过修改表结构或影响物理结构的语句，闪回失败。执行过 DDL 的表进行闪回操作报错：“ERROR：The table definition of %s has been changed.”。涉及 namespace、表名改变等操作的 DDL 执行闪回操作报错：ERROR: recycle object %s desired does not exis；

- RENAME TO ：为从回收站中检索的表指定一个新名称。
- TO BEFORE TRUNCATE ：闪回到 TRUNCATE 之前。

语法示例

```
DROP TABLE t1 PURGE;

PURGE TABLE t1;
PURGE TABLE "BIN$04LhcpndanfgMAAAAAANPw==$0"; PURGE INDEX i1;
PURGE INDEX "BIN$04LhcpndanfgMAAAAAANPw==$0"; PURGE RECYCLEBIN;

TIMECAPSULE TABLE t1 TO BEFORE DROP;
TIMECAPSULE TABLE t1 TO BEFORE DROP RENAME TO new_t1;
TIMECAPSULE TABLE "BIN$04LhcpndanfgMAAAAAANPw==$0" TO BEFORE DROP;
```

```
TIMECAPSULE TABLE "BIN$04LhcpndanfgMAAAAAANPw==$0" TO BEFORE DROP RENAME TO
new_t1;
```

6 扩容和缩容

6.1 扩容

操作步骤

GBase 8s 数据库集群扩容的步骤如下：

步骤 1 在新增节点的主机上配置集群节点间互信，详细操作参见《GBase 8s V8.8_安装部署手册》中 3.4 章节“配置互信”。

⚠ 注意

该步骤必须配置并检查，如配置互信过程出错，可能影响后续步骤。

步骤 2 在 DCS 节点上，在集群中添加新增节点的 IP。

```
[gbase@gbase8s ~]$ gha_ctl deploy host_ip -l http://dcs_ip:2379
```

执行返回以下信息，即为操作成功，

```
{
    "ret":0,
    "msg":"Success"
}
```

步骤 3 执行扩容命令。

将原集群的 M 个 DN 主备组扩容为 N 个主备组（N>M），将新增加的每一个主备组信息用一组引号单独括起来，每个主备的信息是 DN 组名+用括号括起来的所有主备节点的信息，一个节点用一个括号。语法如下：

```
gha_ctl expand datanode 'new_group1 (name1_1 host1_1 port1_1 dir1_1
agent_port1_1 agent_host1_1) (name1_2 host1_2 port1_2 dir1_2 agent_port1_2
agent_host1_2)' 'new_group2 (name2_1 host2_1 port2_1 dir2_1 agent_port2_1
agent_host2_1) (name2_2 host2_2 port2_2 dir2_2 agent_port2_2 agent_host2_2)'
['...'] -l dcslist [-c cluster] -u uuid -j parallel_num
```

其中参数说明：

- new_group：指定新增高可用组的名称。可根据需要，同时扩充多个高可用组。

- name：指定高可用组内的 DN 节点名称，一般格式应为 dn[数字]_[数字]。
- host：指定扩容高可用组内对应 DN 节点的 IP。
- port：指定对应 DN 节点的端口号。同高可用组内端口需一致。当资源限制需重复使用同一节点，在不同高可用组内分别承担主备角色时，需将两个高可用组端口设置不同，避免端口复用而报错。
- dir：指定对应 DN 节点的存储路径。
- agent_port：指定对应 DN 节点的代理端口号。
- agent_host：为可选参数。缺省值为 host 参数值。指定 DN 节点控制面 IP。
- uuid：通过唯一标识指定数据库。uuid 可以用如下命令生成：

```
cat /proc/sys/kernel/random/uuid
```

- parallel_num：指定扩容时并行执行参数，多表同时执行重分布。默认值为 1。

示例

示例一

扩容 DN2 高可用组，组内主节点为 10.0.7.6，备节点 10.0.7.7。

```
[gbase@gbase8s ~]$ cat /proc/sys/kernel/random/uuid
40ac7d83-6be3-486c-83c4-8942a16d3590
[gbase@gbase8s ~]$ gha_ctl expand datanode 'dn2 (dn2_1 10.0.7.6 15466
/home/gbase/data/dn2_1 8011 10.0.7.6) (dn2_2 10.0.7.7 15466
/home/gbase/data/dn2_2 8012 10.0.7.7) ' -l http://10.0.7.7:2379 -u
40ac7d83-6be3-486c-83c4-8942a16d3590
```

返回以下信息，表示扩容成功

```
{
    "ret":0,
    "msg":"Success"
}
```

查看数据库集群状态

```
[gbase@gbase8s ~]$ gha_ctl monitor all -H -l http://10.0.7.7:2379
+----+------------+---------+-------+---------+--------+
```

```
| No |     name     |   host    |  port |  state  | leader |
+----+--------------+-----------+-------+---------+--------+
| 0  | gha_server1 | 10.0.7.7 | 20001 | running |  True  |
+----+--------------+-----------+-------+---------+--------+
+----+------+---------+------+-----------------------------+---------+-------
--+
+----+-------+-------+---------+-------+-----------------------------+------
---+---------+
| No | group |  name |   host    |  port |          work_dir           |  state
|   role  |
+----+-------+-------+---------+-------+-----------------------------+------
---+---------+
| 0  |  dn1  | dn1_1 | 10.0.7.7 | 15432 | /home/gbase/data/dn1/dn1_1 | running
| primary |
| 1  |  dn1  | dn1_2 | 10.0.7.8 | 15432 | /home/gbase/data/dn1/dn1_2 | running
| standby |
+----+-------+-------+---------+-------+-----------------------------+------
---+---------+
+----+---------------------+--------+---------+----------+
| No |         url         |  name  |  state  | isLeader |
+----+---------------------+--------+---------+----------+
| 0  | http://10.0.7.7:2379 | node_0 | healthy |   True   |
| 1  | http://10.0.7.8:2379 | node_1 | healthy |  False   |
| 2  | http://10.0.7.9:2379 | node_2 | healthy |  False   |
+----+---------------------+--------+---------+----------+
```

6.2 缩容

语法格式

```
gha_ctl shrink datanode del_group1 [del_group2 ...] -l dcslist [-c cluster] -u
uuid -j parallel_num
```

将原集群的 M 个 DN 主备组缩容为 N 个主备组（N<M），del_group1, del_group2 为缩容时需要删除的 DN 主备组名称。其他参数说明同上。

示例

```
[gbase@gbase8s ~]$ gha_ctl shrink datanode dn3 dn4 -l http://10.0.7.7:2379 -u
b99ee57c-8b90-4196-896c-19d58bdaae6a
```

6.3 查询扩缩容结果

语法格式

```
gha_ctl get expand latest/history -l dcslist [-c cluster]
```

示例

```
[gbase@gbase8s ~]$ gha_ctl get expand latest -l http://10.0.7.16:2379
[gbase@gbase8s ~]$ gha_ctl get expand history -l http://10.0.7.16:2379
```

7 高危操作一览表

在产品的操作与维护阶段，进行日常各项操作时请严格遵守指导书。同时避免执行如下严禁、高危操作，见表 7-1、表 7-2。

表 7-1 禁用操作

操作名称	操作风险
严禁修改数据目录下文件名，权限，内容不能修改，不能删除内容。	导致数据库节点实例出现严重错误，并且无法修复。
严禁删除数据库系统表或系统表数据。	删除系统表将导致无法正常进行业务操作。

表 7-2 高危操作

操作分类	操作名称	操作风险	风险等级	规避措施	重大操作观察项目
数据库	直接在配置文件中手动修改端口号。	导致数据库启动不了或者连接不上。	▲▲ ▲▲ ▲	尽量使用工具修改，不要手动操作。	无。
	修改 pg_hba.conf 配置文件中的内容。	导致客户端连接不上。	▲▲ ▲▲ ▲	严格根据产品手册操作。	无
	手动修改 pg_xlog 的内容。	导致数据库无法启动，数据不一致。	▲▲ ▲▲ ▲	尽量使用工具修改，不要手动操作。	无
作业	使用 kill -9 终止作业进程。	导致作业占用的系统资源无法释放。	▲▲ ▲	登录数据库尽量使用 pg_terminate_backend，pg_cancel_backend	观察资源使用情况。

				操作终止作业，或使用 Ctrl+C 终止作业进程。	

8 日志参考

8.1 日志类型简介

在数据库运行过程中，会出现大量日志，既有保证数据库安全可靠的 WAL 日志（预写式日志，也称为 Xlog），也有用于数据库日常维护的运行和操作日志等。在数据库发生故障时，可以参考这些日志进行问题定位和数据库恢复的操作。

日志类型

日志类型的详细说明，请参见下表。

表 8-1 日志类型

类型	说明
系统日志	数据库系统进程运行时产生的日志，记录系统进程的异常信息。
操作日志	通过客户端工具（例如 gs_guc）操作数据库时产生的日志。
Trace 日志	通过 gstrace 工具指定启动 trace 功能的 DN 实例后，会记录大量的 Trace 日志。这些日志可以用来分析数据库的异常信息。
黑匣子日志	数据库系统崩溃的时候，通过故障现场堆、栈信息可以分析出故障发生时的进程上下文，方便故障定位。黑匣子具有在系统崩溃时，dump 出进程和线程的堆、栈、寄存器信息的功能。
审计日志	开启数据库审计功能后，将数据库用户的某些操作记录在日志中，这些日志称为审计日志。
WAL 日志	又称为 REDO 日志，在数据库异常损坏时，可以利用 WAL 日志进行恢复。由于 WAL 日志的重要性，所以需要经常备份这些日志。
性能日志	数据库系统在运行时检测物理资源的运行状态的日志，在对外部资源进行访问时的性能检测。

8.2 系统日志

GBase 8s 数据库运行时，集群和节点安装部署时产生的日志统称为系统日志。如果在数据库运行时发生故障，可以通过这些系统日志及时定位故障发生的原因，根据日志内容制定恢复 GBase 8s 的方法。

日志文件存储路径

- 数据库节点的运行日志放在“/home/gbase/gbase8s/log/pg_log”中各自对应的目录下。
- OM GBase 8s 安装卸载时产生的日志放在“/home/gbase/gbase8s/log/om”目录下。

日志文件命名格式

- 数据库节点运行日志的命名规则：

```
postgresql-创建时间.log
```

默认情况下，每日 0 点或者日志文件大于 16MB 或者数据库实例（数据库节点）重新启动后，会生成新的日志文件。

- CM 的运行日志的命名规则：
  - cm_agent 的日志：

```
cm_agent-创建时间.log/cm_agent-创建时间-current.log/system_call-创建时间.log/system_call-创建时间-current.log
```

  - cm_server 的日志：

```
cm_server-创建时间.log/cm_server-创建时间-current.log/key_event-创建时间.log/key_event-创建时间-current.log
```

  - om_monitor 的日志：

```
om_monitor-创建时间.log/om_monitor-创建时间-current.log
```

其中，不带 current 标识符的文件是历史日志文件，带 current 标识符的文件是当前日志文件。最初调用进程时，进程会先创建一个带 current 标识符的日志文件，当该日志文件的大小超过 16MB 时，会将当前日志文件重命名为历史日志文件，并以当前时间生成新的当前日志文件。

日志内容说明

- 数据库节点每一行日志内容的默认格式：

日期+时间+时区+用户名称+数据库名称+会话 ID+日志级别+日志内容

- cm_agent、cm_server、om_monitor 每一行日志内容的默认格式：

时间+时区+会话 ID+日志内容

SYSTEM_CALL 系统调用日志，记录 CM_AGENT 在运行过程中调用工具命令的情况。

- key_event 每一行日志内容的默认格式:

时间+线程号+线程名：关键事件类型+仲裁对象实例 ID+仲裁细节

8.3 操作日志

数据库管理员使用工具操作数据库时，或工具被数据库调用时，会产生操作日志。如果 GBase 8s 发生故障，可以通过这些日志信息跟踪用户对数据库进行了哪些操作，重现故障场景。

日志文件存储路径

默认在“$GAUSSLOG/bin”目录下，如果环境变量$GAUSSLOG 不存在或者变量值为空，则工具日志信息不会记录到对应的工具日志文件中，日志信息只会打印到屏幕上。

其中，环境变量$GAUSSLOG 默认为“/home/gbase/gbase8s/log”。

说明

如果使用 om 脚本部署时，则日志路径为”/home/gbase/gbase8s/log/om”。

日志文件命名格式

日志文件命名格式为：

工具名-日志创建时间.log
工具名-日志创建时间-current.log

其中，“工具名-日志创建时间.log”是历史日志文件，“工具名-日志创建时间- current.log”是当前日志文件。

如果日志大小超过 16MB，在下一次调用该工具时，会重命名当前日志文件为历史日志文件，并以当前时间生成新的当前日志文件。例如，将“gs_guc-2015-01-16_183728-current.log”重命名为“gs_guc-2015-01-16_183728.log”，然后重新生成“gs_guc-2015-01-17_142216-current.log”。

维护建议

建议定时对过期的日志文件进行转储，以避免发生大量日志占用太多的磁盘空间、重要日志丢失等问题。

8.4 审计日志

审计功能开启时会不断产生大量的审计日志，占用磁盘空间。用户可以根据磁盘空间的大小设置审计日志维护策略。

关于如何设置审计日志维护策略请参见《GBase 8s V8.8_开发者指南》中“管理数据库安全 > 设置数据库审计 > 维护审计日志”章节。

预写式日志 WAL（Write Ahead Log，也称为 Xlog）是指如果要修改数据文件，必须是在这些修改操作已经记录到日志文件之后才能进行修改，即在描述这些变化的日志记录刷新到永久存储器之后。在系统崩溃时，可以使用 WAL 日志对 GBase 8s 进行恢复操作。

日志文件存储路径

以一个数据库节点为例，默认在“/home/gbase/data/data_dn/pg_xlog”目录下。其中“/home/gbase/data/”代表 GBase 8s 节点的数据目录。

日志文件命名格式

日志文件以段文件的形式存储的，每个段为 16MB，并分割成若干页，每页 8KB。对 WAL 日志的命名说明如下：一个段文件的名称由 24 个十六进制组成，分为三个部分，每个部分由 8 个十六进制字符组成。第一部分表示时间线，第二部分表示日志文件标号，第三部分表示日志文件的段标号。时间线由 1 开始，日志文件标号和日志文件的段标号由 0 开始。

例如，系统中的第一个事务日志文件是 000000010000000000000000。

说明

这些数字一般情况下是顺序增长使用的（要把所有可用数字都用光也需要非常长的时间），但也存在循环使用的情况。

日志内容说明

WAL 日志的内容取决于记录事务的类型，在系统崩溃时可以利用 WAL 日志进行恢复。默认配置下，GBase 8s 每次启动时会先读取 WAL 日志进行恢复。

维护建议

WAL 日志对数据库异常恢复有重要的作用，建议定期对 WAL 日志进行备份。

8.5 性能日志

性能日志主要关注外部资源的访问性能问题。性能日志指的是数据库系统在运行时检测物理资源的运行状态的日志，在对外部资源进行访问时的性能检测，包括磁盘、Hadoop GBase 8s等外部资源的访问检测信息。在出现性能问题时，可以借助性能日志及时的定位问题发生的原因，能极大地提升问题解决效率。

日志文件存储路径

数据库节点的性能日志目录在“$GAUSSLOG/gs_profile”中各自对应的目录下。日志文件命名格式：

- 数据库节点的性能日志的命名规则：

```
postgresql-创建时间.prf
```

默认情况下，每日 0 点或者日志文件大于 20MB 或者数据库实例（数据库节点）重新启动后，会生成新的日志文件。

日志内容说明

- 数据库节点每一行日志内容的默认格式：

```
主机名称+日期+时间+实例名称+线程号+日志内容
```

GBASE®

南大通用数据技术股份有限公司
General Data Technology Co., Ltd.

微信二维码

技术支持热线：400-013-9696