**集群间数据传输命令**

**使用说明**

作者：陈淼

版本：1.7 (命令文件MD5值为67736fd7584659164bcc2ca282da5f15)

日期：2017/09/10

# 命令介绍

gpdbtransfer命令，用于在两个GP集群之间同步数据，此命令是使用PERL语言编写，运行此命令需要有PERL语言环境(缺省都有)，此命令使用psql客户端访问GP数据库，需要确保执行该命令的用户已经具备psql环境。

gpdbtransfer命令，可以运行在需要同步数据的两个集群之外的主机上，但需要确保该机器可以访问需要同步数据的两个集群的数据库，命令仅访问Master机器，无需访问GP数据库的数据节点。

需要同步数据的两个集群之间，所有节点之间网络需要互通，否则命令将无法正常运行。通常，难以保证集群之间互相识别主机名，目标集群访问源集群节点的IP地址，可以通过配置文件配置主机名对应的IP地址，以确保访问路径。

命令会自动判断两个集群之间是否适合使用并行模式传输数据，如果目标端实例数量不小于源端实例数量，将会采用全并行模式，如果目标端实例数量小于源端实例数量，且不小于源端主机数量，将会采用普通并行模式，否则，数据传输需要经由Master主机进行传输。对于需要有Motion的视图传输，也需要经由Master主机进行传输。

此命令完全废除了命名管道的实现方案，采用外部表的transform方案，极大的提高了命令的稳定性和可靠性，同时，修复了一些小BUG，传输数据的效率更高，命令的参数选择很多，大大的增加了使用的灵活性，可以根据不同的需求场景调整参数，以达到最高的效果。

建议在大规模同步表数据之前，尽量选择一个小表做验证测试。

**注意**：

目前版本会在传输时对两端的表进行加锁处理。如果加锁失败，该表的同步将会失败。尤其是源端的表，因为实现原理的缘故，直接访问的是数据节点，在有些时候可能会出现死锁问题。由于视图不支持LOCK，因此命令会判断视图锁依赖的表情况，但目前处于慎重考虑，仅支持到3层依赖，第4层的视图将无法正确识别其依赖的表清单，一般不太会有更多层次的依赖情况。另外，为了避免对很多的表进行加锁，视图依赖的表数量超过8个的情况也不进行源端加锁处理。目前的版本在对失败的清理上进行了强化，原本只清理残余SQL，后发现这样是不足够的，目前还会对残余的psql命令进行强杀，以彻底清除残余进程。

# 退出码说明

退出值含义：

0 全部任务SUCCESS

1 脚本报错退出

11 无表需要传输

33 传输的表中有异常(未全SUCCESS)

# 命令部署

将gpdbtransfer文件复制到需要运行该命令的用户的~/bin目录下。修改owner和mod。例如:



# 命令用法

|  |
| --- |
| gpdbtransfer --src-host hostname  [--src-port database port]  [--src-user database user]  [--src-passwd database password]  --src-database database name  --dest-host hostname  [--dest-port database port]  [--dest-user database user]  [--dest-passwd database password]  --dest-database database name  [--key identity number]  --src-mapfile filename  [--by-leaf]  [--force]  [--owner owner]  [-t <schema.relation> [-t <schema.relation>] ...]  [-T <schema.relation> [-T <schema.relation>] ...]  [-f file include table name]  [-F file include table name]  [-s <schema> [-s <schema>] ...]  [-S <schema> [-S <schema>] ...]  [--delete]  [--truncate]  [--analyze]  [--encoding encoding]  [-B batch\_size]  [--where condition]  [--log-path directory]  [--parameter-file filename]  [-h|--help] |

# 参数说明

**如果不指定任何表名，则缺省传输指定数据库中的全部用户表。**

**除了**--parameter-file**参数和**-h|--help**参数，其他参数均可以通过参数文件的形式提供。详见**--parameter-file**参数说明。**

* **--src-hostname**

必选参数，源端数据库集群的Master主机名或IP地址。

* **--src-port**

可选参数，源端数据库集群的服务端口。缺省为5432。

* **--src-user**

可选参数，源端数据库集群的登录用户名。缺省为gpadmin。

建议使用超级用户，避免权限不足问题。

* **--src-passwd**

可选参数，源端数据库集群的登录密码。

缺省需要pg\_hba中有trust支持。

* **--src-database**

必选参数，源端GP数据库的名称，即，要从哪个数据库向外提供数据。

* **--dest-hostname**

必选参数，目标数据库集群的Master主机名或IP地址。

* **--dest-port**

可选参数，目标数据库集群的服务端口。缺省为5432。

* **--dest-user**

可选参数，目标数据库集群的登录用户名。缺省为gpadmin。

建议使用超级用户，避免权限不足问题。

* **--dest-passwd**

可选参数，目标数据库集群的登录密码。

缺省需要pg\_hba中有trust支持。

* **--dest-database**

必选参数，目标GP数据库的名称，即，目标端接收数据的数据库名称。

* **--key**

用于区分不同gpdbtransfer操作的标识符，缺省的值为20140825，选择这个值仅仅是因为作者女儿的生日，你可以修改为任意你喜欢的数字。由于性能原因，从您看到这个新的文档开始，作者已经永远放弃之前基于gpfdist的原理的实现，gpfdist原理实现的性能瓶颈严重，根本无法满足大规模全量数据迁移的需求。

命令启动时会在源端和目标端检查对应这个key的锁文件，如果发现已经存在锁文件，命令会尝等待该锁文件的状态，最长等待时间为60秒，期间，如果发现锁文件被其他gpdbtransfer进程更新了状态，命令会认为该key处于不可用状态，命令直接报错退出。如果在60秒内，锁文件一直无更新，命令会认为锁文件是之前执行gpdbtransfer时异常退出的残留锁文件，命令会尝试继续使用这个key并开始工作。

gpfdist服务的服务目录是/tmp/fifo，这也是gpdbtransfer的工作目录，在该目录下你可能会看到类似如下文件列表：

|  |
| --- |
| 20140825\_0.where  20140825\_1.where  20140825\_2.where  20140825\_3.where  20140825.info  gpdbtransfercat.sh |

$key.info文件会随着不同的命令运行而发生变化，其存储的是源端集群的节点信息，$key.where文件在命令运行过程中会不断更新用于存放源端的过滤条件，gpdbtransfercat.sh文件一般不会发生变化，其内容由命令根据版本更新情况自动维护，请不要手动修改这些文件。

* **--src-mapfile**

必选参数，源端主机名与IP地址的配置文件。

一般来说，不同集群之间的主机名互相不识别，但IP地址是可以互相访问的，该文件指定源端数据库主机名所对应的IP地址，以确保两个集群之间可以传输数据。

例如：

mdw,171.28.4.250

smdw,171.28.4.251

sdw1,172.28.4.1

sdw2,172.28.4.2

sdw3,172.28.4.3

sdw4,172.28.4.4

建议将gp\_segment\_configuration表中的所有hostname都配置对应的IP地址，以确保任何情况下网络都是可以访问的，如果你的主机名包含大写，请以gp\_segment\_configuration系统表中的

address字段为准。

如果出现未配置的主机名，命令会输出提示信息，但命令仍会继续运行，在正式开始transfer之前，命令会尝试检查源端集群的Content是否与根据src-mapfile生成的源端集群信息文件中的一致，如果有任何的不一致，命令将会报错退出，所以，务必确保src-mapfile文件的配置正确。如果目标集群可以识别源端集群的hostname，可以提供一个空的配置文件，命令会继续执行。

* **--by-leaf**

只传输那些没有子表的叶子表，这样做对于那些分区粒度特别细，分区规模特别庞大的场景很有必要，此参数将只允许那些没有子表的数据表被传输，避免传输超大分区表时同时打开巨量数据文件。

此参数需要尽量避免与--force参数同时出现，如果叶子分区在目标端不存在，命令强制建的表只是保证了表的结构，无法保证分区关系。

对于需要在2个集群之间复制所有表数据的情况，建议事先将ddl在目标端恢复，确保两端的表字段定义完全一致。

* **--force**

如果命令发现目标数据库中表结构与源端不一致，将强制按照源端的结构重建目标表。

在不指定该参数的情况下，如果出现表结构不一致的情况，该表的同步将会被跳过并给出提示信息。

慎用该参数，尽量不要与参数--by-leaf一起使用。

使用--force的时候，命令不会理会源端数据库表的其他信息，只会按照字段定义重建表，并使用zlib5行压缩存储格式，此参数仅在必要时使用。对于那些源端对象有comment信息的场景，命令会将relation及其字段的comment信息一并同步到目标表中——仅当需要在目标端重建表时。

* **--owner**

重建目标表时表的所有者。仅当需要重建目标表时才会发生这种操作。

因此，当指定了--force参数和--owner参数，且目标端表结构与源端不一致时，才会设定目标表的owner。

* **-t**

被传输的表名，以schema.relname的方式提供，命令会自动检查该表是否存在，如不存在，会忽略该表并给出提示信息。

* **-T**

命令将会忽略该表的传输。

以schema.relname的方式提供。

* **-f**

需要被传输的表清单文件。

可以被接受的格式为：

schema\_a.relname\_a[=>schema\_b.relname\_b][;some\_code='021']

分别表示：

源端表名=>目标表名;过滤条件

目标表名，过滤条件，都是可选项，仅当需要在不同表名之间传输数据时使用目标表名，仅当需要按条件传输数据时使用过滤条件。使用过滤条件时，需要确保条件的正确定，否则会导致传输失败。

允许从源端的视图传输到目标端的实体表中。这样，传输的数据等同于从视图中直接查询到的数据。

* **-F**

命令将会忽略的表清单文件。

以schema.relname的方式提供。

* **-s**

被传输的模式名称，命令会将指定schema中的表数据全部传输，不过那些系统模式和临时模式是永远会被忽略的。

* **-S**

需要被命令忽略的模式名称。这些模式中的表将不会被传输，即便在其他参数中指定了的表也会被忽略。

* **--delete**

在传输之前，将目标表中符合过滤条件的数据删除。

如果没有过滤条件，将采取全表清空的方式，与--truncate效果等效。

该参数不能与--truncate参数一同使用。

* **--truncate**

在传输之前，将目标表的数据清空。

该参数不能与--delete参数一同使用。

* **--analyze**

收集目标表的统计信息。

如果不指定该参数，将不会收集目标表的统计信息。

命令中会显示关闭gp\_autostats\_mods参数。

当指定analyze时，会收集目标表的ctid字段，仅收集数据量的统计信息，如需收集字段统计信息，请另行安排脚本跑批。这样的设计可以避免传输过程中消耗过多的资源用于收集统计信息，确保尽快完成数据传输操作。

* **--encoding**

指定数据从源端数据库导出，以及在目标端数据库导入时使用的编码方式。缺省使用UTF8编码，该编码为GP数据库库内编码，对于大多数用户来说，此编码已经足够，如果你的数据库使用了李巍大神的delimiter外部表，那么你可能需要用到此参数，因为库内可能会有中文乱码，此时，建议将附件中的2个文件选择一个覆盖数据库中的原始so包文件，该文件修改了原有的编码函数，可以确保数据乱码不会报错，但首先需要确保该编码方式在项目中永远不会被使用，否则可能会影响正常转码需求，不过好在一般都用不到此编码函数。.cm的文件是由作者尝试创作，.liwei文件由李巍大神提供，至于你喜欢用哪个，我就不管了。

/usr/local/greenplum-db-4.3.7.1/lib/postgresql/utf8\_and\_iso8859\_1.so



或者使用如下文件进行base64解码得到so文件



文件MD5值为

f714c8658f36c7d202f174c5795bca6d

也可以使用如下两个文件进行手工编译，make.sh为编译命令，2个文件放在一个目录，执行sh make.sh即可



* **-B**

并发数量，同时运行数据传输的表的个数。

对于目标端的实例数量不小于源端实例数量的情况，获取源端数据的脚本是全并行的，即，源端的每个Primary Instance都会有一个COPY在取数，而对于目标端的实例数量小于源端实例数量而不小于源端主机数量的情况，源端每个主机上只会有一个COPY在运行，其会循环访问该主机上的所有Primary Instance，此时，如果需要增加资源使用，可以适当再加大一些该参数。

* **--where**

传输数据的过滤条件。此为全局条件，对于在-f中指定了条件的情况，此参数不能覆盖。-f表清单中指定的条件优先级高于--where指定的条件。

注意条件部分不能包含where关键字，且条件最好用双引号引起来，例如：

--where “somecode=’201’ and date=’20160101’”

* **--log-path**

日志目录，缺省是用户目录的gpAdminLogs目录。

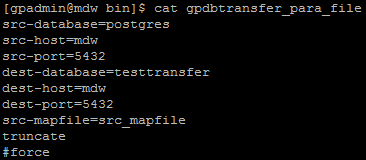
生成的日志文件格式为gpdbtransfer\_YYYYMMDD.log。

* **--parameter-file**

通过文件的方式指定参数，建议将通常不会发生变化的参数使用参数文件的方式指定。

此前的所有参数均可以通过参数文件的方式指定。

格式为：



在命令行指定的参数优于参数文件中指定的参数。

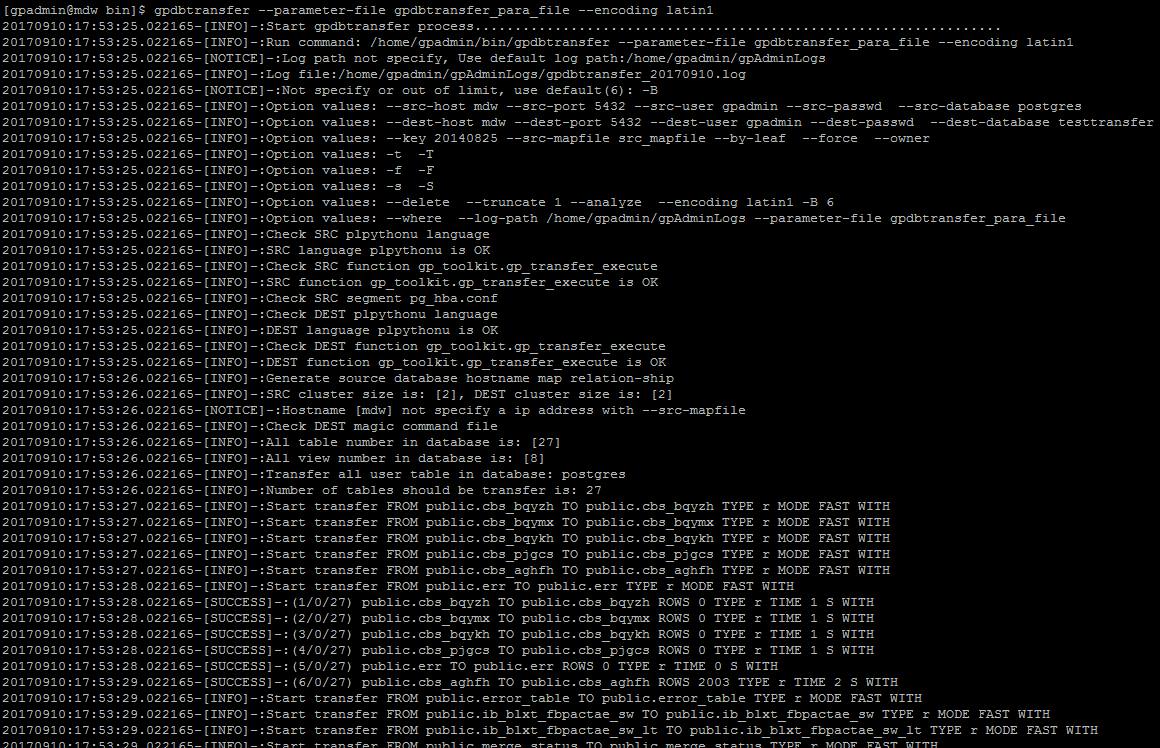
对于多选类参数，不存在覆盖特性，两者都是有效值。

对于开关类参数，不存在覆盖特性，因为这些参数缺省是关闭的，唯一可选择的就是打开，因此，有一处打开即为打开。

* **-h|--help**

显示帮助信息。

# 命令示例



命令的输出为同步进度信息，成功的信息以[SUCCESS]为标记，失败的以[FAILED]为标记，之后的三个数字分别代表，成功数量/失败数量/任务总数。

# 跨集群全量数据迁移建议

## 新Cluster初始化安装配置

根据官方建议在新的硬件集群或者使用不同的服务端口和目录在现有集群初始化一个新的GP集群，此处细节不在此文档范围，不做细说。

## 原有Cluster DDL备份

pg\_dumpall -p $OLD\_PORT –h $OLD\_HOST -s -g -r -f $DBNAME.global.ddl

pg\_dump -p $OLD\_PORT –h $OLD\_HOST -s dp\_bidb -f $DBNAME.ddl

## 在新Cluster上恢复DDL

psql -p $NEW\_PORT –h $NEW\_HOST postgres -c "create database $DBNAME "

psql -p $NEW\_PORT –h $NEW\_HOST $DBNAME -f $DBNAME.global.ddl > /dev/null

psql -p $NEW\_PORT –h $NEW\_HOST $DBNAME -f $DBNAME.ddl > /dev/null

psql -p $NEW\_PORT –h $NEW\_HOST $DBNAME -f $DBNAME.global.ddl > /dev/null

## 从原有Cluster上获取删除重建索引脚本

psql -p $OLD\_PORT –h $OLD\_HOST $DBNAME <<EOF

copy (SELECT 'drop index '||schemaname||'.'||indexname||';' from pg\_indexes where schemaname

in (select nspname from pg\_namespace where oid > 16384

or oid=2200) and indexname not like '%pkey' order by 1)

to '$DBNAME.drop.index.sql';

copy (SELECT indexdef||';' from pg\_indexes where schemaname

in (select nspname from pg\_namespace where oid > 16384

or oid=2200) and indexname not like '%pkey' order by 1)

to '$DBNAME.create.index.sql';

copy (SELECT 'drop index '||schemaname||'.'||indexname||';' from pg\_indexes where schemaname

in (select nspname from pg\_namespace where oid > 16384

or oid=2200) and indexname like '%pkey' order by 1)

to '$DBNAME.drop.pkey.sql';

copy (SELECT indexdef||';' from pg\_indexes where schemaname

in (select nspname from pg\_namespace where oid > 16384

or oid=2200) and indexname like '%pkey' order by 1)

to '$DBNAME.create.pkey.sql';

EOF

有时，可能pkey的删除有问题，需要从pg\_constraint系统表中获取contype为p的记录生成pkey删除脚本。

## 在新Cluster上临时删除索引信息

cat $DBNAME.drop.index.sql|

xargs -i -P 8 psql -p $NEW\_PORT –h $NEW\_HOST $DBNAME -c "{}"

cat $DBNAME.drop.pkey.sql|

xargs -i -P 8 psql -p $NEW\_PORT –h $NEW\_HOST $DBNAME -c "{}"

## 执行数据传输命令

gpdbtransfer --parameter-file parameter\_file

所有的参数都可以在parameter\_file中指定，此处参考前述参数说明和命令示例。

在做全量数据传输时，建议关闭truncate选项，因为目标库是空的，truncate也会浪费时间，建议关闭analyze选项，在数据传输完之后再做analyze。

## 从原有Cluster上获取恢复Sequence信息

psql -p $OLD\_PORT –h $OLD\_HOST $DBNAME <<EOF

copy (SELECT 'select setval('''||nspname||'.'||relname||''','||nextval(c.oid)||');'

from pg\_class c,pg\_namespace n where c.relnamespace = n.oid and relkind='S'

and (n.oid > 16384 or n.oid = 2200))

to '$DBNAME.sequence.next.sql';

EOF

## 在新Cluster上恢复Sequence信息和索引

psql -p $NEW\_PORT –h $NEW\_HOST dp\_bidb -f $DBNAME.sequence.next.sql > /dev/null

cat $DBNAME.create.index.sql|

xargs -i -P 8 psql -p $NEW\_PORT –h $NEW\_HOST $DBNAME -c "{}"

cat $DBNAME.create.pkey.sql|

xargs -i -P 8 psql -p $NEW\_PORT –h $NEW\_HOST $DBNAME -c "{}"

## 交换原有Cluster和新Cluster端口

如果是是同一套硬件的迁移升级，应有此操作。

## 执行新库的ANALYZE操作

cat ~/gpAdminLogs/gpdbtransfer\_$DATE.log|grep SUCCESS|awk '{print "analyze "$2}'|xargs -i -P 8 psql -p $NEW\_PORT –h $NEW\_HOST $DBNAME -c "{}"

# 实用案例

在增加Content验证之前，已经在浦发和德邦的集群迁移扩容升级操作中使用并验证。Content验证用于确保src-mapfile的正确定，加入使用时，故意写错src-mapfile文件，将可能导致获取数据的错误，而不熟悉的使用者可能会犯这样的错误，目前版本，在开始数据传输之前，会先做一个Content的测试，根据src-mapfile生成的源端集群信息中会包含Content信息，在做Test时会验证根据IP|hostname和PORT获取的Content是否与其一致，如果不一致，就报错退出。

德邦做了2次，第一次是22个节点扩展到34个，由于索引和主键的影响，30T压缩数据耗时5小时，其集群每个机器仅有2个Primary实例，运行的并发为16。由于新硬件性能问题，第二期从34节点缩回22节点，在消除索引和主键的影响后，30T压缩数据传输时间为4小时。通过观察，新集群涉及所机器的双万兆网卡网速长时间达到1.5GB/S。

浦发历史库新集群数据迁移，从15节点迁移一个10T压缩数据到一个9节点集群。由于环境仅有单万兆带宽，考虑性能影响，每节点4个Primary实例，仅使用6个并发，不过网卡速度已经很高，耗时9小时。