LightDB数据库运维手册

1 前言

本文档为恒生电子企业级数据库LightDB日常运维手册，主要介绍日常运维常用操作的指南。

2 LightDB单机

2.1 GUI安装界面为什么弹不出来？是否支持命令行安装模式?

GUI安装界面弹不出来，一般来说有两种原因：

- Linux系统未安装GUI程序所需的依赖包
- Linux系统未正确设置DISPLAY环境变量，或者Windows未正确运行Xmanager - Passive

如果无法满足上述条件，可以使用命令行安装模式，LightDB支持命令行安装模式，且与GUI安装相比仅在安装向导上有所差异，其余并无不同。

2.2 查看LightDB安装目录、实例目录、归档目录

```
ls $PGHOME           # 查看安装目录
ls $PGDATA           # 查看实例目录
ls $PGHOME/archive   # 查看归档目录
```

2.3 LightDB包含哪些日志?

数据库日志，位于$PGDATA/log目录中。

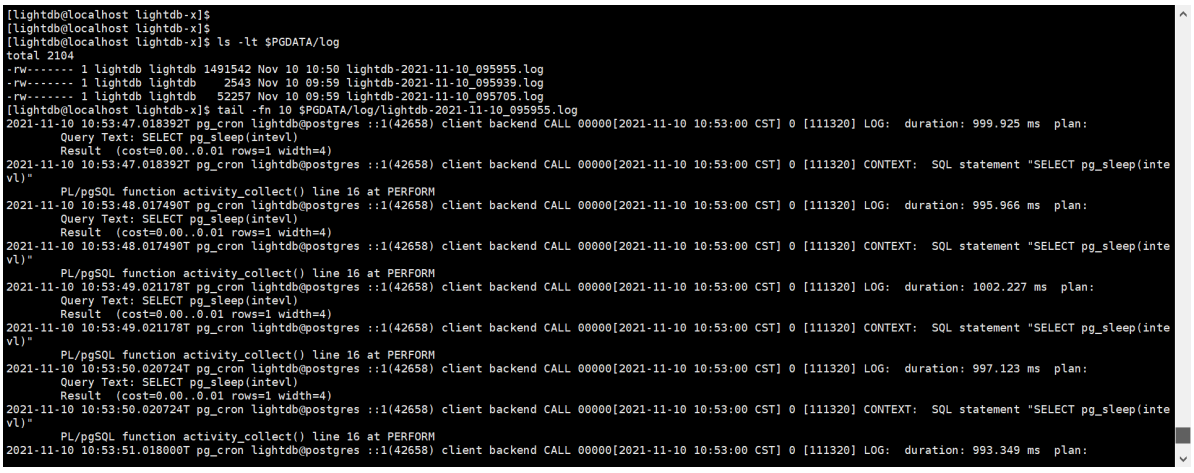
ltcluster日志，位于$PGHOME/etc/ltcluster/ltcluster.log，仅高可用版本有。

keepalived日志，位于$PGHOME/etc/keepalived/keepalived_lightdb.log，仅高可用版本且需启用keepalived。

2.4 查看数据库最新日志

LightDB数据库日志路径为`$PGDATA/log/`，日志文件命名格式为`lightdb-yyyy-mm-dd_hhmmss.log`，可以此找到最新的日志文件，然后用`tail`命令循环查看指定行数的最新日志内容，如下图所示。

```
tail -fn 10 lightdb-yyyy-mm-dd_hhmmss.log
```



2.5 查看数据库日志中的错误信息

LightDB日志中的错误信息包含`ERROR`或`FATAL`标签，可以此为关键词从日志文件中过滤错误行。

```
cat lightdb-yyyy-mm-dd_hhmmss.log | grep -E 'ERROR|FATAL'          # 单次查看当前错误日志
tail -fn 10 lightdb-yyyy-mm-dd_hhmmss.log | grep -E 'ERROR|FATAL'  # 实时监控最新错误日志
```

2.6 查看是否开启了慢日志，开启与关闭慢日志

在LightDB中慢日志配置参数有两处：数据库自身和auto_explain插件，使用`show`可以查看这两个参数。

```
show log_min_duration_statement;     -- 数据库慢日志，默认值-1
show auto_explain.log_min_duration; -- auto_explain慢日志，默认值100ms
```

数据库慢日志仅记录SQL，auto_explain慢日志同时记录SQL和执行计划，二者参数值的含义完全相同：

- -1表示关闭慢日志
- 0表示启用慢日志，且记录所有SQL
- 大于0（如100ms、1s）表示启用慢日志，且仅记录`elapsed time`大于等于该时间的SQL

在LightDB中，`log_min_duration_statement`默认值为-1，`auto_explain.log_min_duration`默认值为100ms，若在`postgresql.conf`中修改了这两个参数，不用重启数据库，仅需`reload`重新加载即可生效。

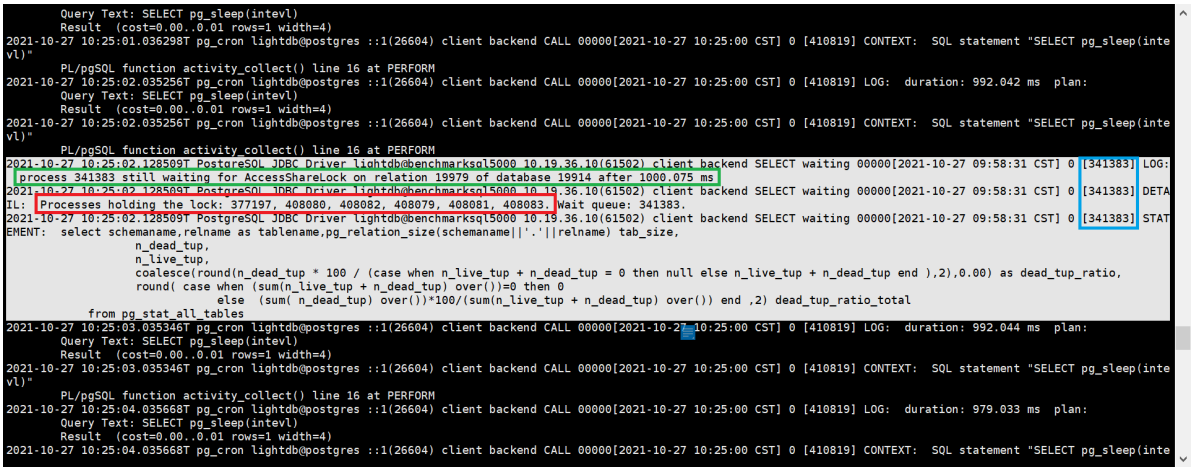
```
lt_ctl -D $PGDATA reload
```

2.7 查看锁表、阻塞者、阻塞者正在执行的SQL

该语句可以查出当前数据库中的所有锁，注意是当前数据库，不是整个实例。虽然pg_locks本身是实例级的，但是pg_class是数据库级的，所以关联之后，其他数据库的锁会查询不到。

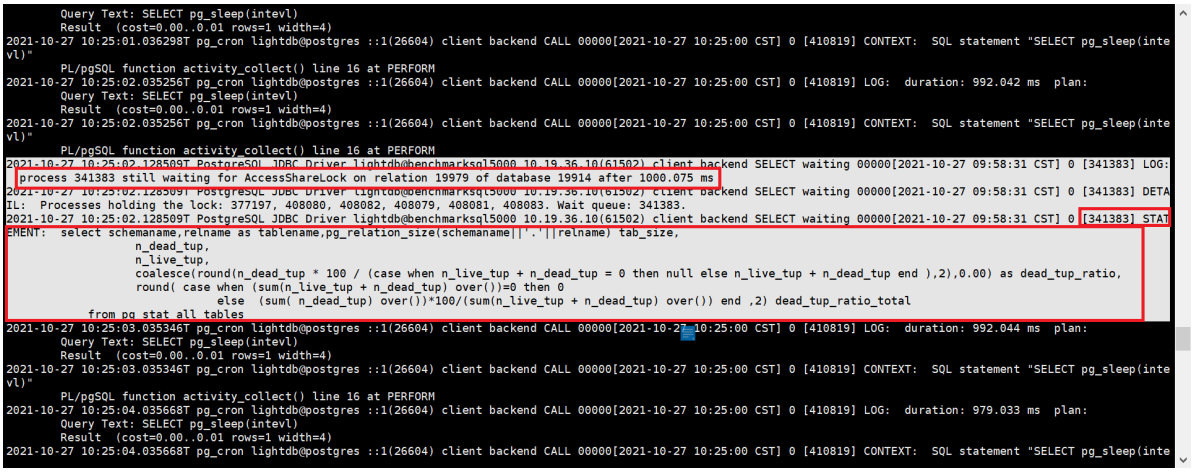
```
--查询当前数据库中的所有锁
SELECT d.datname, c.relname, c.reltype, a.*
FROM pg_catalog.pg_locks a, pg_catalog.pg_database d, pg_catalog.pg_class c
WHERE d.oid = a.database AND c.oid = a.relation;
```

对于长时间的锁监控，可以查看LightDB数据库日志，里面记录了阻塞者的PID，如图中红圈所示，顺着PID向前查找蓝圈位置值（这个值代表当前日志行对应的进程ID）等于PID的日志行，就可以找到阻塞者正在执行的SQL。



2.8 查看当前正在执行的SQL是否被阻塞了

可以查看LightDB数据库日志，看是否有`"process pid still waiting for xxxLock ......"`的字样，如果有的话，顺着pid在上下文查找，就可以找到process pid对应的SQL。



2.9 查看安装了哪些extension

- 查看所有可用的extension

```
select * from pg_available_extensions;
```

- 查看当前启用的extension

```
select * from pg_extension;
```

2.10 查看按大小排序的前20张表

```
--查出按大小（table_size + index_size）排序的前20张表，并分离table_size和index_size
SELECT
    table_name,
    pg_size_pretty(table_size) AS table_size,
    pg_size_pretty(index_size) AS index_size,
    pg_size_pretty(total_size) AS total_size
FROM (
    SELECT
        table_name,
        pg_table_size(table_name) AS table_size,
        pg_indexes_size(table_name) AS index_size,
```

```
        pg_total_relation_size(table_name) AS total_size
    FROM (
        SELECT table_schema || '.' || table_name AS table_name
        FROM information_schema.tables
    ) AS all_tables
    ORDER BY total_size DESC
) AS pretty_sizes
LIMIT 20;
```

2.11 查看LightDB当前的整体负载

查看LightDB当前整体负载，可以简单地使用top命令查看CPU利用率、内存使用情况、IO等指标信息，也可以使用LightDB EM来实时监控LightDB与服务器主机的各项指标。

2.12 查看LightDB的生效配置，修改会话配置、全局配置

可以用show语句查看LightDB当前的生效配置，show语句有以下几种用法：

```
SHOW name;    --查看指定的para配置参数
SHOW ALL;     --查看所有配置参数
SHOW name%;   --查看前缀为name的配置参数
SHOW %name;   --查看后缀为name的配置参数
SHOW %name%; --查看名字中间包含name的配置参数
```

修改配置参数有两种级别：会话级和全局级。

```
--会话级修改，并非所有参数都支持会话级修改
SET [ SESSION | LOCAL ] configuration_parameter { TO | = } { value | 'value' |
DEFAULT };

--全局修改有两种方法：一是修改postgresql.conf，二是使用下面的SQL语句，然后按要求reload或
restart生效
ALTER SYSTEM SET configuration_parameter { TO | = } { value | 'value' | DEFAULT
};
```

2.13 什么是vacuum？为什么要执行vacuum？怎么确定vacuum是否成功?

vacuum用于清理数据库表中的dead tuples，因为LightDB MVCC不使用undo日志，而是将update、delete修改或删除前的记录保留在表中，并打上一个标记，对于update还会插入一条更新后的新纪录，带有这种标记的tuple叫做dead tuple，也就是死元组。

当执行过checkpoint之后，之前的死元组就没有用了，vacuum就是用来清除这些无用的死元组的，如果长时间不进行vacuum，表中的死元组就会堆积的越来越多，导致表膨胀。

vacuum语句基本用法有两种，一种是直接执行vacuum，另一种是vcuum tablename，前者对当前database中的所有表进行清理，后者仅对指定的表进行清理，执行成功时，客户端会返回一行VACUUM信息。

2.14 查看最近的检查点执行时间

```
lt_controldata $PGDATA | grep "Time of latest checkpoint:"
```

2.15 怎么查看checkpoint执行频率？怎么查看auto vacuum频率？

```
show checkpoint_timeout;  --查看checkpoint频率
show autovacuum_naptime;  --查看autovacuum频率
```

2.16 pg_wal目录过大，怎么确定是否可以删除？如何删除？

先使用`lt_controldata`获得`Latest checkpoint's REDO WAL file`，如下所示。

```
lt_controldata $PGDATA | grep "Latest checkpoint's REDO WAL file:"
```

```
[lightdb@localhost ~]$ lt_controldata $PGDATA | grep "Latest checkpoint's REDO WAL file:"
Latest checkpoint's REDO WAL file:    000000010000000000000002
[lightdb@localhost ~]$
```

`Latest checkpoint's REDO WAL file`之前的WAL文件（包括已归档和未归档）都可以删除.

```
lt_archivecleanup -d $PGDATA/pg_wal last_checkpoint_redo_wal_file   # 删除未归档的WAL文件
lt_archivecleanup -d $PGHOME/archive last_checkpoint_redo_wal_file  # 删除已归档的WAL文件
```

```
[lightdb@localhost ~]$ lt_archivecleanup -d $PGDATA/pg_wal 000000010000000000000002
lt_archivecleanup: keeping WAL file "/home/lightdb/stage/lightdb-x/data/defaultCluster/pg_wal/000000010000000000000002" and later
[lightdb@localhost ~]$
```

2.17 查看LightDB启动时间

```
select * from pg_postmaster_start_time();
```

2.18 查看当前事务号

```
select * from pg_current_xact_id();
```

2.19 查看LightDB实例概要信息

https://www.hs.net/lightdb/docs/html/functions-info.html

```
pg_control_checkpoint(), pg_control_init(), pg_control_system(),
pg_control_recovery()
```

2.20 复制管理功能

https://www.hs.net/lightdb/docs/html/functions-admin.html#FUNCTIONS-ADMIN-BACKUP

2.21 其他管理功能函数

https://www.hs.net/lightdb/docs/html/functions-admin.html

2.22 数据库迁移注意事项

2.23 高可用归档清理

高可用归档清理通过配置`lightdb_archive_dir`（归档目录） 和`lightdb_archive_retention_size`（归档目录中Latest checkpoint's REDO WAL file 之前的文件保留数，建议配置为10以上，具体根据磁盘空间和主备间延迟配置，尽可能大）使用。

如：Latest checkpoint's REDO WAL file 为000000010000000100000049，lightdb_archive_retention_size配为10,则清理小于 000000010000000100000039 的wal文件。

2.24 日志清理

日志清理通过配置`lightdb_log_retention_age`来清理，单位为分钟（可配置为3d，内部会转为分钟）。

如：配置`lightdb_log_retention_age=7d`,则只保留7天的日志，在切换新文件时清理旧文件，根据文件的最新更新时间来清理。

3 LightDB高可用

3.1 查看LightDB是否高可用、集群信息、主从节点

如果是单机版，则没有ltcluster库，可使用命令`ltsql ltcluster`尝试连接`ltcluster`库来确认，预期提示数据库不存在。单机版也不会有`$PGHOME/etc/ltcluster/ltcluster.conf`这个配置文件。

如果是高可用部署，使用主节点或从节点运行下面的命令查看集群节点信息：

```
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf cluster show
```

示例结果：

```
 ID | Name    | Role    | Status    | Upstream | Location | Priority | Timeline |
Connection string
----+---------+---------+-----------+----------+----------+----------+----------
+-----------------------------------
 1  | node199 | primary | * running |          | default  | 100  | 1 |
host=node199 port=5432 user=ltcluster dbname=ltcluster connect_timeout=2
 2  | node193 | standby |   running | node199  | default  | 100  | 1 |
host=node199 port=5432 user=ltcluster dbname=ltcluster connect_timeout=2
```

也可以使用LightDB-EM查看是单机部署还是高可用部署。

3.2 判断集群健康状态

在主节点或从节点运行命令`ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf cluster show`展示的信息中没有`WARNING`； `Status`和`Upstream`字段没有出现`?`和`!`符号。

```
   ID | Name     | Role    | Status    | Upstream| Location | Priority | Timeline
| Connection string
   ----+---------+---------+-----------+---------+----------+----------+--------
--+----------------------------------
    1  | node199 | primary | * running |         | default  | 100      | 1
|
    2  | node193 | standby |   running | node199 | default  | 100      | 1
|
```

在各个节点运行命令`ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf node check`展示的各个检查项的均为`OK`。

示例结果：

```
Node "node193":
Server role: OK (node is standby)
Replication lag: OK (0 seconds)
WAL archiving: OK (0 pending archive ready files)
Upstream connection: OK (node "node193" (ID: 2) is attached to expected upstream
node "node199" (ID: 1))
Downstream servers: OK (this node has no downstream nodes)
Replication slots: OK (node has no physical replication slots)
Missing physical replication slots: OK (node has no missing physical replication
slots)
Configured data directory: OK (configured "data_directory" is "/data1/data5432")
```

3.3 查看集群事件

在排查集群问题，或监控集群事件时，除了查看$PGHOME/etc/ltcluster/ltcluster.log，ltcluster在events表中记录了更清晰有效的信息。

可运行`ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf cluster events`查看集群事件，最新的事件排在最上面，示例结果如下：

```
 Node ID | Name    | Event           | OK | Timestamp           | Details


----+---------+---------------------+----+---------------------+--------------
----------------------------
 1  | node199 | child_node_reconnect | t  | 2021-11-22 21:06:58 | standby node
"node193" (ID: 2) has reconnected after 1303 seconds
 1  | node199 | child_node_reconnect | t  | 2021-11-22 21:06:58 | standby node
"node193" (ID: 2) has reconnected after 1303 seconds
 2  | node193 | standby_register     | t  | 2021-11-22 21:06:55 | standby
registration succeeded; upstream node ID is 1

 2  | node193 | standby_recovery     | t  | 2021-11-22 21:06:42 | reconnected to
local node "node193" (ID: 2), marking active
 2  | node193 | standby_clone        | t  | 2021-11-22 21:05:44 | cloned from
host "node199", port 5432; backup method: lt_basebackup; --force: N
```

上述命令实际读取了ltcluster.events这张表，所以也可以通过SQL直接查询：

```
$ ltsql ltcluster # 连接ltcluster库
ltsql (13.3-21.2)
```

```
Type "help" for help.

ltcluster=# select * from ltcluster.events ;
 node_id |           event            | successful |        event_timestamp
|                                                 details

---------+----------------------------+------------+----------------------------
---+----------------------------
       1 | cluster_created            | t          | 2021-11-21
22:17:20.421939+08 |
       1 | primary_register           | t          | 2021-11-21
22:17:20.423033+08 |
       2 | standby_clone              | t          | 2021-11-21
22:27:39.853675+08 | cloned from host "node199", port 5432; backup method:
lt_basebackup; --force: N
       2 | standby_register           | t          | 2021-11-21
22:31:49.270459+08 | standby registration succeeded; upstream node ID is 1
       1 | child_node_reconnect       | t          | 2021-11-21
22:31:55.155461+08 | standby node "node193" (ID: 2) has reconnected after 440552
seconds
       1 | child_node_disconnect      | t          | 2021-11-21
22:35:49.769979+08 | standby node "node192" (ID: 2) has disconnected
```

3.4 查看主从同步模式与延时

可在主节点执行`select application_name, client_addr, client_hostname, sync_state from pg_stat_replication;`得到各个节点的同步模式信息。

```
postgres=# select application_name, client_addr, client_hostname, sync_state
from pg_stat_replication;
       application_name       |  client_addr  | client_hostname | sync_state
------------------------------+---------------+-----------------+------------
 172.16.56.103-defaultcluster | 172.16.56.103 |                 | sync
```

在从节点执行`select standby_name,replication_lag,replication_time_lag,apply_lag from ltcluster.replication_status ;`，其中replication_time_lag表示落后的时间，replication_lag，apply_lag分别表示复制、应用WAL的落后数据大小。

```
$ ltsql ltcluster
ltsql (13.3-21.3)
Type "help" for help.

ltcluster=# select standby_name,replication_lag,replication_time_lag,apply_lag
from ltcluster.replication_status ;
         standby_name         | replication_lag | replication_time_lag |
apply_lag
------------------------------+-----------------+----------------------+---------
--
 172.16.56.103-defaultcluster | 144 bytes       | 00:00:01.647838      | 0 bytes
(1 row)
```

可以从表ltcluster.monitoring_history中获取各个时间段的延时：

```
ltcluster=# select * from ltcluster.monitoring_history order by
last_monitor_time limit 10 ;
 primary_node_id | standby_node_id |       last_monitor_time       |
last_apply_time        | last_wal_primary_location | last_wal_standby_location |
replication_lag | apply_lag
-----------------+-----------------+-------------------------------+-------------
-----------------+---------------------------+---------------------------+------
-----------+-----------
               1 |               2 | 2021-12-21 16:57:48.537956+08 | 2021-12-21
16:57:48.52187+08  | 0/60012308                | 0/60012308                |
          0 |         0
               1 |               2 | 2021-12-21 16:57:50.561467+08 | 2021-12-21
16:57:50.294248+08 | 0/6001C540                | 0/6001C540                |
          0 |         0
               1 |               2 | 2021-12-21 16:57:52.577251+08 | 2021-12-21
16:57:52.55301+08  | 0/6001F1B0                | 0/6001F1B0                |
          0 |         0
               1 |               2 | 2021-12-21 16:57:54.590478+08 | 2021-12-21
16:57:53.66048+08  | 0/60020878                | 0/60020878                |
          0 |         0
               1 |               2 | 2021-12-21 16:57:56.6056+08   | 2021-12-21
16:57:55.944149+08 | 0/60023598                | 0/60023598                |
          0 |         0
               1 |               2 | 2021-12-21 16:57:58.618428+08 | 2021-12-21
16:57:58.19143+08  | 0/600278E0                | 0/600278E0                |
          0 |         0
               1 |               2 | 2021-12-21 16:58:00.638982+08 | 2021-12-21
16:58:00.615274+08 | 0/600C3150                | 0/600C3150                |
          0 |         0
               1 |               2 | 2021-12-21 16:58:02.686736+08 | 2021-12-21
16:58:01.813462+08 | 0/6023B0A8                | 0/6023B0A8                |
          0 |         0
               1 |               2 | 2021-12-21 16:58:04.712443+08 | 2021-12-21
16:58:04.117613+08 | 0/6023FA10                | 0/6023FA10                |
          0 |         0
               1 |               2 | 2021-12-21 16:58:06.730236+08 | 2021-12-21
16:58:06.310637+08 | 0/60242C48                | 0/60242C48                |
          0 |         0
```

也可以从LightDB-EM监控页面查看延时。

3.5 集群复制级别

不同的业务场景对数据库主备一致性有不同的要求。一致性越高对性能影响越大。用户可通过配置`synchronous_commit`来达到不同级别的一致性。

```
# 同步模式，在主节点修改
synchronous_commit = 'on'
synchronous_standby_names = '*'

# 异步模式，在主节点修改
synchronous_commit = 'local'
synchronous_standby_names = ''

# 修改后，主节点调用reload生效
lt_ctl -D $PGDATA reload
```

下表概括了 `synchronous_commit` 不同设置对应不同的一致性级别：

synchronous_commit 设置	本地提交持久化	备库提交持久化(数据库崩溃)	备库提交持久化(OS崩溃)	备库查询一致
remote_apply	是	是	是	是
on	是	是	是	
remote_write	是	是		
local	是			
off				

更详细的 `synchronous_commit` 及 `synchronous_standby_names` 请参考LightDB官方文档。

3.6 主备切换

在需要维护primary节点时，可做switchover，互换主从角色。switchover操作的内部执行比较复杂，非必要尽量不要执行。

具体操作时，请严格按照下面步骤执行:

1. 主备之间需要有SSH免密访问(LightDB安装时有要求)
2. 尽量减少应用程序的访问
3. 检查主备间的网络状况是否良好，确保有良好的网络
4. 确保当前主备之间没有明显的复制延迟，尤其在集群复制级别较低的情况下(参考 3.4节，3.5节)
5. 检查等待归档的文件是否有积压, 可通过下面的命令来检查

   ```
   ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf node check --archive-ready
   ```

   确保输出是： `OK (0 pending archive ready files)`。

   如果是其他输出，则应检查归档进程是否正常。如果归档正常，则可以等待一会儿再试下。
6. 使用dry-run试运行switchover命令，查看输出是否有警告和错误

   ```
   ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf standby switchover --
   siblings-follow --dry-run
   ```

   如果最后一行信息为：prererequisites for executing STANDBY SWITCHOVER are met，则表示成功
7. **在备机上正式执行switchover**, 打开最详细的日志级别(注意保存输出日志)

   ```
   ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf standby switchover \
       --log-level=DEBUG --verbose --siblings-follow
   ```
8. 在各节点上查看集群状态，确认各节点执行结果中primary和standby角色确实已互换

   ```
   ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service status
   ```

   输出要确保没有警告和错误信息

9. 查看paused状态是否为no

   确认Paused列为no (如果switchover过程出现异常，经过处理后，switchover成功, 此时在这一步可能处于yes)

   如果为yes，则执行

   ```
   ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service unpause
   ```

10. 如果使用同步模式，则需要把新主改成同步模式(和旧主一样)，新备改成local模式(参考3.5节)
11. 确认VIP是否切换到新的主机上(参考3.11节)
12. 确认应用程序是否可以正常访问数据库

3.7 故障恢复，主节点重新加入作为从节点

当主库发生故障（如宕机）failover后，备库会自动提升为新主库，以确保集群继续可用。

此后，如果原主库故障修复后启动，keepalived会主动停止原主库，导致原主库一启动就被停止，避免双主同时运行。
若需要想启动原主库，应使用rejoin让原主库恢复成为新备库，然后再执行一次主备切换，恢复到最初的主备关系。

在原主库上rejoin的步骤如下：

```
# 1. 确认LightDB已停止

# 2. 确认ltclusterd是否启动，若不存在则启动它
ps aux | grep ltcluster
ltclusterd -d -f $PGHOME/etc/ltcluster/ltcluster.conf -p
$PGHOME/etc/ltcluster/ltclusterd.pid

# 3. rejoin试运行，将new_primary_host替换为原备，也就是新主的host
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf node rejoin -d
'host=new_primary_host dbname=ltcluster user=ltcluster' --verbose --force-rewind
--dry-run

# 4. 确认试运行成功，进入下一步

# 5. 正式执行rejoin，new_primary_host同上
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf node rejoin -d
'host=new_primary_host dbname=ltcluster user=ltcluster' --verbose --force-rewind

# 6. 按本文档5.4.2.3所述，在新备上执行主备切换，恢复到最初的主备关系

# 7. 确认keepalived是否启动，若不存在则启动它，启动方法请参照本文档5.3
ps aux | grep keepalived
```

3.8 什么时候会rejoin失败、如何确定肯定无法rejoin了？无法rejoin的节点如何重新加入？

主节点修复后，如果能够正常rejoin回来固然好，但实际更多的时候是rejoin失败，这通常发生在failover后，备库提升为主库，然后经过了一段时间的数据写入，之后原主rejoin（立刻rejoin几乎不会有问题）。

在这种情况下，可以在原主库上使用clone来重新初始化实例，步骤如下：

```
# 1. 确认备库LightDB已停止

# 2. 确认ltclusterd是否启动，若不存在则启动它
ps aux | grep ltcluster
ltclusterd -d -f $PGHOME/etc/ltcluster/ltcluster.conf -p
$PGHOME/etc/ltcluster/ltclusterd.pid

# 3. 清空备库实例目录($PGDATA)下的内容（若有需要，清空前可先备份）

# 4. 清空备库归档目录($PGHOME/archive)下的内容（若有需要，清空前可先备份）

# 5. clone试运行，将new_primary_host替换为原备，也就是新主的host
ltcluster -h new_primary_host -U ltcluster -d ltcluster -f
$PGHOME/etc/ltcluster/ltcluster.conf standby clone --dry-run

# 6. 确认试运行结果显示all prerequisites for "standby clone" are met

# 7. clone实例目录，new_primary_host同上，
# 如果库比较大，这里执行时间会很长，具体执行时间取决于网络情况和数据量大小
# 在我们的测试中800G左右的库大概需要一个小时
# 我们建议采用异步的方式执行这个命令，以避免执行过程中终端意外关闭的影响。
# 另外我们开启了最详细的日志级别，以便协助定位问题
nohup ltcluster -h new_primary_host -U ltcluster -d ltcluster \
      -f $PGHOME/etc/ltcluster/ltcluster.conf \
      standby clone -F \
      --log-level=DEBUG --verbose \
      >standby_clone.log  2>&1 &

# 8. 把主库的归档目录下的所有文件复制到备库的归档目录中($PGHOME/archive)

# 9. 启动数据库
lt_ctl -D $PGDATA start

# 10. 重新注册为standby
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf standby register -F

# 11. 在新备上执行主备切换，恢复到最初的主备关系

# 12. 确认keepalived是否启动，若不存在则启动它，启动方法请参照本文档5.3
ps aux | grep keepalived
```

3.9 什么是timeline，timeline什么时候变化？如何查看当前的timeline id?

timeline可以认为是数据库wal的分支（类比版本管理系统，比如svn）。

当进行一次恢复，或发生主备切换，会生成一个timeline。每个timeline有一个id，从1开始编号。当生成一个新的timeline时，它的wal是独立的，不会覆盖其它timeline的wal，这就保证了可以多次来回恢复。如果没有timeline，即恢复后wal覆盖写，则只能一直往“以前”恢复。

可以查看pg_wal中的history文件，来确定当前有几个timeline、各自创建时的LSN、创建的原因，如

```
$ cat ./data/defaultcluster/pg_wal/00000004.history
1   16/F20000A0 no recovery target specified

2   16/F50000A0 no recovery target specified

3   16/F60000A0 no recovery target specified
```

序号最大的history文件即是当前timeline id。

可以通过sql查看当前timeline id: `ltsql "dbname=postgres replication=database" -c "IDENTIFY_SYSTEM"`; 或在主库执行 `select substring(pg_walfile_name(pg_current_wal_lsn()), 1, 8);`

高可用命令 `ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf cluster show` 获取的timeline是当前最近做checkpoint的timeline，可能不是最新的timeline。

3.10 当出现双主时如何处理

如果出现双主，把老主停掉，重新加入集群作为standby。参考 `node rejoin` 章节，如果rejoin失败，老主通过 `standby clone` 重新加入集群。

3.11 如何查看VIP当前在哪个节点

使用命令 `ip a` 可看到vip是否在当前节点，比如

```
$ ip a | grep 251
    inet 10.19.36.251/32 scope global enp2s0f0
```

如果grep没有匹配行，则vip不在当前节点。

可以在keepalived.conf中查看vip配置，比如

```
$ cat $PGHOME/etc/keepalived/keepalived.conf
...
    interface enp2s0f0
...
    virtual_ipaddress {
        10.19.36.251
    }
...
```

3.12 如何触发VIP漂移

在以下场景会触发VIP漂移:

- 主库崩溃、意外停止，导致自动主从切换 (failover)
- 手动进行主从切换 (switchover)

3.13 为什么会出现VIP同时在两个节点?

如果主从之间网络出现问题，从节点可能误判主节点故障，把自己提升，这时会出现两个VIP。

建议集群中加入witness节点，避免网络问题引起主从切换或从节点自动切主。

3.14 重启主库

主库因修改数据库参数或其他原因需要重启，可以按以下步骤操作。(**注意: 重启期间数据库不提供服务**)

1. **先停止从库的**`keepalived`**（重要）**，在root用户下执行以下命令

```
# 1. 获得备库keepalived进程pid
cat /var/run/keepalived.pid

# 2. 杀死keepalived进程
kill keepalived_pid

# 3. 确认keepalived进程确实已不存在
ps aux | grep keepalived
```

2. 主库重启，需要在lightdb用户下执行

```
# 1. 暂停ltclusterd，防止自动failover
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service pause

# 2. 查看集群状态，确认primary的Paused?状态为yes
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service status

# 3. 先断开所有连接到数据库的客户端和应用程序（否则数据库将stop failed），然后停止主库
lt_ctl -D $PGDATA stop  # 默认会回滚所有未断开的连接

# 如果有连接存在导致stop failed，则可以尝试使用
lt_ctl -D $PGDATA stop -m smart

# 如果仍然stop failed，且因条件限制无法或不希望断开所有客户端连接，则可以使用-m immediate强制停止数据库,此方式下没有回滚连接，即强制断开、强制停止，没有完全shutdown，会导致在启动时recovery
lt_ctl -D $PGDATA stop -m immediate

# 4. 等待数据库停止成功，确认步骤3执行结果中出现server stopped信息

# 5. 修改数据库参数，或做其他事情

# 6. 启动主库
lt_ctl -D $PGDATA start

# 7. 等待数据库启动成功，确认步骤6执行结果中出现server started的信息

# 8. 恢复ltclusterd
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service unpause

# 9. 查看集群状态，确认primary的Paused?状态为no
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service status
```

3. **从库重新启动**`keepalived`**（需root用户）**，启动方法请参照本文档5.3。

3.15 重启从库

备库因修改数据库参数或其他原因需要重启，可以在`lightdb`用户下按以下步骤操作。

```
# 1. 暂停ltclusterd,防止自动failover
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service pause

# 2. 查看集群状态，确认standby的Paused?字段为yes
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service status

# 3. 先断开所有连接到数据库的客户端和应用程序（否则数据库将stop failed），然后停止备库
lt_ctl -D $PGDATA stop  # 默认会回滚所有未断开的连接

# 如果有连接存在导致stop failed，则可以尝试使用
lt_ctl -D $PGDATA stop -m smart

# 如果仍然stop failed，且因条件限制无法或不希望断开所有客户端连接，则可以使用-m immediate强制停止数据库,此方式下没有回滚连接，即强制断开、强制停止，没有完全shutdown，会导致在启动时recovery
lt_ctl -D $PGDATA stop -m immediate

# 4. 等待数据库停止成功，确认步骤3执行结果中出现server stopped信息

# 5. 修改数据库参数，或做其他事情

# 6. 启动备库
lt_ctl -D $PGDATA start

# 7. 等待数据库启动成功，确认步骤6执行结果中出现server started的信息

# 8. 恢复ltclusterd
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service unpause

# 9. 确认standby的Paused?字段为no
ltcluster -f $PGHOME/etc/ltcluster/ltcluster.conf service status
```

3.16 高可用归档清理与lt_probackup备份归档清理

当同时使用高可用归档与lt_probackup备份归档时建议建立两个归档目录，归档两份，分别给高可用和备份使用，不然如果使用同一个，然后只开启备份的清理，有可能出现误删高可用所需的wal文件；只开启高可用的归档清理，可能导致误删备份所需的wal文件。

高可用归档清理参见2.23一节。

4 LightDB EM

4.1 LightDB安装EM报错Redis连接失败怎么办?

确保Redis启动正常

Redis默认安装目录$PGHOME/../../em/redis，默认端口是18331。确认Redis是否正常，通过Redis安装目录$PGHOME/../../em/redis中的redis-cli来访问服务是否正常，执行命令$PGHOME/../../em/redis/redis-cli -h Redis的IP -p 18331，成功后执行auth Redis的密码(无密码可以跳过此步)，再执行keys *既可以看到数据，以上步骤都成功则说明Redis正常，否则要通过$PGHOME/../../em/redis/redis.log查看Redis产生了哪些错误，针对性解决。

确保em访问Redis配置正常

em访问Redis的配置文件在$PGHOME/../../em/config/jrescloud.properties，对应spring.redis前缀的配置。重点关注IP(spring.redis.host)、端口(spring.redis.port)和密码(spring.redis.password)，确保配置跟第1步中Redis的配置一致。

4.2 LightDB安装EM组件启动或重启顺序

首先确保lightdb数据库正常，然后一次启动或重启以下服务：

启动Redis，如果Redis服务存在，先执行$PGHOME/../../em/scripts/redis_stop.sh，不存在跳过此步，然后执行执行命令$PGHOME/../../em/scripts/redis_start.sh。通过ps -ef|grep redis查看em的Redis进程是否存在

启动Nginx，如果Nginx服务存在，先执行$PGHOME/../../em/scripts/nginx_stop.sh，不存在跳过此步，然后执行执行命令$PGHOME/../../em/scripts/nginx_start.sh。通过ps -ef|grep nginx查看em的Nginx进程是否存在

启动Em java服务，如果Em java服务存在，先执行$PGHOME/../../em/scripts/em_stop.sh，不存在跳过此步，然后执行执行命令$PGHOME/../../em/scripts/em_start.sh。通过ps -ef|grep /em/查看em的进程是否存在

4.3 lightdb em组件配置文件和日志对应的路径

Redis配置文件路径：$PGHOME/../../em/redis.conf

Redis日志路径：$PGHOME/../../em/redis.log

Nginx配置文件路径：$PGHOME/../../em/nginx/conf/nginx.conf

Nginx日志路径：$PGHOME/../../em/nginx/logs

em java配置文件路径：$PGHOME/../../em/config jrescloud.properties log4j.properties

4.4 lightdb em启动失败常见问题

1、Swap交换区没有设置，设置方法参考《LightDB 13.3-21.2数据库安装手册.pdf》中3.11 开启Swap交换区

2、服务器内存空间不够，推荐给em java应用至少8G内存，配置详见$PGHOME/em/scripts/em_start.sh中app_Xms项的值

4.5 lightdb em java日志处理

em java日志放在$PGHOME/em/logs中，lightdb-em.log就是日志文件。日志对应的配置在$PGHOME/em/config/jrescloud.properties中对应配置项logging.config=classpath:log4j2.xml指定日志的配置文件，然后在$PGHOME/em/config/log4j.properties配置日志分割规则。目前的日志配置规则基本都满足要求，如果有特殊的要求可以参考https://blog.csdn.net/fz13768884254/article/details/81214773