# 计算节点

## 通过proxy删除db上存在库表失败

### 故障描述

Proxy上执行drop database|table命令失败后，库表的元数据在GoldenDB中已经不存在，直接drop database | table报错：

ERROR 1008(HY000):Can’t drop database ‘test’;database doesn’t exists;part of DDL may succes,please manual recovery!

### 故障分析

proxy上执行drop database|table命令失败后，库表的元数据在GoldenDB中已经不存在，直接drop database | table报错，此时是因为proxy和db的元数据不一致导致的（即proxy的元数据认为表已经不存在了，但是db元数据中还保存着）。

### 解决方法

在drop语句中加上“if exists”:

drop databse if exists test;

## 通过proxy创建db上存在的表

### 故障描述

Proxy创建db上已经存在的库表报错：

ERROR 1008(HY000):Can’t create database ‘test’;database doesn’t exists;part of DDL may succes,please manual recovery!

### 故障分析

Proxy和db的元数据不一致。

### 解决方法

在create语句中加上“if not exists”。

## Proxy无法配置proxy最大连接数

### 故障描述

通过OMM界面配置的proxy连接实例最大连接数为5000，但是实际业务跟proxy之间创建的连接数达不到那么多（即客户端无法使用5000个连接），并且proxy会出现无法进一步创建客户端连接的情况，使用dbtool -p -c -i命令报错：link failure。

### 故障分析

实际proxy能够创建的最大客户端连接数在内部受到os控制，OS，ini中的配置项max\_link\_sessions决定了本proxy所有客户端连接数（包含dbtool命令创建的连接）。

### 解决方法

修改proxy下os.ini中max\_link\_session参数，使得值大于proxy绑定的所有连接实例中“客户端最大连接数”的总和加9（proxy内部耗费9个连接）。

## Proxy由于服务器配置低导致OMM

### 故障描述

GoldenDB的组件DB与proxy合设在一台2核7G内存的X86机器上，组件正常启动后，开启业务的情况下proxy会经常发生链接中断，连接不上。发现proxy进程会小时，PM日志会报错：proxy heart lose time[3]!Begin exception deal!

重启proxy后过段时间进程还是会消失，进一步了解后，DB进程也有一定概率被强制终止。

### 故障分析

通过分析系统的dmesg日志与mysqld日志，发现系统存在资源不足的情况。Linux系统在系统资源不足的情况下，会激活OMM机制，将占用资源的进程强制终止释放资源。结合机器的配置，2核PENTIUM的CPU、7G内存，并且是与DB合设的情况下，存在抢占资源的情况。

### 解决方法

将proxy迁移至其他机器，降低该机器DB的配置，例如：

max\_connection、open\_files\_limit、table\_open\_cache配置调低后，问题解决。

## 业务运行中有一组事务发生失败

### 故障描述

业务代码报错：unable to clear batch for prepare statement:Communication link failure。经过分析业务在执行跑批时执行clear batch，链路中断了，该中断是由服务端引起的。该时间段的proxy日志已经被清理。

### 故障分析

经过分析，发现在业务失败的时刻，产生了dbproxy\_dump.log，说明当时出现了proxy的正常宕机，同时在dbstop.log和history中均发现有人执行了dbstop的证据。因此，该问题是由于运维人员不通知业务的情况下重启proxy引起的。

### 解决方法

排查问题前需要先查看各个日志生成的时间，确认出现问题时的人工操作有哪些，尽可能避免人为问题导致的业务故障。

## Table lock/unlock proxy timeout

### 故障描述

用户执行DDL语句，例如ALTER、DROP操作时，报错：[table lock/unlock] the end of notification event(2467),fail num is 1,sucess num is 1,errinfo is [proxy 2 timeout]。

### 故障分析

通过对磁盘的监控，发现proxy网元所在磁盘的I/O存在很高的avgqu-sz，该指标表示等待处理的I/O请求，当请求超过磁盘处理能力，则该值将增加。假设该值等于2，表示持续有两倍读写能力的I/O请求在等待。由于过高的磁盘IO导致proxy修改本地元数据超时，导致报错。

### 解决方法

将proxy部署到单独的磁盘上，或者避免proxy所在磁盘有大量的IO。

## 业务遇到空指针问题

### 故障描述

业务返回空指针。

### 故障分析

根据业务代码分析，proxy返回的字段导致业务产生了空指针，proxy在oracle模式下返回的字段为纯大写形式，而业务比对的时候使用字符串比对的，导致业务结果集比对不出来，产生了空指针。

### 解决方法

Proxy的模式修改为mysql模式。

# 全局一致性控制节点

## DB表入库GTID比GTM查找的GTID申请最大值大

### 故障描述

DB表数据入库的GTID的值比GTM上查找的GTID申请的最大值大。

### 故障分析

两个GoldenDB系统合并，其中迁移过来的数据的最大GTID大于现有系统的GTID。

### 解决方法

停止GTM进程，修改GTM的最大GTID。

## GTM单主异常状态INACTIVE后无法自动恢复

### 故障描述

在该版本，大部分集群会只有一个GTM的情况，如果这个GTM出现了异常，比如磁盘空间满内存不足状态被INACTIVE后，当资源被恢复后，状态无法自动恢复为ACTIVE。

### 故障分析

GTM目前的逻辑是故障后需要手动恢复，手动进行状态设置。但是针对单GTM的情况，由于没有修改状态的入口，OMM界面无法设置，dbtool也没有设置的命令，所以需要重启GTM进程。

### 解决方法

需要对GTM进程重启，重启后状态被MDS修改为ACTIVE。

## 磁盘I/O持续过高导致GTM Abnormal

### 故障描述

业务或者proxy客户端报错：ERROR(HY000):ERR:GTM Abnormal!

### 故障分析

现在通过对磁盘的监控，发现GTM网元所在的磁盘I/O存在很高的avgqu-sz，该指标表示等待处理的I/O请求，当请求超过磁盘处理能力，则该值将增加。假设该值等于2，表示持续有两倍读写能力的I/O请求在等待。

通过监控的数据可以发现，avgqu-sz的值达到了8395，从而导致GTM网元持久化增量数据时，等待写入的时间过长，从而在日志上反映出：GTM process run too too long，一直到最后的GTM Abnormal！

### 解决方法

由于GTM网元需要持续写增量数据到磁盘，请不要将GTM网元与其他网元共用磁盘。

# 数据迁移

## 重分布前后数据不一致

### 故障描述

数据重分布，原表结构含有自增主键，重分布失败，报错全量数据导入后，数据总量检验不一致。

### 故障分析

Proxy建表主键是自增主键，当重分布由hash/range/list变更为duplicate时，由于主键冲突，导致重分布新表的数据量减少。

### 解决方法

1. 集群各个Group节点DB配置自增主键配置项后，重启DB

示例：集群2个group，主DB配置如下，主备配置一致：

g1：

auto\_increment\_increment=[group数目]

auto\_increment\_offset=1

g2：

auto\_increment\_increment=[group数目]

auto\_increment\_offset=2

1. 业务使用sequence代替自增列

## 导入oracle数据，无法识别

### 故障描述

将oracle的导出的数据文件导入GoldenDB的时候发现，文件导入失败，报错信息为字符无法识别。

### 故障分析

Oracle生成的文件编码格式为GB18030，而在导入的时候，GoldenDB设置的编码格式为GBK，GBK无法兼容所有的GB18030的字符。

### 解决方法

在导入文件之前对文件进行格式转换，将格式转换为UTF-8，并通过默认的UTF-8格式进行导入，在Linux执行命令“iconv -f GB18030 -t UTF-8 data.txt -o data\_utf8.txt”进行转码。

## LDS导入无法识别时间0001-01-01

### 故障描述

LDS导入数据报错：ERROR 1292：Incorrect datetime value:’0001-01-01’ for column ‘tb\_up\_time’ at row 1。

### 故障分析

tb\_up\_time字段类型为timestamp，而timestamp默认的时间是“0000-00-00”，而“0001-01-01”对mysql来说是非法的时间格式。

timestamp的时间范围UTC时间：1970-01-01 00:00:01.000000 to 2038-01-19 03:14:07.999999

### 解决方法

修改业务，将传递给LDS的默认时间修改为“0000-00-00”。

# 存储节点

## 集群缩容失败

### 故障描述

OMM界面发起删除dbgroup操作，失败日志如下：

MetaDataServer table [test.t1] ddl include groupid[1]

### 故障分析

待删除的dbgroup上仍有ddl和数据分片到其上。

### 解决方法

1. 登录到DB主上检查是否存在数据和表结构在该节点上；
2. 可能是重分布（缩容）后，原表rename之后的备份表（与原分布方式相同）仍然在该节点上

## 运行过程DB无法生成错误日志

### 故障描述

巡检环境发现，DB在运行过程中一直不输出日志到mysqld.log文件，并且mysqld.log文件也不存在。

### 故障分析

经过分析，维护人员的清理环境的时候，直接使用rm -rf命令删除所有日志，而运行态的DB不会再打开新的文件句柄输出日志文件。

### 解决方法

重启DB，启动DB时会自动生成日志文件。C语言的日志文件在运行态都不能直接删除。如果需要清理，可以使用echo > xx.log进行清空。

## 拷贝DATA目录的DB无法启动或建立主备关系

### 故障描述

备机的data目录是从主机全量拷贝的，拷贝完成后新的DB无法正常启动或者建立复制关系。

### 故障分析

Data目录中auto.cnf保存有原设备的uuid，master.info中保存有原设备（如果是备）的复制主关系，my.cnf中保存有原设备的serverid、ip等信息，均可能导致以上问题。

### 解决方法

把上述字段修改为原设备配置不同的即可（对于auto.cnf文件可以删除，重启DB的时候会自动生成的）。

## Slave sql回放线程很慢

### 故障描述

Slave回放线程很慢，导致备机严重落后于主机。

### 故障分析

1. 大事务
2. 回放的表没有索引

### 解决方法

1. 大事务需要慎用，可能会出现备机追不上主机的情况。如果多个大事务同时操作，可能会导致主备复制关系异常，甚至DBAgent无法与DB进行通信而导致集群异常
2. 添加索引或过滤该表不进行复制

## 备机IO线程出现1236报错

### 故障描述

备机复制中断，show slave status出现Last\_IO\_Error：Got fatal error 1236 from master binary log: ’The slave is connecting using CHANGE MASTER TO MASTER\_AUTO\_POSITION=1, but the master has purged binary logs containing GTIDs that the slave requires.’

### 故障分析

在主库上手动执行清除二进制日志文件，包括手动删除binlog以及reset master等操作

### 解决方法

1. 在主库上执行以下命令，查询gtid\_purged，记录下该值

show global variables like ‘%gtid%’\G

1. 在从库上执行以下命令，查询已经执行过的gtid即gtid\_executed，记录下主库的值，本机的不需要

show global variables like ‘%gtid%’\G

1. 在从库上执行以下命令停止同步线程及重置同步相关信息

stop slave;

reset slave;

reset master;

1. 在从库上设置gtid\_purged

该值有两个来源，一个在主库上查询gtid\_purged，二是在从库上查询的已经执行过的gtid\_executed值（本机的就不需要主库上GTID）

注意：一定记得加上从库上已经执行过的GTID，若设置了主库上的gtid\_purged，此时从库会重新拉取主库上所有的二进制日志文件，同步过程会出现其他错误，导致同步无法进行

set @@global.gtid\_purged=’\*\*\*\*:\*\*-\*\*,\*\*\*\*:\*\*-\*\*’;

注意：设置gtid\_purged值时，gtid\_executed值必须为空否则报错，该值清空的方法就是reset master命令。

执行完再次查看相关性信息

1. 重新开启同步

change master to master\_host=’’,master\_port=3306,

master\_user=’repl’,master\_password=’’,master\_auto\_position=1;

start slave;

当从库追赶上主库，此时测试主从数据是否一致，测试结果一切正常

## Join执行走全表扫描，未走主键匹配

### 故障描述

Join字段为主键字段进行匹配时，未走主键匹配而是全表扫描。

### 故障分析

mysql会对语句进行改写，而改写后的语句则可以在explain的show warnings中展示。

通过执行show warnings获取arcard\_auth取改写后的语句发现，在全表扫描的语句执行主键关联时多了一个“convert (‘1111’=`starcard\_autu`.`accl`.`ami\_account\_manage\_id` using utf8mb4)”，原来是做了主键的格式转换。

做全表扫描的关联表，其中一个表主键是utf8格式，另一个表的主键格式为utf8mb4，导致主键在做=关联的时候做了格式转换。

### 解决方法

将表重建，统一主键字段为UTF8或者utf8mb4

## MySQL网络超时错误

### 故障描述

mysql server网络超时。

### 故障分析

mysqld server和网络超时相关的参数有：

interactive\_timeout、wait\_timeout、net\_read\_timeout、net\_write\_timeout、connect\_timeout

wait\_timeout是给读请求用的，这个时候，连接是空闲的，等待用户的请求。等读完用户的请求包后，连接就变成active的，在调用dispatch\_command执行SQL前，把超时设置回net\_read\_timeout，之后在执行SQL请求过程中，server和client基本不会有网络交互，所以这个超时基本用不上。

一个特殊的情况是LOAD DATA LOCAL FILE命令，server在执行过程中，需要和client再做网络交互。

interactive\_timeout是给交互模式的客户端使用的，比如我们常用的mysql client工具，这个是在认证过程中设置的，如果设置的能力位上设置了CLIENT\_INTERACTIVE，会用interactive\_timeout的值覆盖wait\_timeout的值。而一般情况下，我们应用在建立连接时，是不会设置这个能力位的。

net\_write\_timeout对应写超时，在连接认证完成后，server和client交互过程中写超时一直是不变的。

connect\_timeout是给连接认证过程用的，读和写都用这个值，认证完成后，读和写分别设置为net\_read\_timeout和net\_write\_timeout。

### 解决方法

如果是认证过程中超时，不管是读还是写，都是connect\_timeout。

对于读网络超时，一般是wait\_timeout/interactive\_timeout，基本不会是net\_read\_timeout（特例是业务用LOAD DATA LOCAL FILE）；对于写网络超时，都是net\_write\_timeout。

在遇到超时情况时，可以根据这些原则判断对哪个参数做调整：

比如下面这种情况：

[Warning] Abort connection 6 to db:’unconnected’ user:’root’ host:’localhost’ (Got timeout reading communication packets)

很可能需要调整的wait\_timeout/interactive\_timeout。

[Warning] Aborted connection 12 to db:’test’ user:’root’ host”’localhost’(Got timeout writing communication packets)

需要调整net\_write\_timeout。

需要注意的是，MySQL的关于网络的错误，除了超时意外都认为是error，没有做进一步的细分，比如可能会看到下面这种日志，有可能是客户端异常退出了，也有可能是网络链路异常。

[Warning] Aborted connection 8 to db:’unconnected’ user:’root’ host:’localhost’ (Got an error reading communication packets)

[Warning] Aborted connection 13 to db:’test’ user:’root’ host:’localhost’(Got an error writing communication packets)

## packet for query is too large

### 故障描述

客户端返回错误日志packet for query is too large(19063787>1677216)

### 故障分析

MySQL根据配置文件会限制Server接受的数据包大小。有时候，插入、更新或者查询时数据包的大小，会受max\_allowed\_packet参数限制，导致操作失败。默认设置为16M。

### 解决方法

1. 在mysql控制台输入以下命令，设置max\_allowed\_packet为32M

set global max\_allowed\_packet = 33554432；

该方法仅对设置后的新链接生效（当前连接不生效）。

1. 修改$HOME/etc/my.cnf，设置max\_allowed\_packet=32M，然后重启生效

## show tables可以查看表名，但查询该表发现表结构不存在

### 故障描述

Show tables可以查看到表名，但是查询该表发现表结构不存在。

### 故障分析

在大小写敏感的时候，创建了大写的表，然后修改成大小写不敏感后，mysql对所有的表都修改成小写，对于大写表名就不识别了。

### 解决方法

1. 修改成大小写不敏感模式，然后重启mysql
2. 将所有大写表都rename成小写表，然后设置lower\_case\_table\_names=1，重启mysql

## waiting for table metadata lock

### 故障描述

Show processlist查看发现很多waiting for table metadata lock。

### 故障分析

一般是由于事务未提交导致元数据锁不释放引起的。

### 解决方法

**场景一：长事务运行，阻塞DDL，继而阻塞所有同表的后续操作**

通过show processlist可以看到tableA上有正在进行的操作（包括读），此时alter table语句无法获取到metadata独占锁，会进行等待。

这是最基本的一种场景，这个和mysql5.6中的online DDL并不冲突。一般alter table的操作过程中，在after create步骤会获取metadata独占锁，当进行到altering table的过程时（通常是最花时间的步骤），对该表的读写都可以正常运行，这就是online DDL的表现，并不会像之前在整个alter table过程中阻塞写入。（当然，也并不是所有类型的alter操作都能online的，具体参考官方手册说明）

处理方法：kill掉DDL所在的session。

**场景二：未提交事务，阻塞DDL，继而阻塞所有同表的后续操作**

通过show processlist看不到tableA上有任何操作，但是实际上存在有未提交的事务，可以在information\_schema.innodb\_trx中查看到。在事务没有完成之前，tableA的锁不会释放，alter table同样获取不到metadata的独占锁。

处理方法：通过select \* from information\_schema.innodb\_trx\G，找到未提交事务的sid，然后kill掉，让其回滚。

**场景三：执行失败操作未释放锁**

通过show processlist看不到tableA上有任何操作，在information\_schema.innodb\_trx中也没有任何进行中的事务，这很可能是因为在一个显式的事务中，对于tableA进行了一个失败的操作（比如查询了一个不存在的字段），这时候事务还没有开始，但是失败语句获取到的锁依然有效，没有释放。从performance\_schema.events\_statements\_current表中可以查看到失败的语句。

根据官方文档，除了语法错误，其他错误语句获取到的锁在这个事务提交或回滚之前，仍然不会释放掉。Because the failed statement is written to the binary log and the locks protect log consistency，但是解释这一行为的原因很难理解，因为错误的语句根本不会记录到二进制日志中。

处理方法：通过performance\_schema.events\_statement\_current找到其sid，kill掉该session，也可以kill掉DDL所在的sessioon。

# 统一运维平台

## OMM安装并启动后界面无法访问

### 故障描述

OMM在安装启动后无法通过浏览器打开界面，出现以下几种情况错误：

1、404

2、空白页面

3、页面错乱报js错误

### 故障分析

1、404错误

业务在omm的机器上安装了其他应用，并配置了安装变量参数，导致omm读取该变量参数，导致界面打开失败。

2、空白页面

1）尝试在omm所在服务器通过wget命令访问omm

2）检查java进程是否存在

3）检查omm所连接的mysql/rddb是否启动，是否可访问，同时检查“was/tomcat/webapps/uniportal/WEB-INF/classes/com/zte/config/xml”中mysql的数据源、账号、密码配置是否正确

3、页面错乱

浏览器兼容问题

### 解决方法

1、404错误

将/etc/profile 中配置的spring的全局变量注释掉

2、空白页面

1）关闭防火墙

2）检查进程是否存在

3）检查访问数据库的配置是否正确

3、界面错乱

使用推荐的浏览器版本

## OMM安装后显示为中兴通讯界面

### 故障描述

OMM安装后，显示界面为中性通讯界面，行方生产环境不允许出现厂商信息。

### 故障分析

OMM默认安装使用的是中兴通讯表示，需要替换为行方的标识。

### 解决方法

1. 连接OMM底层数据库，执行语句

insert into zxinsys.portal\_sysparam(param\_name,param\_value,paramgroup) values(‘LONGLONG’,’XX银行’,’平台系统配置’)

upudate zxinsys.portal\_sysparam set param\_value=’XX银行’ where param\_name=’COMPANY\_NAME’;

1. 替换文件

/home/omm/was/tomcate/webapps/ROOT/icon.ico

/home/omm/was/tomcate/webapps/uniportal/\*

# 业务使用

## 业务侧连接数据库报错:Communications link failure

### 故障描述

业务侧连接数据库报错：Communications link failure，少量连接数可以连接，连接数一增大就报错。

### 故障分析

可能原因如下：

1. proxy连接实例中的连接池大小太小
2. DB的最大连接数过小
3. BD或proxy的句柄数过小

### 解决方法

1. OMM界面proxy连接实例中将最大连接数改大，proxy下os.ini的max\_link\_sessions改大
2. Mysql客户端连上mysql，show variables like ‘%max\_connections%’;确认是否是10000。如过小，则修改my.cnf中mysqld段，增加配置max\_connecions=10000，同时在客户端执行set global max\_connection=10000，不重启DB
3. DB和proxy用户下检查系统句柄数

## Communication link failure during commit() .Transaction resolution unknow

### 故障描述

业务抛出异常，日志如下：

Com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionExcep:Communications link failure during commit(),Transaction resolution unknow.

### 故障分析

此类异常抛出均在写语句的提交阶段。本模型中1insert+1update均会出现抛出异常日志。如下两种场景会导致该错误：

1. 原因是因为jdbc的socket超时时间小于dbgroup主备之间同步的超时时间（10s），当dbgroup的最后一个活跃备机异常（同步超时），或在业务socket超时时间内未完成提交则抛出此类错误
2. 单语句执行时间超时，超过jdbcsocket超时时间

### 解决方法

1. 请检查主备状态是否正常，复制关系是否正常
2. 请检查jdbccurl和业务代码中是否有socketTieout相关的配置

## 通过JDBC插入数据内容为空

### 故障描述

业务通过setString(‘2018-01-01’)的方式给date类型复制的时候发现赋值失败，导致数据未能入库，查询出来数据内容为空。

### 故障分析

原因是生成的连接类型是MariadbConnection导致消息不兼容。具体原因是业务在引用jar包的时候使用了mariadb的connect/j。

### 解决方法

修改引用jar包，删除mariadb的驱动包，改为mysql驱动包。