# MySQL galera cluster

[mnstory.net](http://mnstory.net)

galera cluster是一个MySQL的多主高可用方案，相对于异步复制，最大的长处是不丢数据，因为他是同步复制。

## 优点与缺点

### 优点

1. 真正的多主模式（True Multi-master）

意味着你可以在任意节点读写，适度的规模可以提高集群整体的性能。

不会出现主从模式的故障转移（Master-Slave Failover）操作，也不需要VIP。

1. 同步复制（Synchronous Replication）

意味着没有slave lag，没有节点crash的时候出现的丢数据现象(Hot Standby)。

并且是Multi-threaded Slave，性能也不错。

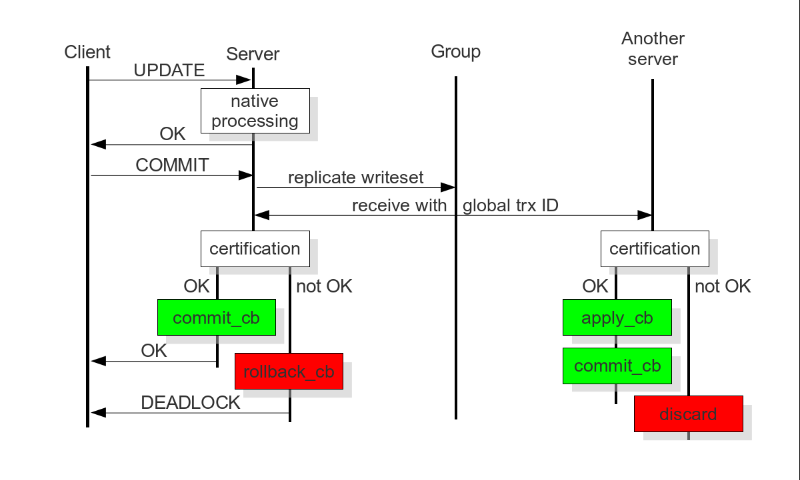
紧耦合（Tightly Coupled）所有节点数据和状态一致。

1. 自动节点管理（Automatic Node Provisioning）

不需要人工去备份数据库恢复到新节点。

参考：["Benefits of Galera Cluster"](http://galeracluster.com/documentation-webpages/)

我们看看galera的同步复制原理：



1. 先提交到当前节点的write-set内。
2. 将write-set改变复制到其他节点。
3. 到其他节点后，最关键的一步，用Primary key探测是否有冲突，有冲突就rollback，没有就commit，如果有问题的几点怎么办，把它踢出去。

### 缺点

1. 只支持InnoDB

对MyISAM，只支持DDL语句，就是说，创建了表可以在其他节点看到，但插入的数据各是各的。

为什么是InnoDB？

多主进行更新的时候，每个节点执行的是乐观策略（假定没有冲突），然后开始更新数据，等到要commit的时候，再问大家是不是有冲突，这样一来，如果有冲突，执行的语句是不是得回滚，而你知道的MyISAM它不支持事务，所以需要支持事务的引擎，而当前MySQL就InnoDB支持事务。

2. 表里面需要有PK

如果没有Primary Key，那不同节点DELETE后，可能顺序不一致，表现在select limit语句在不同节点可能返回不一致。

1. 不支持lock/unlock tables, lock functions (GET\_LOCK(), RELEASE\_LOCK()... )

当两个transaction从不同节点更新同一行数据的时候，只有一个transaction会成功，另外一个会返回ER\_LOCK\_DEADLOCK。

这其实也没啥大问题，因为逻辑设计上也有问题，不应该将事同一时刻的一个写重定向到两个节点的情况，最好的解决方法是，只有一个节点写。

当然，这是一件极掉逼格的一件事，人家的多主模式一下子被干成了单主，说出去不好听。

1. 查询日志不能存到表里，只能存文件。

log\_output=FILE，这个一般也不用，特别是对我来说没啥影响。

1. 最大transaction size受 wsrep\_max\_ws\_rows，wsrep\_max\_ws\_size两变量控制。

超出会被reject。准确的说，不是一个缺点。

参考：["Limitations in Galera Cluster for MySQL"](http://support.severalnines.com/hc/en-us/community/posts/207460763-Limitations-in-Galera-Cluster-for-MySQL)

对于上述缺点，在部署之前，就应该考虑清楚能否接受，特别是现有表结构是否有不符合规范的地方，这里有个SQL语句可以查询不符合规矩的表：

mysql> SELECT DISTINCT

CONCAT(t.table\_schema,'.',t.table\_name) as tbl,

t.engine,

IF(ISNULL(c.constraint\_name),'NOPK','') AS nopk,

IF(s.index\_type = 'FULLTEXT','FULLTEXT','') as ftidx,

IF(s.index\_type = 'SPATIAL','SPATIAL','') as gisidx

FROM information\_schema.tables AS t

LEFT JOIN information\_schema.key\_column\_usage AS c

ON (t.table\_schema = c.constraint\_schema AND t.table\_name = c.table\_name

AND c.constraint\_name = 'PRIMARY')

LEFT JOIN information\_schema.statistics AS s

ON (t.table\_schema = s.table\_schema AND t.table\_name = s.table\_name

AND s.index\_type IN ('FULLTEXT','SPATIAL'))

WHERE t.table\_schema NOT IN ('information\_schema','performance\_schema','mysql')

AND t.table\_type = 'BASE TABLE'

AND (t.engine <> 'InnoDB' OR c.constraint\_name IS NULL OR s.index\_type IN ('FULLTEXT','SPATIAL'))

ORDER BY t.table\_schema,t.table\_name;

+-------------------------------+--------+------+-------+--------+

| tbl | engine | nopk | ftidx | gisidx |

+-------------------------------+--------+------+-------+--------+

| neutron.alembic\_version | InnoDB | NOPK | | |

| neutron.alembic\_version\_fwaas | InnoDB | NOPK | | |

| neutron.alembic\_version\_lbaas | InnoDB | NOPK | | |

+-------------------------------+--------+------+-------+--------+

3 rows in set (0.34 sec)

如上，我的数据库输出三个没有PK的表。

## 部署

编译安装就不说了，文档太多。记得几个关键步骤：

1. 创建mysql用户

groupadd -g 3306 mysql

useradd -g mysql -u 3306 -r -s /bin/false mysql

1. 修改相关目录权限

chown mysql:mysql -R /var/log/mysql

chown mysql:mysql -R /var/run/mysqld

1. 初始化

/usr/bin/mysql\_install\_db --user=mysql --datadir=/var/lib/mysql

### 配置

最主要的my.cnf信息，差不多这样配置，就能用：

[mysqld]

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock

user=mysql

#使用galera一定要是ROW格式而不能是SQL格式，不然会影响性能和一致性

binlog\_format=ROW

#不要bind到本地地址，例如127.0.0.1，这样其他机器无法访问

bind-address=0.0.0.0

#galera在非事务性的存储引擎，例如MyISAM上无法工作

default\_storage\_engine=innodb

#AUTO\_INCREMENT字段，全部使用interleaved lock mode，此种方式不适用于SQL模式的日志复制，但是比较适合ROW模式，由于insert语句不使用表级auto-inc lock，所以速度比较快

innodb\_autoinc\_lock\_mode=2

#为了提高性能，galera官方建议使用1s刷一次日志的方式，但是同时也是不安全的方式，如果有备用电源，我认为设置为2比较好

innodb\_flush\_log\_at\_trx\_commit=0

#几台机器的必须一样，用来区分不同的集群

wsrep\_cluster\_name=YourClusterName

#这个插件位置最关键，没有的话运行不起来

wsrep\_provider=/usr/lib/galera/libgalera\_smm.so

#这个就是从谁那儿同步，后面可以是用逗号分隔开来的IP或者域名

wsrep\_cluster\_address=gcomm://200.200.102.207,200.200.102.208

#本地连出去的时候用的地址

wsrep\_node\_address=200.200.102.203

#如果你没看文档，还是不要配了

#wsrep\_provider\_options="gcache.size=300M; gcache.page\_size=300M"

#sst(State Snapshot Transfer)的默认同步方式就是rysnc，具体请参考：<http://galeracluster.com/documentation-webpages/sst.html>

#wsrep\_sst\_method=rsync

[mysql\_safe]

log-error=/var/log/mysqld.log

pid-file=/var/run/mysqld/mysqld.pid

### 现有数据迁移

比如说从原来的semi-sync方式迁移到galera上，推荐使用原来的master作为first node，其实我觉得，最简单的是，将原来的数据mysqldump出来：

$ mysqldump -u root -p --skip-create-options --all-databases > migration.sql

skip-create-options保证了导入数据的时候，用的默认存储引擎，这样一来，原来的MyISAM也会顺利转换为InnoDB。

在galera上导入：

$ mysql -u root -p < migration.sql

### 启动顺序

必须要有一个不带连接NODE方式（参数--wsrep-new-cluster）先启动，作为first node，其他节点掉线了，first node还是能正常运行：

# systemctl start mysql --wsrep-new-cluster

运行 SHOW STATUS LIKE 'wsrep\_cluster\_size'检测是否启动成功（这个结果显示的是多少个节点加入了这个cluster）。

如果集群已经有节点存活，后续启动节点还是按不带连接NODE的方式启动，会提示错误：

[ERROR] WSREP: It may not be safe to bootstrap the cluster from this node. It was not the last one to leave the cluster and may not contain all the updates. To force cluster bootstrap with this node, edit the grastate.dat file manually and set safe\_to\_bootstrap to 1 .

[ERROR] WSREP: wsrep::connect(gcomm://) failed: 7

[ERROR] Aborting

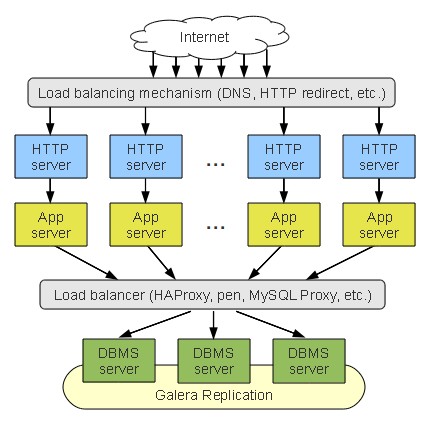
在重启整个galera cluster的时候，推荐使用most advanced node作为first node。

1. Identify the node with the most advanced node state ID.
2. Start the most advanced node as the first node of the cluster.
3. Start the rest of the node as usual.

具体参考<http://galeracluster.com/documentation-webpages/restartingcluster.html>

## 高可用

准备前端用HAProxy做负载，这样后端节点如果有故障，能自动跳过，而且不需要配置虚拟IP，简化工作，并且可以退一步做读写分离：



关于galera的推荐配置模型，详情可以参考：<http://galeracluster.com/documentation-webpages/deploymentvariants.html>

HAProxy相关配置可以参考：<http://galeracluster.com/documentation-webpages/haproxy.html>，此处不再赘述。

## 故障

### 状态确认

通过SHOW STATUS LIKE 'wsrep\_%'语句，可以查看当前的cluster状态：

+---------------------------+------------+

| Variable\_name | Value |

+---------------------------+------------+

| wsrep\_local\_state\_comment | Synced (6) |

| wsrep\_cluster\_size | 3 |

| wsrep\_ready | ON |

+---------------------------+------------+

这三个可以重点关注：

1. wsrep\_local\_state\_comment

Synced表示自己已经和集群连接，并可操作。

1. wsrep\_cluster\_size

查看当前集群的节点个数。

1. wsrep\_ready

为ON表示可以接客了。

状态分类介绍，参考自[《MySQL的Galera Cluster配置说明》](http://blog.sina.com.cn/s/blog_53b13d950102uxpx.html)。

mysql> SHOW STATUS LIKE 'wsrep%';

+------------------------------+--------------------------------------+

| Variable\_name | Value |

+------------------------------+--------------------------------------+

| wsrep\_local\_state\_uuid | 07bfd155-ac9c-11e6-bcfd-8a32fed329f7 |

| wsrep\_protocol\_version | 7 |

| wsrep\_last\_committed | 2 |

| wsrep\_replicated | 0 |

| wsrep\_replicated\_bytes | 0 |

| wsrep\_repl\_keys | 0 |

| wsrep\_repl\_keys\_bytes | 0 |

| wsrep\_repl\_data\_bytes | 0 |

| wsrep\_repl\_other\_bytes | 0 |

| wsrep\_received | 6 |

| wsrep\_received\_bytes | 398 |

| wsrep\_local\_commits | 0 |

| wsrep\_local\_cert\_failures | 0 |

| wsrep\_local\_replays | 0 |

| wsrep\_local\_send\_queue | 0 |

| wsrep\_local\_send\_queue\_max | 2 |

| wsrep\_local\_send\_queue\_min | 0 |

| wsrep\_local\_send\_queue\_avg | 0.500000 |

| wsrep\_local\_recv\_queue | 0 |

| wsrep\_local\_recv\_queue\_max | 1 |

| wsrep\_local\_recv\_queue\_min | 0 |

| wsrep\_local\_recv\_queue\_avg | 0.000000 |

| wsrep\_local\_cached\_downto | 18446744073709551615 |

| wsrep\_flow\_control\_paused\_ns | 0 |

| wsrep\_flow\_control\_paused | 0.000000 |

| wsrep\_flow\_control\_sent | 0 |

| wsrep\_flow\_control\_recv | 0 |

| wsrep\_cert\_deps\_distance | 0.000000 |

| wsrep\_apply\_oooe | 0.000000 |

| wsrep\_apply\_oool | 0.000000 |

| wsrep\_apply\_window | 0.000000 |

| wsrep\_commit\_oooe | 0.000000 |

| wsrep\_commit\_oool | 0.000000 |

| wsrep\_commit\_window | 0.000000 |

| wsrep\_local\_state | 4 |

| wsrep\_local\_state\_comment | Synced |

| wsrep\_cert\_index\_size | 0 |

| wsrep\_causal\_reads | 0 |

| wsrep\_cert\_interval | 0.000000 |

| wsrep\_incoming\_addresses | , |

| wsrep\_desync\_count | 0 |

| wsrep\_evs\_delayed | |

| wsrep\_evs\_evict\_list | |

| wsrep\_evs\_repl\_latency | 0/0/0/0/0 |

| wsrep\_evs\_state | OPERATIONAL |

| wsrep\_gcomm\_uuid | 162f9df1-acb3-11e6-99a4-cb92a5f5e579 |

| wsrep\_cluster\_conf\_id | 2 |

| wsrep\_cluster\_size | 2 |

| wsrep\_cluster\_state\_uuid | 07bfd155-ac9c-11e6-bcfd-8a32fed329f7 |

| wsrep\_cluster\_status | Primary |

| wsrep\_connected | ON |

| wsrep\_local\_bf\_aborts | 0 |

| wsrep\_local\_index | 0 |

| wsrep\_provider\_name | Galera |

| wsrep\_provider\_vendor | Codership Oy <info@codership.com> |

| wsrep\_provider\_version | 3.19(rb98f92f) |

| wsrep\_ready | ON |

+------------------------------+--------------------------------------+

1. 集群完整性检查
2. wsrep\_cluster\_state\_uuid

在集群所有节点的值应该是相同的，有不同值的节点，说明其没有连接入集群。

1. wsrep\_cluster\_conf\_id

正常情况下所有节点上该值是一样的，如果值不同，说明该节点被临时"分区"了。当节点之间网络连接恢复的时候应该会恢复一样的值。

1. wsrep\_cluster\_size

当前集群的节点个数，如果这个值跟预期的节点数一致，则所有的集群节点已经连接。

1. wsrep\_cluster\_status

集群组成的状态如果不为"Primary"，说明出现"分区"或是"split-brain"状况。

1. 节点状态检查
2. wsrep\_ready

该值为ON,则说明可以接受SQL负载，如果为Off，则需要检查wsrep\_connected。

1. wsrep\_connected

如果该值为Off，且wsrep\_ready值也为Off，说明该节点没有连接到集群，可能是wsrep\_cluster\_address或wsrep\_cluster\_name等配置错造成的，具体错误需要查看日志。

1. wsrep\_local\_state\_comment

如果wsrep\_connected为On，但wsrep\_ready为OFF，则可以从该项查看原因。

1. 复制健康检查
2. wsrep\_flow\_control\_paused

表示复制停止了多长时间，集群因为Slave延迟而慢多少。该值为0~1之间，越靠近0越好，值为1表示复制完全停止。可优化wsrep\_slave\_threads来改善。

1. wsrep\_cert\_deps\_distance

有多少事务可以并行处理，wsrep\_slave\_threads设置的值不应该高出该这个太多。

1. wsrep\_flow\_control\_sent

表示该节点已经停止复制了多少次。

1. wsrep\_local\_recv\_queue\_avg

表示slave事务队列的平均长度，slave有没有瓶颈，可以从这里看。最慢节点的wsrep\_flow\_control\_sent和wsrep\_local\_recv\_queue\_avg值都是最高的，这两个值越低越好。

1. 检测慢网络问题
2. wsrep\_local\_send\_queue\_avg

网络瓶颈的预兆，如果这个值比较高的话，可能存在网络瓶颈。

1. 冲突或死锁的数目
2. wsrep\_last\_committed

最后提交的事务数目。

1. wsrep\_local\_cert\_failures/wsrep\_local\_bf\_aborts

回滚/检测到的冲突数目。

### 故障模拟

理论上，只要有超过三个的节点，单点故障是不会出现集群不可用的：

1. 模拟单点服务故障

killall -9 mysqld

此时，其他节点会提示节点掉线，但集群可用：

[Note] WSREP: forgetting 1168c050 (tcp://200.200.102.208:4567)

[Note] WSREP: cleaning up 1168c050 (tcp://200.200.102.208:4567)

1. 模拟单点宕机故障

crash system

1. 模拟网络故障

node1上目前运行了MySQL，并且和204, 208都有连接：

node1$ netstat -nap | grep 4567

tcp 0 0 0.0.0.0:4567 0.0.0.0:\* LISTEN 76299/mysqld

tcp 0 0 200.200.102.203:48938 200.200.102.208:4567 ESTABLISHED 76299/mysqld

tcp 0 0 200.200.102.203:50728 200.200.102.204:4567 ESTABLISHED 76299/mysqld

先确保所有iptables规则已经清除：

node1$ iptables –F

将进出4567端口的链接都断掉：

node1$ iptables -A OUTPUT -p tcp --dport 4567 -m state --state NEW,ESTABLISHED,RELATED -j DROP

node1$ iptables -A INPUT -p tcp --dport 4567 -m state --state NEW,ESTABLISHED,RELATED -j DROP

node1$ iptables -L -n --line-number

Chain INPUT (policy ACCEPT)

num target prot opt source destination

1 DROP tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:4567 state NEW,RELATED,ESTABLISHED

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

1 DROP tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:4567 state NEW,RELATED,ESTABLISHED

此时本机状态会变化，并提示其他节点链接不上：

/sf/bin/galera-callback --status Undefined --uuid 07bfd155-ac9c-11e6-bcfd-8a32fed329f7 --primary no --index 0 --members 4cb0894f-ad6e-11e6-b756-fe900259a6d0/200.200.102.203.acloud.vt/200.200.102.203:3306

2016-11-18 17:43:39 76299 [Note] WSREP: (4cb0894f, 'tcp://0.0.0.0:4567') connection to peer 00000000 with addr tcp://200.200.102.208:4567 timed out, no messages seen in PT3S

2016-11-18 17:43:43 76299 [Note] WSREP: (4cb0894f, 'tcp://0.0.0.0:4567') connection to peer 00000000 with addr tcp://200.200.102.204:4567 timed out, no messages seen in PT3S

数据库也不能使用：

node1> select \* from test;

ERROR 1047 (08S01): WSREP has not yet prepared node for application use

其他机器会提示状态变化，如下之前的三台变为两台，数据库能使用，但是肯定不会向203同步了：

/sf/bin/galera-callback --status Synced --uuid 07bfd155-ac9c-11e6-bcfd-8a32fed329f7 --primary yes --index 2 --members 4cb0894f-ad6e-11e6-b756-fe900259a6d0/200.200.102.203.acloud.vt/200.200.102.203:3306,66938990-ad66-11e6-8515-f7194270ea10/node-103.acloud.vt/node-103.acloud.vt:3306,ffeb7999-ad6e-11e6-b924-bbe152c16f21/node-108.acloud.vt/node-108:3306

/sf/bin/galera-callback --status Synced --uuid 07bfd155-ac9c-11e6-bcfd-8a32fed329f7 --primary yes --index 1 --members 66938990-ad66-11e6-8515-f7194270ea10/node-103.acloud.vt/node-103.acloud.vt:3306,ffeb7999-ad6e-11e6-b924-bbe152c16f21/node-108.acloud.vt/node-108:3306

2016-11-18 17:43:04 58856 [Note] WSREP: (ffeb7999, 'tcp://0.0.0.0:4567') reconnecting to 4cb0894f (tcp://200.200.102.203:4567), attempt 0

2016-11-18 17:43:07 58856 [Note] WSREP: (ffeb7999, 'tcp://0.0.0.0:4567') connection to peer 00000000 with addr tcp://200.200.102.203:4567 timed out, no messages seen in PT3S

删除规则后：

node1$ iptables -D OUTPUT 1

node1$ iptables -D INPUT 1

node1再次上线，数据同步。

### 常见故障

**未配置wsrep\_node\_address地址**

wsrep\_node\_address地址是用来链接其他节点的地址，如果没有配置，默认取第一个interface上的第一个IP，问题来了，这可能取到一些不正确的IP，例如，我的测试设备取到了127.0.0.1，于是，mysqld进程挂了。

进程堆栈为：

Program received signal SIGABRT, Aborted.

[Switching to Thread 0x7fffd0752700 (LWP 203059)]

0x00007ffff5d695f7 in raise () from /lib64/libc.so.6

Missing separate debuginfos, use: debuginfo-install glibc-2.17-106.el7\_2.4.x86\_64 glusterfs-3.6.0.29-265029.278635.x86\_64 libaio-0.3.109-13.el7.x86\_64 libgcc-4.8.5-4.el7.x86\_64 libstdc++-4.8.5-4.el7.x86\_64 nss-softokn-freebl-3.16.2.3-14.2.el7\_2.x86\_64 tcp\_wrappers-libs-7.6-77.el7.x86\_64 zlib-1.2.7-15.el7.x86\_64

(gdb) bt

#0 0x00007ffff5d695f7 in raise () from /lib64/libc.so.6

#1 0x00007ffff5d6ace8 in abort () from /lib64/libc.so.6

#2 0x00007fffdb169667 in gu\_abort () at galerautils/src/gu\_abort.c:34

#3 0x00007fffdb274113 in gcs\_core\_recv (conn=0x145cd60, recv\_act=recv\_act@entry=0x7fffd0751e80, timeout=9223372035999999999) at gcs/src/gcs\_core.cpp:1141

#4 0x00007fffdb27a91c in gcs\_recv\_thread (arg=0x145cb70) at gcs/src/gcs.cpp:1191

#5 0x00007ffff7bc6dc5 in start\_thread () from /lib64/libpthread.so.0

#6 0x00007ffff5e2a28d in clone () from /lib64/libc.so.6

错误日志：

2016-11-18 10:25:17 179929 [Note] WSREP: Running: 'wsrep\_sst\_rsync --role 'joiner' -**-address '127.0.0.1'** --datadir '/var/lib/mysql/' --defaults-file '/etc/mysql/my.cnf' --defaults-group-suffix '' --parent '179929' '' '

2016-11-18 10:25:18 179929 [Note] WSREP: Prepared SST request: rsync|127.0.0.1:4444/rsync\_sst

2016-11-18 10:25:18 179929 [Warning] WSREP: 1.0 (200.200.102.203.acloud.vt): State transfer to 0.0 (node-108.acloud.vt) failed: -255 (Unknown error 255)

2016-11-18 10:25:18 179929 [ERROR] WSREP: gcs/src/gcs\_group.cpp:gcs\_group\_handle\_join\_msg():736: Will never receive state. Need to abort.

2016-11-18 10:25:18 179929 [Note] WSREP: gcomm: terminating thread

对比一下正确日志：

2016-11-18 10:26:46 197844 [Note] WSREP: Running: 'wsrep\_sst\_rsync --role 'joiner' -**-address '200.200.102.208'** --datadir '/var/lib/mysql/' --defaults-file '/etc/mysql/my.cnf' --defaults-group-suffix '' --parent '197844' '' '

2016-11-18 10:26:46 197844 [Note] WSREP: Prepared SST request: rsync|200.200.102.208:4444/rsync\_sst

2016-11-18 10:26:48 197844 [Note] WSREP: 1.0 (200.200.102.203.acloud.vt): State transfer to 0.0 (node-108.acloud.vt) complete.

2016-11-18 10:26:48 197844 [Note] WSREP: Member 1.0 (200.200.102.203.acloud.vt) synced with group.

## 脑裂

"Split brain" is a condition whereby two or more computers or groups of computers lose contact with one another but still act as if the cluster were intact. This is like having two governments trying to rule the same country. If multiple computers are allowed to write to the same file system without knowledge of what the other nodes are doing, it will quickly lead to data corruption and other serious problems.

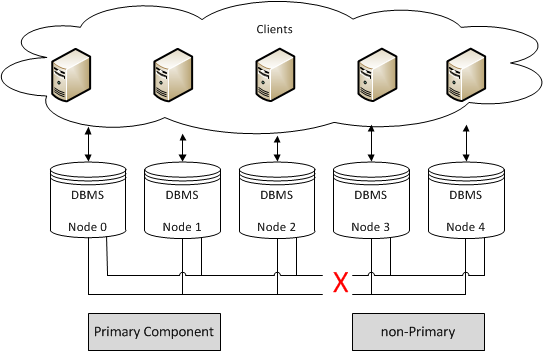
一般来说，物理上我们会考虑尽量避免脑裂，例如，为了避免交换机或者网口故障，我们使用双网卡加双交换的方式，做汇聚口提供服务。当然，架构复杂了，成本就上升了。

脑裂出现后，一般的稳定集群的办法：

### 投票

一般来说票数（不一定是节点数，可能一个节点拥有更多权重，可以占多票），哪个group能占到二分之一以上的票数，则哪个group对资源具有控制权。

在galera里面叫做 [WEIGHTED QUORUM](http://galeracluster.com/documentation-webpages/weightedquorum.html)机制，实现了这机制，如下图，当左边的节点数大于右边的时候，左边为Primary Component，提供服务：



当然，遇到两边权重一样的，就没得辄了，所以一般会避免节点数为偶数的情况，或者，为偶数的时候，调整为不同的权重：

SET GLOBAL wsrep\_provider\_options="pc.weight=3"

例如两台机器，这样node2一定不可能是Primary Component：

node1: pc.weight = 2

node2: pc.weight = 1

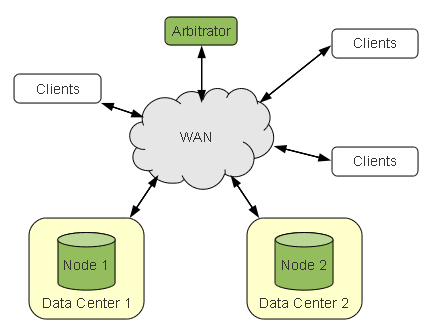
### 仲裁

最简单的仲裁模型，例如两节点，谁能PING通网关谁就获得所有权，但是，单纯的PING网关这种仲裁并不严谨，如果两个都能PING通呢？如果两个都不能PING通呢？

所以，一般的仲裁都不是简单的用PING实现，而是在仲裁上，加上资源加锁机制，第一个取得仲裁上资源的节点，获得控制权。

现实中往往不是那么简单，例如获取仲裁资源的节点何时释放资源？冲裁节点如何物理部署？

galera里的仲裁机制叫做[Galera Arbitrator](http://galeracluster.com/documentation-webpages/arbitrator.html)，其实可以理解为不落地的模拟节点，由一个叫garbd的模拟进程实现。



### fencing

前面的两种机制相对比较友好，咱们公平竞争，fencing的话比较腹黑，一般是利用某种技术把对方搞残，对方残了就不能和我竞争资源了。

这里的某种技术可能是某种能关闭电源的设备，当然，具体得根据场景，例如是两台虚拟机，完全可以是一个POWER OFF的API调用。

这里fencing的动作也是有考虑的，你觉得是reboot好呢还是power off好？假设是reboot，问题没解决，两个节点互相fencing对方，那不成了不停重启了？如果是power off，两个节点互相fencing对方，不就全部挂掉了？

所以，我不太喜欢fencing。

### 共享资源自带锁

共享资源自带锁，就是在要争取的资源上而不是节点上设置所属标记，这并不适用于所有场景，而且这个标记要用什么姿势来打，值得考虑。

共享资源自带锁，最典型的例子是磁盘锁，两方争抢的是共享磁盘上的文件的时候，我们将锁放在磁盘上，这样谁占有锁谁用就可以了。但是很多时候争取的共享资源不一定是共享文件，比如MySQL的主从复制，我数据并不放在共享存储上，而是放本地，问题出在谁对外提供服务，提供服务必然写数据库，会导致最终又两份数据。

## 测试

### 同步效果测试

1. 在node1上创建表

node1> create table test(id int, description varchar(10));

Query OK, 0 rows affected (0.01 sec)

node1> show tables;

+----------------+

| Tables\_in\_test |

+----------------+

| test |

+----------------+

1 row in set (0.00 sec)

1. 在node2上查看DDL语句是否同步过来

node2> show tables;

+----------------+

| Tables\_in\_test |

+----------------+

| test |

+----------------+

1 row in set (0.00 sec)

1. 在node2上插入数据

node2> insert into test values(1,'hello');

Query OK, 1 row affected (0.00 sec)

node2> select \* from test;

+------+-------------+

| id | description |

+------+-------------+

| 1 | hello |

+------+-------------+

1 row in set (0.00 sec)

1. 数据顺利同步到node1

node1> select \* from test;

+------+-------------+

| id | description |

+------+-------------+

| 1 | hello |

+------+-------------+

1 row in set (0.00 sec)

### 回调脚本

wsrep\_notify\_cmd选项，可以用于指定回调脚本，回调时会将集群状态等信息当参数传入你的命令行，详情请参考：<http://galeracluster.com/documentation-webpages/notificationcmd.html>

加入端回调示例：

2016-11-18 11:55:54.586872 --status Joiner --uuid 07bfd155-ac9c-11e6-bcfd-8a32fed329f7 --primary yes --index 1 --members 8f210044-ad40-11e6-8704-2b622281540b/node1/200.200.102.203:3306,e6eae266-ad42-11e6-ae64-c3455ec5818c/node-108.acloud.vt/node-108:3306

2016-11-18 11:55:57.785254 --status Synced

接受端回调示例：

2016-11-18 11:55:54.508756 --status Synced --uuid 07bfd155-ac9c-11e6-bcfd-8a32fed329f7 --primary yes --index 0 --members 8f210044-ad40-11e6-8704-2b622281540b/node1/200.200.102.203:3306,e6eae266-ad42-11e6-ae64-c3455ec5818c/node-108.acloud.vt/node-108:3306

2016-11-18 11:55:54.595162 --status Donor

2016-11-18 11:55:56.219732 --status Synced

如果接受失败，还会有一条会滚，例如：

2016-11-18 11:49:07.316609 --status Synced --uuid 07bfd155-ac9c-11e6-bcfd-8a32fed329f7 --primary yes --index 0 --members 8f210044-ad40-11e6-8704-2b622281540b/node1/200.200.102.203:3306

### 脑裂测试

两节点可能导致脑裂，测试步骤为：

1. 断开两节点之间的连线。

此时会发现服务都不能提供服务。

1. 重新链接网线。

还是不能提供服务，这个时候，需要在其中一个节点上，重置quorum，集群变得可用：

SET GLOBAL wsrep\_provider\_options='pc.bootstrap=1'

详情参考：<http://galeracluster.com/documentation-webpages/quorumreset.html>

## 参考资料

1. Ronald Bradford - 《Effective MySQL: Replication Techniques in Depth》
2. galera官方文档

<http://galeracluster.com/documentation-webpages/index.html>

2016/11/24