# PXC 运维总结

PXC 运维总结 1

PXC 简介 2

原理 2

优势 3

劣势 3

特性 4

PXC集群管理 6

集群初始化 6

安装 6

配置 7

启动 8

节点管理 8

配置调优 8

自动化 9

高可用 10

备份恢复 11

备份 11

恢复 11

监控 12

pxc迁移 14

迁移场景 14

迁移评估 14

迁移方案 14

问题汇总 15

参考资料 17

原理 17

配置 17

测试 17

工具 18

bugs 18

wiki 18

## PXC 简介

### 原理

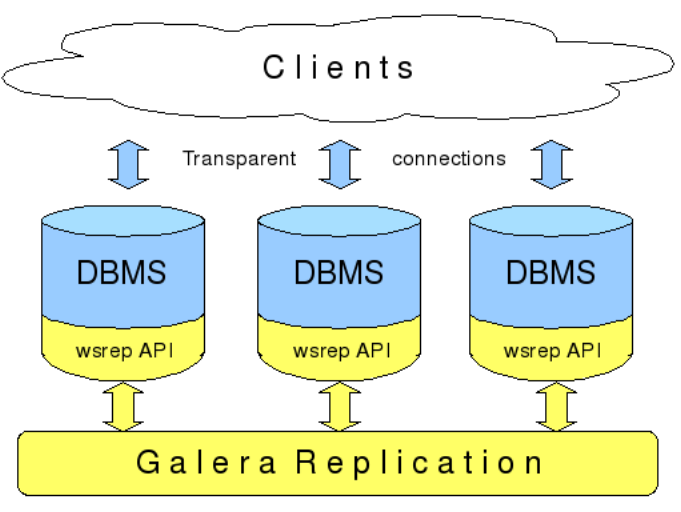


图 1 pxc 结构图

Percona XtraDB Cluster（PXC）：（ Percona XtraDB Cluster Server + WSREP API+ Galera ）

PXC应用write set replication 替换原生复制，在事务提交之前，集群内所有节点本地验证返回后根据验证结果判断是否提交；因此可保证集群数据的强一致性，并可以通过增加节点来扩展集群读写能力。

Wsrep api：(write set replication patches)

位于dbms和复制集群框架（galera）之间，提供基于writeset复制的接口服务。

Galera：(galera library)

复制核心类库，实现集群节点数据同步、保证数据强一致性、集群成员通信以及成员管理等功能。

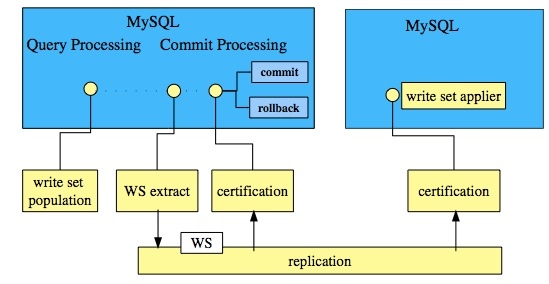


图 2 pxc sql生命周期

数据更新类sql语句在pxc中的生命周期：

节点接收sql请求后，mysql完成一般流程后准备commit之前，由wsrep API 调用galera 库进行包装（分配GTID等）后向集群内广播事务，其他节点接受到事务后进行验证，并返回验证结果，事务在集群所有节点验证通过后在写入节点提交，反之roll back。其他节点对于验证成功的事务，通过复制线程并行地做事务提交。因此，pxc尝试保证整个集群所有节点数据的强一致性，满足CAP理论中的Consistency 和  Availability。

相比传统复制，pxc额外的消耗：

* 集群通信：事务广播，数据传输，网络通信
* 节点事务验证

### 优势

* + - 高可用性，单个节点不可用不影响集群正常服务
    - 强一致性，可以将读扩展到多个节点上
    - 数据同步自动化（IST,SST）
    - 多节点读写

### 劣势

* 事务的提交需要所有节点作本地验证，集群单事务写入性能由集群中性能最差节点决定
* 为保证一致性，galera 总是优先保证数据一致性，在多点并发写时，乐观锁带来的冲突问题
* 新节点加入或延后较大的节点重新加入需全量拷贝数据（sst），作为donor的节点在同步过程中无法提供读写服务
* 高度数据冗余，数据冗余度为节点数
* 使用限制
  + - 1. 存储引擎（innodb>other）
      2. 主键
      3. 外键
      4. 触发器

### 特性

**1.端口**

mysql port : port=3306

group communication: gmcast.listen\_addr=tcp://10.10.58.168:4030

sst port : wsrep\_sst\_receive\_address=10.10.58.168:4020

ist port: ist.recv\_addr = tcp://10.10.58.168:4031

**2.UUID**

节点状态改变及顺序的唯一标识:

0e3bc86a-0014-11e3-a876-e7050fefda45

**3.GTID**

Global Transaction ID 。wsrep api 中定义的集群内全局事务id。UUID+seqno:

0e3bc86a-0014-11e3-a876-e7050fefda45:17121

**4.IST**

Incremental State Transfer. 实现节点数据与集群数据的增量同步。

发起条件：

* 节点的uuid和集群一致
* 偏移数据在donor节点的gcache文件中。

**5.SST**

State Snapshot Transfer 。实现节点数据的全量拷贝，执行时间决定数据备份策略和数据传输。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **speed** | **blocks the donor** | **can be done on live node?** | **logical/physical** | **requires root access to MySQL server?** |
| mysqldump | slow | yes | yes | logical | both donor and joiner |
| Rsync | fastest | yes | no | physical | none |
| Xtrabackup | fast | a very short time | no | physical | donor only |

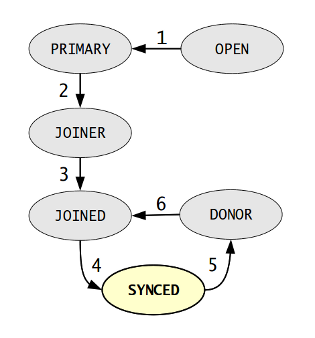
**6.DDL执行方式**

结构更改的2种方式：

TOI：Total Order Isolation 以statement模式复制到其他节点，并产生系统、库、表级别的独占锁，可造成整个集群在ddl执行过程中被锁住。

RSU：Rolling Schema Upgrade轮询方式执行语句，正在执行的节点脱离集群并在执行完成后重新加入集群。在全部完成之前各节点结构不一致。

7.状态机

**

状态机变化阶段：

     OPEN: 节点启动成功，尝试连接到集群，如果失败则根据配置退出或创建新的集群

     PRIMARY: 节点处于集群PC中，尝试从集群中选取donor进行数据同步

     JOINER: 节点处于等待接收/接收数据文件状态，数据传输完成后在本地加载数据

     JOINED: 节点完成数据同步工作，尝试保持和集群进度一致

     SYNCED：节点正常提供服务：数据的读写，集群数据的同步，新加入节点的sst请求

     DONOR：节点处于为新节点准备或传输集群全量数据状态，对客户端不可用。

状态机变化因素：

     新节点加入集群

     节点故障恢复

节点同步实效

## PXC集群管理

### 集群初始化

#### 安装

统一安装脚本：

/root/ovm/install\_mysql-XtraDB.sh

配置文件模块：

ftp://192.168.1.201/pub/software/unix/Oracle/server-storage/vm/template/SOHU\_CP\_PVM/mysql/my.cnf.pxc

依赖工具：

  yum install nc          #用于文件传输

  yum install xtrabackup     #sst 备份使用

1.**yum**

yum install cmake gcc gcc-c++ libaio libaio-devel automake autoconf bzr bison libtool ncurses5-devel

yum install Percona-XtraDB-Cluster-server Percona-XtraDB-Cluster-client xtrabackup

2.**apt**

gpg –keyserver hkp://keys.gnupg.net –recv-keys 1C4CBDCDCD2EFD2A

gpg -a –export CD2EFD2A | sudo apt-key add -

apt-get update

apt-get install build-essential flex bison automake autoconf bzr libtool cmake libaio-dev mysql-client libncurses-dev zlib1g-dev

apt-get install percona-xtradb-cluster-client-5.5 percona-xtradb-cluster-server-5.5 percona-xtrabackup

3.**rpm**

rpm -Uvh Percona-XtraDB-Cluster-galera-2.5-1.150.rhel6.x86\_64.rpm

rpm -Uvh Percona-XtraDB-Cluster-shared-5.5.30-23.7.4.404.rhel6.x86\_64.rpm

rpm -Uvh Percona-XtraDB-Cluster-devel-5.5.30-23.7.4.404.rhel6.x86\_64.rpm

rpm -Uvh Percona-XtraDB-Cluster-client-5.5.30-23.7.4.404.rhel6.x86\_64.rpm

rpm -Uvh Percona-XtraDB-Cluster-server-5.5.30-23.7.4.404.rhel6.x86\_64.rpm

#### 配置

1.必选配置：

* 1. wsrep\_provider   #galera 库文件路径
  2. wsrep\_cluster\_address  #集群地址

2.强制参数配置：

query\_cache\_size=0

#禁用查询缓存

binlog\_format=ROW

# galera 以GTID方式同步数据，对于存储过程，函数，触发器等需要记录其最终更改进行同步

default\_storage\_engine=innodb

#新版本已经支持mysaim

innodb\_autoinc\_lock\_mode=2

#auto increment 锁控制：interleaved，并发插入性能高，但id会出现间隙

innodb\_locks\_unsafe\_for\_binlog=1

innodb\_doublewrite=1

# this is the default and it should stay this way

3.wsrep provider 配置项

wsrep\_provider=/usr/lib64/galera/libgalera\_smm.so

#galera 库文件路径

wsrep\_provider\_options="gcache.size=32G; gcache.page\_size=1G"

#调优项

wsrep\_cluster\_address=gcomm://10.10.59.209, 10.10.59.210, 10.10.59.211

#集群地址，包括一个或多个可用节点地址，格式：ip/hostname:port 端口默认为4567

wsrep\_cluster\_name='my\_galera\_cluster'

#集群名称，用于区分不同集群以及新节点加入时验证是否为目标集群

wsrep\_node\_address='192.168.0.2'

#本地地址：ip/hostname：port

wsrep\_node\_name='node2'

#节点名称，pxc内部使用UUID进行标识节点，这个名称便于管理

wsrep\_sst\_method=xtrabackup

#sst 方式

wsrep\_sst\_auth=root:passwd

#sst 认证信息

wsrep\_node\_incoming\_address='192.168.10.2'

#配置server 监听地址，默认为本地ip和端口

wsrep\_sst\_donor='node3'

  #指定sst donor 节点，可以是节点名称或者具体ip地址。

wsrep\_slave\_threads=16

#默认为1。galera中用于应用队列中的事务到本地的线程数量，这些线程与mysqld 保持长连接并行进行事务处理，以提高数据同步的效率，降低同步延迟，同时，并发将带来系统资源的消耗，官方建议为cpu 核数的个数。状态值wsrep\_cert\_deps\_distance显示了slave threads 的上限值。

wsrep\_provider\_options 系列配置见参考文档。

#### 启动

第一个节点启动：service mysql start --wsrep-cluster-address=gcomm://

service mysql bootstrap-pxc

指定donor 节点启动：service mysql start --wsrep-sst-donor=node3

常规启动： service mysql start

### 节点管理

1. 增加节点

完成初始配置，指定集群地址，集群名称和本地节点名称后启动自动加入集群，进行SST/IST 后和其他节点保持同步:

wsrep\_cluster\_address=gcomm://10.10.58.168:4030,10.10.58.232:4030

#指定要加入的集群的其他节点的通信地址

wsrep\_sst\_receive\_address=10.10.58.209:4020

#指定SST接收地址

wsrep\_cluster\_name=PXCS\_10-10-57-2

#指定集群名称，如果和加入集群其他节点的配置名称不一致将会加入失败

wsrep\_provider\_options=ist.recv\_addr = tcp://10.10.58.209:4031;

#指定IST监听地址

wsrep\_node\_name=PXCN\_10-10-58-209

#指定本地节点名称，保持集群内唯一

wsrep\_sst\_method=xtrabackup

wsrep\_sst\_auth=root

#指定SST方式和认证信息

2.删除节点

mysql服务停止（手动停止，系统故障，软件bug等）

更新其他节点的配置文件中的wsrep\_provider\_options 配置。

如果被删除节点处于集群高可用结果中，需更新高可用配置，保证高可用架构的完整性。

如果被删除节点为集群备份节点，需选择集群内其他节点或增加节点并部署备份，保证集群灾难情况下的恢复能力。

### 配置调优

1.gcache.size=1G  #用于ist，如果节点重启则实效。

2.wan网超时配置：

evs.keepalive\_period = PT3S; evs.inactive\_check\_period = PT10S; evs.suspect\_timeout = PT30S; evs.inactive\_timeout = PT1M; evs.install\_timeout = PT1M"

3.wsrep\_slave\_threads=16

### 自动化

pxc资源申请自动化项目：<http://10.10.58.195/svn/cp/pxc/>

集群初始化：

1. 状态检测

pxc\_node\_init.pl --prepare --cluster-name=sohupxc\_sce01 --node-name=node210 --ip=10.10.59.210 --cluster-addr=10.10.59.209,10.10.59.215,10.10.59.210

2.部署

pxc\_node\_init.pl -init --run --cluster-name=sohupxc\_sce01 --node-name=node209

--ip=10.10.59.209 --cluster-addr=10.10.59.209,10.10.59.215,10.10.59.210

pxc\_node\_init.pl --run --cluster-name=sohupxc\_sce01 --node-name=node210

--ip=10.10.59.210 --cluster-addr=10.10.59.209,10.10.59.215,10.10.59.210

perl pxc\_node\_init.pl --run --cluster-name=sohupxc\_sce01 --node-name=node215

--ip=10.10.59.215 --cluster-addr=10.10.59.209,10.10.59.215,10.10.59.210

3.添加节点

perl pxc\_node\_init.pl --run --cluster-name=sohupxc\_sce01 --node-name=node215

--ip=10.10.59.215 --cluster-addr=10.10.59.209,10.10.59.215,10.10.59.210

4.高可用配置

root@[NODE\_IP\_1]/etc/keepalived/hamgrl2.sh --add --dbtype=pxcr -- dest\_ip=[]NODE\_IP\_1],[NODE\_IP\_2] vip=[FIRST\_RVIP] realservergrp=[NODE\_IP\_3]

5.备份部署

root@[192.168.1.128] /home/oracle/dbadmin/scripts/auto\_deploy\_mysqlbak.sh [NODE\_IP\_3]

6.上线检查

root@[NODE\_IP] /home/mysql/dbadmin/scripts/pxc/online\_precheck\_pxc.pl --dest\_ip=[NODE\_IP] --dest\_port=[PORT]

## 高可用

keepalived实现vip代理和切换。基于db分配读写vip，lvs实现负载均衡。

client

Wvip

Rvip

keepalived +lvs

node\_n

node\_3

node\_2

node\_1

pxc高可用管理和传统主从集群区别：

1. 不涉及主从切换，写vip可以直接切换到其他节点
2. 读写vip可以分配到任意节点，实现多点读写

3个节点pxc高可用运行机制：2个节点作为读写节点，第3个节点作为realserver分摊部分读压力和集群备份。

## 备份恢复

### 备份

     1.pxc集群正常运行模式下每个节点的数据是一致的，可使用正常备份工具进行备份，如mysqldump，xtrabackup等

     2.需要记录GTID的备份，用于新节点的恢复：

* 节点断开集群连接进行备份。
* 使用sst 接口进行备份，gardb 调用donor 提供的sst接口并运行脚本：

wsrep\_sst\_backup

* xtrabackup 带参数--galera-info运行备份

### 恢复

1. 新节点加入集群

* 新节点的加入最直接的办法使用sst 进行数据同步。
* 如果使用记录GTID的备份恢复，取决于备份的GTID与集群当前GTID是否全部存在donor 节点的gcache中，是则启用IST，反之启用SST做全量。

2.集群恢复

* mysqldump备份直接导入集群。可实现基于db和表级别的数据恢复
* xtrabackup 备份在一个节点上恢复，其他节点以新节点加入集群，无法实现基于db级别恢复。

1. 部分数据恢复

* mysqldump 备份过滤相应db/table 备份导入进行恢复
* xtrabackup 备份需在恢复到一个新实例后，使用mysqldump备份相应db/table 数据，再采用mysqldump恢复方式。
* binlog备份恢复，需要先对binlog基于db[table]级别的过滤，再应用binlog

## 监控

|  |  |  |
| --- | --- | --- |
| 状态参数 | 样例 | 说明 |
| wsrep\_local\_state\_uuid | e2c9a15e-5485-11e0-0800-6bbb637e7211 | 节点本地UUID，正常情况应该和集群的UUID一致 |
| wsrep\_last\_committed | 409745 | 最后提交事务的偏移值 |
| wsrep\_replicated | 16109 | 发送到集群的writeset 数量 |
| wsrep\_replicated\_bytes | 6526788 | 发送到集群的writeset 大小 |
| wsrep\_received | 17831 | 从集群接收的writeset 数量 |
| wsrep\_received\_bytes | 6637093 | 从集群接收的writeset 大小 |
| wsrep\_local\_commits | 14981 | 本地提交的事务总数 |
| wsrep\_local\_cert\_failures | 333 | 与slave thread 等待队列验证失败次数 |
| wsrep\_local\_bf\_aborts | 960 | 与slave thread 正在执行事务验证冲突的次数 #这2个值区别在产生锁冲突的阶段 |
| wsrep\_local\_replays | 0 | asymmetric lock granularity 产生的重试次数 |
| wsrep\_local\_send\_queue | 1 | 当前广播队列长度 |
| wsrep\_local\_send\_queue\_avg | 0.145000 | 本地广播队列平均长度， |
| wsrep\_local\_recv\_queue | 0 | 当前接收队列长度 |
| wsrep\_local\_recv\_queue\_avg | 3.348452 | 本地接收队列平均长度，值越小则节点性能越好 |
| wsrep\_flow\_control\_paused | 0.184353 | 流量控制阻塞，反应集群整体性能，越小集群性能越好，1表示集群一直未复制。 |
| wsrep\_flow\_control\_sent | 7 | FC\_PAUSE 发送次数 |
| wsrep\_flow\_control\_recv | 11 | FC\_PAUSE 接收次数 |
| wsrep\_cert\_deps\_distance | 23.88889 | 应用队列偏移量平局值，slave thread 并发应用事务的能力指标，threads 数可适当大于这个值 |
| wsrep\_apply\_oooe | 0.671120 | 未按顺序应用事务的次数 |
| wsrep\_apply\_oool | 0.195248 | How often writeset was so slow to apply that writeset with higher seqno's were applied earlier. Values closer to 0 mean greater gap between slow and fast write sets. |
| wsrep\_apply\_window | 5.163966 | 本地应用事务偏移量平均值，反应集群压力情况 |
| wsrep\_commit\_oooe | 0.000000 | 未按顺序事务提交的次数 |
| wsrep\_commit\_oool | 0.000000 | No meaning. |
| wsrep\_commit\_window | 0.000000 | 本地应用事务偏移量平均值，反应集群压力情况 |
| wsrep\_local\_state | 4 | 节点状态机 |
| wsrep\_local\_state\_comment | Synced | 节点状态 |
| wsrep\_incoming\_addresses | 10.0.0.1:3306,10.0.0.2:3306,undefined | 集群内部通讯配置信息 |
| wsrep\_cluster\_conf\_id | 34 | 集群结构变化次数 |
| wsrep\_cluster\_size | 3 | 当前集群节点数 |
| wsrep\_cluster\_state\_uuid | e2c9a15e-5485-11e0-0800-6bbb637e7211 | [Cluster state UUID](http://www.codership.com/wiki/doku.php?id=info#global_transaction_id_gtid). #集群当前的UUID ，正常情况和wsrep\_local\_state\_uuid一致 |
| wsrep\_cluster\_status | Primary | PRIMARY/NON\_PRIMARY.集群可用性状态 |
| wsrep\_local\_index | 1 | This node index in the cluster (base 0). |
| wsrep\_ready | ON | 节点加入节点开关 |

已有监控项：

<http://wiki.no.sohu.com/index.php/XtraDB_Cluster%E9%A1%B9%E7%9B%AE#.E9.97.AE.E9.A2.98.E6.B1.87.E6.80.BB>

## pxc迁移

### 迁移场景

* mysql复制拓扑扩展结构实例迁移到pxc
* mysql复制拓扑扩展数据库[部分表]实例迁移到pxc
* pxc集群实例迁移到mysql复制拓扑结构
* pxc集群实例迁移到mysql复制拓扑结构

### 迁移评估

评估标准：

* 主读业务：R:W>10:1，（轻量级业务仅参考）
* 迁移初期单db数据大小不超过50G
* 无myisam表或较少
* 没有无主键表或较少
* 无外键使用
* 无触发器使用
* db名和集群中已存在db不重复

迁移评估脚本及结果：

  bash pxc\_migrate\_check.sh -s 10 –h locahost

**Basic info : localhost:3306**Version : 5.1.59-rel13.0-log  
SLEEP : 10 s  
QPS : 0.00  
TPS : 20.60  
R/W : 0:445

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| database | rows | data-size | table-total | innodb | myisam | other | no\_pk\_table | forien\_key | procedure | tigger | event | view |
| recover | 220254 | 0.06GB | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| database | rows | data-size | table-total | innodb | myisam | other | no\_pk\_table | forien\_key | procedure | tigger | event | view |
| sbox | 162630906 | 62.82GB | 194 | 194 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |

**No PK tables :   sbox\_bind\_log\_00  sbox\_bus\_log**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| database | rows | data-size | table-total | innodb | myisam | other | no\_pk\_table | forien\_key | procedure | tigger | event | view | |
| transit | 98443 | 0.02GB | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 迁移方案

1. 基于原生复制
2. 基于trigger
3. 基于中间节点（blackhole）实现过滤复制
4. percona toolkit
5. tungsten replicator

## 问题汇总

1.为什么禁用query cache？

     1.避免多节点写入场景，数据同步过程中，读数据不一致

     2.避免query cache 频繁实效带来的性能影响

2. innodb\_autoinc\_lock\_mode=2

     并发应用中避免多点写入同一个表数据产生的auto increment 表级锁冲突，binlog模式为row，提高并发写入能力。

3.为何必须使用row 模式bin log，基于bin log进行复制？

？

4.新节点如何快速sst？

* gcs.max\_throttle 值设置为0.0，禁用流量控制以达到快速进行sst，但增加网络压力。
* 增大gcache文件大小，使用备份恢复后再做IST

5.如何减少节点？

* service mysql stop
* set **wsrep\_on=off**  测试发现没效果，数据依然同步，本地修改操作不同步到集群中。

6.galera 占用系统资源如何？

     ？运行在mysqld 进程内。

7.如何避免split-brain?

* 使用3个以上节点
* 配置文件中：wsrep\_provider\_options="pc.ignore\_sb = yes"
* 使用arbitrator。

10.writeset cache设置多大比较合适？

   writeset cache存储方式：

* 系统分配常驻内存，默认关闭
* 持久化到磁盘文件，常用模式，128M。分配时需加入到mysql内存计算公式中。一般节点1G，用于做IST节点2G~4G
* 实时分配，运行过程中按需产生内存数据页文件，上限为磁盘空间大小，内存占用较高。

控制参数：

**gcache.dir、gcache.name、gcache.size、gcache.page\_size、gcache.keep\_pages\_size、gcache.mem\_size**

11.节点失效检测过程？

   某节点在evs.keepalive\_period 周期内未发出数据包，其他节点等待其心跳信号时间超过evs.suspect\_timeout设置后，该节点被标识为可能非活跃，当其他所有节点返回同样的检测结果或者等待evs.inactive\_timeout时间之后未收的数据，节点被标识为非活跃。推荐配置：

evs.keepalive\_period <= evs.inactive\_check\_period <= evs.suspect\_timeout <= evs.inactive\_timeout <= evs.consensus\_timeout

12.trigger，procedure,event 在pxc中如何工作？

13.节点验证过程中如何解决冲突？

同步过程中，本地事务和等待队列中的锁冲突：

     innodb内部使用悲观锁，保证事务的成功进行和提交。pxc中使用乐观锁，以避免在每个节点获取锁以及网路开销，在写入节点上，事务在提交之前与单点的innodb一样，到达提交点时，向集群其他节点广播（galera 库完成 并发）事物并等待各节点验证结果，如果所有节点都返回成功，则提交，反之，回滚。

     pxc 中先提交的事物成功，其他事务（本地或其他节点同步）将回滚或报死锁错误。

相关状态值

* wsrep\_local\_cert\_failures  同步过程中节点认证失败计数，冲突来自本地提交的事务和同步队列中事务存在锁冲突，则本地验证失败（保证全局数据一致性）
* wsrep\_local\_bf\_aborts     强制放弃，本地事务和同步队列中正在执行的事务存在锁冲突时，将强制保证先提交的事务成功，后者回滚会报错

14.节点验证详细过程？

？

15.如何确认集群数据一致性？

？

## 参考资料

### 原理

wsrep source code ：<http://freecode.com/projects/mysqlgalera>

wsrep wiki：<http://www.codership.com/wiki/doku.php?id=info>s

galera wiki： <http://www.codership.com/wiki/doku.php?id=Galera%20Wiki>

mariadb galera cluster ：<https://kb.askmonty.org/en/about-galera-replication/>

flow control ： <http://en.wikipedia.org/wiki/Flow_control>

pxc 锁原理：<http://www.mysqlperformanceblog.com/2012/08/17/percona-xtradb-cluster-multi-node-writing-and-unexpected-deadlocks/>

pxc 中 dead lock ：<http://www.mysqlperformanceblog.com/2012/11/20/understanding-multi-node-writing-conflict-metrics-in-percona-xtradb-cluster-and-galera/>

pxc选举权重算法：<http://www.codership.com/wiki/doku.php?id=weighted_quorum>

innodb 自增锁控制模式：

<http://docs.oracle.com/cd/E17952_01/refman-5.1-en/innodb-auto-increment-handling.html>

### 配置

pxc+haproxy 高可用架构: <http://www.mysqlperformanceblog.com/2012/06/20/percona-xtradb-cluster-reference-architecture-with-haproxy/>

pxc 参数配置详细说明：<http://www.codership.com/wiki/doku.php?id=mysql_galera_configuration>

pxc 集群升级方案：<http://www.codership.com/wiki/doku.php?id=mysql_galera_cluster_upgrade>

galera 使用限制：<http://www.codership.com/wiki/doku.php?id=limitations>

galera cluster 最佳实践（非常完善、全面的ppt）：<http://www.percona.com/files/presentations/percona-live/nyc-2012/PLNY12-galera-cluster-best-practices.pdf>

pxc迁移方案 ：<http://www.percona.com/files/presentations/WEBINAR-Migrating-to-Xtradb-Cluster.pdf>

基于blackhole的过滤迁移：<http://jroller.com/dschneller/entry/mysql_replication_using_blackhole_engine>

### 测试

semi-replication 与pxc 性能对比测试：<http://www.mysqlperformanceblog.com/2012/06/14/comparing-percona-xtradb-cluster-with-semi-sync-replication-cross-wan/>

pxc 复制延迟测试：<http://www.mysqlperformanceblog.com/2013/03/03/investigating-replication-latency-in-percona-xtradb-cluster/>

mysql replication ，pxc，semisync replication 性能对比测试：<http://www.mysqlperformanceblog.com/2011/10/13/benchmarking-galera-replication-overhead/>

pxc 中myisam和innodb性能对比测试：

<http://dimitrik.free.fr/blog/archives/2012/11/mysql-performance-innodb-vs-myisam-in-56.html>

### 工具

备份工具：git clone git://github.com/severalnines/s9s-admin.git

监控工具 myq\_gadgets: <https://github.com/jayjanssen/myq_gadgets/>

tpcc压力测试：<https://code.launchpad.net/~percona-dev/perconatools/tpcc-mysql>

### bugs

myisam数据不一致：<https://bugs.launchpad.net/percona-xtradb-cluster/+bug/1171640>

触发器导致数据不一致：

<https://bugs.launchpad.net/codership-mysql/+bug/1064057>

外键问题：

<https://bugs.launchpad.net/codership-mysql/+bug/1078346>

### wiki

pxc 方案验证及性能测试：

<http://wiki.no.sohu.com/index.php/MySQL%E5%A4%A7%E9%9B%86%E7%BE%A4%E6%96%B9%E6%A1%88%E9%AA%8C%E8%AF%81>

pxc 和mysql 对比测试：

<http://wiki.no.sohu.com/index.php/Percona%E5%92%8Cmysql%E7%9A%84%E5%AF%B9%E6%AF%94%E6%B5%8B%E8%AF%95>

pxc集群项目：

<http://wiki.no.sohu.com/index.php/XtraDB_Cluster%E9%A1%B9%E7%9B%AE>

passport pxc集群运维文档：

<http://wiki.no.sohu.com/index.php/Percona_XtraDB_Cluster%E8%BF%90%E7%BB%B4%E7%AE%A1%E7%90%86>

pxc 使用文档（包括pxc原理、源码等内容）：

<http://wiki.no.sohu.com/index.php/Percona_XtraDB_Cluster>