MySQL Plus 运维手册

MySQL Plus 运维手册
1. MySQL Plus 虚拟机
    1.1. 虚拟机登录方式
    1.2. MySQL 登录方式
    1.3. 常用路径
2. 查询性能与CPU/内存负载
    2.1. 查询很慢?
    2.2. CPU占用率过高?
    2.3. 内存占用率过高?
3. 磁盘相关
    3.1. 如何查看磁盘 IO 压力?
    3.2. MySQL 磁盘容量报警，为何没多少数据量磁盘会满?
    3.3. 磁盘满导致节点异常怎么处理?
4. 主从同步
    4.1. MySQL主从延迟的原因?
    4.2. 从库重放 hang 了，但是IO和SQL都是YES?
5. IP 相关
    5.1. 数据库集群IP分配规则?
    5.2. 高可用读IP、高可用写IP、主节点IP、从节点IP的作用分别是什么?
    5.3. 读写请求是否负载到主节点了呢?
6. MySQL 配置参数
    6.1. MySQL 配置参数修改方式?
    6.2. 如何设置 MySQL 字符集编码?
7. MySQL 日志
    7.1. 如何获取 MySQL 慢日志、错误日志和审计日志?
    7.2. 如何获取 MySQL binlog?
    7.3. 如何获取 MySQL general log?
    7.4. 如何查看 MySQL binlog?
8. 账户
    8.1. 如何创建和获取数据库高级权限的账户?
    8.2. MySQL Plus 添加账户 创建的账户有哪些权限?
    8.3. 为什么添加数据库账户失败?
    8.4. 为什么添加数据库账户成功后，该账户无法正常访问数据库?
    8.5. 为什么删除数据库账户失败?
9. 服务监控
    9.1. 控制台提供的 MySQL 服务相关的监控数据是怎么来的?
    9.2. 用户对某些监控项有疑问或想知道与 show status 结果的对应关系?
    9.3. MySQL Plus 支持监控服务吗?
10. 备份
    10.1. MySQL Plus 备份机制?
    10.2. 备份的磁盘容量大小?
    10.3. 为什么 MySQL Plus 无法正常手动创建备份，或者自动创建备份失败呢?
    10.4. 基于备份创建集群需要多久？速度为多少?
11. 数据迁移
    11.1. 什么情况下支持在线迁移?
    11.2. MySQL 5.5 如何迁移到 MySQL Plus?
    11.3. 在线迁移如何操作?
12. 异地灾备
    12.1. 什么情况下支持异地灾备?
    12.2. 异地灾备如何操作?
13. 审计
    13.1. MySQL Plus 支持审计服务吗?

14. 扩容
14.1. 扩容是否会重启?
14.2. 更改主机类型耗时多久?
15. 升级
15.1. 升级方式是什么?
15.2. 升级过程中影响业务时间大概多久?
15.3. MySQL Plus 是否能直接升级到 MySQL Plus NeonSAN 版本?
16. 常见异常处理
16.1. 如何判断服务是否正常?
16.2. 如何确认 MySQL Plus 主从复制是否正常?
16.3. checkservice.log 报错 `try lock file failed`
16.4. checkservice.log 报错 `check service is ignored`
16.5. checkservice.log 报错 `disk is nearly full`
16.6. checkservice.log 报错 `env is not inited`
16.7. checkservice.log 报错 `mysql is not running`
16.8. checkservice.log 报错 `xenon peers.json is error`
16.9. checkservice.log 报错 `cluster has error node config`
16.10. checkservice.log 报错 `lvs master/slave vip is brain split`
16.11. checkservice.log 报错 `lvs master/slave no lvs list`
16.12. checkservice.log 报错 `wvip is not active`
16.13. checkservice.log 报错 `mysql status is error`
16.14. checkservice.log 报错 `mysql system user is error`
16.15. checkservice.log 报错 `slave gtid is bigger than leader`
17. 危险操作
17.1. 重建
17.1.1. 重建指令
17.1.2. 如何判断当前节点是否需要重建?
17.1.2. 为何重建失败?
17.2. 磁盘满需要清理binlog或relay log
17.3. 手动切主
18. 其他疑问
18.1. MySQL Event Scheduler 不能正常工作?
18.2. 哪里可以查阅 MySQL Plus 完整的产品文档?

1. MySQL Plus 虚拟机

1.1. 虚拟机登录方式

> 账户：ubuntu
>
> 密码：zhu1241jie

1.2. MySQL 登录方式

```
mysql -uroot
```

1.3. 常用路径

路径	作用
/usr/local/etc/ignore_agent	touch该文件后会禁用监控、healthcheck（检测并修复异常情况）功能。
/etc/init.d/xenon-service	MySQL Plus旧版（旧形态）的管理脚本，使用示例：/etc/init.d/xenon-service start
/etc/init.d/rds	MySQL Plus新版（新形态）的管理脚本，使用示例：/etc/init.d/rds start
/opt/xenon/xenoncli	用来与xenon通信的客户端工具
/opt/xenon/xenon	xenon主服务程序
/data/mysql/	mysql数据目录，mysql的日志文件 mysql-error.log、mysql-slow.log、mysql-audit.log 也在该目录
/data/log/	包含所有指令（比如：start、stop、restart）的日志及 xenon.log
/etc/mysql/my.cnf	mysql的配置文件
/etc/xenon8801.json	xenon的配置文件
/etc/xenon/xenon.conf	AppCenter confd程序根据metadata生成的配置文件

2. 查询性能与CPU/内存负载

2.1. 查询很慢?

1. 连接 MySQL 执行 `show engine innodb status\G` 查看数据库读写状态，重点关注最后的每秒读写情况。
2. 在 集群详情 - 基本属性 右上角的三道杠中，执行下 `同步日志`，再通过 wget 下载慢日志文件 `mysql-slow.log`。
3. 分析下慢日志里面运行很慢的 SQL，看是否能创建索引来优化，减少全表扫描。

2.2. CPU占用率过高?

1. 首先通过监控项（当前线程连接数、活跃线程连接数）或 `show processlist` 确认并发情况。如果并发较高，CPU占用率高就是并发太多引起的，建议扩容 CPU。
2. 如果并发并不是很多，那很有可能是复杂的查询造成的，建议下载分析慢日志，如果涉及聚合（count, sum, avg, max, min），分组（group by），排序（sort）等运算，是很消耗CPU资源的。如果 SQL 不好优化，可以选择扩容 CPU。

2.3. 内存占用率过高?

1. 为了达到最优的数据查询性能，数据库是有缓存机制的，MySQL Plus 会预留 65% 的内存用作 `innodb_buffer_pool` 缓存大小。

2. MySQL 最大内存分配情况：

```
key_buffer_size + query_cache_size + tmp_table_size +
innodb_buffer_pool_size + innodb_additional_mem_pool_size +
innodb_log_buffer_size
+ 连接数 * (read_buffer_size + read_rnd_buffer_size + sort_buffer_size+
join_buffer_size + binlog_cache_size + thread_stack)
```

3. `innodb_buffer_pool_size` 是边使用边增加的，直到达到设置的上限值，再之后就根据 LRU 算法置换。变量是 `连接数`，连接数越多，占用的内存会相对高些，其他的内存设置是固定的。
4. 建议：
    - 在配置参数页面调小 `innodb_buffer_pool_size` 可以减少内存占用。
    - 扩容内存也可以减少内存占用。

3. 磁盘相关

3.1. 如何查看磁盘 IO 压力?

```
# 查看当天 10:50:00~11:10:00 之间磁盘压力，若 %util 较大则表明磁盘繁忙
sar -d -s 10:50:00 -e 11:10:00
```

3.2. MySQL 磁盘容量报警，为何没多少数据量磁盘会满?

1. MySQL 产生的 binlog 是会占用空间的。除了定期删除 binlog 外，还可以设置 expire_logs_days 参数，适当调小（但不要小于2），以减少磁盘占用。
2. 对于从节点，除了 binlog，暂存的 relay log 也会占用空间的。一般 relay log 不会占用很多，除非有大事务或从库压力大导致重放慢。
3. MySQL 其他日志以及 MySQL Plus 日志文件也会占用部分数据盘。

3.3. 磁盘满导致节点异常怎么处理?

1. 清理较老的 binlog 日志文件。
2. 扩容磁盘容量。

4. 主从同步

4.1. MySQL主从延迟的原因?

4. binlog 重放是一个异步过程，即使 binlog 由主节点同步过来了，当从节点压力较大时也会导致 SQL 线程重放延迟，是会出现主库和从库读不一致的现象。
5. 若从节点正在重放一个大事务，也会造成主从延迟。
6. 建议：
    - 若业务对延迟不太敏感，可等待重放完成自动追上。
    - 若延迟较大，可重建从节点。

4.2. 从库重放 hang 了，但是IO和SQL都是YES?

1. 首先执行下 `/opt/xenon/xenoncli cluster gtid`，可以看到 FOLLOWER 的 `Executed_GTID_Set` 无变化，而 `Retrieved_GTID_Set` 却在增加（前提是主库有新事务产生），这表明从库在重放大事务或从库重放hang 了。

2. 执行 mysql -uroot -e'select * from replication_applier_status_by_worker;'，其中 `LAST_SEEN_TRANSACTION` 就是 hang 的事务，比如：`063a0edb-dab5-11e9-9a2c-525496bdd51a:3628`。
3. 执行 `mysql -uroot -e"show slave status\G"` 查看当前从库的 `Seconds_Behind_Master`，比如为 7200，那么可以结合这个时间及 binlog 文件的时间（`ll mysql-bin.xxxxx`）确定需要找哪些 binlog。
4. 逐个解析并查看主库的 binlog 内容，直到能找出该 gtid 所在的 binlog 及其事务内容：

```
mysqlbinlog --no-defaults -vv /data/mysql/mysql-bin.xxxxxx > /tmp/bxxxxxx

# 如果确定uuid不重复，grep的字符串可以少写一点，比如写为525496bdd51a:3628
# -A表示输出匹配行的后20行，-B表示输出匹配行的前20行
grep "063a0edb-dab5-11e9-9a2c-525496bdd51a:3628" /tmp/bxxxxxx -nr -A 20 -B
20
如果grep指令有结果输出，则表明查到了，反之，则删除 /tmp/bxxxxxx，继续查询下一个binlog
```

5. 注意：
   - 一般这种大事务生成的 binlog 会大于 1G，因此可以优先解析检查符合问题时间点的、容量最大的binlog，极大可能就在这些 binlog 中。
   - 如果业务量较大，binlog 过多，建议先结合 mysql-slow.log，以及 SQL `select * from performance_schema.events_statements_current order by TIMER_WAIT desc limit 10\G`、`select * from performance_schema.events_statements_history order by TIMER_WAIT desc limit 10\G` 的结果分析下。
6. 可通过重建从节点解决。

5. IP 相关

5.1. 数据库集群IP分配规则?

1. 自动分配IP：读写IP是从后面254往前分，节点IP都是从前面2开始向后面分。
2. 手动指定IP：指定的IP被占用, 创建的时候就会报错，有提示。
3. 高可用写IP和高可用读IP也是挂在网卡上的，和普通IP的区别是会飘移。

5.2. 高可用读IP、高可用写IP、主节点IP、从节点IP的作用分别是什么?

1. 主节点可读可写，从节点只读。
2. 高可用读IP可将请求在多个从节点和主节点（可配置是否分发给主节点）之间进行负载分发，以提高数据库的查询性能；发生从节点故障, 会自动从高可用读IP列表里面摘掉故障节点IP, 不影响业务查询。
3. 高可用写IP可以在主节点发生故障时自动切换到新的主节点上，提供高可用机制, 以减少故障时间。
4. 建议写业务连接高可用写IP，读业务连接高可用读IP。

5.3. 读写请求是否负载到主节点了呢?

1. 如果使用高可用写IP读写，则读写请求只负载到主节点。
2. 如果使用高可用读IP只读，当参数 `Load_read_requests_to_master` 为 `NO` 则读请求只负载到从节点，为 `YES` 则读请求会负载到包括主节点的每个正常节点。

6. MySQL 配置参数

6.1. MySQL 配置参数修改方式?

1. 暴露在控制台的参数，用户在控制台修改，若后续重启后不会失效。
2. 控制台修改参数，有些参数修改后会重启数据库服务，在参数配置里面有说明。
3. 对于用户在线更改的参数，或者是研发人员在后台临时帮忙改的配置文件，重启会失效。
4. 控制台没有暴露出来的参数，可反馈给 MySQL Plus 研发人员评估是否暴露在控制台，提供用户自己配置。
5. 控制台没有暴露出来的参数，若需要修改并持久化，请先评估影响，确认可以修改后，可按如下示例修改：

> 1. 执行 `sudo /etc/init.d/xenon-service addclusterparam '{"query_cache_type":"ON","join_buffer_size":"67108864"}'` 添加参数到自定义配置文件。
> 2. 执行 `sudo /etc/init.d/xenon-service showclusterparam` 查看是否添加成功，如输出结果中包含刚添加的参数，即为成功。
> 3. 在所有节点执行 `sudo touch /usr/local/etc/ignore_agent ; sudo /etc/init.d/xenon-service stop ; sudo /etc/init.d/xenon-service start ; sudo rm /usr/local/etc/ignore_agent` 重启服务后，参数即可生效。
> 4. 如重启后报错，请先看下 `/data/deploy/custom_params` 文件中内容是否为正确的 json 格式，同时看下 MySQL 错误日志确认启动失败原因。

6.2. 如何设置 MySQL 字符集编码?

1. 可以在控制台修改 `character_set_server` 参数项，修改完后需要重启生效。
2. 也可以用 root 身份用mysql命令行登陆到数据库进行修改，能及时生效，但是重启数据库之后改动会丢失。
3. MySQL 除了系统全局变量可以设定编码之外，还可以在 create database 的 SQL 中设置默认编码，也可以通过 alter table 修改表上或字段上设置编码，具体的 SQL 可以参照 MySQL 手册。

> https://dev.mysql.com/doc/refman/5.7/en/charset-applications.html
>
> http://dev.mysql.com/doc/refman/5.7/en/charset-table.html
>
> http://dev.mysql.com/doc/refman/5.7/en/charset-conversion.html

4. 客户端需要在连接数据库时使用 `set names` 语句来选择显示的编码，如果编码与数据库中存储的 charset 不一样，MySQL 会自动转换。

7. MySQL 日志

7.1. 如何获取 MySQL 慢日志、错误日志和审计日志?

1. MySQL Plus 控制台提供 slow-log, error-log 和 audit-log 同步和下载。
2. 用户可以在控制台服务栏里面点击 `同步日志`，选择下载文件个数，然后提交。
3. 在内网通过 FTP 下载：`wget ftp://node_ip/mysql-slow[mysql-error | mysql-audit] –ftp-user=ftpuser –ftp-password=ftppassword`。

7.2. 如何获取 MySQL binlog?

1. 在对应节点上将 binlog 文件拷贝到 ftp 目录并修改权限。示例如下：

```
sudo cp /data/mysql/mysql-bin.000008 /srv/ftp/home/ftpuser/
chmod 644 /srv/ftp/home/ftpuser/mysql-bin.000008
```

2. 用户可以在内网下载 `wget ftp://node_ip/mysql-bin.000008 --ftp-user=ftpuser --ftp-password=ftppassword` 。

7.3. 如何获取 MySQL general log?

1. MySQL 的 general log 默认没有开启，需要用户自己开启 `set global general_log=on;` 。
2. 用户开启之后，会在 /data/mysql 目录下生成以 hostname 命名的文件 `i-xxx.log` 。如果不确定，可以执行 `show variables like '%general_log_file';` 确认下。
3. 将它拷贝到 ftp 目录并修改权限：

```
cp /data/mysql/i-xxx.log /srv/ftp/home/ftpuser/general.log
chmod 644 /srv/ftp/home/ftpuser/general.log
chown ftpuser:ftp /srv/ftp/home/ftpuser/general.log
```

4. 操作完要退出 /data 目录，退出窗口 。
5. 用户可以在内网下载：`wget ftp://node_ip/general.log --ftp-user=ftpuser --ftp-password=ftppassword` 。

7.4. 如何查看 MySQL binlog?

- 使用 mysqlbinlog 工具查看，使用示例：

```
mysqlbinlog --base64-output=decode-rows -v mysql-bin.000001
```

8. 账户

8.1. 如何创建和获取数据库高级权限的账户?

1. MySQL Plus 1.5.2 版本开始已经支持用户在控制台创建高级账户，可点击 `添加账户` 创建。
2. 创建成功的账户可在 `账户` 页查看。

8.2. MySQL Plus `添加账户` 创建的账户有哪些权限?

普通账户权限列表：

> ["ALTER", "ALTER ROUTINE", "CREATE", "CREATE ROUTINE",
> "CREATE TEMPORARY TABLES", "CREATE VIEW", "DELETE",
> "DROP", "EXECUTE", "EVENT", "INDEX", "INSERT",
> "LOCK TABLES", "PROCESS", "RELOAD", "SELECT",
> "SHOW DATABASES", "SHOW VIEW", "UPDATE", "TRIGGER", "REFERENCES"]

高级账户权限列表：

> 高级权限的账户拥有全部权限

用户也可以执行SQL查询，比如：

```
mysql> select * from mysql.user where user='admin'\G
*************************** 1. row ***************************
                  Host: %
                  User: admin
           Select_priv: Y
           Insert_priv: Y
           Update_priv: Y
           Delete_priv: Y
           Create_priv: Y
             Drop_priv: Y
           Reload_priv: Y
```

```
          Shutdown_priv: Y
           Process_priv: Y
              File_priv: Y
             Grant_priv: Y
        References_priv: Y
             Index_priv: Y
             Alter_priv: Y
           Show_db_priv: Y
             Super_priv: Y
  Create_tmp_table_priv: Y
       Lock_tables_priv: Y
           Execute_priv: Y
        Repl_slave_priv: Y
       Repl_client_priv: Y
       Create_view_priv: Y
         Show_view_priv: Y
    Create_routine_priv: Y
     Alter_routine_priv: Y
       Create_user_priv: Y
             Event_priv: Y
           Trigger_priv: Y
Create_tablespace_priv: Y
               ssl_type:
             ssl_cipher:
            x509_issuer:
           x509_subject:
          max_questions: 0
            max_updates: 0
        max_connections: 0
   max_user_connections: 0
                 plugin: mysql_native_password
 authentication_string: *38C59B51E78BEBC39594D0A17FDE8D29BF7D78DD
      password_expired: N
 password_last_changed: 2019-09-23 10:01:51
     password_lifetime: NULL
        account_locked: N
1 row in set (0.00 sec)

mysql> show grants for 'admin'@'%';
+-----------------------------------------------------------------+
| Grants for admin@%                                              |
+-----------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'admin'@'%' WITH GRANT OPTION |
+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

Host就是你允许哪些host可以访问，`ON *.*` 就是允许访问所有的数据库。

8.3. 为什么添加数据库账户失败?

1. 建议检查添加的数据库账户是否是已经存在于这个集群中的同名用户。
2. 建议检查是否添加了高级权限账户。目前一个 MySQL Plus 仅允许添加一个高级权限的数据库账户。

8.4. 为什么添加数据库账户成功后，该账户无法正常访问数据库?

1. 建议检查在添加数据库账户时，是否在创建节点指定了 `加密认证` 选项为 YES，但是没有开启这个集群的 `SSL 传输加密` 服务功能。

8.5. 为什么删除数据库账户失败?

1. 建议检查删除的数据库账户是否就不存在于这个集群。
2. 建议检查集群各节点状态是否正常。

9. 服务监控

9.1. 控制台提供的 MySQL 服务相关的监控数据是怎么来的?

1. 通过在 MySQL 客户端执行 `show global status\G` 获取 MySQL 服务(查询, 事务, 锁信息)相关的监控参数。
2. 通过在 MySQL 客户端执行 `show slave status\G` 获取从库同步相关的监控参数。

9.2. 用户对某些监控项有疑问或想知道与 show status 结果的对应关系?

1. 比如，用户对监控项 `慢查询` 有疑问，可以先将 console 语言调整为英文，则可以看到 `慢查询` 对应的英文名是 `SLOW_QUERIES`。

9.3. MySQL Plus 支持监控服务吗?

1. MySQL Plus 1.4.3 版本开始提供了 zabbix agent 服务（zabbix 3.4)，参考 zabbix监控服务。

10. 备份

10.1. MySQL Plus 备份机制?

1. 若配置了自动备份，系统会在您选定的时间段里随机挑选一个时间点来进行备份。
2. 若没有配置自动备份，需要手动点击备份来自行备份。
3. MySQL Plus 每个备份链的默认备份节点是 30 个配额，默认 2 条备份链，每个备份链首次备份是全量备份，后面的都是增量备份。
4. 如果两个备份链都用满了，新增的自动备份会删除最早的一条备份链，然后重新创建一条新的全量备份链。
5. 默认使用主节点备份，如果主节点切换会重新生成一个新的全量备份链。
6. 备份的文件都是远程异地备份，有分布式存储集群来存储备份文件。

10.2. 备份的磁盘容量大小?

1. 对于 kvm 主机，只备份数据盘。现在云平台上创建的 MySQL Plus 节点均是 kvm 主机。
2. 对于 lxc 主机，除了备份数据盘外，还会备份系统盘。

10.3. 为什么 MySQL Plus 无法正常手动创建备份，或者自动创建备份失败呢?

1. 创建备份逻辑：选择 MySQL Plus 主节点，交由 appcenter 备份。因此，当执行备份任务时没有主节点会备份失败。
2. 排查 /data/log/backup.log 文件，如果出现 “get_backup_node_id failed to get ip of leader” 或者 “get_backup_node_id failed to get id of leader” 则说明执行备份时无主。可以交由 MySQL Plus 研发排查。
3. 若 /data/log/backup.log 中在对应备份时间有 “backup success” 则交由 appcenter 研发排查备份失败原因。

10.4. 基于备份创建集群需要多久？速度为多少?

1. 对于企业型e2集群大概速度为`350G/小时`。

11. 数据迁移

11.1. 什么情况下支持在线迁移?

1. 远端 MySQL 版本必须大于 MySQL 5.5。
2. 远端 MySQL 版本不得大于当前目标集群 MySQL 版本。

11.2. MySQL 5.5 如何迁移到 MySQL Plus?

1. MySQL 5.5 版本可以将数据 dump 出来，再导入 MySQL Plus，具体参考 MySQL Plus 数据迁移。

11.3. 在线迁移如何操作?

1. 关于在线迁移到 MySQL Plus 注意事项和操作步骤，可以参考产品文档:
    - 旧版（旧形态）在线迁移手册
    - 升级版（新形态）在线迁移手册

12. 异地灾备

12.1. 什么情况下支持异地灾备?

1. 源集群 MySQL Plus 需要大于等于 1.5.6 版本。
2. 源集群 MySQL 版本不得大于当前目标集群 MySQL 版本。

12.2. 异地灾备如何操作?

1. 关于 MySQL Plus 异地灾备功能注意事项和操作步骤，可以参考产品文档 异地灾备手册。

13. 审计

13.1. MySQL Plus 支持审计服务吗?

1. 默认没有开启审计服务，支持通过配置参数开启审计日志，可参考产品文档 配置参数 中审计相关的参数说明。

14. 扩容

14.1. 扩容是否会重启?

1. 只扩容磁盘，不会重启服务，不影响业务。
2. 扩容CPU或内存，会按串行方式扩容，对于 1.5.5 及更高的版本，会按照先从后主顺序扩容，期间只切主一次。

14.2. 更改主机类型耗时多久?

1. 更改主机类型，比如从超高性能型改为企业型e2，和扩容类似，会按串行方式执行，对于 1.5.5 及更高的版本，会按照先从后主顺序扩容，期间只切主一次。
2. 但又有所不同，由于不同主机默认绑定的磁盘也有所不同（比如超高性能型主机绑定的是超高性能型磁盘，企业型e2绑定的是企业型磁盘），因此，更改主机类型会涉及主机及磁盘的迁移，耗时会更久。
3. **目前现状（2020-07-09）:**
    - 16C32G 超高性能型更改为企业型e2 磁盘大小 500G，耗时如下:

```
cl-uaaqxb6i 实际占用238G 迁移耗时36分钟/节点，ratio:6.61，
https://boss.qingcloud.com/cloud/pek3/jobs/j-fy1pc263ehx/
cl-kye7w60h 实际占用277G 迁移耗时50分钟/节点，ratio:5.54，
https://boss.qingcloud.com/cloud/pek3/jobs/j-80i27z7wmab/
cl-t2bonfzi 实际占用391G 迁移耗时60分钟/节点，ratio:6.51，
https://boss.qingcloud.com/cloud/pek3/jobs/j-pa3sasn398r/
```

15. 升级

15.1. 升级方式是什么?

1. MySQL Plus 从 1.5.9 版本已经支持了集群节点滚动升级。
2. 节点升级顺序为先从后主。

15.2. 升级过程中影响业务时间大概多久?

1. 对于三个及以上节点数的集群，滚动升级中只会在最后升级主节点时造成主从切换，影响时间极短。
2. 对于两个节点的集群，滚动升级中由于要保证数据一致性，影响时间稍长。

15.3. MySQL Plus 是否能直接升级到 MySQL Plus NeonSAN 版本?

1. MySQL Plus 不支持从普通版本升级到 NeonSAN 大容量型版本。
2. 用户可以通过 MySQL Plus 在线迁移服务来平滑迁移到 NeonSAN 版本。

16. 常见异常处理

16.1. 如何判断服务是否正常?

1. 执行 `/opt/xenon/xenoncli cluster status`，如果 `IO/SQL_RUNNING` 是 `true/true` 且 `Raft` 列有 `LEADER`，则表明 MySQL 服务正常。

```
+-----------------+----------------------------------+---------+---------
+-------------------------+--------------------+----------------+---------
---------+
|        ID        |               Raft                | Mysqld  | Monitor |
         Backup          |        Mysql        | IO/SQL_RUNNING |
MyLeader     |
+-----------------+----------------------------------+---------+---------
+-------------------------+--------------------+----------------+---------
---------+
| 192.168.0.7:8801 | [ViewID:10872 EpochID:0]@FOLLOWER | RUNNING | ON      |
state:[NONE]
            | [ALIVE] [READONLY]  | [true/true]    | 192.168.0.6:8801 |
|                  |                                    |         |         |
LastError:               |                     |                |
       |
+-----------------+----------------------------------+---------+---------
+-------------------------+--------------------+----------------+---------
---------+
| 192.168.0.6:8801 | [ViewID:10872 EpochID:0]@LEADER   | RUNNING | ON      |
state:[NONE]
            | [ALIVE] [READWRITE] | [true/true]    | 192.168.0.6:8801 |
|                  |                                    |         |         |
LastError:               |                     |                |
       |
```

```
+------------------+----------------------------------+---------+---------
+-------------------------+--------------------+---------------+---------
---------+
| 192.168.0.5:8801 | [ViewID:10872 EpochID:0]@FOLLOWER | RUNNING | ON      |
state:[NONE]
            | [ALIVE] [READONLY]  | [true/true]    | 192.168.0.6:8801 |
|                  |                                  |         |         |
LastError:               |                     |               |
       |
+------------------+----------------------------------+---------+---------
+-------------------------+--------------------+---------------+---------
---------+
(3 rows)
```

2. 执行 `tailf /data/log/checkservice.log`，如果本地 healthcheck 最后输出的是 `check_service success`，则表明 healthcheck 正常。（console `节点服务状态` 要等连续两次 healthcheck 成功后才会恢复正常）

16.2. 如何确认 MySQL Plus 主从复制是否正常?

1. 执行 `/opt/xenon/xenoncli cluster status`，如果 `IO/SQL_RUNNING` 是 `true/true` 且 `Raft`列有`LEADER`，则服务正常。
2. 执行 `/opt/xenon/xenoncli cluster gtid` 指令，如果 LEADER 的 `Executed_GTID_Set` 各个 GTID 段的数值都包含所有从节点对应段的数值（`1-3628`包含`1-3600`），且在业务有写入的情况下连续多次执行，所有节点的某个GTID段（`leader节点的/data/mysql/auto.conf中对应的server-uuid所在的段`）都同步变化，即为正常。

```
+------------------+----------+-------+-----------------------------------
------------+-----------------------------------------------------+
|        ID        |   Raft   | Mysql |                Executed_GTID_Set
           |                 Retrieved_GTID_Set                  |
+------------------+----------+-------+-----------------------------------
------------+-----------------------------------------------------+
| 192.168.0.7:8801 | FOLLOWER | ALIVE | 063a0edb-dab5-11e9-9a2c-
525496bdd51a:1-3628,
   | cdd35972-daa7-11e9-a328-5254967940fe:116271-116272 |
|                  |          |       | 3a932f66-f60d-11e9-b4fd-
525496bdd51a:1-15003,
  |                                                     |
|                  |          |       | 92937469-daa7-11e9-937d-
52549634f4b9:1-123540,
 |                                                    |
|                  |          |       | cdd35972-daa7-11e9-a328-
5254967940fe:1-116272   |
|
+------------------+----------+-------+-----------------------------------
------------+-----------------------------------------------------+
| 192.168.0.5:8801 | FOLLOWER | ALIVE | 063a0edb-dab5-11e9-9a2c-
525496bdd51a:1-3628,
   | cdd35972-daa7-11e9-a328-5254967940fe:116271-116272 |
|                  |          |       | 3a932f66-f60d-11e9-b4fd-
525496bdd51a:1-15003,
  |                                                     |
|                  |          |       | 92937469-daa7-11e9-937d-
52549634f4b9:1-123540,
 |                                                    |
```

```
|                   |          |       | cdd35972-daa7-11e9-a328-
5254967940fe:1-116272   |
|
+-------------------+----------+-------+-----------------------------------
------------+-----------------------------------------------------+
| 192.168.0.6:8801 | LEADER   | ALIVE | 063a0edb-dab5-11e9-9a2c-
525496bdd51a:1-3628,
   |                                                          |
|                   |          |       | 3a932f66-f60d-11e9-b4fd-
525496bdd51a:1-15003,
  |                                                           |
|                   |          |       | 92937469-daa7-11e9-937d-
52549634f4b9:1-123540,
 |                                                           |
|                   |          |       | cdd35972-daa7-11e9-a328-
5254967940fe:1-116272   |
|
+-------------------+----------+-------+-----------------------------------
------------+-----------------------------------------------------+
(3 rows)
```

16.3. checkservice.log 报错 `try lock file failed`

1. 如果反复报该错误，需要查看 `/tmp/` 下是否存在 `xxxxxxxx.lock`（`xxxxxxxx` 是个日期，比如20180518）。
2. 如果存在，执行 `unlink xxxxxxxx.lock` 删除。

16.4. checkservice.log 报错 `check service is ignored`

1. 须删除 `/usr/local/etc/ignore_agent` 文件。

16.5. checkservice.log 报错 `disk is nearly full`

1. 当磁盘占用量大于等于95%时会报该错。
2. 建议扩容，清理或减少保存的 binlog 数量。

16.6. checkservice.log 报错 `env is not inited`

1. 该错误一般是start指令失败，可通过 `/data/log/start.log` 查看具体原因。
2. 如果 start.log 中报错信息为 cluster_node_ips is none，表明 AppCenter 的 confd 服务有问题（/etc/xenon/xenon.conf中cluster_nodes为空）。可进一步查看/opt/qingcloud/app-agent/log/confd.log，确认具体的报错。 如果最近报错类似 `ERROR connection to [http://192.168.253.3:80], error: [Get http://192.168.253.3:80/: dial tcp 192.168.253.3:80: getsockopt: connection timed out]`，请联系 AppCenter 研发解决。

16.7. checkservice.log 报错 `mysql is not running`

1. 可能是 MySQL 无法启动，查看 /data/mysql/mysql-error.log，确认具体报错信息。
2. 也可能是手动关闭了 MySQL 自动拉起服务且关闭了 MySQL。可执行 `/opt/xenon/xenoncli mysql startmonitor` 启用 MySQL 的自动拉起功能。

16.8. checkservice.log 报错 `xenon peers.json is error`

1. 当 `/data/raft/peers.json` 文件不存在时，会尝试执行 `/opt/xenon/xenoncli cluster add` 将集群节点信息注册到当前xenon，如果执行失败，则会报该错误。

2. 查看 `/etc/xenon/xenon.conf` 中的字段 `cluster_nodes`，提取其IP地址，比如分别为ip1、ip2、ip3。
3. 执行 `/opt/xenon/xenoncli cluster add ip1:8801,ip2:8801,ip3:8801`。
4. 执行 `/opt/xenon/xenoncli cluster status`，结果为三列且ID分别为ip1、ip2、ip3，则表示问题修复。

16.9. checkservice.log 报错 `cluster has error node config`

1. 该报错表示没有 leader 或 `cluster status` 结果中的列数与 `/etc/xenon/xenon.conf` 中 `cluster_nodes` 的节点数量不一致。
2. 执行 `/opt/xenon/xenoncli cluster status`，查看是否有leader，如果没有，需要分析 `/data/log/xenon.log` 中错误日志。
3. 如果有leader：
    - 查看 `/etc/xenon/xenon.conf` 中的字段 `cluster_nodes`，提取其IP地址，比如分别为ip1、ip2、ip3。
    - 执行 `/opt/xenon/xenoncli cluster add ip1:8801,ip2:8801,ip3:8801`。
    - 执行 `/opt/xenon/xenoncli cluster status`，结果为三列且ID分别为ip1、ip2、ip3，则表示问题修复。

16.10. checkservice.log 报错 `lvs master/slave vip is brain split`

1. lvs master 即为高可用读IP所在的节点。
2. 该错误只会出现在 lvs master/slave，表示 lvs master/slave 探测到其他节点也设置了 rvip（高可用读IP），即 rvip 发生了脑裂。
3. 正常情况下，自动修复（tackleservice）后就正常了。 但是，对于私有云或防火墙中禁用了 vrrp 协议的vxnet，需要在防火墙中放开 vrrp 协议，否则会一直报错的。

16.11. checkservice.log 报错 `lvs master/slave no lvs list`

1. 该错误表示 lvs master/slave 的 rvip 分发列表为空，可通过 `sudo ipvsadm -ln` 查看。
2. 需要先执行 `/opt/xenon/xenoncli cluster status` 和 `/opt/xenon/xenoncli cluster gtid` 查看 `集群状态是否正常` 和 `从节点的GTID中是否存在local commit（本地事务）`。
3. 如果集群状态异常或从库存在本地事务，请参考本文 `危险操作-重建`。

16.12. checkservice.log 报错 `wvip is not active`

情形一：网络问题

> 1. 先执行 `/opt/xenon/xenoncli cluster status` 查看集群状态，如果有超过半数节点为 `UNKNOWN`，可通过ping查看下集群节点之间网络是否互通。
> 2. 若网络有问题，联系IAAS网络组修复网络。
> 3. 再次查看集群状态和checkservice是否正常。

情形二：xenon主节点频繁当选、降级（并行复制问题）

> 1. 查看MySQL错误日志 或 xenon日志，如果其中有报错信息 `The slave coordinator and worker threads are stopped, possibly leaving data in inconsistent state. A restart should restore consistency automatically, although using non-transactional storage for data or info tables or DDL queries could lead to problems. In such cases you have to examine your data (see documentation for details).`，则表明是大事务引起的并行复制报错。
> 2. 在从库执行mysql -uroot -e"show processlist"，发现有特别多的 `STOP SLAVE IO_THREAD`，则表明从库MTS在等待大事务的重放，无法stop slave io线程，需要将文

> 件`/etc/mysql/my.cnf`中的`slave_parallel_workers`改为0，然后执行`/opt/xenon/xnonecli mysql shutdown 或 kill`关闭mysql，等几秒mysql会自动拉起来，这之后复制就可以正常进行，主节点成功选出。
> 3. 再对比集群节点GTID，以决定是否重建。

情形三：xenon无法选主（超过半数节点的mysql crash无法启动，极少出现）

> 执行`/opt/xenon/xenoncli cluster gtid`，能看到集群所有节点所有xenon存活，所有节点`Executed_GTID_Set`都不为空，若集群长时间无法自行选主，则按紧急情况处理：
>
> 1. 从集群（节点A、B、C）中选出`Executed_GTID_Set`最大的一个节点，比如节点A 。
> 2. 在所有节点执行`sudo touch /usr/local/etc/ignore_agent` 。
> 3. 在节点B、C执行`/opt/xenon/xenoncli raft disable` 。
> 4. 在节点A执行`/opt/xenon/xenoncli cluster remove 节点B的IP:8801,节点C的IP:8801` 。
> 5. 在节点A执行`/opt/xenon/xenoncli raft trytoleader`，由于节点A只有自身，肯定可以当选为leader。
> 6. 待节点A当选为leader后，在节点A执行`/opt/xenon/xenoncli cluster add 节点B的IP:8801` 。如果节点B的IO或SQL线程报错，则重建节点B，否则，在节点B执行`/opt/xenon/xenoncli raft enable` 。
> 7. 确定节点B的IO和SQL都为YES的情况下，在节点A执行`/opt/xenon/xenoncli cluster add 节点C的IP:8801` 。如果节点C的IO或SQL线程报错，则重建节点C，否则，在节点C执行`/opt/xenon/xenoncli raft enable` 。
> 8. 所有节点执行`sudo rm /usr/local/etc/ignore_agent` 。

16.13. checkservice.log 报错 `mysql status is error`

1. 该错误表示`leader节点只读`或`从库show slave status中SQL线程出错`或`当前从节点存在local commit`。
2. 若是`leader节点只读`，1.5.5 及之后的版本会自动订正主节点只读状态。
3. 若是后两种情况，需要先执行`/opt/xenon/xenoncli cluster status`和`/opt/xenon/xenoncli cluster gtid`查看`集群状态是否正常`和`从节点的GTID中是否存在local commit（本地事务）`。
4. 如果集群状态异常或从库存在本地事务，请参考本文`危险操作-重建`。

16.14. checkservice.log 报错 `mysql system user is error`

1. 该错误表示 MySQL 某个系统账户（`root`，`mysql.sys`，`mysql.session`，`qc_repl`或`repl_xxxx`）不存在。
2. 如果缺少`mysql.session`，可执行`mysql -uroot -e"set global super_read_only=off"; mysql_upgrade -uroot --force`。（**对于1.5.6及之后版本，关闭再启动集群会自动修复**）

16.15. checkservice.log 报错 `slave gtid is bigger than leader`

1. 该错误表示从库存在local commit 本地事务。
2. 解决方案：
   - 首先查看`/opt/xenon/xenoncli cluster status`，如果存在local commit的slave节点的`IO/SQL_RUNNING`是`true/true`且基本不落后于主节点（Seconds_Behind_Master间歇性为0），则尝试将本地事务拿到 leader：

     > 1）任意节点执行`grep "user" /etc/xenon8801.json -nr -B 2`查看复制账户、密码。
     >
     > 2）任意节点执行`grep "port" /etc/mysql/my.cnf -nr`查看 MySQL 的端口号。

> 3）在leader节点执行`stop slave; change master to master_host='存在local commit的节点IP', master_user='复制账户', master_password='复制账户密码', master_port=端口号, master_auto_position=1; start slave;`
>
> 4）在leader节点连续多次（连续两次之间等一会儿）执行`show slave status\G`，查看IO和SQL线程：
>
> > a. 若IO和SQL线程都为YES，且`Executed_GTID_Set`还未追上local commit的从库（即从库的本地事务还未拿到），则等待一会继续执行`show slave status\G`查看。
> >
> > b. 若IO和SQL线程都为YES，且`Executed_GTID_Set`追上了local commit的从库（即从库的本地事务已经拿到），则在leader节点执行`stop slave; reset slave all;`，**问题解决**。此时如果local commit的从库状态为`INVALID`，则在该从库执行`/opt/xenon/xenoncli raft disable; /opt/xenon/xenoncli raft enable`，集群即可恢复正常。
> >
> > c. 若IO或SQL线程为NO，则表示无法获取从库的本地事务，则在leader节点执行`stop slave; reset slave all;`，进入第下一步执行重建。

- 如果第上一步无法将本地事务拿到主库则：

> 1）首先备份涵盖问题时间点的binlog。
>
> 2）参考本文`危险操作-重建`重建节点。

17. 危险操作

17.1. 重建

17.1.1. 重建指令

```
# 要在有问题的follower节点执行，如需指定重建的数据源，则重建指令改为 xenoncli mysql rebuildme --from=ip:8801
sudo touch /usr/local/etc/ignore_agent; /opt/xenon/xenoncli mysql rebuildme ;
sudo rm /usr/local/etc/ignore_agent
```

17.1.2. 如何判断当前节点是否需要重建?

> 本部分情形一、情形四最简单，这两种情况只要确认GTID没问题，一般可以重建。情形二、情形三可找研发协助。

执行`/opt/xenon/xenoncli cluster status`和`/opt/xenon/xenoncli cluster gtid`查看集群状态和GTID信息。根据结果的不同，操作亦有不同：

情形一：节点总数为3或5，有超过半数（5的`超过半数`为3，3的`超过半数`为2）节点cluster status为`true/true`，此时集群还能正常提供服务。则可能需要`逐个`重建异常节点（非`true/true`）。

> **重建前先检查异常节点的gtid：**
>
> 1. 如果异常节点不存在local commit，可以直接重建。
> 2. 如果异常节点存在local commit（比leader多一些事务），一般这种情况下节点状态为INVALID，同时该节点比leader多的事务只在该节点有（其他主、从都没有），则可以重建。

情形二：节点总数为3或5，有少于半数节点为`true/true`（节点总数为3时少于2，节点总数为5时少于3），且加上为`true/false`（表示从库可以正常接收，只是无法重放）状态的节点数量后`超过半数`。

> 按**情形一**的方式`检查异常节点的gtid`，确保`异常节点的mysql可连接（如果不可连接，请查看错误日志并找值班人员协助），没有local commit或者有local commit但该节点比leader多的事务只在该节点有`的情况下：
>
> 1. 先重建集群状态为`false/false`或`false/true`的节点。
> 2. 如果存在状态为`true/false`的节点，对其重建。

情形三：节点总数为3或5，有少于半数节点为`true/true`（节点总数为3时少于2，节点总数为5时少于3），且加上为`true/false`状态的节点数量后`依然不超半数`。这种情况下，集群主节点的写入事务可能会hang。

> 按**情形一**的方式`检查异常节点的gtid`，确保`mysql可连接（如果不可连接，请查看错误日志并找值班人员协助），没有local commit或者有local commit但该节点比leader多的事务只在该节点有`，然后按如下步骤处理：
>
> **对于1.5.7及之后的版本，如果`少于半数节点的MySQL存活（cluster gtid中超过半数节点的GTID不为空）`，按如下步骤重建：**
>
> 1. 在leader执行`/opt/xenon/xenoncli raft disablechecksemisync`。
> 2. 在leader执行`mysql -uroot -e"SET GLOBAL rpl_semi_sync_master_enabled=OFF; SET GLOBAL rpl_semi_sync_master_timeout=5000;"`。
> 3. 重建有问题的follower节点：`sudo touch /usr/local/etc/ignore_agent; /opt/xenon/xenoncli mysql rebuildme --from=数据源IP地址:8801 ; sudo rm /usr/local/etc/ignore_agent;`。
> 4. 在leader执行`/opt/xenon/xenoncli raft enablechecksemisync`。
>
> **PS：** 如果迟迟无法选主，可以先在有最新数据的节点按如下步骤让其当选：
>
> 1. 在预期主节点执行`mysql -uroot -e"stop slave;reset slave all;"`，等待十秒后，若能当选，则继续第2步。反之，若预期主节点无法当选且集群是3节点，则在预期主节点执行`sudo touch /usr/local/etc/ignore_agent; /opt/xenon/xenoncli cluster remove 某个坏掉的slave1节点IP:8801`，这样最多等一分钟该节点就会当选；
> 2. 重建slave2节点（`xenoncli cluster remove`中未删除的节点）；
> 3. 在新主节点（也就是前面的预期主节点）执行`/opt/xenon/xenoncli cluster add 某个坏掉的slave1节点IP:8801`，然后重建坏掉的slave1节点。
> 4. 删除ignore_agent：`sudo rm /usr/local/etc/ignore_agent`。

情形四：节点总数为2，从节点状态不为`true/true`。

> 确保mysql可连接的情况下，按**情形一**的方式`检查异常节点的gtid`：
>
> 1. 如果异常从节点有本地事务，需要用户确认以哪个节点数据为准。如果以当前从节点为准，则在该从节点执行`/opt/xenon/xenoncli raft trytoleader`，确保当前从节点变为新主后，重建旧主节点。
> 2. 如果异常从节点没有本地事务，则可以直接重建。

17.1.2. 为何重建失败?

1. 客户集群防火墙禁用了22端口。
2. 3节点集群，有2个节点IO线程都异常（false）。
3. 查看重建源和目的的节点的`/data/log/xenon.log`确认出错原因。

17.2. 磁盘满需要清理binlog或relay log

说明：

> 1. 对于主从节点，都可以清理binlog。

> 2. 对于从节点，不要直接清理relay log。

一般情况下，建议先扩容磁盘。

步骤：

1. 查看`/data/mysql/`目录中`mysql-bin.xxxxxx`格式的文件，如果过多：

   > a）通过`echo > mysql-bin.xxxxxx`清理**数字最小的几个binlog**，让集群先恢复正常（`cluster status`所有节点都为`true/true`）。
   >
   > b）让客户通过配置参数页面调小`expire_logs_days`，减少保存的binlog天数。

2. 反之，如果binlog很少而relay log过多（一般发生在从节点），这种一般是因为主节点大事务产生的逻辑日志过大（比如大表DDL）导致写满从库磁盘。

   > a）这种要建议`扩容磁盘`，同时`优化下业务`，尽可能`将大事务拆分`。
   >
   > b）如果不接受的话，只能先帮助重建从库了。

17.3. 手动切主

对于1.5.0及之后的版本，可以在控制台集群详情页面运行`【指定Master节点】`服务来切换。

18. 其他疑问

18.1. MySQL Event Scheduler 不能正常工作?

1. 在控制台可以更改 event_scheduler 参数来启用和关闭 MySQL Event Scheduler 。
2. 查看创建的event的方法：

```
select * from mysql.event;
```

3. event 成功执行需要的条件是：
   - event_scheduler 参数设置为on或者1
   - mysql.event 里面的 event 的 status 是 enable
4. 创建event的时候可以指定为 enable 语法：

```
CREATE
[DEFINER = { user | CURRENT_USER }]
EVENT
[IF NOT EXISTS]
event_name
ON SCHEDULE schedule
[ON COMPLETION [NOT] PRESERVE]
[ENABLE | DISABLE | DISABLE ON SLAVE]
[COMMENT 'string']
DO event_body;
```

5. 还可以通过 SQL 命令来关闭或者打开一个 event：

```
ALTER EVENT ..... | ENABLE | DISABLE
```

6. 上面 4 和 5 能成功启动一个 event， event 在主库 ENABLE 后，在从库上依旧是 DISABLE （这是MySQL Event Scheduler的机制）。
7. 所以要保证 EVENT 正常运行，需要在主库上执行：

    - SET GLOBAL event_scheduler = 1; (在控制台可以设置)
    - ALTER EVENT xxx ON COMPLETION PRESERVE ENABLE;
8. 如果主节点切换了，就算之前所有的节点都设置了：SET GLOBAL event_scheduler = 1， 但是mysql.event里面的 status 还是 disabled，是会导致 event 不运行的，需要再在新的主节点上去启动 event。

18.2. 哪里可以查阅 MySQL Plus 完整的产品文档?

MySQL Plus 产品分为新旧两个产品形态，产品详细介绍见官网产品文档：

> 旧版（旧形态）产品文档
>
> 升级版（新形态）产品文档