Klustron参数优化设置

**注意**：

 如无特别说明，文中的版本号可以使用任何已发布版本的版本号代替。所有已发布版本详见：<http://doc.klustron.com/zh/Release_notes.html>

**本文目标：**

默认配置的Klustron集群，一些重要的参数相对都比较保守，而实际客户安装Klustron的硬件环境各异，为了更好的发挥Klustron的性能，有必要针对特定的环境和业务负载的特性，对这些重要的性能参数进行调整，以达到业务性能指标要求。本文旨在描述这些相关的参数的意义和用途，并给出设置建议。

1. 计算节点(PG)重点参数：
   1. shared\_buffers：KunlunBase的计算节点本地不存储数据，从存储节点获取的数据不存储在buffer pool中，而是流式地（增量地）按需从存储节点获取。因此计算节点的buffer pool仅仅用于存储元数据表的数据页，总量也就几MB，因此我们设置该值较小。如果用户使用的若干个临时表的总数据量很大，那么可以把temp\_buffers设置更大。
   2. statement\_timeout：在计算节点被用来控制SQL语句执行时长，单位是ms, 如果设置为10000，那就意味着语句最多可以执行 10000ms = 10s，通常情况下，建议此值设为：6000000。
   3. mysql\_read\_timeout，mysql\_write\_timeout：mysql\_read\_timeout 和 mysql\_write\_timeout：计算节点与存储节点/元数据节点之间的通信收发（读写）超时。读取超过 mysql\_read\_timeout 或者写入超过 mysql\_write\_timeout 那么计算节点使用的 mysql 客户端库就会报错并且从读取/写入等待中返回，这样SQL语句执行就提前终止了。如果一条发送给计算节点的一条insert语句会插入 100 万行数据，或者一条 select 语句会从存储节点返回上百万行数据，那么最好增加这两个变量的值，它们默认是 50 秒，建议设为比如:1200秒。另外，这种情况下还要增大 mysql\_max\_packet\_size 变量，确保这样的超大数据包可以正确发送给存储节点。
   4. lock\_timeout：计算节点等待表锁的时间，并发执行的增删改查语句对表的锁是相容的，不需要等待锁。但是如果某个alter table语句正在执行，那么同一个计算节点上其他连接中无法执行针对这个表的 DML 语句，它们最多等这么久，还拿不到锁就会报错返回（默认 100 秒），建议设为比如：1200000秒。
   5. log\_min\_duration\_statement：超过这个时间的语句会作为慢查询，会记录到日志文件中。如果要在每条insert语句中插入数万行甚至更多，那么一定要把这个变量增大，否则会在日志文件中记录大量数据，导致计算节点磁盘空间用尽（默认 10 秒），建议设为比如：1200000秒。
   6. effective\_cache\_size：是计算节点能够使用的最大缓存，这个数字应该足够大，比如物理内存的50%以上。
   7. work\_mem：每个计划节点执行时可以使用的内存量。在执行排序操作时，会根据work\_mem的大小决定是否将一个大的结果集拆分为几个小的和 work\_mem差不多大小的临时文件，显然拆分的结果是降低了排序的速度，因此增加work\_mem有助于提高排序的速度，对于OLTP类型的负载，该值通常设置为8M是足够的，但对于OLAP的负载，该值设为物理内存的2% -4%也是合理的。
   8. wal\_buffers：是预写日志(wal)缓冲区，缓冲区的默认大小由wal\_buffers设置，最初为16MB，对于KunlunBase来说，由于用户数据不存储在计算节点本地，因此只有修改vacuum元数据表才会需要写redo log到WAL buffer，因此该参数不需要很大，默认值够用，设置很大之后对系统性能不会有明显的提升。
   9. autovacuum：autovacuum是postgresql数据库是一个后台进程，随数据库自启动，autovacuum作用如下：

1、清理表中的过期死元组

2、防止业务表膨胀

3、更新表的统计信息以供优化器使用

4、autovacuum launcher使用stats collector后台进程收集统计信息来确定autovacuum候选列表

建议将此值设为false。

以上参数，都可以用alter system set <参数名>=<参数值> 的方式进修改，但随后，需要重启计算节点服务来让参数生效，示例如下：

pg\_ctl restart -D /kunlun/server\_datadir/47001

1. 存储节点(MySQL)重点参数：
   1. innodb\_buffer\_pool\_size：指定该 Klustron-Storage实例的 Innodb buffer pool 大小, 对于元数据节点，该值默认 128MB，对于生产环境可能偏小，一般需要根据需要进行设置，最大值取决于CPU的架构。在32-bit平台上，最大值为2\*\*32 -1,在64-bit平台上最大值为2\*\*64-1。当缓冲池大小大于1G时，将innodb\_buffer\_pool\_instances设置大于1的值可以提高服务器的可扩展性。大的缓冲池可以减小多次磁盘I/O访问相同的表数据。在专用数据库服务器上，可以将缓冲池大小设置为服务器物理内存的80%。
   2. lock\_wait\_timeout：mysql server 层的锁超时变量，等待server层的表锁的最大时间。如果一个 DDL 语句在 alter table, 那么所有对该表做 DML 语句的事务会阻塞等待最多这么多表，还得不到表锁就会返回错误。在 MySQL8.0 时代，加列和加索引这种最常见的曾经要锁住全表才能完成的操作已经不需要全表长期锁定了，已经变成了 online ddl，因此默认一般来说足够了,对于性能测试，不管是TPCC还是TPCH，建议此值可设为： 1200秒。
   3. innodb\_lock\_wait\_timeout：mysql innodb 的锁超时变量，等待 innodb 行锁的最大时间，超过了那么 DML 语句就会报错返回。如果要做全表更新，并且表的数据量非常大，比如几百 GB 甚至更多，那么 update 语句会锁住大量的行很长时间，此时其他事务通常会发生锁超时，除非增大了该参数（默认 20 秒），这种情况下建议设为：1200秒。
   4. fullsync\_timeout：该参数的完整释义，请参考：http://doc.klustron.com/zh/Klustron\_Storage\_Cluster\_Fullsync.html?h=fullsync\_timeout 获得完整的理解，该参数建议设置为：1200000。
   5. enable\_fullsync：该参数的完整释义，请参考：http://doc.klustron.com/zh/Klustron\_Storage\_Cluster\_Fullsync.html?h=enable\_fullsync 获得完整的理解。一般情况下务必将该参数设置为true。如果在TPCC/TPCH测试时要灌入大量的数据（比如大于10G以上），为提升数据加载速度，建议将该参数设置为：false。
   6. innodb\_flush\_log\_at\_trx\_commit：是 InnoDB 引擎特有的，ib\_logfile的刷新方式，取值:0/1/2，innodb\_flush\_log\_at\_trx\_commit=0，表示每隔一秒把log buffer刷到文件系统中(os buffer)去，并且调用文件系统的“flush”操作将缓存刷新到磁盘上去。也就是说一秒之前的日志都保存在日志缓冲区，也就是内存上，如果机器宕掉，可能丢失1秒的已完成提交的事务数据。innodb\_flush\_log\_at\_trx\_commit=1，表示在每次事务提交的时候，都把log buffer刷到文件系统中(os buffer)去，并且调用文件系统的“flush”操作将缓存刷新到磁盘上去。这样的话，数据库对IO的要求就非常高了。尽管InnoDB会合并同时提交的一组事务只做一次刷盘，这些刷盘带来的IO负载也是比较高的。如果底层的硬件提供的IOPS比较差，那么MySQL数据库的并发很快就会由于硬件IO的问题而无法提升。innodb\_flush\_log\_at\_trx\_commit=2，表示在每次事务提交的时候会把log buffer刷到文件系统的Page cache中去，但并不会立即刷写到磁盘。如果只是MySQL数据库挂掉了，由于文件系统没有问题，那么对应的事务数据并没有丢失。只有在数据库所在的主机操作系统损坏或者突然掉电的情况下，数据库的事务数据可能丢失1秒之内的事务数据。这样的好处，减少了事务数据丢失的概率，而对底层硬件的IO要求也没有那么高(log buffer写到文件系统中，一般只是从log buffer的内存转移的文件系统的内存缓存中，对底层IO没有压力)。如果在TPCH测试时要灌入大量的数据（比如大于10G以上），为提升速度，建议将该参数设置为：2。
   7. sync\_binlog：MySQL提供一个sync\_binlog参数来控制数据库的binlog刷到磁盘上去，默认，sync\_binlog=0，表示MySQL不控制binlog的刷新，由文件系统自己控制它的缓存的刷新。这时候的性能是最好的，但是风险也是最大的。因为一旦系统Crash，操作系统page cache中缓存的binlog文件更新都会被丢失。如果sync\_binlog>0，表示每sync\_binlog次事务提交，MySQL调用文件系统的刷新操作将缓存刷下去。最安全的就是sync\_binlog=1了，表示每次事务提交，MySQL都会把binlog刷下去，是最安全但是性能损耗最大的设置。尽管MySQL会合并同时提交的一组事务只做一次刷盘，这些刷盘带来的IO负载也是比较高的。这样的话，在数据库所在的主机操作系统损坏或者突然掉电的情况下，系统不会丢失任何确认完成提交的事务。对于高并发事务的系统来说，“sync\_binlog”设置为0和设置为1的系统写入性能差距可能高达5倍甚至更多。根据需要，sync\_binlog可能设置为并不是最安全的1，而是100或者是0。这样牺牲一定的一致性，可以获得更高的并发和性能。在性能测试中，尤其是TPCH类别的测试，建议设置为：0。
   8. max\_binlog\_size：指定了单个二进制日志文件的最大值，如果超过该值，则产生新的二进制日志文件，后缀名+1，并记录到.index文件。从MySQL 5.0开始的默认值为1073 741824，代表1G。建议使用KunlunBase配置模板的默认值500MB。

以上参数，可以通过：

set global <参数名>=<参数值>;

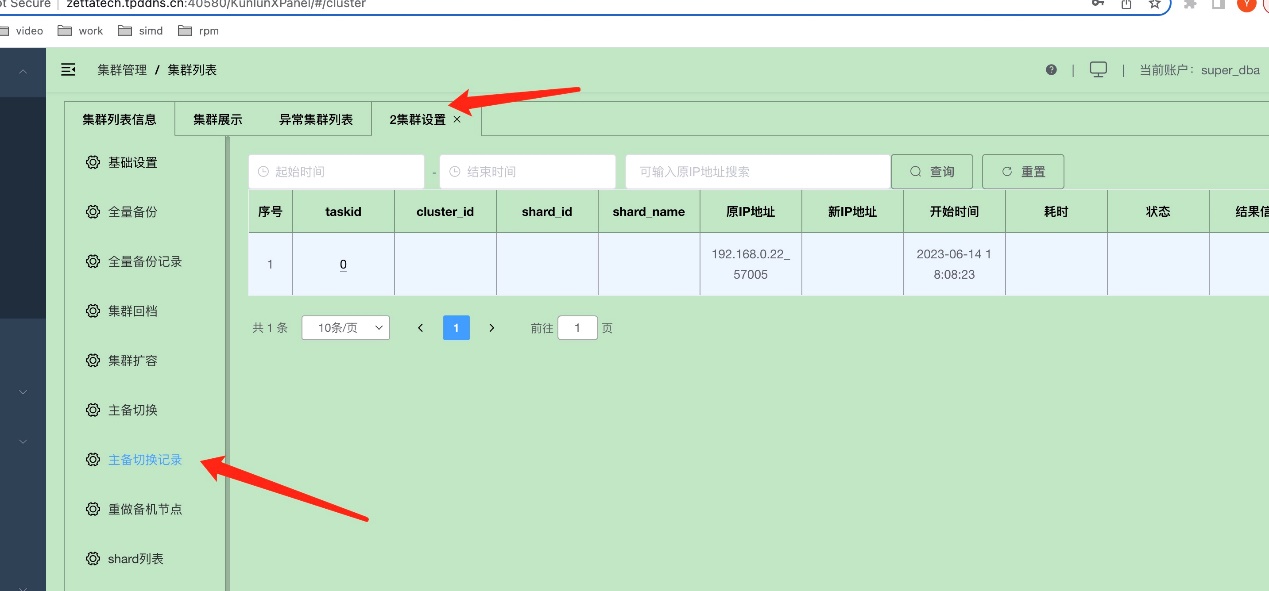
临时生效，则如果存储节点重启，参数将失效，如果需要永久设置，需要在my.cnf中将上述参数写入保存。

附：连接存储节点的方式：mysql -h <存储节点主机IP> -P <存储节点端口> -upgx -ppgx\_pwd

另外，在进行TPCC/TPCH测试场景下，数据灌入时，为提升速度，减少不必要的故障，建议对shard进行免切设置，该参数的概念是：设置免切，就是这个shard在设置免切时间内，shard主节点异常不会触发主备切换。比如设置10800秒（3小时），从点”设置”开始计时，在10800秒内，不会发生切换。在设置行为需要在XPanel上完成操作。

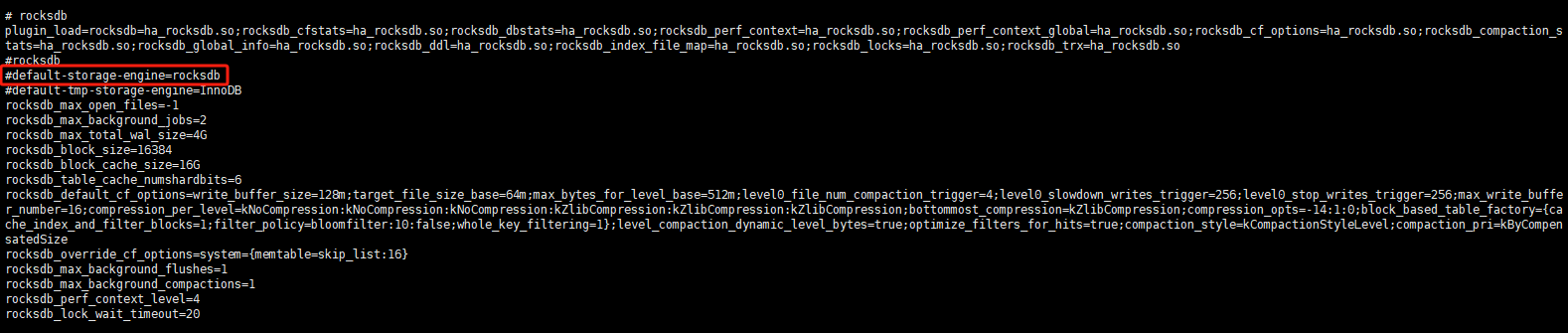


监控是否发生过主备切换，通过“主备切换记录”查看：



* 1. 存储节点存储引擎参数说明：

默认情况下，Klustron存储节点所使用的存储引擎为innodb，但在有一些使用场景下，为了提高数据装载入库的效率，比如对大量日志记录入库的场景，可以将存储节点的存储引擎设置为rocksdb，通过修改存储节点的mysql配置文件可启用rocksdb存储插件，该文件的路径，示例如：vi /kunlun/storage\_datadir/57001/data/57001.cnf



将 #default-storage-engine=rocksdb 前面的“#”号去掉，保存退出后，重启存储节点 MySQL服务启用rocksdb存储引擎，但针对具体配置及环境，还有其他一些rocksdb的相关参数设定值得进行调整，说明如下。

* + 1. rocksdb\_max\_open\_files：rocksdb 将所有文件描述符保存在表缓存中。如果文件描述符的数量超过 max\_open\_files，则某些文件将从表缓存中逐出，并关闭其文件描述符。这意味着每次读取都必须通过表缓存来查找所需的文件。建议将 max\_open\_files 设置为 -1 以始终保持所有文件打开，从而避免昂贵的表缓存调用。
    2. rocksdb\_max\_background\_jobs：此变量用于替换 rocksdb\_base\_background\_compactions、rocksdb\_max\_background\_compactions 和 Rocksdb\_max\_background\_flushes 变量。该变量指定后台作业的最大数量。它自动决定分配多少线程用于刷新/压缩。它的实现是为了减少用户（令人困惑的）选项的数量，常见的配置是 CPU 核心数 / 4。线程太少，会因限制而导致更多停顿。如果线程太多，处理用户查询的线程可能会受到影响。该参数的默认值是2。
    3. rocksdb\_max\_background\_flushes：该变量已被 rocksdb\_max\_background\_jobs 取代。
    4. rocksdb\_max\_background\_compactions：该变量已被 rocksdb\_max\_background\_jobs 取代。
    5. rocksdb\_max\_total\_wal\_size：指定 WAL（预写日志）文件的最大大小，用于控制 Write-Ahead Logging (WAL) 日志文件的总大小。WAL 日志用于保证数据的持久性和一致性。通常，你不会希望 WAL 日志文件过大，以免消耗大量磁盘空间。同时，你也需要确保 WAL 日志文件足够大，以容纳你的应用中的事务数据。较小的 WAL 文件会减少写入延迟，但可能增加了持久性风险。较大的 WAL 文件可以提高持久性，但可能会增加写入延迟。因此，你需要根据应用需求来选择合适的大小，默认该值为4G。
    6. rocksdb\_table\_cache\_numshardbits：是rocksdb 中用于控制表缓存的参数。它用于指定表缓存的分片数量，以便更好地管理内存资源。默认情况下，rocksdb\_table\_cache\_numshardbits 的值为 6。这意味着 rocksdb 将使用 64 个分片。你可以将其视为一个起点，然后根据需要进行调整。需要监控 rocksdb 的性能。观察读取操作的延迟和缓存命中率，以确保表缓存的设置能够满足应用的性能需求。你可能需要不断调整 rocksdb\_table\_cache\_numshardbits 以优化性能。
    7. rocksdb\_default\_cf\_options之write\_buffer\_size：设置单个内存表的大小。一旦 memtable 超过此大小，它就会被标记为不可变并创建一个新的memtable。不同的工作负载需要不同大小的写缓冲区，如果你的应用执行大量写入操作，较大的写缓冲区可能有助于减少写放大效应，提高性能。然而，如果你的工作负载主要是读取操作，可能可以使用较小的写缓冲区。该参数的默认值为128M。
    8. rocksdb\_default\_cf\_options之target\_file\_size\_base：和target\_file\_size\_multiplier - 级别 1 中的文件将具有 target\_file\_size\_base 字节。下一个级别的文件大小将比前一个级别大 target\_file\_size\_multiplier。但是，默认情况下 target\_file\_size\_multiplier 为 1，因此所有 L1..Lmax 级别中的文件都是相等的。增加 target\_file\_size\_base 将减少数据库文件的总数，这通常是一件好事。我们建议将 target\_file\_size\_base 设置为max\_bytes\_for\_level\_base / 10，以便级别 1 中有 10 个文件。该参数的默认值为64M。
    9. rocksdb\_default\_cf\_options之cache\_index\_and\_filter\_blocks：参数用于控制是否缓存 SST 文件的索引块和过滤器块，这些块可以显著影响读取性能。如果你的应用经常执行范围查询或需要快速随机读取，缓存索引块和过滤器块通常是有利的，因为它们可以提高读取性能。但如果你的应用主要是写入操作，或者内存有限，你可能需要减少或禁用这个缓存以腾出更多内存。该参数的默认值为1。
    10. rocksdb\_default\_cf\_options之filter\_policy=bloomfilter：参数用于指定默认列簇（CF）的布隆过滤器（Bloom Filter）选项。Bloom Filter 是一种用于快速判断某个元素是否存在于集合中的数据结构，通常用于加速读取操作，减少不必要的磁盘读取。通常有几个相关的参数需要设置，例如 Bloom Filter 的误判率（false positive rate）、布隆过滤器的位数等。这些参数需要根据应用的需求进行调整。误判率较低的 Bloom Filter 会更准确，但会占用更多内存。你可以通过设置适当的位数来权衡准确性和内存占用。
    11. rocksdb\_lock\_wait\_timeout：是用于控制锁等待超时的参数，它指定了 rocksdb 在等待锁释放时的最长时间（秒）。这个参数可以帮助你在锁冲突的情况下避免无限期等待，从而减少潜在的死锁风险。如果你的应用有频繁的写入操作或锁冲突可能性较高，可能需要设置较短的等待超时时间，以减少锁等待时间。如果你的应用主要是读取操作，可以设置较长的等待超时时间。该参数的默认值为20秒。

1. 元数据集群参数调整：

Klustron元数据集群，测试中也建议对少量关键参数进行适当调整，按如下示例操作：

登陆集群一个节点：

export LD\_LIBRARY\_PATH=/kunlun/instance\_binaries/storage/57001/kunlun-storage-1.2.1/lib:$LD\_LIBRARY\_PATH

/kunlun/instance\_binaries/storage/57001/kunlun-storage-1.2.1/bin/mysql --connect-expired-password -S/nvme2/kunlun/storage\_logdir/57001/mysql.sock -uroot -proot

set global innodb\_buffer\_pool\_size=1\*1024\*1024\*1024; (缺省128M)

该参数需要在3个节点都进行设置。同样，以上设置只是在内存中临时变更了该参数值，如果需要永久设置，需要在my.cnf中进行设置并保存，重启各节点生效。

【END】