# maya（3期）详细设计

王达心

目录

[maya（3期）详细设计 1](#_Toc354061979)

[1. 实现功能 5](#_Toc354061980)

[2. 性能指标 5](#_Toc354061981)

[3. 名词解释 5](#_Toc354061982)

[4. 总体思路及折中 5](#_Toc354061983)

[4.1. 读写分离 5](#_Toc354061984)

[4.2. 负载均衡 6](#_Toc354061985)

[4.3. 虚拟表 6](#_Toc354061986)

[4.4. 多应用支持 6](#_Toc354061987)

[4.5. 事务 6](#_Toc354061988)

[4.6. 连接管理 6](#_Toc354061989)

[4.7. buffer pool利用率优化 7](#_Toc354061990)

[4.8. 监控与从库自动切换 7](#_Toc354061991)

[4.9. node多进程 7](#_Toc354061992)

[4.10. 基于zookeeper的配置变更 7](#_Toc354061993)

[4.10.1. zookeeper目录结构 7](#_Toc354061994)

[4.10.2. 动态配置变更 8](#_Toc354061995)

[4.10.3. 主库故障切换 8](#_Toc354061996)

[4.11. 配置文件无缝重载 9](#_Toc354061997)

[4.12. 日志 9](#_Toc354061998)

[4.13. 中间层不支持的mysql功能 9](#_Toc354061999)

[5. 模块图 10](#_Toc354062000)

[6. 模块 10](#_Toc354062001)

[6.1. SqlParser 10](#_Toc354062002)

[6.1.1. 功能 10](#_Toc354062003)

[6.1.2. 关键点说明 11](#_Toc354062004)

[6.2. GreetingPack 11](#_Toc354062005)

[6.2.1. 功能 11](#_Toc354062006)

[6.2.2. 关键点说明 11](#_Toc354062007)

[6.3. MonitorDBInstance 12](#_Toc354062014)

[6.3.1. 功能 12](#_Toc354062015)

[6.3.2. 关键点说明 12](#_Toc354062016)

[6.4. ServerNetwork 12](#_Toc354062017)

[6.4.1. 功能 12](#_Toc354062018)

[6.4.2. 关键点说明 12](#_Toc354062019)

[6.5. ClientNetwork 12](#_Toc354062020)

[6.5.1. 功能 12](#_Toc354062021)

[6.5.2. 关键点说明 12](#_Toc354062022)

[6.6. DBInstance 13](#_Toc354062023)

[6.6.1. 功能 13](#_Toc354062024)

[6.6.2. 关键点说明 13](#_Toc354062025)

[6.7. DBGroup 13](#_Toc354062026)

[6.7.1. 功能 13](#_Toc354062027)

[6.7.2. 关键点说明 13](#_Toc354062028)

[6.8. Router 13](#_Toc354062029)

[6.8.1. 功能 13](#_Toc354062030)

[6.8.2. 关键点说明 13](#_Toc354062031)

[6.9. VirtualTable 14](#_Toc354062032)

[6.9.1. 功能 14](#_Toc354062033)

[6.9.2. 关键点说明 14](#_Toc354062034)

[6.10. ZKManager 14](#_Toc354062035)

[6.10.1. 功能 14](#_Toc354062036)

[6.10.2. 关键点说明 14](#_Toc354062037)

[7. 配置文件 14](#_Toc354062038)

[7.1. 配置文件maya.json 14](#_Toc354062039)

[7.2. 分库分表配置方法 14](#_Toc354062040)

[7.2.1. 按表名序号分实例，库名相同 14](#_Toc354062041)

[7.2.2. 按库名序号分实例，每个库中表序列固定 15](#_Toc354062042)

[7.2.3. 按表名序号分实例，库名不同 15](#_Toc354062043)

[8. 异常处理 16](#_Toc354062044)

**修订记录**

|  |  |  |
| --- | --- | --- |
| **日期** | **摘要** | **备注** |
| **2013/4/18** | **连接session保持** |  |
|  |  |  |

# 实现功能

数据库服务的读写分离、负载均衡、故障自动切换等问题。

支持集群化的数据库/表分片路由，包括表分片部署在多个数据库实例上。maya 平台化管理，包括基于zookeeper的主库故障决策和切换、支持多应用共用中间层等。

# 性能指标

单机qps不低于30000次/秒，在性能未达到极限时对单次mysql交互延迟不超过50ms（与直接访问mysql相比）。

# 名词解释

* 管理节点（api）  
  数据库集群管理节点。
* Proxy node  
  数据库中间层节点，用于sql请求代理。
* Data node  
  用于数据存储的节点，即数据库节点。分为主库和从库两种类型。
* 虚拟表  
  应用可见的数据表，即虚拟的传统意义上一个数据库表。一个虚拟表的数据可分布存储于多个Data Node。
* DB group  
  数据库分组，由若干mysql实例组成，结构为1主n从。
* APP  
  应用，由若干个DB group组成，为一组虚拟表提供数据访问。
* 数据库集群  
  一个数据库集群由若干个Proxy node、若干个APP构成。
* 分片表  
  将虚拟表按一定规则划分成多个分片，每个分片表只属于一个DB Group。
* 数据rowkey  
  用于确定一条记录所属的分片表，每个rowkey只属于一个分片表。
* 分区方式  
  记录如下信息：
  + 虚数据表的分片信息
  + 分片表对应的rowkey范围
  + 每个分片表所属的DB Group

# 总体思路及折中

# 读写分离

一般情况下，数据库写请求分配到主库，读请求分配到从库。

如果遇到sql无法解析的情况，则分配到主库执行。

如果所有从库都不可连接或无从库，则所有请求都分配到主库执行。

某些情况下，应用对从库同步延迟比较敏感，要求写完数据之后立即从主库读数据以保证最新，为了简化客户端的工作，中间层配置一个参数，指定在客户端写之后n秒内的所有请求都放在主库执行。

# 负载均衡

这里的负载均衡指将客户端请求按指定比例分配到多个从库上。由于从库的服务器性能不一定相同，所以需要指定各从库的权重。默认权重为10。

# 虚拟表

虚拟表按序号分为n个分表，再按range分布在m个DB Group上。本期支持按单个rowkey hash分布数据，hash算法支持自定义。

hash算法配置示例：

"hash\_func": "parseInt(hash('crc32', value), 16);"

这种配置利用了nodejs的eval特性，将用户配置的字符串作为代码运行，其中“hash”是maya内置的hash算法库，支持常见的各种hash算法，包括'crc32', 'md5'，'sha1'等; value表示rowkey值, 仅作为内部传参使用，用户无需修改；16表示将hash返回的字符串以16进制解析为整数，用户无需修改。

# 多应用支持

为了减少maya的部署量，降低运维成本，一个maya需要为多个app提供数据库中间层服务。

每个APP会有唯一的app\_name标识，与客户端访问中间层时使用的dbname一一对应。中间层服务使用mysql连接协议中的dbname区分不同客户端所属APP.

# 事务

支持客户端以”start”或”begin”方式使用事务，不支持”set autocommit=0”。原因是”set autocommit=0”会影响服务端连接的属性，而我们的设计中服务端连接是可复用的，两者有冲突，并且线上业务一般不会使用”set autocommit=0”，所以不支持。

中间层在执行事务中的请求时，只根据事务中的第一个请求选择服务端数据库，之后的所有请求均在已分配的服务端连接上执行。事务内请求一定在主库上执行，即使第一个请求是查询，也不会在从库执行，此特性也可用于强制主库查询。

不支持跨mysql实例的事务，中间层不做这方面的检查，由应用保证。

# 连接管理

客户端与maya之间的连接有空闲超时控制，如果一定时间内不活动，将被强行关闭。

maya与mysql服务端之间保持长连接，使用连接池方式共享连接。连接池的容量可设置。

客户端通过密码验证后, 中间层服务会根据query路由规则为其分配服务端连接, 比如读连接和写连接, 所属分片等, 在客户端连接断开之前, 这些连接只服务于该客户端. 当客户端连接关闭之后, 相关的服务端连接会被释放回连接池.

除了query请求之外, 还有一类mysql请求是设置连接属性的, 比如字符集/变量等, 这些请求不仅要在所有已关联的服务端连接上执行, 还要记录当前值, 在未来新分配的连接上执行.

因为服务端连接复用的关系, 当释放连接时, 需要将所有session级系统变量改回系统默认值(DEFAULT).

# buffer pool利用率优化

通常来说，来自单个客户端机器的请求重复的可能性较大，如果可以根据客户端ip分配从库读请求，也就可以较好地利用mysql自身的buffer pool，但也有一定的风险，如果各个客户端的请求数量很不均匀，就不宜按这种策略分配，所以作为参数放在配置中。

# 监控与从库自动切换

每个中间层节点对后端数据库服务都有监控，其中对所有数据库都有可连接状态和连接数监控("Threads\_connected)，对从库还有"Slave\_IO\_Running"/ "Slave\_SQL\_Running"/ "Seconds\_Behind\_Master"的监控。如果从库异常，则不再将新的客户端请求分配到对应的从库上，但已分配的请求不会重新分配。

# node多进程

node目前(v0.8)的多线程支持非常有限，需要使用多进程来充分利用多核。这也带来一些缺点，例如每个工作进程都要独立执行监控数据库的工作，不过这些缺点对整体服务影响不大，可以接受，并且node也在考虑支持多线程，后续版本可保持关注。

node cluster模块提供多进程支持框架，以此为基础，可以实现进程异常崩溃自动重启。

对于未捕获的异常，node默认的行为是退出进程。为了尽量保证服务的连续性，避免局部异常导致整个进程上正在处理的连接均断开，我们对未捕获的异常仅做日志记录，不退出进程。

# 基于zookeeper的配置变更

# zookeeper目录结构

动态配置是指需要经常变动的配置，包括数据库节点信息、主从配置、从库监控禁用开关、从库查询权重等。

我们使用zookeeper作为动态配置信息的数据存储和同步服务。

zk结构如下：

|-- sina\_maya

| |-- maya\_cluster\_name1

| | |-- apps

| | | |-- app\_name1

| | | | |-- db\_group\_1

| | | | | |-- db11\_ip:port

| | | | | | `-- ip:process\_id(maya1)

| | | | | | `-- ip:process\_id(maya2)

| | | | | |-- db12\_ip:port

| | | | |-- db\_group\_2

| | | | | |-- db21\_ip:port

| | | | | | `-- ip:process\_id(maya1)

| | | | | | `-- ip:process\_id(maya2)

| | | | | |-- db22\_ip:port

| | | |-- app\_name2

| | |-- maya\_nodes

| | | `-- ip:process\_id(maya1)

| | | `-- ip:process\_id(maya2)

| |-- maya\_cluster\_name2

其中“`”开头的为EPHEMERAL节点，由maya中间层维护；其他为永久节点，由管理平台维护。由于maya可能与其他服务共用zookeeper，所以根目录不一定是”/”，上图只是相对路径，maya配置文件中可以指定绝对根路径（如/sina\_maya/dpool/dba）。

# 动态配置变更

数据库节点（如db11\_ip:port）的data信息如下：

{ "is\_master": 1 , "weight": 10, "disable\_monitor": 0, "db\_status": 1 }

"is\_master": 1表示主库, 0表示从库；"weight": 整数表示的读权重； "disable\_monitor": 0表示不禁用监控，1表示禁用监控；"db\_status": 1表示启用，0表示未用；

maya中间层监听每个数据库节点的数据，如果发生变更，则实时载入生效。

# 主库故障切换

各maya启动后将自己以瞬时节点（EPHEMERAL）方式注册到maya\_nodes，可用于外部获取maya存活节点列表。注册节点名称为“ip:process\_id”。这里未使用ip+port作为节点名称的原因是maya以多进程方式运行，每个进程是独立的，但服务端口是共享的，如果以port注册会造成冲突。其中ip对一个机器来说可能有多个，线上的惯例是取eth1的第一个ip（有可能存在多个ip），如果eth1不存在则取eth0，可能还有其他需要兼容的情况（如虚拟机无ethx），我们使用node os库的os.networkInterfaces()来获取全部ip信息，然后取其中最后一项的第一个address作为ip。

中间层自身监控数据库服务状态。如果正常则向zookeeper的相应数据库节点注册。如果异常则删除数据库节点下自己创建的节点，并调用管理api通知管理节点处理。中间层管理节点收到中间层的请求后，读取当前各db节点的子节点（认为该db正常的maya服务）数量，并与maya\_nodes的子节点（当前运行中的maya服务）数量做比较，如果认为主库异常的maya服务达到一定比例，则执行主库切换操作，并将操作结果返回。

# 配置文件无缝重载

由于服务重启会短暂的中断服务，对业务有影响，而配置变更可能是常态，所以需要做到配置文件重载不中断服务。除了工作进程数量和进程服务端口不能重载之外，其他参数均可重载。

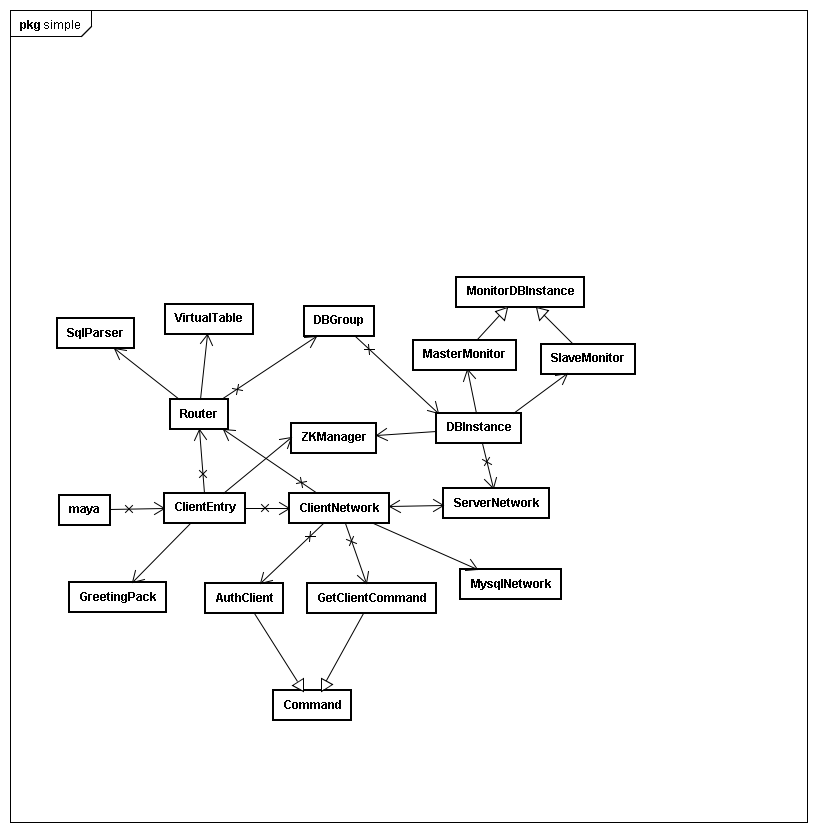
# 日志

maya日志级别从低到高依次为："TRACE", "DEBUG", "INFO", "WARN", "ERROR", "FATAL"。其中debug级别会记录客户端的所有sql语句，info级别会周期性记录累计请求数，warn及以上级别会记录各种错误信息。所有日志均写入maya.log，warn及以上级别还会额外写入maya.wf.log，便于追踪问题。

# 中间层不支持的mysql功能

* 不支持全部取出虚拟表数据。
* 不支持对虚拟表中的数据做跨分片访问，如查询rowkey在某一范围内的数据。对于in操作，仅在第一个value所属分片表中执行查询，如果其他value与第一个value不在同一个分片中，则查询不到对应数据。
* 不支持跨分片的事务。
* mysql\_thread\_id()函数得到的是虚假的mysql thread，不能对应到数据库的thread id。
* show系列命令只能获取到主库的相关信息，不会得到从库信息，需要直连数据库才能查看。
* 虚拟表的insert语句必须带列名,如"insert into t (a, b) values (1, 2)"不能简写为"insert into t values (1, 2)"。

# 模块图



# 模块

# SqlParser

# 功能

对外提供如下接口：

* 解析sql语句
* 获取where子句中的virtual table
* 区分读操作
* 获取rowkey值
* 重组sql，将虚拟表名替换为数据库分片表名
* 识别事务操作

# 关键点说明

考虑到sql解析的CPU耗费较高，所以采用c语言开发node扩展的方式开发本模块。

从sql语句中获取virtual table时如果有多个table，则取第一个table。

分片表的sql语句必须带rowkey，并且只支持”rowkey = ‘value’”和”rowkey in (‘value1’, ‘value2’)”操作，其中in操作在中间层取第一个value作为分片依据，由外部应用保证所有值在同一个分区。不支持其他比较操作符。

# GreetingPack

# 功能

为客户端鉴权提供mysql协议规定的GreetingPack。

# 关键点说明

如果直接用mysql服务端来鉴权，有如下缺点：

* 鉴权服务器不好选取。如果是固定用一个服务器做鉴权，可能会有性能问题；如果是随机选择，则要求所有后端mysql服务器都用相同的用户名和密码，不安全；
* 对短连接的鉴权每次都要经过mysql服务端，性能和响应时间上也有损失；

所以决定proxy server自身充当鉴权服务器，模拟mysql协议对客户端连接鉴权。mysql的连接及鉴权简要过程如下：

1、服务器发送greeting packet给客户端，其中包括随机字符串（scramble）.

2、客户端作如下计算:

stage1\_hash = SHA1(明文密码).

token = SHA1(scramble + SHA1(stage1\_hash)) XOR stage1\_hash

3、客户端将token发送给服务端

4、服务端作如下计算：

stage1\_hash = token XOR SHA1(scramble + mysql.user.Password)

5、服务端比对SHA1(stage1\_hash)和mysql.user.Password，如果匹配，则认证正确。

由于对于客户端来说proxy充当了mysql服务，所以需要proxy直接向client发送greeting packet，而greeting packet中有很多mysql服务的属性，不是完全固定的，不宜写死，所以在服务初始化时随机选取一个slave，只需要新建一个连接即可收到这个数据包。

其中服务端对比中mysql.user.Password是加密后的字符串，我们为了简化设计，proxy server上以未加密的字符串记录密码，所以验证方法也需要相应修改，服务端采用与客户端相同的token生成方法，两边得到的结果一致即表明通过验证。

# MonitorDBInstance

# 功能

监控mysql服务是否正常。另外有两个子类MasterMonitor和SlaveMonitor，分别负责监控主库和从库，主库只监控连接数是否正常，从库还监控主从同步是否正常运行，从库延迟是否超过阈值。

# 关键点说明

这部分功能相对独立，并且可以直接使用node的mysql客户端模块实现，不需要深入到mysql协议层处理。

主从同步是否正常判断依据直接取自从库的“show slave status”结果，包括如下几个属性：

Seconds\_Behind\_Master：主从落后，不超过设定阈值。

Slave\_IO\_Running：值为“Yes”，表示正常工作。

Slave\_SQL\_Running。：值为“Yes”，表示正常工作。

数据库连接数取自数据库的“show status”结果中Threads\_connected属性。

对外通过事件方式传递监控结果。

# ServerNetwork

# 功能

建立及关闭与mysql server的连接，发送客户端请求，将服务端回应转发给客户端。

# 关键点说明

对外提供执行客户端请求的接口execClientReq()，如果当前未建立连接，则新建连接并向mysql server发起鉴权，鉴权通过后再向server端发送请求。如果已建立连接，则直接向server端发送请求。并将server端的返回数据直接发送到客户端连接。

为了支持连接复用，对外提供release()接口，解除与客户端连接之间的关联，并产生’release’事件，供外部回收本对象。

如果执行客户端请求期间服务端连接被关闭，则直接关闭相应的客户端连接。

当与mysql server之间的连接断开时，将产生dead事件，表示本实例已经无用，可以销毁。

# ClientNetwork

# 功能

管理来自客户端的连接，包括鉴权、交互处理等。

# 关键点说明

通过鉴权后，进入数据交互过程，收到client packet之后，交给Router处理。

每个ClientNetwork对象进入数据交互阶段后，会被分配一个当前对应的ServerNetwork对象。

# DBInstance

# 功能

数据库实例，ServerNetwork连接池管理。

# 关键点说明

ServerNetwork池中的对象是在需要时新建的，当执行client端请求时，如果ServerNetwork池中没有空闲实例，则新建ServerNetwork实例来执行。

对单个mysql实例的连接数设置限额，超过之后不再新建连接，并将新请求所属的客户端连接断开。

监听ServerNetwork实例的’release’和’dead’事件，分别用于复用回收和连接池容量回收。

# DBGroup

# 功能

主从数据库管理，一个DBGroup包括1个主库和若干个从库。根据sql请求类型（读写）分配到主库/从库DBInstance上执行。处理从库禁用、恢复请求。

# 关键点说明

从库支持按权重分配读请求，权重可以是任意16位整数，跟连接数无直接关系，仅表示从库被选择的几率。

禁用从库的最大允许比例可配置，以防禁用过多导致雪崩效应逐个压死从库。一个db group中可禁用的从库数量为从库总数\*可禁用比例后取整，如果配置值>=100%或为非法值，则记录日志，并且不禁用任何从库。

# Router

# 功能

数据库路由功能入口，管理所有的DBGroup，执行sql解析，并选择适当的DBGroup执行client端请求。

# 关键点说明

本类为单实例类，进程全局共享。

如果sql解析失败，或未找到相应的分片所属db group，则将该sql请求交给default db group（配置文件中第一个db group）处理。

判别事务类请求，并将其分配到主库执行。

# VirtualTable

# 功能

虚拟表，对外提供获取虚拟表的rowkey名、获取rowkey值所属db group的方法。

# 关键点说明

虚拟表的配置在配置文件中，详见配置文件说明。

# ZKManager

# 功能

管理zookeeper连接/自动重连，向zookeeper注册。

# 关键点说明

zookeeper的连接地址参数支持ip和域名（不支持二者混合），如果是域名，则解析出所有ip供zookeeper客户端连接。

连接/重连成功后对外抛出'connected'事件，通知其他模块执行zookeeper相关操作。

由于zookeeper客户端的特性，连接失败后会在内部自动重试，外部不会知晓，所以简单设定了一个超时，如果1秒未连接成功，则对外抛出'init\_failed'事件，通知其他模块使用默认配置（配置文件参数）运行。

如果连接失败或中途断开，则自动重连。由于使用的zookeeper客户端有限制，连接断开后必须新建zookeeper对象，否则可能导致进程崩溃

# 配置文件

为了便于程序操作配置文件，格式均为json文本，可以直接转换为js对象使用。日后可由RootServer向proxy node下发配置修改，直接生成配置文件。

# 配置文件maya.json

详见wiki

http://wiki.intra.sina.com.cn/pages/viewpage.action?pageId=7160780

# 分库分表配置方法

# 按表名序号分实例，库名相同

库名相同，表名在各库中无重复，按表名分区间分布到不同数据库

dbname.my\_table[0..31] -> group\_0

dbname.my\_table[32..63] -> group\_1

{

virtual\_table. assign\_db\_instance\_by: "table",

virtual\_table.partition\_table\_name\_pattern: "my\_table\_%.02x",

virtual\_table. partitions: [

{

"db\_group": "group\_0",

“assign\_range”: [0..31]

}

…

]

}

# 按库名序号分实例，每个库中表序列固定

按库名分区间分布到不同的数据库，每个库中表名序列均为0~n

dbname[0..3].my\_table[0..31] -> group\_0

dbname[4..7].my\_table[0..31] -> group\_1

{

virtual\_table. assign\_db\_instance\_by: "db",

virtual\_table. partition\_db\_name\_pattern: " db\_%.02d ",

virtual\_table.partition\_table\_name\_pattern: " my\_table\_%.02x",

virtual\_table. partitions: [

{

"db\_group": "group\_0",

“assign\_range”: [0..3]

}

…

]

}

# 按表名序号分实例，库名不同

库名不同，表名在各库中无重复，仅按表名分区间分布到不同数据库

dbname0.my\_table[0..31] -> group\_0

dbname1.my\_table[32..63] -> group\_1

{

db\_groups: [

{

"name": "group\_0",

"mysql\_db\_name ": " dbname0",

},

{

"name": "group\_1",

"mysql\_db\_name ": " dbname1",

}

]

virtual\_table. assign\_db\_instance\_by: "table"

virtual\_table.partition\_table\_name\_pattern: " my\_table\_%.02x",

virtual\_table. partitions: [

{

"db\_group": "group\_0",

“assign\_range”: [0..31]

},

{

"db\_group": "group\_1",

“assign\_range”: [32..63]

}

]

}

# 异常处理

当发生代码级未捕获exception时，会使用node默认的uncaughtException处理，记录fatal日志，正在运行的服务和连接都不受影响。

如果工作进程异常退出，会自动重启。

如果在执行客户端请求期间捕获异常，将会记录warn日志并返回错误。

如果启动时发现配置文件错误，会记录fatal日志，并不断尝试重启服务。目前的配置文件错误检查包括：

* 非法json格式
* db group重名
* 数据库host为非法ip
* db group无主库
* db group有多于1个主库
* client\_port与management\_port相同
* server\_conn\_wait\_timeout不小于5秒
* max\_disabled\_slaves\_percent < 0 或者 >= 100
* partition\_table\_name\_pattern非法（如%后跟随错误的类型）

如果重载配置文件时发现配置错误，则记录异常日志，并继续使用原配置提供服务。