# Master模块和Server容错容灾模块详细设计

3200105851 李毅桐

## 1. 模块设计

### 1.1 Master Server 模块设计

本模块主要实现管理Master 服务机对各个Region的table进行管理，实现负载均衡；监听服务器上下线，实现容错容灾的功能。建立临时数据库，辅助复杂查询。

* 管理和维护tableRegionMap。tableRegionMap储存着每张表存储的所在的region。当发生表的创建和删除操作时，更新tableRegionMap同时写入zookeeper节点，实现数据永久化。
* 实现负载均衡。当进行表的创建操作时，找到含有最少表的数量的region，让该region下的主服务器和副服务器创建该表。
* 实现容错容灾。当某一region下的主服务器或者副服务器掉线时，通过zookeeper获知该消息，并且唤醒一台闲置的server。
* 处理Client发送的create和drop请求。
* 辅助实现对不同region下多个表的复杂查询。

### 1.2 Server容错容灾

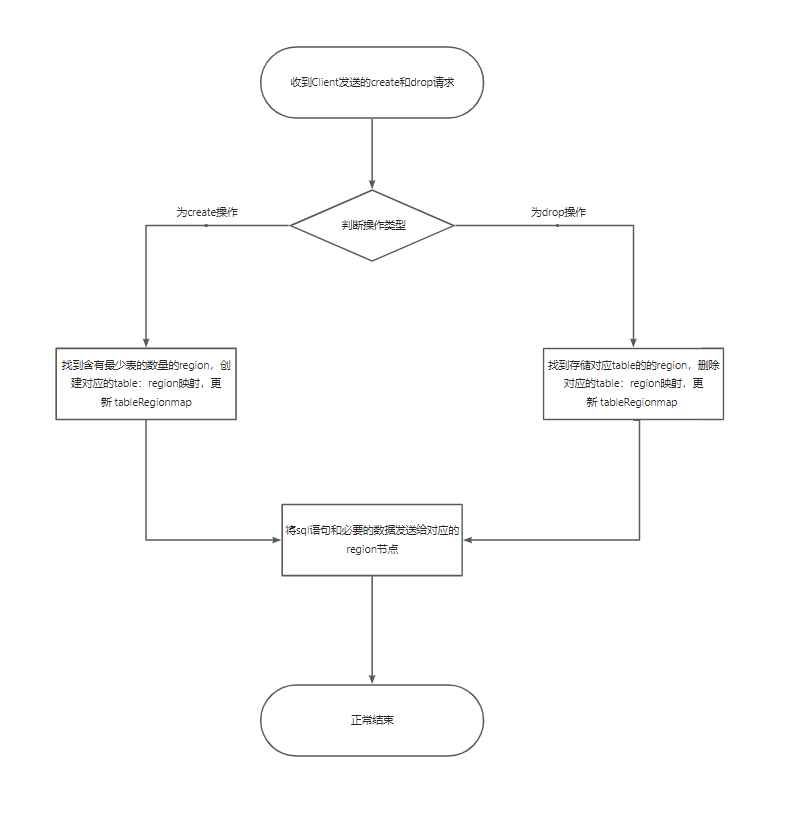
该模块主要是配合master实现容错容灾功能。当Master唤醒一个新的服务器时，它会复制同一个region下的另外一台服务器的数据库。保证数据库类型一致。

* 监听同一个region下是否有两条服务器，若只剩下一台，则向master发送evokeServer请求，请求唤起一台闲置的服务器，并注册到region节点下。
* 当一台副服务器刚刚被唤醒时，需要与主服务器通信，复制其数据库，保证主服务器和副服务器的数据一致。

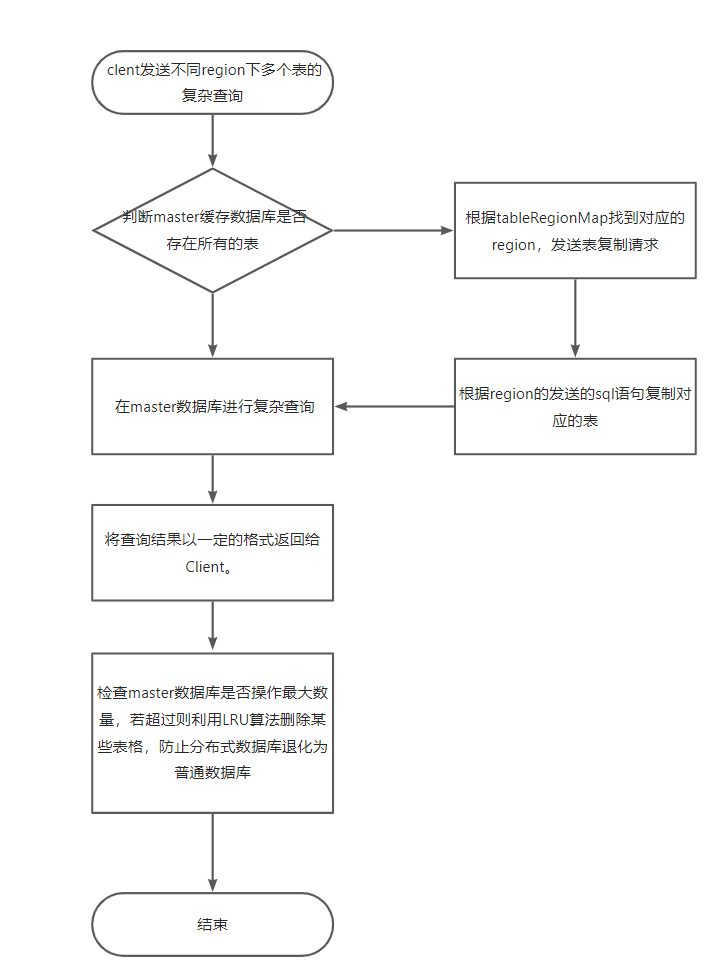
## 2流程图

### 2.1 Master模块的流程图

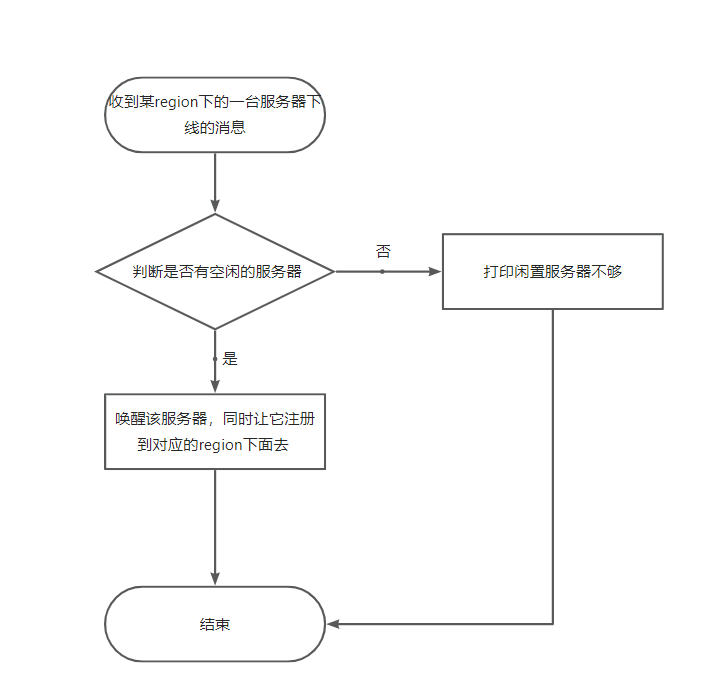
#### 2.1.1 处理Client发送的create和drop请求



#### 2.1.2 辅助实现对不同region下的复杂查询

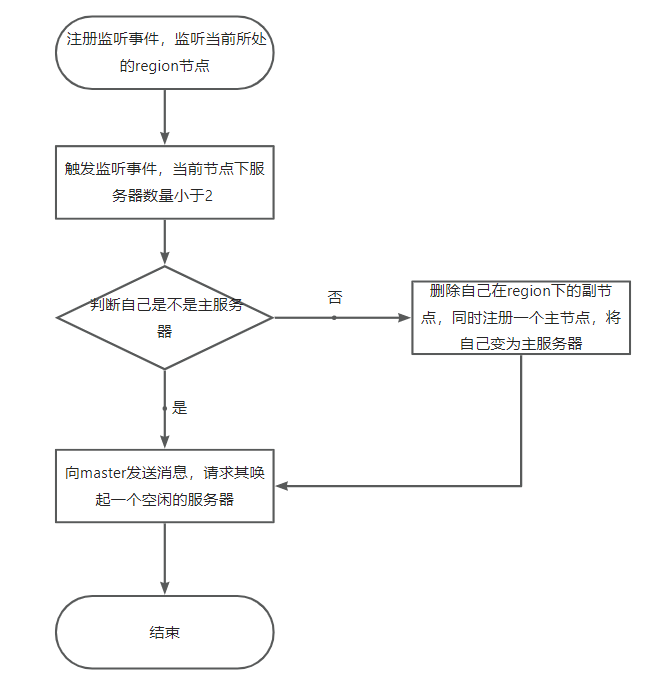


#### 2.1.3 实现容错容灾

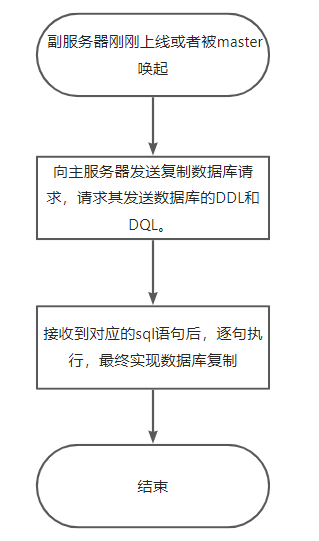


### 2.2 Server容错容灾模块流程图

#### 2.2.1 Server向master发送evokeServer请求



#### 2.2.2 数据库复制



## 3 重要函数

#### 3.1 Master模块

* retrieveMapFromZooKeeper();//// 从ZooKeeper的/master节点获取存储的字符串，解析字符串并将键值对存储到tableRegionMap中。
* storeMapToZooKeeper(); //当执行create或者drop时，需要更新tableRegionMap,并将值存储到/master节点
* createTable ();//根据接收到的region数据复制对应的数据库
* findMinNumRegion;()//找到存储表数量最少的region
* sendCreateData();//当需要复制数据库时，往对应的region发送消息。
* evokeServer (String node);监听对应的节点，当某region下服务器下线时触发，唤起一个空的闲置的服务器。

#### 3.2 Server容错容灾模块

* keepTwoServers()//保证同一个region节点下有两个服务器，一主一副。若没有则调用evokeServer();
* evokeServer();//将消息发给master，让它唤醒一个新的服务器。同时如果自己是副本机，那就将自己变为主机，同时修改zookeeper里面的节点。
* sendCopyRequest()//副服务器刚刚上线时调用，向主服务器发送复制数据库请求
* sendDbData()//主服务器向副服务器发送对应的DDL和DQL，以便于副服务器实现数据库复制

## 4 个人心得

本次项目，我作为组长。主要职责是负责Master模块和Server容错容灾模块。同时我协调各个同学之间的工作，并且统筹开发的进度等，所以整体压力还是较大的。不过，圆满完成任务之后，还是感觉收获较多的。

首先技术方面，我学习了zookeeper框架，Java的maven包管理工具，jdbc连接数据库以及使用Java API连接zookeeper。这些经历在之后的工作学习中都对自己大有裨益。

再者由于刚开始对实现分布式框架的认识不清楚，我们小组进行了多次讨论，在讨论过程中了解到了不同成员的不同想法，通过比较这些想法最终得出一个比较清晰的分布式数据库的实现框架。这个项目让我明白的沟通的重要性。