**BlotMQ-Registry**

**技术文档说明**

针对版本V1.0.0

**©成都基础平台架构**

**2017/11/21**

BoltMQ-Registry修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 修订内容 | 作者 | 审核 | 修订日期 |
| V1.0.0 | 初始版本 | 田玉粮 | 基础平台架构组 | 2017/11/21 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1 概述 4](#_Toc499226505)

[2 Registry模块交互 4](#_Toc499226506)

[2.1 Broker 4](#_Toc499226507)

[2.2 Console 4](#_Toc499226508)

[2.3 Net 4](#_Toc499226509)

[2.4 Registry 4](#_Toc499226510)

[3 专业术语 5](#_Toc499226511)

[4 Registry实现原理 5](#_Toc499226512)

[4.1 Broker注册业务 5](#_Toc499226513)

[4.1 扫描活跃Broker 6](#_Toc499226514)

[4.1 Registry与ZooKeeper 7](#_Toc499226515)

[4.2 Registry核心数据结构 8](#_Toc499226516)

[4.3 Registry内存数据变化 8](#_Toc499226517)

[4.4 普通Topic与顺序Topic 9](#_Toc499226518)

[4.1 Topic与Broker映射关系 10](#_Toc499226519)

[附件一 BoltMQ开发者联系方式 11](#_Toc499226520)

# 概述

Registry模块维护了很多broker和topic等信息，通过net和broker建立长连接，来保持与broker的通信，同时会提供心跳检测、数据更新与查询等常规服务。它保存活跃的broker列表，包括Master和Slave；同时也保存所有topic和该topic所有队列的列表。

# Registry模块交互

## Broker

每个Registry定时收到broker注册信息，并维护每个broker节点地址、角色、ID等信息，同时维护每个Broker活跃信息。

## Console

console模块通过client底层接口，间接与所有Registry建立连接并通信。

## Net

registry通过Net创建Service（目前端口为9876），注册并发布服务，供Broker、Client、Web等模块调用。

## Registry

Registry节点之间没有交互。大部分Registry维护的数据都存储于内存，顺序Topic数据存储与文件。

# 专业术语

◼Topic

Topic是一个消息主题，一个在线Producer实例只能对应一个Topic，一个在线Consumer实例可以对应多个Topic，一条消息必须属于一个Topic。

◼QueueData

Topic队列，包含broker名称、读队列数、写队列数、broker权限、topic是否同步标记值。

◼BrokerData

描述Broker详细信息的数据结构，包括broker名称、broker地址、brokerId等。

◼TopicQueueTable

描述topic、queue、broker三类数据结构的映射关系。

◼BrokerAddrTable

描述brokerName、brokerId、brokerAddr三类数据结构的映射关系。

◼ClusterAddrTable

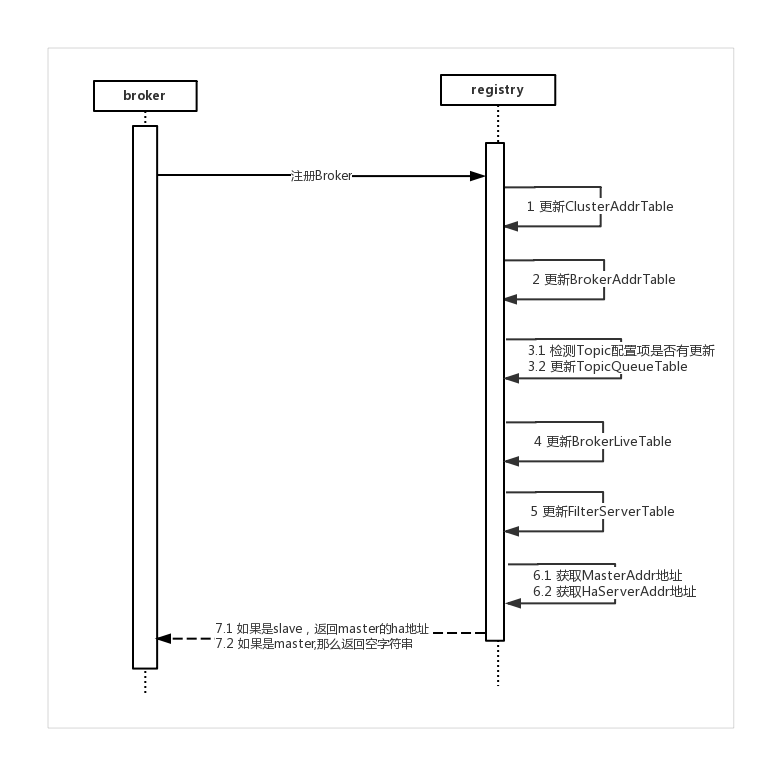
描述Broker与Cluster二者对应关系，可以通过brokerName查询详细的broker信息。

◼BrokerLiveTable

描述broker心跳信息的结构，每个broker发送心跳信息后，Registry将会维护broker心跳的最后更新时间。

# Registry实现原理

## Broker注册业务

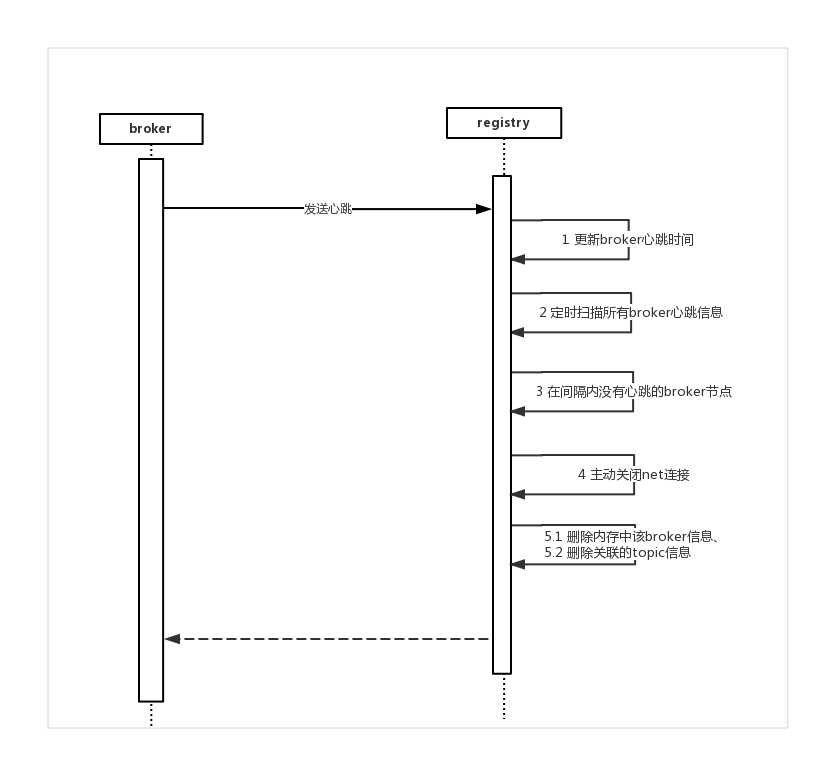


broker注册由broker发起请求，Registry收到请求后维护topic、broker、cluster等等映射关系。

Registry收到请求后，根据请求参数，先后更新ClusterAddrTable、BrokerAddrTable、TopicQueueTable、BrokerLiveTable、FilterServerTable等数据接口的数据。

注意：每来一个Master注册，创建一个QueueData对象；如果是新建topic，就是添加QueueData对象；如果是修改topic，就是把旧的QueueData删除，加入新的。

## 扫描活跃Broker



broker每隔30秒上报一次心跳信息，Registry收到心跳信息后，在内存中维护BrokerLiveTable结构体，该数据结构存储了每个broker最后心跳更新时间、broker地址、net连接等等信息。

Registry启动后就执行定时任务：每隔10秒执行一次， 扫描2分钟内没有心跳上报的broker。

如果扫描到结果，即存在2分钟内没有心跳上报的broker节点，那么Registry就主动关闭net连接，并删除内存中维护的数据，同时一并删除broker、topic、cluster、filter等等关联信息。

## Registry与ZooKeeper

(1)对于BoltMQ来说，topic的数据在每个Master上是对等的，没有哪个Master上有topic上的全部数据，所以ZooKeeper的选举leader功能并不适合BoltMQ。

(2)BoltMQ集群中，需要有构件来处理一些通用数据，比如broker列表，broker刷新时间，使用ZooKeeper客户端处理数据之间的一些逻辑关系却比较麻烦，并且ZooKeeper还得保证多个master之间的一致性，这点更增加代码复杂度。如果有多种角色，那么ZooKeeper代码就更复杂了。

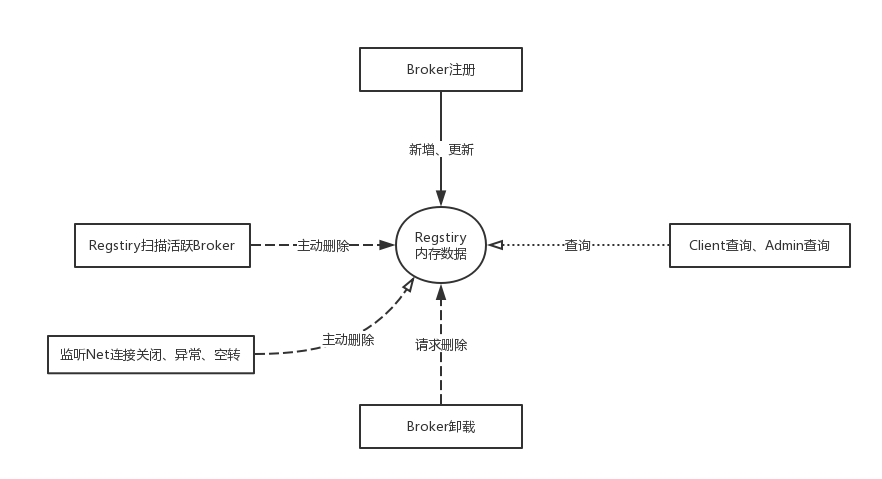
(3)既然BoltMQ集群中没有用到ZooKeeper的一些重量级的功能，只是使用ZooKeeper的数据一致性和发布订阅的话，与其依赖重量级的ZooKeeper，还不如写个轻量级的Registry。

(4)Registry也可以集群部署，Registry与Registry之间无任何信息同步，只有一千多行代码的Registry稳定性肯定高于ZooKeeper。

## Registry核心数据结构

|  |  |  |  |
| --- | --- | --- | --- |
| 数据结构 | 类型 | 数据格式 | 存储数据 |
| TopicQueueTable | HashMap | topic[list<QueueData>] | 保存topic-queue信息 |
| BrokerAddrTable | HashMap | brokerName[BrokerData] | 保存broker地址信息 |
| ClusterAddrTable | HashMap | clusterName[set<brokerName>] | 保存broker-cluster信息 |
| BrokerLiveTable | HashMap | brokerAddr[brokerLiveTable] | 保存broker心跳信息 |

## Registry内存数据变化



整个Registry模块维护的内存数据，包括TopicQueueTable、BrokerAddrTable、ClusterAddrTable、BrokerLiveTable、FilterServerTable等5个部分，这5个部分的数据结构都存在内存中。不同的请求到达Registry模块后，内存中的数据有不同的处理方式。

(1)Broker注册业务请求发出后，Registry模块的内存数据就会新增或更新。Registry将会在内存中维护topic、broker、queue、cluster、filter等相互的映射关系。

(2) Broker卸载业务请求发出后，Registry模块的内存数据就会清除对应的映射关系。如果BrokerAddrs节点没有数据，则会将内存中整个BrokerAddrs节点都删除。

(3) Registry定时任务，每隔10秒扫描一次，如果在2分钟内不活跃Broker(也就是2分钟内没有心跳信息的Broker)，也会在内存中删除对应的映射关系。

(4)Client和Admin模块，则通过接口查询Registry内存中的数据，例如查询Topic路由信息。

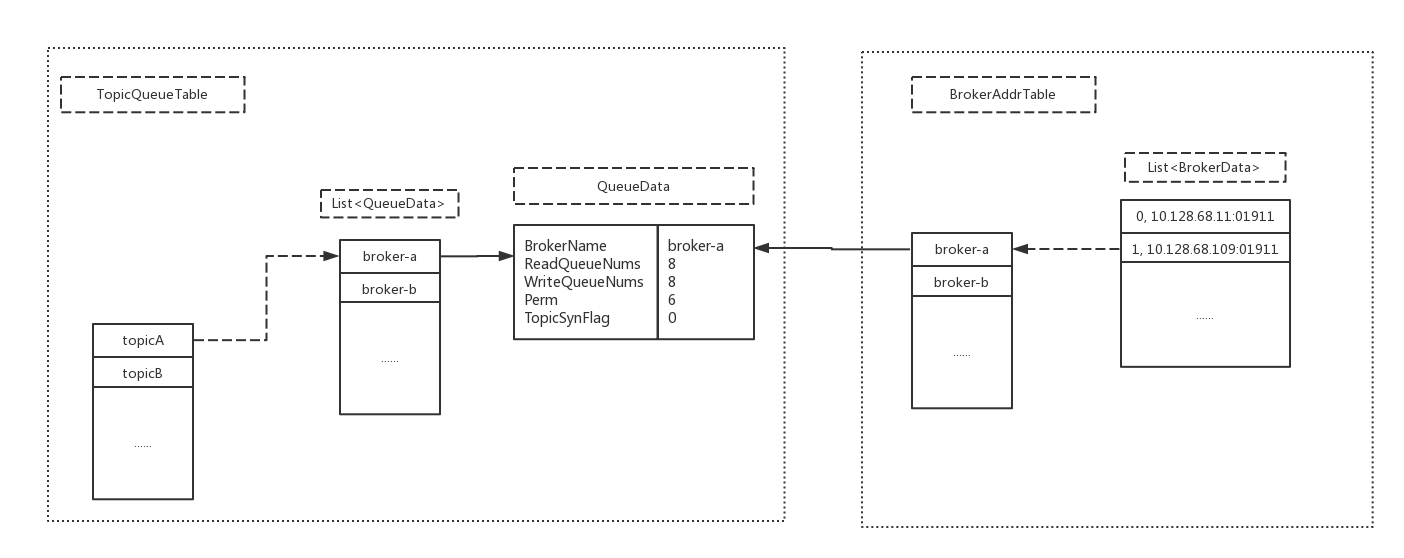
(5)如果网络通信模块Net的连接关闭、异常、空转，则Registry侧收到请求后，也会删除内存数据。

## 普通Topic与顺序Topic

|  |  |  |
| --- | --- | --- |
|  | **普通Topic** | **顺序Topic** |
| 创建方式 | Broker注册，注册成功则创建成功 | 调用PUT\_KV\_CONFIG接口，调用成功则创建成功 |
| 存储文件 | 存储于Broker模块文件,路径是  $SMARTGO\_HOME/store/config/topics.json | 存储与Registry模块文件，路径是  $HOME/namesrv/kvConfig.json |
| 查询方式 | Broker注册会加载topics.json内容，方式传递给最终将所有Topic数据加载到Registry内存中 | Registry启动，直接读取kvConfig.json并加载到内存中 |
| Broker与  Topic | Registry模块维护TopicQueueTable数据 | kvConfig.json配置文件，  {"jcpt-example-200":"broker-a:8"}  以brokerName:queueNum区分 |
| 数据状态 | 维护于Registry内存中 | 维护于Registry内存中 |

## Topic与Broker映射关系

Registry维护topic与broker对应的关系的数据结构是 TopicQueueTable，结构类型是map，内存中映射关系如下图所示：



(1)一个Topic对应多个QueueData， 而每个QueueData包括brokerName名称、queue队列个数；

(2)brokerName恰恰又是BrokerAddrTable结构的key值，根据brokerName即可把BrokerAddrTable结构体、TopicQueueTable结构体 关联起来。

(3)每次客户端查询topic路由信息，只需把TopicQueueTable结构体解析，然后把broker与topic在内存中的数据结构解析即可。

# 附件一 BoltMQ开发者联系方式

|  |  |  |
| --- | --- | --- |
| 姓名 | 联系方式 | 更新日期 |
| 郜焱磊 | gyl\_adaihao@163.com | 2017/11/21 |
| 田玉粮 | idistyl@gmail.com | 2017/11/21 |
| 尹同强 | tongqiangyin@gmail.com | 2017/11/21 |
| 罗继 | gunsluo@gmail.com | 2017/11/21 |
| 周飞 | he236555699@163.co | 2017/11/21 |
| 戎志宏 | hzrong007@163.com | 2017/11/21 |
| 苟建军 | xyclr2010@gmail.com | 2017/11/21 |