BlotMQ-Registry

技术文档说明

针对版本V1.0.0

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

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。

2 Registry 模块交互

2.1 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。

2.2 Console

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

2.3 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。

2.4 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。

3 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

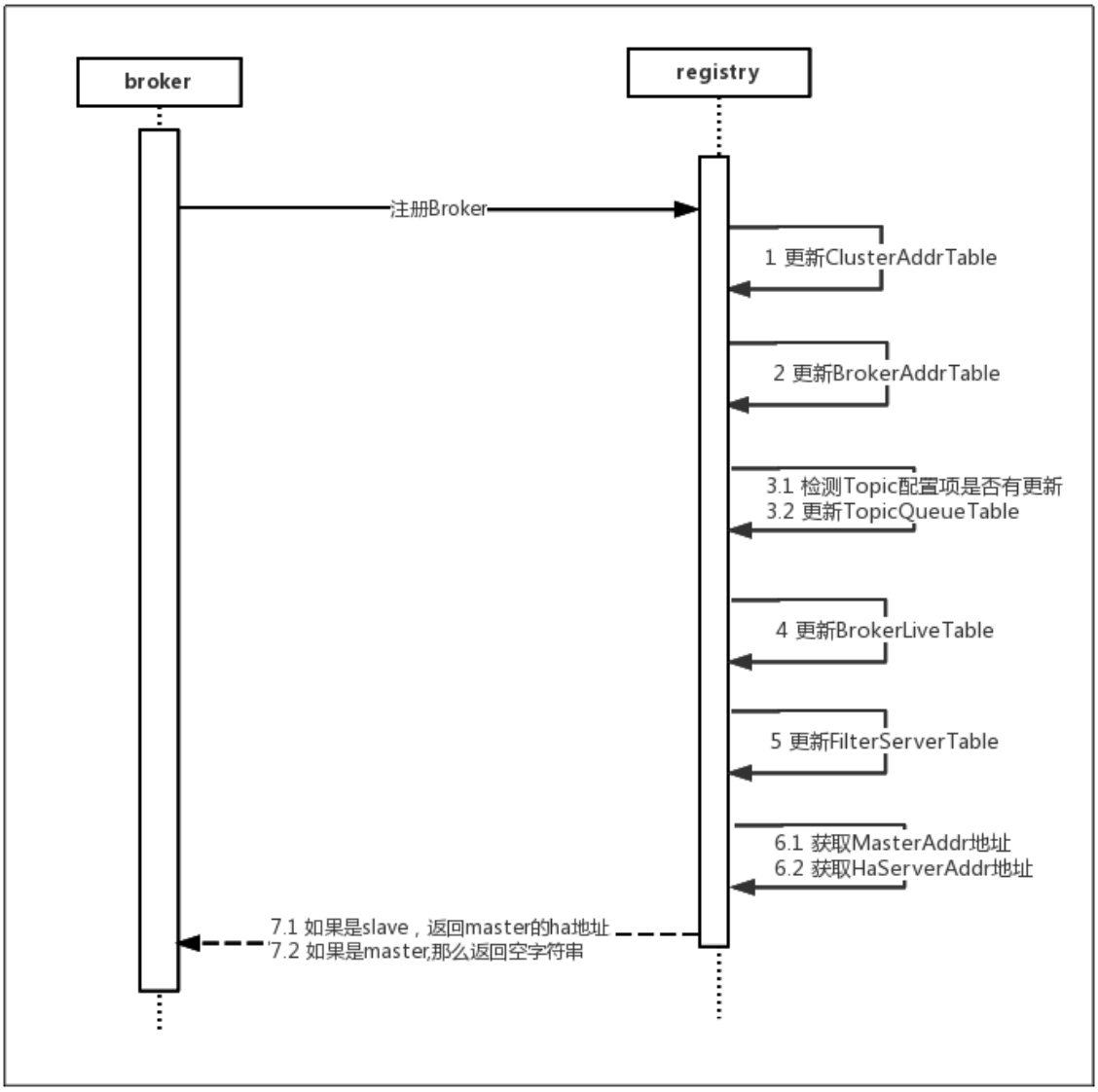
描述 Broker 与 Cluster 二者对应关系，可以通过 brokerName 查询详细的 broker 信息。

- 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。

4 Registry 实现原理

4.1 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。

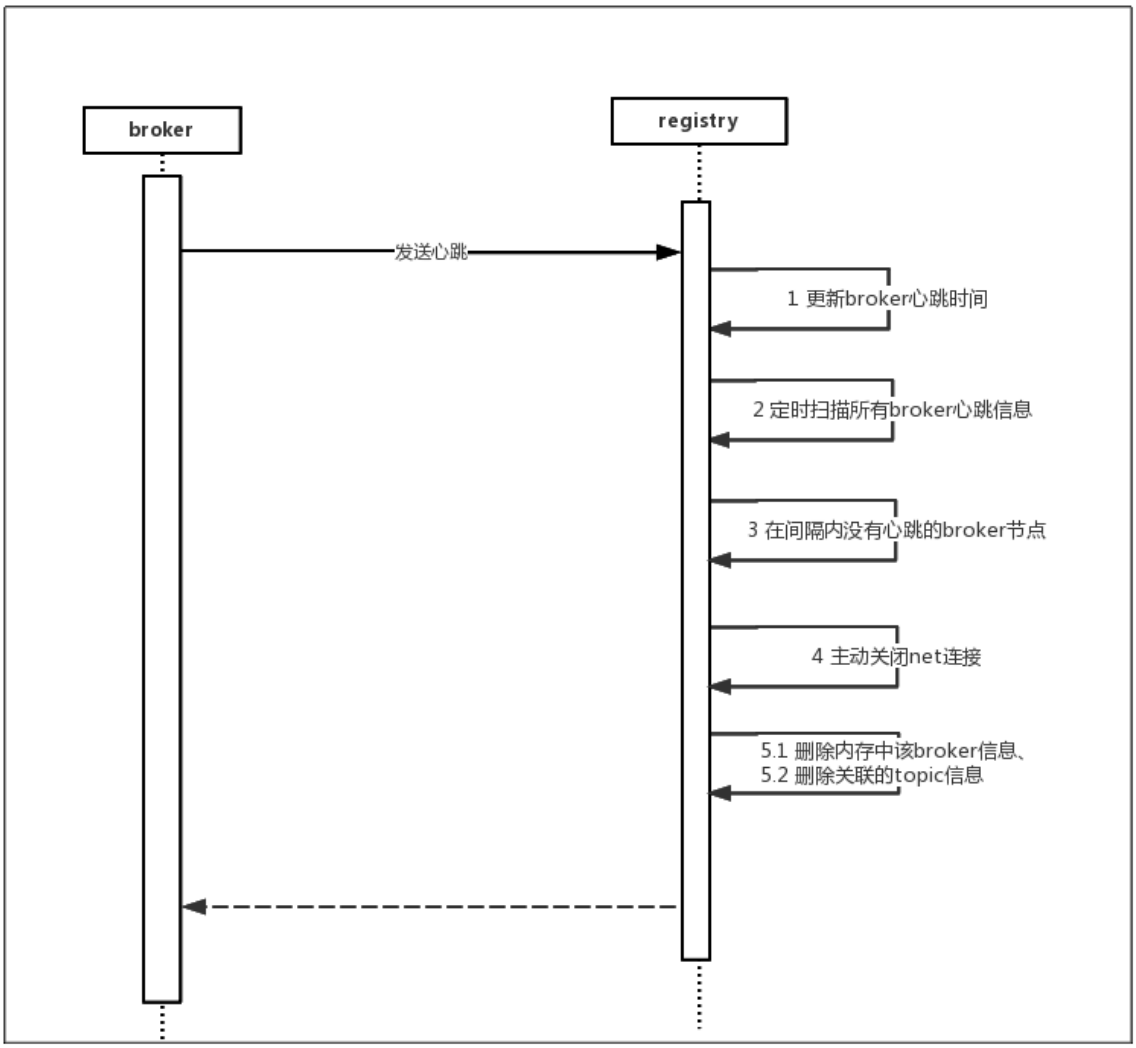
Registry 收到请求后，根据请求参数，先后更新 ClusterAddrTable、BrokerAddrTable、

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。

4.1 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。

4.1 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。

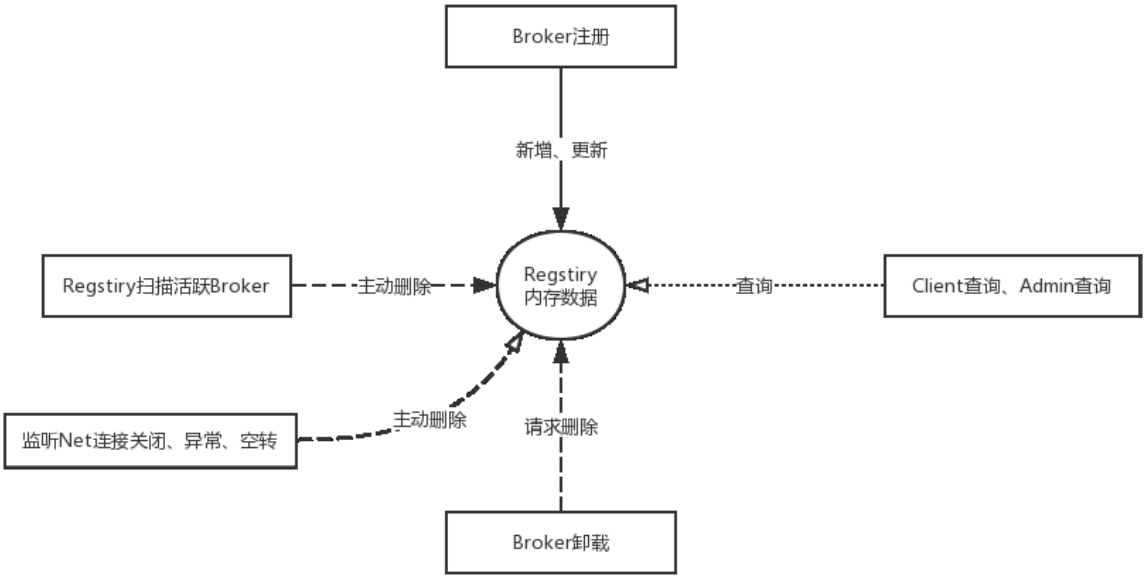
(3)既然 BoltMQ 集群中没有用到 ZooKeeper 的一些重量级的功能，只是使用 ZooKeeper 的数据一致性和发布订阅的话，与其依赖重量级的 ZooKeeper，还不如写个轻量级的 Registry。

(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。

4.2 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

4.3 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。

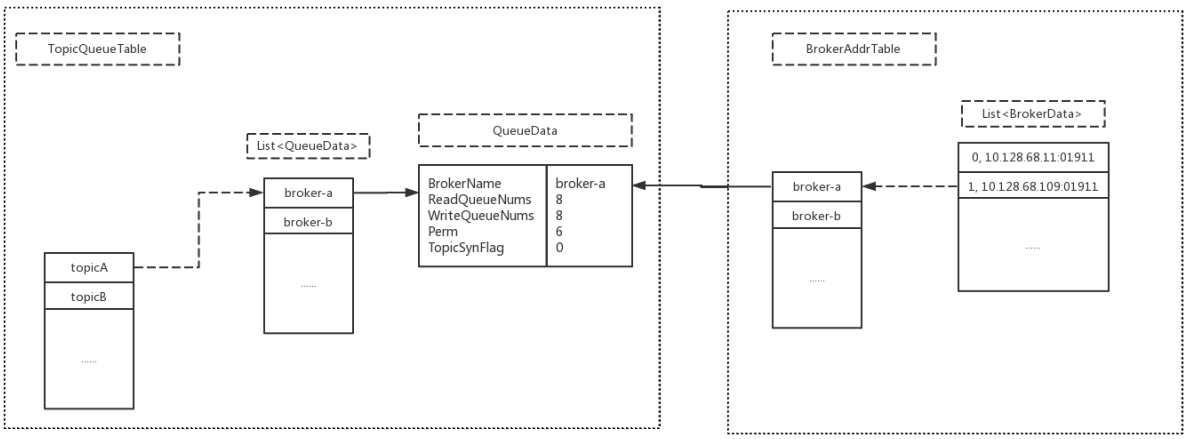
4.4 普通 Topic 与顺序 Topic

	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

4.1 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