BlotMQ-Broker

技术文档说明

针对版本V1.0.0

2017/11/21

BoltMQ-broker 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	郜焱磊 、戎志宏	基础平台架构组	2017/11/21

目 录

1 概述

Broker 消息中转角色，负责存储消息，转发消息，一般也称为 Server。在 JMS 规范中称为 Provider。

Broker 通过自身实现方法并且发布，提供 Client 调用。也就是说相对于 Client，Broker 是一个 Service。

Broker 在 BoltMQ 的角色很多，它是给 Producer、Consumer 提供服务的 Service、又是调用 Registry 服务的 Client、而它自身还承载着运维接口、消息统计等一系列的功能。

2 Borker 模块交互

2.1 Registry

- 每个 Broker 与每个 Registry 保持长连接。
- 启动时会向每一个 Registry 注册，启动过后 Broker 每隔 30 秒向 Register 发送心跳，注册和发送心跳都包含了将自身的 clusterName，brokerName，topic 信息发。如果 Broker 2 分钟内没有发送心跳数据，则断开连接。
- Broker 挂掉或者断开，Registry 会有自动感应，会更新删除该 Broker 与 Topic 的关系。

2.2 Client

- 每个 Client 通过 Registry 拿到 BrokerList 地址，Client 与 BrokerList 保持长连接。
- Producer 向 Broker 发送消息，Broker 负者处理解析消息，然后转发到 Stroe 进行消息持久化。
- Consumer 从 Broker 拉取消息进行消费，Broker 会维护 Consumer 与 Topic 之间订阅关系，并且会维护与 Topic 消费的 Offset。

2.3 Net

- Broker 通过 Net 创建 Service（目前端口为 10911），注册并发布服务，供 Client 调用。
- Broker 通过 Net 创建 Client，调用 Registry 的方法。

2.4 Store

- Broker 收到消息，经过一些列的验证，解析，重新封装后将消息交给 Store 做后续的处理。

2.5 Broker

- Broker 主节点之间没有交互，主节点与备节点同步 Topic 信息，Consumer Offset，延迟队列的 Offset，订阅关系等。

3 专业术语

3.1 **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。

3.1 **ConsumerOffset**

ConsumerOffset主要记录了Consumer GroupName与Topic每个Queue的消费进度。

3.2 **SubscriptionGroup**

SubscriptionGroup用来管理订阅组的订阅信息，包含订阅权限、重试队列，重试次数等。

4 Broker 实现原理

4.1 **Topic** 管理

- 默认 Topic

目前 Broker 启动时会生成六个默认的 Topic，OFFSET_MOVED_EVENT、SELF_TEST_TOPIC、DEFAULT_TOPIC、BENCHMARK_TOPIC、集群名称 Topic、Broker 名称 Topic。其中 DEFAULT_TOPIC 最为关键，应为 Topic 的创建会以 DEFAULT_TOPIC 为模板进行创建。目前 BoltMQ 中 DEFAULT_TOPIC 的读写队列默认为 16 个；并且是一个可读可写 Topic。

- 持久化

每个 Broker 会将其下的每一个 Topic 进行统一的持久化，这些 Topic 被全部保存到一个以 JSON 的形式都保存到一个文件中，BoltMQ 保存 Topic 文件的路径为/当前用户目录下/store/config/topic.json 文件。该文件主要保存了每一个 Topic 的主要信息如：TopicName（topic 名称）、ReadQueueNums（读队列个数）、WriteQueueNums（写队列个数）、Perm（topic 权限）、Order（是否为顺序 Topic）、topicSysFlag（系统标识）。

文件存储内容如下：

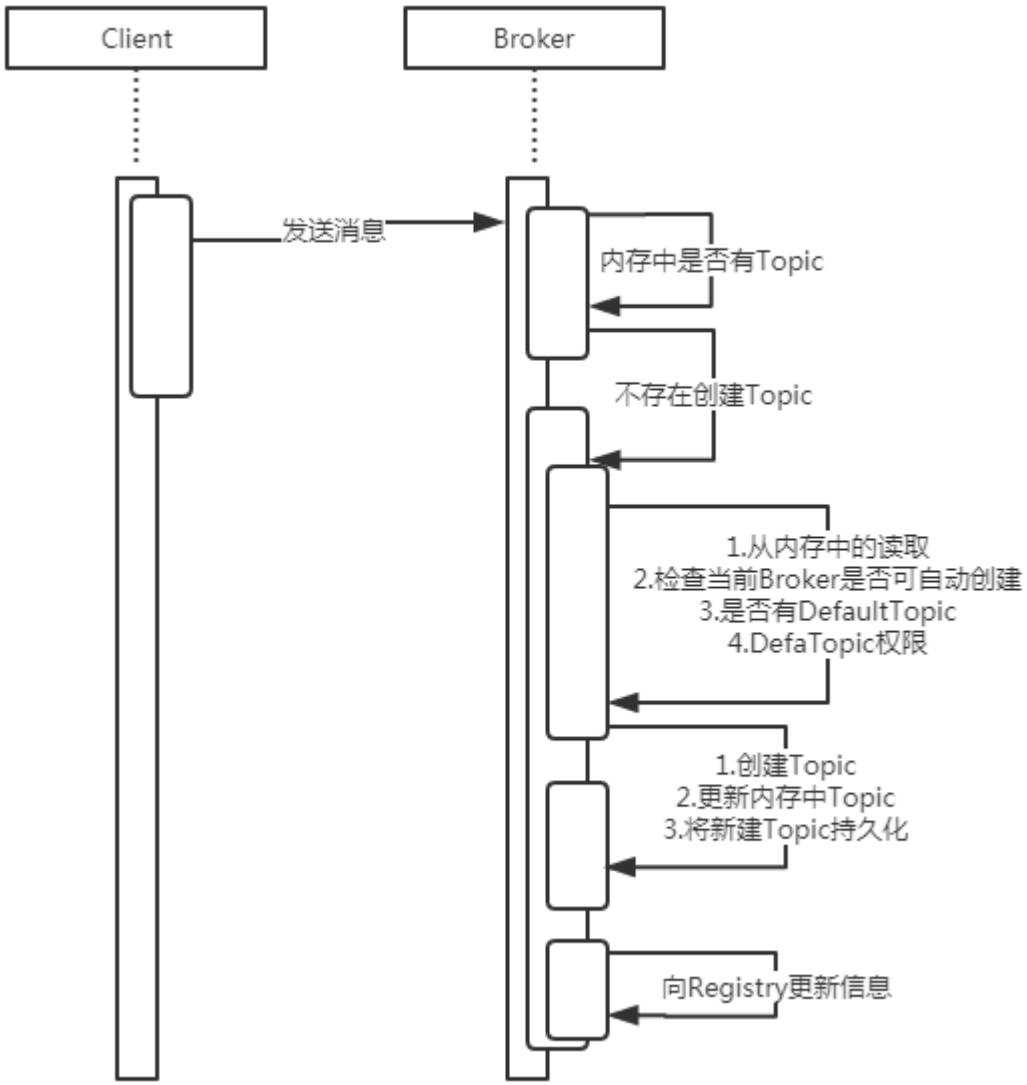
```
{
    "topicConfigTable": {
        "topicConfigTable": {
            "%RETRY%consumerGroupId-example-200": {
                "SEPARATOR": "",
                "topicName": "%RETRY%consumerGroupId-example-200",
                "readQueueNums": 1,
                "writeQueueNums": 1,
                "perm": 6,
                "topicFilterType": 0,
                "topicSysFlag": 0,
                "order": false
            }
        }
    },
    "dataVersion": {
        "timestamp": 1511333414604049700,
        "counter": 2023
    }
}
```

- 初始化

在 Broker 启动时，Broker 会将 Topic.json 文件进行加载，在内存中维护一套 Topic 名称与 Topic 对象之间的关系，对 Topic 进行任何操作，都会更新内存所维护的关系以及 Topic.json 的文件。

- 创建 Topic

创建 Topic 由 Client 发起，Broker 没有检测到 Client 所需要发送的 Topic，其创建如图所示：



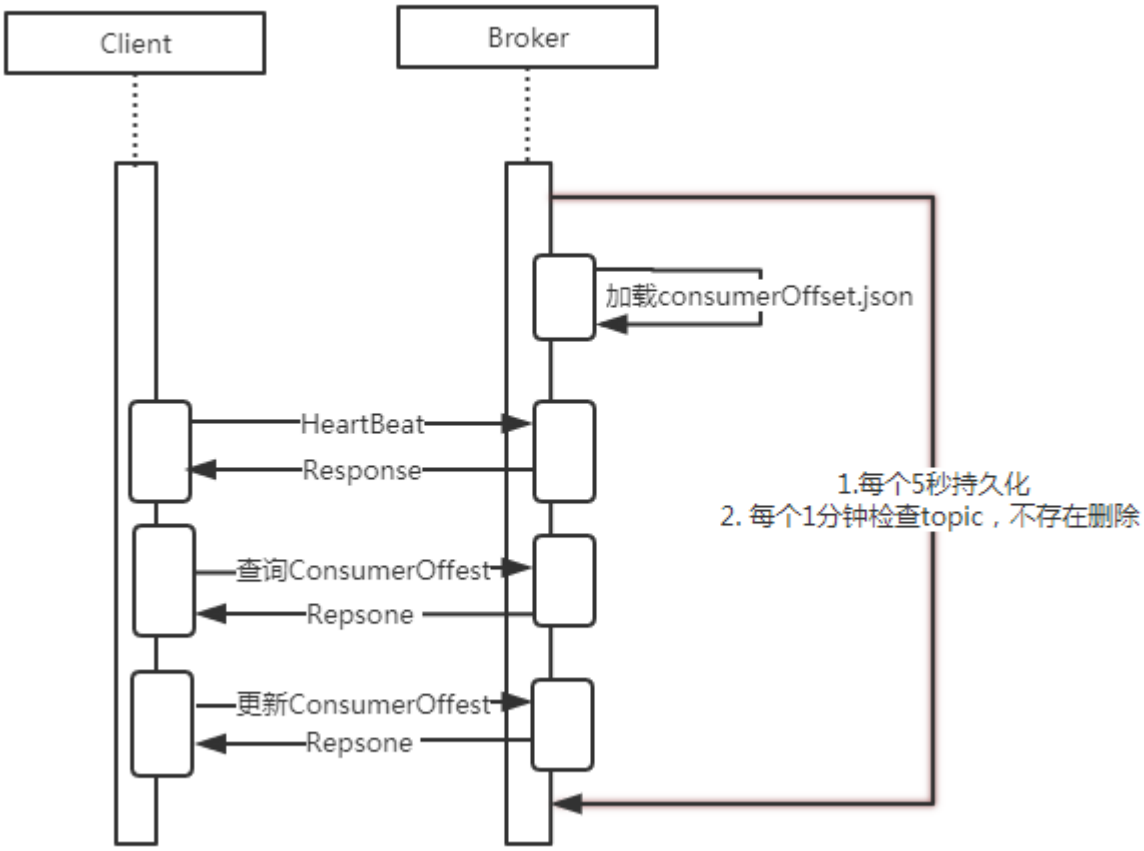
在创建 Topic 的过程中，会将创建的 Topic 的队列数与 DefaultTopic 队列数对比，取其小的队列数为新建 Topic 的队列数。创建成功后会立马向所有 Registry 注册。

- 其他操作

如果对原有的 Topic 进行了操作，会第一时间将内存维护的信息进行更新并且会刷入磁盘中。Broker 启动时会开启一个线程，每隔 30 秒向 Registry 注册，将更新的 Topic 维护到 Registry 中。

4.2 ConsumerOffset 管理

ConsumerOffset主要管理的是订阅组与Topic Queue消费进度的管理。具体流程如下：



■ 初始化

在 Broker 启动时，Broker 会将 ConsumerOffset.json 文件进行加载，在内存中维护一套以订阅组名称与 Topic 名称组合为 key，以当前 Topic 队列消费 offset 为 value 的关系。

■ 持久化

在 Broker 启动时候，Broker 会开启一个线程每个 5 秒对 Client 上报的 Consumer 与 Topic Offset 进行持久化。

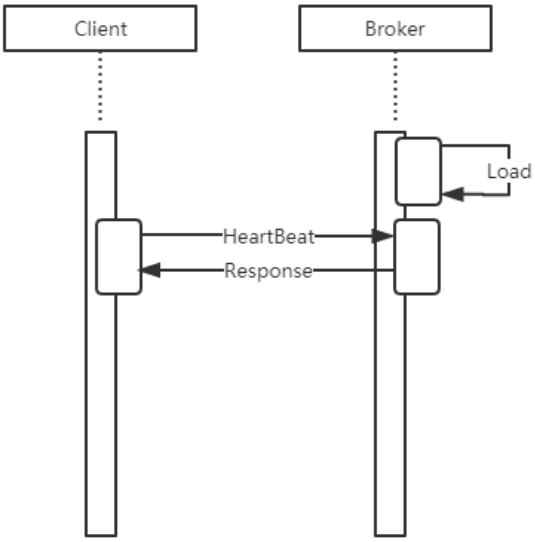
BoltMQ 保存 consumerOffset 文件的路径为/当前用户目录下/store/config/consumerOffset.json 文件。存储结构如下：

```
{
    "offsets": {
        groupName@TopicName: {
            queue 1: offset，
                ...
            queue x: offset，
        }
```

```
    }
}
```

4.3 SubscriptionGroup 管理

SubscriptionGroup用来管理订阅组的订阅信息，包含订阅权限、重试队列，重试次数等。其流程如下：



Consumer通过心跳服务进行对SubscriptionGroup来维护。

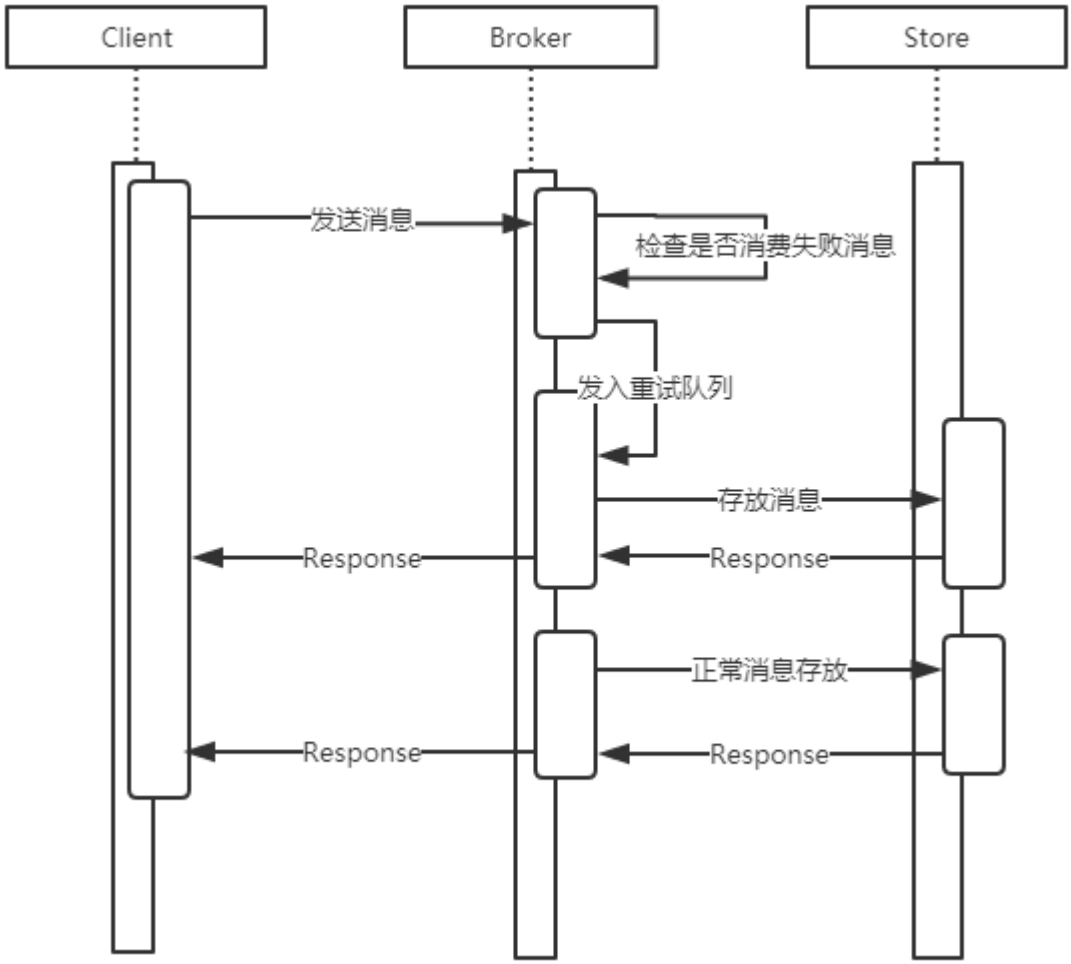
- 持久化

每当 Broker 维护 SubscriptionGroup 关系发生改变，都会进行一次持久化。BoltMQ 保存 subscriptionGroup 文件的路径为/当前用户目录下/store/config/subscriptionGroup.json。存储结构如下：

```
{
    topicName: {
            "groupName": "xx", //订阅组名称
            "consumeEnable": true, //是否可以消费
            "consumeFromMinEnable": true, //是否允许从队列最小位置开始消费，线上默认会设置为 false
            "consumeBroadcastEnable": true, //是否允许广播方式消费
            "retryQueueNums": 1, //消费失败的消息放到一个重试队列，每个订阅组配置几个重试队列
            "retryMaxTimes": 16, //重试消费最大次数，超过则投递到死信队列，不再投递，并报警
            "brokerId": 0,   //从哪个 Broker 开始消费
            "whichBrokerWhenConsumeSlowly": 0   //发现消息堆积后，将 Consumer 的消费请求重定向到另外一台 Slave 机器
        }
    },
```

```
    "dataVersion": {
        "timestamp": 1511342161274071800,
        "counter": 3
    }
}
```

4.4 发送消息



整个消息的发送流程分为两个步骤流程：

- Consumer 回退消息

  针对Consumer消费失败消息投放重试队列，Broker接收到消息检测到如果是消费失败的Consumer端回退消息。会经历一下流程：

  a) 检测当前消息的订阅组是否存在。

b) 检测当前Broker是否有写入权限

c) 获取到重试队列Topic（重试队列Topic一般为%RETRY%+groupName），计算QueueID。

d) 如果当前消息消费次数大于设置重试消费次数则投入死信队列(死信队列Topic一般为%DLQ%+GroupName)消息将不会再被消费。

e) 如果当前消息消费次数小于设置重试消费次数，则会将当前消费次数+3个等级延迟，延迟该消息的消费。

f) 重新组装消息对象调用store服务。

g) 统计

- Producter 发送正常消息

正常消息的发送情况远没有重试消息流程复杂，其流程如下：

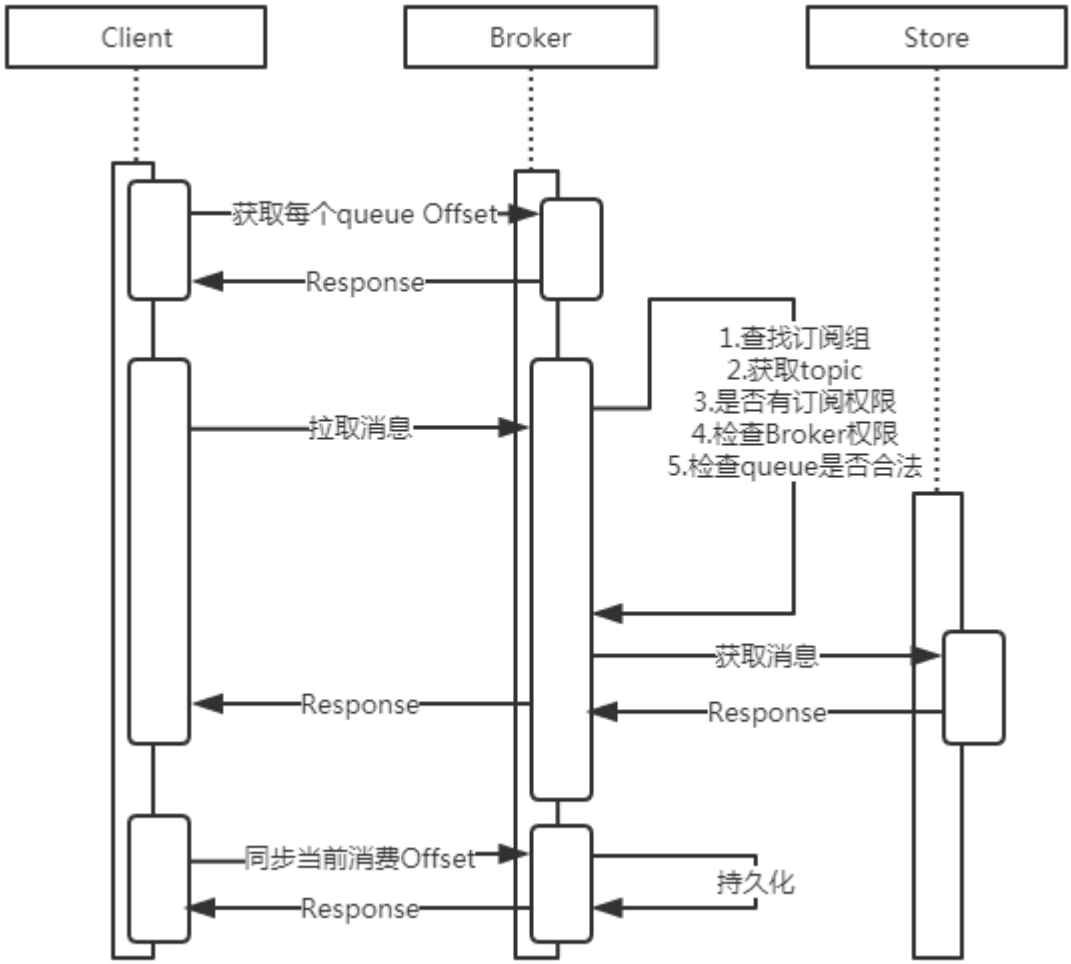
a) 检查发送消息的合法性（Topic、Broker权限等）。

b) 重新组装消息对象调用store服务。

c) 统计

4.5 消费消息

Bolt在consumer端的消费方式有两种：一种push（推）、一种pull（拉）。不管是pull与push对Broker而言都是由Consumer主动发起的pull操作。其主要流程如下：



a) 在拉取消息时，Consumer会先同步当前Broker持久化当前定于组每个Queue的Offset。

b) Consumer通过获取到的Offset作为拉取消息的开始位置，向Broker发起拉取消息请求。

c) Broker校验信息（Broker权限、订阅关系、Topic是否存在、QueueID是否有效）。

d) 去Store拉取消息。

e) Broker通过返回结果返回消息。在push模式下如果拉取消息为空，Consumer则会每隔15秒在进行一次拉取。

f) 进行统计。

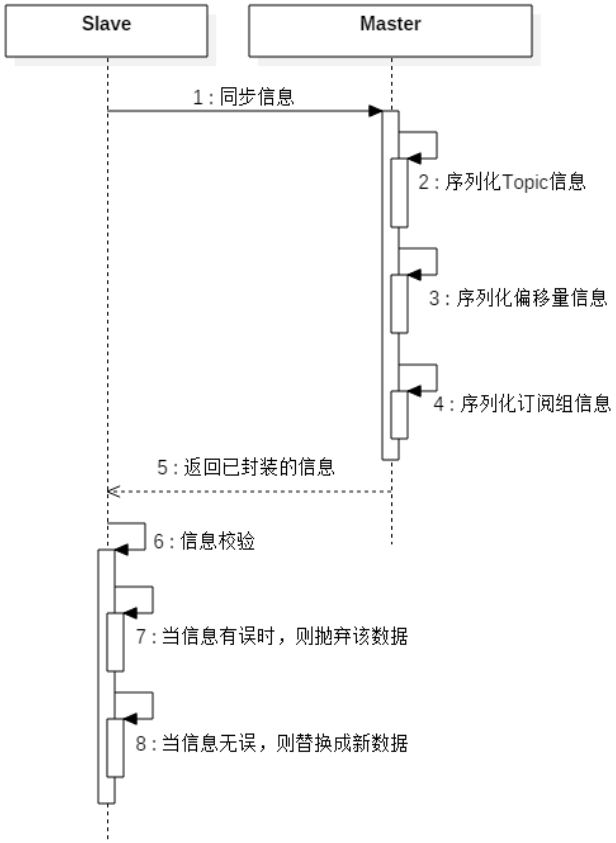
4.6 主从同步

- 主从关联

Broker集群只允许存在一个主节点，而一个主节点下可挂载一个或多个从节点。主节点与从节点通过心跳保持关联，而从节点彼此间不通信。从节点会定时从Registry查询主节点的地址，当主节点地址发生变化后，原关联就会中断，新的主从关联会重新建立。

- 同步流程

在正常情况下，主节点对外提供Topic创建、消息存储、拉取等服务，而从节点只是用于备份数据。业务Topic从节点定时同步主节点的信息。只有主节点挂了之后，才允许客户端到从节点拉取消息，但是依然不允许从节点存储消息。主从同步时序图如下：



4.7 Hold

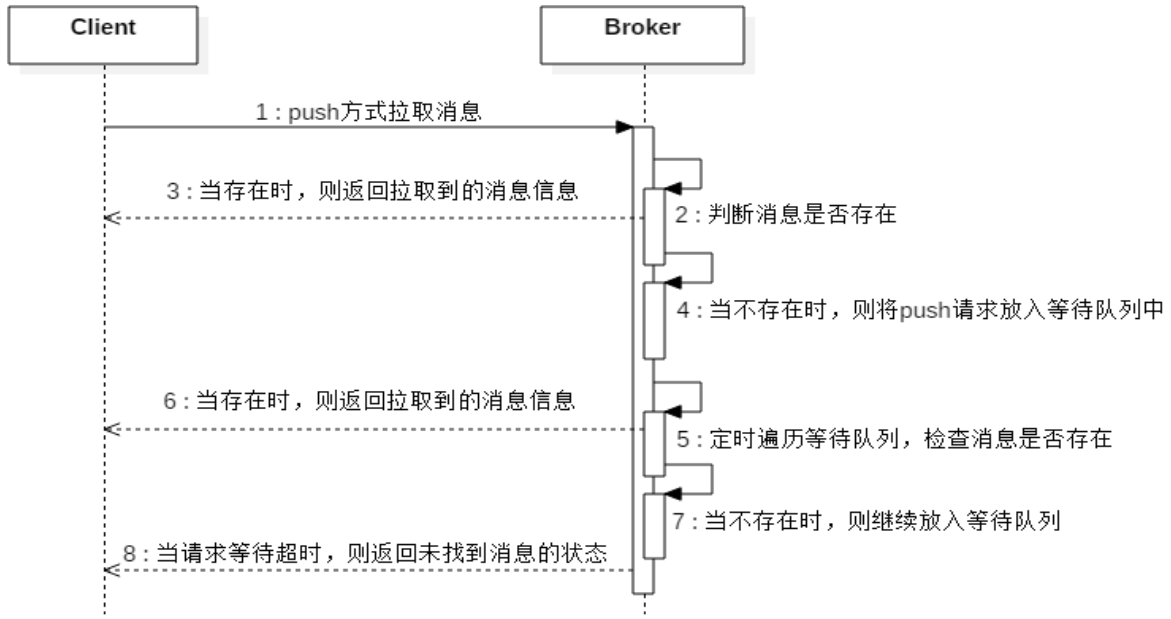
- 用途

消费端从Broker拉取消息，当消息不存在时，通过长连接或者定时发送拉取请求来实现当有新消息时能拉取到结果。当新消息到来较迟时，长连接或者定时发送请求的方式，都会造成带宽的浪费，造成Broker的

没必要的压力。Broker延缓Hold住拉取消息请求的方式可解决上诉问题，将未拉取到消息的请求放入等待队列，待新消息到来时或者等待超时的唤醒。因Hold服务记录了此次拉取请求的通道信息，所以Broker可从该通道将查询到的消息或者超时状态主动推送给给客户端，从而避免了带宽的浪费。

- Hold流程

消费端从Broker拉取某个偏移量的消息，当该偏移量比最大偏移量都大，则表明该消息还未存入Broker，无法被拉取到。将该拉取请求放入等待队列中，等待有新消息到来时的唤醒。Hold请求的序列图如下：



4.8 消息统计

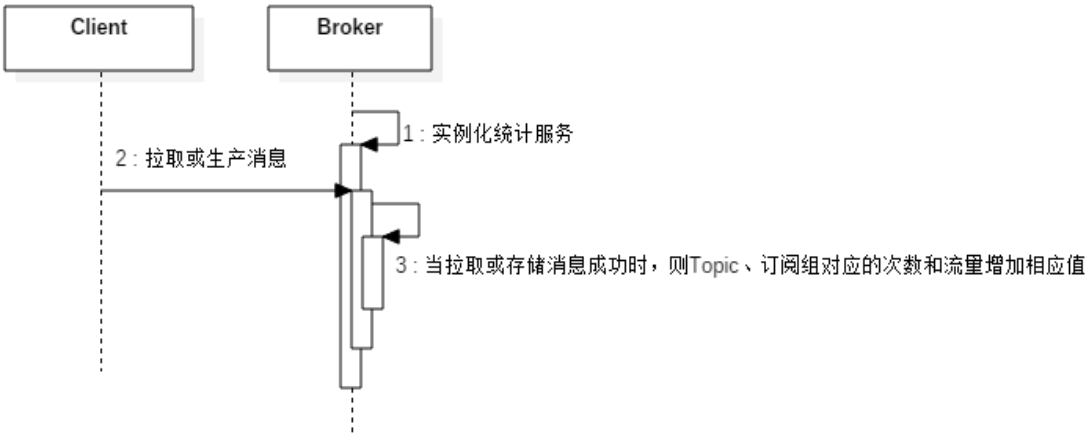
- 用途

Broker从时间、消息来源、消费等多个维度统计了的Topic、Group的put、get消息的次数、流量等，实时计算、记录了对应的Tps。根据统计数据，用户即可了解自己的业务量、业务波动等，运维即可知道当前Broker集群的处理能力，为扩容等提供可靠数据依据。

- 处理流程

Broker启动时，初始化统计服务，当Broker拉取或存储消息成功时，会在原统计数据的基础上增加相应次

数、流量值，计算并记录不同时间段的Tps。消息统计时序图如下：



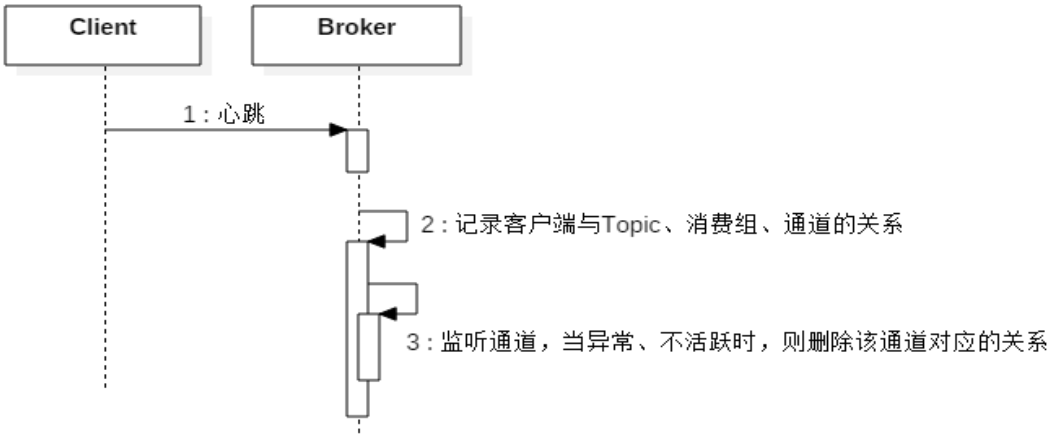
4.9 Producer、Consumer 连接管理

■ 用途

记录当前生产者、消费者与Broker相连的通道信息，维护通道其与Topic、Group、SubList关系。开发、运维人员可查看当前生产者、消费者数量，Topic对应的客户端数量、同一订阅组下客户端的数量等。

■ 连接流程

客户端通过Registry查询到Broker路由信息列表，遍历路由列表，找到业务Topic对应的Broker地址。根据Broker地址，客户端连接Broker，定时发送心跳来维护该通道连接，而Broker接收到心跳后，实时维护Channel、Topic、Group、SubList的彼此之间的关系。客户端连接管理时序图如下：



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.com	2017/11/21
戎志宏	hzrong007@163.com	2017/11/21