## Goulette设计方案

1. 项目简介

**Goulette是一个基于Go语言开发的高性能分布式消息队列，各组件之间通过grpc实现高性能的数据传递。**

1. 主要组件

客户端，哨兵集群，Broker集群。

1. 各组件详细设计

3.1. 客户端

客户端又可细分为producer和consumer，二者在初始化与broker的连接时略有不同：producer在向一个或多个broker发送消息之前，需要先向哨兵集群发送rpc请求，获取相关Topic的brokers地址，获取到brokers的地址后，与其建立grpc连接，连接建立后（调用broker的rpc函数建立一个双向流），首先发送一条元信息以表明自己将要发送的消息属于哪个Topic，随后producer借助滑动窗口的方式向broker发送消息并接收ack（发送消息给broker时需要附带一个uint64类型的递增id（从0开始），broker处理完消息后向producer发送一条ack（附带之前收到的id），producer接收到ack后就认为此条消息发送成功，滑动窗口向前移动一个单位）。consumer在从一个或多个broker拉取消息之前，需要先向哨兵集群发送rpc请求，获取相关Topic的brokers地址，获取到brokers的地址后，与其建立grpc连接，连接建立后，首先发送一条元信息以表明自己将要拉取的消息属于哪个Topic，同样采用滑动窗口的方式从broker处拉取消息。

注意：若topic1存在3个broker副本，则发送消息时需要同时向3个副本建立连接，一条消息在得到至少2（n/2 + 1）个broker的ack之后就被视为发送成功。消费时先随机建立一个连接，然后根据当前最早未被消费的fragment（见下文）得出对应的broker地址，将连接重定向至对应broker地址，从该broker上拉取消息。

3.2. broker

功能：接收/发送消息，消息的持久化存储，对每个Topic维护一个cache加速消息的消费过程，与相同Topic的其他副本进行消息的同步，与相同Topic的其他副本合作将消息数据复制到另一个新建副本中。

需要对外提供的rpc函数有：接收和发送消息的函数(客户端调用)，将某个Topic下的指定消息文件传至指定的broker(哨兵调用，用于补全副本，保证可用性)，用于相同Topic的副本之间同步消息的接口。

整体分为两大部分：第一部分负责处理rpc请求，第二部分负责将接收到的消息写入文件或从文件中读取消息。

消息的持久化方案：

逻辑结构：一个Topic下的消息按照写入时间被分为一个个fragment，每个fragment中存放有多条消息（同样按照写入时间排列），一个fragment就是一个文件，文件可存放的消息受到min(规定的最大消息总条数，规定的最大文件大小)的限制，每个fragment都会被赋予一个按时间连续递增的id。

写入过程：先通过WAL写入binlog，然后每n秒刷一次盘，将binlog中的消息按照Topic写入相应的fragment中去（每次都写最新的fragment）。

读取过程：有consumer想要读取消息时，需要根据其连接到的broker在此Topic中的id来从对应id中取出消息（例如，若topic1有3个副本（broker1, broker2, broker3），consumer1连接到broker1和broker2并拉取消息，副本之间的数据（fragment）都相同的情况下，若存在10个fragment，则broker1只能读取fragment0, fragment3, fragment6, fragment9，broekr2只能读取fragment1, fragment4, fragment7，这么做是为了提升吞吐量并且避免冲突，注意：每个broker中都拥有完整的消息数据，但只能读取指定的部分且每个fragment同时只能有一个consumer消费，consumer准备消费时，先找到最早未消费完的fragment，然后重定向至对应broker，例如topic1的最早未消费完的fragment为fragment0，但当前consumer连接的是broker2，此时就需要将连接重定向至broker1）。

3.3.哨兵

功能：通过内嵌etcd服务以提供服务发现功能，并且每个哨兵中都需要维护一个哈希环（一致性哈希）以实现自动故障转移。

需要对外提供的rpc函数有：broker keepalive（broker调用，表明此broker处于存活状态，实现上直接借助etcd的lease，每次调用就将对应的key进行续约），获取对应Topic的可用broker的函数，broker注册/下线的函数。

一致性哈希的实现：

要求需要实现的功能：

Broker注册时将其分成多个虚拟节点插入到哈希环上(例如根据broker地址生成多个字符串，通过哈希得到一个uint64后插入到哈希环上(broker1->broker1-1, broker1-2, broker1-3等))，插入的节点需要存储broker的地址信息。

给出一个topic，找到哈希环上在此topic之后的n（副本数量）个不同的broker地址。

提供一个cache结构，用于在查询之后将结果缓存到其中，下次再有查询请求就先查cache，若有则直接返回。（注意此处需要提供2种查询方式，一种会查cache，另一种不查cache(通过此方式调用时说明之前返回的结果中有Broker无法连接)，此时取出该Topic对应的brokers，逐个尝试建立grpc连接，选出其中连接正常的a个broker，随后根据此Topic再从哈希环上选取在此Topic之后的且不重复的n(副本数量)-a个broker，并开始作为协调者指挥正常的a个broker将完整的消息数据复制到新选出的副本中去，直到完整的副本大于b(最小可接受副本数)个后再将查询结果返回给客户端）。

一致性哈希环的维护：

每个哨兵节点内部都需要维护一个哈希环且内嵌一个etcd服务，每次有broker调用注册/取消注册的rpc函数时（连接的任意一个哨兵节点），此哨兵节点不立即进行注册操作，而是将此条函数调用追加到etcd的指定key下（key:prefix/序号，序号取当前时间戳。value:函数名称+参数，序列化后写入），同时所有哨兵节点都需要监听此key，检测到追加操作后取出内容并反序列化后得到相关函数名和参数，然后再到本地执行，以此做到哨兵节点之间数据的统一（写入数据的那个节点也需要通过监听来获取需要执行的操作）。新的哨兵节点重建哈希环时也只需要取出对应key下的所有函数调用记录，依次执行即可与其他节点保持同步（在重建之前需要先对追加操作进行监听并记录，在初步重建完成后再执行重建期间监听到的函数操作）。其他会改变哈希环结构的rpc调用也是同理。