RocketMQ方案介绍

个人建议:

各环境: 一台Master。

生产:1Master-2Slave - 同步双写

先弄一个简单业务接入，后续会根据上线的情况做一些简单监控告警，模拟一些并发测试。

选择理由

1. Java语言开发，降低学习维护成本。
2. 由阿里开源，并捐献给Apache维护。
3. 成熟的文档，搭建简单方便，入门快。
4. 性能好。
5. 支持延时消息、事务消息、顺序 消息。
6. 配套监控管理后台webConsole。
7. 支持消息查询，根据Key或者MessageID

更多特性介绍

消息发送方法

- 可靠同步发送 : 可靠同步发送在众多场景中被使用，例如重要的通知消息、短信通知、短信营销系统，等等。
- 可靠异步发送 : 对响应时间不敏感的推送。
- 单向发送: 例如日志收集。只发送不需要回调结果。

消息发送类型

- 普通消息：性能最高。
- 顺序消息: 先进先出顺序消息实现。
- 全局顺序消息: 性能差，一个主题只能一个读写队列，不能负载。但是能保证全局的顺序一致。
- 广播消息 : 发送之后，订阅该主题的消费者将会被查收。
- 延时消息:延时消息提供了一种不同于普通消息的实现形式——它们只会在设定的时限到了之后才被递送出去。例如支付超时等等。
- 批量消息: 批量发送消息可以提升投递小内存消息时的性能。

消息可靠性

RocketMQ支持消息的高可靠，影响消息可靠性的几种情况：

1. Broker正常关闭
2. Broker异常Crash
3. OS Crash
4. 机器掉电，但是能立即恢复供电情况
5. 机器无法开机（可能是cpu、主板、内存等关键设备损坏）
6. 磁盘设备损坏

1)、2)、3)、4) 四种情况都属于硬件资源可立即恢复情况，RocketMQ在这四种情况下能保证消息不丢，或者丢失少量数据（依赖刷盘方式是同步还是异步）。

5)、6)属于单点故障，且无法恢复，一旦发生，在此单点上的消息全部丢失。RocketMQ在这两种情况下，通过异步复制，可保证99%的消息不丢，但是仍然会有极少量的消息可能丢失。通过同步双写技术可以完全避免单点，同步双写势必会影响性能，适合对消息可靠性要求极高的场合，例如与Money相关的应用。注：RocketMQ从3.0版本开始支持同步双写。

总的来说最可靠稳定的设置方式：

1. 多Master，每个Master带有Slave
2. 主从之间设置为SYNC_MASTER
3. Producer用同步方式写；
4. 刷盘策略设置冲SYNC_FLUSH.

可以消除单点依赖，即使某台机器出现极端故障也不会丢失消息。

集群监控

RocketMQ提供了Console来查看一些可视化界面来查看nameServer、broker、message等情况。

环境搭建地址: http://192.168.0.24:9875/#/

目前已有功能:

- nameServer在线列表
- broker消费情况
- broker集群信息
- topic列表
- 消费者列表
- 生产者列表
- 消息查询
- 消息轨迹查询

目前告警这块还需要额外拓展。

- 消息追踪
- 消费者挂了，告警(目前Cat也能解决部分问题.)
- NameServer、Broker挂了告警。

高可用

RocketMQ目前支持主备模式，并且在4.5.0中引入Dledger进行容灾切换。

目前已经在192.168.0.24、192.168.0.25中搭建了两台Broker集群。

单机1Master+2Slave模式供两台。

多Master多Slave模式-异步复制

每个Master配置一个Slave，有多对Master-Slave，HA采用异步复制方式，主备有短暂消息延迟（毫秒级），这种模式的优缺点如下：

- 优点：即使磁盘损坏，消息丢失的非常少，且消息实时性不会受影响，同时Master宕机后，消费者仍然可以从Slave消费，而且此过程对应用透明，不需要人工干预，性能同多Master模式几乎一样；
- 缺点：Master宕机，磁盘损坏情况下会丢失少量消息。

多Master多Slave模式-同步双写

每个Master配置一个Slave，有多对Master-Slave，HA采用同步双写方式，即只有主备都写成功，才向应用返回成功，这种模式的优缺点如下：

- 优点：数据与服务都无单点故障，Master宕机情况下，消息无延迟，服务可用性与数据可用性都非常高；
- 缺点：性能比异步复制模式略低（大约低10%左右），发送单个消息的RT会略高，且目前版本在主节点宕机后，备机不能自动切换为主机。

可用性方面的测试:

当Master挂了之后，会直接从Slave中选举一台作为Master，客户端的收发消息不受影响。

> *建议数据复制的方式采用单Master多Slave模式-同步双写模式，因为我们目前量不大，机器资源紧缺。*

客户端集成方面

我们的项目目前都是SpringBoot的，apache目前也提供了Rocketmq-Spring

不过有些地方使用起来不是特别友好，例如tag设置方面。它是基于Spring的messaging规范进行封装的，所以里面非常多的对象转换。

当然也可以自己封装一套基于RocketMQ-example，最原始，而且如果后期不会打算更换其他消息队列组件的话，个人倾向于这种。

最佳实践

官网的最佳实践

提高消费者的处理能力

一 . 提高消费者并行度。

增加消费者

其实就是增加同一个组的内的消费者，把消息均衡处理掉。

需要注意的是: 消费者数量不要超过topic下Read Queue数量，超过的Consumer实例接收不到消息。

提高处理线程数

其次就是提高单个Consumer实例中的并行处理线程数，可以在同一个Consumer内增加并行度来提高吞吐量。【设置方式是修改consumer.setConsumeThreadMin和consumer.setConsumeThreadMax】

二. 以批量的方式进行消费

某些业务的场景下，多条消息同时处理的时间会大大小于逐个处理都是时间总和，比如批量修改10条数据比一次次修改10条数据会快。

实现方式是通过设置consumer.setConsumeMessageBatchMaxSize这个参数，默认是1。

应用场景实践类文章

微众银行的金融级消息服务平台建设实践和思考

消息规模超千亿，同程艺龙的消息系统建设实践

客户端最佳实践

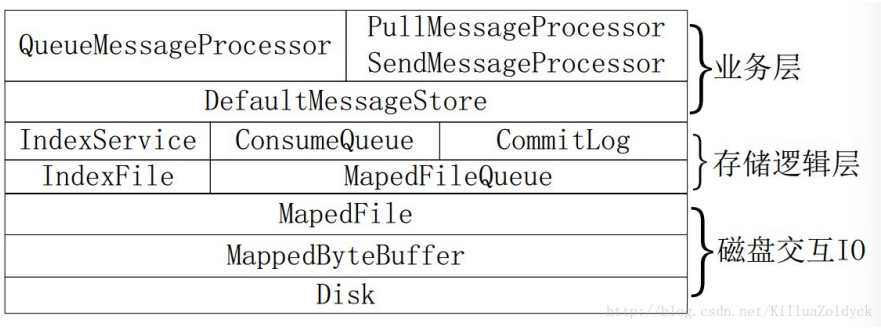
信用算力基于 RocketMQ 实现金融级数据服务的实践

作者简书

提问

消息类型

消息如何存储?



参考博客

存储在磁盘中

1. 顺序消息是如何实现的?

顺序消息是一对一发送的，也就是说这一类型的消息会被发往同一个队列(重写加入队列的选择部分`MessageQueueSelector接口`)，而这个队列会被单独的一个消费者消费掉，这就保证了顺序性质。

那么如何确保这组消息能被发往同一个队列呢?

举例:同一个topic默认会有固定的4个读写队列，那么如果保证topic下面的一组消息落到同一个队列呢?

可以通过消息队列的Key来做，比如同一个订单，用订单编号来发送这一组消息。

2. 如何知道该消息是否被消费过?消费不成功如何查询?重试的消息如何查看?

这个可以根据客户端的业务，定义一个消息日志表，这个表里面分别处理消息被消费过几次。哪个消费者消费的。

当然Rocketmq控制台只需要提供该消息是否被消费过。

3. 如果发送的消息没有对应的订阅消费者，那么这类型的消息会怎么样?

消息会保存到队列中，直到有消费者上线。注意：每个不同的消费组第一次上线都会获取该类型的所有消息。

4. 如果有两个不同组的消费者，都是消费消息类型A的消息的，如果这时候消费者2挂掉了，那么类型A消息过来，消费1消费过了，消费2重启之后还会收到这条消息吗?

会收到，这涉及到消费消息的顺序点。非新的消费组会接着上次没有消费的消息开始消费。

这个需要通过 `setConsumeFromWhere` 配置。

注意: 这个参数只对一个新的consumeGroup第一次启动时有效

ConsumeFromWhere

- CONSUME_FROM_LAST_OFFSET: 第一次启动则会从最后开始消费，后续再启动接着上次消费的进度开始消费
- CONSUME_FROM_FIRST_OFFSET: 从头开始消费，后续再启动接着上次消费的进度开始消费
- CONSUME_FROM_TIMESTAMP: 从指定的时间点开始消费，后续再启动接着上次消费的进度开始消费.