

Rocket MQ 使用排查指南

云运维工程师从入门到精通

作者：雪萌 良琪

- 6个要点掌握RocketMQ原理
- 5步教程快速入门RocketMQ
- 100+常见问题排查精解



阿里云开发者电子书系列



云服务技术大学
云产品干货高频分享



云服务技术课堂
和大牛零距离沟通



阿里云开发者“藏经阁”
海量免费电子书下载

目录

RocketMQ 原理及快速入门	4
什么是消息队列 RocketMQ	4
RocketMQ 快速入门教程	16
发送问题排查	24
客户端发送性能问题	24
客户端发送常见异常报错	27
消费问题排查	43
堆积负载常见问题	43
消息投递重试问题	52
订阅关系问题	57
客户端消费常见异常报错	59
消息轨迹常见问题	75
其他问题排查	81
控制台相关问题	81
告警监控相关问题	86
资源包费用相关问题	87
日志相关问题	91
权限相关问题	97
使用常见问题	98

RocketMQ 原理及快速入门

什么是消息队列 RocketMQ

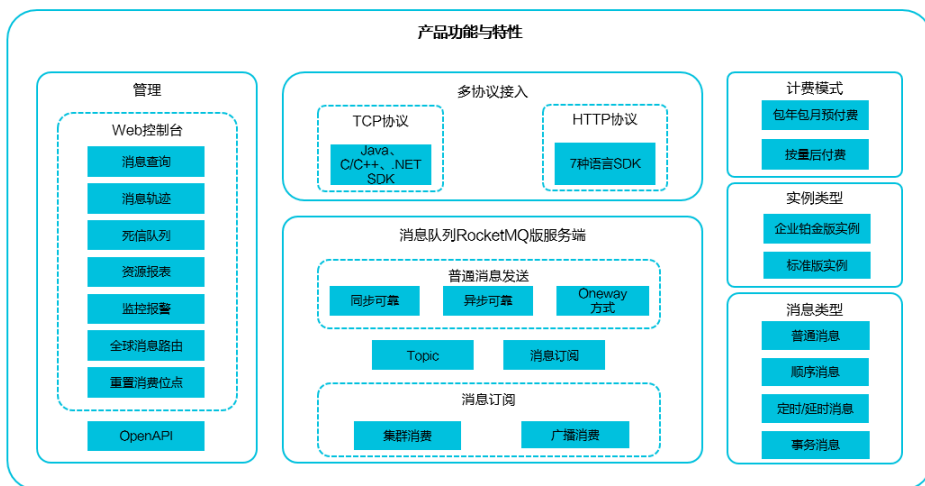
核心概念

消息队列 RocketMQ 版是阿里云基于 Apache RocketMQ 构建的低延迟、高并发、高可用、高可靠的分布式消息中间件。消息队列 RocketMQ 版既可为分布式应用系统提供异步解耦和削峰填谷的能力，同时也具备互联网应用所需的海量消息堆积、高吞吐、可靠重试等特性。

产品功能与特性

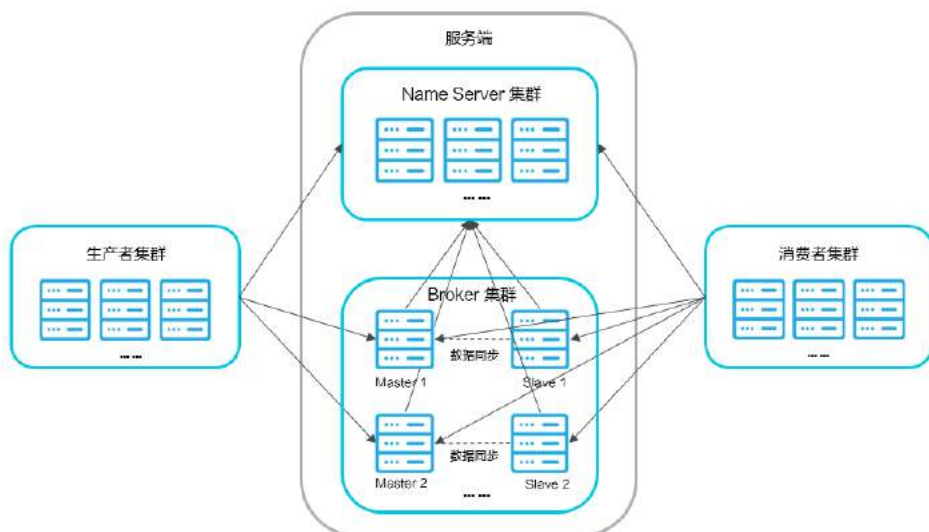
消息队列 RocketMQ 版在阿里云多个地域 (Region) 提供了高可用消息云服务。单个地域内采用多机房部署，可用性极高，即使整个机房都不可用，仍然可以为应用提供消息发布服务。

消息队列 RocketMQ 版提供 TCP 和 HTTP 协议的多语言接入方式，方便不同编程语言开发的应用快速接入消息队列 RocketMQ 版消息云服务。您可以将应用部署在阿里云 ECS、企业自建云，或者嵌入到移动端、物联网设备中与消息队列 RocketMQ 版建立连接进行消息收发；同时，本地开发者也可以通过公网接入消息队列 RocketMQ 版服务进行消息收发。



系统部署架构

系统部署架构如下图所示。



图中所涉及到的概念如下所述：

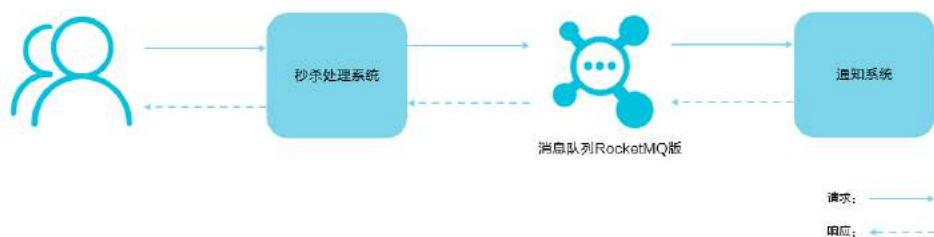
- Name Server：是一个几乎无状态节点，可集群部署，在消息队列 RocketMQ 版中提供命名服务，更新和发现 Broker 服务。
- Broker：消息中转角色，负责存储消息，转发消息。分为 Master Broker 和 Slave Broker，一个 Master Broker 可以对应多个 Slave Broker，但是一个 Slave Broker 只能对应一个 Master Broker。Broker 启动后需要完成一次将自己注册至 Name Server 的操作；随后每隔 30s 定期向 Name Server 上报 Topic 路由信息。
- 生产者：与 Name Server 集群中的其中一个节点（随机）建立长链接（Keep-alive），定期从 Name Server 读取 Topic 路由信息，并向提供 Topic 服务的 Master Broker 建立长链接，且定时向 Master Broker 发送心跳。
- 消费者：与 Name Server 集群中的其中一个节点（随机）建立长连接，定期从 Name Server 拉取 Topic 路由信息，并向提供 Topic 服务的 Master Broker、Slave Broker 建立长连接，且定时向 Master Broker、Slave Broker 发送心跳。Consumer 既可以从 Master Broker 订阅消息，也可以从 Slave Broker 订阅消息，订阅规则由 Broker 配置决定。

应用场景

削峰填谷

流量削峰也是消息队列 RocketMQ 版的常用场景，一般在秒杀或团队抢购活动中使用广泛。

在秒杀或团队抢购活动中，由于用户请求量较大，导致流量暴增，秒杀的应用在处理如此大量的访问流量后，下游的通知系统无法承载海量的调用量，甚至会导致系统崩溃等问题而发生漏通知的情况。为解决这些问题，可在应用和下游通知系统之间加入消息队列 RocketMQ 版。



秒杀处理流程如下所述：

1. 用户发起海量秒杀请求到秒杀业务处理系统。
2. 秒杀处理系统按照秒杀处理逻辑将满足秒杀条件的请求发送至消息队列 RocketMQ 版。
3. 下游的通知系统订阅消息队列 RocketMQ 版的秒杀相关消息，再将秒杀成功的消息发送到相应用户。
4. 用户收到秒杀成功的通知。

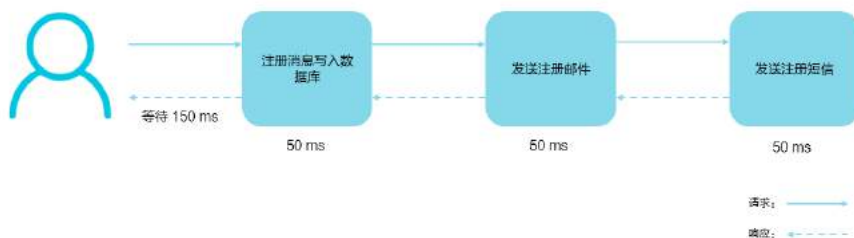
异步解耦

传统处理

最常见的一个场景是用户注册后，需要发送注册邮件和短信通知，以告知用户注册成功。传统的做法有以下两种：

(1) 串行方式

串行方式下的注册流程如下图所示。



数据流动如下所述：

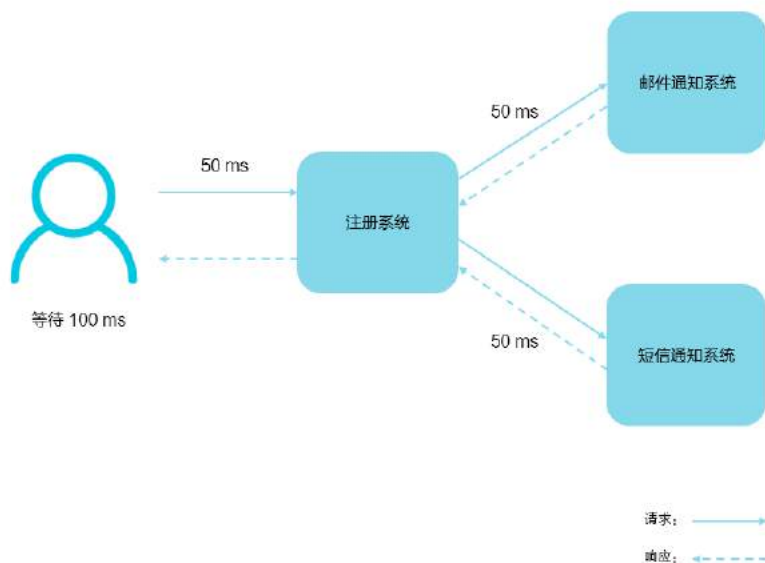
- 用户在注册页面填写账号和密码并提交注册信息，这些注册信息首先会被写入注册系统成功。
- 注册信息写入注册系统成功后，再发送请求至邮件通知系统。邮件通知系统收到请求后向用户发送邮件通知。
- 邮件通知系统接收注册系统请求后再向下游的短信通知系统发送请求。短信通知系统收到请求后向用户发送短信通知。

以上三个任务全部完成后，才返回注册结果到客户端，用户才能使用账号登录。

假设每个任务耗时分别为 50 ms，则用户需要在注册页面等待总共需要 150 ms 才能登录。

(2) 并行方式

并行方式下的注册流程如下图所示。



数据流动如下所述：

- 用户在注册页面填写账号和密码并提交注册信息，这些注册信息首先会被写入注册系统成功。
- 注册信息写入注册系统成功后，再同时发送请求至邮件和短信通知系统。邮件和短信通知系统收到请求后分别向用户发送邮件和短信通知。

以上两个任务全部完成后，才返回注册结果到客户端，用户才能使用账号登录。

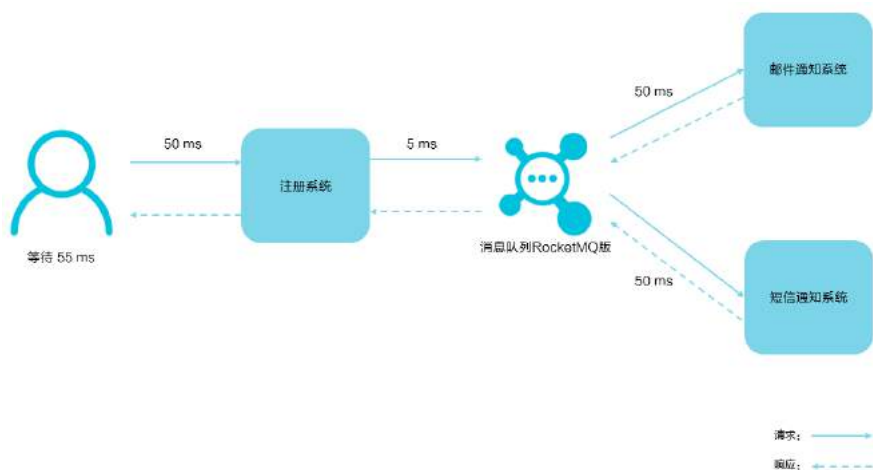
假设每个任务耗时分别为 50 ms，其中，邮件和短信通知并行完成，则用户需要在注册页面等待总共需要 100 ms 才能登录。

以下就注册场景中使用了消息队列 RocketMQ 版的效果进行说明。

异步解耦

对于用户来说，注册功能实际只需要注册系统存储用户的账户信息后，该用户便可以登录，后续的注册短信和邮件不是即时需要关注的步骤。

对于注册系统而言，发送注册成功的短信和邮件通知并不一定要绑定在一起同步完成，所以实际当数据写入注册系统后，注册系统就可以把其他的操作放入对应的消息队列 RocketMQ 版中然后马上返回用户结果，由消息队列 RocketMQ 版异步地进行这些操作。



数据流动如下所述：

- 用户在注册页面填写账号和密码并提交注册信息，这些注册信息首先会被写入注册系统成功。
- 注册信息写入注册系统成功后，再发送消息至消息队列 RocketMQ 版。消息队列 RocketMQ 版会马上返回响应给注册系统，注册完成。用户可立即登录。
- 下游的邮件和短信通知系统订阅消息队列 RocketMQ 版的此类注册请求消息，即可向用户发送邮件和短信通知，完成所有的注册流程。

用户只需在注册页面等待注册数据写入注册系统和消息队列 RocketMQ 版的时间，即等待 55 ms 即可登录。

异步解耦是消息队列 RocketMQ 版的主要特点，主要目的是减少请求响应时间和解耦。主要的适用场景就是将比较耗时而且不需要即时（同步）返回结果的操作作为消息放入消息队列。同时，由于使用了消息队列 RocketMQ 版，只要保证消息格式不变，消息的发送方和接收方并不需要彼此联系，也不需要受对方的影响，即解耦和。

顺序收发

消息队列 RocketMQ 版顺序消息分为两种情况：

全局顺序：对于指定的一个 Topic，所有消息将按照严格的先入先出（FIFO）的顺序，进行顺序发布和顺序消费；

分区顺序：对于指定的一个 Topic，所有消息根据 Sharding Key 进行区块分区，同一个分区内的消息将按照严格的 FIFO 的顺序，进行顺发布和顺序消费，可以保证一个消息被一个进程消费。

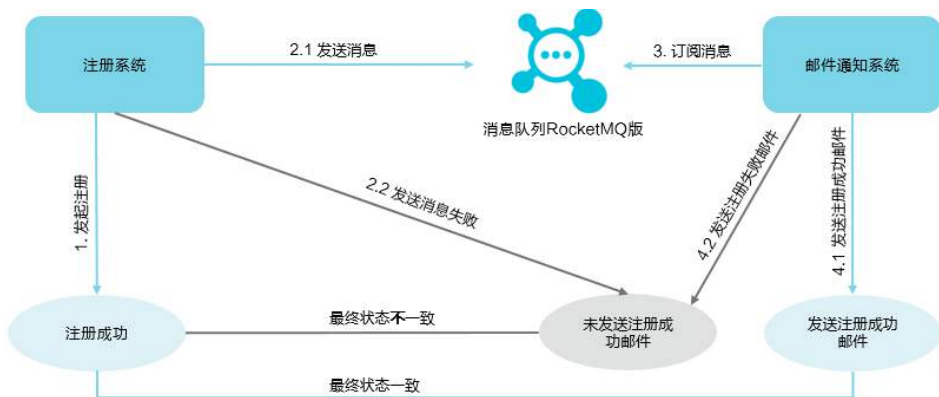
在注册场景中，可使用用户 ID 作为 Sharding Key 来进行分区，同一个分区下的新建、更新或删除注册信息的信息必须按照 FIFO 的顺序发布和消费。

分布式事务一致性

注册系统注册的流程中，用户入口在网页注册系统，通知系统在邮件系统，两个系统之间的数据需要保持最终一致。

普通消息处理

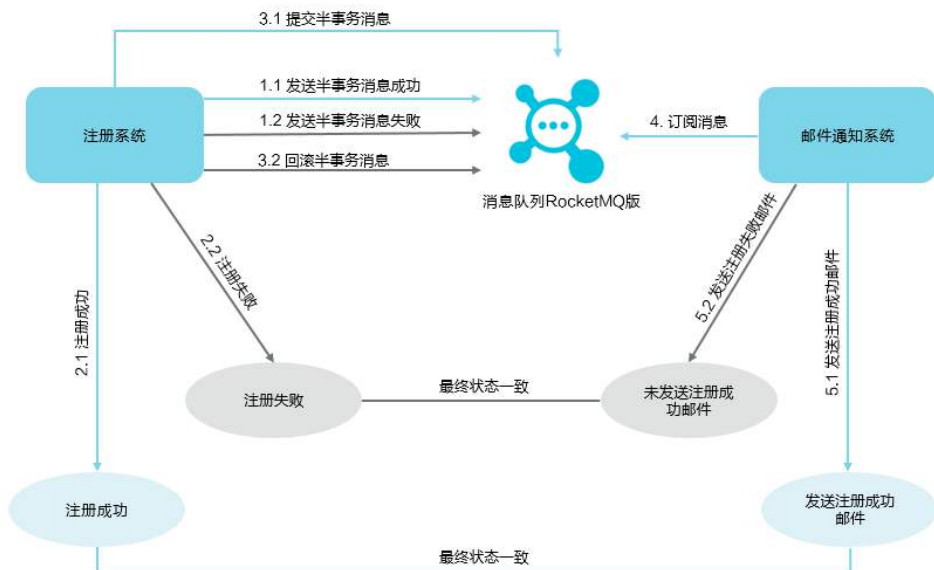
如上所述，注册系统和邮件通知系统之间通过消息队列进行异步处理。注册系统将注册信息写入注册系统之后，发送一条注册成功的消息到消息队列 RocketMQ 版，邮件通知系统订阅消息队列 RocketMQ 版的注册消息，做相应的业务处理，发送注册成功或者失败的邮件。



流程说明如下：

1. 注册系统发起注册。
2. 注册系统向消息队列 RocketMQ 版发送注册消息成功与否的消息。
 - 2.1 消息发送成功，进入 3。
 - 2.2 消息发送失败，导致邮件通知系统未收到消息队列 RocketMQ 版发送的注册成功与否的消息，而无法发送邮件，最终邮件通知系统和注册系统之间的状态数据不一致。
3. 邮件通知系统收到消息队列 RocketMQ 版的注册成功消息。
4. 邮件通知系统发送注册成功邮件给用户。

在这样的情况下，虽然实现了系统间的解耦，上游系统不需要关心下游系统的业务处理结果；但是数据一致性不好处理，如何保证邮件通知系统状态与注册系统状态的最终一致。



流程说明如下：

1. 注册系统向消息队列 RocketMQ 版发送半事务消息。
 - 1.1 半事务消息发送成功，进入 2。
 - 1.2 半事务消息发送失败，注册系统不进行注册，流程结束。（最终注册系统与邮件通知系统数据一致）
2. 注册系统开始注册。
 - 2.1 注册成功，进入 3.1。
 - 2.2 注册失败，进行 3.2。
3. 注册系统向消息队列 RocketMQ 版发送半消息状态。
 - 3.1 提交半事务消息，产生注册成功消息，进入 4。
 - 3.2 回滚半事务消息，未产生注册成功消息，流程结束。（最终注册系统与邮件通知系统数据一致）
4. 邮件通知系统接收消息队列 RocketMQ 版的注册成功消息。
5. 邮件通知系统发送注册成功邮件。（最终注册系统与邮件通知系统数据一致）

大规模机器的缓存同步

双十一大促时，各个分会场会有琳琅满目的商品，每件商品的价格都会实时变化。使用缓存技术也无法满足对商品价格的访问需求，缓存服务器网卡满载。访问较多次商品价格查询影响会场页面的打开速度。

此时需要提供一种广播机制，一条消息本来只可以被集群的一台机器消费，如果使用消息队列 RocketMQ 版的广播消费模式，那么这条消息会被所有节点消费一次，相当于把价格信息同步到需要的每台机器上，取代缓存的作用。

消息类型

普通消息

普通消息是指消息队列 RocketMQ 版中无特性的消息，即发送到服务端会立马被消费的消息，且消息是无序消费，不会按照发送的顺序一次顺序消费。

定时消息

Producer 将消息发送到消息队列 RocketMQ 版服务端，但并不期望立马投递这条消息，而是推迟到在当前时间点之后的某一个时间投递到 Consumer 进行消费，该消息即定时消息。

延时消息

Producer 将消息发送到消息队列 RocketMQ 版服务端，但并不期望立马投递这条消息，而是延迟一定时间后才投递到 Consumer 进行消费，该消息即延时消息。

定时消息与延时消息在代码配置上存在一些差异，但是最终达到的效果相同：消息在发送到消息队列 RocketMQ 版服务端后并不会立马投递，而是根据消息中的属性延迟固定时间后才投递给消费者。

全局顺序消息

对于指定的一个 Topic，所有消息按照严格的先入先出（FIFO）的顺序进行发布和消费。

分区顺序消息

对于指定的一个 Topic，所有消息根据 Sharding Key 进行区块分区。同一个分区内的消息按照严格的 FIFO 顺序进行发布和消费。Sharding Key 是顺序消息中用来区分不同分区的关键字段，和普通消息的 Message Key 是完全不同的概念。

事务消息

消息队列 RocketMQ 版提供类似 X/Open XA 的分布事务功能，通过消息队列 RocketMQ 版的事务消息能达到分布式事务的最终一致。

SDK 支持语言及协议

RocketMQ 支持 tcp 协议以及 http 协议的接入。

其中推荐使用阿里推出的 tcp 协议下的三大 sdk: java, C/C++, .NET。

除了阿里推出的 sdk，我们还支持开源的多语言 sdk 接入阿里云 RocketMQ: java, go, python, C++。

如果您想使用多语言的 sdk，推荐使用 http 协议接入: java, PHP, go, Python, Nodejs, C#, C++。

具体协议以及 sdk 的获取参考链接：

https://help.aliyun.com/document_detail/124693.html?spm=a2c4g.11186623.6.582.104e425cW5Tbm2

RocketMQ 快速入门教程

如果您使用的是阿里云主账号，则可以通过本文来体验从开通服务、创建资源、到使用 SDK 收发消息的完整流程，快速上手消息队列 RocketMQ 版。

无论您使用的是消息队列 RocketMQ 版支持的何种协议、何种语言，前三个步骤都一致，只是在控制台上具体填写的信息会略有不同，请以控制台说明为准。但在调用 SDK 时，不同协议和语言的示例代码有所不同，本文以 TCP 协议下的 Java SDK 为例进行说明。

步骤一：开通服务

1. 在[消息队列 RocketMQ 版产品页](#)，单击立即开通。
2. 在确认订单页面，选择我已阅读并同意《消息队列 MQ 服务协议》，再单击立即开通即可完成开通。

步骤二：创建资源

在使用消息队列 RocketMQ 版时，请注意以下网络访问限制：

- Topic 和 Group ID 需创建在同一个地域 (Region) 下的同一个实例中才能互通。例如，当某 Topic 创建在华东 1 (杭州) 下的实例 A 中，那么该 Topic 只能被在华东 1 (杭州) 下的实例 A 中创建的 Group ID 对应的生产端和消费端访问。
- 如果只是测试，或者需要在本地 (非阿里云 ECS 服务器) 使用消息队列 RocketMQ 版的服务，请将 Topic 和 Group ID 都创建在公网地域下的实例中。生产端和消费端可以部署在本地或者部署在任意地域的 ECS 上，前提是本地服务器或者相应的 ECS 需要能够访问公网。

创建实例

实例是用于消息队列 RocketMQ 版服务的虚拟机资源，会存储消息主题 (Topic) 和客户端 ID (Group ID) 信息。

1. 登录[消息队列 RocketMQ 版控制台](#)。在页面顶部导航栏，选择地域，如公网地域。
2. 在左侧导航栏，单击实例详情。
3. 在实例详情页面右上角，单击创建实例按钮。
4. 在创建实例对话框，选择实例类型，并输入实例名和描述，然后单击确认。

创建 Topic

Topic 是消息队列 RocketMQ 版里对消息的一级归类，例如可以创建 Topic_Trade 这一 Topic 来识别交易类消息，消息生产者将消息发送到 Topic_Trade，而消息消费者则通过订阅该 Topic 来获取和消费消息。

- Topic 不能跨实例使用，例如在实例 A 中创建的 Topic A 不能在实例 B 中使用。
- Topic 名称必须在同一实例中是唯一的。
- 您可创建不同的 Topic 来发送不同类型的消息，例如用 Topic A 发送普通消息，Topic B 发送事务消息，Topic C 发送定时 / 延时消息。

1. 在控制台左侧导航栏，单击 Topic 管理。
2. 在 Topic 管理页面上方选择刚创建的实例，单击创建 Topic 按钮。
3. 在创建 Topic 对话框中的 Topic 一栏，输入 Topic 名称，选择该 Topic 对应的消息类型，输入该 Topic 的备注内容，然后单击确定。

您创建的 Topic 将出现在 Topic 列表中。

创建 Group ID

创建完实例和 Topic 后，您需要为消息的消费者（或生产者）创建客户端 ID，即 Group ID 作为标识。

- Group ID 必须在同一实例中是唯一的。
- Group ID 和 Topic 的关系是 N: N，即一个消费者可以订阅多个 Topic，同一个 Topic 也可以被多个消费者订阅；一个生产者可以向多个 Topic 发送消息，同一个 Topic 也可以接收来自多个生产者的消息。

说明：消费者必须有对应的 Group ID，生产者不做强制要求。

1. 在控制台左侧导航栏，单击 Group 管理。
2. 在 Group 管理页面上方选择刚创建的实例，然后选择 TCP 协议 > 创建 Group ID。本文以 TCP 协议为例。

说明：TCP 和 HTTP 协议下的 Group ID 不可以共用，因此需分别创建。

3. 在创建 Group ID 对话框中，输入 Group ID 和描述，然后单击确认。

创建阿里云 AccessKey

阿里云 AccessKey 用于收发消息时进行账户鉴权。

在调用 SDK 发送和订阅消息的时候，除了需要指定创建的 Topic 和 Group ID 以外，还需输入您在 RAM 控制台创建的身份验证信息，即 AccessKey。AccessKey 的信息包含 AccessKeyId 和 AccessKeySecret。

步骤三：获取接入点

在控制台创建好资源后，您需通过控制台获取实例的接入点。在收发消息时，您需要为生产端和消费端配置该接入点，以此接入某个具体实例或地域的服务。

1. 在控制台左侧导航栏，单击实例详情。
2. 在实例详情页面上方选择刚创建的实例。
3. 在默认显示的实例信息页签的获取接入点信息区域，您可以分别看到新创建实例的 TCP 和 HTTP 协议接入点。接入点性质因协议而异，具体说明如下：
 - TCP 协议：您在控制台看到的 TCP 协议接入点是地域下某个具体实例的接入点。同一地域下的不同实例的接入点各不相同。
 - HTTP 协议：您在控制台看到的 HTTP 协议接入点是某个地域的接入点，跟具体实例无关。您在收发消息时还需另外设置实例 ID。
4. 在 TCP 协议的接入点区域，单击复制。

对于 TCP 协议的接入点，您还可以单击示例代码，查看在各种开发语言的程序中如何设置接入点。

完成以上准备工作后，您就可以运行示例代码，用消息队列 RocketMQ 版进行消息发送和订阅了。

步骤四：发送消息

您可以通过以下方式发送消息：

控制台发送消息：用于快速验证 Topic 资源的可用性，主要用作测试。

1. 在控制台左侧导航栏，单击 Topic 管理。
2. 在 Topic 管理页面，找到您刚刚创建的 Topic，单击右侧操作列的发送。
3. 在发送消息对话框中的 Message Body 一栏，输入消息的具体内容，单击确定。

控制台会返回消息发送成功通知以及相应的 Message ID。

调用 SDK 发送消息：用于生产环境下使用消息队列 RocketMQ 版。

下文以调用 TCP Java SDK 为例进行说明。

调用 TCP Java SDK 发送消息

1. 通过以下任一方式引入依赖：

- Maven 方式引入依赖：

```
<dependency>
  <groupId>com.aliyun.openservices</groupId>
  <artifactId>ons-client</artifactId>
  <version>"XXX"</version>
  // 设置为 Java SDK 的最新版号
</dependency>
```

- 下载依赖 JAR 包：

2. 根据以下说明设置相关参数，运行示例代码：

```
import com.aliyun.openservices.ons.api.Message;
import com.aliyun.openservices.ons.api.Producer;
import com.aliyun.openservices.ons.api.SendResult;
import com.aliyun.openservices.ons.api.ONSFactory;
import com.aliyun.openservices.ons.api.PropertyKeyConst;

import java.util.Properties;

public class ProducerTest {
    public static void main(String[] args) {
        Properties properties = new Properties();
        // 您在控制台创建的 Group ID
        properties.put(PropertyKeyConst.GROUP_ID, "XXX");
        // 鉴权用 AccessKeyId, 在阿里云服务器管理控制台创建
        properties.put(PropertyKeyConst.AccessKey, "XXX");
        // 鉴权用 AccessKeySecret, 在阿里云服务器管理控制台创建
        properties.put(PropertyKeyConst.SecretKey, "XXX");
        // 设置 TCP 接入域名, 进入控制台的实例详情页面, 在页面上方选择实例后, 在实例信息中的“获取
        接入点信息”区域查看
        properties.put(PropertyKeyConst.NAMESRV_ADDR, "XXX");
```

```

Producer producer = ONSFactory.createProducer(properties);
// 在发送消息前，必须调用 start 方法来启动 Producer，只需调用一次即可
producer.start();

// 循环发送消息
while(true){
    Message msg = new Message( //
        // 在控制台创建的 Topic，即该消息所属的 Topic 名称
        "TopicTestMQ",
        // Message Tag,
        // 可理解为 Gmail 中的标签，对消息进行再归类，方便 Consumer 指定过滤条件在消息队列
        // RocketMQ 版服务器过滤
        "TagA",
        // Message Body
        // 任何二进制形式的数据，消息队列 RocketMQ 版不做任何干预，
        // 需要 Producer 与 Consumer 协商好一致的序列化和反序列化方式
        "Hello MQ".getBytes());
    // 设置代表消息的业务关键属性，请尽可能全局唯一，以方便您在无法正常收到消息情况下，可通过控
    // 制台查询消息并补发
    // 注意：不设置也不会影响消息正常收发
    msg.setKey("ORDERID_100");
    // 发送消息，只要不抛异常就是成功
    // 打印 Message ID，以便用于消息发送状态查询
    SendResult sendResult = producer.send(msg);
    System.out.println("Send Message success. Message ID is: " + sendResult.
        getMessageId());
}

// 在应用退出前，可以销毁 Producer 对象
// 注意：如果不销毁也没有问题
producer.shutdown();
}
}

```

查看消息是否发送成功

消息发送后，您可以在控制台查看消息发送状态，步骤如下：

1. 在控制台左侧导航栏，选择消息查询 > 按 Message ID 查询。
2. 在搜索框中输入发送消息后返回的 Message ID，单击搜索查询消息发送状态。

储存时间表示消息队列 RocketMQ 版服务端存储这条消息的时间。如果查询到

此消息，表示消息已经成功发送到服务端。

步骤五：调用 SDK 订阅消息

消息发送成功后，需要启动消费者来订阅消息。下文以调用 TCP Java SDK 为例说明如何订阅消息。

1. 调用 TCP Java SDK 订阅消息。

您可以运行以下示例代码来启动消费者，并测试订阅消息的功能。请按照说明正确设置相关参数。

```
import com.aliyun.openservices.ons.api.Action;
import com.aliyun.openservices.ons.api.ConsumeContext;
import com.aliyun.openservices.ons.api.Consumer;
import com.aliyun.openservices.ons.api.Message;
import com.aliyun.openservices.ons.api.MessageListener;
import com.aliyun.openservices.ons.api.ONSFactory;
import com.aliyun.openservices.ons.api.PropertyKeyConst;

import java.util.Properties;

public class ConsumerTest {
    public static void main(String[] args) {
        Properties properties = new Properties();
        // 您在控制台创建的 Group ID
        properties.put(PropertyKeyConst.GROUP_ID, "XXX");
        // 鉴权用 AccessKeyId, 在阿里云服务器管理控制台创建
        properties.put(PropertyKeyConst.AccessKey, "XXX");
        // 鉴权用 AccessKeySecret, 在阿里云服务器管理控制台创建
        properties.put(PropertyKeyConst.SecretKey, "XXX");
        // 设置 TCP 接入域名, 进入控制台的实例详情页面, 在页面上方选择实例后, 在实例信息中的 "获取接入点信息" 区域查看
        properties.put(PropertyKeyConst.NAMESRV_ADDR, "XXX");

        Consumer consumer = ONSFactory.createConsumer(properties);
        consumer.subscribe("TopicTestMQ", "*", new MessageListener() {
            public Action consume(Message message, ConsumeContext context) {
                System.out.println("Receive: " + message);
                return Action.CommitMessage;
            }
        });
        consumer.start();
    }
}
```

```
System.out.println("Consumer Started");  
}  
}
```

2. 查看消息订阅是否成功。

完成上述步骤后，您可以在控制台查看消费者是否启动成功，即消息订阅是否成功。

- a. 在控制台左侧导航栏，单击 Group 管理。
- b. 找到要查看的 Group ID，单击该 Group ID 所在行操作列的订阅关系。

如果是否在线显示为是，且订阅关系一致，则说明订阅成功。否则说明订阅失败。

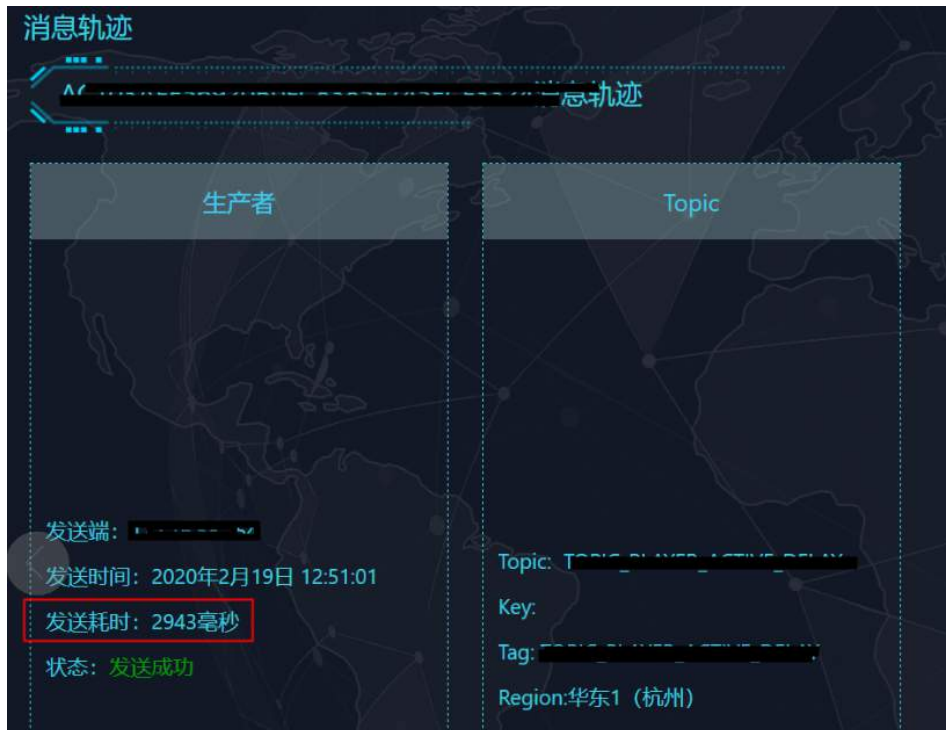
发送问题排查

客户端发送性能问题

发送消息耗时久

【问题描述】：

通过消息轨迹查询到，MQ 消息发送的耗时达到秒级之上。



【排查步骤】:

1. 首先确认该现象是偶发还是频繁出现。
2. 其次确认是部分机器发送有问题还是大量机器发送都存在此现象。
3. 接着检查是少量的 topic 发送有此现象还是大部分机器如此。
4. 确认您的应用机器的外网带宽。
5. 检查发送消息的时间占比。

【问题原因】:

1. 如果是偶发现象, 检查现象存在期间您发送端的 TPS、网络状态以及出口 ip 的连接情况、确认期间应用是否有 Full GC(Full GC 会造成网络延迟), 可结合 ons.log 来综合分析。
2. 如果部分机器有问题, 那么登录该机器, 查看网络流量以及 tps。
3. 若客户端带宽过小, 可建议升级带宽。
4. 如果频繁现象, 建议收集好信息反馈给技术支持人员, 技术支持人员需要查看下后端的 topic 集群是否有问题。

延时消息发送性能问题

【问题描述】:

业务大量使用发送延时消息, 可以指定延时 15 天之后再投递消费吗?

看开源的代码是单线程去处理这些延迟消息的队列, 如果有大量消息在队列中, 性能上也是没有问题的嘛。

【问题回答】:

延时消息我们会先存 DB，在指定的时间扫描 DB，再将消息存储到 broker 上，投递给消费端。并且这个存 DB 和扫码 DB 都是后端完成的。

也是因为这个存储 DB 和扫码的 DB 的损耗，所以我们的定时消息性能是没有普通消息高的。普通消息是直接发送存储到 broker 上的。延时消息就是多了两层的消耗的。

普通消息和延迟消息是不一样的，这二者不存在转换的关系的。

为什么用普通消息的 topic 发送延迟消息也可以成功

【问题描述】:

topic 主题类型分为好几种，但是测试发现无论使用普通消息类型还是延时定时消息类型都一样没问什么区别。

比如创建的普通消息类型 topic 但是发送延时消息也是可以使用的。总结来说 :topic 消息类型有作用吗。

【问题回答】:

往普通消息类型 topic 当中发送延时消息，是可以成功，但是不保证一定成功，也不保证相应的性能。

但是如果您严格按照消息类型创建对应的 topic，并且发送相应类型的消息，性能是会得到保障的。

阿里云生产环境中消息队列 For RocketMQ 除了公网 region 之外，其他 region 不允许在本地使用，必须在对应区域的 ECS 机器上部署使用。

服务端限流 system|broker busy

【问题描述】：

在 ons.log 日志当中出现 system busy, start flow control for a while 或者 broker busy, start flow control for a while 等异常信息。

```
2019-06-28 16:10:45.045 WARN RocketmqClient - execute the pull request exception
com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 2 DESC: (TIMEOUT_CLEAN_QUEUE)broker busy, start flow control for a while; period in queue: 5265ms, size of queue: 26702
For more information, please visit the url: http://rocketmq.apache.org/docs/faq/
    at com.alibaba.rocketmq.client.impl.MQClientAPIImpl.processPullResponse(MQClientAPIImpl.java:971)
    at com.alibaba.rocketmq.client.impl.MQClientAPIImpl.access$700(MQClientAPIImpl.java:192)
    at com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.operationComplete(MQClientAPIImpl.java:923)
    at com.alibaba.rocketmq.remoting.netty.NettyRemotingClient$InvokerCallbackWrapper.operationComplete(NettyRemotingClient.java:735)
    at com.alibaba.rocketmq.remoting.netty.ResponseFuture.executeInvokeCallback(ResponseFuture.java:51)
    at com.alibaba.rocketmq.remoting.netty.NettyRemotingAbstract$2.run(NettyRemotingAbstract.java:112)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471)
    at java.util.concurrent.FutureTask.run(FutureTask.java:262)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
    at java.lang.Thread.run(Thread.java:745)
```

【问题原因】：

1. 共享集群，当时 broker 压力大，会出现这个问题。
2. broker 出现了网络，磁盘，IO 等抖动时，会出现这个问题。

【排查步骤】：

1. 首先看下是短暂偶尔抖动还是持续，持续多长时间？
2. 如果是偶尔抖动，是系统升级或 broker 压力大，或者抖动异常，但是 sdk 有重试策略，会重试到其它的 broker，不影响消息的发送。
3. 如果是长时间持续出现这种情况，那么需要收集一下信息：uid/ 实例 id/ 地域 /topic 等给到技术支持人员，技术人员需要核实下后端集群状态是否正常。
4. 报错建议您在自己的业务代码层面 try...catch 进行重试。

1.8.4 版本的 skd 会自定进行 brokerbusy 的重试【不一定保证重试到其他的 broker 一定是成功的】。

发送端出现发送超时异常

【问题描述】：

在 ons.log 日志中出现 RemotingConnectException: connect to <118.190.213.56:80> failed 或者 RemotingTimeoutException 等异常信息。

```
com.aliyun.openservices.ons.api.exception.ONSClientException: defaultMQProducer send order exception
    at com.aliyun.openservices.ons.api.impl.rocketmq.OrderProducerImpl.send(OrderProducerImpl.java:114)
    at com.taikyun.databus.queue.impl.ons.OnsQueue.offer(OnsQueue.java:82)
    at com.taikyun.databus.node.reader.service.impl.TransportServiceImpl.sendMessage(TransportServiceImpl.java:189)
    at com.taikyun.databus.node.reader.service.impl.TransportServiceImpl.transportData(TransportServiceImpl.java:124)
    at com.taikyun.databus.node.reader.stage.Transport.doRun(Transport.java:108)
    at com.taikyun.databus.node.core.service.AbstractThread.run(AbstractThread.java:42)
Caused by: com.aliyun.openservices.shade.com.alibaba.rocketmq.remoting.exception.RemotingConnectException: connect to <118.190.213.56:80> failed
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.remoting.netty.NettyRemotingClient.invokeSync(NettyRemotingClient.java:360)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.sendMessageSync(MQClientAPIImpl.java:267)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.sendMessage(MQClientAPIImpl.java:251)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.sendMessage(MQClientAPIImpl.java:214)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.producer.DefaultMQProducerImpl.sendKernelImpl(DefaultMQProducerImpl.java:671)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.producer.DefaultMQProducerImpl.sendSelectImpl(DefaultMQProducerImpl.java:882)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.producer.DefaultMQProducerImpl.send(DefaultMQProducerImpl.java:859)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.producer.DefaultMQProducerImpl.send(DefaultMQProducerImpl.java:854)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.producer.DefaultMQProducer.send(DefaultMQProducer.java:156)
    at com.aliyun.openservices.ons.api.impl.rocketmq.OrderProducerImpl.send(OrderProducerImpl.java:96)
    ... 5 more
```

【排查步骤】：

1. 首先可以确认该时间段内是否属于服务升级时间段内（可以看官网公告）这是 MQ 服务升级过程中，会出现短暂的网络闪断，但是我们的 mq 服务是集群部署的，一台网络的闪断是不会影响消息的。
2. 在自己的应用服务器上执行 telnet brokerip port，确认服务端的端口是否通畅。
3. 执行 ping brokerip，查看网络是否延迟。同时检查网络监控指标，观察在问题时间点流量是否有下降的情况。如果 ping 或者 telnet 不通，需要检查下服务器的防火墙，网络设置等。

4. 检查应用的网络带宽情况，是否打满。
5. 可执行 `jstack -l 进程号 > 文件名.dump` 来分析堆栈信息，判断当时应用系统有没有 Full GC 现象 (Full GC 会造成一定的网络延迟)。
6. 确认下使用的 sdk，如果是较低的 javasdk 版本，建议升级 sdk 版本到 1.8.4。这个版本里容灾策略较之前版本优化了许多内容。

同时建议客户端做一下补偿机制，可以 `try...catch` 一下异常，做下消息的重试。

启动发送端连接异常

【问题描述】：

启动消费端的时候，报错连接，异常如下：

```
com.aliyun.openservices.ons.api.exception.ONSClientException: Can not find name server with
onsAddr http://****.mqrest.cn-hangzhou.aliyuncs.com
```

See http://docs.aliyun.com/cn#/pub/ons/faq/exceptions&namesrv_not_exist for further details.

```
at com.aliyun.openservices.ons.api.impl.rocketmq.ONSClientAbstract.fetchNameServerAd-
dr(ONSClientAbstract.java:131)
```

```
at com.aliyun.openservices.ons.api.impl.rocketmq.ONSClientAbstract.<init>(ONSClientAbstract.
java:84)
```

```
at com.aliyun.openservices.ons.api.impl.rocketmq.ProducerImpl.<init>(ProducerImpl.java:35)
```

```
at com.aliyun.openservices.ons.api.impl.ONSTransactionImpl.createProducer(ONSTransactionImpl.
java:30)
```

```
at com.aliyun.openservices.ons.api.ONSTransactionFactory.createProducer(ONSTransactionFactory.
java:89)
```

【排查方向】：

看报错的异常是和接入点有关的，报错是接入点不存在，需要和核实一下代码当中填入的接入点信息。

【问题原因】：

代码当中使用的是杭州地域的 http 协议的公网接入点。

但是使用的是 tcp 协议的 sdk，代码配置却是 http 的接入点。

开源 python 发送端发送报错

【问题描述】：

使用开源 python 的 tcp 官方提供的 demo，发送报错。

```
Traceback (most recent call last):
  File "rocketmq_producer.py", line 32, in <module>
    send_message_sync(10)
  File "rocketmq_producer.py", line 24, in send_message_sync
    ret = producer.send_sync(msg)
  File "/home/di/anaconda3/lib/python3.7/site-packages/rocketmq/client.py", line 210, in send_sync
    ffi_check(dll.SendMessageSync(self._handle, msg, ctypes.pointer(c_result)))
  File "/home/di/anaconda3/lib/python3.7/site-packages/rocketmq/exceptions.py", line 44, in ffi_check
    raise exc_cls(msg)
```

【排查步骤】：

到 {home}/logs/rocketmqlogs 路径下查看 mq 的日志。

从日志当中看到有连接报错。

```
2020-02-18 10:13:46,046 WARN RocketmqClient - updateTopicRouteInfoFromNameServer Exception
com.alibaba.rocketmq.remoting.exception.RemotingConnectException: connect to <null> failed
    at com.alibaba.rocketmq.remoting.netty.NettyRemotingClient.invokeSync(NettyRemotingClient.java:424)
    at com.alibaba.rocketmq.client.impl.MQClientAPIImpl.getTopicRouteInfoFromNameServer(MQClientAPIImpl.java:1660)
    at com.alibaba.rocketmq.client.impl.MQClientAPIImpl.getTopicRouteInfoFromNameServer(MQClientAPIImpl.java:1611)
    at com.alibaba.rocketmq.client.impl.factory.MQClientInstance.updateTopicRouteInfoFromNameServer(MQClientInstance.java:643)
    at com.alibaba.rocketmq.client.impl.factory.MQClientInstance.updateTopicRouteInfoFromNameServer(MQClientInstance.java:530)
    at com.alibaba.rocketmq.client.impl.factory.MQClientInstance.findConsumerIdList(MQClientInstance.java:1128)
    at com.alibaba.rocketmq.client.impl.consumer.RebalanceImpl.rebalanceByTopic(RebalanceImpl.java:262)
    at com.alibaba.rocketmq.client.impl.consumer.RebalanceImpl.doRebalance(RebalanceImpl.java:225)
    at com.alibaba.rocketmq.client.impl.consumer.DefaultMQPushConsumerImpl.doRebalance(DefaultMQPushConsumerImpl.java:1016)
    at com.alibaba.rocketmq.client.impl.factory.MQClientInstance.doRebalance(MQClientInstance.java:1031)
    at com.alibaba.rocketmq.client.impl.consumer.RebalanceService.run(RebalanceService.java:41)
    at java.lang.Thread.run(Thread.java:748)
    at com.oracle.svm.core.thread.JavaThreads.threadStartRoutine(JavaThreads.java:473)
    at com.oracle.svm.core.posix.thread.PosixJavaThreads.pthreadStartRoutine(PosixJavaThreads.java:193)
```

根据报错怀疑是接入点接入点接入的有问题

【问题原因】：

代码当中填入的是 http 的接入点，但是使用的是 tcp 的 sdk，应该使用控制台提供的 tcp 的接入点。

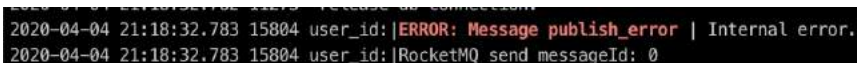
需要注意的是，使用什么连接类型的 sdk，就一定要对应使用控制台提供的对应连接类型的接入点，如果使用错误，那么代码将不会运行成功。

Php 的 http 接入方式发送消息失败

【问题描述】：

使用 php 的 http 接入方式，发送消息报错，导致消息发送失败。

报错如图。



```
2020-04-04 21:18:32.783 15804 user_id: |ERROR: Message publish_error | Internal error.  
2020-04-04 21:18:32.783 15804 user_id: |RocketMQ send messageId: 0
```

【问题回答】：

internal error 一般是后端记录有类似于网络抖动的报错异常。

遇到此类错误，需要将实例 id/ 地域 /topic/sdk 语言版本等信息收集反馈给到技术人员进行后端服务器核实。

发送消息报错 NumberFormatException: For input string: "//XXX"

【问题描述】：

启动发送端，发送消息报错 NumberFormatException: For input string: "//XXX"。

发送消息时日志报错

【问题描述】：

发送消息的时候，日志记录有报错

com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException:
CODE: 1 DESC: [REJECTREQUEST]Broker in slave mode

```
message: 2020-05-27 09:49:47,047 WARN RocketmqClient - execute the pull request exception com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 1 DESC: [REJECTREQUEST]Broker in slave mode. For more information, please visit the url, http://rocketmq.apache.org/docs/faq/ at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.processPullResponse(MQClientAPIImpl.java:971) at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.access$5700(MQClientAPIImpl.java:192) at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.operationComplete() at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.run() at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) at java.lang.Thread.run(Thread.java:748)
message: 2020-05-27 09:49:47,047 WARN RocketmqClient - execute the pull request exception com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 1 DESC: [REJECTREQUEST]Broker in slave mode. For more information, please visit the url, http://rocketmq.apache.org/docs/faq/ at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.processPullResponse(MQClientAPIImpl.java:971) at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.access$5700(MQClientAPIImpl.java:192) at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.operationComplete() at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.run() at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) at java.lang.Thread.run(Thread.java:748)
message: 2020-05-27 09:49:47,047 WARN RocketmqClient - execute the pull request exception com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 1 DESC: [REJECTREQUEST]Broker in slave mode. For more information, please visit the url, http://rocketmq.apache.org/docs/faq/ at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.processPullResponse(MQClientAPIImpl.java:971) at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.access$5700(MQClientAPIImpl.java:192) at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.operationComplete() at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.run() at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) at java.lang.Thread.run(Thread.java:748)
message: 2020-05-27 09:49:47,047 WARN RocketmqClient - execute the pull request exception com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 1 DESC: [REJECTREQUEST]Broker in slave mode. For more information, please visit the url, http://rocketmq.apache.org/docs/faq/ at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.processPullResponse(MQClientAPIImpl.java:971) at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.access$5700(MQClientAPIImpl.java:192) at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.operationComplete() at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl$8.run() at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) at java.lang.Thread.run(Thread.java:748)
```

【问题原因】：

这个报错是因为优化后端集群节点的时候导致的报错。对使用是没有影响的哈，只是日志会有异常记录下来。

也可以再观察下，如果是持续大量报错，并且消息会发送失败，需要提供实例id/ 地域 /topic 等信息给到技术人员进行查看。

消息体太大导致发送失败

【问题描述】：

发送消息的时候报错：

CODE: 13 the message is illegal, maybe msg body or properties length not matched. msg body

以上，3.5.3 版本及其以上版本升级了 on-client 插件的版本。

运行时环境升降级

发送时报错 fetch name server address exception

【问题描述】：

发送消息的时候日志出现 fetch name server address exception 异常信息。

```
2019-09-06 12:46:38,038 ERROR RocketmqCommon - fetch name server address exception
java.net.SocketException: Unexpected end of file from server
    at sun.net.www.http.HttpClient.parseHTTPHeader(HttpClient.java:851)
    at sun.net.www.http.HttpClient.parseHTTP(HttpClient.java:678)
    at sun.net.www.http.HttpClient.parseHTTPHeader(HttpClient.java:848)
    at sun.net.www.http.HttpClient.parseHTTP(HttpClient.java:678)
    at sun.net.www.protocol.http.HttpURLConnection.getInputStream0(HttpURLConnection.java:1587)
    at sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1492)
    at java.net.HttpURLConnection.getResponseCode(HttpURLConnection.java:480)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.common.utils.HttpTinyClient.httpGet(HttpTinyClient.java:46)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.common.namesrv.TopAddressing.fetchNSAddr(TopAddressing.java:73)
    at com.aliyun.openservices.shade.com.alibaba.rocketmq.common.namesrv.TopAddressing.fetchNSAddr(TopAddressing.java:64)
    at com.aliyun.openservices.ons.api.impl.rocketmq.ONSClientAbstract.fetchNameServerAddr(ONSClientAbstract.java:126)
    at com.aliyun.openservices.ons.api.impl.rocketmq.ONSClientAbstract.<init>(ONSClientAbstract.java:84)
    at com.aliyun.openservices.ons.api.impl.rocketmq.ONSConsumerAbstract.<init>(ONSConsumerAbstract.java:33)
    at com.aliyun.openservices.ons.api.impl.rocketmq.ConsumerImpl.<init>(ConsumerImpl.java:32)
    at com.aliyun.openservices.ons.api.impl.ONSFactoryImpl.createConsumer(ONSFactoryImpl.java:36)
    at com.aliyun.openservices.ons.api.ONSFactory.createConsumer(ONSFactory.java:167)
    at com.aliyun.openservices.ons.api.ONSFactory.createConsumer(ONSFactory.java:167)
```

【问题原因】：

根据代码配置中的 ONSAddr 这个地址来获取 NAMESRV_ADDR 时失败了。

【排查步骤】：

1. 检查代码配置，看配置的 ONSAddr 是否和控制台上的一致，有可能配置错了，将 NAMESRV_ADDR 当成了 ONSAddr，进行了如下图配置。

```
producerProperties.setProperty(PropertyKeyConst.ONSAddr, NAMESRV_ADDR);
```

2. nslookup 接入点的域名看下。检查 dns 是否正确。正确情况下，会获取到绑定。

这个域名的 ip，在 ping 下 ip 是否是通的

nslookup onsaddr-internet.aliyun.com

发送消息报错 not set any response code

【问题描述】：

发送延迟消息报错。

```
com.aliyun.openservices.ons.api.exception.ONSClientException: defaultMQProducer send exception
at com.aliyun.openservices.ons.api.impl.rocketmq.ProducerImpl.checkProducerException(ProducerImpl.java:242)
at com.aliyun.openservices.ons.api.impl.rocketmq.ProducerImpl.send(ProducerImpl.java:136)
at com.aliyun.openservices.ons.api.bean.ProducerBean.send(ProducerBean.java:58)
at com.baijia.uqun.individual.management.biz.rocket.listener.AutoAcceptFriendListener.sendLogLoopDelay(AutoAcceptFriendListener.java:162)
at com.baijia.uqun.individual.management.biz.rocket.listener.AutoAcceptFriendListener.consume(AutoAcceptFriendListener.java:121)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.consumer.ConsumerImpl.consumeMessage(ConsumerImpl.java:181)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.consumer.ConsumerImpl.consumeMessageConcurrently(ConsumerImpl.java:181)
at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at java.lang.Thread.run(Thread.java:745)
Caused by: com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 4 DESC: not set any response code
For more information, please visit the url, http://rocketmq.apache.org/docs/faq/
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.processSendResponse(MQClientAPIImpl.java:630)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.sendMessageSync(MQClientAPIImpl.java:421)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.sendMessage(MQClientAPIImpl.java:483)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl.sendMessage(MQClientAPIImpl.java:365)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.producer.DefaultMQProducerImpl.sendKernelImpl(DefaultMQProducerImpl.java:752)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.producer.DefaultMQProducerImpl.send(DefaultMQProducerImpl.java:484)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.producer.DefaultMQProducerImpl.send(DefaultMQProducerImpl.java:1137)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.producer.DefaultMQProducerImpl.send(DefaultMQProducerImpl.java:1091)
at com.aliyun.openservices.shade.com.alibaba.rocketmq.client.producer.DefaultMQProducer.send(DefaultMQProducer.java:257)
at com.aliyun.openservices.ons.api.impl.rocketmq.ProducerImpl.send(ProducerImpl.java:127)
16/0000
```

【问题原因】：

设置的 message properties 中某个属性的值过长，造成发送失败。

【排查过程】：

1. 查询问题节点的消息，mq 控制台对应的接口是 ConsoleMessageGetByPagedTopic，openApi 对应的接口是 OnsMessagePageQueryByTopic。
2. 登录 mq 控制台，打开浏览器开发工具（win os: F12 mac os: command+F12），找到如下图所示的接口，查看返回的 response。将其进行 json 化，查看 properties 里是否存在长度比较长的属性值。

息出错了，可以重新换个 ak 试试。

启动发送者时报错 UnknownHostException

【问题描述】：

启动消息队列 RocketMQ 版的客户端时提示异常 UnknownHostException 信息。

【问题原因】：

导致此问题的主要原因是客户端无法获取系统的主机名 (Hostname) 或者系统的 IP 地址。

【解决方案】：

请参考以下步骤进行排查：

1. 登录客户端所在机器。

2. 执行 hostname 命令，检查能否正常返回主机名。

(1) 如果该命令报错，请检查是否为该命令定义了别名 (alias)，比如在 .bash_profile 文件或者 .bashrc 文件中设置了 alias hostname='/usr/bin/*****' 的别名。确保 hostname 命令能够正常返回主机名。

(2) 如果该命令正常执行，记录返回的主机名并继续下一步。

3. 检查能否 ping 通记录的主机名。

(1) 如果无法 ping 通，请参考 127.0.0.1[\$Hostname]，将记录的主机名绑定到 /etc/hosts 文件中。

(2) 如果可以 ping 通，请继续下一步。

4. 检查 `/etc/sysconfig/network` 文件中的 `Hostname` 是否与 `/etc/hosts` 文件中的主机名一致。
 - (1) 如果不一致, 请修改 `/etc/sysconfig/network` 文件中的 `Hostname` 参数值, 使其与 `/etc/hosts` 文件中的主机名一致。
 - (2) `/etc/sysconfig/network` 文件中不存在 `Hostname` 配置, 请参考 `hostnamectl set-hostname [Hostname]` 命令更新主机名。
 - (3) 如果一致, 请继续下一步。
5. 重新启动消息队列 RocketMQ 版的客户端, 确认不再提示有关未知主机名的异常信息。

发送消息时报错消息不合法

【问题描述】:

发送消息时报错消息不合法。

【问题原因】:

一般是消息属性、消息内容不合法, 不合法的情况有:

1. 消息为空;
2. 消息内容为空;
3. 消息内容长度为 0;
4. 消息内容超过限定长度。

Tcp 协议开源 sdk 连接 mq 失败

【问题描述】：Tcp 协议开源 sdk 无法连接 mq，代码层面报错。

【问题回答】：

可能原因：

开源 sdk 接入阿里云，阿里云都是与各个开源语言的 sdk 做过适配的。因此一定需要使用阿里云官方文档上推荐的各个语言的 sdk 版本，如果使用的开源版本非阿里云提供的官网推荐使用的 sdk 版本，那么是无法成功连接到阿里云 mq 的。

消费问题排查

堆积负载常见问题

Java 进程消息堆积严重

【问题描述】：

消费者状态页面，Group ID 的实时消息堆积量的值高于预期，且性能明显下降。

【排查步骤】：

1. 在消息队列 MQ 控制台，通过查看消费者状态获取消息堆积的消费者实例所对应的宿主机 IP，并登录该宿主机或容器。
2. 执行以下任一命令查看进程 pid：

```
- ps -ef |grep java  
- jps -lm
```

3. 执行以下命令查看堆栈信息：

```
- jstack -l pid > /tmp/pid.jstack
```

4. 执行以下命令查看 ConsumeMessageThread 的信息，重点关注线程的状态及堆栈：

```
cat /tmp/pid.jstack|grep ConsumeMessageThread -A 10 -color
```

只需要注意两种状态就可以。

- BLOCKED 此状态说明消费者线程被阻塞了，导致消息的消费被停滞了，从而导致消息堆积的产生。这时要通过上面的堆栈信息来查看具体是阻塞在那个接口。
- WAITING 这个状态要分两种情况来说明：

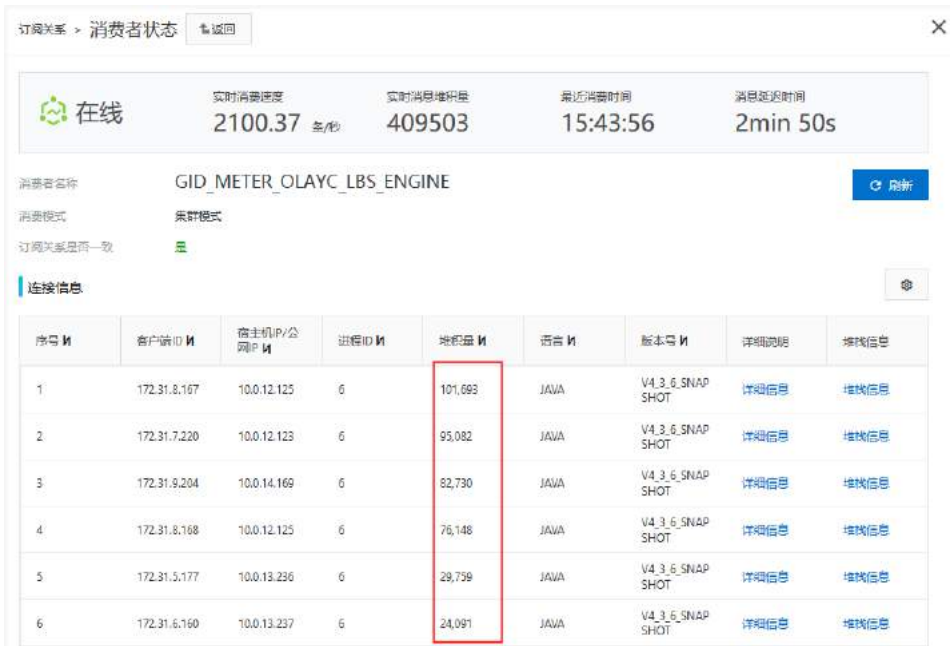
```
ConsumeMessageThread_23
TID: 592 STATE: WAITING
sun.misc.Unsafe.park(Native Method)
java.util.concurrent.locks.LockSupport.park(Unknown Source)
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(
Unknown Source)
java.util.concurrent.LinkedBlockingQueue.take(Unknown Source)
java.util.concurrent.ThreadPoolExecutor.getTask(Unknown Source)
java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
java.lang.Thread.run(Unknown Source)
```

- (1) 如果是上图所显示的堆栈信息，说明消费者线程在等待消息消费，此为正常状态。
- (2) 如果不是上图所显示的堆栈信息，可以重复上面的 3、4 两步，检查这个这个消费者线程是否一直处于这种状态，而且堆栈信息显示也是一样的。这个现象说明在消费逻辑的代码中，由于某种资源紧张，导致获取这种资源的时间较长，从而导致消息消费的耗时增长，TPS 下降，消息消费的速度跟不上消息生产的速度，从而导致消息堆积。

为什么消费者堆积负载不均

【问题描述】：

每个消费者堆积数量差异很大，负载非常不均，具体现象如图。



【问题原因】：

这个是分配策略，当所有消费实例都达到消费能力上限后，不再完全分配到所有都队列，会集中分配到前面的队列。这种现象一般建议提升代码消费能力和增加消费实例数解决堆积问题后就不会出现这种堆积。

可以理解为消费能力已经饱和，其他队列本身也有堆积情况，那么饱和之后分配就会直接到指定的某个队列。这种情况需要增加消费者或者优化代码来提升能力减少堆积，堆积多肯定会更加满负荷在跑。

这个上限不是挤压的或者 mq 端大量堆积导致程序高负载运行的上限，是正常消费有堆积就认为达到上限，是一个正常消费的理论上限。

这个是产品侧的一个策略，无法做到堆积后再均衡对堆积量做进一步对均衡分配。建议增加消费实例或者优化代码，减少堆积压力。

广播模式消费堆积

【问题描述】：

广播模式下控制台显示有大量堆积。

【问题回答】：

广播模式下控制台看到的这个堆积数据是无效的，广播模式下服务端不维护消费进度，消费进度在客户端维护，所以是否有堆积这个需要您客户端来维护，广播模式下我们服务端不维护消费进度，您的消费端的消费位点，我们这边不保存，这个只能您自己消费端配置好消费日志，查看消费情况的。

实时消息堆积量和各个具体消费端堆积量不一致

【问题描述】：

消费者的实时消息堆积量有堆积量，但是各个具体的消费端却显示没有堆积量。

消费者状态

在线	实时消费速度 0.08 条/秒	实时消息堆积量 13	最近消费时间 09:13:37	消息延迟时间 1h 1min 16s
----	--------------------	---------------	--------------------	-----------------------

Group ID: [redacted]
消费模式: 集群模式
订阅关系是否一致: 是

连接信息

序号	客户端ID	虚拟机IP/公网IP	进程ID	堆积量	语言	版本号	详细说明	堆栈信息
1	[redacted]	[redacted]	12289	0	JAVA	V4_3_9	详细信息	堆栈信息
2	[redacted]	[redacted]	9850	0	JAVA	V4_3_9	详细信息	堆栈信息

查看详细信息也显示没有积压

【问题原因】：

实时消息堆积量是订阅的整个 topic 的堆积量，包括重试队列的。

如果说这个 gid 曾经订阅过 topicA，现在不订阅 A 了，换成订阅 TopicB，那么这个堆积量也是包含曾经订阅 A 的堆积量。

下面各个 c 端的堆积量，是精确到这个 c 端订阅的 tag 的堆积量。下面的 c 端堆积量之和应该小于等于实施堆积量。

【建议方案】：

可以点击这个消费者的【重置消费位点】，看下可以选择到的 topic 是不是有目前没有在订阅的。如果有的话，可以把这个 topic 的位点清除下，那么这个消费者的关于这个不再订阅的 topic 的实时堆积量就会清除掉了。

mq 是否支持积压消息后批量消费

【问题描述】：

开源 RocketMQ 里有 interval 积压时间，可以积压几秒钟再进行批量消费。

无序消费消息时，是否有参数可以设置 Pull 的实际间隔，从而保障拉取下来的消息有一定的量，而不是 1 个，主要目的是想提升批量。

有序分区消息时，是否支持批量的有序消费？

【问题回答】：

RocketMQ 现在的消费模式本身就是 push 的。

1. 批量消费可在消费客户端中设置 ConsumeMessageBatchMaxSize 参数，批量消费消息个数。

批量拉取也要有对应的消息数才行，这个是在预期范围内的。这个是根据当时队列有多少条消息未消费来决定的，并不能做到每次消费的消息是一个固定值。PropertyKeyConst.ConsumeMessageBatchMaxSize 取值范围为 1-32。

mq 的消费逻辑是这样的，启动消费者之后，消费者会一直在线。

如果 topic 当中一直没有消息过来，我们会每 15 秒长轮询一次，消费者主动到 topic 当中拉取消息。如果有消息，就会将消息拉取到进行消费。所以暂时不支持挤压一定的消息之后再批量消费的。

2. 有序消息不支持批量消费。

消费负载不均衡，有台消费者实例没有消费消息

【问题描述】：

一个 gid 下面，多个消费端，有一 / 几台机器显示堆积量为 0，其他的机器堆积量都很大。

消费者状态								
消费名称		0 条/秒	843	14:58:16	1h 57min 46s			
消费者名称						刷新		
消费模式		集群模式						
订阅关系是否一致		是						
连接信息								
序号	客户端ID	主机IP/公网IP	进程ID	堆积量	语言	版本号	详细说明	堆栈信息
1		1	8	843	JAVA	V4_3_6_SNAPSHOT	详细信息	堆栈信息
2	1	7	8	0	JAVA	V4_3_6_SNAPSHOT	详细信息	堆栈信息

【排查步骤】：

可以查看下堆积量为 0 的机器的 ons.log 日志。

通过关键字搜索日志信息（注意要找到对应的 topic）。

只用关注其中的 perm 的值就行，6 表示正常，如果是 4 表示则表示 mq 后端做了相关的运维操作，后面会恢复正常的，此时没有消费的客户端可以拿来容灾。

分区顺序消息消费负载不均

【问题描述】：

三台同样的机器，一台收到 95% 的消息，一台 5% 左右的消息，一台一直是 0。

个 pod 的代码部署的是一样的就是一套代码，都去作为消费者。

三台机器的属性是一模一样，都是同一个业务。

【问题回答】：

可能原因：

发送分区顺序消息的时候，指定的 shardingKey 不太合适。导致消息发送打散的不够均匀，基本上都集中少量逻辑分区里面了。

shardingKey 会影响消息分布在不同队列，然后消费机器是对应到多个队列的。

发送分区顺序消息指定 sharding Key, 满足业务约束的前提下，尽量引进一些变量，分布就会均匀了，不是随机。

举例来说：如果是订单系统，用性别作为 sharding key, 就不太合适。如果 user id/ 或者订单号，就是更好一些的选择。

为什么消费堆积显示为负数？

【问题描述】：

为什么消费堆积显示的是负数。

【排查步骤】：

1. 是否有删除过这个 topic，如果之前已经使用了这个 topic 进行收发消息，后面又删除了，然后短时间内又创建了这个 topic，就会导致这个问题。可以

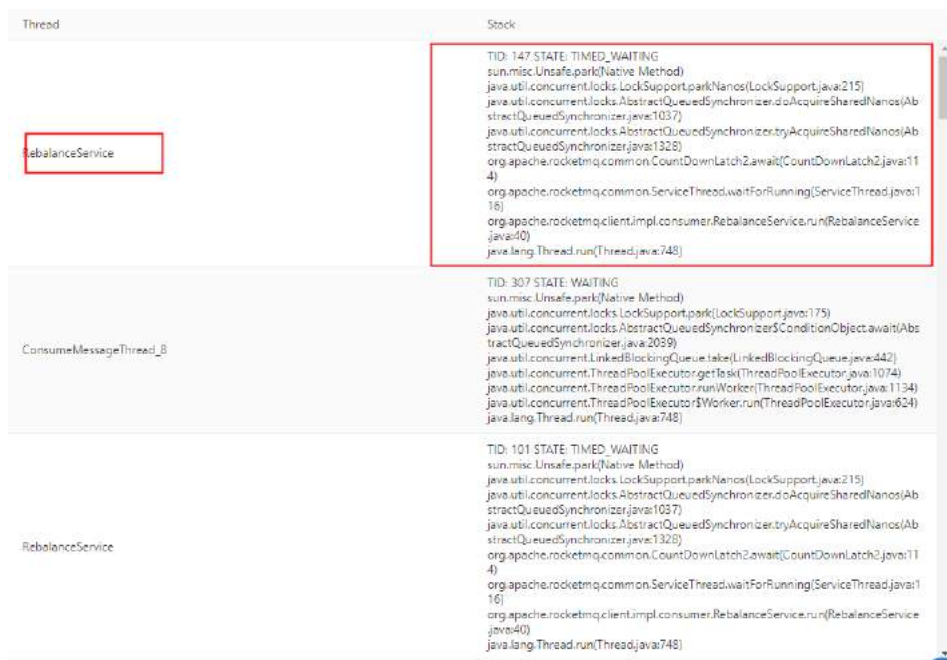
通过重置消费位点来解决这个问题。

2. 如果没有，那么需要提供实例 id/ 地域 /topic 等信息给到技术人员进行进一步的查看。

消费者 rebalanceservice 线程

【问题描述】：

在消费者线程当中看到 rebalanceservice 线程是正常的吗？



【问题回答】：

这个是消费端负载均衡的线程。是一个正常状态的线程。rebalance 是有时间间隔的。

http 协议消费者实时堆积量

【问题描述】：

http 协议下的消费者实时堆积量都包含了哪些？

【问题回答】：

这个堆积量是表示该消费者订阅的 topic 当中的所有 tag 的总堆积量。

举例：topic 当中有 5 个 tag，该消费者指订阅了其中 1 个，还有 4 个没订阅，但是这个四个没订阅的 tag，在 topic 当中有堆积了，这个堆积量也会显示在控制台当中的。

消息投递重试问题

如果消息业务消费时间太久，那过多久会重新投递该消息

【问题描述】：

在订阅消费消息时，假如消费这条消息需要 10 分钟，那么 MQ 大概多久会重新投递该消息。

【问题回答】：

消费端到 mq 当中拉到消息进行消费之后，会反馈给 mq 已经成功拿到消息并且消费。如果 mq 没有接收到消费端的反馈，那么 mq 就会认为这个消息没有消费成功，那么这个消息在 topic 当中就会再次可见，那么消费者就可以再次来拉取这个消息（重试）。

第一次发消息算是第 0 次，如果在 10 秒内没有给 mq 反馈，那么这个消息就会再次可见，消费端就会再去拉取这个消息，就是第一次重试了，如果在 10 秒之后，消费者给了第一次发消息（第 0 次）的响应，那么这个消费也是失败的。比如消费者现在拉取到的是 10s 之后第一次重试的，那么在之后的 30 秒之内，如果消费者给了 mq 第一次重试拉到的消息的响应，这个才算是消费成功的。

所以建议不要把业务逻辑放在返回给 mq 消息的代码之前，最好是保证 10 秒之内要给到 mq 响应。可以将业务逻辑放在其他地方做并发处理。

```

Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("testTpoic-lxm", "TagA", new MessageListener() { //订阅多个Tag
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return Action.CommitMessage;
    }
});

```

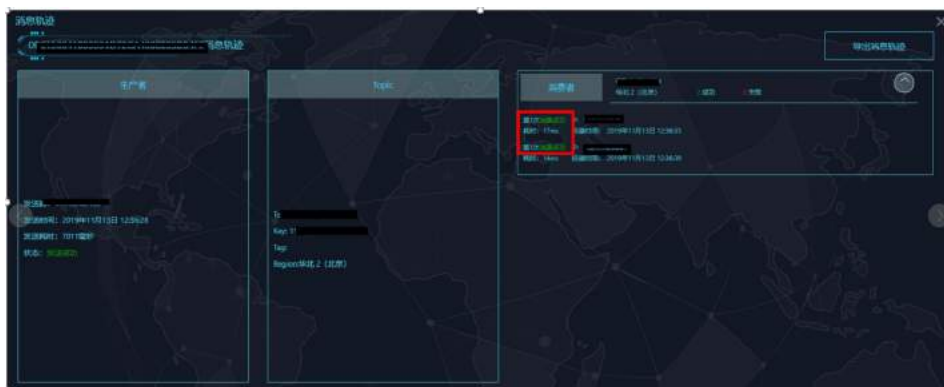
如果这边添加了这种业务逻辑，那么就会导致给到mq响应的时间过久，就会导致消息重试

给到mq响应

消息重复投递

【问题描述】：

同一个消息有多次成功消费记录。



【排查步骤】：

1. 如果只是偶尔有消息这样，是因为绝大多数情况下，消息是不重复的。作为一款分布式消息中间件，在网络抖动、应用处理超时等异常情况下，无法保证消息不重复，但是能保证消息不丢失。

这种情况下，需要做好消费幂等，具体可以参考文档：

https://help.aliyun.com/document_detail/44397.html?spm=5176.11065259.1996646101.searchclickresult.4a8f70e3uKZijr

2. 如果是大量消息重复消费，需要核实您的代码是否和 mq 官方提供的一致，在重复消费消息的时候，消费端的网络情况，是否有 fullgc 等。

如果以上都没有问题，则需要提供下具体的异常信息：实例 id/ 地域 /topic/ msgid 等，给到技术支持人员进行确认。

消息消费失败，一直重试

【问题描述】：

消息消费失败，一直消息重试，并且重试的间隔时间都是正常的。



【排查步骤】：

1. 需要查看下业务上关于这个消息有没有什么报错，一般消费失败是业务处理消息失败，业务如果有问题，重试多少次还是会有问题的。
2. 可以查看下 ons.log 日志，看下有没有异常。

3. 需要检查下消费的逻辑，返回的状态需要是 `Action.CommitMessage` 才对的。

广播模式消费失败重试

【问题描述】：

有 A,B,C 3 个节点，使用广播模式发送一条消息，实际的目标是 让 B 处理，A 和 C 直接放弃，那么问题是 A 和 C 直接 `Action.CommitMessage`。

掉这条消息后，B 里面业务处理出了问题 我消息 `Action.ReconsumeLater` 后消息是否会重发，重发是发给 B 还是 A,B,C 都会收到。

【问题回答】：

广播模式不支持消费失败重投。

消费失败的消息没有被重试

【问题描述】：

消费失败的消息，一直没有进行重试或者重试间隔的时间不对，与官方文档提供的重试时间不符合。

【排查步骤】：

1. 检查下否是广播消费模式，因为广播方式不提供失败重试特性，即消费失败后，失败消息不再重试，继续消费新的消息。
2. 如果确认使用的是集群消费模式，检查消费者启动时的配置，是否有将最大重试次数 `MaxReconsumeTimes` 的值设置为 0，设置为 0 后将不在重试。这时可以去死信队列里确认下是否有此条消息。

3. 如果配置没问题，可以过会再去检查消息轨迹，因为在共享集群下，某个时间段 broker 压力比较大的时候，消费失败重试会有一定的延迟。可以多检查下，确认没有重试。
4. 如果以上都确认没有问题了，需要提供实例 id/ 地域 /topic/msgid 等信息给技术人员进行查看。

订阅关系问题

订阅关系不一致

【问题描述】：

为什么消费端不消费了 / 为什么我的消息堆积了许多 / 为什么消费速度不及预期，只是偶尔消费一两条 / 为什么一条消息被重复消费了？

【问题原因】：

消费者的订阅关系不一致了。

订阅关系一致指的是同一个消费者 Group ID 下所有 Consumer 实例的处理逻辑必须完全一致。一旦订阅关系不一致，消息消费的逻辑就会混乱，甚至导致消息丢失。

由于消息队列 RocketMQ 版的订阅关系主要由 Topic + Tag 共同组成，因此，保持订阅关系一致意味着同一个消费者 Group ID 下所有的消费者实例需在以下两方面均保持一致：

- 订阅的 Topic 必须一致
- 订阅的 Topic 中的 Tag 必须一致

消费者状态

 **在线**

实时消费速度
0 条/秒

实时消息堆积量
3

消费者名称
GID_gateway

消费模式
集群模式

订阅关系是否一致
否

http 的消费者看不到订阅关系

【问题描述】：

http 的 gid 正在消费，但是 topic 当中查看不到这个订阅关系。



【问题回答】：

http 协议的消费者的订阅关系控制台目前不支持显示的。

客户端消费常见异常报错

启动消费端时报错

【问题描述】：

启动消费端时报错找不到 gid。

2019-11-18 20:29:04,004 WARN RocketmqClient - execute the pull request exception

com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 26 DESC: subscription group [MQ_INST_****_BaSIIRW%GID_****] does not exist,

See <http://rocketmq.apache.org/docs/faq/> for further details.

【问题原因】：

后台创建 GID 失败了。

【解决方案】：

可以在控制台当中将已经创建好的 gid 删除，在重新创建，创建完成之后，再次启动消费端。

消费线程被卡住

【问题描述】：

查看消费者状态—连接详情，看到消费者存在消费线程被卡组和现象，卡住时间已超过 10min 25s。

连接状态

消费者名称

消费类型

PUSH

消费线程数

20

消费启动时间

2020年2月19日 23:28:31

异常信息

存在消费线程被卡住现象,卡住时间已超过10min 25s.

订阅关系

【问题回答】:

线程阻塞一般是 CPU 或内存消耗过高导致，一般是业务代码处理复杂和线程数较多导致。

1. 建议重启一下消费端看下堆积是否会降下去。
2. 建议看看能否升级一下机器配置。
3. 优化一下接收到消息后的业务处理代码。
4. 减少消费并发线程数目。

消费者启动后不消费

【问题描述】：

一个全新的消费者 `qid` 启动，但是不消费已经发送到 `topic` 当中的消息。

【问题原因】：

一个 topic 如果从来没有任何一个 gid 连接消费过，假设这个 topic 当中已经有一千万条消息发进来了，第一个启动的 gid 是不会去消费这一千万条消息的。是会从

这个 gid 启动当下开始，发送到 topic 当中的消息开始消费。

如果说这个 topic 已经有 gid 连接过了，如果再次启动一个全新的 gid 连接，是会去消费 topic 当中还存在的，tag 匹配的消息的。

消费的时候，如何获取消息体内容

【问题描述】：

消费的时候，如何获取消息体内容。

【问题回答】：

以 java 代码为例，可以使用 `new String(message.getBody())` 来获取消息体内容。

消费消息时 ack 报错

【问题描述】：

消费消息时候 ack 报错，ErrorMessage: The receipt handle you provided has expired.code MessageNotExist。

【问题原因】：

报错是消息句柄过期了，这个是访问超时了。

【解决方案】：

重新发送下这条消息，然后重新消费下。

全局顺序消息为什么启动多个消费者只有一个在消费

【问题描述】：

全局顺序消息的 topic 下，一个 gid 启动了多个消费者，但是只有一个消费者在消费，其他的消费者都不消费？

【问题回答】：

全局顺序创建 topic 时会指定到固定一台 broker，只会分配一个队列。也就是一台消费者能消费到消息。

全局顺序消息，不管启动几个消费端，实际上都只会会有一个消费端来消费，也会分配到一个 broker 一个 queue。

此时多启动的消费者可以用来容灾。

消费失败的消息可以直接丢弃掉吗？

【问题描述】：

消费失败的消息可以手动丢弃掉吗。

【问题回答】：

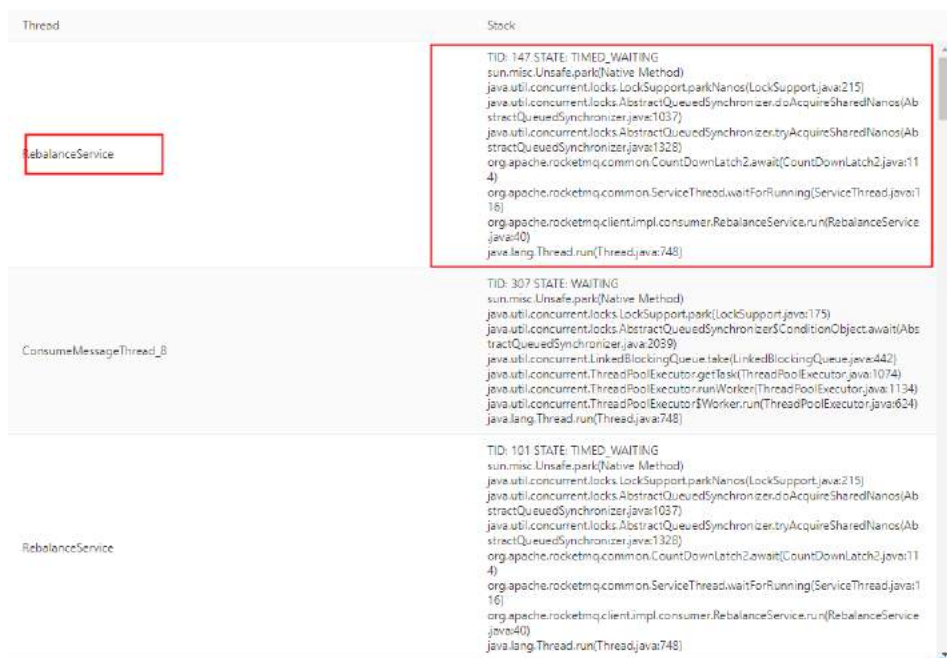
直接 catch 异常，然后返回成功，这个是有风险的，如果处理消息异常，会导致不会再收到。

消息默认重试 16 次后会进入死信队列中，可以选择重新发送或者三天后会自动清理，或者可以在生产者中 try catch 异常直接返回成功状态，该条消息则不会再重试和消费。

消费者出现 rebalanceService 线程

【问题描述】：

消费者出现 rebalanceService 线程。



【问题回答】：

这个是消费端负载均衡的线程，rebalance 是有时间间隔的。

这个是正常的一个线程状态。

延迟消息被立马 / 提前消费

【问题描述】：

发送的是延时消息，为何立马就被消费端消费了或者设置的延时时间还没有到就被消费了。

【排查步骤】:

1. 确认发送延时消息的 api 是否和官方提供的 demo 是一致的。
2. 是否设置了如图所示的属性。



3. 如果核实这两者都没有问题, 需要将实例 id/ 地域 /topic/gid/ 发送代码信息收集完整, 提供给技术支持进行查看。

顺序消息没有被顺序消费

【问题描述】:

为什么发送的顺序消息, 到那时消费的时候乱序了。

【排查步骤】:

1. 检查下发送消费的相关代码, 是否使用的是顺序消息的 api。收发顺序消息。
2. 如果检查发现使用的 api 都正确, 这时可以通过消息轨迹和消息的存储时间来判断消息发送的先后顺序。可能是当时发送时间的先后顺序问题导致您所认为的消费乱序。
3. 如果以上都排查了, 没有问题, 需要将实例 id/ 地域 /topic/gid/ 发送消费代码提供给技术支持, 进行进一步排查。

消息一直没有被消费

【问题描述】：

某条 / 些消息一直没有被消费。

【排查步骤】：

1. 检查消费者是否在线。
2. 检查消息的 tag，该消费者是否有订阅。
3. 检查消费者是否订阅关系保持一致。一旦订阅关系不一致，消息消费的逻辑就会混乱，甚至导致消息丢失。
4. 业务上核实是否真的没有收到该消息，mq 是保证以消息准确到达服务端被消费为第一优先准则，所以在网络环境较差或抖动的情况下，消息轨迹可能会丢失，根据消息轨迹查询到的消费状态是未消费，但是实际上该消息已经被消费了。

消息没有被立马消费，延迟很久才被消费

【问题描述】：

消息发送时间到消费中间这段时间延迟比较久，例如上午发送的消息，到下午才被消费。

【排查步骤】：

1. 查看下发送的时间段，topic 的发送量和消费量对比，看下是否处于一个业务高峰期，导致消息有堆积，如果有消息堆积，那么延迟投递是正常现象。

如果一直有堆积，建议增大消费能力，比如优化消费逻辑业务代码，或者增加消费者机器。

2. 查看 ons.log 日志，如果在 ons.log 中看到了。

```
2019-07-05 16:46:39,039 WARN RocketmqClient - the cached message count exceeds the threshold 1000, so do flow control, minOffset=4731, maxOffset=5738, count=1008, size=0 MiB, pullRequest=PullRequest [ consumerGroup=GID_MR_SG_ACTIVITY_TEST, messageQueue=MessageQueue [ topic=T_MR_SG_USER_TASK_NORMAL_TEST, broker-Name=hz-share2-03, queueId=7 ], nextOffset=5739 ], flowControlTimes=7235001。
```

这个警告是因为客户端消费能力不足，导致拉取消息也被暂停了。此时就需要增加消费能力，比如增加消费线程，或者增加消费者实例。

3. 查看下是否是消费速度太低，可以根据消费者状态当中的实时消费速度判断，如果消费速度过低，那么消息存放的队列堆积就会比较严重，影响消费时间，建议配置上堆积监控。配置后超过多少堆积告警，或者考虑增加消费实例，让消费能力稍微大于生成能力。

重置消费位点失败

【问题描述】：

重试消费者的消费位点没有效果，还是有许多堆积。

【问题回答】：

1. 确认是否为集群消费（因广播模式下，消费位点都由客户端本地维护，因此广播消费模式下不支持重置消费位点）。
2. 确认当前消费者是否在线，消费者必须在线才能重置消费位点。
3. 确认 sdk 版本，sdk 版本过低也会导致此问题。（建议升级 sdk，至少是 1.8.0 版本。
4. 可以点击单个 c 端的详情查看当前消费者当中的消息是否是在重试队列当中的，如果选择的是清除所有堆积消息，从最新位点开始消费方式重置消费位点，是不能清除重试队列里的消息的。这个时候可以选择使用按时间点进行

消费位点重置方式进行重置，此方式没有这个限制。

同一条消息，为什么发送方和订阅方的 msgid 不一样？

【问题描述】：

同一条消息，为什么发送方和订阅方的 msgid 不一样？

【排查步骤】：

1. 检查是否是事务消息或延时消息，如果是，则是正常现象。需要做幂等的话可以使用消息 key 来做幂等。事务消息或延时消息订阅方的 msgid 取的是 transactionId。

通过 `message.getMessageID()` 这个获取的 transactionId 在控制台上是查不到消息的，这时候可以通过 `message.getUserProperties().getProperty("UNIQ_KEY")` 这个来获取真正的 msgid。

2. 因为某些 sdk 的原因，有可能发送方和订阅方的 msgid 也会不一样。任何类型的消息都是这样，这时订阅方的 msgid 是 offsetmsgid。这个很好区分，transactionId 中英文字母是小写的，msgid 和 offsetmsgid 中英文字母都是大写的。

消费者显示离线

【问题描述】：

查看消费者状态，发现消费者是离线状态。

【问题原因】：

1. 消费者实例没有启动。

2. 路由信息未注册到 NameServer 上面。

【解决方案】：

1. 原因里第一点直接启动消费者皆可。
2. 首先确认在 MQ 控制台已经创建了 GID。
3. 客户端可根据 ons.log 来排查是否有 No route info of this topic 的相关报错信息。
4. 此时重点检查实例化消费者参数时 NAMESRV_ADDR 参数配置是否与 MQ 控制台上的一致；排查接入点是否异常。

步骤：删除客户端日志 -----> 重启应用 -----> 查看最新的 ons.log。

同时确认 NameServer 是可连接的，telnet 控制台提供的接入点看下是否 telnet 通。

如果不通，需要看下是不是该机器所属地域与 mq 实例所属不是同一个地域。

ackmessage 时报错 Number of receiptHandles in XML is out of range

【问题描述】：

http 协议的 RocketMQ ackmessage 时返回了 message: Number of receiptHandles in XML is out of range 错误。

【问题原因】：

ack 删除消息时，在 xml body 体中的 receiptHandles 数量大于 16 条。

一次最多是 16 个，超过就会报错了。

启动消费端提示 Group ID 重复

【问题描述】：

启动消息队列 Rocket MQ 版的 Producer (生产者) 实例或 Consumer (消费者) 实例时，提示 Group ID 重复。

【问题原因】：

单个 JVM 进程中出现下列情况时，会导致客户端启动失败，提示 Group ID 重复：

启动了一个以上的 Producer 实例，并且这些实例使用相同的 Group ID。

启动了一个以上的 Consumer 实例，并且这些实例使用相同的 Group ID。

启动了一个以上的 Producer 实例和一个以上的 Consumer 实例，并且这些实例使用相同的 Group ID。

【解决方案】：

单个 JVM 进程中使用相同 Group ID 的 Producer 实例或 Consumer 实例支持情况如下表：

实例	0个Consumer	1个Consumer	1个以上Consumer
0个Producer	不涉及	支持	不支持
0个Producer	支持	支持	不支持
1个以上Producer	不支持	不支持	不支持

如果出现上述不支持的情况，请改正后重新启动应用。

消费者启动时提示获取 Topic 队列失败

【问题描述】：

消息队列 RocketMQ 版中 Consumer（消费者）主动订阅消息，启动客户端时系统提示获取 Topic 队列失败。

【问题原因】：

导致此问题的主要原因是客户端中订阅的 Topic 未在消息队列 RocketMQ 版的控制台创建。

【解决方案】：

1. 登录消息队列 RocketMQ 版的控制台，创建 Topic 和 Group ID。
2. 更新客户端中订阅的 Topic 和 Group ID，确保其与控制台中创建的信息一致。
3. 重新启动客户端，确认问题已经修复。

应用内存不足

【问题描述】：

1. 在应用部署的机器上通过查看内存已消耗完。
2. 在 `{user.home}/logs/ons.log` 能搜索到 `OutOfMemory` 关键字。
3. 在消息队列 RocketMQ 版控制台：进入消费者管理 > 消费者状态，堆积量栏显示消息堆积较多，连接状态栏显示了各个已连接客户端的消息堆积。通过 Jstack 排查，`ConsumeMessageThread_` 线程无消费卡住现象。

【问题分析】:

在 1.7.0.Final 版本之前, 默认客户端最多会给每个 Topic 的每个队列缓存 1000 条消息。假设每个 Topic 的队列数是 16 个 (集群 2 主 2 备, 每台 broker 上 8 个队列), 该 Topic 下单条消息平均大小为 64 KB, 那么最终该 Topic 在客户端缓存的消息 Size: $16 * 1000 * 64 \text{ KB} = 1 \text{ GB}$ 。如果用户同时订阅了 8 个 Topic 都在客户端内存缓存消息, 最终占用内存将超过用户的 JVM 配置, 导致 OOM。

原因 1:

依赖 1.7.0.Final 之前的 ons-client 版本, Topic 的平均消息大小超过 4 KB, 并且消息消费较慢容易在客户端内存缓存消息。

确认方式:

在 `{user.home}/logs/ons.log` 能搜索到 `OutOfMemory` 关键字; 或者通过 `jmap -dump:live,format=b,file=heap.bin <pid>` 命令确认哪些对象占用了大量内存。

恢复方案:

升级 ons-client 版本至 1.7.0.Final 或以上, 同时给对应的 ConsumerBean 配合设置 `com.aliyun.openservices.ons.api.PropertyKeyConst#MaxCachedMessageSizeInMiB` 参数, 然后重启应用。

原因 2:

依赖 1.7.0.Final 及以上的 ons-client 版本, 默认最大消耗内存 512 MB (Group ID 订阅的所有 Topic 缓存总和); 如果应用仍然出现 OOM 现象, 可在 ConsumerBean 启动时, 配置:

`com.aliyun.openservices.ons.api.PropertyKeyConst#MaxCachedMessageSizeInMiB` 参数自行定义最大消耗内存 (范围在 16 MB ~ 2048 MB)。

确认方式:

确认应用依赖的 ons-client 版本, 并通过 JVM 确认给进程分配的内存大小。

恢复方案:

根据应用机器的内存使用情况给对应的 ConsumerBean 设置 `com.aliyun.openservices.ons.api.PropertyKeyConst#MaxCachedMessageSizeInMiB` 参数, 然后重启应用。

消费者报错参数不合法

【问题描述】:

消费者报错参数不合法。

【异常及描述】:

参数不合法的情况有以下几种:

1. 异常: `consumeThreadMin Out of range [1, 1000]`

描述: 消费端线程数设置不合理。

2. 异常: `consumeThreadMax Out of range [1, 1000]`

描述: 消费端线程数设置不合理。

3. 异常: `messageListener is null`

描述: 未设置 `messageListener`。

4. 异常: `consumerGroup is null`

描述: 未设置 Group ID

5. 异常: msg delay time more than 40 day

描述: 定时消息延时不能超过 40 天。

消息显示 `_Consumed_`, 但消费端未感知到

【问题描述】:

消息状态显示 “Consumed”，但是消费端业务日志显示没有收到消息。

【问题原因】:

1. 业务代码在接收到消息后，不立即打印消息

收到消息后，如果直接进入业务逻辑，一旦代码遗漏某个逻辑分支，就会导致消息信息没有被留在业务日志里，造成没有收到消息的假象。

建议您收到消息后，立即打印消息信息留存 `messageId`, `timestamp`, `reconsumeTime` 等。

2. 消费端部署了多个消费实例

尤其是在调试阶段，消费端不可避免会多次重启，一旦多个消费进程同时存在（进程未退出），那么相当于进入集群的消费模式，多个消费实例会共同分担消费消息。以为没有收到的消息，其实是被另一个消费端接收了。

到消息队列 RocketMQ 版控制台，进入 Group 管理 > 消费者状态 > 连接状态，会显示消费端的实例部署情况（有几个消费实例，各自的连接 IP 等等），然后可以自行排查。

3. 消息消费过程出现未被 Catch 的异常，导致消息被重新投递。

批量消费不起作用

【问题描述】：

使用的是批量消费的 api BatchConsumer，为何每次消费的消息都是一条或者不是我设置的属性 PropertyKeyConst.ConsumeMessageBatchMaxSize 的值？

【问题原因】：

这个是在预期范围内的。这个是根据当时队列有多少条消息未消费来决定的，并不能做到每次消费的消息是一个固定值。PropertyKeyConst.ConsumeMessageBatchMaxSize 取值范围为 1-32。

消息轨迹常见问题

消息轨迹显示消费结果未返回

【问题描述】：

消息轨迹显示消费结果未返回。



【问题原因】：

消费消息的方法尚未返回结果，或者中断，导致本次消费结果未传回服务端。

【建议方案】：

建议您不要把业务逻辑放在返回给 mq 服务端的代码之前，最好是保证尽快给到 mq 响应。这样可以避免消息消费失败，在进行消息重试。如果您的业务逻辑时间的确很长，建议您可以将信息拉取到之后，存到数据库、redis 当中，然后尽快给到 mq ackMessage，之后再异步进行业务消费。

消息已经消费，但是消费轨迹显示尚未消费

【问题描述】：

消息本地日志记录该消息已经完成消费，但是在控制台查询消费轨迹却显示该消息尚未消费。

【排查步骤】：

1. 首先确认发送方或订阅方应用的 sdk 版本 (1.2.7 版本及以上)。
2. 查看发送端或消费端启动时，将 PropertyKeyConst.MsgTraceSwitch 这个属性值设置是否为 false。这个属性是启用消息轨迹的开关。
3. 如果 sdk 版本和发送端消费端实例化时参数配置都正常，可以让查看那个时间段的 ons.log, 检查日志中是否有：

```
send trace data,the traceData is Pub ...  
send trace data,the traceData is SubBefore ...  
send trace data,the traceData is SubAfter ...
```

等等相关日志。出现了相关日志，说明消息轨迹发送失败了。所以会缺失，不完整。

因为消息轨迹是异步发送的，存在的发送失败的可能，具体以您业务中是否有被消费为准。

轨迹消息在网络不稳定，客户端机器压力大等一些情况下，可能存在发送失败的可能性。因为轨迹消息重要性不及业务消息，这种情况一般不会重试，所以存在少量的消息无法采集到的情况。所以会看到上面的问题。后面我们会继续优化。

批量导出消息的消费轨迹

【问题描述】：

需要批量导出某个 topic 某段时间内所有消息的消费轨迹。

【问题回答】：

目前后端没有存储某个 topic 的消息轨迹，无法支持批量导出。

提供思路建议：

1. 新建一个 GID, 订阅指定的 topic. 控制台上 Reset 这个新建的 gid 的消费位。

点到指定开始时间点。消费程序调用 OpenAPI 查询轨迹信息：

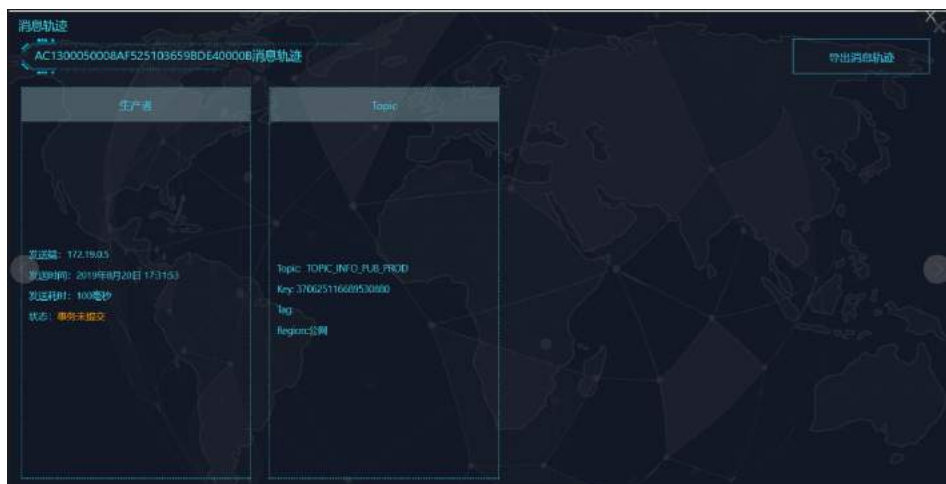
https://help.aliyun.com/document_detail/59830.html?spm=a2c4g.11186623.6.711.7fd27e80IB-F0mT

2. 注意一点，OpenAPI 本身是运维接口，需要控制查询速度。

事务消息事务已提交，但是轨迹显示事务未提交

【问题描述】：

事务消息事务已提交，但是轨迹显示事务未提交。事务消息提交的轨迹记录丢失如下图所示。



【问题原因】:

事务消息的半消息存在 db 当中，消息 commit 的时候才会存到 broker，但是轨迹是发到其他机器完成轨迹的记录，收到轨迹记录请求的目标机器并非原来发送方的机器，导致轨迹记录失败。

消息轨迹名词解释

【问题描述】:

关于消息轨迹名词的解释。



【问题回答】:

1. 发送耗时:

这个耗时的时间是调用发送消息的 `com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.MQClientAPIImpl#sendMessage()` 的耗时时间。如果是同步发送，则会包含存储到 broker 的时间，异步和 `oneWay` 则不包含。

发送端的代码逻辑可以看 `DefaultMQProducerImpl.sendKernellImpl()` 中执行的

```
com.aliyun.openservices.ons.api.impl.tracehook.
OnsClientSendMessageHookImpl#sendMessageBefore
com.aliyun.openservices.ons.api.impl.tracehook.
OnsClientSendMessageHookImpl#sendMessageAfter
```

2. 消费耗时

这个时间是消费耗时，是调用客户端的 `consume()` 的整个花费的时间。也就是图 2 的 `SubAfter` 的 `TimeStamp` 减去 `SubBefore` 的 `TimeStamp` 的时间。

消费段的代码逻辑可以看

```
com.aliyun.openservices.shade.com.alibaba.rocketmq.client.impl.consumer.  
ConsumeMessageConcurrentlyService.ConsumeRequest#run() (注意, 这是非顺序消费的执行  
逻辑) 中执行的  
com.aliyun.openservices.ons.api.impl.tracehook.  
OnsConsumeMessageHookImpl#consumeMessageBefore  
com.aliyun.openservices.ons.api.impl.tracehook.  
OnsConsumeMessageHookImpl#consumeMessageAfter
```

3. 消费轨迹的 ip 就是本机 ip。linux 环境下利用 ifconfig 查看, windows 环境利用 ipconfig 查看。

消息轨迹中记录消费的 ip 与实际消费该消息的 ip 不一致

【问题描述】:

消息轨迹当中记录的消费端的消费 ip 与实际业务上消费该消息的机器的 ip 不一致。

【问题原因】:

如果消费消息的 ecs 当中有 docker, mq 取的客户端 ip 会取到 docker 网卡的 ip, 而不是 ecs 的内网 ip。

这个是您运行环境下拿出来的 ip, 如果是 docker 环境, 不一定是真实 ip。

具体消费的 ip 以实际您自己业务当中记录的 ip 为准。

其他问题排查

控制台相关问题

【实例详情】--【数据统计】消息堆积 top

【问题描述】：

在【实例详情】--【数据统计】当中看到某个 group 有堆积，实际上【group 管理】--【消费者状态】当中查看这个 group 并没有堆积。

【问题原因】：

实例详情 -- 数据统计当中的消息堆积量是有一定延迟的，是采样数据，不是实时数据，服务消费者状态始终是离线，也看不到订阅关系，导致消息一直没法消费可能会是之前的历史数据，具体的还是要以 group 管理当中的实时消息堆积量为准。

为什么实例下没有该 topic，创建的时候却提示该 topic 已存在

【问题描述】：

创建 topic 的时候，该实例下不存在新建的 topic，创建的时候却提示该 topic 已存在。



【问题原因】：

没有命名空间的实例下的 topic 我们是要求全局唯一的，也就是说这个 topic 名称在全阿里都是要求全局唯一的。这个 topic 名称应该是已经被其他的用户使用过了。

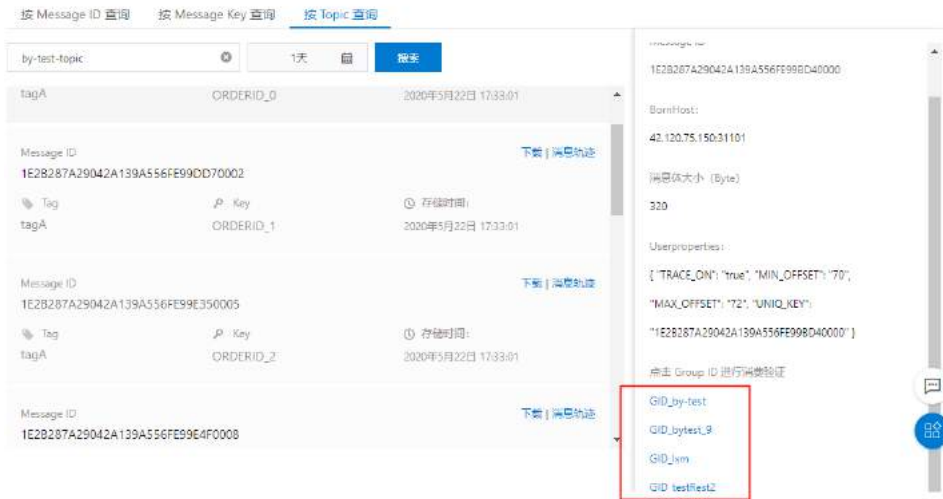
可以在【实例详情】--【实例信息】当中查看该实例是否有命名空间。



为查询出来的消息，显示的消费者超过实际的消费者

【问题描述】：

这个 topic 只有一个消费者在消费，但是实际上查出来可以进行消费验证的消费者却不止当前有订阅关系的消费者。



【问题回答】:

如果这个 gid 曾经订阅过这个 topic(现在不订阅了)/ 现在正在订阅这个 topic, 并且在控制台的 group 管理可以查得到这个 gid, 那么查询消息的时候就会显示在可以进行消费验证的地方。

如果您这个 gid 已经在控制台被删除了, 那么查询的时候, 就不会再看到有这个 gid 了。

消费者状态连接信息的客户端 ID, 公网 IP 是什么?

【问题描述】:

消费者状态连接信息的客户端 ID, 公网 IP 是什么?

【问题回答】:

1. 客户端 id 就是获取的本地真实 ip, linux 环境下利用 ifconfig 查看, windows 环境下利用 ipconfig 查看, 使用的是 NetworkInterface.getNetworkInterfaces() 获取的。

- 公网 ip 可以理解为代理 ip，有了这个 ip 才可以上网。linux 环境可以利用 curl ifconfig.me 或 curl cip.cc 查看，windows 环境直接百度搜索 ip 查看。

资源报表当中采集类型的 tps 是什么意思

【问题描述】：

资源报表当中采集类型的 tps 是什么意思？



【问题回答】：

是 TransactionsPerSecond 的缩写，也就是每秒处理的事务数。MQ 中主要用来表示 某个 topic 的每秒钟生产了多少消息数量或某个消费组每秒消费了多少消息数量。

为什么控制台查看不到 pid 了

【问题描述】：

为什么代码当中的 pid 现在在控制台查看不到了。

【问题回答】：

- RocketMQ 现在已经没有 pid 的概念了，所以原先代码当中填入的 pid 都是

无法查询到的。

2. 如果不想代码当中有 messageProducerId 配置，建议将 sdk 升级到最新版本，参照文档当中的最新代码的写法，修改代码。

不建议直接将 pid 改成 gid。因为现在的最新的消息生产者当中只需要填写发送消息的 topic，不需要填写 pid 和 gid 了。

需要注意的是新版本的 sdk 是兼容旧版本的，

虽然都是兼容的，切换版本后，也请做好线下测试，再线上灰度发布。

如果后续新建实例，有命名空间的，接入点配置项需要变化，代码中的 ON-SAddr 需要修改为 NAMESRV_ADDR。

有命名空间的实例，每个实例的接入点不同，在控制台“实例管理”页面可以看到接入点地址。

消费者状态当中的消息延迟时间是如何计算的

【问题描述】：

消费者状态当中的消息延迟时间是如何计算的。

消费者状态				
 离线	实时消费速度 0 条/秒	实时消息堆积量 0	最近消费时间	消息延迟时间 0s

【问题回答】：

消息延迟时间的计算公式：延迟时间 = 未消费消息在队列中的最大时间 = 当前时间 - 最开始堆积的那条消息的时间。

告警监控相关问题

收到告警，但是控制台却查看不到报警规则

【问题描述】：

接到了监控告警消息，但是在控制台查看该监控告警的规则，却没有看到。

【问题原因】：

Mq 的监控报警只能有创建该监控报警的账号登陆才能查看，主账号以及其他账号不能查看监控。

因此如果自己的账号查看不到该告警，那么表明是其他账号设置了报警。

设置了报警，没有收到报警短信

【问题描述】：

为什么设置了报警，查看资源也的确符合报警的规则，但是却没有收到报警的短信通知。

【排查步骤】：

1. 首先确认实例所在地域（Region）支持监控报警功能。
2. 确认设置的报警接收人的手机号码为中国大陆地区的手机号。
3. 确认当前消费者是否在线，报警必须要当前消费者在线才可以。

资源包费用相关问题

Api 计费标准

【问题描述】：

Api 是如何计费的？

【问题回答】：

API 请求次数 = 发送消息 API 请求次数 + 订阅消息 API 请求次数 + 长轮询 API 请求次数；

以普通消息为例，发一条消息调用 api 和消费该条消息 算是调用了两次 api。

长轮询就是说针对每一个队列 (queue)，消费端会每 15 秒拉取一次消息，看看 topic 当中是否有可见的需要消费的消息。每一次拉取都算是一次调用 api，都会收取费用。

所以一个 topic 下面如有有 3 个 broker，每个 broker 有 8 个队列，如果该 topic 当中一整天都没有消息进来，那么一天就是 $24 \times 24 \times 60 \times 4$ 次调用。所以建议如果消费消息不多，可以隔一段时间重启一下消费段。

长轮询是建立在消费端在线的状态下。

队列 queue 个数是本身实例决定的，这个目前不支持用户侧配置 Topic 底层对应的 queue 数目。并且不支持修改 queue 的个数。

标准版实例 tps 用量限制

【问题描述】：

标准版实例的 tps 使用限制是多少。

【问题回答】：

单实例收发 TPS 限制 5000 条 / 秒，单个实例的使用限制是保证 tps 最大到 5000 是一定有的，但是实际会超过 5000，超过的性能不保障。

但是铂金版实例的 tps 是可以在购买的时候定制的，如果想要更加高的 tps 建议直接购买铂金版，因为如果在标准版实例的基础上申请调大 tps，可能花费的金额会比直接购买铂金版实例还要高。

计费常见问题

【问题描述及回答】：

1. 问：每天扣除 2 元，这是什么费用？

答：消息队列 RocketMQ 版费用 = API 调用费 + Topic 资源占用费。每个 Topic 资源占用费为 2 元 / 天。

2. 问：我昨天删除了 Topic，为什么今天会收到账单并被扣费？

答：Topic 资源占用费以每天 00:00-23:59:59 为周期进行计费，第二天收费。所以，昨天删除 Topic，昨天的计费系统已经计入，今天会出账单进行扣费，明天就不会出账单扣费。

3. 问：我没有使用消息队列 RocketMQ 版服务，为什么会扣费？

答：请查看账单，检查您是否有使用消息队列 RocketMQ 版服务。

如果您不需要使用消息队列 RocketMQ 版服务，请及时删除消息队列 RocketMQ 版控制台上所有资源，避免不必要的支出。

4. 问：我有看到消息队列 RocketMQ 版的扣费，但是我在控制台上看到没有开通消息队列 RocketMQ 版？

答：消息队列 RocketMQ 版处于欠费状态超过 72 小时，阿里云将暂停为您提供服务，即您不能再访问消息队列 RocketMQ 版控制台与消息队列 RocketMQ 版 API。但是您的消息队列 RocketMQ 版服务被释放之前的欠费仍然需要结清。

5. 问：如何关闭消息队列 RocketMQ 版服务？

答：请删除所有地域 Topic 资源，并将所有发送端，消费端停止。

6. 问：一天的消息总量为 631,238 条，但是通过费用查询结果：API 调用次数为 126,315,056 次，那么多 API 调用次数怎么来的？

答：API 调用次数 = 发送消息 API 调用次数 + 订阅消息 API 调用次数 + 长轮询 API 调用次数（长轮询说明：消息队列 RocketMQ 版 Consumer 为保证消息实时推送而产生的 API 调用，每个队列 15 秒一次长轮询，在此期间，若队列内有消息产生，则不计长轮询次数）。

如何查看整个实例的 tps 以及总存储用量

【问题描述】：

如何如何查看整个实例的 tps 以及该实例的总存储用量。

【问题回答】：

整个实例的总 tps：

取资源报表里每个 topic 一天当中高峰期 tps 最大值，然后加起来，就可以算出来总的 tps。

如果您要在该实例下添加新的 topic，原来的 topic 上分配的 tps 会分流一部分到新创建的 topic 上面（如果实例下原有的所有 topic 的 tps 总和已经快达到上限）。

就是说原来的 topic 的 tps 可能会降低一些，然后会分流给到新创建的 topic。

消息存储空间 也是有计算公式的：每个 topic 下每天发送的总量 相加 再乘以每条消息的字节数 就可以算出一天的大小。

而且 broker 磁盘使用率超过总量的 80%，就会去删除过期的消息，不会存在磁盘不够用的情况（过期消息：已经消费过的还没有被删除的消息）。

在原有实例上升级成铂金版，支付后却多出一个新的实例？

【问题描述】：

在原有的标准版实例上直接点击铂金版升级，完成订单支付后，原有实例没有变成铂金版实例，但是却新增多出一个铂金版实例。

【问题回答】：

由于之前很多用户是不想把标准版上的所有资源都升级到铂金版，所以目前是调整为升级铂金版默认生成一个新的实例。

客户侧把确认需要资源迁移到铂金版之后，再决定是否保留原来的标准版实例。

如果需要铂金版实例的接入点还是和原有实例的接入点保持一致，后端会将接入点同步成原来的。

日志相关问题

升级 c++ 版本到 2.0 后，日志占用磁盘问题

【问题描述】：

c++ 的 sdk 升级到 2.0 后日志过多，甚至导致开发环境磁盘 100%，这个怎么限制日志大小。

【问题答案】：

C++ 的 tcp 的 sdk 内置的是 java 的日志，目前这个日志过多，建议写个日志切割脚本，定时执行日志切割删除，目前服务端没有对日志进行定时切割和删除的动作。

Ons.log 客户端日志说明

【异常描述及说明】：

1. 打印信息：

```
[persistAll] Group: CID_XXXX ClientId: 10.31.40.100@171374#14159488#-2036649484#20931314294957812
updateConsumeOffsetToBroker MessageQueue [topic=XXXX, brokerName=qdinternetorder-02,
queueId=5] 1013923
```

说明：这种现象说明消息已经消费成功、并且在 MQ 服务端已持久化消费进度；

MessageQueue 里包括了消息主题、对应的 brokerName, 消费队列的 id。

解决方案：暂无。

2. 打印信息：

```
HeartbeatData [clientId=XXX, producerDataSet=[
```

```
ProducerData [groupName=XXXX], ProducerData [groupName=XXX], consumerDataSet=[]
send heart beat to broker[XX] success
```

说明：该现象是向服务端发送心跳包成功，HeartbeatData 是心跳相关的信息，包括客户端 id、发送组信息、消费组信息。

解决方案：暂无。

3. 打印信息：

```
[PULL_TPS] [CID_XXXX@CID_XXXX] Stats In One Minute, SUM: 0 TPS: 0.00 AVGPT: 0.00
[PULL_RT] [%RETRY%CID_XXXX@CID_XXXX] Stats In One Minute, SUM: 0 TPS: 0.00 AVGPT: 0.00
```

说明：该类信息打印的是从 consumeQueue 中拉取消息时的 TPS（每秒 request 的数量）。

解决方案：暂无。

4. 打印信息：

```
[TIMEOUT_CLEAN_QUEUE]broker busy, start flow control for a while, period in queue: 905ms,
size of queue: 1164
```

说明：服务端压力过大、处理不了过多的请求；由于 mq 服务端在存储数据时是先写入 pageCache, 然后去刷盘、因此每隔 10s 会去清理过期的请求（此过程会判断缓存页是否繁忙）。

解决方案：（1）扩容，增加 broker，分担压力。

（2）osPageCacheBusyTimeOutMills 属性值调大。

5. 打印信息：

```
execute the pull request exception
com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException:
CODE: 25 DESC: the consumer's subscription not lates
```

说明：1. updateConsumeOffsetToBroker 日志是正常的，是一个定时任务，定期同步 offset 到 Broker。

2. theconsumer's subscription not latest 这个日志在重启后是当前的一个正常处理逻辑，为了达到 "Consumer 集群的负载均衡" 的一种机制：当一个新的 consumer 进程加入，客户端需要重新分配 queue 以实现负载均衡，此时会短暂的阻止 pull 请求，直到该 consumer 被重新分配了 queue。

解决方案：暂无。

6. 打印信息：

```
[WRONG]mq is consuming, so can not unlock it, MessageQueue [topic=XX, broker-Name=szorder2-02, queueId=1]. maybe hanged for a while, 2
```

说明：进行负载均衡时，对消息处理队列尝试加锁，如果 1s 内还未加锁成功，说明当前消息处理队列已经有消费者在访问，不能进行解锁。

解决方案：暂无。

7. 打印信息：

```
doRebalance, XXX-CID, add a new mq failed, MessageQueue [topic=XXXX, broker-Name=szorder2-02, queueId=5], because lock failed
```

说明：用户使用的是顺序 Topic，为了保证单个分区中消息的顺序消费，这个会有个 lock 的机制。客户端有这个日志说明其中某个分区已经有客户端在消费了。

解决方案：暂无。

8. 打印信息：

```
get Topic [XXXXXX] RouteInfoFromNameServer is not exist value
com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQClientException: CODE:
17 DESC: No topic route info in name server for the topic: TOPIC_XXXXX
See http://rocketmq.apache.org/docs/faq/ for further details.
```

说明：nameServer 列表非空，可连接到 nameServer，但 topicList 为空，或

topicList 状态 非 OK，比如 curl 接入点 onsaddr 异常。

解决方案：

- (1) AK/SK 配置错误
- (2) 用户没有控制台于当前实例下创建了 GID
- (3) 实例化的代码中，NameServerAddr 没有配置正确
- (4) 确认 topic 权限可用，topic 要求 permission 是 6 (rw-) 至少为 2 (-w-),
用 mqadmin topicRoute 可排查路由信息

9. 打印信息：

```
com.aliyun.openservices.ons.api.impl.authority.exception.AuthenticationException: signature
validate by dauth failed
```

说明：AK/SK 配置错误。

解决方案：AK/SK 要配置创建该 GID 使用的 AK/SK。

10. 打印信息：

```
NettyClientPublicExecutor_3 - execute the pull request exception
com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException:
CODE: 26 DESC: subscription group [CID_XXX] does not exist,
See http://rocketmq.apache.org/docs/faq/for further details.
```

说明：订阅关系没有推送到 MQ 服务器上。

解决方案：subscription.json 文件里直接添加 GID 对应的信息即可。

11. 打印信息：

```
execute the pull request exception
com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException:
CODE: 24 DESC: the consumer's subscription not exist
```

说明：缺少订阅关系。

解决方案：暂无。

12. 打印信息:

```
sendKernellmpl exception, resend at once, InvokeID: -4054089080884425405, RT: 7183ms,
Broker: MessageQueue [topic=xxxx, brokerName=xxx, queueId=2]
com.aliyun.openservices.shade.com.alibaba.rocketmq.remoting.exception.RemotingTimeoutEx-
ception: wait response on the channel <xx.xxx.xxx.xx:8080> timeout, 5000(ms)
```

说明: 客户端应用到 MQ 服务器的网络不通。

解决方案: 可根据客户端当时的网络监控情况、来判断网络流量是否有下跌或者异常情况。

13. 打印信息:

```
decode exception, xxx.xxx.xx.xx:8080
```

说明: 发送的消息 body 长度超范围。

解决方案: 检查消息 body 的大小, 最大为 4MB, 为了提高性能、一般建议消息在 4kb 之内。

14. 打印信息:

```
system mqtrace hook init failed ,maybe can't send msg trace data
```

说明: 消息轨迹初始化失败, 不影响消息的正常收发功能。

解决方案: 通过设置 `producerProperties.setProperty(PropertyKeyConst.MsgTraceSwitch, String.valueOf(false))`。

暂时关闭消息轨迹功能。

15. 打印信息:

```
updateConsumeOffsetToBroker exception, MessageQueue [topic=xxxx, brokerName=xxxx,
queueId=1] com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQClientEx-
ception: The broker[xxxx] not exist For more information
```

说明: 服务端更新消费位点失败; 出现此现象, 要确认下是否在升级窗口时间,

升级期间可能会出现此异常。MQ 是集群部署，一台发送失败，会自动重试到其他集群上面，不影响消息的核心功能。

解决方案：暂无。

16. 打印信息：

```
sendMessageBack Exception, CID_XXXX
```

```
com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 16 DESC: the broker[xxx.xxx.xxx.xxx] sending message is forbidden
```

说明：服务端发送消息失败，出现此现象，要确认下是否在升级窗口时间，升级期间可能会出现此异常。

解决方案：暂无。

各个 sdk 的 ons.log 日志位置

【问题描述】：

各个 sdk 的 ons.log 日志位置。

【问题回答】：

```
Java tcp:
window 系统: C:\Users\用户名\logs\ons.log
linux 系统: /{user.home}/logs/ons.log

.net tcp:
C://log

Python tcp:
C++ tcp:
{home}/logs/rocketmqlogs/rocketmq_client.log
```


权限相关问题

访问控制策略当中添加的 ip 限制没有生效

【问题描述】：

在自定义的 mq 访问控制的授权策略当中添加了 ip 的限制，但是使用非策略当中的 ip 还是可以成功发送 / 接收消息。

【问题回答】：

策略当中的 ip 一定是公网 ip/ 宿主机 ip，下图绿框的。不可以是客户端 ID。

并且策略当中的 ip 不可以是 ecs 的弹性公网 ip (eip)。

AliyunMQReadOnlyAccess 权限可以创建 gid

【问题描述】：

给了子账号 AliyunMQReadOnlyAccess 系统的只读权限，但是账号却可以创建 gid 和删除 gid。

【问题回答】：

拥有 AliyunMQReadOnlyAccess 只读权限的子用户是可以创建 gid 的，目前 mq 只针对 topic 有创建和删除的较细粒度的权限支持，目前针对 gid 是没有这么细粒度的支持的。

使用常见问题

如何删除 / 清理消息

【问题描述】：

如何删除已经发送到 topic，但是尚未消费的消息。

【问题回答】：

无法主动删除这样的消息。

- (1) 一条消息到了 topic 当中最多保留 3 天，超过时间将自动滚动删除。
- (2) 如果消息已经发送到 topic，但是不想消费该消息，可以使用重置位点的方式，选择跳过这条消息不进行消费，具体的重置位点可以参考：https://help.aliyun.com/document_detail/63390.html?spm=a2c4g.111186623.6.622.263812bdT-feFRo。

本地 IDC 服务可以接入非公网地域的 RocketMQ 吗

【问题描述】：

idc 的网络与云上的网络是不通的，但是 idc 需要调用云上的这个 rocketmq，可以实现吗？

【问题回答】：

1. 有专线或 vpn 就可以加到 RocketMQ 阿里云 IP 段 (100.64.0.0/10) 的内部网段路由。

2. 可以使用 http 协议的公网接入，需要注意的是使用 http 接入点，需要使用 http 对应的 sdk，不可以使用 tcp 协议的 sdk。

实例购买在北京地域，可以添加 tcp 公网的接入点吗？

【问题描述】：

购买了北京地域的实例，想要使用 tcp 公网进行访问，但是只看到 tcp 内网接入点，可以申请添加一个公网接入点吗？

【问题回答】：

不可以的。

如果要用公网，只能在公网地域创建实例。会提供 tcp 公网接入点。

非公网地域，只能使用内网接入点接入的。

原本 mq 是只有 tcp 协议的。最开始设置的时候，从网络，安全各方面考虑，所以设计了地域区别。

也不支持将实例进行地域之前的迁移。

Ecs 由经典网络切换到 vpc 网络接入 mq 问题

【问题描述】：

目前是经典网络的 ecs，想要将 ecs 换成 vpc 环境，并且不给 ecs 添加公网 ip。在不变动代码接入点的情况下，直接将 ecs 的网络环境更换，会影响 mq 的使用吗？

【问题回答】：

如果使用的是 mq 的内网接入点，是不会影响使用的。

Mq 的内网接入点，经典网络和 vpc 网络都是可以访问的。

如何输出消息的 msgid

【问题描述】：

如何将消息的 msgid 输出打印出来。

【问题回答】：

MSGID 可以使用 `message.getUserProperties("UNIQ_KEY")` 来获取控制台可以查询到的 ID。

是否支持消息优先级

【问题描述】：

ons 里面的消息队列是否支持设置消息的优先级？需求就是同一批消息（同一批消息，里面有个字段表示不同的处理方式），想要实现不同业务的一个优先级。

【问题回答】：

不支持。



Tcp 社区版 python 接入操作系统问题

【问题描述】:

开源 python 的 sdk 接入阿里云需要限制的操作系统

【问题回答】:

在开始做准备工作前, 请确保您的操作系统满足以下条件:

Linux: CentOS 6.8、CentOS 7.2、RHEL 6.x、RHEL 7.x

Darwin: macOS Mojave 10.14.x

Docker 官方镜像:

python:2.7-buster

python:3.5-buster

python:3.6-buster

python:3.7-buster

请确保您的 Python 版本满足以下条件:

Python 2.7.10 及以上 2.x.x 版本;

Python 3.5、3.6 或 3.7。

目前测试可以成功安装 sdk 的有如下操作系统、python 版本、pip 版本

centos6.8 python2.7.13 pip20.0.2

centos6.8 python2.7.8 pip20.0.2

Tcp 开源 python 当中消费位点设置

【问题描述】:

Python tcp 开源的 sdk 中有类似 java 的

consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_FIRST_OFFSET) 设置吗?

【问题回答】:

默认 last offset。

实时计算输出结果到 mq。

【问题描述】:

实时计算结果表是 mq 类型的，这个投递出来的消息的具体内容或者报文格式是什么样的呢?

blink 产品这边是直接用的结果表配置的，输出到 mq 的动作应该是内部的操作。

想知道消息的结构，消费这边好去解析。比如结果表是个的话。

```
CREATE TABLE stream_result (  
  live_id VARCHAR,  
  link_url VARCHAR,  
  click_num BIGINT,  
  click_user_num BIGINT  
) WITH (  
  type = 'mq',  
  endpoint = 'c[REDACTED]:8080',  
  accessID = '[REDACTED]oXP',  
  accessKey = '[REDACTED]kF0',  
  topic = '[REDACTED]et',  
  producerGroup = '[REDACTED]net',  
  tag = 'l[REDACTED]tics',  
  encoding = 'utf-8',  
  fieldDelimiter = ',',  
  retryTimes = '5',  
  sleepTimeMs = '500',  
  instanceID = '[REDACTED]bcx'  
);
```

【问题回答】:

实时计算写入目的表 MQ 的格式会是和从源表中读取的格式保持一致的, 例如源表日志服务中记录的是 json 格式的话, 写入 MQ 中的数据也会是 json 格式的。

分区顺序消息的 topic 可以分多少个区

【问题描述】:

分区顺序消息 一个 topic 可以分多少个区? 是否可以自定义设置。

【问题回答】:

分区个数没有限制, 也不可以自定义设置。后端会弹性的根据发送量进行分区数的扩缩。

RocketMQ 商用版 sdk 可以接入自建的 mq 吗?

【问题描述】:

阿里云 rocket 商用版 java sdk 依赖可以接入 自己搭建的 rocket 开源版本吗。

【问题回答】:

不支持的, 商业版本的 sdk 目前都是适配阿里云的 MQ 的。

如果接入希望接入开源和商业版的话, 建议使用 4.5.2 社区版本来统一接入。

https://help.aliyun.com/document_detail/150025.html?spm=a2c4g.11186623.6.639.6e8d7e80przhXa

tcp 连接数不断增多

【问题描述】：

TCP 的连接没有释放，ESTABLISHED 不断增加。如下图所示。



【排查步骤】：

1. 使用 `netstat -nat|grep ESTABLISHED|awk '{print$5}'|awk -F : '{print$1}'|sort|uniq -c|sort -rn` 命令查看服务器 tcp 的连接数，可排查下排在前几位的连接 ip 是客户端还是服务端的 ip。
2. `netstat -an` 查看与本机连接的 ip 有哪些，检查 NameSrv 的连接数
3. 如果应用机器与 NameSrv 的连接数较多，那么在客户的程序中肯定存在循环创建 producer 或者 consumer 的地方，发送者、消费者只需启动时初始化即可。

调用 api 出现 Request was denied due to user flow control

【问题描述】：

调用管控 api 的相关接口时，出现 Request was denied due to user flow control。

【问题原因】：调用相关接口太频繁了，管控 api 都是有限流控制的，需要降低下

调用频率。

Java 的 sdk 是否支持 jdk1.6

【问题描述】：

Java 的 sdk 是否支持 jdk1.6。

【问题回答】：

对于 sdk 版本 1.7.9 及以上版本的，是不支持 jdk1.6 的，最低要求 jdk1.7。

重试时间含义

【问题描述】：

MQ 消费的形式包含无序消费，顺序消费两种；消费模式又包含集群消费和广播消费；消费时由于各种原因会存在消费失败，失败了就会重试，下面将解释一些和消息重试相关的参数。

【问题回答】：

一、MaxReconsumeTimes

含义：consumer 客户端参数，最大重试次数，超过最大重试次数，消息将被转移到私信队列。

作用域：无序和顺序的集群消费起作用。

设置方式：默认值 无序消息 16 次，顺序消息 -1 表示无限次本地重试。

```

// **
// MQ 接收消息示例 Demo
//
public class SimpleMQConsumer {

    public static void main(String[] args) {
        Properties consumerProperties = new Properties();
        consumerProperties.setProperty(PropertyKeyConst.ConsumerId, MqConfig.CONSUMER_ID);
        consumerProperties.setProperty(PropertyKeyConst.AccessKey, MqConfig.ACCESS_KEY);
        consumerProperties.setProperty(PropertyKeyConst.SecretKey, MqConfig.SECRET_KEY);
        consumerProperties.setProperty(PropertyKeyConst.ONSAddr, MqConfig.ONSADDR);
        consumerProperties.setProperty(PropertyKeyConst.MaxReconsumeTimes, "16");
        consumerProperties.put(PropertyKeyConst.MessageModel, PropertyValueConst.CLUSTERING);

        Consumer consumer = ONSFactory.createConsumer(consumerProperties);
        consumer.subscribe("sjyong-test-timed", MqConfig.TAG, new MessageListenerImpl());
        consumer.start();
        System.out.println("Consumer start success.");

        // 等待固定时间防止进程退出
        try {
            Thread.sleep(1000000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

原理：设置在 consumer 端，如何起作用呢？

消息在消费失败重试时会发送回给 broker，会发回 broker 端，MaxReconsumeTimes 的值就会在请求头内设置。

Broker 接受后的处理会把消息的 topic 改成死信队列的 topic，然后进行存储。

二、messageDelayLevel

含义：broker 端参数，控制无序消息并发消费是重试的时间间隔。

作用域：无序并且集群的消费模式；由于有序消息只有在消息超过最大重试次数后才会发回 broker，其他时间一直在本地重试。

设置方式：默认值：1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h，需要再 broker 一侧配置修改。

原理：ScheduleMessageService 通过解析这个配置，将延迟时间不同的消息放到不同的队列，通过 timer 监控每个对列内到期的消息，把消息的 topic 改回成重试的 topic 发回到重试队列内，实现消息重试。

三、ConsumeTimeout

含义：最长的消费时间，如果超过了这个时间就认为消息消费失败了；主要目的是为了能尽早的将 offset 确认，避免消息重复，例如对于并发消息有 100 条 0~99，第 0 条长时间消费没成功，后面的 99 条消费成功了，如果不将第 0 条失败掉，那么 offset 一直不能后移到 100，假如此时进行 rebalance 后面的 99 条就有可能进行重新消费。

设置方式：默认 15 分钟。

```

public class SimpleMQConsumer {
    public static void main(String[] args) {
        Properties consumerProperties = new Properties();
        consumerProperties.setProperty(PropertyKeyConst.ConsumerId, MqConfig.CONSUMER_ID);
        consumerProperties.setProperty(PropertyKeyConst.AccessKey, MqConfig.ACCESS_KEY);
        consumerProperties.setProperty(PropertyKeyConst.SecretKey, MqConfig.SECRET_KEY);
        consumerProperties.setProperty(PropertyKeyConst.ONSAddr, MqConfig.ONSADDR);
        consumerProperties.setProperty(PropertyKeyConst.MaxReconsumeTimes, "16");
        consumerProperties.setProperty(PropertyKeyConst.ConsumeTimeout, "15");
        consumerProperties.put(PropertyKeyConst.MessageModel, PropertyValueConst.CLUSTERING);

        Consumer consumer = ONSFactory.createConsumer(consumerProperties);
        consumer.subscribe(topic: "siyong-test-timed", MqConfig.TAG, new MessageListenerImpl());
        consumer.start();
        System.out.println("Consumer start success.");

        //等待固定时间防止进程退出
        try {
            Thread.sleep( millis: 200000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

作用域：无序集群消费。

原理：对于并发消费 ConsumerMessageConcurrentlyService 启动一个线程按照固定的周期清理 queue 消费时长超过的消息，并发回 broker，这样 offset 就可以前移了。

注意：

1. 由于 ConsumeTimeout 在 broker 启动时，启动应用程序，所以之后修改无效。

2. 由于定期时间是 consumeTimeout，并且每次判断消息开始消费到 现在是否超时，这样有效期可能是两倍。

四、suspendTimeMills

含义：顺序消费时延时的时间。

设置方式：通过 properties 注入，参见之前 默认 1000（1 秒）最大的时间为 30s。

作用域：仅对顺序消费起作用。

原理：每次顺序消费消费失败时，重新提交消费，间隔的时间。



```
case SUSPEND_CURRENT_QUEUE_A_MOMENT:
    this.getConsumerStatsManager().incConsumeFailedTP5(this.consumerGroup, consumeRequest.getMessageQueue().getTopic(), 1);
    if (this.checkReconsumeTimes(msgs)) {
        consumeRequest.getProcessQueue().makeMessageToConsumeAgain(msgs);
        this.submitConsumeRequestLater(consumeRequest.getProcessQueue(), consumeRequest.getMessageQueue(), context.getSuspendTimeMills());
        continueConsume = false;
    } else {
        commitOffset = consumeRequest.getProcessQueue().commit();
    }
}
```

注意：

- 1: suspendTimeMills 和 maxReconsumeTimes 同时设置，就会在 suspendTimeMills*maxReconsumeTimes 毫秒后投递到私信队列。

五、事务消息

Rocketmq 的事务消息，如果在本地的事务完成后服务挂了，返回的结果是 unknown，这时 broker 会有事务回查机制每隔 5 秒进行一次回查，这个是没有次数限制的。

RocketMQ 商用版 sdk 是否可以接入自建开源 MQ

【问题描述】：

阿里云 RocketMQ 商用版 java sdk 依赖可以接入自己搭建的 RocketMQ 开源

版本吗？

【问题回答】：

这个不支持的，商业版本的 sdk 目前都是适配阿里云的 MQ 的。

如果接入希望接入开源和商业版的话，建议使用 4.5.2 社区版本来统一接入。

顺序消息常见问题

【问题描述及回答】：

1. 问：同一条消息是否可以既是顺序消息，又是定时消息和事务消息？

答：不行。顺序消息、定时消息、事务消息是不同的消息类型，三者是互斥关系，不能叠加在一起使用。

2. 问：顺序消息支持哪些地域？

答：支持消息队列 RocketMQ 版所有公共云地域和金融云地域。

3. 问：为什么全局顺序消息性能一般？

答：全局顺序消息是严格按照 FIFO 的消息阻塞原则，即上一条消息没有被成功消费，那么下一条消息会一直被存储到 Topic 队列中。如果想提高全局顺序消息的 TPS，可以升级实例配置，同时消息客户端应用尽量减少处理本地业务逻辑的耗时。

4. 问：顺序消息支持哪种消息发送方式？

答：顺序消息只支持可靠同步发送方式，不支持异步发送方式，否则将无法严格保证顺序。

5. 问：顺序消息是否支持集群消费和广播消费？

答：顺序消息暂时仅支持集群消费模式，不支持广播消费模式。



云服务技术大学
云产品干货高频分享



云服务技术课堂
和大牛零距离沟通



阿里云开发者“藏经阁”
海量免费电子书下载