|  |
| --- |
|  |
| **RocketMQ用户指南** |
| v3.0.0 |
|  |
| ©Alibaba 淘宝消息中间件项目组 |
| 2013/10/5 |

**文档变更历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **主要更改内容** | **更改人** | **更改时间** |
| 1 | 建立初始版本 | 誓嘉vintage.wang@gmail.com | 2013/9/28 |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |

**目录**

[1 前言 1](#_Toc368738010)

[2 客户端使用指南 1](#_Toc368738011)

[2.1 客户端如何寻址 1](#_Toc368738012)

[2.2 自定义客户端行为 2](#_Toc368738013)

[2.2.1 客户端API形式 2](#_Toc368738014)

[2.2.2 客户端的公共配置 2](#_Toc368738015)

[2.2.3 Producer配置 2](#_Toc368738016)

[2.2.4 PushConsumer配置 3](#_Toc368738017)

[2.2.5 PullConsumer配置 4](#_Toc368738018)

[2.3 Message数据结构 4](#_Toc368738019)

[2.3.1 针对Producer 4](#_Toc368738020)

[2.3.2 针对Consumer 5](#_Toc368738021)

[2.4 收发消息例子 6](#_Toc368738022)

[2.5 发送顺序消息 6](#_Toc368738023)

[2.6 顺序消费与乱序消费 6](#_Toc368738024)

[2.7 集群消费与广播消费 6](#_Toc368738025)

[2.8 消息发送失败重试 6](#_Toc368738026)

[2.9 消息消费失败重试 6](#_Toc368738027)

[2.10 主动Pull方式消费 6](#_Toc368738028)

[3 Broker使用指南 6](#_Toc368738029)

[3.1 Broker集群搭建 6](#_Toc368738030)

[3.2 Broker主备部署 6](#_Toc368738031)

[3.3 Broker重启对客户端的影响 6](#_Toc368738032)

[3.4 Broker配置参数 6](#_Toc368738033)

[4 Name Server使用指南 6](#_Toc368738034)

[5 mqadmin管理工具 6](#_Toc368738035)

[6 常见异常处理方式 6](#_Toc368738036)

[6.1 fastjson版本冲突问题 6](#_Toc368738037)

[6.2 单机只能启动一个进程的问题 7](#_Toc368738038)

# 前言

本文档旨在描述RocketMQ如何使用，以及服务器集群的部署方式，面向应用方和运维人员。

# 客户端使用指南

## 客户端如何寻址

RocketMQ有多种配置方式可以令客户端找到Name Server, 然后通过Name Server再找到Broker，分别如下，优先级由高到低，高优先级会覆盖低优先级。

1. 代码中指定Name Server地址

|  |
| --- |
| producer.setNamesrvAddr("192.168.0.1:9876;192.168.0.2:9876");  或  consumer.setNamesrvAddr("192.168.0.1:9876;192.168.0.2:9876"); |

1. Java启动参数中指定Name Server地址

|  |
| --- |
| -Drocketmq.namesrv.addr=192.168.0.1:9876;192.168.0.2:9876 |

1. 环境变量指定Name Server地址

|  |
| --- |
| export NAMESRV\_ADDR=192.168.0.1:9876;192.168.0.2:9876 |

1. HTTP静态服务器寻址（默认）

客户端启动后，会定时访问一个静态HTTP服务器，地址如下：

http://jmenv.tbsite.net:8080/rocketmq/nsaddr

这个URL的返回内容如下

|  |
| --- |
| 192.168.0.1:9876;192.168.0.2:9876 |

客户端默认每隔2分钟访问一次这个HTTP服务器，并更新本地的Name Server地址。

URL已经在代码中写死，可通过修改/etc/hosts文件来改变要访问的服务器，例如在/etc/hosts增加如下配置

|  |
| --- |
| 10.232.22.67 jmenv.taobao.net |

推荐使用HTTP静态服务器寻址方式，好处是客户端部署简单，且Name Server集群可以热升级。

## 自定义客户端行为

### 客户端API形式

DefaultMQProducer、TransactionMQProducer、DefaultMQPushConsumer、DefaultMQPullConsumer都继承于ClientConfig类，ClientConfig为客户端的公共配置类。

客户端的配置都是get、set形式，每个参数都可以用spring来配置，也可以在代码中配置，例如namesrvAddr这个参数可以这样配置，其他参数同理。

|  |
| --- |
| producer.setNamesrvAddr("192.168.0.1:9876"); |

### 客户端的公共配置

|  |  |  |
| --- | --- | --- |
| 参数名 | 默认值 | 说明 |
| namesrvAddr |  | Name Server地址列表，多个NameServer地址用分号隔开 |
| clientIP | 本机IP | 客户端本机IP地址，某些机器会发生无法识别客户端IP地址情况，需要应用在代码中强制指定 |
| instanceName | DEFAULT | 客户端实例名称，客户端创建的多个Producer、Consumer实际是共用一个内部实例（这个实例包含网络连接、线程资源等） |
| clientCallbackExecutorThreads | 4 | 通信层异步回调线程数 |
| pollNameServerInteval | 30000 | 轮询Name Server间隔时间，单位毫秒 |
| heartbeatBrokerInterval | 30000 | 向Broker发送心跳间隔时间，单位毫秒 |
| persistConsumerOffsetInterval | 5000 | 持久化Consumer消费进度间隔时间，单位毫秒 |

### Producer配置

|  |  |  |
| --- | --- | --- |
| 参数名 | 默认值 | 说明 |
| producerGroup | DEFAULT\_PRODUCER | Producer组名，多个Producer如果属于一个应用，发送同样的消息，则应该将它们归为同一组 |
| createTopicKey | TBW102 | 在发送消息时，自动创建服务器不存在的topic，需要指定Key。 |
| defaultTopicQueueNums | 4 | 在发送消息时，自动创建服务器不存在的topic，默认创建的队列数 |
| sendMsgTimeout | 10000 | 发送消息超时时间，单位毫秒 |
| compressMsgBodyOverHowmuch | 4096 | 消息Body超过多大开始压缩（Consumer收到消息会自动解压缩），单位字节 |
| retryAnotherBrokerWhenNotStoreOK | FALSE | 如果发送消息返回sendResult，但是sendStatus!=SEND\_OK，是否重试发送 |
| maxMessageSize | 131072 | 客户端限制的消息大小，超过报错，同时服务端也会限制 |
| transactionCheckListener |  | 事务消息回查监听器，如果发送事务消息，必须设置 |
| checkThreadPoolMinSize | 1 | Broker回查Producer事务状态时，线程池大小 |
| checkThreadPoolMaxSize | 1 | Broker回查Producer事务状态时，线程池大小 |
| checkRequestHoldMax | 2000 | Broker回查Producer事务状态时，Producer本地缓冲请求队列大小 |

### PushConsumer配置

|  |  |  |
| --- | --- | --- |
| 参数名 | 默认值 | 说明 |
| consumerGroup | DEFAULT\_CONSUMER | Consumer组名，多个Consumer如果属于一个应用，订阅同样的消息，且消费逻辑一致，则应该将它们归为同一组 |
| messageModel | CLUSTERING | 消息模型，支持以下两种 1、集群消费 2、广播消费 |
| consumeFromWhere | CONSUME\_FROM\_LAST\_OFFSET | Consumer启动后，默认从什么位置开始消费 |
| allocateMessageQueueStrategy | AllocateMessageQueueAveragely | Rebalance算法实现策略 |
| subscription | {} | 订阅关系 |
| messageListener |  | 消息监听器 |
| offsetStore |  | 消费进度存储 |
| consumeThreadMin | 10 | 消费线程池数量 |
| consumeThreadMax | 20 | 消费线程池数量 |
| consumeConcurrentlyMaxSpan | 2000 | 单队列并行消费允许的最大跨度 |
| pullThresholdForQueue | 1000 | 拉消息本地队列缓存消息最大数 |
| pullInterval | 0 | 拉消息间隔，由于是长轮询，所以为0，但是如果应用为了流控，也可以设置大于0的值，单位毫秒 |
| consumeMessageBatchMaxSize | 1 | 批量消费，一次消费多少条消息 |
| pullBatchSize | 32 | 批量拉消息，一次最多拉多少条 |

### PullConsumer配置

|  |  |  |
| --- | --- | --- |
| 参数名 | 默认值 | 说明 |
| consumerGroup | DEFAULT\_CONSUMER | Consumer组名，多个Consumer如果属于一个应用，订阅同样的消息，且消费逻辑一致，则应该将它们归为同一组 |
| brokerSuspendMaxTimeMillis | 20000 | 长轮询，Consumer拉消息请求在Broker挂起最长时间，单位毫秒 |
| consumerTimeoutMillisWhenSuspend | 30000 | 长轮询，Consumer拉消息请求在Broker挂起超过指定时间，客户端认为超时，单位毫秒 |
| consumerPullTimeoutMillis | 10000 | 非长轮询，拉消息超时时间，单位毫秒 |
| messageModel | BROADCASTING | 消息模型，支持以下两种 1、集群消费 2、广播消费 |
| messageQueueListener |  | 监听队列变化 |
| offsetStore |  | 消费进度存储 |
| registerTopics | [] | 注册的topic集合 |
| allocateMessageQueueStrategy | AllocateMessageQueueAveragely | Rebalance算法实现策略 |

## Message数据结构

### 针对Producer

|  |  |  |
| --- | --- | --- |
| 字段名 | 默认值 | 说明 |
| Topic | null | 必填，线下环境不需要申请，线上环境需要申请后才能使用 |
| Body | null | 必填，二进制形式，序列化由应用决定，Producer与Consumer要协商好序列化形式。 |
| Tags | null | 选填，类似于Gmail为每封邮件设置的标签，方便服务器过滤使用。目前只支持每个消息设置一个tag，所以也可以类比为Notify的MessageType概念 |
| Keys | null | 选填，代表这条消息的业务关键词，服务器会根据keys创建哈希索引，设置后，可以在Console系统根据Topic、Keys来查询消息，由于是哈希索引，请尽可能保证key唯一，例如订单号，商品Id等。 |
| Flag | 0 | 选填，完全由应用来设置，RocketMQ不做干预 |
| DelayTimeLevel | 0 | 选填，消息延时级别，0表示不延时，大于0会延时特定的时间才会被消费 |
| WaitStoreMsgOK | TRUE | 选填，表示消息是否在服务器落盘后才返回应答。 |

Message数据结构各个字段都可以通过get、set方式访问，例如访问topic

msg.getTopic();

msg.setTopic("TopicTest");

其他字段访问方式类似。

### 针对Consumer

在Producer端，使用com.alibaba.rocketmq.common.message.Message这个数据结构，由于Broker会为Message增加数据结构，所以消息到达Consumer后，会在Message基础之上增加多个字段，Consumer看到的是com.alibaba.rocketmq.common.message.MessageExt这个数据结构，MessageExt继承于Message，MessageExt多出来的数据字段如下表所述。

## 收发消息例子

## 发送顺序消息

## 顺序消费与乱序消费

## 集群消费与广播消费

## 消息发送失败重试

## 消息消费失败重试

## 主动Pull方式消费

# Broker使用指南

## Broker集群搭建

## Broker主备部署

## Broker重启对客户端的影响

## Broker配置参数

# Name Server使用指南

# mqadmin管理工具

# 常见异常处理方式

## fastjson版本冲突问题

## 单机只能启动一个进程的问题