|  |
| --- |
|  |
| **RocketMQ用户指南** |
| v3.0.4 |
|  |
| ©Alibaba 淘宝消息中间件项目组 |
| 2013/12/4 |

**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 | 增加Broker配置 | 誓嘉vintage.wang@gmail.com | 2013/12/2 |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |

**目录**

[1 前言 1](#_Toc373917856)

[2 客户端使用指南 1](#_Toc373917857)

[2.1 客户端如何寻址 1](#_Toc373917858)

[2.2 自定义客户端行为 2](#_Toc373917859)

[2.2.1 客户端API形式 2](#_Toc373917860)

[2.2.2 客户端的公共配置 2](#_Toc373917861)

[2.2.3 Producer配置 2](#_Toc373917862)

[2.2.4 PushConsumer配置 3](#_Toc373917863)

[2.2.5 PullConsumer配置 4](#_Toc373917864)

[2.3 Message数据结构 4](#_Toc373917865)

[2.3.1 针对Producer 4](#_Toc373917866)

[2.3.2 针对Consumer 5](#_Toc373917867)

[2.4 收发消息例子 6](#_Toc373917868)

[2.5 发送顺序消息 6](#_Toc373917869)

[2.6 顺序消费与乱序消费 6](#_Toc373917870)

[2.7 集群消费与广播消费 6](#_Toc373917871)

[2.8 消息发送失败重试 6](#_Toc373917872)

[2.9 消息消费失败重试 6](#_Toc373917873)

[2.10 主动Pull方式消费 6](#_Toc373917874)

[3 Broker使用指南 6](#_Toc373917875)

[3.1 Broker配置参数 6](#_Toc373917876)

[3.2 Broker集群搭建 8](#_Toc373917877)

[3.3 Broker重启对客户端的影响 10](#_Toc373917878)

[4 Name Server使用指南 11](#_Toc373917879)

[5 mqadmin管理工具 11](#_Toc373917880)

[6 常见异常处理方式 11](#_Toc373917881)

[6.1 fastjson版本冲突问题 11](#_Toc373917882)

[6.2 单机只能启动一个进程的问题 11](#_Toc373917883)

# 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配置参数

**获取Broker的默认配置**

|  |
| --- |
| sh mqbroker -m |

**Broker启动时，如何加载配置**

|  |
| --- |
| ### 第一步生成Broker默认配置模版  sh mqbroker -m > broker.p  ### 第二步修改配置文件, broker.p  ### 第三步加载修改过的配置文件  nohup sh mqbroker -c broker.p |

**Broker运行过程中，动态改变Broker的配置，注意，并非所有配置项都支持动态变更**

|  |
| --- |
| ### 修改地址为192.168.1.100:10911的Broker消息保存时间为24小时  sh mqadmin updateBrokerConfig -b 192.168.1.100:10911 -k fileReservedTime -v 24 |

|  |  |  |
| --- | --- | --- |
| 字段名 | 默认值 | 说明 |
| listenPort | 10911 | Broker对外服务的监听端口 |
| namesrvAddr | null | Name Server地址 |
| brokerIP1 | 本机IP | 本机IP地址，默认系统自动识别，但是某些多网卡机器会存在识别错误的情况，这种情况下可以人工配置 |
| brokerName | 本机主机名 |  |
| brokerClusterName | DefaultCluster | Broker所属哪个集群 |
| brokerId | 0 | BrokerId，必须是大等于0的整数，0表示Master，>0表示Slave，一个Master可以挂多个Slave，Master与Slave通过BrokerName来配对 |
| autoCreateTopicEnable | TRUE | 是否允许Broker自动创建Topic，建议线下开启，线上关闭 |
| autoCreateSubscriptionGroup | TRUE | 是否允许Broker自动创建订阅组，建议线下开启，线上关闭 |
| rejectTransactionMessage | FALSE | 是否拒绝事务消息接入 |
| fetchNamesrvAddrByAddressServer | FALSE | 是否从web服务器获取Name Server地址，针对大规模的Broker集群建议使用这种方式 |
| storePathCommitLog | $HOME/store/commitlog | commitLog存储路径 |
| storePathConsumeQueue | $HOME/store/consumequeue | 消费队列存储路径 |
| storePathIndex | $HOME/store/index | 消息索引存储路径 |
| storeCheckpoint | $HOME/store/checkpoint | checkpoint文件存储路径 |
| abortFile | $HOME/store/abort | abort文件存储路径 |
| deleteWhen | 4 | 删除文件时间点，默认凌晨4点 |
| fileReservedTime | 48 | 文件保留时间，默认48小时 |
| maxTransferBytesOnMessageInMemory | 262144 | 单次Pull消息（内存）传输的最大字节数 |
| maxTransferCountOnMessageInMemory | 32 | 单次Pull消息（内存）传输的最大条数 |
| maxTransferBytesOnMessageInDisk | 65536 | 单次Pull消息（磁盘）传输的最大字节数 |
| maxTransferCountOnMessageInDisk | 8 | 单次Pull消息（磁盘）传输的最大条数 |
| messageIndexEnable | TRUE | 是否开启消息索引功能 |
| messageIndexSafe | FALSE | 是否提供安全的消息索引机制，索引保证不丢 |
| haMasterAddress |  | 在Slave上直接设置Master地址，默认从Name Server上自动获取，也可以手工强制配置 |
| brokerRole | ASYNC\_MASTER | Broker的角色 - ASYNC\_MASTER 异步复制Master - SYNC\_MASTER 同步双写Master - SLAVE |
| flushDiskType | ASYNC\_FLUSH | 刷盘方式 - ASYNC\_FLUSH 异步刷盘 - SYNC\_FLUSH 同步刷盘 |
| cleanFileForciblyEnable | TRUE | 磁盘满、且无过期文件情况下 TRUE 表示强制删除文件，优先保证服务可用 FALSE 标记服务不可用，文件不删除 |

## Broker集群搭建

推荐的几种Broker集群部署方式，这里的Slave不可写，但可读，类似于Mysql主备方式。

1. **单个Master**

这种方式风险较大，一旦Broker重启或者宕机时，会导致整个服务不可用，不建议线上环境使用

1. **多Master模式**

一个集群无Slave，全是Master，例如2个Master或者3个Master

优点：配置简单，单个Master宕机或重启维护对应用无影响，在磁盘配置为RAID10时，即使机器宕机不可恢复情况下，由于RAID10磁盘非常可靠，消息也不会丢（异步刷盘丢失少量消息，同步刷盘一条不丢）。性能最高。

缺点：单台机器宕机期间，这台机器上未被消费的消息在机器恢复之前不可订阅，消息实时性会受到受到影响。

|  |
| --- |
| ### 先启动Name Server，例如机器IP为：192.168.1.1:9876  nohup sh mqnamesrv &  ### 在机器A，启动第一个Master  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-noslave/broker-a.properties &  ### 在机器B，启动第二个Master  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-noslave/broker-b.properties & |

1. **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。

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，因为Master宕机后，消费者仍然可以从Slave消费，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，会丢失少量消息。

|  |
| --- |
| ### 先启动Name Server，例如机器IP为：192.168.1.1:9876  nohup sh mqnamesrv &  ### 在机器A，启动第一个Master  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-2s-async/broker-a.properties &  ### 在机器B，启动第二个Master  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-2s-async/broker-b.properties &  ### 在机器C，启动第一个Slave  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-2s-async/broker-a-s.properties &  ### 在机器D，启动第二个Slave  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-2s-async/broker-b-s.properties & |

1. **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，主备都写成功，向应用返回成功。

优点：数据与服务都无单点，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。目前主宕机后，备机不能自动切换为主机，后续会支持自动切换功能。

|  |
| --- |
| ### 先启动Name Server，例如机器IP为：192.168.1.1:9876  nohup sh mqnamesrv &  ### 在机器A，启动第一个Master  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-2s-sync/broker-a.properties &  ### 在机器B，启动第二个Master  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-2s-sync/broker-b.properties &  ### 在机器C，启动第一个Slave  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-2s-sync/broker-a-s.properties &  ### 在机器D，启动第二个Slave  nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ\_HOME/conf/2m-2s-sync/broker-b-s.properties & |

**以上Broker与Slave配对是通过指定相同的brokerName参数来配对，Master的BrokerId必须是0，Slave的BrokerId必须是大于0的数。另外一个Master下面可以挂载多个Slave，同一Master下的多个Slave通过指定不同的BrokerId来区分。**

**$ROCKETMQ\_HOST指的RocketMQ安装目录，需要用户自己设置此环境变量。**

## Broker重启对客户端的影响

Broker重启可能会导致正在发往这台机器的的消息发送失败，RocketMQ提供了一种优雅关闭Broker的方法，通过执行以下命令会清除Broker的写权限，过40s后，所有客户端都会更新Broker路由信息，此时再关闭Broker就不会发生发送消息失败的情况，因为所有消息都发往了其他Broker。

sh mqadmin wipeWritePerm -b brokerName -n namesrvAddr

# 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