RocketMQ 用户指南

v3.0.0

2013/12/3

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

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。

2 客户端使用指南

2.1 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。

一、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");
```

二、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
```

三、环境变量指定 Name Server 地址

```
export NAMESRV_ADDR=192.168.0.1:9876;192.168.0.2:9876
```

四、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。

2.2 自定义客户端行为

2.2.1 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");
```

2.2.2 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

2.2.3 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

2.2.4 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

2.2.5 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

2.3 Message 数据结构

2.3.1 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。

2.3.2 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。

2.4 收发消息例子

2.5 发送顺序消息

2.6 顺序消费与乱序消费

2.7 集群消费与广播消费

2.8 消息发送失败重试

2.9 消息消费失败重试

2.10 主动 Pull 方式消费

3 Broker 使用指南

3.1 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
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

3.2 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

2. **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
```

3. 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
```

4. 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。

3.3 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

4 Name Server 使用指南

5 mqadmin 管理工具

6 常见异常处理方式

6.1 fastjson 版本冲突问题

6.2 单机只能启动一个进程的问题