运维管理

1 集群搭建

1.1 单Master模式

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,可以用于本地测试。

1）启动 NameServer

```
### 首先启动Name Server
$ nohup sh mqnamesrv &

### 验证Name Server 是否启动成功
$ tail -f ~/logs/rocketmqlogs/namesrv.log
The Name Server boot success...
```

2）启动 Broker

```
### 启动Broker
$ nohup sh bin/mqbroker -n localhost:9876 &

### 验证Name Server 是否启动成功，例如Broker的IP为：192.168.1.2，且名称为broker-a
$ tail -f ~/logs/rocketmqlogs/Broker.log
The broker[broker-a, 192.169.1.2:10911] boot success...
```

1.2 多Master模式

一个集群无Slave，全是Master，例如2个Master或者3个Master，这种模式的优缺点如下：

- 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，消息实时性会受到影响。

1）启动NameServer

NameServer需要先于Broker启动，且如果在生产环境使用，为了保证高可用，建议一般规模的集群启动3个NameServer，各节点的启动命令相同，如下：

```
### 首先启动Name Server
$ nohup sh mqnamesrv &
```

```
3.
4. ### 验证Name Server 是否启动成功
5. $ tail -f ~/logs/rocketmqlogs/namesrv.log
6. The Name Server boot success...
```

2）启动Broker集群

```
1. ### 在机器A，启动第一个Master，例如NameServer的IP为：192.168.1.1
2. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-noslave/broker-
   a.properties &
3.
4. ### 在机器B，启动第二个Master，例如NameServer的IP为：192.168.1.1
5. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-noslave/broker-
   b.properties &
6.
7. ...
```

如上启动命令是在单个NameServer情况下使用的。对于多个NameServer的集群，Broker启动命令中 `-n` 后面的地址列表用分号隔开即可，例如 `192.168.1.1:9876;192.161.2:9876` 。

1.3 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），这种模式的优缺点如下：

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

1）启动NameServer

```
1. ### 首先启动Name Server
2. $ nohup sh mqnamesrv &
3.
4. ### 验证Name Server 是否启动成功
5. $ tail -f ~/logs/rocketmqlogs/namesrv.log
6. The Name Server boot success...
```

2）启动Broker集群

```
1. ### 在机器A，启动第一个Master，例如NameServer的IP为：192.168.1.1
2. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-2s-
   async/broker-a.properties &
3.
4. ### 在机器B，启动第二个Master，例如NameServer的IP为：192.168.1.1
5. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-2s-
   async/broker-b.properties &
```

```
6.
7. ### 在机器C，启动第一个Slave，例如NameServer的IP为：192.168.1.1
8. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-2s-
   async/broker-a-s.properties &
9.
10. ### 在机器D，启动第二个Slave，例如NameServer的IP为：192.168.1.1
11. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-2s-
   async/broker-b-s.properties &
```

1.4 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，即只有主备都写成功，才向应用返回成功，这种模式的优缺点如下：

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

1）启动NameServer

```
1. ### 首先启动Name Server
2. $ nohup sh mqnamesrv &
3.
4. ### 验证Name Server 是否启动成功
5. $ tail -f ~/logs/rocketmqlogs/namesrv.log
6. The Name Server boot success...
```

2）启动Broker集群

```
1. ### 在机器A，启动第一个Master，例如NameServer的IP为：192.168.1.1
2. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-2s-sync/broker-
   a.properties &
3.
4. ### 在机器B，启动第二个Master，例如NameServer的IP为：192.168.1.1
5. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-2s-sync/broker-
   b.properties &
6.
7. ### 在机器C，启动第一个Slave，例如NameServer的IP为：192.168.1.1
8. $ nohup sh mqbroker -n 192.168.1.1:9876 -c $ROCKETMQ_HOME/conf/2m-2s-sync/broker-
   a-s.properties &
9.
10. ### 在机器D，启动第二个Slave，例如NameServer的IP为：192.168.1.1
11. $ 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_HOME指的RocketMQ安装目录，需要用户自己设置此环境变量。

2 mqadmin管理工具

> *注意：*
>
> 1. *执行命令方法：* `./mqadmin {command} {args}`
> 2. *几乎所有命令都需要配置-n表示NameServer地址，格式为ip:port*
> 3. *几乎所有命令都可以通过-h获取帮助*
> 4. *如果既有Broker地址（-b）配置项又有clusterName（-c）配置项，则优先以Broker地址执行命令，如果不配置Broker地址，则对集群中所有主机执行命令，只支持一个Broker地址。-b格式为ip:port，port默认是10911*
> 5. *在tools下可以看到很多命令，但并不是所有命令都能使用，只有在MQAdminStartup中初始化的命令才能使用，你也可以修改这个类，增加或自定义命令*
> 6. *由于版本更新问题，少部分命令可能未及时更新，遇到错误请直接阅读相关命令源码*

2.1 Topic相关

名称	含义	命令选项
		-b
		-c
		-h-

updateTopic	创建更新Topic配置	-n
		-p
		-r
		-w
		-t
deleteTopic	删除Topic	-c
		-h
		-n
		-t
topicList	查看 Topic 列表信息	-h
		-c
		-n
topicRoute	查看 Topic 路由信息	-t
		-h
		-n

topicStatus	查看 Topic 消息队列offset	-t -h -n
topicClusterList	查看 Topic 所在集群列表	-t -h -n
updateTopicPerm	更新 Topic 读写权限	-t -h -n -b -p -c
updateOrderConf	从NameServer上创建、删除、 获取特定命名空间的kv配置，目前还未启用	-h -n -t -v -m

		-t
allocateMQ	以平均负载算法计算消费者列表负载消息队列的负载结果	-h -n -i
statsAll	打印Topic订阅关系、TPS、积累量、 24h读写总量等信息	-h -n -a -t

2.2 集群相关

名称	含义	命令选项	说明
clusterList	查看集群信息，集群、BrokerName、BrokerId、TPS等信息	-m	打印更多信息 (增加打印出如下信息 #InTotalYest, #OutTotalYest, #InTotalToday ,#OutTotalToday)

		-h	打印帮助
		-n	NameServer 服务地址，格式 ip:port
		-i	打印间隔，单位秒
clusterRT	发送消息检测集群各Broker RT。消息发往 ${BrokerName} Topic。	-a	amount，每次探测的总数，RT = 总时间 / amount
		-s	消息大小，单位B
		-c	探测哪个集群
		-p	是否打印格式化日志，以|分割，默认不打印
		-h	打印帮助
		-m	所属机房，打印使用
		-i	发送间隔，单位秒
		-n	NameServer 服务地址，格式 ip:port

2.3 Broker相关

名称	含义	命令选项	说明
updateBrokerConfig	更新 Broker 配置文件，会修改Broker.conf	-b	Broker 地址，格式为ip:port
		-c	cluster 名称
		-k	key 值
		-v	value 值
		-h	打印帮助
		-n	NameServer服务地址，格式ip:port
brokerStatus	查看 Broker 统计信息、运行状态（你想要的信息几乎都在里面）	-b	Broker 地址，地址为ip:port
		-h	打印帮助
		-n	NameServer服务地址，格式ip:port
brokerConsumeStats	Broker中各个消费者的消费情况，按Message Queue维度返回Consume Offset，Broker Offset，Diff，TImestamp等信息	-b	Broker 地址，地址为ip:port
		-t	请求超时时间
		-l	diff阈值，超过阈值才打印
		-o	是否为顺序topic，一般为false
		-h	打印帮助
		-n	NameServer服务地址，格式ip:port
getBrokerConfig	获取Broker配置	-b	Broker 地址，地址为ip:port
		-n	NameServer服务地址，格式ip:port
wipeWritePerm	从NameServer上清除Broker写权限	-b	Broker 地址，地址为ip:port
		-n	NameServer服务地址，格式ip:port

		-h	打印帮助
cleanExpiredCQ	清理Broker上过期的ConsumeQueue， 如果手动减少对列数可能产生过期队列	-n	NameServer服务地址，格式ip:port
		-h	打印帮助
		-b	Broker 地址，地址为ip:port
		-c	集群名称
cleanUnusedTopic	清理Broker上不使用的Topic， 从内存中释放Topic的ConsumeQueue， 如果手动删除Topic会产生不使用的Topic	-n	NameServer服务地址，格式ip:port
		-h	打印帮助
		-b	Broker 地址，地址为ip:port
		-c	集群名称
sendMsgStatus	向Broker发消息， 返回发送状态和RT	-n	NameServer服务地址，格式ip:port
		-h	打印帮助
		-b	BrokerName，注意不同于Brok地址
		-s	消息大小，单位I
		-c	发送次数

2.4 消息相关

名称	含义	命令选项	说明
queryMsgById	根据offsetMsgId查询msg， 如果使用开源控制台， 应使用offsetMsgId， 此命令还有其他参数， 具体作用请阅读QueryMsgByIdSubCommand。	-i	msgId
		-h	打印帮助
		-n	NameServer 格式 ip:port
queryMsgByKey	根据消息 Key 查询消息	-k	msgKey
		-t	Topic 名称
		-h	打印帮助
		-n	NameServer 格式 ip:port
queryMsgByOffset	根据 Offset 查询消息	-b	Broker 名称， （这里需要注 填写的是 Brok 不是 Broker 的 Broker 名称可 clusterList 查
		-i	query 队列 id
		-o	offset 值
		-t	topic 名称
		-h	打印帮助

		-n	打印帮助
		-n	NameServer 格式 ip:port
queryMsgByUniqueKey	根据msgId查询， msgId不同于offsetMsgId， 区别详见常见运维问题。-g，-d配合使用， 查到消息后尝试让特定的消费者消费消息并返回消费结果	-h	打印帮助
		-n	NameServer 格式 ip:port
		-i	uniqe msg id
		-g	consumerGrc
		-d	clientId
		-t	topic名称
checkMsgSendRT	检测向topic发消息的RT， 功能类似clusterRT	-h	打印帮助
		-n	NameServer 格式 ip:port
		-t	topic名称
		-a	探测次数
		-s	消息大小
sendMessage	发送一条消息， 可以根据配置发往特定Message Queue，或普通发送。	-h	打印帮助
		-n	NameServer 格式 ip:port
		-t	topic名称
		-p	body，消息体
		-k	keys
		-c	tags
		-b	BrokerName
		-i	queueId
consumeMessage	消费消息。可以根据offset、开始&结束时间戳、 消息队列消费消息， 配置不同执行不同消费逻辑， 详见ConsumeMessageComma	-h	打印帮助
		-n	NameServer 格式 ip:port
		-t	topic名称
		-b	BrokerName
		-o	从offset开始消
		-i	queueId

	开发consumeMessageCommand。	-g	消费者分组
		-s	开始时间戳，格
		-d	结束时间戳
		-c	消费多少条消息
printMsg	从Broker消费消息并打印，可选时间段	-h	打印帮助
		-n	NameServer 地址，格式 ip:port
		-t	topic名称
		-c	字符集，例如UTF-8
		-s	subExpress，过滤表达式
		-b	开始时间戳，格式
		-e	结束时间戳
		-d	是否打印消息体
printMsgByQueue	类似printMsg，但指定MessageQueue	-h	打印帮助
		-n	NameServer 地址，格式 ip:port
		-t	topic名称
		-i	queueId
		-a	BrokerName
		-c	字符集，例如UTF-8
		-s	subExpress，过滤表达式
		-b	开始时间戳，格式
		-e	结束时间戳
		-p	是否打印消息
		-d	是否打印消息体
		-f	是否统计tag数量
		-h	打印帮助
		-n	NameServer 地址，格式 ip:port
		-g	消费者分组

resetOffsetByTime	按时间戳重置offset， Broker和consumer都会重置	-g	消费者所在
		-t	topic名称
		-s	重置为此时间戳
		-f	是否强制重置， 只支持回溯off 如果true， 不管时间戳对应 sumeOffset关
		-c	是否重置c++客

2.5 消费者、消费组相关

名称	含义	命令选项	说明
consumerProgress	查看订阅组消费状态， 可以查看具体的client IP的消息积累量	-g	消费者所属组名
		-s	是否打印client IP
		-h	打印帮助
		-n	NameServer 服务地
consumerStatus	查看消费者状态， 包括同一个分组中是否都是相同 的订阅，分析Process Queue是否堆积， 返回消费者jstack结果， 内容较多，	-h	打印帮助
		-n	NameServer 服务地
		-g	consumer group

	使用者参见ConsumerStatusSubCommand	g	consumer group
		-i	clientId
		-s	是否执行jstack
updateSubGroup	更新或创建订阅关系	-n	NameServer 服务地
		-h	打印帮助
		-b	Broker地址
		-c	集群名称
		-g	消费者分组名称
		-s	分组是否允许消费
		-m	是否从最小offset开
		-d	是否是广播模式
		-q	重试队列数量
		-r	最大重试次数
		-i	当slaveReadEnable 且还未达到从slave消 费，可以配置备机id
		-w	如果Broker建议从sl 配置决定从哪个slav 例如1
		-a	当消费者数量变化时
deleteSubGroup	从Broker删除订阅关系	-n	NameServer 服务地
		-h	打印帮助
		-b	Broker地址
		-c	集群名称

		-c	集群名称
		-g	消费者分组名称
cloneGroupOffset	在目标群组中使用源群组的offset	-n	NameServer 服务地
		-h	打印帮助
		-s	源消费者组
		-d	目标消费者组
		-t	topic名称
		-o	暂未使用

2.6 连接相关

名称	含义	命令选项	说明
consumerConnection	查询 Consumer 的网络连接	-g	消费者所属组名
		-n	NameServer 服务地址，格式 ip:port
		-h	打印帮助
producerConnection	查询 Producer 的网络连接	-g	生产者所属组名
		-t	主题名称
		-n	NameServer 服务地址，格式 ip:port
		-h	打印帮助

2.7 NameServer相关

名称	含义	命令选项	说明
updateKvConfig	更新NameServer的kv配置，目前还未使用	-s	命名空间
		-k	key
		-v	value
		-n	NameServer服务地址，格式 ip:port
		-h	打印帮助
deleteKvConfig	删除NameServer的kv配置	-s	命名空间
		-k	key
		-n	NameServer服务地址，格式 ip:port
		-h	打印帮助
getNamesrvConfig	获取NameServer配置	-n	NameServer服务地址，格式 ip:port
		-h	打印帮助
updateNamesrvConfig	修改NameServer配置	-n	NameServer服务地址，格式 ip:port
		-h	打印帮助
		-k	key

		-v	value

2.8 其他

名称	含义	命令选项	说明
startMonitoring	开启监控进程， 监控消息误删、 重试队列消息数等	-n	NameServer 服务地址，格式 ip:port
		-h	打印帮助

3 运维常见问题

3.1 RocketMQ的mqadmin命令报错问题

> *问题描述：有时候在部署完RocketMQ集群后，尝试执行“mqadmin”一些运维命令，会出现下面的异常信息：*
>
> ```
> 1. org.apache.rocketmq.remoting.exception.RemotingConnectException: connect to
>    <null> failed
> ```

解决方法：可以在部署RocketMQ集群的虚拟机上执行 `export NAMESRV_ADDR=ip:9876` （ip指的是集群中部署NameServer组件的机器ip地址）命令之后再使用“mqadmin”的相关命令进行查询，即可得到结果。

3.2 RocketMQ生产端和消费端版本不一致导致不能正常消费的问题

> *问题描述：同一个生产端发出消息，A消费端可消费，B消费端却无法消费，rocketMQ Console中出现：*
>
> ```
> 1. Not found the consumer group consume stats, because return offset table is
>    empty, maybe the consumer not consume any message的异常消息。
> ```

解决方案：RocketMQ 的jar包：rocketmq-client等包应该保持生产端，消费端使用相同的version。

3.3 新增一个topic的消费组时，无法消费历史消息的问题

> *问题描述：当同一个topic的新增消费组启动时，消费的消息是当前的offset的消息，并未获取历史消息。*

解决方案：rocketmq默认策略是从消息队列尾部，即跳过历史消息。如果想消费历史消息，则需要设置：`org.apache.rocketmq.client.consumer.DefaultMQPushConsumer#setConsumeFromWhere`。常用的有以下三种配置：

- 默认配置,一个新的订阅组第一次启动从队列的最后位置开始消费，后续再启动接着上次消费的进度开始消费,即跳过历史消息；

```
1. consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_LAST_OFFSET);
```

- 一个新的订阅组第一次启动从队列的最前位置开始消费，后续再启动接着上次消费的进度开始消费,即消费Broker未过期的历史消息；

```
1. consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_FIRST_OFFSET);
```

- 一个新的订阅组第一次启动从指定时间点开始消费，后续再启动接着上次消费的进度开始消费，和consumer.setConsumeTimestamp()配合使用，默认是半个小时以前；

```
1. consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_TIMESTAMP);
```

3.4 如何开启从Slave读数据功能

在某些情况下，Consumer需要将消费位点重置到1-2天前，这时在内存有限的Master Broker上，CommitLog会承载比较重的IO压力，影响到该Broker的其它消息的读与写。可以开启`slaveReadEnable=true`，当Master Broker发现Consumer的消费位点与CommitLog的最新值的差值的容量超过该机器内存的百分比（`accessMessageInMemoryMaxRatio=40%`），会推荐Consumer从Slave Broker中去读取数据，降低Master Broker的IO。

3.5 性能调优问题

异步刷盘建议使用自旋锁，同步刷盘建议使用重入锁，调整Broker配置项`useReentrantLockWhenPutMessage`，默认为false；异步刷盘建议开

启 `TransientStorePoolEnable` ；建议关闭transferMsgByHeap，提高拉消息效率；同步刷盘建议适当增大 `sendMessageThreadPoolNums` ，具体配置需要经过压测。

3.6 在RocketMQ中msgId和offsetMsgId的含义与区别

使用RocketMQ完成生产者客户端消息发送后，通常会看到如下日志打印信息：

```
SendResult [sendStatus=SEND_OK, msgId=0A42333A0DC818B4AAC246C290FD0000,
offsetMsgId=0A42333A00002A9F000000000134F1F5, messageQueue=MessageQueue
[topic=topicTest1, BrokerName=mac.local, queueId=3], queueOffset=4]
```

- msgId，对于客户端来说msgId是由客户端producer实例端生成的，具体来说，调用方法 `MessageClientIDSetter.createUniqIDBuffer()` 生成唯一的Id；
- offsetMsgId，offsetMsgId是由Broker服务端在写入消息时生成的（采用“IP地址+Port端口”与“CommitLog的物理偏移量地址”做了一个字符串拼接），其中offsetMsgId就是在RocketMQ控制台直接输入查询的那个messageId。