高鹏 姜洋

ROCKETMQ消息中间件

目录

MQ特点

- 先进先出
- 发布订阅
- 持久化
- 分布式

MQ使用场景

- 系统解耦
- 消除流量峰值
- 提高应用响应速度
- 消息分发
- 分布式事务(有些mq不支持)
- 以上所有都需要考虑是否接受异步

常用MQ

	Rocketmq	IBM WebSphere MQ	Kafka	RabbitMQ
所属社区/公司	Alibaba	IBM	Apache	Mozilla Public License
特点	支持分布式扩展、主备高可靠；支持海量订阅；丰富的消息订阅模式；高可用且性能很好	负责在两个系统之间传递消息。具有强大的跨平台性，它支持的平台数多达 35 种，保证对消息的"Once and Once only"的传输，做到不丢失、不复传	为追求吞吐量而设计，性能很好。但 Broker 并不保存消费状态，可靠性较差	实现了 AMQP 协议，支持丰富的路由模式；消息可靠性较高；Broker 部署复杂
开发语言	Java	?	Scala	Erlang
单机 TPS 性能	~=110,000/sec	~=20,000/sec	~=170,000/sec	~=60,000/sec
单机最大队列数	<10 万	<1 千	<1 千	<1 千
集群	支持，对应用透明的集群式实现	支持，通过在仓储库队列管理器设置集群队列来实现，部署较复杂	支持，但集群对发送者并不透明，发送者需要确认消息发送到哪个分区	支持，对应用透明的集群式实现
负载均衡	支持	支持	支持	支持
订阅形式	点对点、广播、基于 topic/messageTag 的发布订阅、按照消息类型或属性进行正则匹配订阅。 订阅形式灵活丰富	点对点、广播、基于队列或 topic 的发布订阅、按照消息属性正则匹配订阅	基于 topic 的发布订阅	提供了四种 Exchange：fanout,direct,topic,header
事务	支持	支持	不支持	支持
可靠性	高	高	一般	高
消息重试	支持	支持	不支持	支持
顺序消息	支持	支持	支持	不支持
定时消息	支持延时、定时消息	支持延时消息	不支持	不支持
消息加密	不支持	SSL/TLS	SSL	SSL
管理控制台	命令行或 web 页管理，web 页包括管理、查询、统计、监控运维等功能	命令行或 MQ explorer 管理。但是 MQ explorer 需要额外安装	命令行或 Kafka Manager web 页管理	命令行或 web 页管理
消息轨迹查询	可查询每条消息完整的链路，包括消费情况	不支持	不支持	不支持
消息回溯	支持指定时间点的消息回溯	不支持	只支持队列 offset 回溯	不支持

rocketmq vs kafka

产品	Topic数量	发送端并发数	发送端RT（ms）	发送端TPS	消费端TPS
Rocket MQ 3.4.6	64	800	8	9w	8.6w
	128	800	9	7.8w	7.7w
	256	800	10	7.5w	7.5w
Kafka 0.8.2	64	800	5	13.6w	13.6w
	128	256	23	8500	8500
	256	256	133	2215	2352

CPU	24核
内存	94G
硬盘	Seagate Constellation ES (SATA 6Gb/s) 2,000,398,934,016 bytes [2.00 TB] 7202 rpm
网卡	1000Mb/s

压力端	Jmeter的java客户端
消息大小	128字节
并发数	能达到服务端最大TPS的最优并发
Topic分区数量	8
刷盘策略	异步落盘

- https://yq.aliyun.com/articles/25379
- Kafka,topic少的时tps高因为Producer端将多个小消息合并,批量发向Broker,但是不可靠;topic多的时候tps下降因为,每个topic一个partition,topic多时写数据时磁盘更加不连续
- RocketMQ,也支持批量发送,如果使用,吞吐量比上图还要高,所有消息(即使不同topic)连续存储,顺序写磁盘,增加topic对写无影响
- 更多:
- https://blog.csdn.net/damacheng/article/details/42846549(其中1点问题:rcketmq已经支持事务)

专业术语

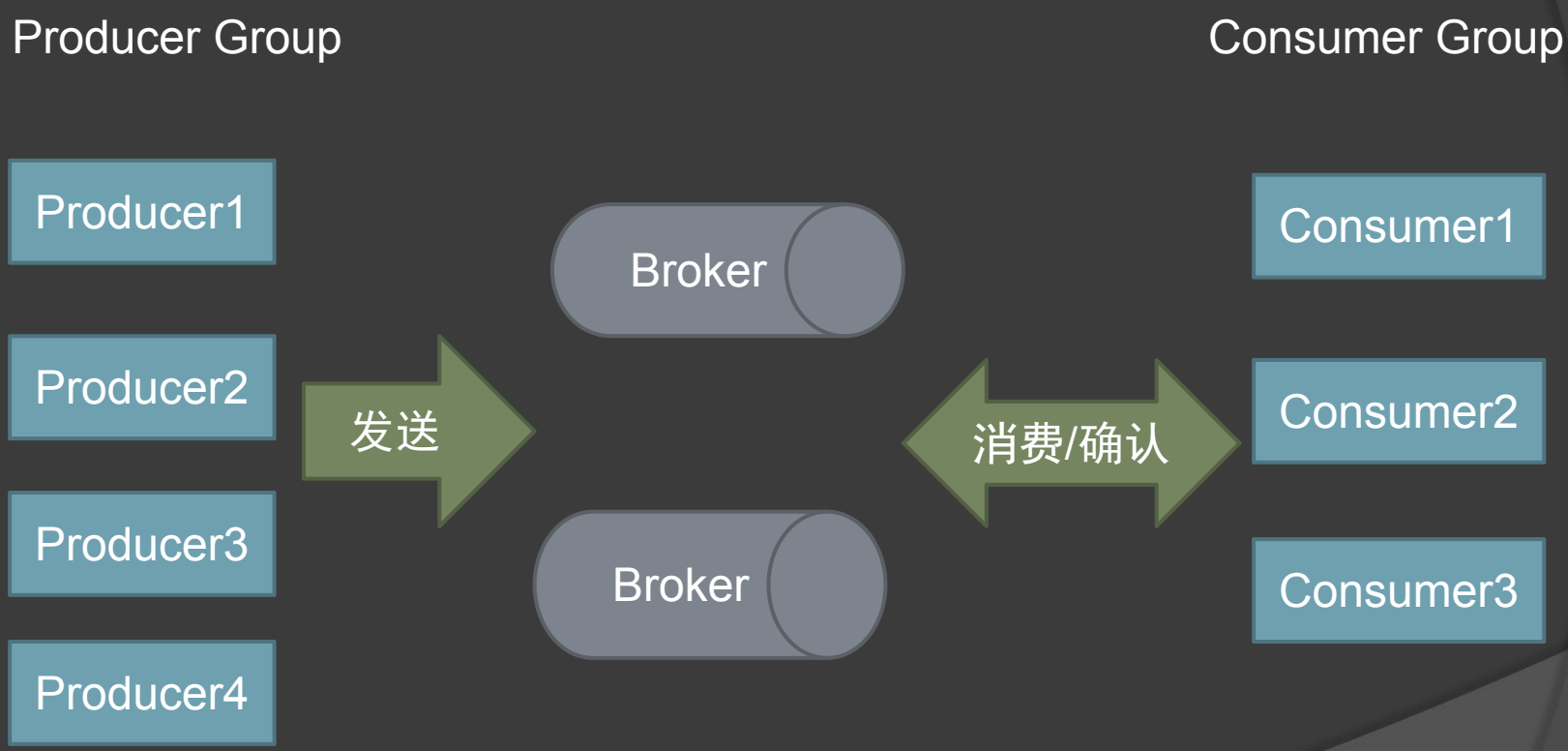
- Producer
  - 消息生产者，负责产生消息，一般由业务系统负责产生消息。
- Consumer
  - 消息消费者，负责消费消息，一般是后台系统负责异步消费。
  - Push Consumer，Consumer的一种，应用通常向Consumer对象注册一个Listener接口，一旦收到消息，Consumer对象立 刻回调Listener接口方法
  - Pull Consumer，Consumer的一种，应用通常主动调用Consumer的拉消息方法从Broker拉消息，主动权由应用控制
- Producer Group
  - 一类Producer的集合名称，可理解为同一项目多个节点组成的集群，一类Producer可发送多种topic消息，每种topic下的消息发送逻辑相同
- Consumer Group
  - 一类Consumer的集合名称，可理解为同一项目多个节点组成的集群，一类Consumer通常可消费多种topic消息，每种topic的消息消费逻辑相同

专业术语

- topic
  - 一类消息的集合，同一类消息往往具有相同消息格式
- Broker
  - 消息中转角色，负责存储消息，转发消息，一般也称为Server。在JMS规范中称为Provider。
- Message Queue
  - 每个消息一个元素，元素按照先进先出的顺序放在Message Queue中，一个broker下的一个topic往往有多个Message Queue

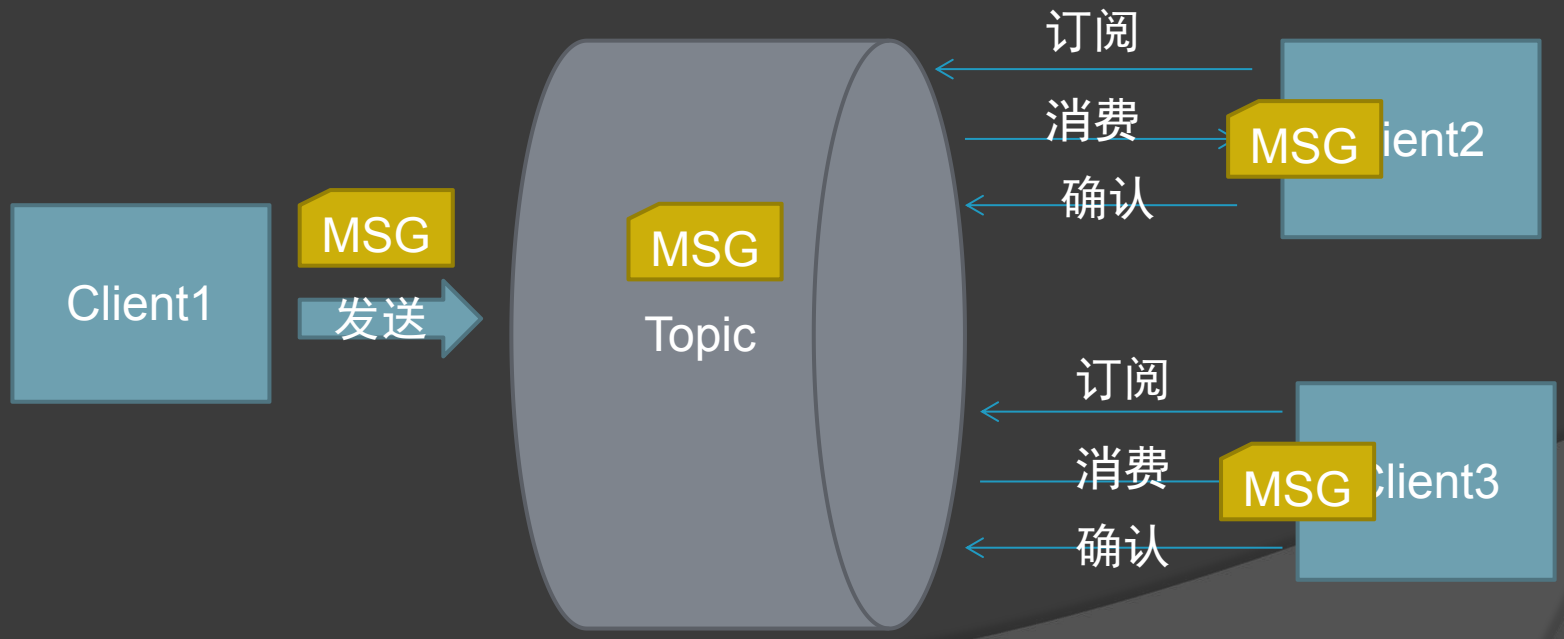
简单系统结构

集群结构

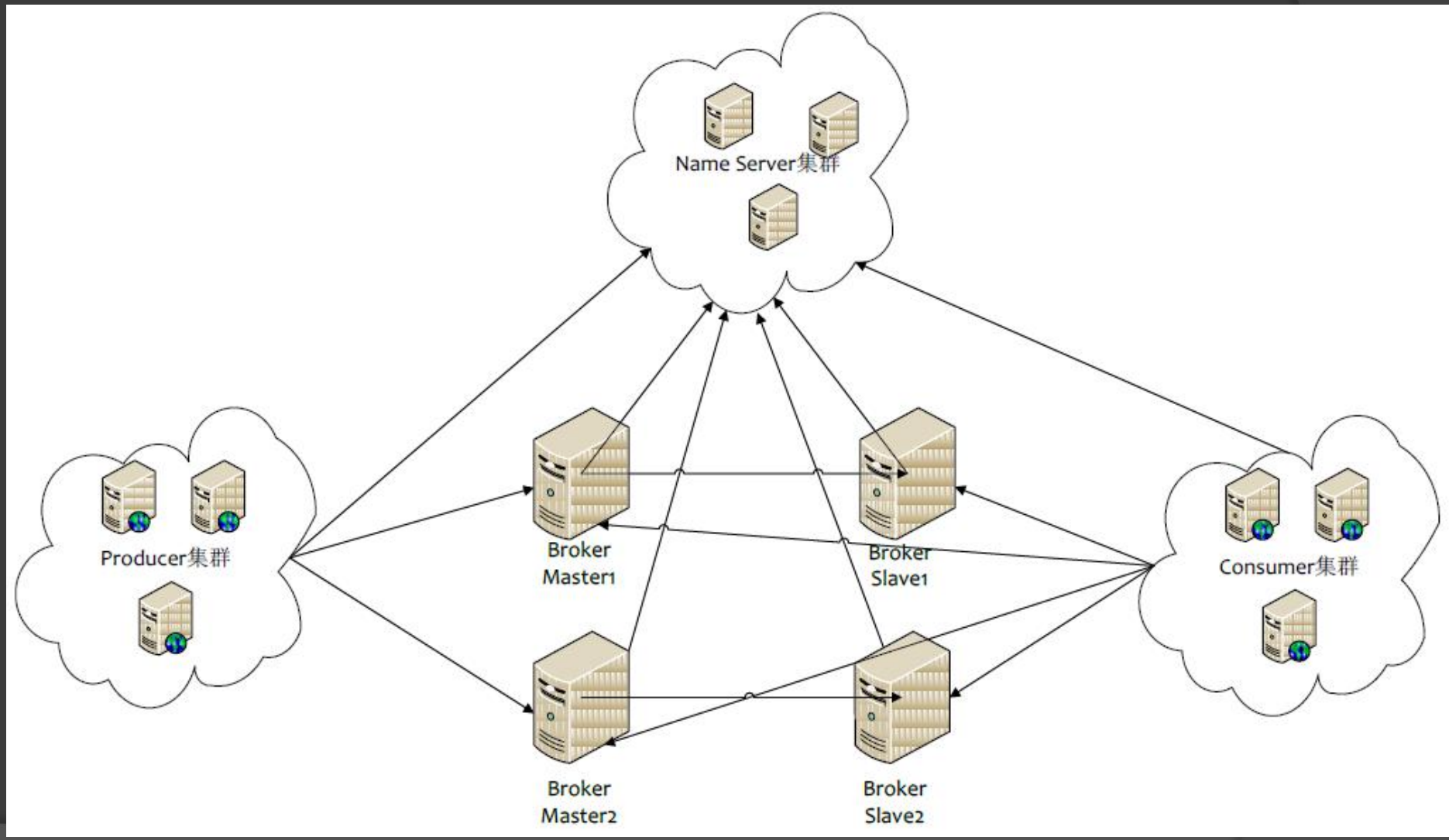

Producer Group
Consumer Group
Producer1
Producer2
Producer3
Producer4
发送
Broker
Broker
消费/确认
Consumer1
Consumer2
Consumer3

消息模型

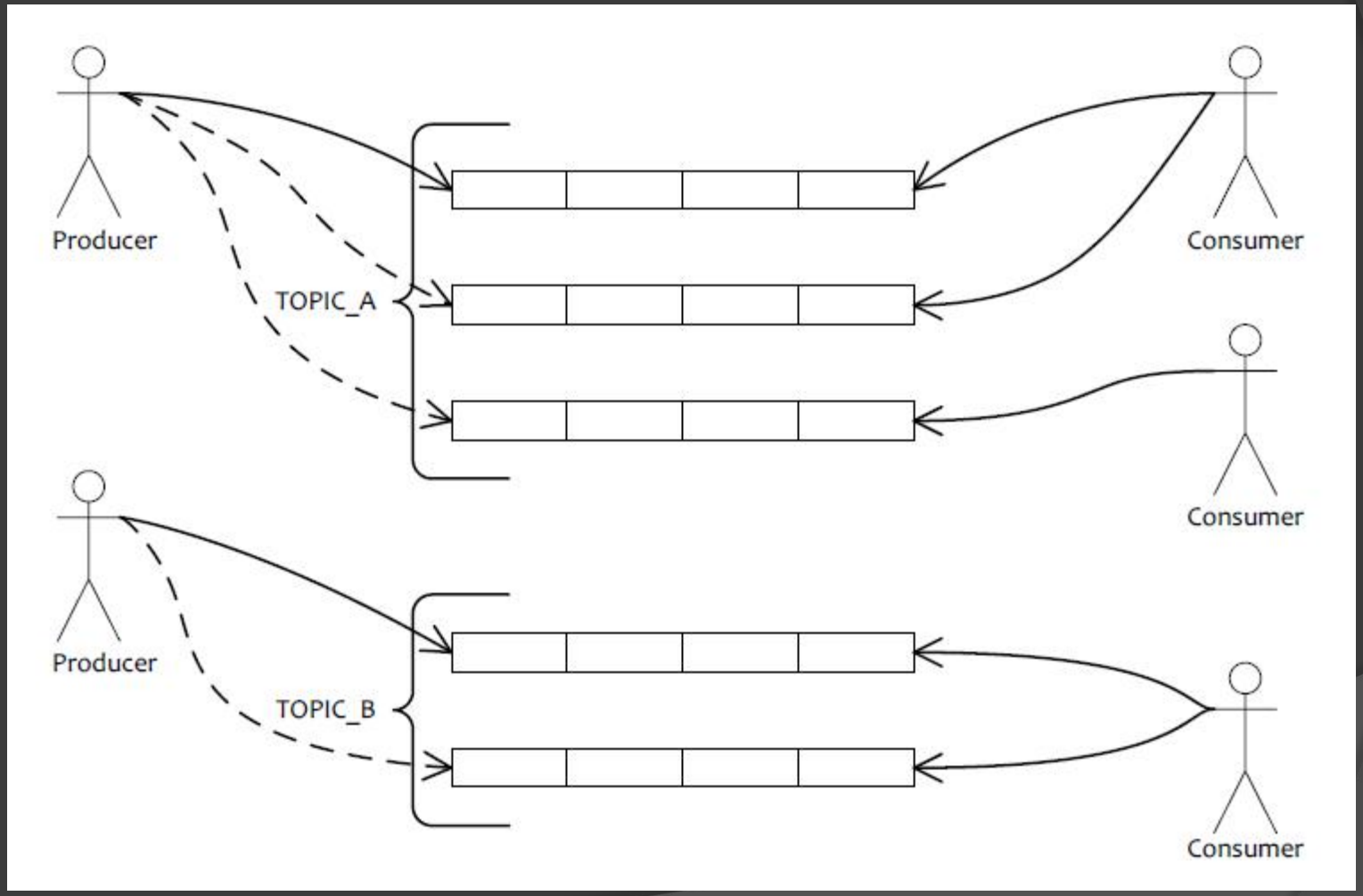
- Publish/Subscribe（发送/订阅）
  - 每个消息可以有多个订阅者
  - 客户端只有订阅后才能接收消息



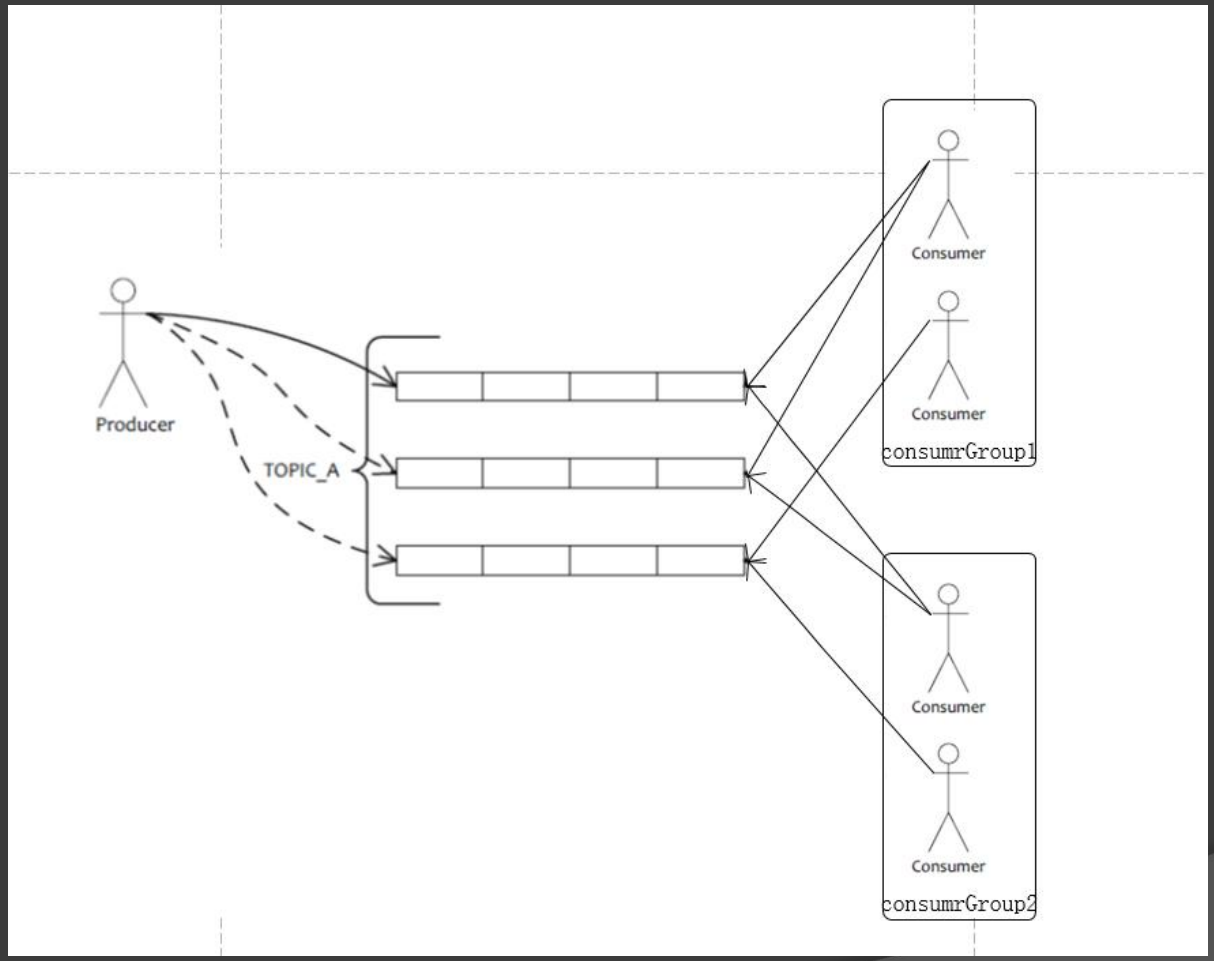
RocketMQ部署结构



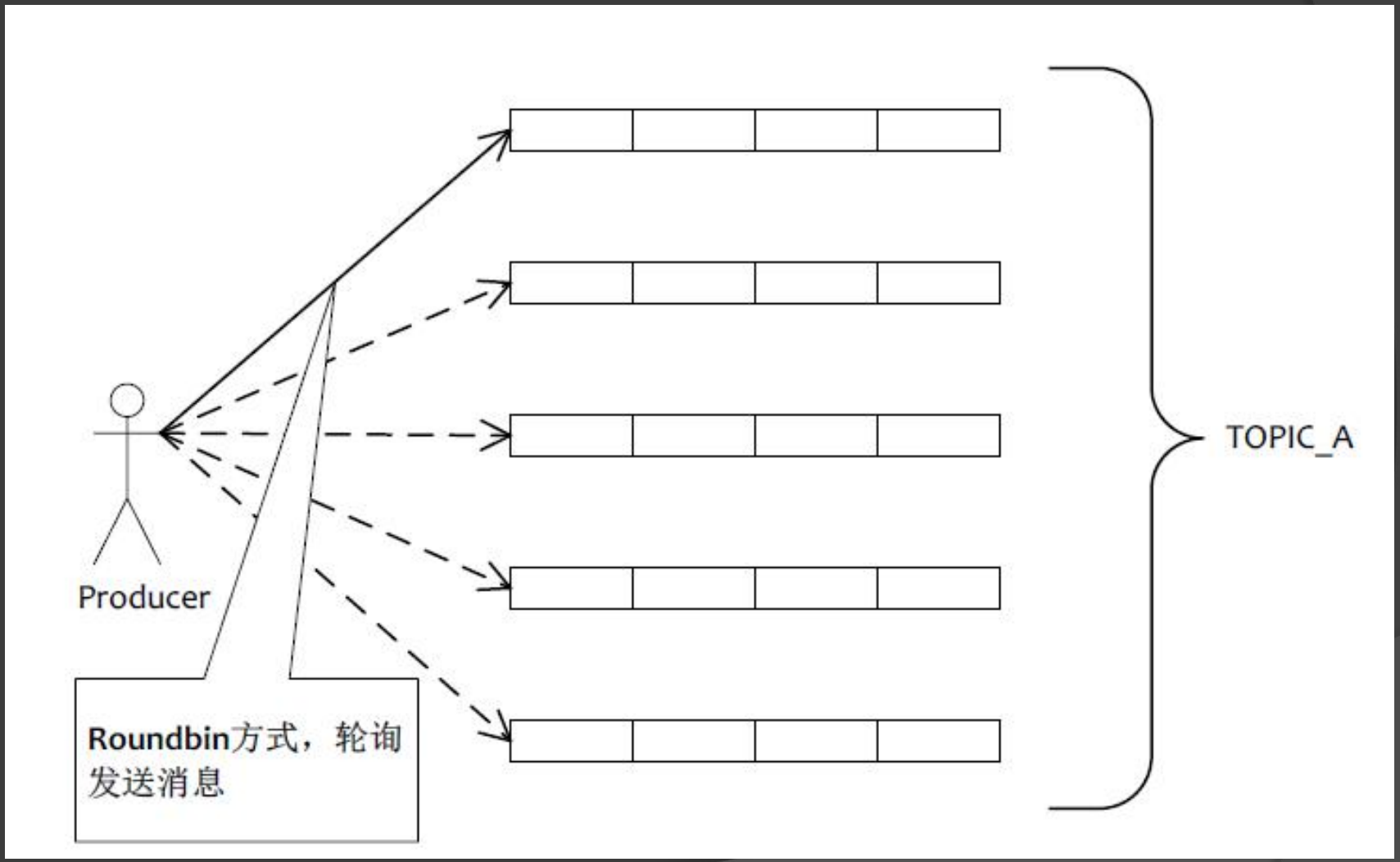
队列模型


Producer
TOPIC_A
Consumer
Consumer
Producer
TOPIC_B
Consumer

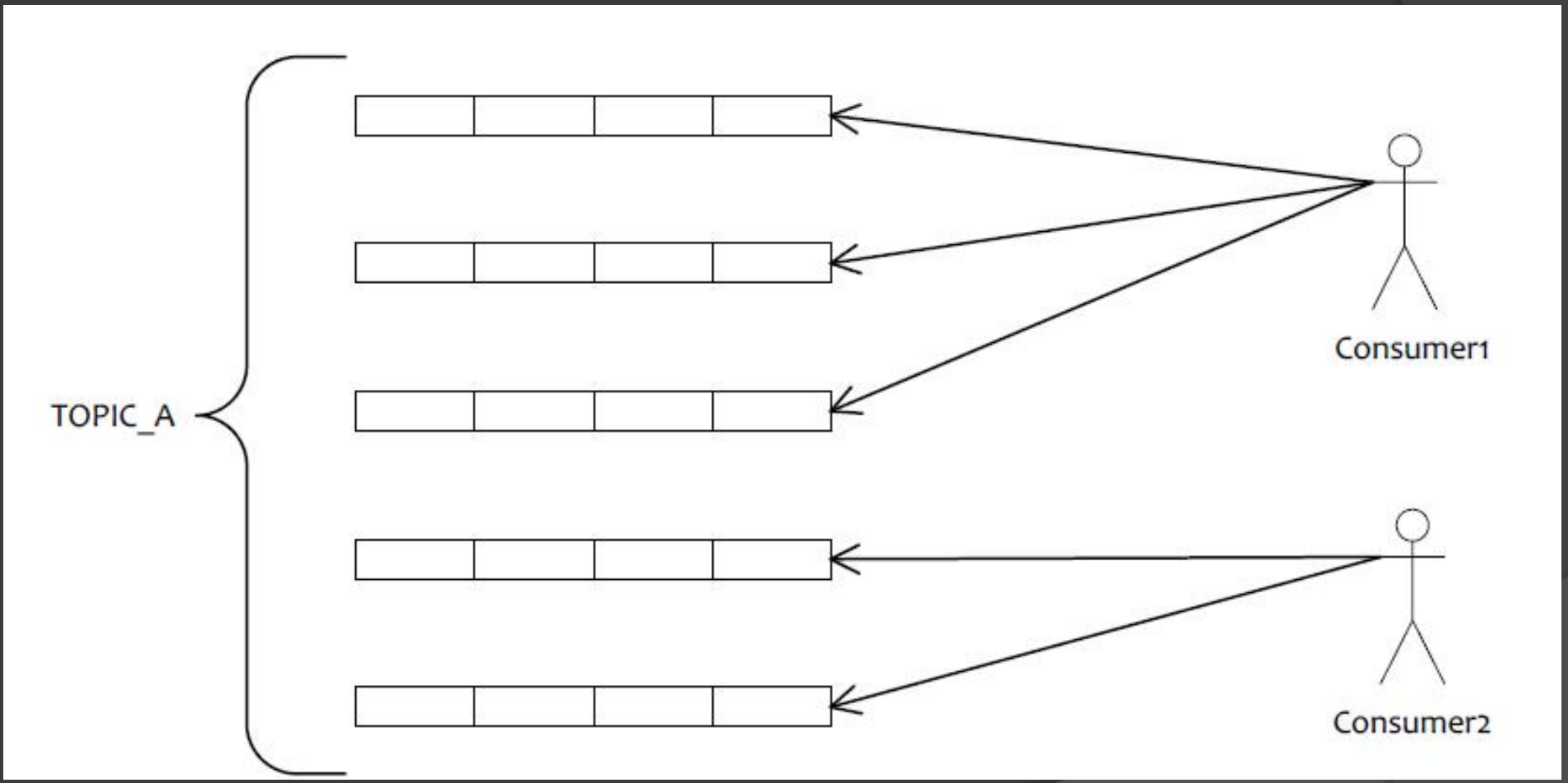
队列模型


Producer
TOPIC_A
Consumer
Consumer
consumrGroup1
Consumer
Consumer
consumrGroup2

发送负载均衡



消费负载均衡



消费负载均衡

队列数量	Consumer 数量	Rebalance 结果
5	2	C1: 3 C2: 2
6	3	C1: 2 C2: 2 C3: 2
10	20	C1~C10: 1 C11~C20: 0 无法消费
20	6	C1: 4 C2: 4 c3~C6: 3

消息质量（QOS）级别

- Exactly-once
  - 投递且仅投递一次
- At-least-once
  - 最少投递一次，可能有重复消息
- At-most-once
  - 最多投递一次，有可能丢消息

XA分布式事务

- 优点
  - 多资源之间的ACID
  - 编程模型简单统一
- 缺点
  - 性能和可用性不高
  - 故障难于恢复

事务消息


Client
发送prepare消息
执行本
地事务
发送commit消息
发送rollback消息
Broker
Client
发送check消息
发送commit消息
发送rollback消息

顺序消息

- 场景

  一笔订单产生了 3 条消息，分别是订单创建、订单付款、订单完成。消费时，要按照顺序依次消费才有意义。

- rocketmq实现

  发送时程序指定发送到哪一个queue

  消费时即可顺序消费

  问题：顺序消费失败代价比较大，前面的消息消费失败，后面的消息堆积

消息结构

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。

发送消息注意事项

- 一个应用尽可能用一个Topic，消息子类型用tags来标识，tags可以由应用自由设置。只有发送消息设置了tags，消费方在订阅消息时，才可以利用tags在broker做消息过滤。

- 每个消息在业务层面的唯一标识码，要设置到keys字段，方便将来定位消息丢失问题。服务器会为每个消息创建索引（哈希索引），应用可以通过topic，key来查询这条消息内容，以及消息被谁消费。由于是哈希索引，请务必保证key尽可能唯一，这样可以避免潜在的哈希冲突

- 消息发送成功或者失败，要打印消息日志，务必要打印sendresult和key字段，尤其是SendResult中的msgId
- 对于消息不可丢失应用，务必要有消息重发机制 。例如如果消息发送失败，存储到数据库，能有定时程序尝试重发，或者人工触发重发。
- 对于性能要求高，而对消息可靠性要求低的，可以通过sendOneway进行发送，如果需要处理结果的，可以通过异步发送接口异步处理发送结果

消费端注意事项

- 由于不能丢消息，所以选择At-least-once
- 有可能产生重复消息
- 消费端自己处理幂等

  可以通过数据库唯一键做到完全幂等

  也可以通过缓存等其它组件做到伪幂等
- 打印消费消息日志，利于排查问题
- 错误处理
  - 业务逻辑错误，例如某个用户的个人信息不符合业务逻辑，调用一个外部的rpc接口超时这种小规模的失败，可以直接让broker稍后投递
  - Db不可用或者外部资源短时间内不可用的，可以消费端进行sleep，减少broker的重投压力

消费失败重试

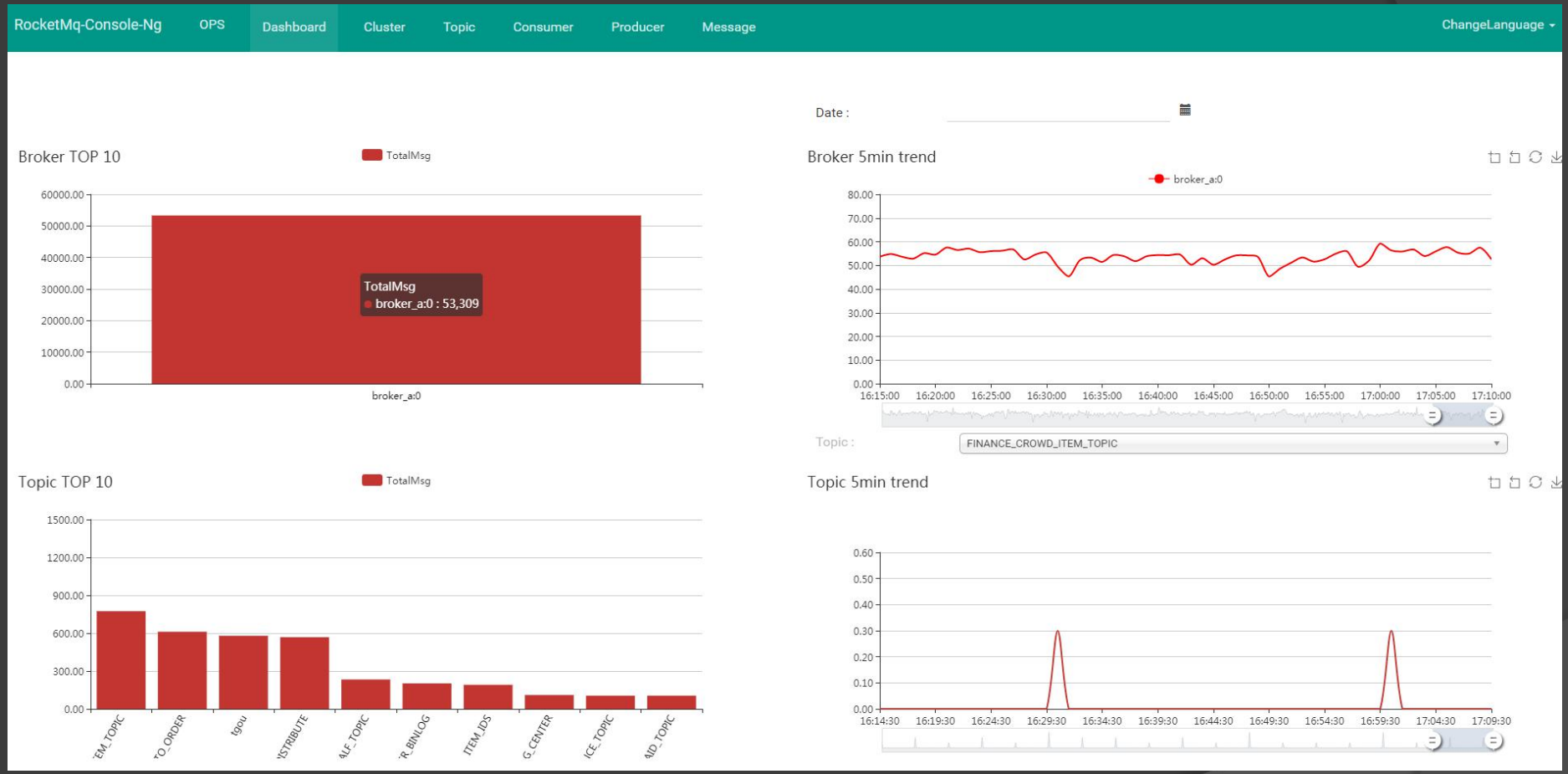
- 1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h
- 第1次失败1s后重试
- 第2次失败5s后重试
- 第3次失败10s后重试
- ……

监控控制台

- 查看集群
- 更新topic配置
- 查看生产者、消费者连接
- 更新name server配置
- 查询消息
- 配置broker
- 更新offset消费位点，重新消费
- 更改订阅关系配置，查询消费进度
- 测试环境地址

  http://mq-console.test.66buy.com.cn/#/

监控界面


RocketMq-Console-Ng
OPS
Dashboard
Cluster
Topic
Consumer
Producer
Message
ChangeLanguage
Date :
Broker TOP 10
TotalMsg
60000.00
50000.00
40000.00
30000.00
20000.00
10000.00
0.00
TotalMsg
broker_a:0 : 53,309
broker_a:0
Broker 5min trend
broker_a:0
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
16:15:00
16:20:00
16:25:00
16:30:00
16:35:00
16:40:00
16:45:00
16:50:00
16:55:00
17:00:00
17:05:00
17:10:00
Topic :
FINANCE_CROWD_ITEM_TOPIC
Topic TOP 10
TotalMsg
1500.00
1200.00
900.00
600.00
300.00
0.00
Topic 5min trend
0.60
0.50
0.40
0.30
0.20
0.10
0.00
16:14:30
16:19:30
16:24:30
16:29:30
16:34:30
16:39:30
16:44:30
16:49:30
16:54:30
16:59:30
17:04:30
17:09:30

重要配置

- Name Server的地址
  - 格式为”192.168.1.1:9876;192.168.1.2:9876”
- Producer Group
  - 发送端应用集群的所有机器配置的，要全局唯一，跟其它应用区别
- Consumer Group
  - 消费端应用集群所有机器配置的，要全局唯一，跟其它应用区别，要不然会出现消息投递问题
- dataId
  - 与Name Server地址选填一个，dataId在apollo中配置了name Server地址

升级4.4.0

- 新增功能：发送多tag消息

producer发消息

```
SimpleData simpleData = new SimpleData( name: "1",  password: "1");
List<String> tags = new ArrayList<>();
tags.add("test_tag2");
tags.add("test_tag1");
AbstractMessage message = new AbstractMessage<SimpleData>( topic: "DiwayouTest", tags, simpleData) {
};
```

consumer如下配置：两个consumer_group分别订阅test_tag1和test_tag2，这样这两个consumer可分别处理各自的逻辑

```
<entry key="DiwayouTest">                                    consumer_group1
    <map>
        <entry key="test_tag1" value-ref="haSimpleMessageHandler"/>
    </map>
</entry>
```

```
<entry key="DiwayouTest">                                    consumer_group2
    <map>
        <entry key="test_tag2" value-ref="haSimpleMessageHandler"/>
    </map>
</entry>
```

升级4.4.0

- consumer如下配置：同一个consumer_group的两个handler会处理两次该消息，按照配置顺序处理，先执行emptyHandler，再执行haSimpleMessageHandler，如果多个handler为同一个实例，会做去重处理，去重完仍按照配置的顺序执行，如果其中一个失败则停止执行剩下的handler，等待mq重试，请考虑幂等情况

```
<entry key="DiwayouTest">
    <map>
        <entry key="test_tag1" value-ref="emptyHandler"/>
        <entry key="test_tag2" value-ref="haSimpleMessageHandler"/>
    </map>
</entry>
```

升级4.4.0

- 新增功能：可将发送的消息可以在指定tag下打上多个sub tag

producer发送多subTag消息

```
subTags.add("subTag1");
subTags.add("subTag2");
AbstractMessage message = new AbstractMessage( topic: "DiwayouTest",  tags: "test_tag", subTags, simpleData) {
};
```

consumer如下配置：两个consumer_group分别订阅test_tag这个tag下的子tag：subTag1和subTag2，这样这两个consumer可分别处理各自的逻辑。注意tag（本例中为test_tag）与subTag（本例中为subTag1/subTag2）之间用 > 分割

```
<entry key="DiwayouTest">
    <map>                                consumer_group1
        <entry key="test_tag>subTag1" value-ref="handler"/>
    </map>
</entry>
```

```
<entry key="DiwayouTest">
    <map>                                consumer_group2
        <entry key="test_tag>subTag2" value-ref="handler"/>
    </map>
</entry>
```

升级4.4.0

- consumer如下配置：同一个consumer_group的两个handler会处理两次该消息，按照配置顺序处理，先执行emptyHandler，再执行haSimpleMessageHandler，如果多个handler为同一个实例，会做去重处理，去重完仍按照配置的顺序执行，如果其中一个失败则停止执行剩下的handler，等待mq重试，请考虑幂等情况

```
<entry key="DiwayouTest">
    <map>
        <entry key="test_tag>subTag2" value-ref="emptyHandler"/>
        <entry key="test_tag>subTag1" value-ref="haSimpleMessageHandler"/>
    </map>
</entry>
```

升级4.4.0

- 注意：如果发送了事务消息，本地事务执行失败，则broker会回查producer如果，则消息回查时，回查出大于一个handler（多个不同的handler实例对象），注意此处与consumer端的不同：这时候会报错find more than one handler for topic={},tag={},subscriptions={}。并返回unknow让broker继续回查。
- 举例子1：关于broker回查producer的listener的配置：DefaultTransactionCheckListener的属性topicTagHandlerRoute配置如下图，如果发送了该消息（topic=DiwayouTest,tag=test_tag，subTag={subTag1,subTag2}），则会找到两个handler，报错，并返回unknow让broker继续回查。所以应当只配置一个test_tag项，事务回查处理，不需要配置subTag

```
<entry key="DiwayouTest">
    <map>
        <entry key="test_tag>subTag2" value-ref="transMessageCheckHandler"/>
        <entry key="test_tag>subTag1" value-ref="emptyTransMessageCheckHandler"/>
    </map>
</entry>
```

升级4.4.0

- 举例子2：关于broker回查producer的listener的配置：DefaultTransactionCheckListener的属性topicTagHandlerRoute配置如下图，如果发送了该消息（topic=DiwayouTest,tag={test_tag1,test_tag2}），则会找到两个handler，则报错，并返回unknow让broker继续回查。所以一条消息可能含有多个 主tag的时候，producer的回查listener的配置应注意这几个主tag不可以配置多个不同的handler实例对象

```
<entry key="DiwayouTest">
    <map>
        <entry key="test_tag1" value-ref="transMessageCheckHandler"/>
        <entry key="test_tag2" value-ref="emptyTransMessageCheckHandler"/>
    </map>
</entry>
```

升级4.4.0

- 详情见wiki:

https://c.51tiangou.com/pages/viewpage.action?pageId=27099874

链接

- apache:
  https://github.com/apache/rocketmq
- 天狗rocketmq中间件：
  http://svn.66buy.com.cn:3443/svn/coding/third-jar/rocketmq
- 监控平台测试环境：
  http://mq-console.test.66buy.com.cn/#/

总结

- RocketMQ挺好用的
- 有问题随时找姜洋，姜洋不会找高鹏