# 读书笔记4 应用场景分析
可靠性有限场景

顺序消息：
举例： 订单场景： 订单生成、付款、发货 必须按顺序
全局顺序： 某个Topic夏所有消息都要保证顺讯
部分顺序： 每一组消息被顺序消费即可， 例如订单中，上述3个消息按照顺序消费

## 1. 全局顺序消息
RocketMq 默认不保证顺序，例如创建一个Topic 默认8读队列，8写队列，都是随机的
所以要保证全局顺序
==》Topic 读写队列数设置为1
==〉producer、 consumer并发设置1
退化为单线程
所以大部分都是部分顺序

## 2. 部分顺序消息
需要发送、消费配合处理

发送端： 同一业务ID 发送到同一个Message Queue； =》MessageQueueSelector实现
消费端： 从同一个MessageQueue 读取的信息不被并发处理=》MessageListenerOrderly 实现

发送端实例：

```
for (int i = 0; i < 1000; i++) {
    int orderId = i;
    Message message = new Message( topic: "orderTopic8",  tags: "tags",  keys: "KEY" + i, ("hello rocketmq" + i).getBytes());
    SendResult sendResult = producer.send(message, new MessageQueueSelector() {
        @Override
        public MessageQueue select(List<MessageQueue> mqs, Message msg, Object arg) {
            System.out.println("queue selector mq nums" + mqs.size());
            System.out.println("msg info" + msg.toString());

            for (MessageQueue mq : mqs) {
                log.info(mq.toString());
            }
            Integer id = (Integer) arg;
            int index = id % mqs.size();
            return mqs.get(index);
        }
    }, orderId);
}
```

```
@Override
public SendResult send(Message msg, MessageQueueSelector selector, Object arg)
    throws MQClientException, RemotingException, MQBrokerException, InterruptedException {
    msg.setTopic(withNamespace(msg.getTopic()));
    return this.defaultMQProducerImpl.send(msg, selector, arg);
}

/**
```

消费端实例：

```
consumer.setConsumeThreadMin(); // consumer线程数
consumer.setConsumeThreadMax();
consumer.setPullBatchSize();     // 一次从broker的一个messageQueue获取消息的最大数目（默认32）
consumer.setConsumeMessageBatchMaxSize();  // consumer的executor 一次传入的消息数 List<MessageExt> msgs链表最大长度，默认1)
consumer.registerMessageListener(new MessageListenerOrderly() {
    AtomicLong consumerTimes = new AtomicLong( initialValue: 0);
    @Override
    public ConsumeOrderlyStatus consumeMessage(List<MessageExt> msgs, ConsumeOrderlyContext context) {
        log.info("received new message" + new String( original: msgs.get(0).getBody() + "%n"));
        return ConsumeOrderlyStatus.SUCCESS;
    }
});
```

每个consumer queue加锁，自己不并发消费，其他人不管

## 消息重复消费问题：

Rocketmq确保一定投递成功，消息不丢，不保证重复消费

解决方案：
业务逻辑幂等；
维护一个已消费的记录

动态增减机器问题：

1. **增减NameServer：**
   设置NamerServer的四个优先级：
   代码写死　　：　例如producer中，　producer.setNameSrvAddr()
   java启动参数：　rocketmq.namesrv.addr
   linux环境变量：NAMESRV_ADDR
   HTTP服务设置：　上述3个都没有，会默认向一个地址请求获取nameSrv地址
   通过rocketmq.namesrv.domain　（唯一可以动态增加的）

## 2. 增减 Broker：

增加 Broker 不会对已有 Topic 产生影响 =》 updateTopic 命令
sh ./bin/mqadmin updateTopic -b  newBrokerip:port -t TestTopic -n nowBrokerip:port

减少 Broker 分两种情况：
只有一个 Master broker，要先看是否还有 producer， 停止 broker 前 停止 producer

多 master，要看消息发送方式，
同步发送，DefaultMqProducer 内有重试机制

```
private SendResult sendDefaultImpl(
    Message msg,
    final CommunicationMode communicationMode,
    final SendCallback sendCallback,
    final long timeout
) throws MQClientException, RemotingException, MQBrokerException, InterruptedException {
    this.makeSureStateOK();
    Validators.checkMessage(msg, this.defaultMQProducer);
    final long invokeID = random.nextLong();
    long beginTimestampFirst = System.currentTimeMillis();
    long beginTimestampPrev = beginTimestampFirst;
    long endTimestamp = beginTimestampFirst;
    TopicPublishInfo topicPublishInfo = this.tryToFindTopicPublishInfo(msg.getTopic());
    if (topicPublishInfo != null && topicPublishInfo.ok()) {
        boolean callTimeout = false;
        MessageQueue mq = null;
        Exception exception = null;
        SendResult sendResult = null;
        int timesTotal = communicationMode == CommunicationMode.SYNC ? 1 + this.defaultMQProducer.getRetryTimesWhenSendFailed() : 1;
        int times = 0;
        String[] brokersSent = new String[timesTotal];
        for (; times < timesTotal; times++) {
            String lastBrokerName = null == mq ? null : mq.getBrokerName();
            MessageQueue mqSelected = this.selectOneMessageQueue(topicPublishInfo, lastBrokerName);
            if (mqSelected != null) {
                mq = mqSelected;
                brokersSent[times] = mq.getBrokerName();
                try {
                    beginTimestampPrev = System.currentTimeMillis();
                    if (times > 0) {
                        //Reset topic with namespace during resend.
                        msg.setTopic(this.defaultMQProducer.withNamespace(msg.getTopic()));
                    }
                    long costTime = beginTimestampPrev - beginTimestampFirst;
                    if (timeout < costTime) {
                        callTimeout = true;
                        break;
```

如果是异步方式. / sendOneWay，发送失败不会重试
30s 后 DefaultMqProducer 会拉去 broker 信息，后续不再发送

最好是先关停 producer

## 各种故障对消息的影响：

1、broker 正常关闭、启动
内存数据不丢， 如果重启过程中有 consumer，master 故障后，consumer 会连接

slave
注意启动master机器不要关闭consumer（ 因为恢复后重连 master，停止 consumer 再启动master的话，offset不对，会有大量消息重复读）

如果重启过程有producer ，如上分析，会连接到slave 同步写的消息不丢，其他方式会丢

2、broker 异常crash、然后启动
3、OS crash 、启动
4、机器断电、马上恢复
234属于软件问题，内存可能丢失，根据 刷盘磁盘去看 同步刷盘和第一种情况一致

5、磁盘损坏
6、CPU、主板、内存损坏

硬件故障，原有磁盘数据丢失，如果master 和 slave配置成同步复制，消息不会丢，异步复制的话，两次sync间的消息丢失