【第51话：京东三面当RabbitMQ瞬间来了几万条消息时如何处理】

Hello 小伙伴们，这节课给大家讲解下之前一名学生在京东第三轮面试时被问到的一个问题：“当RabbitMQ突然来了几万条数据怎么办”。

京东总部就是我们校区附近。每天顺着窗口可以看到屹立在经海路地铁站的几座高楼。

我们学生每天学习之余看看京东大楼，可以让自己更加有学习动力。很多毕业的学生，学习结束后就想要去京东试一试。毕竟是离我们最近的大厂。所以京东我们积累了很多面试题。

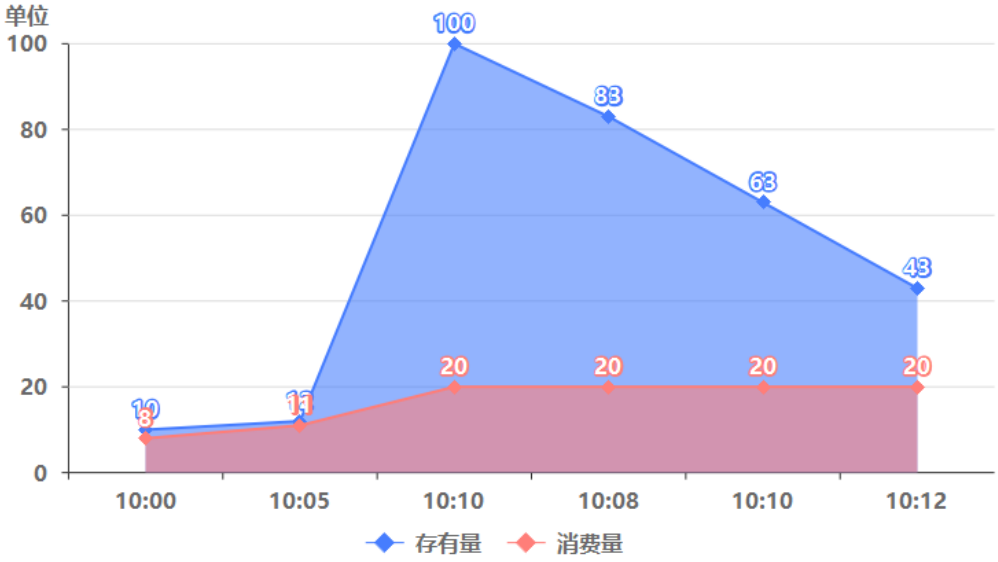
其中一名学生在京东三面的时候，就被问到了：“当RabbitMQ突然来了几万条数据怎么办”。

其实这个问题换一种问法可能就更好回答了“如何防止RabbitMQ消息堆积”。

那么为什么会出现RabbitMQ消息堆积呢？只要消费者消费的速度低于发布者发布消息的速度，经过一段时间积累就会产生消息堆积。或者发布者瞬间发来大量消息也会产生消息堆积。

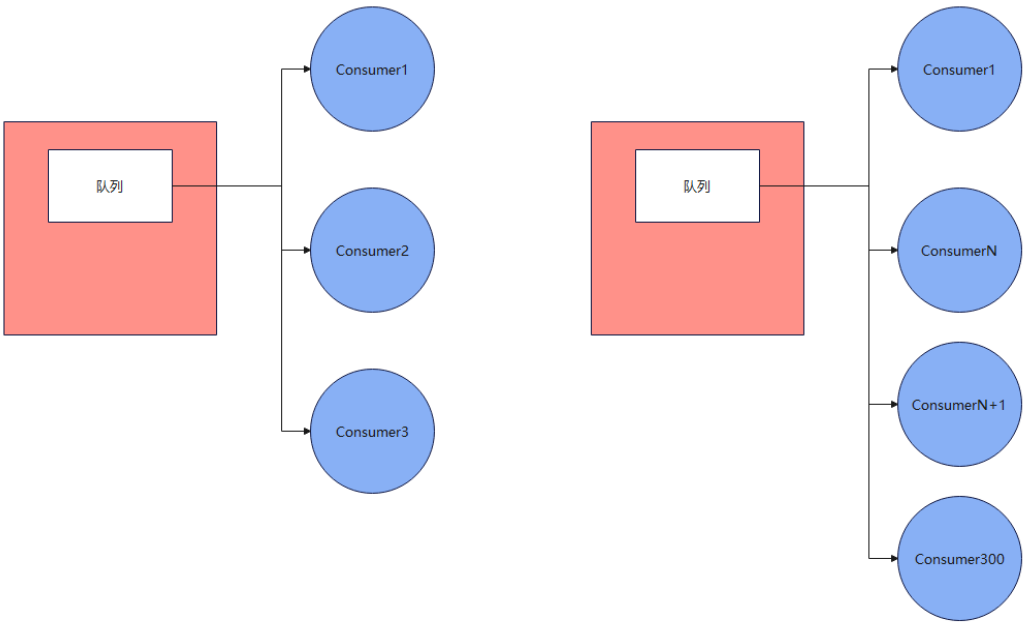
消息堆积一旦出现，可能导致新的消息无法进入队列，丢失消息。系统反应时间变长等问题。所以在使用了RabbitMQ的项目中一定要注意消息堆积的问题。这也是面试官为什么爱问这个问题的原因，毕竟谁也不希望把我们招聘进去了，用到RabbitMQ的业务经常出现卡顿、响应时间变长等问题。

Rabbit消息堆积过程

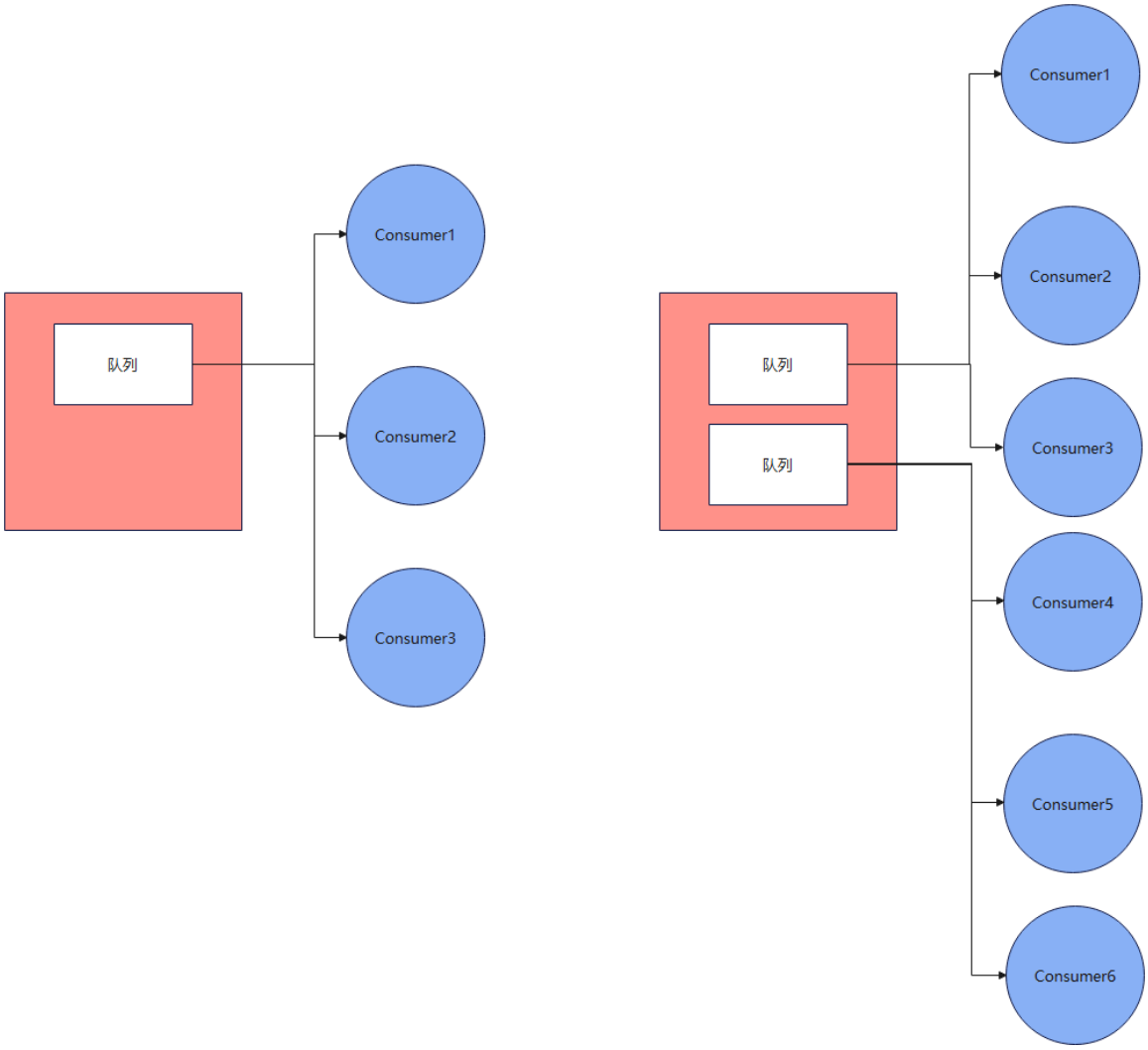


想要防止消息堆积，可以从这几个方面入手：

（1）增加消费者数量。假设如果只有3个消费者，每秒可以处理50个消息。增加为30个，就可以出了500个消息了。



（2）增加队列的数量。以前消息都发送到一个队列中，现在让消息发送到多个队列中。



（3）在消费者开启线程。在消费者内部使用多线程代码，让每个任务处理都不阻塞线程。

```
@Component
public class StringMessageConsumer {
    @RabbitListener(queues = {"queue.first"})
    public void onMessage(String messageBody){
        new Thread(){
            @Override
            public void run(){
                System.out.println("接收到的消息：" + messageBody);
            }
        }.start();
    }
}
```

（4）设置RabbitMQ最大并行数量。在使用Spring-AMQP框架时可以配置下面的三个参数。

```
# 并行消费者数量
spring.rabbitmq.listener.simple.concurrency=10
# 最大并行消费者数量
spring.rabbitmq.listener.simple.max-concurrency=30
# 消费者从队列每次获取消息数量
spring.rabbitmq.listener.simple.prefetch=3
```

（5）采用临时方案，上线临时消费者，不考虑任何业务逻辑，就是为了取消息，把消息都存储到Redis中。后期在慢慢从Redis中把消息进行消费。这种方案一般都是无奈之举，实在没有办法的时候才会采取。

那么我们回到面试官的问题：“当RabbitMQ突然接收几万条消息导致消息堆积如何处理”。

我们的回答应该是，如果当前项目没有处理这种情况的能力，可以考虑临时增加多个Consumer来临时解决这个问题，后期固定增加一定数量的Consumer，也可以增加多个队列来处理消息。

如果还是无法处理完成，临时上线新的消费者，把消息存放到Redis中。

对于这样的问题还是应该从预防为主。在项目架构时就应该考虑到这种情况。所以除了增加消费者数量，增加队列数量，临时存储Redis作为应急方案以外。还应该在项目中配置合理的并行消息处理数量，在编码时考虑多线程编码。