# 背景引入

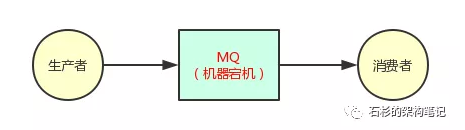
这篇文章，我们来聊一下消息中间件高可用架构的一些原理。

对于一个合格的高级Java工程师而言，你肯定会碰到在系统里用到MQ的场景，那么这个时候你需要基于你的业务场景和需求，考虑在使用MQ的时候可能遇到的一些技术问题。

接着，你必须得针对这些技术问题设计一套完整的技术方案。

你需要从消息的订阅模式、消息的生产到消费全链路不丢数据、消息中间件本身如何保证高可用，等各个角度切入，来考虑好你的系统和MQ对接之后的完整技术方案。

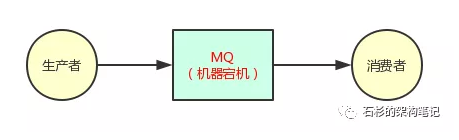
所以，本文就来聊聊消息中间件高可用的架构原理。



# 先来思考一下消息中间件的可用性问题

咱们先抛开各种具体的技术，就来思考一下，啥是MQ的可用性问题？

大家看看下面的图，其实道理很简单，假如你的MQ就部署在一台机器上，那么正常情况下，生产者都会发送消息到MQ去，然后让消费者获取到。



但是万一天有不测风云，MQ部署的那台机器，因为一些莫名的原因，MQ自己本身的进程挂掉了，或者是那台机器直接就宕机了，那么此时怎么办呢？

很尴尬，是不是，结果是很明显的，生产者没法发送数据出去，然后消费者也没法获取到数据了。

然后整个系统不就完蛋了？因为系统的核心流程根本无法跑通了，对不对？

MQ宕机就直接导致你的系统本身也故障了，然后可能会导致你的公司对外的APP、网站等产品就无法运作了，用户无法使用你们公司的服务了。

如果你们公司是电商平台、外卖平台、社交平台。那么来这么一出，不是会导致公司损失惨重？

如果你的系统持续几个小时无法被人使用，本来你公司电商平台一天营收可以达到1亿，结果现在导致几个小时内无法下单购买商品，最后当天营收就5000万，那么你的公司是不是直接活生生损失了5000万？

这个真的不是开玩笑的，如果大家留意互联网行业的新闻的话和小道消息的话，就应该知道近几年一些大型互联网公司都出现过类似的情况，损失惨重，咱们做码农的就得被祭天了是不是？

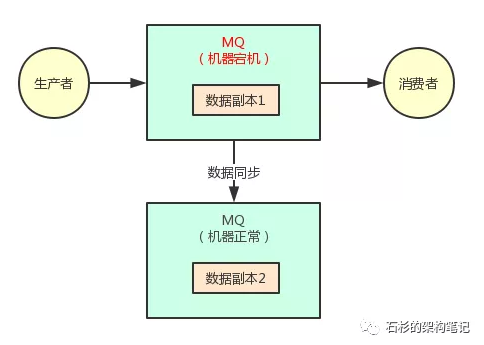
# 集群化部署 + 数据多副本冗余

好，问题来了！现在你感觉一个MQ中间件应该如何实现高可用呢？

这里的方式有很多种，比如说数据多副本冗余，集群镜像同步机制，我们就抛开具体的技术来从本质层面思考一下MQ集群实现高可用的几种方式。

先来看下面的一张图，假设我们写到MQ的数据都被多副本冗余了，也就是你写的每一条消息都被复制到了其他的机器上去了。

那么此时任何一台机器宕机，似乎都不会影响我们跟MQ继续通信，而且写出去的数据似乎也都还在。



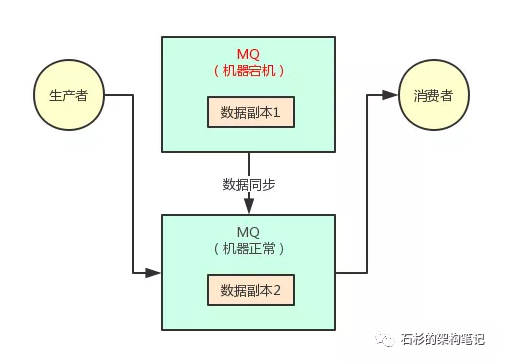
上面的图里，MQ采用集群模式部署到了2台机器上去，然后生产者给其中一台机器写入一条消息，该机器自动同步复制给另外一台机器。

此时数据在2台机器上，就有2个副本了，那么如果第一台机器宕机了，会影响我们吗？

答案是：不会。

因为数据本身是多副本冗余的，此时消费者完全可以从第二台机器消费到这条消息，并且生产者还可以继续给第二台机器写入消息，数据没丢失。

而且，系统根本不用中断流程，还可以继续运行，我们看下面的图。



这种感觉是不是很棒？实际上这种MQ集群化部署架构以及数据多副本冗余机制，是非常常见的一种高可用架构。

Kafka这个极为优秀的消息中间件，就是采用的这种架构保证高可用、数据容错性。

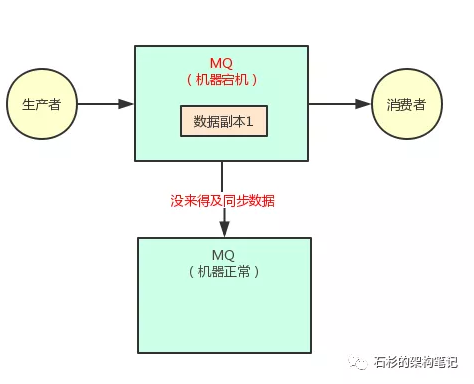
# 多副本同步复制强制要求

但是这里你要思考另外几个问题，第一个就是：你在写数据到其中一台机器的时候，是不是得要求，必须得让那台机器复制数据到另外一台机器了，保证集群里一定有这条数据双副本了，才可以认为本次写成功了？

没错，假如你要是不能保证这一点，比如你就写数据给了其中一台机器，然后他还没来得及复制给另外一台机器呢，直接第一台机器就宕机了。

此时虽然你可以继续基于第二台机器发送消息和消费消息，但是你刚才发送的一条消息就丢失了。

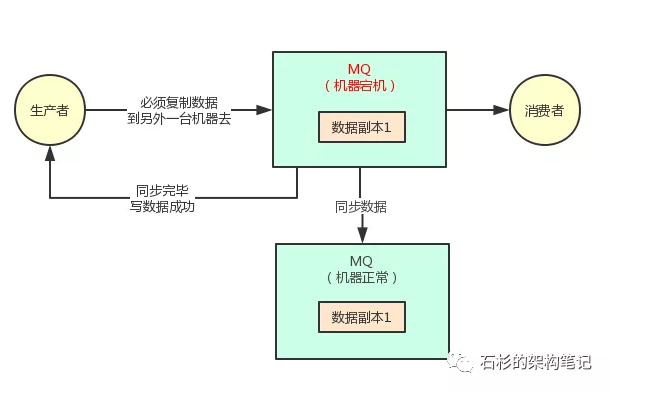
大家看下面的图来理解一下这个场景。



所以对于采用这种机制的时候，你必须得让生产者通过一些参数的设置，保证说写一条消息到某台机器，他必须同步这条消息到另外一台机器成功，集群里有双副本了，然后此时才可以认为这条消息写成功了。

但凡刚写一台机器他就宕机，还没来得及复制到另外一台机器的话，本次写应该报错失败，然后你应该重试再次写入数据到MQ集群里去。

大家看看下面的图。只要你一次写成功了，他就保证肯定已经同步数据为双副本了，此时哪怕一台机器宕机，数据不会丢失，生产和消费都可以有条不紊的继续进行。



# 多机器承载多副本强制要求

第二个问题，假如说现在你的集群中本来有两台机器，现在宕机了其中的一台，只有一台机器了，你还能允许你的生产者对唯一的一台机器继续写入数据吗？

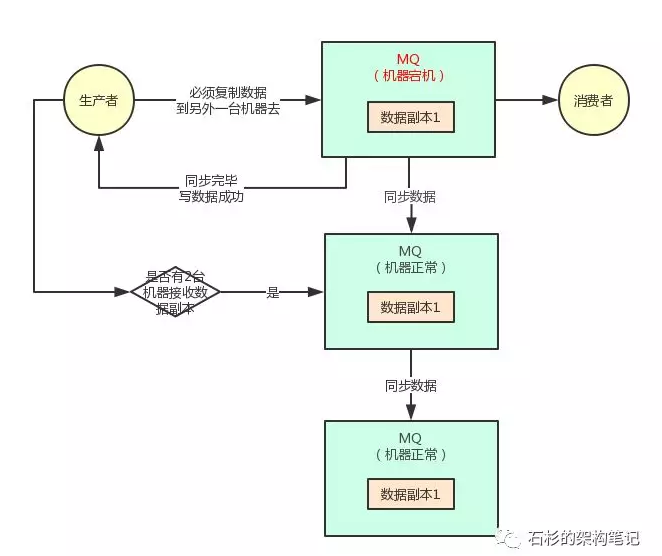
答案是：否。

因为如果集群里只有一台机器可以承载写入，那么万一剩余的一台机器又宕机了呢？是不是还是会导致数据丢失，集群完蛋？

所以说，你的生产者同理应该基于参数设置一下，集群里必须有超过2台机器可以接收你的数据副本复制。

否则如果只有1台机器可以接受你的数据副本复制的话，那么还是算了。

大家看看下面的图，感受一下那个场景。



假设集群里有3台机器，那么其中一台宕机了，你后续再写入另外一台的时候，判断一下集群里还有剩余两台机器，足以保证数据双副本的高可用性和容错性，所以可以继续正常的写入数据到MQ集群里去。

实际上，上面说的那一整套的机制，在Kafka里都可以采用，他有对应的一些参数可以配置数据有几个副本，包括你每次写入必须复制到几台机器才可以算成功，否则就要重新发送，以及你的集群剩余机器必须可以承载几个副本才能继续写入数据。

通过这一整套方案的设计和基于具体技术的落地，才可以保证在集群化部署的情况下，集群必须有几台机器承载多副本，同时数据写入之后必须是保证多副本冗余的。

此时，任何机器宕机，数据都不会丢失，还可以正常让系统继续运行。

# 架构原理与技术无关性

其实本文对消息中间件的集群高可用架构的探讨，是完全脱离于某个具体技术的，非常朴素的从本质的原理层面来讨论这个话题。

具体的RabbitMQ、Kafka、RocketMQ等各种不同的消息中间件，对这种高可用架构的实现，都有一定的相似想通性，但是也都有各自不同的技术实现，以及相对应的区别。