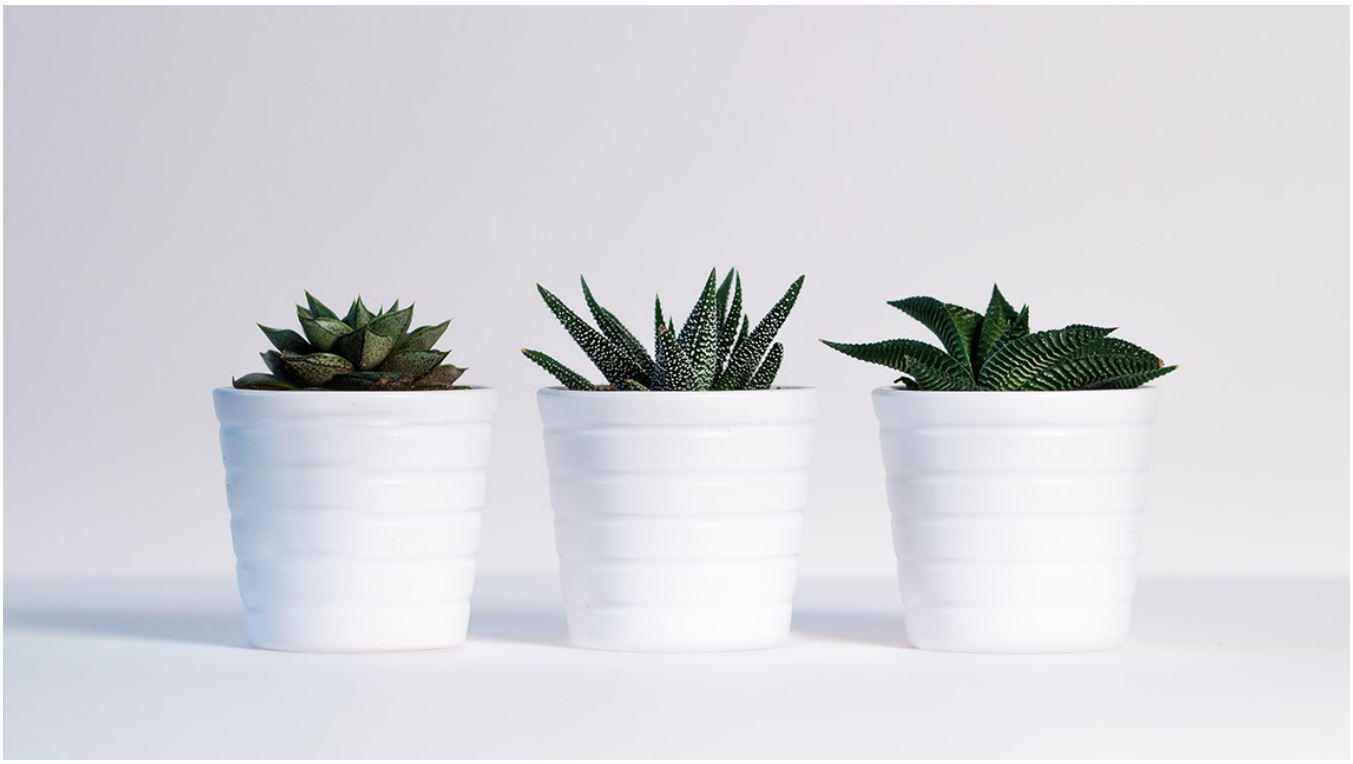
02 | 该如何选择消息队列？

2019-07-25 李玥



你好，我是李玥。这节课我们来聊一下几个比较常见的开源的消息队列中间件。如果你正在做消息队列技术选型，不知道该选择哪款消息队列，你一定要先听一下这节课的内容。

作为一个程序员，相信你一定听过“没有银弹”这个说法，这里面的银弹是指能轻松杀死狼人、用白银做的子弹，什么意思呢？我对这句话的理解是说，在软件工程中，不存在像“银弹”这样可以解决一切问题的设计、架构或软件，每一个软件系统，它都是独一无二的，你不可能用一套方法去解决所有的问题。

在消息队列的技术选型这个问题上，也是同样的道理。并不存在说，哪个消息队列就是“最好的”。常用的这几个消息队列，每一个产品都有自己的优势和劣势，你需要根据现有系统的情况，选择最适合你的那款产品。

选择消息队列产品的基本标准

虽然这些消息队列产品在功能和特性方面各有优劣，但我们在选择的时候要有一个最低标准，保证入选的产品至少是及格的。

接下来我们先说一下这及格的标准是什么样的。

首先，必须是开源的产品，这个非常重要。开源意味着，如果有一天你使用的消息队列遇到了一个影响你系统业务的Bug，你至少还有机会通过修改源代码来迅速修复或规避这个Bug，解决你的系统火烧眉毛的问题，而不是束手无策地等待开发者不一定什么时候发布的下一个版本来解

决。

其次，这个产品必须是近年来比较流行并且有一定社区活跃度的产品。流行的好处是，只要你的使用场景不太冷门，你遇到Bug的概率会非常低，因为大部分你可能遇到的Bug，其他人早就遇到并且修复了。你在使用过程中遇到的一些问题，也比较容易在网上搜索到类似的问题，然后很快的找到解决方案。

还有一个优势就是，流行的产品与周边生态系统会有一个比较好的集成和兼容，比如，Kafka和Flink就有比较好的兼容性，Flink内置了Kafka的Data Source，使用Kafka就很容易作为Flink的数据源开发流计算应用，如果你用一个比较小众的消息队列产品，在进行流计算的时候，你就不得不自己开发一个Flink的Data Source。

最后，作为一款及格的消息队列产品，必须具备的几个特性包括：

- 消息的可靠传递：确保不丢消息；
- Cluster：支持集群，确保不会因为某个节点宕机导致服务不可用，当然也不能丢消息；
- 性能：具备足够好的性能，能满足绝大多数场景的性能要求。

接下来我们一起看一下有哪些符合上面这些条件，可供选择的开源消息队列产品。

可供选择的消息队列产品

1. RabbitMQ

首先，我们说一下老牌儿消息队列RabbitMQ，俗称兔子MQ。RabbitMQ是使用一种比较小众的编程语言：Erlang语言编写的，它最早是为电信行业系统之间的可靠通信设计的，也是少数几个支持AMQP协议的消息队列之一。

RabbitMQ就像它的名字中的兔子一样：轻量级、迅捷，它的Slogan，也就是宣传口号，也很明确地表明了RabbitMQ的特点：Messaging that just works，“开箱即用的消息队列”。也就是说，RabbitMQ是一个相当轻量级的消息队列，非常容易部署和使用。

另外RabbitMQ还号称是世界上使用最广泛的开源消息队列，是不是真的使用率世界第一，我们没有办法统计，但至少是“最流行的消息中间之一”，这是没有问题的。

RabbitMQ一个比较有特色的功能是支持非常灵活的路由配置，和其他消息队列不同的是，它在生产者（Producer）和队列（Queue）之间增加了一个Exchange模块，你可以理解为交换机。

这个Exchange模块的作用和交换机也非常相似，根据配置的路由规则将生产者发出的消息分发到不同的队列中。路由的规则也非常灵活，甚至你可以自己来实现路由规则。基于这个Exchange，可以产生很多的玩儿法，如果你正好需要这个功能，RabbitMQ是个不错的选择。

RabbitMQ的客户端支持的编程语言大概是所有消息队列中最多的，如果你的系统是用某种冷门

语言开发的，那你多半可以找到对应的RabbitMQ客户端。

接下来说下RabbitMQ的几个问题。

第一个问题是，RabbitMQ对消息堆积的支持并不好，在它的设计理念里面，消息队列是一个管道，大量的消息积压是一种不正常的情况，应当尽量去避免。当大量消息积压的时候，会导致RabbitMQ的性能急剧下降。

第二个问题是，RabbitMQ的性能是我们介绍的这几个消息队列中最差的，根据官方给出的测试数据综合我们日常使用的经验，依据硬件配置的不同，它大概每秒钟可以处理几万到十几万条消息。其实，这个性能也足够支撑绝大多数的应用场景了，不过，如果你的应用对消息队列的性能要求非常高，那不要选择RabbitMQ。

最后一个问题是RabbitMQ使用的编程语言Erlang，这个编程语言不仅是非常小众的语言，更麻烦的是，这个语言的学习曲线非常陡峭。大多数流行的编程语言，比如Java、C/C++、Python和JavaScript，虽然语法、特性有很多的不同，但它们基本的体系结构都是一样的，你只精通一种语言，也很容易学习其他的语言，短时间内即使做不到精通，但至少能达到“会用”的水平。

就像一个以英语为母语的人，学习法语、德语都很容易，但是你要是让他去学汉语，那基本上和学习其他这些语言不是一个难度级别的。很不幸的是，Erlang就是编程语言中的“汉语”。所以如果你想基于RabbitMQ做一些扩展和二次开发什么的，建议你慎重考虑一下可持续维护的问题。

2. RocketMQ

RocketMQ是阿里巴巴在2012年开源的消息队列产品，后来捐赠给 Apache 软件基金会，2017正式毕业，成为Apache的顶级项目。阿里内部也是使用RocketMQ作为支撑其业务的消息队列，经历过多次“双十一”考验，它的性能、稳定性和可靠性都是值得信赖的。作为优秀的国产消息队列，近年来越来越多的被国内众多大厂使用。

我在总结RocketMQ的特点时，发现很难找出RocketMQ有什么特别让我印象深刻的特点，也很难找到它有什么缺点。

RocketMQ就像一个品学兼优的好学生，有着不错的性能，稳定性和可靠性，具备一个现代的消息队列应该有的几乎全部功能和特性，并且它还在持续的成长中。

RocketMQ有非常活跃的中文社区，大多数问题你都可以找到中文的答案，也许会成为你选择它的一个原因。另外，RocketMQ使用Java语言开发，它的贡献者大多数都是中国人，源代码相对也比较容易读懂，你很容易对RocketMQ进行扩展或者二次开发。

RocketMQ对在线业务的响应时延做了很多的优化，大多数情况下可以做到毫秒级的响应，**如果你的应用场景很在意响应时延，那应该选择使用RocketMQ。**

RocketMQ的性能比RabbitMQ要高一个数量级，每秒钟大概能处理几十万条消息。

RocketMQ的一个劣势是，作为国产的消息队列，相比国外的比较流行的同类产品，在国际上还没有那么流行，与周边生态系统的集成和兼容程度要略逊一筹。

3. Kafka

最后我们聊一聊Kafka。Kafka最早是由LinkedIn开发，目前也是Apache的顶级项目。Kafka最初的设计目的是用于处理海量的日志。

在早期的版本中，为了获得极致的性能，在设计方面做了很多的牺牲，比如不保证消息的可靠性，可能会丢失消息，也不支持集群，功能上也比较简陋，这些牺牲对于处理海量日志这个特定的场景都是可以接受的。这个时期的Kafka甚至不能称之为一个合格的消息队列。

但是，请注意，重点一般都在后面。随后的几年Kafka逐步补齐了这些短板，你在网上搜到的很多消息队列的对比文章还在说Kafka不可靠，其实这种说法早已经过时了。当下的Kafka已经发展为一个非常成熟的消息队列产品，无论在数据可靠性、稳定性和功能特性等方面都可以满足绝大多数场景的需求。

Kafka与周边生态系统的兼容性是最好的没有之一，尤其在大数据和流计算领域，几乎所有的相关开源软件系统都会优先支持Kafka。

Kafka使用Scala和Java语言开发，设计上大量使用了批量和异步的思想，这种设计使得Kafka能做到超高的性能。Kafka的性能，尤其是异步收发的性能，是三者中最好的，但与RocketMQ并没有量级上的差异，大约每秒钟可以处理几十万条消息。

我曾经使用配置比较好的服务器对Kafka进行过压测，在有足够的客户端并发进行异步批量发送，并且开启压缩的情况下，Kafka的极限处理能力可以超过每秒2000万条消息。

但是Kafka这种异步批量的设计带来的问题是，它的同步收发消息的响应时延比较高，因为当客户端发送一条消息的时候，Kafka并不会立即发送出去，而是要等一会儿攒一批再发送，在它的Broker中，很多地方都会使用这种“先攒一波再一起处理”的设计。当你的业务场景中，每秒钟消息数量没有那么多的时候，Kafka的时延反而会比较高。所以，**Kafka不太适合在线业务场景。**

第二梯队的消息队列

除了上面给你介绍的三大消息队列之外，还有几个第二梯队的产品，我个人的观点是，这些产品之所以没那么流行，或多或少都有着比较明显的短板，不推荐使用。在这儿呢，我简单介绍一下，纯当丰富你的知识广度。

先说ActiveMQ，ActiveMQ是最老牌的开源消息队列，是十年前唯一可供选择的开源消息队列，

目前已进入老年期，社区不活跃。无论是功能还是性能方面，ActiveMQ都与现代的消息队列存在明显的差距，它存在的意义仅限于兼容那些还在用的爷爷辈儿的系统。

接下来说说ZeroMQ，严格来说ZeroMQ并不能称之为一个消息队列，而是一个基于消息队列的多线程网络库，如果你的需求是将消息队列的功能集成到你的系统进程中，可以考虑使用ZeroMQ。

最后说一下Pulsar，很多人可能都没听说过这个产品，Pulsar是一个新兴的开源消息队列产品，最早是由Yahoo开发，目前处于成长期，流行度和成熟度相对没有那么高。与其他消息队列最大的不同是，Pulsar采用存储和计算分离的设计，我个人非常喜欢这种设计，它有可能会引领未来消息队列的一个发展方向，建议你持续关注这个项目。

总结

在了解了上面这些开源消息队列各自的特点和优劣势后，我相信你对于消息队列的选择已经可以做到心中有数了。我也总结了几条选择的建议供你参考。

如果说，消息队列并不是你将要构建系统的主角之一，你对消息队列功能和性能都没有很高的要求，只需要一个开箱即用易于维护的产品，我建议你使用RabbitMQ。

如果你的系统使用消息队列主要场景是处理在线业务，比如在交易系统中用消息队列传递订单，那RocketMQ的低延迟和金融级的稳定性是你需要的。

如果你需要处理海量的消息，像收集日志、监控信息或是前端的埋点这类数据，或是你的应用场景大量使用了大数据、流计算相关的开源产品，那Kafka是最适合你的消息队列。

如果我说的这些场景和你的场景都不符合，你看了我之前介绍的这些消息队列的特点后，还是不知道如何选择，那就选你最熟悉的吧，毕竟这些产品都能满足大多数应用场景，使用熟悉的产品还可以快速上手不是？

思考题

本节课的思考题也是围绕着消息队列的技术选型来设置的。你开发过的或是正在开发的系统，对消息队列的需求是什么样的？现在选择的消息队列是哪款产品？在学完了本节课后，你觉得当前选择的消息队列是否是最佳的选择？理由是什么？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有一些启发，也欢迎把它分享给你的朋友。

精选留言

傻白田先森

18

仔细阅读了三遍，每一字都是精华

选择中间件的考量维度：可靠性，性能，功能，可运维行，可拓展性，是否开源及社区活跃度

rabbitmq：

优点：轻量，迅捷，容易部署和使用，拥有灵活的路由配置

缺点：性能和吞吐量较差，不易进行二次开发

rocketmq：

优点：性能好，稳定可靠，有活跃的中文社区，特点响应快

缺点：兼容性较差，但随意影响力的扩大，该问题会有改善

kafka：

优点：拥有强大的性能及吞吐量，兼容性很好

缺点：由于“攒一波再处理”导致延迟比较高

pulsar：

采用存储和计算分离的设计，是消息队里产品中黑马，值得持续关注

2019-07-25

盛

14

一套架构中是否可能存在多套中间件？在线的生产业务使用rockmq，运维/监控方面使用kafka

。

2019-07-25

作者回复

当然可以，架构无所谓好坏，关键是适合。用多套MQ好处是发挥各自的长处，代价是维护成本比较高。具体是不是适合，还是要架构师根据各种实际情况来权衡。

2019-07-25

WL

8

请问一下老师rocketMQ是怎么做到低延时的？

2019-07-25

作者回复

主要是设计上的选择问题，Kafka中到处都是“批量和异步”设计，它更关注的是整体的吞吐量，而RocketMQ的设计选择更多的是尽量及时处理请求。

比如发消息，同样是用户调用了send()方法，RockMQ它会直接把这个消息发出去，而Kafka会把这个消息放到本地缓存里面，然后择机异步批量发送。

所以，RocketMQ它的时延更小一些，而Kafka的吞吐量更高。

2019-07-26

公号：猿人谷

6

我所在公司用rabbitmq也遇到消息的有序性无法保证的问题，通过在业务层面去弥补，终究不是种好方案。
请问老师在保证有序性消费上有什么好的方案？

2019-07-25

作者回复

正确的使用RabbitMQ是可以保证严格有序的，你在学习完“03 消息模型：主题和队列有什么区别？”之后，再看一下RabbitMQ的配置应该就会知道该如何解决你的问题了。

2019-07-26

吴青

3

老师 exchange是rabbitmq独有的么？exchange好像属于amqp协议，看了看amqp似乎说到了。

2019-07-28

作者回复

exchange确实是AMQP协议中定义的，RabbitMQ是AMQP的一个实现。

2019-07-29

Mark Yao

2

我们就是那种对消息队列功能和性能都没有很高的要求，所以选择RabbitMQ。不管选择哪种消息队列其中还有个很关键的因素，团队里面有人能hold它，最起码熟悉掌握其详细配置。选择不熟悉的MQ会变成不定时炸弹，在生产遇到问题无法快速解决。

2019-07-27

wmg

2

“因为当客户端发送一条消息的时候，Kafka 并不会立即发送出去，而是要等一会儿攒一批再发送，在它的 Broker 中，很多地方都会使用这种“先攒一波再一起处理”的设计。当你的业务场景

中，每秒钟消息数量没有那么多的时候，Kafka 的时延反而会比较高。所以，Kafka 不太适合在线业务场景。”，老师，批次的大小是可配置的，在我们的使用场景中，如果消息没有积压的情况下，延迟基本上小于10ms，我想问一下rocketmq的延迟一般是多少？

2019-07-26

作者回复

配置得当的情况下，可以做到2-3ms。

2019-07-26

David Mao 2

我们的云平台有多个项目，每个项目用的消息中间件不同，有的用RabbitMQ, 有的用Kafka，请教老师，这些消息中间件可以建设成一个统一的，集中式的架构吗？也就是建设成一个消息中间件平台，所有的项目来共用。

2019-07-25

作者回复

当然可以，在京东就是这样的集中式大集群，为所有业务提供消息服务。

2019-07-25

zwh 1

正如老师所说，性能优秀rocketmq在国外的知名度真的挺低的。国外除了rabbitmq kafka还有还多公司使用Amazon的simple queue service

2019-07-28

我已经设置了昵称 1

一直不明白一个问题，每秒处理几万～几十万数据，这不应该是按照下游业务的消费能力来讲的吗，比如我们现在的服务tps就只能打到几百。还是说老师和网上讲的都是发送到消息队列的能力？又或是几万几十万只针下游业务只打个日志专门用来做性能测试的情况？希望解答下

2019-07-27

作者回复

我们提到的都是消息队列本身的性能，不包括生产者和消费者处理各自业务逻辑的时间。一般测试消息队列性能的时候，生产者和消费者是没有任何业务逻辑的。

2019-07-27

青舟 1

老师您好，
请问是否有消息队列支持延时消息，并可以撤回延时消息的？

需求：物联网平台要对设备离线事件做监控，我想在每次设备上报数据时，发送一个延时判断离线的延时消息到MQ，如果设备正常再次上报数据，就取消上次的延时消息，并发送一个新的延时消息。
目的是不希望每次上报数据都会导致消费端做一次离线判断，只需要对最后一次上报做判断即可。

2019-07-26

作者回复

但我个人的意见：这个需求不太适合用MQ来实现。

另外，是不是可以换个思路来考虑这个问题，你一定需要一个设备离线的通知来触发什么业务逻辑吗？还是只是需要查询设备的时候，能正确的给出设备是否在线就可以了？

如果是后者，实现代价要低很多。

2019-07-27

发条橙子 。 1

总结一波：

三大主流消息队列：

Rabbit mq: 开箱即用 主要重点就是在队列上，性能一般对有消息堆积的场景下支持较弱，每秒处理几万左右 ，语言的原因造成不太好二次维护

2019-07-26

QQ怪 1

哔，打卡，文章听一般就懂

2019-07-25

业余草 1

rockmq + kafka。业务 + 日志，都需要！

2019-07-25

Loren 1

卡！数据通道 用的kakfa，

2019-07-25

古夜 1

我们用zeromq，奈何国内资料太少，想深入了解都不行

2019-07-25

作者回复

所以需要好好学英语哦。

2019-07-25

Jaising 1

李老师所说的是不是可以按照CAP理论来讲，不同场景总要做出取舍对吧；

另，目前在线业务使用RocketMQ，做了二次开发的，中文支持不错而且社区响应速度快

2019-07-25

作者回复

是的，架构设计很多情况下是在做取舍和选择。

2019-07-25

努力奋斗的Pisces 0

老师，kafka是批量发送的话，对于处理多个消息处理一个业务的时候是不是会出现先后顺序的问题

2019-07-31

作者回复

批量也是保证顺序的。

2019-08-01

Panda

0

看好 Pulsar

2019-07-31

苦行僧

0

我们用的就是上古的activemq，客户端消息监听，服务端推送，老师broker无法在多个消费者监听的情况下保证有序性，总会出现后入的消息先消费掉？

2019-07-31