01-消息引擎系统ABC

你好，我是胡夕。欢迎你来到“Kafka核心技术与实战”专栏。如果你对Kafka及其背后的消息引擎、流处理感兴趣，很高兴我们可以在此相聚，并在未来的一段日子里一同学习有关Kafka的方方面面。

毫无疑问，你现在对Apache Kafka一定充满了各种好奇，那么今天就允许我先来尝试回答下Kafka是什么这个问题。对了，先卖个关子，在下一期我还将继续回答这个问题，而且答案是不同的。那么，Kafka是什么呢？用一句话概括一下：**Apache Kafka是一款开源的消息引擎系统**。

倘若“消息引擎系统”这个词对你来说有点陌生的话，那么“消息队列”“消息中间件”的提法想必你一定是有所耳闻的。不过说实话我更愿意使用消息引擎系统这个称谓，因为消息队列给出了一个很不明确的暗示，仿佛Kafka是利用队列的方式构建的；而消息中间件的提法有过度夸张“中间件”之嫌，让人搞不清楚这个中间件到底是做什么的。

像Kafka这一类的系统国外有专属的名字叫Messaging System，国内很多文献将其简单翻译成消息系统。我个人认为并不是很恰当，因为它片面强调了消息主体的作用，而忽视了这类系统引以为豪的消息传递属性，就像引擎一样，具备某种能量转换传输的能力，所以我觉得翻译成消息引擎反倒更加贴切。

讲到这里，说点题外话。我觉得目前国内在翻译国外专有技术词汇方面做得不够标准化，各种名字和提法可谓五花八门。我举个例子，比如大名鼎鼎的Raft算法和Paxos算法。了解它的人都知道它们的作用是在分布式系统中让多个节点就某个决定达成共识，都属于Consensus Algorithm一族。如果你在搜索引擎中查找Raft算法，国内多是称呼它们为一致性算法。实际上我倒觉得翻译成共识算法是最准确的。我们使用“一致性”这个字眼太频繁了，国外的Consistency被称为一致性、Consensus也唤作一致性，甚至是Coherence都翻译成一致性。

还是拉回来继续聊消息引擎系统，那这类系统是做什么用的呢？我先来个官方严肃版本的答案。

根据维基百科的定义，消息引擎系统是一组规范。企业利用这组规范在不同系统之间传递语义准确的消息，实现松耦合的异步式数据传递。

果然是官方定义，有板有眼。如果觉得难于理解，那么可以试试我下面这个民间版：

系统A发送消息给消息引擎系统，系统B从消息引擎系统中读取A发送的消息。

最基础的消息引擎就是做这点事的！不论是上面哪个版本，它们都提到了两个重要的事实：

- 消息引擎传输的对象是消息；
- 如何传输消息属于消息引擎设计机制的一部分。

既然消息引擎是用于在不同系统之间传输消息的，那么如何设计待传输消息的格式从来都是一等一的大事。试问一条消息如何做到信息表达业务语义而无歧义，同时它还要能最大限度地提供可重用性以及通用性？稍微停顿几秒去思考一下，如果是你，你要如何设计你的消息编码格式。

一个比较容易想到的是使用已有的一些成熟解决方案，比如使用CSV、XML亦或是JSON；又或者你可能熟知国外大厂开源的一些序列化框架，比如Google的Protocol Buffer或Facebook的Avro。这些都是很酷的办法。那么现在我告诉你Kafka的选择：它使用的是纯二进制的字节序列。当然消息还是结构化的，只是在使

用之前都要将其转换成二进制的字节序列。

消息设计出来之后还不够，消息引擎系统还要设定具体的传输协议，即我用什么方法把消息传输出去。常见的有两种方法：

- **点对点模型**：也叫消息队列模型。如果拿上面那个“民间版”的定义来说，那么系统A发送的消息只能被系统B接收，其他任何系统都不能读取A发送的消息。日常生活的例子比如电话客服就属于这种模型：同一个客户呼入电话只能被一位客服人员处理，第二个客服人员不能为该客户服务。
- **发布/订阅模型**：与上面不同的是，它有一个主题（Topic）的概念，你可以理解成逻辑语义相近的消息容器。该模型也有发送方和接收方，只不过提法不同。发送方也称为发布者（Publisher），接收方称为订阅者（Subscriber）。和点对点模型不同的是，这个模型可能存在多个发布者向相同的主题发送消息，而订阅者也可能存在多个，它们都能接收到相同主题的消息。生活中的报纸订阅就是一种典型的发布/订阅模型。

比较酷的是Kafka同时支持这两种消息引擎模型，专栏后面我会分享Kafka是如何做到这一点的。

提到消息引擎系统，你可能会问JMS和它是什么关系。JMS是Java Message Service，它也是支持上面这两种消息引擎模型的。严格来说它并非传输协议而仅仅是一组API罢了。不过可能是JMS太有名气以至于很多主流消息引擎系统都支持JMS规范，比如ActiveMQ、RabbitMQ、IBM的WebSphere MQ和Apache Kafka。当然Kafka并未完全遵照JMS规范，相反，它另辟蹊径，探索出了一条特有的道路。

好了，目前我们仅仅是了解了消息引擎系统是做什么的以及怎么做的，但还有个重要的问题是为什么要使用它。

依旧拿上面“民间版”举例，我们不禁要问，为什么系统A不能直接发送消息给系统B，中间还要隔一个消息引擎呢？

答案就是“**削峰填谷**”。这四个字简直比消息引擎本身还要有名气。

我翻了很多文献，最常见的就是这四个字。所谓的“削峰填谷”就是指缓冲上下游瞬时突发流量，使其更平滑。特别是对于那种发送能力很强的上游系统，如果没有消息引擎的保护，“脆弱”的下游系统可能会直接被压垮导致全链路服务“雪崩”。但是，一旦有了消息引擎，它能够有效地对抗上游的流量冲击，真正做到将上游的“峰”填满到“谷”中，避免了流量的震荡。消息引擎系统的另一大好处在于发送方和接收方的松耦合，这也在一定程度上简化了应用的开发，减少了系统间不必要的交互。

说了这么多，可能你对“削峰填谷”并没有太多直观的感受。我还是举个例子来说明一下Kafka在这中间是怎么去“抗”峰值流量的吧。回想一下你在极客时间是如何购买这个课程的。如果我没记错的话极客时间每门课程都有一个专门的订阅按钮，点击之后进入到付费页面。这个简单的流程中就可能包含多个子服务，比如点击订阅按钮会调用订单系统生成对应的订单，而处理该订单会依次调用下游的多个子系统服务，比如调用支付宝和微信支付的接口、查询你的登录信息、验证课程信息等。显然上游的订单操作比较简单，它的TPS要远高于处理订单的下游服务，因此如果上下游系统直接对接，势必会出现下游服务无法及时处理上游订单从而造成订单堆积的情形。特别是当出现类似于秒杀这样的业务时，上游订单流量会瞬时增加，可能出现的结果就是直接压跨下游子系统服务。

解决此问题的一个常见做法是我们对上游系统进行限速，但这种做法对上游系统而言显然是不合理的，毕竟问题并不出现在它那里。所以更常见的办法是引入像Kafka这样的消息引擎系统来对抗这种上下游系统TPS

的错配以及瞬时峰值流量。

还是这个例子，当引入了Kafka之后。上游订单服务不再直接与下游子服务进行交互。当新订单生成后它仅仅是向Kafka Broker发送一条订单消息即可。类似地，下游的各个子服务订阅Kafka中的对应主题，并实时从该主题的各自分区（Partition）中获取到订单消息进行处理，从而实现了上游订单服务与下游订单处理服务的解耦。这样当出现秒杀业务时，Kafka能够将瞬时增加的订单流量全部以消息形式保存在对应的主题中，既不影响上游服务的TPS，同时也给下游子服务留出了充足的时间去消费它们。这就是Kafka这类消息引擎系统的最大意义所在。

如果你对Kafka Broker、主题和分区等术语还不甚了解的话也不必担心，我会在专栏后面专门花时间介绍一下Kafka的常见概念和术语。

在今天结束之前，我还想和你分享一个自己的小故事。在2015年那会儿，我花了将近1年的时间阅读Kafka源代码，期间多次想要放弃。你要知道阅读将近50万行源码是多么痛的领悟。我还记得当初为了手写源代码注释，自己写满了一个厚厚的笔记本。不过幸运的是我坚持了下来，之前的所有努力也没有白费，以至于后面写书、写极客时间专栏就变成了一件件水到渠成的事情。

最后我想送给你一句话：**聪明人也要下死功夫**。我不记得这是曾国藩说的还是季羡林说的，但这句话对我有很大影响，当我感到浮躁的时候它能帮我静下心来踏踏实实做事情。希望这句话对你也有所启发。切记：聪明人要下死功夫！

开放讨论

请谈谈你对消息引擎系统的理解，或者分享一下你的公司或组织是怎么使用消息引擎来处理实际问题的。

欢迎写下你的思考和答案，我们一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

精选留言：

- 开发无止境，BUG随身行 2019-06-03 21:27:27

有个问题请教下老师:
之前也用过kafka，怎么解决实时结果响应问题呢？比如秒杀商品，生产者产生订单，消费者处理订单结果，那这结果如何实时返回给用户呢？ [11赞]

作者回复2019-06-04 10:13:01
这个场景使用Kafka Streams比较适合，它就是为read-process-write场景服务的

- Lei Yang 2019-06-03 22:44:37
老师可以讲一讲Kafka和别的mq的区别和最佳选择方法么？例如什么时候选择RabbitMQ什么时候选择Kafka等等 [7赞]

作者回复2019-06-04 10:17:37
RabbitMQ属于比较传统的消息队列系统，支持标准的消息队列协议（AMQP, STOMP，MQTT等），如果你的应用程序需要支持这些协议，那么还是使用RabbitMQ。另外RabbitMQ支持比较复杂的consumer Routing，这点也是Kafka不提供的。

- jeffery 2019-06-03 19:35:44
pulsar高吞吐低延迟和kafka谁会主宰未来？夕哥、能不能拓展下flink+kafka的耦合！谢谢 [5赞]

作者回复2019-06-04 09:15:58
和Pulsar的斯杰、翟佳都相识，不敢妄下结论。Flink + Kafka最近的确有标准套餐的趋势：）

- miofy 2019-06-03 19:04:30
1. consesus algorithm，在区块链中多翻译为共识算法，而在其它领域多被翻译为一致性算法，个人觉得共识算法表意更清楚。

2. 削峰填谷，实际上就是流量整形的形象表达，主要还是为了应对上游瞬时大流量的冲击，避免出现流量毛刺现象，保护下游应用和数据库不被大流量打垮。 [4赞]

- 孙志强 2019-06-04 16:00:06
讲讲怎么把50完行源代码读下来的? 嘿嘿 [3赞]

作者回复2019-06-04 19:11:18
一行一行啃下来的。如果你也有兴趣，我建议可以先从kafka.log包开始读起，会很有收获的~~

- 永恒记忆 2019-06-03 20:13:31
老师好，想问下有些业务用mq来做异步处理，为了削峰填谷，是不是上游发送消息成功就认为业务成功了，可能下游过很久去消费，那实时性要求很高的业务怎么办呢，比如生成了订单但是一直不处理也不好吧。另外想请教下老师的角度来讲下mq和rpc调用的区别是什么呢？ [3赞]

作者回复2019-06-04 09:30:03
mq和rpc的区别往大了说属于数据流模式（dataflow mode）的问题。我们常见的数据流有三种：1. 通过数据库；2. 通过服务调用（REST/RPC）; 3. 通过异步消息传递（消息引擎，如Kafka）
RPC和MQ是有相似之处的，毕竟我们远程调用一个服务也可以看做是一个事件，但不同之处在于：
1. MQ有自己的buffer，能够对抗过载（overloaded）和不可用场景
2. MQ支持重试
3. 允许发布/订阅模式
当然它们还有其他区别。应该这样说RPC是介于通过数据库和通过MQ之间的数据流模式。

- 安不安生 2019-06-03 19:23:28
我们公司用来传输视频切片，然后使用集群进行视频分析，之前曾经用过kafka ，因为没有人熟悉，不会

维护，导致放弃，现在使用aws kinesis 服务，怎么才能说服领导引进kafka 呢？
[3赞]

作者回复2019-06-04 08:15:22
hmmm... 使用Kafka自己把控度会高一些。另外很多公司对数据出公网是有顾虑的，使用云上的服务必然涉及到将 公司数据传给云服务器的问题。如果是敏感数据这也是要考虑的

- huaweichen 2019-06-04 14:37:06
曾国藩：真正聪明人都在下笨功夫！

https://zhuanlan.zhihu.com/p/25100394 [2赞]

- kaiux 2019-06-03 17:28:30
Kafka官网的描述是“Apache Kafka® is a distributed streaming platform.”，我觉得这里的重点在于分布式和流式处理，而且我认为消息引擎也可以看做是流式处理的一种，不知道老师怎么看？ [2赞]

作者回复2019-06-03 18:57:32
Kafka是以消息引擎起家的，后面转型成流处理平台。没有冒犯的意思，我不认为消息引擎是流处理的一种。事实上，流处理在意的是如何处理无限数据集的问题。它们是不同的领域：）

- QQ怪 2019-06-05 12:47:43
我们公司一般用消息引擎用于日志系统，但一般上游业务tps比较多的情况也会像作者一样做削峰填谷处理，但我想问问老师kafka是不是更加适合做日志分发系统？是不是kafka有一定程度上不保证数据一致性？[1赞]

作者回复2019-06-06 08:06:11
你指的数据不一致具体是什么意思呢？ ？

- 曾轼麟 2019-06-05 08:47:16
我们使用kafka做微服务间的数据下发，例如资金服务接口表数据就是来源上游的kafka消息 [1赞]

- 杨俊 2019-06-04 08:00:18
希望后面能说下要是kafka突然宕机或者临时停止服务进行更新，上游服务的消息该怎么正确更好处理呢？怎么保证消息的能够在kafka恢复工作的时候正确传递，谢谢 [1赞]

- skyhackvip 2019-06-04 00:20:11
我们常用Kafka消息引擎接受日志流，然后倒入bi系统。 [1赞]

- tracy 2019-06-03 18:13:45
现在消息中间件很多，想要了解kafka和其他消息中间件的优劣点，系统选型时需要考虑什么？ [1赞]

作者回复2019-06-03 18:53:01
如果是以实现高吞吐量为主要目标，Kafka是不错的首选；如果是以实现业务系统为主要目标，特别是金融类业务，可以考虑应用Kafka的流处理组件Kafka Streams。不过坦率说目前将Kafka应用于纯业务系统的并不多，但是前景依然可期：）

- QQ怪 2019-06-06 12:52:35
我指的数据不一致可能会容许消息丢失，麻烦老师解惑😁

- Savage.M 2019-06-06 00:41:34

老师，您好！感觉kafka和rocketmq的设计有很多相似之处，能否列举一下他们之间的区别呢？如果要选型，哪些场景适合用kafka，哪些场景适合用rocketmq呢？谢谢！

作者回复2019-06-06 08:13:38
我和RocketMQ的冯总也相识，说实话不敢妄言两者的优劣，网上也有一些文章比较过两者的区别。就目前公开的资料查看，RocketMQ宣称擅长主打金融业务领域场景，我个人是比较相信的。Kafka更多还是发家于大数据领域。

- RogerFederer 2019-06-05 23:30:46
胡老师：上文中讲到消息队列要把消息传输出去有2中模型：点对点模型(消息队列模型)；发布/订阅模型；然后举例点对点模式 系统 A 发送的消息只能被系统 B 接收。
一开始我就以为 消息的发送者只能有一个。但当在公司做老系统的队列改造时才明白;选择队列模型,消息的生产者不只是一个。多个系统会往一个队列里面丢消息,消费系统只有一个。
那这么一说来,称这个叫点对点模型是不是不合适？

作者回复2019-06-06 08:10:25
这就是学校和工程界不匹配的情形。P2P的提法是标准的学院派称谓，我觉得我们理解了意思就好。

- star 2019-06-05 17:16:36
请问下线上使用flink1.6和kafka0.9，现在想升级kafka版本，我看了flink-kafka-connector现在只有kafka0.11的版本，是不是我只能把kafka升级到0.11版本？

作者回复2019-06-06 08:07:47
目前Flink社区已经推出了flink-connector-kafka_2.11的connector，支持Kafka 1.0及以后的版本，但仍属于Beta版本，稳定性上不如0.11 connector

- 趙衍 2019-06-05 08:34:14
有个问题想请教老师：

为什么业务生产上都会用rabbitmq或者rocketmq，但一到日志的问题上kafka就是不二选择？根据老师之前的回答，是因为Kafka相比于其他的mq在吞吐量上更有优势，可以详细解释一下为什么Kafka可以提供高吞吐量吗？

作者回复2019-06-05 09:51:31
并没有说Kafka不能用于实际业务，只是说Kafka的确是从大数据场景里面发源出来的，而且很多业务应用要求支持传统的消息队列协议，这一点Kafka就无法做到了。另外，专栏后面会涉及Kafka提供高吞吐的原因，简单来说就是采用了log-structured的结构加上充分利用了操作系统提供的各种优化

- 爱学习的猪 2019-06-04 21:48:08
我们组用来做客服聊天系统了