03 | Kafka只是消息引擎系统吗?

胡夕 · Kafka核心技术与实战

你好，我是胡夕。今天我们来聊一个老生常谈的话题：Kafka 只是消息引擎系统吗?

要搞清楚这个问题，我们不可避免地要了解一下 Apache Kafka 的发展历程。有的时候我们会觉得说了解一个系统或框架的前世今生似乎没什么必要，直接开始学具体的技术不是更快更好吗？其实，不论是学习哪种技术，直接扎到具体的细节中，亦或是从一个很小的点开始学习，你很快就会感到厌烦。为什么呢？因为你虽然快速地搞定了某个技术细节，但无法建立全局的认知观，这会导致你只是在单个的点上有所进展，却没法将其串联成一条线进而扩展成一个面，从而实现系统地学习。

我这么说是有依据的，因为这就是我当初学习 Kafka 的方式。你可能不会相信，我阅读 Kafka 源码就是从 utils 包开始的。显然，我们不用看源码也知道这玩意是干什么用的，对吧？就是个工具类包嘛，而且这种阅读源码的方式是极其低效的。就像我说的，我是在一个点一个点地学习，但全部学完之后压根没有任何感觉，依然不了解 Kafka，因为不知道这些包中的代码组合在一起能达成什么效果。所以我说它是很低效的学习方法。

后来我修改了学习的方法，转而从自上而下的角度去理解 Kafka，竟然发现了很多之前学习过程中忽略掉的东西。更特别的是，我发现这种学习方法能够帮助我维持较长时间的学习兴趣，不会阶段性地产生厌烦情绪。特别是在了解 Apache Kafka 整个发展历史的过程中我愉快地学到了很多运营大型开源软件社区的知识和经验，可谓是技术之外的一大收获。

纵观 Kafka 的发展脉络，它的确是从消息引擎起家的，但正如文章标题所问，**Apache Kafka 真的只是消息引擎吗**？通常，在回答这个问题之前很多文章可能就要这样展开了：那我们先来讨论下什么是消息引擎以及消息引擎能做什么事情。算了，我还是直给吧，就不从“唐尧虞舜”说起了。这个问题的答案是，**Apache Kafka 是消息引擎系统，也是一个分布式流处理平台**（Distributed Streaming Platform）。如果你通读全篇文字但只能记住一句话，我希望你记住的就是这句。再强调一遍，Kafka 是消息引擎系统，也是分布式流处理平台。

众所周知，Kafka 是 LinkedIn 公司内部孵化的项目。根据我和 Kafka 创始团队成员的交流以及查阅到的公开信息显示，LinkedIn 最开始有强烈的数据强实时处理方面的需求，其内部的诸多子系统要执行多种类型的数据处理与分析，主要包括业务系统和应用程序性能监控，以及用户行为数据处理等。

当时他们碰到的主要问题包括：

> 数据正确性不足。因为数据的收集主要采用轮询（Polling）的方式，如何确定轮询的间隔时间就变成了一个高度经验化的事情。虽然可以采用一些类似于启发式算法（Heuristic）来帮助评估间隔时间值，但一旦指定不当，必然会造成较大的数据偏差。
>
> 系统高度定制化，维护成本高。各个业务子系统都需要对接数据收集模块，引入了大量的定制开销和人工成本。

为了解决这些问题，LinkedIn 工程师尝试过使用 ActiveMQ 来解决这些问题，但效果并不理想。显然需要有一个“大一统”的系统来取代现有的工作方式，而这个系统就是 Kafka。

Kafka 自诞生伊始是以消息引擎系统的面目出现在大众视野中的。如果翻看 0.10.0.0 之前的官网说明，你会发现 Kafka 社区将其清晰地定位为一个分布式、分区化且带备份功能的提交日志（Commit Log）服务。

这里引出一个题外话，你可能好奇 Kafka 这个名字的由来，实际上 Kafka 作者之一 Jay Kreps 曾经谈及过命名的原因。

> 因为 Kafka 系统的写性能很强，所以找了个作家的名字来命名似乎是一个好主意。大学期间我上了很多文学课，非常喜欢 Franz Kafka 这个作家，另外为开源软件起这个名字听上去很酷。

言归正传，Kafka 在设计之初就旨在提供三个方面的特性：

提供一套 API 实现生产者和消费者；

降低网络传输和磁盘存储开销；

实现高伸缩性架构。

在专栏后面的课程中，我们将陆续探讨 Kafka 是如何做到以上三点的。总之随着 Kafka 的不断完善，Jay 等大神们终于意识到将其开源惠及更多的人是一个非常棒的主意，因此在 2011 年 Kafka 正式进入到 Apache 基金会孵化并于次年 10 月顺利毕业成为 Apache 顶级项目。

开源之后的 Kafka 被越来越多的公司应用到它们企业内部的数据管道中，特别是在大数据工程领域，Kafka 在承接上下游、串联数据流管道方面发挥了重要的作用：所有的数据几乎都要从一个系统流入 Kafka 然后再流向下游的另一个系统中。这样的使用方式屡见不鲜以至于引发了 Kafka 社区的思考：与其我把数据从一个系统传递到下一个系统中做处理，我为何不自己实现一套流处理框架呢？基于这个考量，Kafka 社区于 0.10.0.0 版本正式推出了流处理组件 Kafka Streams，也正是从这个版本开始，Kafka 正式“变身”为分布式的流处理平台，而不仅仅是消息引擎系统了。今天 Apache Kafka 是和 Apache Storm、Apache Spark 和 Apache Flink 同等级的实时流处理平台。

诚然，目前国内对 Kafka 是流处理平台的认知还尚不普及，其核心的流处理组件 Kafka Streams 更是少有大厂在使用。但我们也欣喜地看到，随着在 Kafka 峰会上各路大神们的鼎力宣传，如今利用 Kafka 构建流处理平台的案例层出不穷，而了解并有意愿使用 Kafka Streams 的厂商也是越来越多，因此我个人对于 Kafka 流处理平台的前景也是非常乐观的。

你可能会有这样的疑问：作为流处理平台，Kafka 与其他主流大数据流式计算框架相比，优势在哪里呢？我能想到的有两点。

第一点是更容易实现端到端的正确性（Correctness）。Google 大神 Tyler 曾经说过，流处理要最终替代它的“兄弟”批处理需要具备两点核心优势：**要实现正确性和提供能够推导时间的工具。实现正确性是流处理能够匹敌批处理的基石**。正确性一直是批处理的强项，而实现正确性的基石则是要求框架能提供精确一次处理语义，即处理一条消息有且只有一次机会能够影响系统状态。目前主流的大数据流处理框架都宣称实现了精确一次处理语义，但这是有限定条件的，即它们只能实现框架内的精确一次处理语义，无法实现端到端的。

这是为什么呢？因为当这些框架与外部消息引擎系统结合使用时，它们无法影响到外部系统的处理语义，所以如果你搭建了一套环境使得 Spark 或 Flink 从 Kafka 读取消息之后进行有状态的数据计算，最后再写回 Kafka，那么你只能保证在 Spark 或 Flink 内部，这条消息对于状态的影响只有一次。但是计算结果有可能多次写入到 Kafka，因为它们不能控制 Kafka 的语义处理。相反地，Kafka 则不是这样，因为所有的数据流转和计算都在 Kafka 内部完成，故 Kafka 可以实现端到端的精确一次处理语义。

可能助力 Kafka 胜出的第二点是它自己对于流式计算的定位。官网上明确标识 Kafka Streams 是一个用于搭建实时流处理的客户端库而非是一个完整的功能系统。这就是说，你不能期望着 Kafka 提供类似于集群调度、弹性部署等开箱即用的运维特性，你需要自己选择适合的工具或系统来帮助 Kafka 流处理应用实现这些功能。

读到这你可能会说这怎么算是优点呢？坦率来说，这的确是一个“双刃剑”的设计，也是 Kafka 社区“剑走偏锋”不正面 PK 其他流计算框架的特意考量。大型公司的流处理平台一定是大规模部署的，因此具备集群调度功能以及灵活的部署方案是不可或缺的要素。但毕竟这世界上还存在着很多中小企业，它们的流处理数据量并不巨大，逻辑也并不复杂，部署几台或十几台机器足以应付。在这样的需求之下，搭建重量级的完整性平台实在是“杀鸡焉用牛刀”，而这正是 Kafka 流处理组件的用武之地。因此从这个角度来说，未来在流处理框架中，Kafka 应该是有一席之地的。

除了消息引擎和流处理平台，Kafka 还有别的用途吗？当然有！你能想象吗，Kafka 能够被用作分布式存储系统。Kafka 作者之一 Jay Kreps 曾经专门写过一篇文章阐述为什么能把

Kafka 用作分布式存储。不过我觉得你姑且了解下就好了，我从没有见过在实际生产环境中，有人把 Kafka 当作持久化存储来用 。

说了这么多，我只想阐述这样的一个观点：Apache Kafka 从一个优秀的消息引擎系统起家，逐渐演变成现在分布式的流处理平台。你不仅要熟练掌握它作为消息引擎系统的非凡特性及使用技巧，最好还要多了解下其流处理组件的设计与案例应用。

Kafka只是消息引擎系统吗?

- Kafka在设计之初就旨在提供三个方面的特性：提供一套API实现生产者和消费者；降低网络传输和磁盘存储开销；实现高伸缩性架构。

- 作为流处理平台，Kafka与其他主流大数据流式计算框架相比，优势有两点：更容易实现端到端的正确性；它自己对于流式计算的定位。

- Apache Kafka是消息引擎系统，也是一个分布式流处理平台。除此之外，Kafka还能够被用作分布式存储系统。不过我觉得你姑且了解下就好了，我从没有见过在实际生产环境中，有人把Kafka当作持久化存储来用 。

开放讨论

你觉得 Kafka 未来的演进路线是怎么样的？如果你是 Kafka 社区的“掌舵人”，你准备带领整个社区奔向什么方向呢？（提示下，你可以把自己想象成 Linus 再去思考）

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。

AI智能总结

Kafka：从消息引擎到分布式流处理平台

Kafka不仅是一款强大的消息引擎系统，更是一个多功能的分布式流处理平台。文章首先介绍了Kafka的发展历程，强调了全局角度理解Kafka的重要性。随后详细阐述了Kafka在LinkedIn内部应用的背景和解决的问题，以及其作为消息引擎系统的初衷。随着不断完善，Kafka逐渐演变为分布式流处理平台，并推出了流处理组件Kafka Streams。相较于其他大数据流式计算框架，Kafka的优势在于实现端到端的正确性和其定位为一个用于搭建实时流处理的客户端库。文章最后展望了Kafka流处理平台的前景，并对其优势进行了深入分析。

参考文章还指出，Kafka除了作为消息引擎和流处理平台外，还可以用作分布式存储系统。作者强调了Kafka从一个优秀的消息引擎系统逐渐演变成分布式流处理平台的观点。因此，读者不仅需要熟练掌握Kafka作为消息引擎系统的特性和使用技巧，还应该多了解其流处理组件的设计与案例应用。这一观点为读者提供了更深入的思考和学习方向。

总之，本文全面介绍了Kafka的技术特点和发展历程，对于想要快速了解Kafka的读者具有重要的参考价值。文章展望了Kafka流处理平台的前景，并对其优势进行了深入分析，为读者提供了更深入的思考和学习方向。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

全部留言 (48)

最新　精选

Michael YZY
2019-06-08

学到了。刚接触， 对一次性处理语义的概念和背后的含义不太明确， 能否结合实例讲解比较一下…

> 作者回复: 举个例子，如果我们使用Kafka计算某网页的PV——我们将每次网页访问都作为一个消息发送的Kafka。PV的计算就是我们统计Kafka总共接收了多少条这样的消息即可。精确一次处理语义表示每次网页访问都会产生且只会产生一条消息，否则有可能产生多条消息或压根不产生消息。

共 3 条评论>　　75

October
2019-06-17

对于kafka streams相对于其他大数据流式计算框架的优势的第一点不是特别理解。spark或者flink读取消息之后再写回kafka，可能会导致多次写入kafka，老师能不能解释一下什么情况下会多次写入kafka?

作者回复: 不用拿Flink或Spark举例。我们就说一个普通的producer程序：producer需要接收到broker发送的response才能认为发送成功，如果response在传输过程中因为网络抖动丢失了或超时了（这种情况非常常见）而broker实际上已经写入了该消息，那么producer就会认为发送失败从而尝试重新发送，这就可能造成同一条消息被发送了多次。

共 11 条评论> 66

清晨吼于林
2019-06-12

老师您好~~~
我了解的：一个partition在一个group内，只能被一个消息者进程消费（一个jvm，启动了一个java进程）。
问题前提：经过分区算法的匹配，A partition 被 B 消费者 消费。
我的问题：在这个B的消费者里面，假如我用多线程消费（多个线程，每个线程维护了一个KafkaConsumer实例。 而不是一个KafkaConsumer然后多个worker线程消费的模式），那这多个线程都能从这个A partition里面取到消息嘛?

作者回复: 同一个组下有多少消费者实例不是看进程数或线程数，而是看创建的KafkaConsumer实例数。所以在你的场景中，B消费者不是一个，而是多个，因为B进程启动了多个线程，而每个线程都维护了单独的KafkaConsumer实例。

共 2 条评论> 26

DarKnight
2019-06-27

胡老师您好！我对于第一点优势那个例子不是很懂，但又很感兴趣。我能否用一个这样的情形去理解呢：

我在spark内部consume了一条数据并要进行有状态的计算，我可以通过roll back确保做到exactly once，当状态计算过程中可以通过捕捉exception从而roll back到初始状态，但状态计算过程中我可能已经将某些结果发送到kafka了（这些结果我并不想重复发送），虽然我可以roll back所有处于spark内部的数据状态，但发送到kafka的所有数据就已经收不回了。

不知道这个例子算不算一种解读呢？谢谢！

作者回复: 嗯嗯，在Spark看来，写入Kafka是一种side effect，它无法控制。所以它无法实现端到端的EOS。Flink 1.4借助了Kafka提供的事务机制来保证E2E EOS，但是没听说Spark也做了这样的改进。

共 2 条评论> 19

钱
2019-08-11

课前思考
kafka除了可以作为一个消息引擎系统，还能用来干什么？这个还真不太清楚，它的核心功能不就是，将消息倒一道手嘛?
课后思考
1：kafka可以作为什么来使用?
1-1：一个分布式消息引擎系统——广泛使用
1-2：一个分布式流处理平台，可以和Storm/Spark/Flink相媲美——越来越多这么玩，根据老师的评论回复，感觉kafka更是一个分布式流处理库。
1-3：一个分布式存储系统——很少使用，关键增删改查的效率好不？如果挺好，也可以这么玩吧！

如果我是kafka的掌舵人，我会逐渐丰富kafka的生态圈，把kafka弄得和Spring全家桶类似，以后的ABC把kafka家族的程员作为标配。

2：啥是流处理?
是指实时处理无限数据集的数据的一种处理方式嘛?
3：啥是批处理?
是指一次处理一批数据，且此数据的集合是有限的?
4：流处理和批处理，没理解，kafka作为分布式流处理平台的优势也没理解？看评论，流处理的数据集是无限数据集，那岂不是永远处理不完，直到天荒地老?
5：数据正确性不足是什么意思？会丢数据？没明白和数据收集的方式的逻辑是什么?

计算机我的理解，就是处理数据的，处理数据无非是针对数据的存储转发增删改查存分析统计，然后就是挖空心思加快速度。
感觉不该如此难以理解😊，一图胜千言，希望后面看到老师有图有真相。

作者回复: 嗯嗯，记下了您的建议

17

平叔叔
2019-09-22

在这样的需求之下，搭建重量级的完整性平台实在是“杀鸡焉用牛刀，的意思中小企业使用Kafka 不用配套提供集群调度、弹性部署?

作者回复: 你不要搭建多套这样重量级的系统，只需要一套Kafka集群就可以。并不是说Kafka集群不需要运维管理

7

Shane
2019-06-15

老师，能举个例子说明下流出来和批处理的区别吗?
目前我的理解就是批处理是一次请求中包含多条消息？然后消费者取出这一整个请求内容进行处理消费。流处理就是每个请求每次只发送一条消息，所以消费者也只能每次消费一条?

感觉自己理解的应该不怎么正确呢？网络上的解释也是非常虚，想看看老师有啥指导的吗?

作者回复: 流处理和批处理的区别是前者主要用于处理无限数据集（unbound data set）

共 2 条评论>

7

你好旅行者
2019-06-12

关于【但是计算结果有可能多次写入到 Kafka，因为它们不能控制 Kafka的语义处理】。我想问老师，Kafka不是在0.11版本实现了exactly once，保证一条消息只会被消费一次吗，为什么说计算结果还有可能会被多次写入到Kafka呢?

作者回复: 嗯嗯，这说的就是0.11之前的故事。事实上，Apache Flink从1.4开始推出了支持E2E Exactly-Once语义的两阶段SinkFunction。它用的就是Kafka 0.11的事务

7

EricJones
2019-06-22

我又仔细意会了一下，流处理大概已经懂了，但是批处理的正确性到底体现在哪里。还是不知道。

作者回复: 假设我们统计单词计数。如果不出现问题，对于相同的有限输入（bounded dataset）批处理是不是总是能够得到相同的输出?


5

东方奇骥
2019-06-16

老师，请问一下，kafka相比于rabbitmq和activemq作为消息引擎系统的优势是什么呢。就是文中所说的消息正确性吗?

作者回复: 如果和rabbitmq和activemq相比，Kafka还是以消息引擎的角色。目前Kafka消息引擎单方面只能提供at least once处理语义，无法实现精确一次的消息交付语义。

另外，正确性一般用在数据计算领域。在消息引擎中我们更多的是谈它的消息交付语义（message delivery semantics）

共 3 条评论>


5