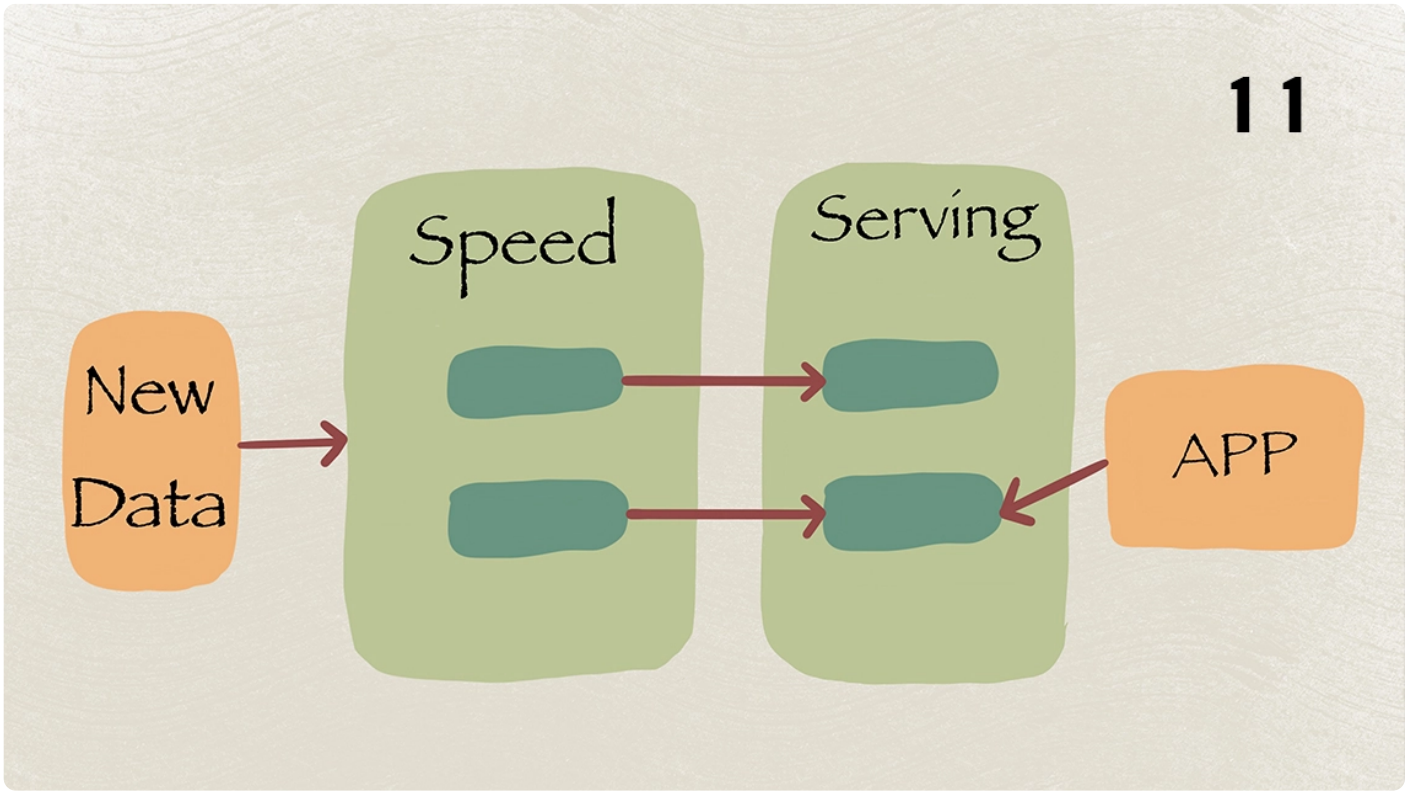
11 | Kappa架构：利用Kafka锻造的屠龙刀

2019-05-10 蔡元楠 来自北京

《大规模数据处理实战》

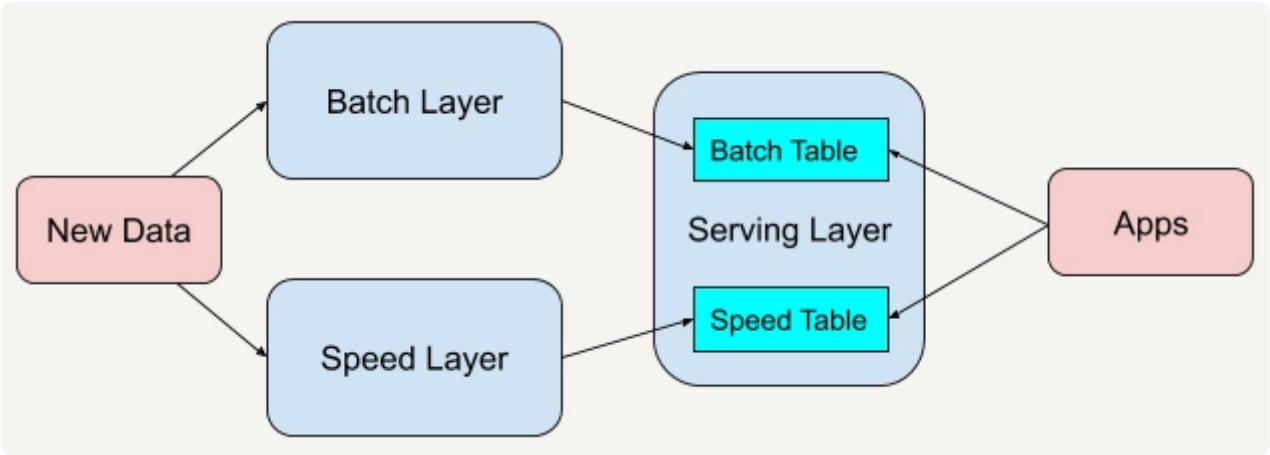


你好，我是蔡元楠。

今天我要分享的主题是 Kappa 架构。

同样身为大规模数据处理架构，Kappa 架构这把利用 Kafka 锻造的“屠龙刀”，它与 Lambda 架构的不同之处在哪里呢?

上一讲中，我讲述了在处理大规模数据时所用到经典架构，Lambda 架构。我先来带你简要回顾一下。



Lambda 架构结合了批处理和流处理的架构思想，将进入系统的大规模数据同时送入这两套架构层中，分别是批处理层（Batch Layer）和速度层（Speed Layer），同时产生两套数据结果并存入服务层。

批处理层有着很好的容错性，同时也因为保存着所有的历史记录，使产生的数据集具有很好的准确性。速度层可以及时地处理流入的数据，因此具有低延迟性。最终服务层将这两套数据结合，并生成一个完整的数据视图提供给用户。

Lambda 架构也具有很好的灵活性，你可以将现有开源生态圈中不同的平台套入这个架构，具体请参照上一讲内容。

Lambda 架构的不足

虽然 Lambda 架构使用起来十分灵活，并且可以适用于很多的应用场景，但在实际应用的时候，Lambda 架构也存在着一些不足，主要表现在它的维护很复杂。

使用 Lambda 架构时，架构师需要维护两个复杂的分布式系统，并且保证他们逻辑上产生相同的结果输出到服务层中。

举个例子吧，我们在部署 Lambda 架构的时候，可以部署 Apache Hadoop 到批处理层上，同时部署 Apache Flink 到速度层上。

我们都知道，在分布式框架中进行编程其实是十分复杂的，尤其是我们还会针对不同的框架进行专门的优化。所以几乎每一个架构师都认同，Lambda 架构在实战中维护起来具有一定的复

杂性。

那要怎么解决这个问题呢？我们先来思考一下，造成这个架构维护起来如此复杂的根本原因是什么呢？

维护 Lambda 架构的复杂性在于我们要同时维护两套系统架构：批处理层和速度层。我们已经说过了，在架构中加入批处理层是因为从批处理层得到的结果具有高准确性，而加入速度层是因为它在处理大规模数据时具有低延时性。

那我们能不能改进其中某一层的架构，让它具有另外一层架构的特性呢？

例如，改进批处理层的系统让它具有更低的延时性，又或者是改进速度层的系统，让它产生的数据视图更具准确性和更加接近历史数据呢？

另外一种在大规模数据处理中常用的架构——Kappa 架构（Kappa Architecture），便是在这样的思考下诞生的。

Kappa 架构

Kappa 架构是由 LinkedIn 的前首席工程师杰伊·克雷普斯（Jay Kreps）提出的一种架构思想。克雷普斯是几个著名开源项目（包括 Apache Kafka 和 Apache Samza 这样的流处理系统）的作者之一，也是现在 Confluent 大数据公司的 CEO。

克雷普斯提出了一个改进 Lambda 架构的观点：

> 我们能不能改进 Lambda 架构中速度层的系统性能，使得它也可以处理好数据的完整性和准确性问题呢？我们能不能改进 Lambda 架构中的速度层，使它既能够进行实时数据处理，同时也有能力在业务逻辑更新的情况下重新处理以前处理过的历史数据呢？

他根据自身多年的架构经验发现，我们是可以做到这样的改进的。

在前面 Publish–Subscribe 模式那一讲中，我讲到过像 Apache Kafka 这样的流处理平台是具有永久保存数据日志的功能的。通过平台的这一特性，我们可以重新处理部署于速度层架构

中的历史数据。

下面我就以 Apache Kafka 为例来讲述整个全新架构的过程。

第一步，部署 Apache Kafka，并设置数据日志的保留期（Retention Period）。这里的保留期指的是你希望能够重新处理的历史数据的时间区间。

例如，如果你希望重新处理最多一年的历史数据，那就可以把 Apache Kafka 中的保留期设置为 365 天。如果你希望能够处理所有的历史数据，那就可以把 Apache Kafka 中的保留期设置为“永久（Forever）”。

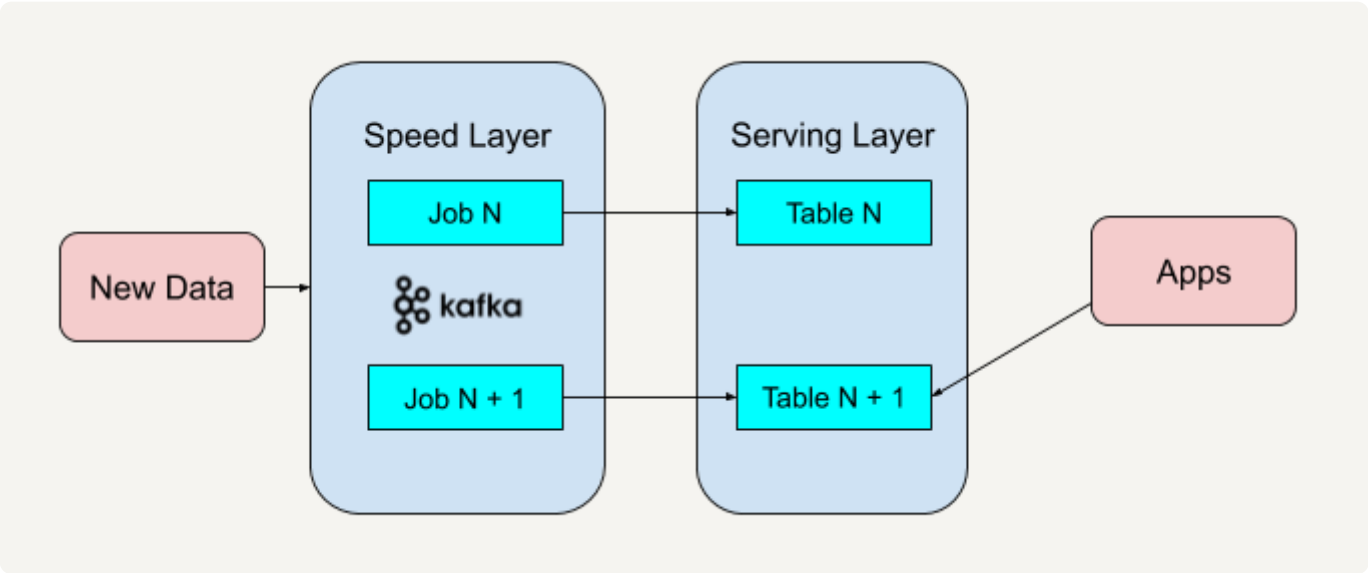
第二步，如果我们需要改进现有的逻辑算法，那就表示我们需要对历史数据进行重新处理。

我们需要做的就是重新启动一个 Apache Kafka 作业实例（Instance）。这个作业实例将重头开始，重新计算保留好的历史数据，并将结果输出到一个新的数据视图中。我们知道 Apache Kafka 的底层是使用 Log Offset 来判断现在已经处理到哪个数据块了，所以只需要将 Log Offset 设置为 0，新的作业实例就会重头开始处理历史数据。

第三步，当这个新的数据视图处理过的数据进度赶上了旧的数据视图时，我们的应用便可以切换到从新的数据视图中读取。

第四步，停止旧版本的作业实例，并删除旧的数据视图。

这个架构就如同下图所示。



与 Lambda 架构不同的是，Kappa 架构去掉了批处理层这一体系结构，而只保留了速度层。你只需要在业务逻辑改变又或者是代码更改的时候进行数据的重新处理。

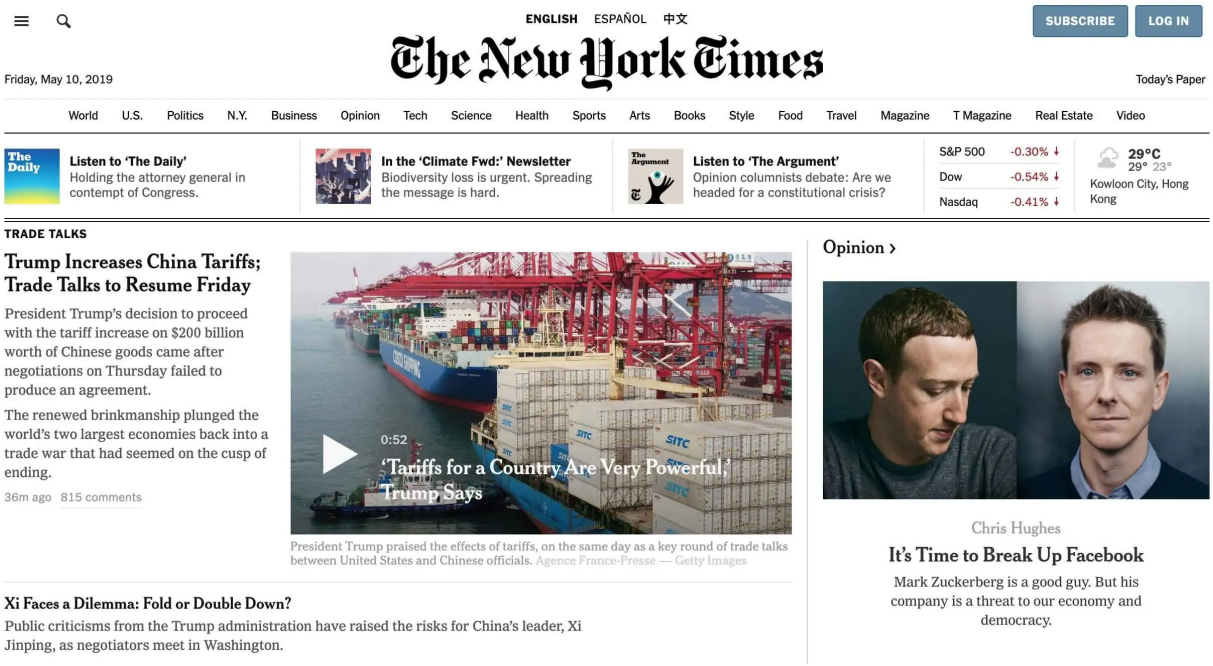
当然了，你也可以在我上面讲到的步骤中做一些优化。

例如不执行第 4 步，也就是不删除旧的数据视图。这样的好处是当你发现代码逻辑出错时可以及时回滚（Roll Back）到上一个版本的数据视图中去。又或者是你想在服务层提供 A/B 测试，保留多个数据视图版本将有助于你进行 A/B 测试。

在介绍完 Kappa 架构的概念后，我想通过一个实战例子，来和你进一步学习 Kappa 架构是如何应用在现实场景中的。

《纽约时报》内容管理系统架构实例

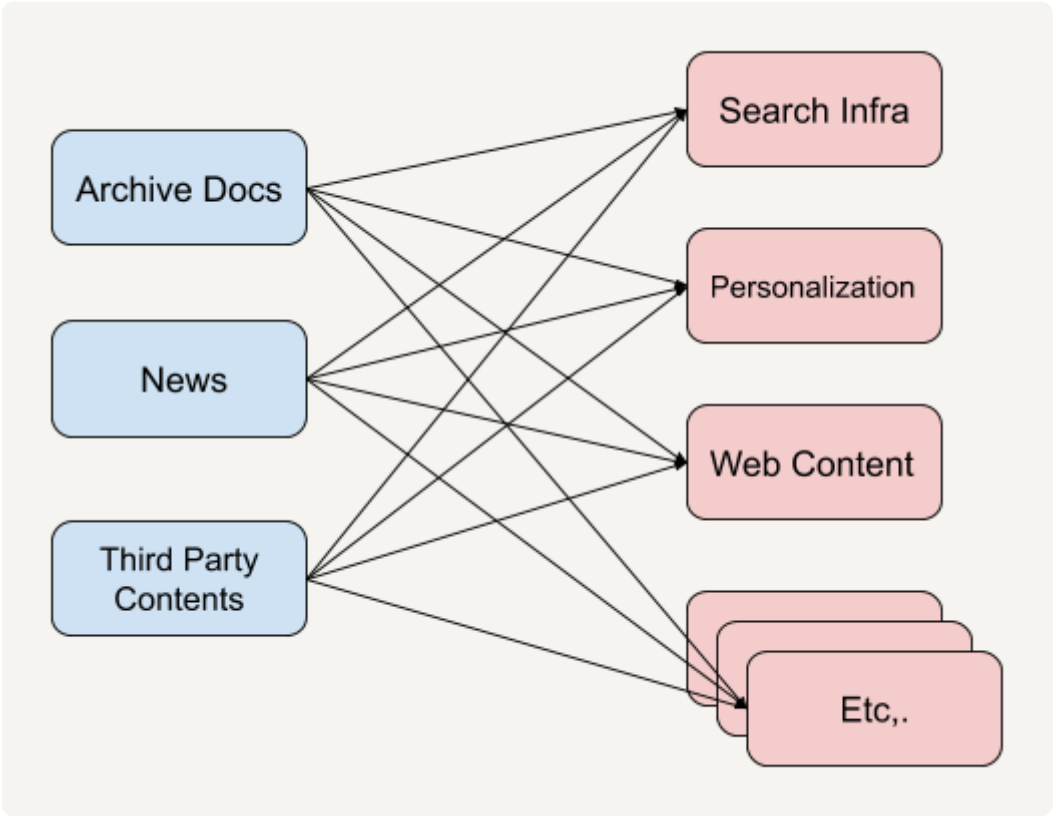
《纽约时报》是一个在美国纽约出版，在整个美国乃至全世界都具有相当影响力的日报。



《纽约时报》的内容管理系统收集、保存着各种各样来源的文档。这些文档有从第三方收集来的资料，也有自己报社编辑部所撰写的故事。当你访问《纽约时报》网站主页时，甚至能够查到 162 年前的新闻报道。

可想而知，要处理这么大规模的内容，并将这些内容提供于在线搜索、订阅的个性化推荐以及前端应用程序等等的服务，是一个非常棘手的任务。

我们先来看看他们曾经使用过的一个老式系统架构。



我们可以看到，这种系统架构是一种相当典型的基于 API 的架构，无论是在系统调度上还是使用场景上都存在着自身的不足。我来给你举一些例子。

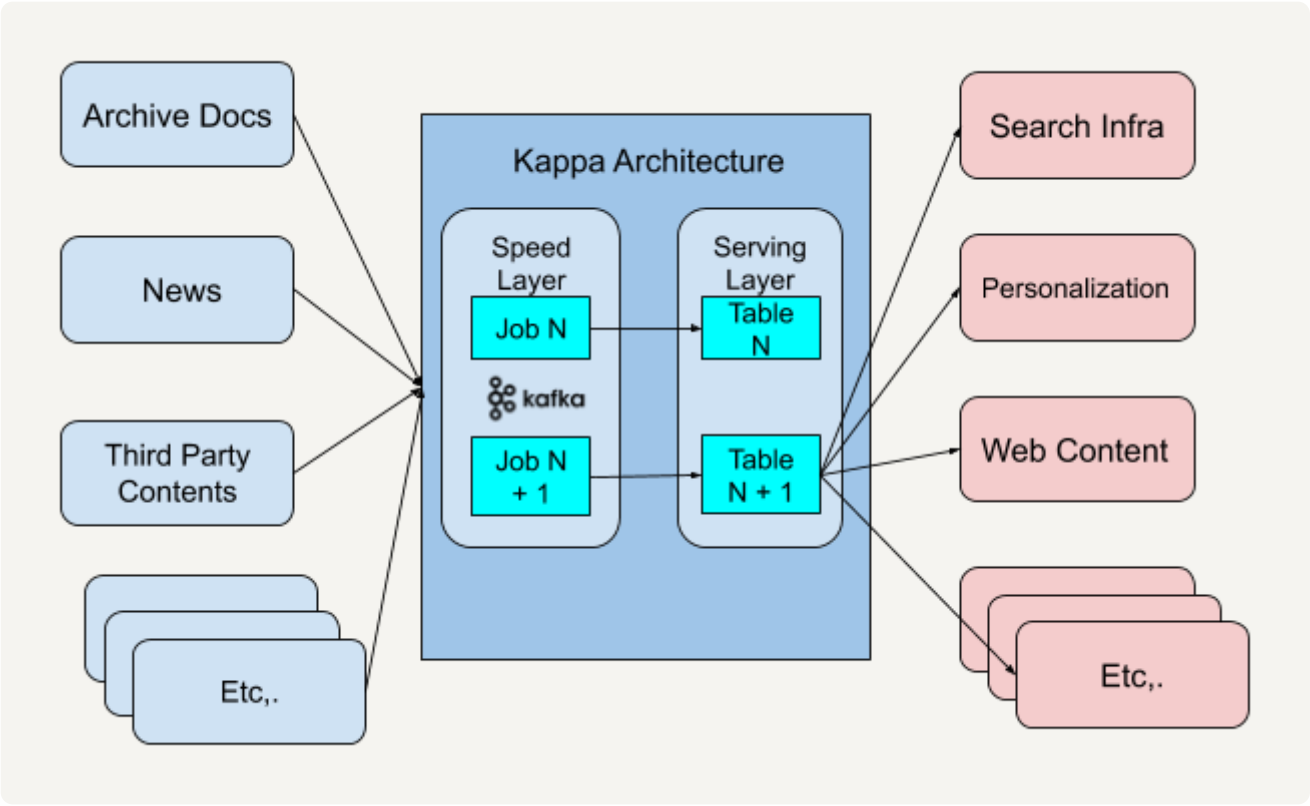
不同的内容 API 可能由不同的团队开发，从而造成 API 有不同的语义，也有可能需要不同的参数。

调用不同 API 所得到的内容结果可能有不同的格式，在应用端需要重新进行规范化（Standardization）。

如果客户端上会实时推送一些新的热点新闻或者突发新闻（Breaking News），那么在上述基于 API 的架构中，想要实时获知新闻的话，就需要让客户端不停地做轮询操作（Polling）。轮询操作在这里指的是客户端定期地重复调用系统 API 来查看是否有新的新闻内容，这无疑增加了系统的复杂性。

客户端很难访问以前发布过的内容。即便我们知道这些已发布过的新闻列表需要从哪里获取，进行 API 调用去检索每个单独的新闻列表还是需要花很长的时间。而过多的 API 调用又会给服务器产生很大的负荷。

那现在你再来看看当《纽约时报》采取了 Kappa 架构之后，新的系统架构是什么样的。



首先，Kappa 架构在系统调度这个层面上统一了开发接口。

你可以看到，中间的 Kappa 架构系统规范好了输入数据和输出数据的格式之后，任何需要传送到应用端的数据都必须按照这个接口输入给 Kappa 架构系统。而所有的应用端客户都只需要按照 Kappa 架构系统定义好的输出格式接收传输过来的数据。这样就解决了 API 规范化的问题。

我们再来看看增加了中间一层 Kappa 架构之后数据传输速度上的变化。

因为 Apache Kafka 是可以实时推送消息数据的，这样一来，任何传输进中间 Kappa 架构的数据都会被实时推送到接收消息的客户端中。这样就避免了在应用层面上做定期轮询，从而减少了延时。而对于重新访问或者处理发布过的新闻内容这一问题，还记得我之前和你讲述过的 Kafka 特性吗？只需要设置 Log Offset 为 0 就可以重新读取所有内容了。

在讲述完 Kappa 架构和它的应用实例之后，我想强调一下，Kappa 架构也是有着它自身的不足的。

因为 Kappa 架构只保留了速度层而缺少批处理层，在速度层上处理大规模数据可能会有数据更新出错的情况发生，这就需要我们花费更多的时间在处理这些错误异常上面。

还有一点，Kappa 架构的批处理和流处理都放在了速度层上，这导致了这种架构是使用同一套代码来处理算法逻辑的。所以 Kappa 架构并不适用于批处理和流处理代码逻辑不一致的场景。

小结

在最近两讲中，我们学习到了 Lambda 架构和 Kappa 架构这两种大规模数据处理架构，它们都各自有着自身的优缺点。我们需要按照实际情况来权衡利弊，看看我们在业务中到底需要使用到哪种架构。

如果你所面对的业务逻辑是设计一种稳健的机器学习模型来预测即将发生的事情，那么你应该优先考虑使用 Lambda 架构，因为它拥有批处理层和速度层来确保更少的错误。

如果你所面对的业务逻辑是希望实时性比较高，而且客户端又是根据运行时发生的实时事件来做出回应的，那么你就应该优先考虑使用 Kappa 架构。

思考题

在学习完 Lambda 架构和 Kappa 架构之后，你能说出 Kappa 架构相对 Lambda 架构的优势吗？

欢迎你把答案写在留言区，与我和其他同学一起讨论。

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 (44)

程序设计的艺术
2019-05-10

有几个地方没太明白：
1.批处理数据具有很高的准确性，实时运算处理就没有？
2.关于数据错误，两者应该都有吧？数据错误后两者的处理不一样吗？比如10个任务中的3个有错误，批处理和实时处理都应该可以找到3个任务重新计算吧？
为什么实时运算需要完全从头计算所有任务？
谢谢

作者回复: 谢谢你的提问！
1. 批处理和实时运算中的处理肯定都具有准确性的，只是这里所说的高准确性是批处理结果相对于流处理结果而言的，因为毕竟批处理所处理的数据是历史数据。举个例子，假设我们拥有一个人近几年的上班目的地的历史数据，大部分时间这个人上班都在A地，只有少部分时间在B地。那批处理层所处理完这些历史数据之后可能会判断出这个人的常规上班地点是A，出差地点是B。如果是实时处理的话，可能刚好拿到的数据就是出差的那几天地点B，那实时处理的判断可能会判断出这个人的常规上班地点是B了。所以相对而言，批处理具有高准确性。
2. 无论是数据出错或者还是逻辑出错，批处理和实时处理肯定都会发生的。流处理的处理方式并不是按照任务为单位来计算的，通常都是把每一份数据当作是一个消息，流入的消息处理过了就不会再回头了。批处理的处理方式就是不断的有定时任务去重新处理所有历史数据。在Kappa架构下，除非你的logging做得非常好，能够知道在从哪一个数据时间点上出错了，那就把offset调回那个点重新从那个点开始重新计算。如果是逻辑有所改变了，那肯定是要全部数据从头完全重新计算了。

共 2 条评论 >　　27

朱同学
2019-05-11

我感觉这是用队列代替了hdfs

作者回复: 谢谢你的留言！哈哈，可以这么理解，而且这个队列还必须具有重新处理所有历史数据的能力。

共 3 条评论 >　　22

:)
2019-05-10

1.kappa架构使用更少的技术栈，实时和历史部分都是同一套技术栈。lambda架构为了解决历史部分和实时部分可能会使用不同的技术栈。

2.kappa架构使用了统一的处理逻辑。而lambda架构分别为历史和实时部分使用了两套逻辑。

一旦需求变更，两套逻辑都要同时变更。

3.kappa架构具有流式处理的特点和优点。比如可以具有多个订阅者，比如具有更高的吞吐量。

作者回复: 谢谢你的留言！不错的总结!

19

Rainbow
2019-05-10

spark算kappa架构吗？批处理 流处理一套

作者回复: 谢谢你的提问！Spark不算Kappa架构，Spark是有能力可以处理批处理和流处理，我们可以利用Spark搭建Kappa架构出来，但是它本身不具备这种架构的思想在里面。这就好比我们可以借助编程语言实现一些算法，但是我们不会说这种编程语言就属于这种算法一样。

17

罗钛龙
2019-05-17

有个问题不明白，实时数据量很大的化，每次都从头计算，是否削弱了速度层实时性的特点，这样不和初心违背了么？

作者回复: 谢谢你的提问！Kappa架构是具有从头计算的能力，但是这个架构并不太适用于需要经常重新计算历史数据的应用场景。当我们发现有重大逻辑错误出现或者修改的时候才会从头计算所有的数据。

共 2 条评论>

10

我只是想改个名字
2019-05-20

在我看来，Kappa架构和lambda架构没有绝对的谁好，就拿我们现在的业务场景，MySQL同步至tidb平台，原数据进kafka就有可能有问题，而我们又是金融公司，就只能选择lambda架构随时对数据进行校验，对最终数据进行纠正。

9

孙稚昊
2019-05-10

这样批和流的压力就全压到kafka 上了，对kafka并发的要求也非常高，应该只有kafka 能做到这件事了吧

作者回复: 谢谢你的留言！是的呢，现在只有用Kafka来实现Kappa架构。

8

CoderLean
2019-05-10

不可能啊，如果要保存长久的数据，那么kafka的集群容量得有多大，按每天就一个t来说，加上默认副本3，那一年要存的数据量就得很多了。 这还可能只是其中一个业务。国内有小公司敢这样做吗

作者回复: 谢谢你的留言！你说得也没有错，现在硅谷这边实践得比较多的可能就Linkedin或者Confluent了。

共 2 条评论>

7

aof
2019-05-10

如果批处理比较多的话，每次都从kafka的earlist offset消费的话，第一会耗费很长很长时间，而且消费者如果资源不够多，会导致任务堆积的吧。所以kappa不适合批处理多的架构。

Kappa架构因为整合了批处理层和速度层，优势就是:
1. 实时性比较高，适合对实时性要求高的场景
2. 业务逻辑可以使用统一的API来编写，那么对于之后的业务需求变更和代码维护都比较友好

作者回复: 谢谢你的留言总结！是的，现在Kappa架构用得比较多的是Confluent的数据平台，不过他们也没有放出Benchmark，具体和Lambda相比性能差距多少还不确定。不过我赞同你说kappa不适合批处理多的架构，毕竟如果常常要重做批处理的话，性能肯定会受影响的。

6

夷，这也可以
2019-06-19

蔡老师好！看了之后有几点疑问：Lamabda基本上很好理解，Kappa还是不理解。目前的理解

是这样的
1、Lamabda是离线每天计算T+1数据 联合 当天实时处理的数据；T+1的数据会更新覆盖掉昨天实时数据。
2、Kappa架构是从设定的时间数据开始，对每一条数据进行处理并和以前实时计算的结果数据聚会出结果，不断实时更新数据，直至实时处理的数据是目前最新的。
a、Kappa从开发来说只有实时逻辑，不涉及T+1的批量逻辑了。
b、数据的实时性来说，Kappa和Lambda都能拿到当前最新的全量"结果"数据。
c、如果有需求变更了，2种架构其实都要开发。
d、如果数据发声错误时，Lambda排查的时候相对Kappa会容易些，但是Kappa也可以通过业务处理逻辑对一个周期（比如天、周、月）的结果进行保存来使得查错和Lambada一样。对于纠错更正来说，如果业务逻辑比较简单只需在最后的迭代中修正即OK，那么2者也没有什么区别。但是如果业务逻辑比较复杂，不能简单的修改最后迭代结果，而需要从新迭代的情况，那么Lambda相对Kappa要容易，毕竟批处理的迭代次数相对较少。实际情况不能简单的修改最后迭代结果的情况应该毕竟少，而且一般发生错误都是就近的也就是当前时间不长，矫正比较容易。而且发现久远错误更正的情况也应该毕竟小。
这样来看Kappa相对Lambda还有些优势的，不过从稳定性来看Lambda要强点。
见识较少。不知道对不对

5

又双叒叕是一年啊
2019-05-16

处理这种大数据量有窗口期的比如近30天的任务聚合计算可以用这种模式吗？是需要用kafka stream？每天都需要计算一次近30天的任务计算全量数据很大都是离线日志产生的数据

作者回复: 谢谢你的提问！按照你的说法你的应用需求是需要定时处理离线数据的，Kappa还是不太适合这种应用场景。这种应用场景可以用crontab加batch job完成。

4

Zoe
2019-05-29

老师，请问一下，纽约时报这个例子，是不是每次只处理delta部分而不是把log offset设成0更好一些?
我粗浅的理解是可以把batch layer想成一个cache，感觉这种基于time series的每次只需要处理new data的部分再把结果和之前的cache聚合在一起就可以了?
还是说业界普遍的做法都是从头开始处理，因为相较于找delta考虑overlap的困难就不那么意额外的处理时间和机器运行的成本?

作者回复: 谢谢你的提问！你的理解没有错，一般来说是只处理delta部分就可以了。需要从头开始处理的场景一般都是在发现之前的逻辑有重大的错误或者说新加了一些字段需要backfill以前的数据。

3

王鹏
2020-04-07

个人感觉像最近出现的有些OLAP的数据，比如TIDB和Clickhouse，可能提供了另一种思路，这个存储引擎即支持批数据有支持流数据，并能提供高效的查询，我们最近的项目就是使用kafka+flink+Clickhouse这种，感觉一个OLAP工具是不是可以解决一些问题。

2

YYY
2019-11-24

https://www.cnblogs.com/xiaodf/p/11642555.html 我在这个博文里面看到了老师讲课一样的内容 是19年10月份发布的 还有微信公众号

2

珅剑
2019-05-11

新数据视图处理过得数据进度有可能赶上旧的数据视图吗？原先的数据视图应该也是实时不断更新的

作者回复: 谢谢你的提问！这个问题问得挺好的，我觉得还是要看应用场景吧。某些场景例如IoT这种，永远有大量数据流入的，而且实时处理IoT数据的场景可能并不需要处理到所有的历史记录，可能kappa架构就不一定合适用了。这种情况下即便逻辑需要修改，可能会采取部分历史数据dual-write，新数据直接switch到新逻辑上。说白了能够用上kappa架构的场景，肯定是历史数据可以赶得上新数据视图的。

2

jasine
2019-05-11

老师您好，麻烦请教下 如果实时与历史批量流程合在一起 那么重跑的时候kafka offset置成0 到latest这段时间是不是实时就无法提供服务了 怎么解决这个问题呢 感谢

作者回复: 谢谢你的提问！这个问题可以这么看，因为重跑Kafka数据的是一个新的实例，它不影响现在正在运行的Kafka实例，所以说这段时间还是可以继续提供服务的。而不同的Kafka实例所产生的数据视图是不同版本的，只有当新的实例数据视图赶上旧的实例时，我们才会进行数据视图的切换。

2

tonyzhang
2019-08-16

老师，我想问一下，像lambda架构分成批处理和实时处理的两层，最终都是输出到对应的结果表供应用层分析；那为什么不能先进行实时计算，再把实时计算的结果写入到批处理的表中，这样批处理的数据就会不断累加，也能达到分析历史整体数据的目的，同时计算任务只需要实时计算的任务，批处理的数据是通过同步的形式进行，维护成本也能降低，不知道我的理解对不对?

作者回复: 这就是kappa的理念

1

滩涂曳尾
2019-06-30

一句话心得:
kappa架构主要就是利用kafka做流批一体化，得益于kafka可以方便地把[任意历史时刻, 当前时刻]的数据"框进去"——
1. kafka保留时间: 滑动窗口size
2. kafka LogOffset: 滑动窗口offset

1

西北偏北
2019-06-27

基于kafka可以永久缓存数据的特性，将kafka当作一个存储引擎

基于kafka可以通过offset回溯回溯的特点来基于推送的获取和处理历史数据

1

科学Jia
2019-05-11

老师，女同学举手提问：我不太理解例子中提到重新计算历史数据得到新的视图，这会对实时

数据产生什么影响么？或者说实时数据需要依赖批处理的视图么？我还没法把批处理的结果和实时处理结果相结合，能否再具体一点呢？

作者回复: 谢谢你的提问！男同学坦然回答：其实Kappa架构和Lambda架构很大不同的一点是Kappa架构没有了批处理层这一概念的。所以在Kappa架构中，并没有批处理结果和实时处理结果相结合这么一说。一般来说Kappa架构不太适用于需要经常处理历史数据的场景，但Kappa架构具备的能力是，如果你发现逻辑出错了，它有能力重新处理之前处理过的数据。而你所问到的重新处理历史数据得到的新视图，它是属于另外一个数据版本了，和老版本的数据视图是没有任何关系的，所以不会对实时数据产生影响。你可以把它们想象成两个不同的平行宇宙，虽然拥有了同样的数据，但处理起来互不干涉对方的世界。可能这个比喻不太恰当，如果还有不懂的话也欢迎你继续留言提问！


1