

02 | MapReduce后谁主沉浮：怎样设计下一代数据处理技术？

蔡元楠 2019-04-19



00:00

讲述：蔡元楠 大小：11.51M

12:34

在上一讲中，我们介绍了 2014 年之前的大数据历史，也就是 MapReduce 作为数据处理的默认标准的时代。重点探讨了 MapReduce 面对日益复杂的业务逻辑时表现出的不足之处，那就是：1. 维护成本高；2. 时间性能不足。

同时，我们也提到了 2008 年诞生在 Google 西雅图研发中心的 FlumeJava，它成为了 Google 内部的数据处理新宠。

那么，为什么是它扛起了继任 MapReduce 的大旗呢？

要知道，在包括 Google 在内的硅谷一线大厂，对于内部技术选择是非常严格的，一个能成为默认方案的技术至少满足以下条件：

1. 经受了众多产品线，超大规模数据量例如亿级用户的考验；
2. 自发地被众多内部开发者采用，简单易用而受开发者欢迎；
3. 能通过内部领域内专家的评审；
4. 比上一代技术仅仅提高 10% 是不够的，必须要有显著的比如 70% 的提高，才能够说服整个公司付出技术迁移的高昂代价。就看看从 Python 2.7 到 Python 3 的升级花了多少年了，就知道在大厂迁移技术是异常艰难的。

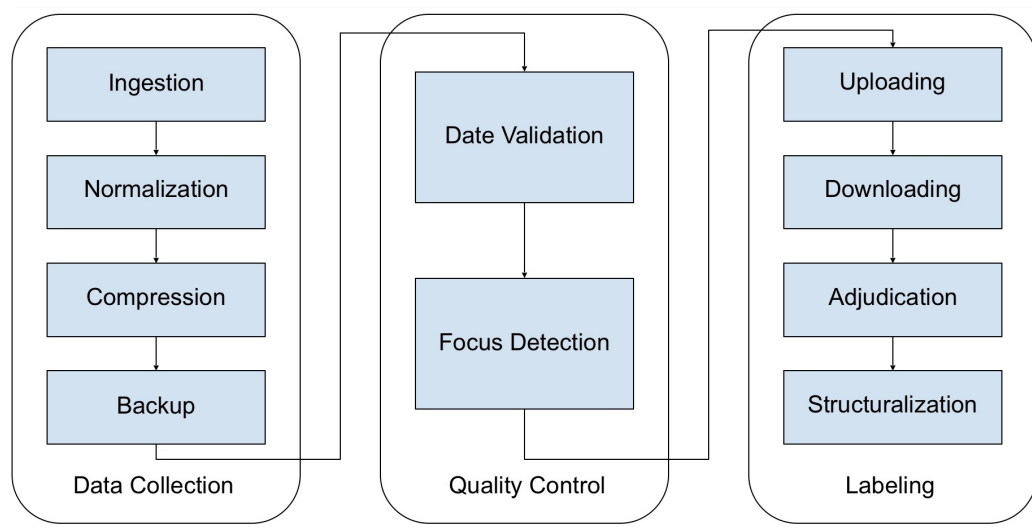
今天这一讲，我不展开讲任何具体技术。

我想先和你一起设想一下，假如我和你站在 2008 年的春夏之交，在已经清楚了 MapReduce 的现有问题的情况下，我们会怎么设计下一代大规模数据处理技术，带领下一个十年的技术革新

呢？

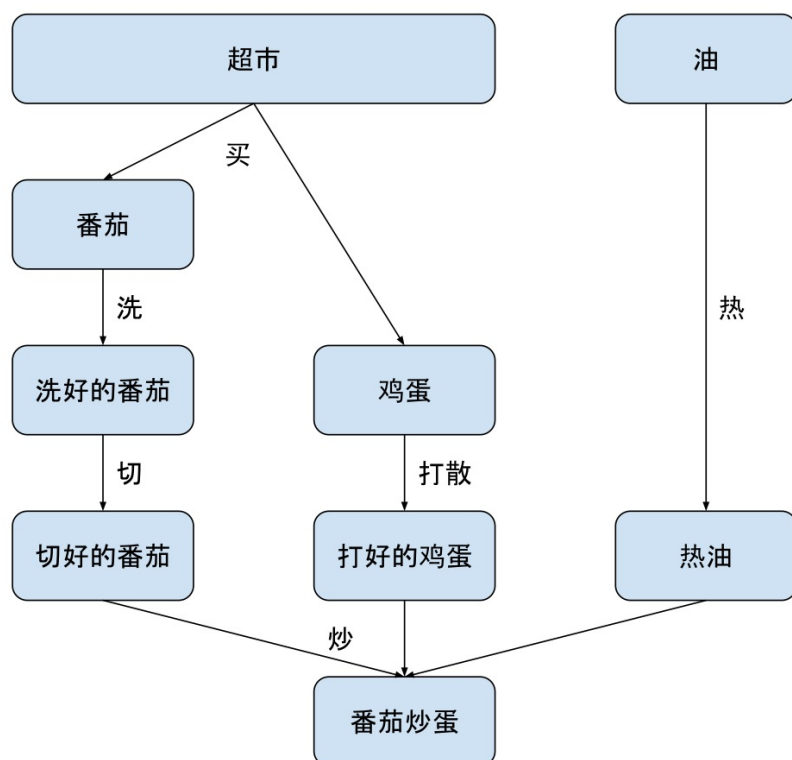
我们需要一种技术抽象让多步骤数据处理变得易于维护

上一讲中我提到过，维护协调多个步骤的数据处理在业务中非常常见。



像图片中这样复杂的数据处理在 MapReduce 中维护起来令人苦不堪言。

为了解决这个问题，作为架构师我们或许可以用有向无环图（DAG）来抽象表达。因为有向图能为多个步骤的数据处理依赖关系，建立很好的模型。如果你对图论比较陌生的话，可能现在不知道我在说什么，你可以看下面一个例子，或者复习一下极客时间的《数据结构与算法之美》。



西红柿炒鸡蛋这样一个菜，就是一个有向无环图概念的典型案例。

比如看这里面番茄的处理，最后一步“炒”的步骤依赖于切好的番茄、打好的蛋、热好的油。而切好的番茄又依赖于洗好的番茄等等。如果用 MapReduce 来实现的话，在这个图里面，每一个箭头都会是一个独立的 Map 或 Reduce。

为了协调那么多 Map 和 Reduce，你又难以避免会去做很多检查，比如：番茄是不是洗好了，鸡蛋是不是打好了。

最后这个系统就不堪重负了。

但是，如果我们用有向图建模，图中的每一个节点都可以被抽象地表达成一种通用的**数据集**，每一条边都被表达成一种通用的**数据变换**。如此，你就可以用**数据集**和**数据变换**描述极为宏大复杂的数据处理流程，而不会迷失在依赖关系中无法自拔。

我们不想要复杂的配置，需要能自动进行性能优化

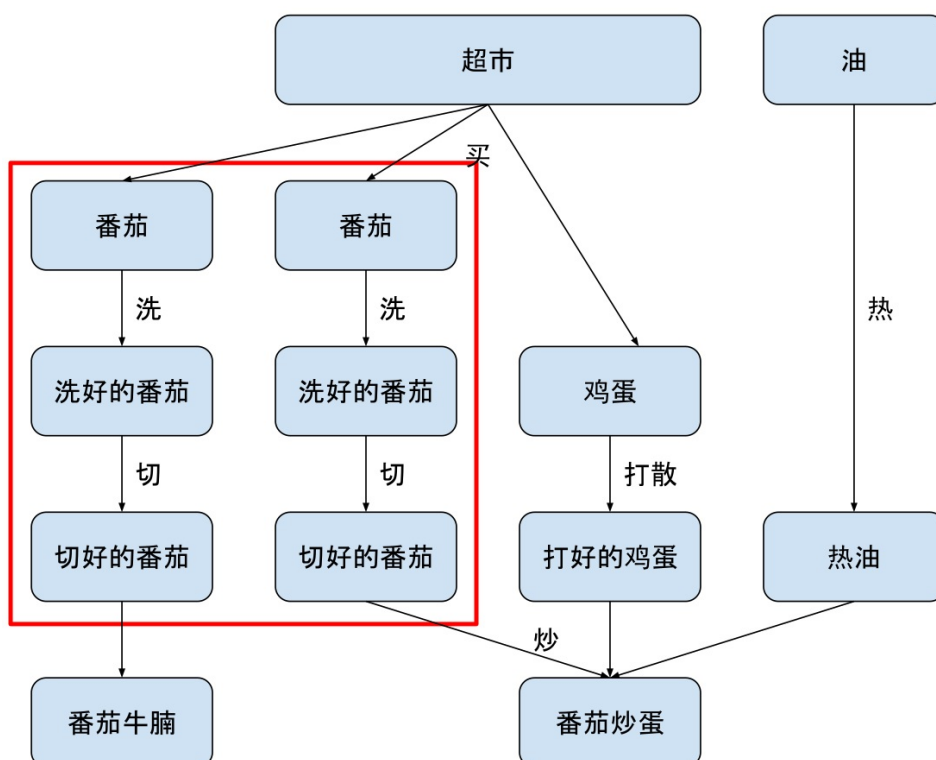
上一讲中提到，MapReduce 的另一个问题是，配置太复杂了。以至于错误的配置最终导致数据处理任务效率低下。

这种问题怎么解决呢？很自然的思路就是，如果人容易犯错，就让人少做一点，让机器多做一点呗。

我们已经知道了，得益于上一步中我们已经用有向图对数据处理进行了高度抽象。这可能就能成为我们进行自动性能优化的一个突破口。

回到刚才的番茄炒鸡蛋例子，哪些情况我们需要自动优化呢？

设想一下，如果我们的数据处理食谱上又增加了番茄牛腩的需求，用户的数据处理有向图就变成了这个样子了。



理想的情况下，我们的计算引擎要能够自动发现红框中的两条数据处理流程是重复的。它要能把两条数据处理过程进行合并。这样的话，番茄就不会被重复准备了。

同样的，如果需求突然不再需要番茄炒蛋了，只需要番茄牛腩，在数据流水线的预处理部分也应该把一些无关的数据操作优化掉，比如整个鸡蛋的处理过程就不应该在运行时出现。

另一种自动的优化是计算资源的自动弹性分配。

比如，还是在番茄炒蛋这样一个数据处理流水线中，如果你的规模上来了，今天需要生产 1 吨的番茄炒蛋，明天需要生产 10 吨的番茄炒蛋。你发现有时候是处理 1000 个番茄，有时候又是 10000 个番茄。如果手动地去做资源配置的话，你再也配置不过来了。

我们的优化系统也要有可以处理这种问题的弹性的劳动力分配机制。它要能自动分配，比如 100 台机器处理 1000 个番茄，如果是 10000 个番茄那就分配 1000 台机器，但是只给热油 1 台机器可能就够了。

这里的比喻其实是很粗糙也不精准的。我想用这样两个例子表达的观点是，在数据处理开始前，我们需要有一个自动优化的步骤和能力，而不是按部就班地就把每一个步骤就直接扔给机器去执行了。

我们要能把数据处理的描述语言，与背后的运行引擎解耦合开来

前面两个设计思路提到了很重要的一个设计就是有向图。

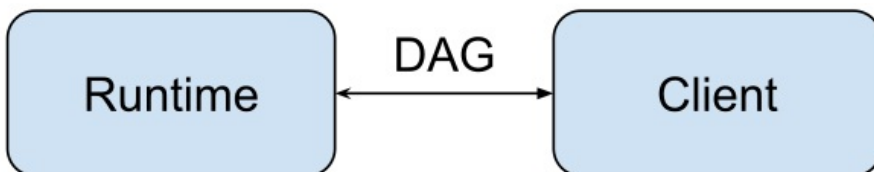
用有向图进行数据处理描述的话，实际上**数据处理描述语言**部分完全可以和后面的**运算引擎**分离了。有向图可以作为**数据处理描述语言**和**运算引擎**的前后端分离协议。

举两个你熟悉的例子可能能更好理解我这里所说的前后端分离（client-server design）是什么意思：

比如一个网站的架构中，服务器和网页通过 HTTP 协议通信。

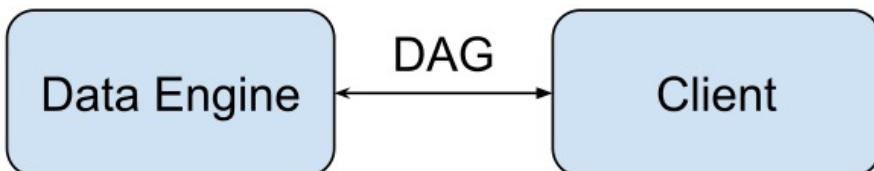


比如在 TensorFlow 的设计中，客户端可以用任何语言（比如 Python 或者 C++）描述计算图，运行时引擎（runtime）理论上却可以在任何地方具体运行，比如在本机，在 CPU，或者在 TPU。



那么我们设计的数据处理技术也是一样的，除了有向图表达需要**数据处理描述语言**和**运算引擎**协商一致，其他的实现都是灵活可拓展的。

比如，我的数据描述可以用 Python 描述，由业务团队使用；计算引擎用 C++ 实现，可以由数据底层架构团队维护并且高度优化；或者我的数据描述在本地写，计算引擎在云端执行。



我们要统一批处理和流处理的编程模型

关于什么是批处理和流处理概念会在后面的章节展开。这里先简单解释下，批处理处理的是有界离散的数据，比如处理一个文本文件；流处理处理的是无界连续的数据，比如每时每刻的支付宝交易数据。

MapReduce 的一个局限是它为了批处理而设计的，应对流处理的时候不再那么得心应手。即使后面的 Apache Storm、Apache Flink 也都有类似的问题，比如 Flink 里的批处理数据结构用 DataSet，但是流处理用 DataStream。

但是真正的业务系统，批处理和流处理是常常混合共生，或者频繁变换的。

比如，你有 A、B 两个数据提供商。其中数据提供商 A 与你签订的是一次性的数据协议，一次性给你一大波数据，你可以用批处理。而数据提供商 B 是实时地给你数据，你又得用流处理。更可怕的事情发生了，本来是批处理的数据提供商 A，突然把协议修改了，现在他们实时更新数据。这时候你要是用 Flink 就得爆炸了。业务需求天天改，还让不让人活了？！

因此，我们设计的数据处理框架里，就得有更高层级的数据抽象。

不论是批处理还是流处理的，都用统一的数据结构表示。编程的 API 也需要统一。这样不论业务需求什么样，开发者只需要学习一套 API。即使业务需求改变，开发者也不需要频繁修改代码。

我们要在架构层面提供异常处理和数据监控的能力

真正写过大规模数据处理系统的人都深有感触：在一个复杂的数据处理系统中，难的不是开发系统，而是异常处理。

事实正是如此。一个 Google 内部调研表明，在大规模的数据处理系统中，90% 的时间都花在了异常处理中。常常发生的问题的是，比如在之前的番茄炒鸡蛋处理问题中，你看着系统 log，明明买了 1000 个鸡蛋，炒出来的菜却看起来只有 999 个鸡蛋，你仰天长叹，少了一个蛋到底去哪里了！

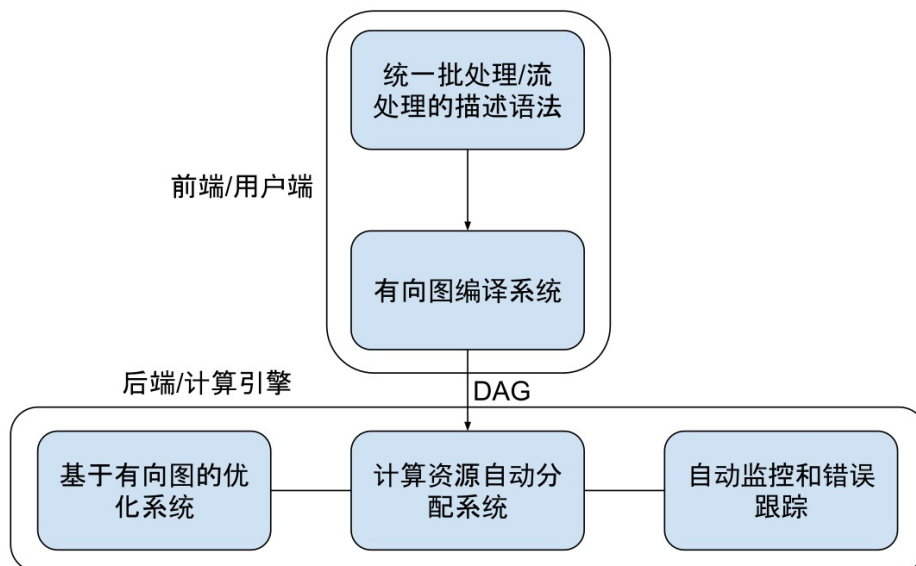
这一点和普通的软件开发不同。比如，服务器开发中，偶尔一个 RPC 请求丢了就丢了，重试一下，重启一下能过就行了。可如果在数据处理系统中，数据就是钱啊，不能随便丢。比如我们的鸡蛋，都是真金白银买回来的。是超市买回来数错了？是打蛋时候打碎了？还是被谁偷吃了？你总得给老板一个合理的交代。

我们要设计一套基本的数据监控能力，对于数据处理的每一步提供自动的监控平台，比如一个监控网站。

在番茄炒蛋系统中，要能够自动的记录下来，超市买回来是多少个蛋，打蛋前是多少个蛋，打完蛋是多少个蛋，放进锅里前是多少个蛋等等。也需要把每一步的相关信息存储，比如是谁去买的蛋，哪些人打蛋。这样出错后可以帮助用户快速找到可能出错的环节。

小结

通过上面的分析，我们可以总结一下。如果是我们站在 2008 年春夏之交来设计下一代大规模数据处理框架，一个基本的模型会是图中这样子的：



但是这样粗糙的设计和思想实验离实现还是太远。你可能还是会感到无从下手。

后面的章节会给你补充一些设计和使用大规模数据处理架构的基础知识。同时，也会深入剖析两个与我们这里的设计理念最接近的大数据处理框架，Apache Spark 和 Apache Beam。

思考题

你现在在使用的数据处理技术有什么问题，你有怎样的改进设计？

欢迎你把自己的想法写在留言区，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。

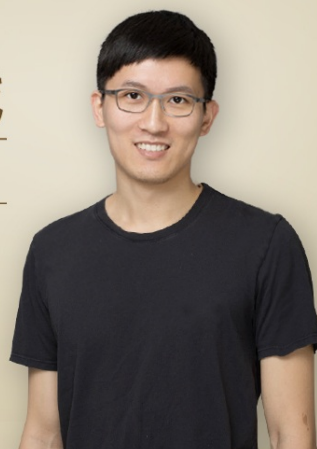


大规模数据处理实战

Google 一线工程师的大数据架构实战经验

蔡元楠

Google Brain 资深工程师



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。



由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Ctrl + Enter 发表

0/2000字

提交留言

精选留言(17)



mjl 置顶

Unify platform和批流统一已经是主要趋势了，而我个人目前只对spark、flink有一定的了解。对于spark来说，无疑是很优秀的一个引擎，包括它的all in one的组件栈，structured streaming出来后的批流api的统一，目前在做的continues Mode。而flink，的确因为阿里的运营，在国内火了。但也展现了它的独有优势，更加贴近dataflow model的思想。同时，基于社区以及阿里、华为小伙伴的努力，flink的table/sql的api也得到的很大的增强，提供了流批统一的api。虽然底层然后需要分化为dataset和datastream以及runtime层的batchTask和StreamTask，但是现在也在rethink the stack，这个point在2019 SF 的大会也几乎吸引了所有人。但就现状而言，flink的确有着理念上的优势（流是批的超集），同时也有迅猛上升的趋势。

👍 1 2019-04-19

作者回复: 谢谢你的留言！感觉活捉了一只技术大牛呢。

是的，Spark的话虽然原生Spark Streaming Model和Dataflow Model不一样，但是Cloudera Labs也有根据Dataflow Model的原理实现了Spark Dataflow使得Beam可以跑Spark runner。

而对于Flink来说的话，在0.10版本以后它的DataStream API就已经是根据Dataflow Model的思想来重写了。现在Flink也支持两套API，分别是DataStream版本的和Beam版本的。其实data Artisans一直都有和Google保持交流，希望未来两套Beam和Flink的API能达到统一。

最后赞一点批处理是流处理的子集，这个观点我在第一讲留言也提到过。

如果觉得有收获，欢迎分享给朋友！同时也欢迎你继续留言交流，一起学习进步！



文洲

老师讲的很明白，有个疑问，现在都说flink是下一代实时流处理系统，这里按老师的对比来看，它还比不上spark，可否理解为flink主要还是专注实时流处理而spark有兼具批处理和流处理的优势

👍 2 2019-04-19

作者回复: 并没有说flink比不上spark。。。这篇没有展开比较。只是带了一句话flink的流处理批处理api不一样。



wmg

现在使用较多的是hive和sparkSql，这两种技术多使用的都是类似关系数据库的数据模型，对于一些复杂的对象必须要通过建立多张表来存储，查询时可能需要多张表进行join，由于担心join的性能损耗，一般又会设计成大宽表，但这样又会浪费存储空间，数据一致性也得不到约束。想问一下老师，有没有支持类似mongodb这样的文档型数据模型的大数据处理技术？

👍 1 2019-04-19

作者回复: 数据的索引查询和这里的数据处理是两个话题。关于怎么提高数据系统的查询效率，我在考虑要不要开一个存储系统高性能索引专栏。

我不知道你的查询有多复杂，简单处理的话，可以先试试建一些常见查询路径的索引表。写的时候往两张表写或者再做一些专门的建索引的数据处理系统。



孙稚昊

Flink在国内也是刚刚兴起火热起来，而Apache Beam也就是Google内部FlumeJava的开源版本早早就被设计出来。国内现在讲实时数据处理，批流统一还是比较推崇Flink和阿里的自主设计的Blink系统，接受Beam可能还需要几年的时间

👍 1 2019-04-19



大王叫我来巡山

经历了mapreduce到spark, 从最具欺骗性的wordcount开始到发现很多业务本身并不适合mapreduce编程模型，一直做日志处理，现在在政府某部门同时处理批处理任务和流处理任务，老师的课太及时了，感觉对我们的业务模型革新会产生很大的影响，前两篇没有涉及技术，已经感觉醍醐灌顶了。

👍 1 2019-04-19

作者回复: 你说的对，WordCount就像helloworld，真实业务场景肯定会复杂很多。



Edwin

Spark 和 Flink 都是通用的开源大规模处理引擎，目标是在一个系统中支持所有的数据处理以带来效能的提升。两者都有相对比较成熟的生态系统。是下一代大数据引擎最有力的竞争者。Spark 的生态总体更完善一些，在机器学习的集成和易用性上暂时领先。Flink 在流计算上有明显优势，核心架构和模型也更透彻和灵活一些。

社区版本的flink有些问题还待进一步完善，比如统一API层、状态管理机制、资源调度等，阿里内部的blink虽然对flink做了大量改造，但依赖内部环境能真正返璞社区还有很长一段路要走~

👍 2019-04-19



cricket1981

请问不定时的来一批大量数据要处理是要用批框架还是流框架呢？

👍 2019-04-19

作者回复: 谢谢你的提问！如果是不确定数据流入时间的话需要采用流处理框架去处理了。



王二

老师您好：

- 1.如您所说flink在批流处理上的不统一，目前社区各个厂商也在努力的实现这种基于SQL或其他引擎的一些统一，这块在您看来有什么好的建议。
- 2.后面文章使用spark举例，这里是用他的struct streaming还是spark streaming呢。
- 3.感同身受，流系统难的不在开发，在于监控，如何处理背压，如何根据负载动态调整资源，除过开源框架自带的一些监控外，后续在任务流监控这块是否有什么好的建议推荐。

👍 2019-04-19



dylan

有没有这么一个问题,mapreduce在大规模数据处理时，由于每个计算阶段都会落盘，所以计算比较慢，但是数据完整，不会丢失；spark基于内存计算，会有数据丢失的风险？

👍 2019-04-19

作者回复: 容错性是很重要的问题，两个的容错性设计方案是不同的。



明翼

课堂笔记: mr有三个重要问题，一是支持的计算粒度太大，描述麻烦；二是性能差，调优复杂；三是只支持批量处理不支持流处理。针对此设计了有向无环图，好处是描述简单，易在上面进行自动优化；描述和计算引擎分开了，解耦合，可以分别实现进一步提升性能；我一直觉得批量处理和流处理完全两回事，老师的视角是将两者融合，想了下，批量处理可以看做时间间隔很长的流，而流按照spark等时间窗口概念可以看做小范围的批量，这个设计可以这样想，设计起来可能很复杂。另外一个由于可以表达更复杂的事件转换，那好的监控异常处理就是必须的了，以上为总结。

关于遇到问题，我们曾经遇到过两个都是很大的流类的日志数据匹配处理问题，是TB级别，而且数据流不同步的一个流的数据跨度范围大，如何设计高效处理是个难题，在hbase缓存还是在hdfs缓存是个难题，不知道老师有何高见？

👍 2019-04-19

作者回复: 我觉得你总结的很好。

问题我似乎没有很明白，难点是在时间范围大还是缓存？



Rainbow

flumejava第一次听说不是很了解，老师的意思google内部目前还是flumejava作为主流框架吗？

👍 2019-04-19

作者回复: 文章里写到了是这个意思



时间是最真的答案

作为一个不会大数据开发的，学完这套课程能不能使用大数据技术做一些初级的工作呢？

👍 2019-04-19

作者回复: 这个目标应该是能达到的。



Liu C.

目前还在学校科研的现个丑：目前我所做的数据任务是下载科研论文和报告并对其中的文字数据进行分析，用的是自己写的工具，分为两部分：第一部分从网上下载所需数据，第二部分进行内容提取和分析。

目前这是个批处理式的系统：先运行下载工具之后再运行分析。这会导致没法及时更新最新出现的数据。解决方法是，加入一个定时查看的模块，在发现新添加数据时进行下载并处理update分析结果。

另一个缺点是，我没有log那些数据抓取失败的情况。可以添加一个这样的模块来协助优化系统。

👍 2019-04-19

作者回复: 谢谢你的留言！我相信第10讲中的Lambda数据处理架构会对你有所帮助，期待那时候再次看到你的留言。



涵

从实施经验看，批量处理往往是频率低，一次性数据量很大，响应无需即时，而流数据处理往往是高频率但小数据量，要求即时响应，很期待看到beam是如何将两种数据量级差异很大的场景统一起来，都可

以处理的很好。

 2019-04-19



流殇忘情

我说说我们在线上遇到过的实际问题，我们是做内容的，当内容产生view数的时候会发送用户行为日志，这个时候通过消费消息队列中的数据为用户进行收益的计算，但是这个数据是有有效期的，曾经出现过由于bug产生一些数据没有及时消费，就需要手动进行补救，看完今天的文章，我觉得可以对有时效性的数据进行标注，如果没有成功消费，系统也可以方便的找到进行自动修复。

 2019-04-19

作者回复: 你说的很好啊



leben krieg

1. DAG是将一系列的转换函数连接起来，最后调用真正的计算函数
2. 番茄炒鸡蛋，炒牛腩这个例子，说明不要进行重复计算，要复用之前的计算。而且在计算之前过滤掉不必要的数据，最大限度的减少计算量。


现在使用的数据处理技术遇到的问题就是在两个大表进行关联的时候，没有太多的优化手段，自己能想到的就是增加计算资源（但是条件不允许），然后过滤掉两个表中不必要的数据，但其实优化之后还是会很慢。

希望老师能分享一下大表和大表join的优化手段！

 2019-04-19



W.T

 赞！期待后续章节...

 2019-04-19

作者回复: 谢谢！