

40 | 大规模数据处理未来之路

2019-07-26 蔡元楠

大规模数据处理实战

[进入课程 >](#)



讲述：巴莫

时长 09:00 大小 8.24M



你好，我是蔡元楠。

今天我要分享的内容是“大规模数据处理实战”专栏的最后一讲。

我相信通过整个专栏的系统学习，你已经掌握了大规模数据处理的基础概念与设计模式。同时，我也相信，专栏中对现实世界中常见的大规模数据处理架构的深入探讨，可以在解决现实难题时为你提供一些思路。

但我更希望的是，通过模块六中对大规模数据处理在未来的应用与展望讲解，让你吃下一颗定心丸，那就是，大规模数据处理技术是在放眼未来的几十年中都依然会是炙手可热的一项技术，不会被淘汰。

你不难发现，我在专栏的后半部分，花了不少的篇幅来专门介绍 Apache Beam 的各种概念、底层思想以及实际应用的。我个人是十分认同 Google 所推崇的 Dataflow Model 的计算模型，也相信未来 Apache Beam 的发展前景是很好的。

所以在专栏的最后一讲，我想和你讲讲我对数据处理框架和对 Beam 的一些看法和展望。

技术迭代带来的烦恼

在专栏的后半部分，我们不断深入探讨了 Apache Beam。有同学曾经在留言中提过一个问题：“我已经掌握好 Spark 了，也觉得 Spark 的语法更简练，为什么还需要学习 Beam 呢？”

对于这个问题，我相信在你刚刚接触 Beam 的时候，多多少少都会有相同的疑问。

我来给你举个例子，带你穿越时间，去看看一个常见的大规模数据处理框架在面临迁移时会遇到的烦恼。

在 2006 年，Hadoop 刚刚发布，在选择数据处理框架的时候，你的首选肯定是 Apache Hadoop。当时 Hadoop 在开源社区横空出世，让所有工程师们都可以利用这个框架来处理自己的数据，尤其是利用 Hadoop 自带的 MapReduce 计算模型，可以让整个数据处理的效率有质的飞跃。

而在 2009 年，Spark 这个计算框架在加州伯克利大学的 AMPLab 实验室中诞生。2010 年，Spark 正式开源了。“Spark 的数据处理效率远在 Hadoop 之上”的结论经过了业界的验证。2014 年，Spark 更是成为了 Apache 的顶级项目。

如果你是在这个时候进入 IT 行业的工程师，更加可能选择直接学习 Spark，而只会把少量时间放在 Hadoop 的学习上。在 2014 年进行创业的小公司同样可能也会直接使用 Spark 来进行底层数据处理基础设施的搭建。

那之前已经花了很多时间在搭建 Hadoop MapReduce 作为公司基础设施的公司，或者是已经很深入学习了 Hadoop 的工程师这个时候该怎么办呢？

一种做法是放弃现有的技术框架，重新花费大量时间去学习新的数据处理框架。这太累了。对于工程师来说，平时本来就有着做不完的任务和业绩压力，还需要抽空学习新的技术和进行代码的迁移，这无疑让工程师们有着非常大的压力和负担。

当然，还有一种做法是保持现有的技术框架，不断优化现有基础设施，并且寄希望于老框架可以有新的功能发布让其性能得以提高。

那我们再把时间往后推一点，到了 2014 年，也就是 Flink 项目刚刚面世的时候。这时候的互联网应用场景已经有了极大的变化，批处理加上流处理同时存在的状况常常遇见。而 Flink 除了在处理大规模数据有着极高效率之外，它的批流统一功能也恰恰是满足了这种批流处理同时存在的场景。

2018 年初，阿里巴巴更是看好 Flink 的前景，收购了 Apache Flink 背后的商业母公司——德国的 Data Artisans。

讲到这里，你不难发现，当有新的技术框架出现的时候，工程师就会陷入一个选择的困难，纠结到底是抛弃原有的技术架构，还是花大量时间去做技术迁移。

其实，如果一开始就有 Beam 模型存在的话，你可能就不必有这个烦恼了。

因为我们完全不需要担心某一个 Runner，也就是具体的数据处理技术框架过时之后所带来的技术迁移成本。如果你想要完成底层处理框架的迁移，只需要更改一些 Runner 的接口就可以了。

Apache Beam 能带来什么？

那么，我们来看看对应用它的工程师来说，Apache Beam 能带来什么？

因为 Apache Beam 是根据 Dataflow Model 倡导 API 的实现的，所以它完全能够胜任批流统一的任务。同时，因为 Apache Beam 有着中间的抽象转换层，工程师可以从 API 中解放出来，不需要学习新 Runner 的 API 语法。这也就是我说过多次的“编写一套程序就可以放在不同的 Runner 上面跑”。

除了对工程师来说能够极大地减少新技术学习的时间成本，Apache Beam 还能够推动大规模数据处理领域的最新技术发展。

从上面我所举的例子中你可以看到，在一项新技术从诞生到流行的过程中，可能会有很多公司因为技术迁移成本太大而选择放弃采用最新的技术，这其实还影响了新技术的传播使用。因为当一部分工程师选择留守原本技术框架的时候，新技术框架也就相应地缺少了这部分的用户群体。

那我们来想象一下，如果所有的工程师都在使用 Beam 来进行项目编写的话会有什么样的效果。

因为有了 Beam 的抽象层，你可以非常轻松地更换不同的底层 Runner。这就意味着我们可以先选择一个处理数据效率最高的一个 RunnerA。如果有其他的 Runner 优化了自身，比如 RunnerB，从而拥有更高效率的时候，工程师们又可以将底层 Runner 换成 RunnerB。

这些 Runner 其实就是大数据处理框架的本身，像 Spark、Apex、Storm、Flink 这些数据处理架构。这对整个技术行业都可以起到一种良性的竞争。如果 Runner 要想争取更多用户的话，就必须努力提升自身数据处理的效率来让用户选择自己。

做底层数据处理框架的工程师则可以专心优化自身的效率和增加自身的功能，而不必担心迁移。

且 Apache Beam 还有着自己的社区。所以，在实际工程中如果遇到一些特别的、没有在官方文档中解释的问题，你就可以到社区去求助了。有了社区交流之后，全世界的工程师们都可以对自己的工程提出问题，解决问题，实现解决问题的思路。

Beam Runner 功能的迭代速度

最后你可能会问，Apache Beam 的一些功能现在还没有，那还是先观望观望吧。那我来以 Flink 支持整个 Dataflow 的功能来告诉你 Beam Runner 功能的迭代速度有多快。

Kostas Tzoumas (Data Artisans 的联合创始人以及 Flink 的作者) 在 Beam 出现的时候就表达过自己的看法，他坚信 Beam Model 是批流数据处理的正确编程模型。所以，在 Dataflow Model 论文发布之初，他们根据 Dataflow Model 的思想重写了 Flink 的 0.10 版本的 DataStream API。

这整个过程花费了多少时间呢？

在 2014 年的 12 月，Google 正式发布 Google Cloud Dataflow SDK。在 2015 年的 1 月，Data Artisans 紧随 Google 的脚步，根据 Dataflow Model 的概念，开始了 DataStream API 的重写工作。

在 2015 年 3 月，Flink 已经可以对批处理进行支持了。到了 2015 年的 12 月，Flink 可以支持设置窗口和水印的流处理。其实这个时候 DataStream API 已经很接近 Dataflow Model 的概念了。所以在 2016 年 3 月，Data Artisans 正式开始在 Apache Beam 的 Runner 代码库中贡献 Flink 的 Runner 代码。

在 2016 年 5 月的时候，Flink 对批处理也支持了窗口的概念，同时也对整个批处理完成了集成测试（Integration Test）。在 2016 年 8 月，Flink 完成了对流处理的集成测试。自此，Flink DataStream API 宣告完成了对 Dataflow Model 的支持。

整个过程从无到有，一共只花费了近一年半的时间。所以你完全可以不必对功能不完整抱有太多的担心。

小结

今天，我给你阐述了我自己对于 Beam 的一些看法，同时也希望借助我所举出的一些例子，能够让你明白 Beam 对于未来数据处理发展的重要性。

思考题

你对 Apache Beam 还有什么疑问吗？欢迎提问与我探讨。

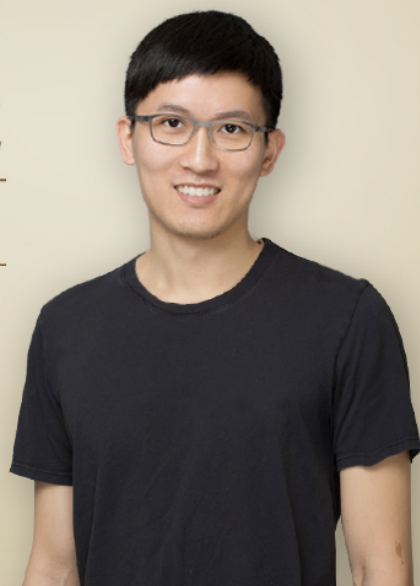
欢迎你把自己的学习体会写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

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。

© 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。

上一篇 39 | 从SQL到Streaming SQL：突破静态数据查询的次元

下一篇 结束语 | 世间所有的相遇，都是久别重逢

精选留言 (2)

 写留言



limeng

2019-07-26

请教个问题，不知道老师对数据中台有什么见解？

展开

 1

 2



limeng

2019-07-26

我现在就用beam做些需求，谢谢老师专栏，学到很多！





