Apache Flink AI： 因为相信，所以看见

去年11月的 Flink Forward Asia 2019 上 Flink 社区提出了未来发展的几个主要方向，其中之一就是拥抱 AI [1]。实际上，近年来 AI 持续火热，各种计算框架、模型和算法层出不穷，从某种角度上来说，这个赛道已经有些拥挤了。在这种情况下， Flink将怎样拥抱AI，又会为用户带来什么新的价值？Flink AI的优劣势分别在哪里？本文将通过对这些问题的讨论来分析Flink AI 的发展方向。

Lambda架构，流批统一和AI实时化

Flink 在AI 中的价值其实和大数据中Lambda架构 [2]和流批统一这两个概念有关系，Flink为大数据实时化带来的价值也将同样使AI受益。

不妨让我们简单回顾一下大数据的发展过程。从Google奠基性的“三架马车” [3][4][5] 论文发表后的很长一段时间内，大数据的发展主线上都只有批计算的身影。后来随着大家认识到数据时效性的重要作用，Twitter 开源的流计算引擎Storm [6] 红极一时，各种流计算引擎也纷纷登场，其中也包括了Flink。由于成本、计算准确性和容错性等方面的考虑，各家企业纷纷使用起了被称为Lambda架构的解决方案，在同一个架构下融合批计算和流计算，以便在成本，容错和数据时效性之间达到一个平衡。

Lambda架构在解决数据时效性的同时也存在一些问题，其中最受诟病的就是其系统复杂度和可维护性。用户需要为 Batch Layer 和 Speed Layer 各维护一套引擎和代码，还需要保证二者之间的计算逻辑完全一致（图1）。

为了解决这个问题，各个计算引擎不约而同的开始了流批统一的尝试，试图使用同一套引擎来执行流和批的任务（图2）。经过若干年的大浪淘沙，Spark [7] 和Flink成为了目前处于第一梯队的两款主流计算引擎。Flink 是从流计算逐渐进入到批计算，一个非常典型的成功案例就是使用同一套标准的SQL语句对流和批进行查询，并保证最终结果一致性[8]。而Spark 则是采用微批 (Micro Batch) 的方式从批计算进入到流计算提出了 Spark Streaming，但是在时延的表现上始终逊色一些。

图2

可以看到，在大数据的发展过程中，Lambda架构和流批一体背后的原始驱动力是数据实时化。*同样是向数据要价值，AI对数据时效性的要求同大数据是一致的*。因此AI实时化也将会是一个重要的发展方向。在观察目前主流的AI场景和技术架构时，我们也会发现它们与大数据平台有很多联系和相似之处。

目前的 AI大致可以分为数据预处理（也称数据准备/特征工程等），模型训练和推理预测三个主要阶段。下面我们逐一来看一看在每个阶段中AI实时化需求有哪些，又有什么样的问题待解决。为了便于与大数据的架构做类比，我们姑且认为流计算和批计算作为一种计算类型的划分维度已经将所有基于数据的计算一分为二，没有遗漏了。AI的各个阶段根据场景不同，也可以归为二者之一。

数据预处理（数据准备/特征工程）

数据预处理阶段是模型训练和推理预测的前置环节，很多时候它更多的是一个大数据问题。根据数据预处理后的下游不同，数据预处理可能是批计算也可能是流计算，计算类型和下游一致。在一个典型的离线训练（批计算）和在线预测（流计算）场景下，训练和预测时要求产生输入数据的预处理逻辑是一致的（比如相同的样本拼接逻辑），这里的需求和Lambda架构中的需求一样，因此一个流批统一的引擎会格外有优势。这样可以避免批作业和流作业使用两个不同的引擎，省去了维护逻辑一致的两套代码的麻烦。

模型训练

目前而言AI训练阶段基本上是批计算（离线训练）产生静态模型（Static Model）的过程。这是因为目前绝大多数的模型是基于独立同分布（IID）的统计规律实现的，也就是从大量的训练样本中找到特征和标签之间的统计相关性（Correlation），这些统计相关性通常不会突然变化，因此在一批样本上训练出的数据在另一批具有相同的特征分布的样本上依然适用。然而这样的离线模型训练产生的静态模型依然可能存在一些问题。

首先样本数据可能随着时间推移会发生分布变化，这种情况下，在线预测的样本分布和训练样本的分布会产生偏移，从而使模型预测的效果变差。因此静态模型通常需要重新训练，这可以是一个定期过程或者通过对样本和模型的预测效果进行监控来实现（注意这里的监控本身其实是一个典型的流计算需求）。

另外，在有些场景下，预测阶段的样本分布可能无法在训练阶段就知晓。举例来说，在阿里双十一，微博热搜，高频交易等这类样本分布可能发生无法预测的分布改变的场景下，如何迅速更新模型来得到更好的预测结果是十分有价值的。

因此一个理想的AI计算架构中，应该把如何及时更新模型纳入考虑。在这方面流计算也有着一些独特的优势。事实上，阿里巴巴在搜索推荐系统中已经在使用在线机器学习，并且在双十一这样的场景下取得了良好的效果。

推理预测

推理预测环节的环境和计算类型比较丰富，既有批处理（离线预测）又有流处理。流式预测又大致可以分为在线 (Online) 预测和近线 (Nearline) 预测。在线预测通常处于用户访问的关键链路 (Critical Path中)，因此对latency的要求极高，比如毫秒级。而近线预测要求略低一些，通常在亚秒级到秒级。目前大多数纯流式分布式计算（Native Stream Processing）引擎可以满足近线数据预处理和预测的需求，而在线数据预处理和预测则通常需要将预测代码写进应用程序内部来满足极致的低延迟要求。因此在线预测的场景也比较少看到大数据引擎的身影。在这方面Flink的Stateful Function [9] 是一个独特的创新，Stateful Function的设计初衷是在Flink上通过若干有状态的函数来构建一个在线应用，通过它可以做到超低延迟的在线预测服务，这样用户可以在离线，近线和在线三种场景下使用同一套代码同一个引擎来进行数据预处理和预测。

综上所述，可以看到在机器学习的每个主要阶段中对AI实时化都有重要的需求，那什么样的系统架构能够有效满足这样的需求呢?

Flink和AI实时化的架构

目前最典型的AI架构示例是离线训练配合在线推理预测（图3）。

图3

正如之前提到的，这个架构存在两个问题：

1. 模型更新的周期通常比较长。
2. 离线和在线的预处理可能需要维护两套代码。

为了解决第一个问题，我们需要引入一个实时训练的链路（图4）。

图4

在这个链路中，线上的数据在用于推理预测之外还会实时生成样本并用于在线模型训练。在这个过程中，模型是动态更新的，因此可以更好的契合样本发生的变化。

不论是纯在线还是纯离线的链路，都并非适合所有的AI场景。和Lambda的思想类似，我们可以把两者结合（图5）。

图5

同样的，为了解决系统复杂度和可运维性的问题（也就是上面提到的第二个问题），我们希望在数据预处理的部分用一个流批统一的引擎来避免维护两套代码（图6）。不仅如此，我们还需要数据预处理和推理预测能够支持离线，近线和在线的各种Latency要求，所以使用Flink是一个非常合适的选择。尤其是对于数据预处理环节而言，Flink 在流和批上全面完整的 SQL支持可以大大提高的开发效率。

图6

除此之外，为了进一步降低系统的复杂度，Flink也在模型训练环节进行了一系列努力（图7）。

- 流批一体算法库 Alink。

在去年的 FFA 2019上，阿里巴巴宣布开源了基于Flink的机器学习算法库Alink [10]，并计划将其逐步贡献回Apache Flink，作为Flink ML Lib随Apache Flink发布。除了离线学习的算法外，Alink的一大特色就是为用户提供了在线学习算法，助推Flink在AI实时化上发挥更大的作用。

- Deep Learning on Flink （flink-ai-extended [11]）。

帮助用户把目前流行的深度学习框架（TensorFlow、PyTorch）整合到Flink中。使除了深度学习算法开发者之外的用户可以基于Flink实现整套AI架构。

- 流批统一的迭代语义和高性能实现。

AI训练中迭代收敛是一个最核心的计算过程。Flink从一开始就使用了原生迭代的方式来保证迭代计算的效率。为了帮助用户更好的开发算法，简化代码，进一步提高运行效率。Flink社区也正在统一流和批上迭代的语义，同时对迭代性能进行更进一步的优化，新的优化将尽可能避免迭代轮次之间的同步开销，允许不同批次的数据、不同轮次的迭代同时进行。

图7

当然，在一个完整的AI架构中，除了以上提到的三个主要阶段，还有很多其他工作需要完成，包括对各种数据源的对接，已有AI生态的对

接，在线的模型和样本监控和各类周边配套支持系统等。阿里巴巴实时计算负责人王峰（花名莫问）在2019年FFA的主题演讲中的一张图（图8）很好的总结了其中许多工作。

图8

Flink社区也正在为此做出努力。大致上来说，这些AI相关的工作可以分成补足，提高和创新三类。下面罗列了其中一部分进行中的工作，有些工作也许与AI不直接相关，但是却会对Flink更好的服务于AI实时化产生影响。

补足 – 人有我无

- Flink ML Pipeline [12]：帮助用户方便的存储和复用一个机器学习的完整计算逻辑。
- Flink Python API（PyFlink [13]）：Python 是AI 的母语，PyFlink为用户提供AI中最重要的编程接口。
- Notebook Integration [14]（Zeppelin）：为用户的AI实验提供友好的API。
- 原生Kubernetes支持 [15]：和Kubernetes集成来支持基于云原生的的开发、部署和运维。

提高 – 人有我强

- Connector 的重新设计和优化 [16]：简化Connector实现，扩大Connector生态。

创新 – 人无我有

- AI Flow：兼顾流计算的大数据 + AI 顶层工作流抽象和配套服务（即将开源）。
- Stateful Function[9]：提供堪比在线应用的超低延迟数据预处理和推理预测。

其中有些是 Flink 作为流行的大数据引擎的自有功能，比如丰富 Connector 生态来对接各种外部数据源。另一些则要依靠Flink之外的生态项目来完成，其中比较重要的是AI Flow。它虽然起源于支持AI实时化架构，但是在引擎层并不绑定Flink，而聚焦于顶层的流批统一工作流抽象，旨在为不同平台，不同引擎和不同系统共同服务于 AI 实时化的架构提供环境支持。由于篇幅关系在此不多赘述，将另文向大家介绍。

写在最后

Apache Flink 从一个简单的流计算想法开始，直到今天成长为一个业界流行的实时计算开源项目，使所有人受益，这个过程中离不开Flink社区中数以百计的代码贡献者和数以万计的用户。我们相信Flink在AI上也能够有所作为，也欢迎更多的人能够加入到Flink社区，同我们一起共创并共享AI实时化的价值。

Flink AI，未来可期。

参考资料

[1] https://ververica.cn/developers/the-number-of-github-stars-doubled-in-only-one-year/

[2] https://en.wikipedia.org/wiki/Lambda_architecture

[3] https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf

[4] https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

[5] https://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf

[6] https://storm.apache.org/

[7] https://spark.apache.org/

[8] https://ci.apache.org/projects/flink/flink-docs-release-1.10//dev/table/sql/index.html

[9] https://statefun.io/

[10] https://github.com/alibaba/alink

[11] https://github.com/alibaba/flink-ai-extended

[12] https://cwiki.apache.org/confluence/display/FLINK/FLIP-39+Flink+ML+pipeline+and+ML+libs

[13] https://ci.apache.org/projects/flink/flink-docs-release-1.10/tutorials/python_table_api.html

[14] https://mp.weixin.qq.com/s/a6Zau9c1ZWTSotl_dMg0Xg

[15] https://ci.apache.org/projects/flink/flink-docs-stable/ops/deployment/kubernetes.html

[16] https://cwiki.apache.org/confluence/display/FLINK/FLIP-27%3A+Refactor+Source+Interface