如何用一套引擎搞定机器学习全流程?

简介： 本文由阿里技术专家陈戊超（仲卓）分享。深度学习技术在当代社会发挥的作用越来越大。目前深度学习被广泛应用于个性化推荐、商品搜索、人脸识别、机器翻译、自动驾驶等多个领域，此外还在向社会各个领域迅速渗透。

作者:陈戊超（仲卓）

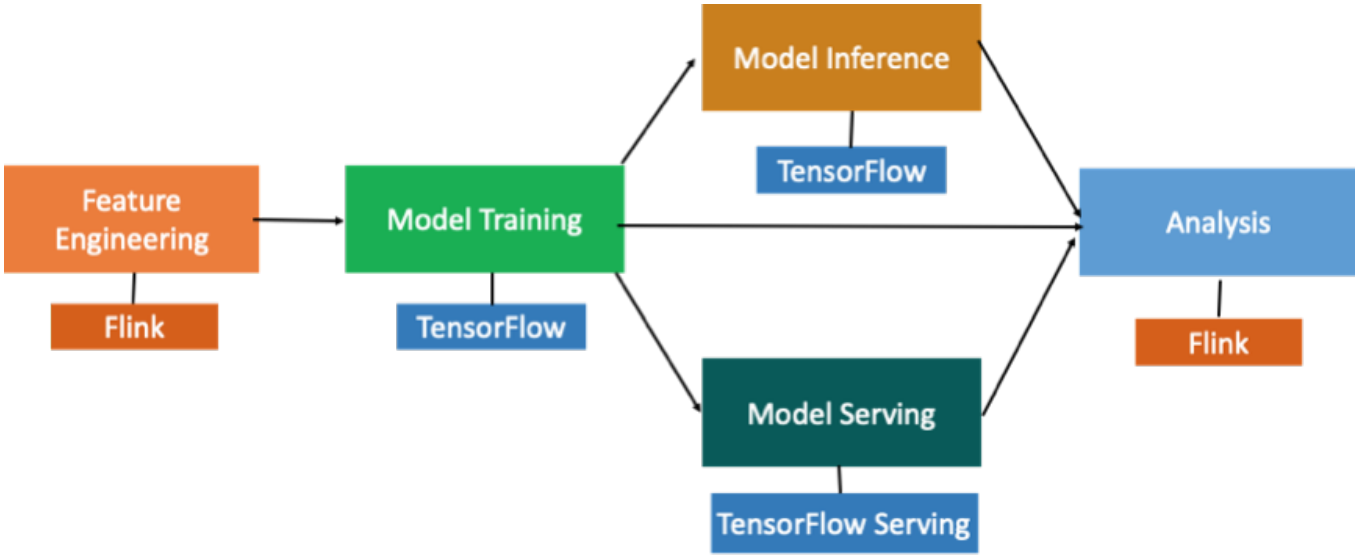
深度学习技术在当代社会发挥的作用越来越大。目前深度学习被广泛应用于个性化推荐、商品搜索、人脸识别、机器翻译、自动驾驶等多个领域，此外还在向社会各个领域迅速渗透。

背景

当前，深度学习的应用越来越多样化，随之涌现出诸多优秀的计算框架。其中 TensorFlow，PyTorch，MXNeT 作为广泛使用的框架更是备受瞩目。在将深度学习应用于实际业务的过程中，往往需要结合数据处理相关的计算框架如：模型训练之前需要对训练数据进行加工生成训练样本，模型预测过程中需要对处理数据的一些指标进行监控等。在这样的情况下，数据处理和模型训练分别需要使用不同的计算引擎，增加了用户使用的难度。

本文将分享如何使用一套引擎搞定机器学习全流程的解决方案。

先介绍一下典型的机器学习工作流程。如图所示，整个流程包含特征工程、模型训练、离线或者是在线预测等环节。



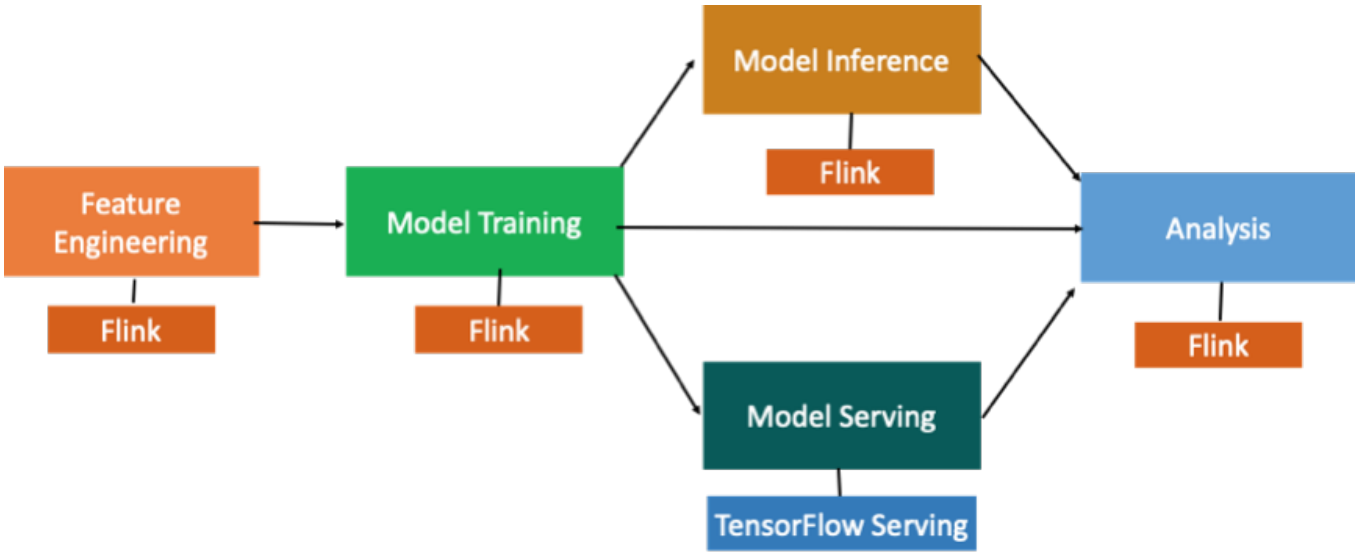
在此过程中，无论是特征工程、模型训练还是模型预测，中间都会产生日志。需要先用数据处理引擎比如 Flink 对这些日志进行分析，然后进入特征工程。再使用深度学习的计算引擎 TensorFlow 进行模型训练和模型预测。

当模型训练好了以后再用 TensorFlow serving 做在线的打分。

上述流程虽然可以跑通，但也存在一定的问题，比如：

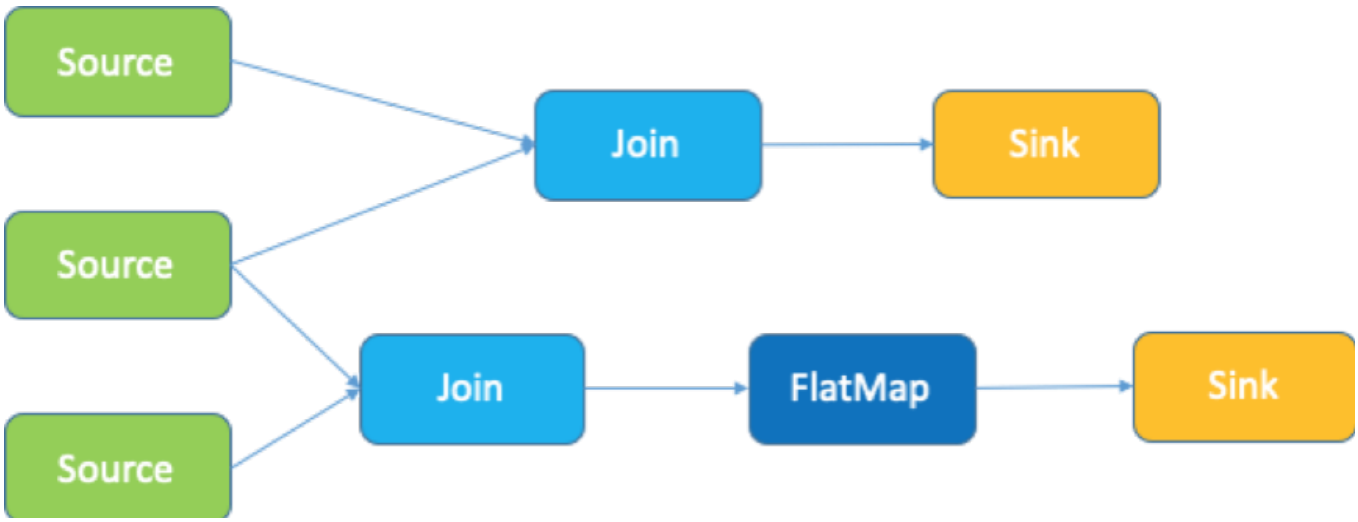
1. 同一个机器学习项目在做特征工程、模型训练、模型预测时需要用到 Flink 和 TensorFlow 两个计算引擎，部署相对而言更复杂。
2. TensorFlow 在分布式的支持上还不够友好，运行过程中需要指定机器的 IP 地址和端口号；而实际生产过程经常是运行在一个调度系统上比如 Yarn，需要动态分配 IP 地址和端口号。
3. TensorFlow 的分布式运行缺乏自动的 failover 机制。

针对以上问题，我们通过结合 Flink 和 TensorFlow，将 TensorFlow 的程序跑在 Flink 集群上的这种方式来解决，整体流程如下：



特征工程用 Flink 去执行，模型训练和模型的准实时预测目标使 TensorFlow 计算引擎可以跑在 Flink 集群上。这样就可以用 Flink 一套计算引擎去支持模型训练和模型的预测，部署上更简单的同时也节约了资源。

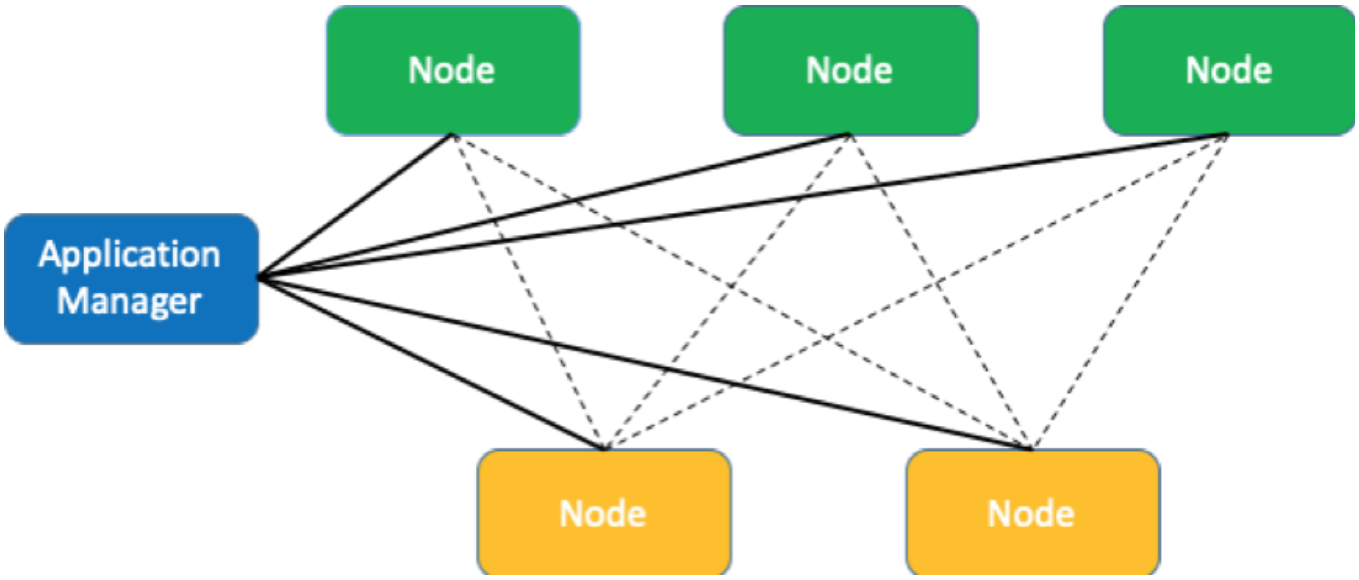
Flink 计算简介



Flink 是一款开源大数据分布式计算引擎，在 Flink 里所有的计算都抽象成 operator，如上图所示，数据读取的节点叫 source operator，输出数据的节点叫 sink operator。source 和 sink 中间有多种多样的 Flink operator 去处理，上图的计算拓扑包含了三个 source 和两个 sink。

机器学习分布式拓扑

机器学习分布式运行拓扑如下图所示：



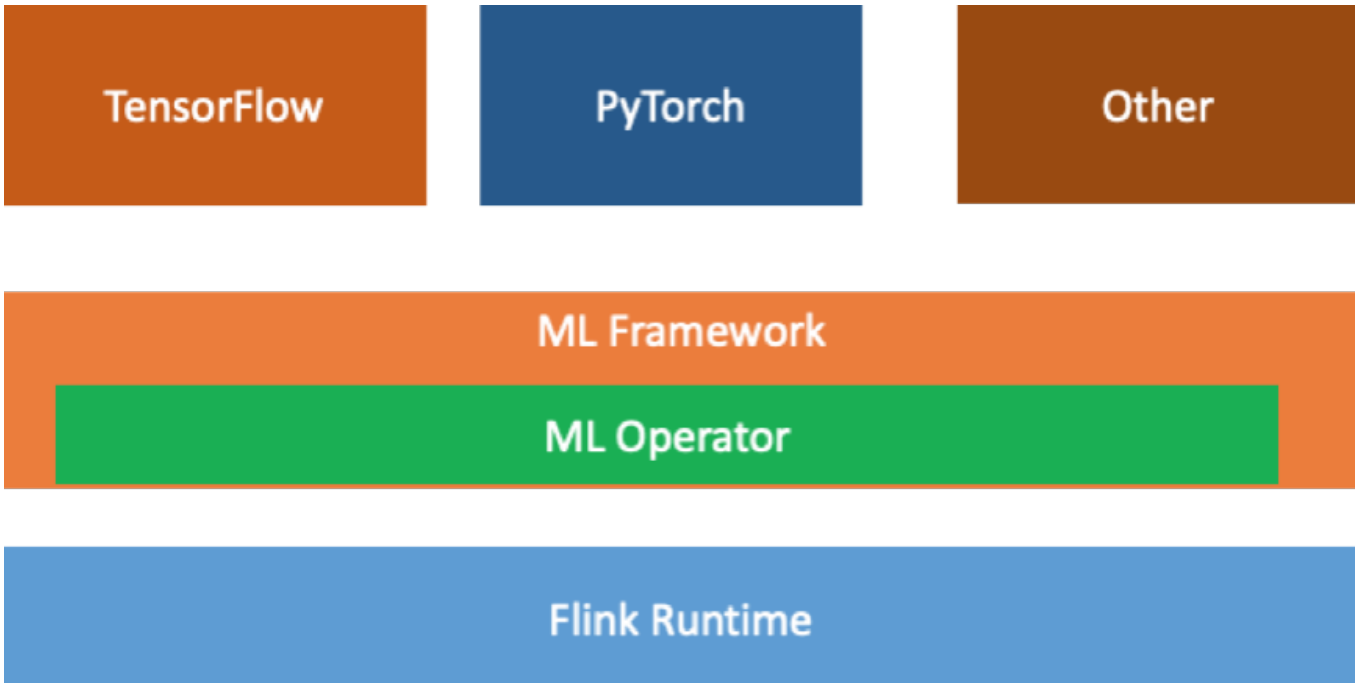
在一个机器学习的集群当中，经常会对一组节点（node）进行分组，如上图所示，一组节点可以是 worker（运行算法），也可以是 ps（更新参数）。

如何将 Flink 的 operator 结构与 Machine Learning 的 node、Application Manager 角色结合起来？下面将详细讲解 flink-ai-extended 的抽象。

Flink-ai-extended 抽象

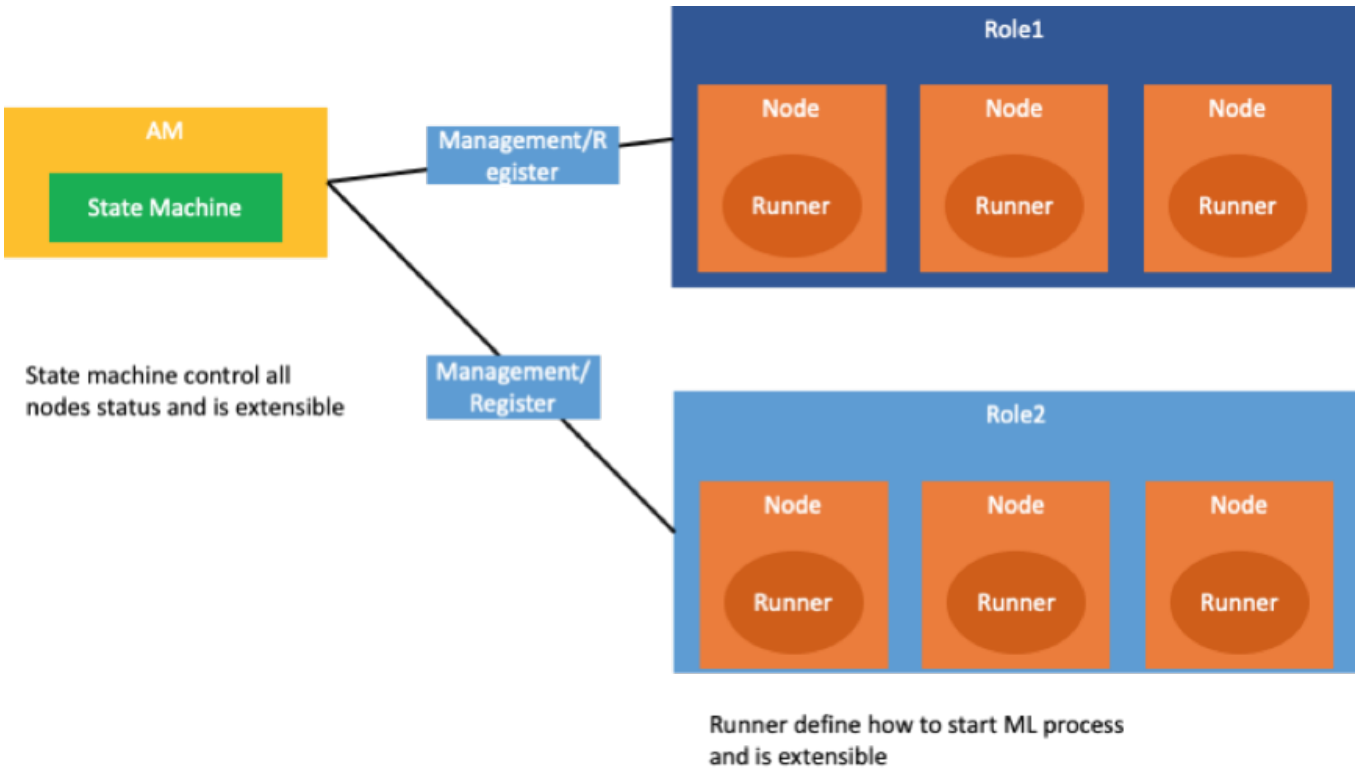
首先，对机器学习的 cluster 进行一层抽象，命名为 ML framework，同时机器学习也包含了 ML operator。通过这两个模块，可以把 Flink 和 Machine Learning Cluster 结合起来，并且可以支持不同的计算引擎，包括 TensorFlow。

如下图所示：



在 Flink 运行环境上，抽象了 ML Framework 和 ML Operator 模块，负责连接 Flink 和其他计算引擎。

ML Framework



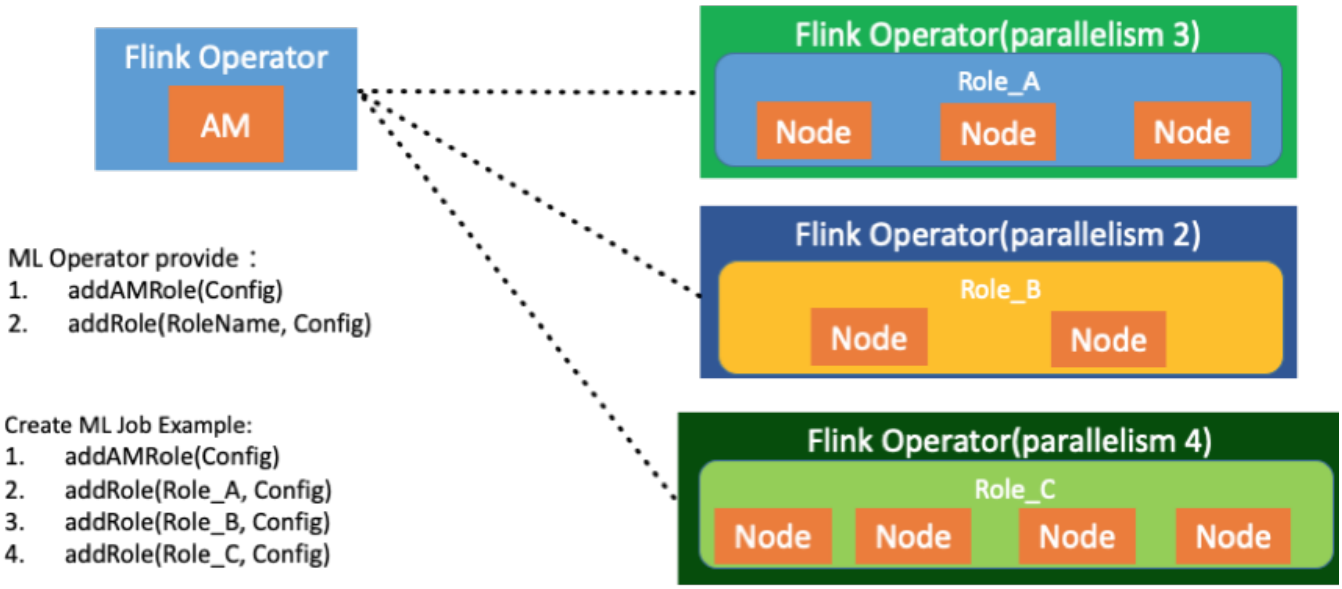
ML Framework 分为 2 个角色。

1. Application Manager（以下简称 am） 角色，负责管理所有 node 的节点的生命周期。
2. node 角色，负责执行机器学习的算法程序。

在上述过程中，还可以对 Application Manager 和 node 进行进一步的抽象，Application Manager 里面我们单独把 state machine 的状态机做成可扩展的，这样就可以支持不同类型的作业。深度学习引擎，可以自己定义其

状态机。从 node 的节点抽象 runner 接口，这样用户就可以根据不同的深度学习引擎去自定义运行算法程序。

ML Operator

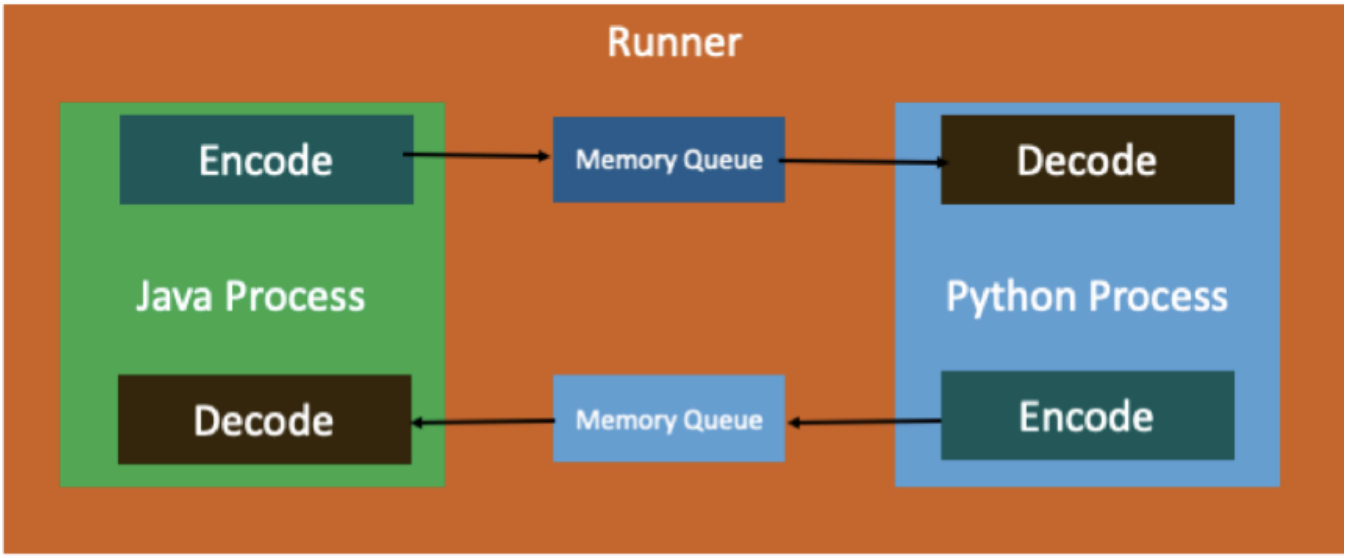


ML Operator 模块提供了两个接口：

1. addAMRole，这个接口的作用是在 Flink 的作业里添加一个 Application Manager 的角色。Application Manager 角色如上图所示就是机器学习集群的管理节点。
2. addRole，增加的是机器学习的一组节点。

利用 ML Operator 提供的接口，可以实现 Flink Operator 中包含一个Application Manager 及 3 组 node 的角色，这三组 node 分别叫 role a、 role b,、role c，三个不同角色组成机器学习的一个 cluster。如上图代码所示。Flink 的 operator 与机器学习作业的 node 一一对应。

机器学习的 node 节点运行在 Flink 的 operator 里，需要进行数据交换，原理如下图所示：



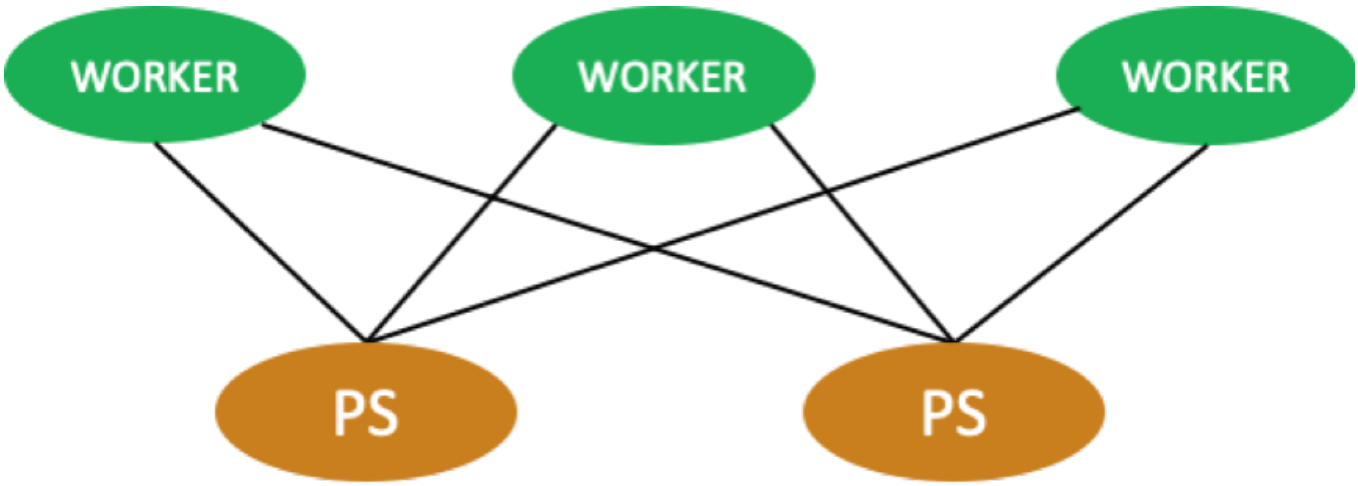
1. Encode transfer user define object to byte[]
2. Decode transfer byte[] to user define object

Encode and Decode is extensible

Flink operator 是 java 进程，机器学习的 node 节点一般是 python 进程，java 和 python 进程通过共享内存交换数据。

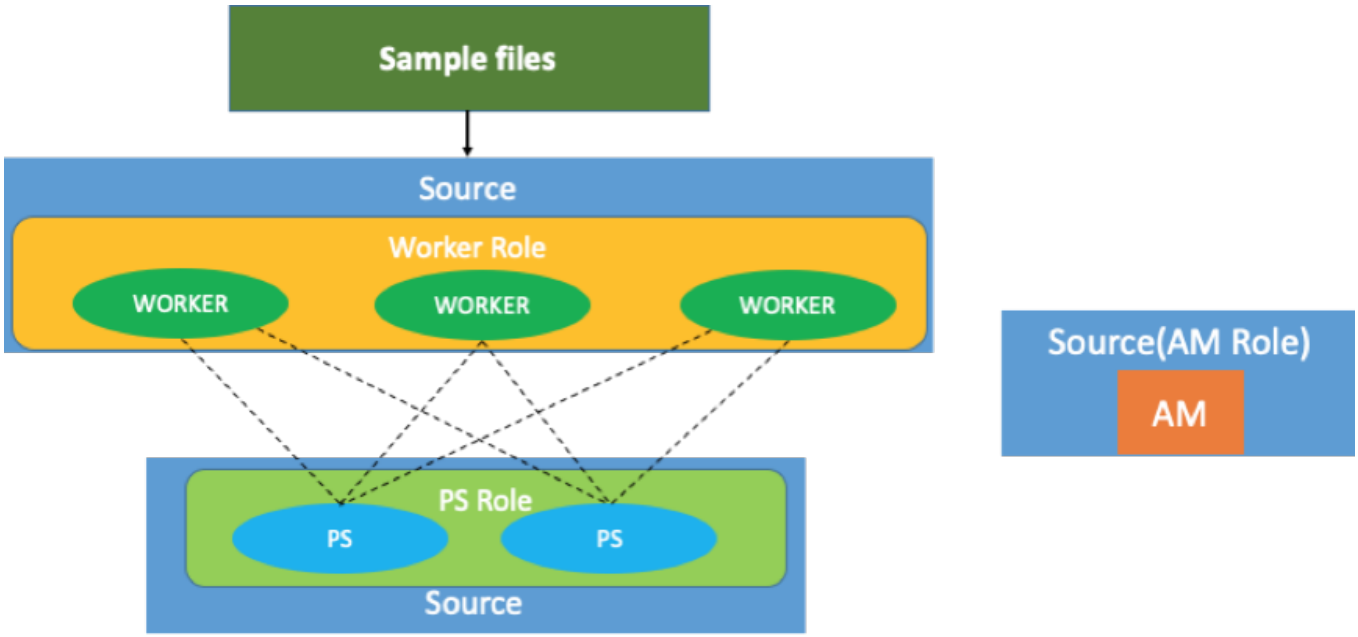
TensorFlow On Flink

TensorFlow 分布式运行



TensorFlow 分布式训练一般分为 worker 和 ps 角色。worker 负责机器学习计算，ps 负责参数更新。下面将讲解 TensorFlow 如何运行在 Flink 集群中。

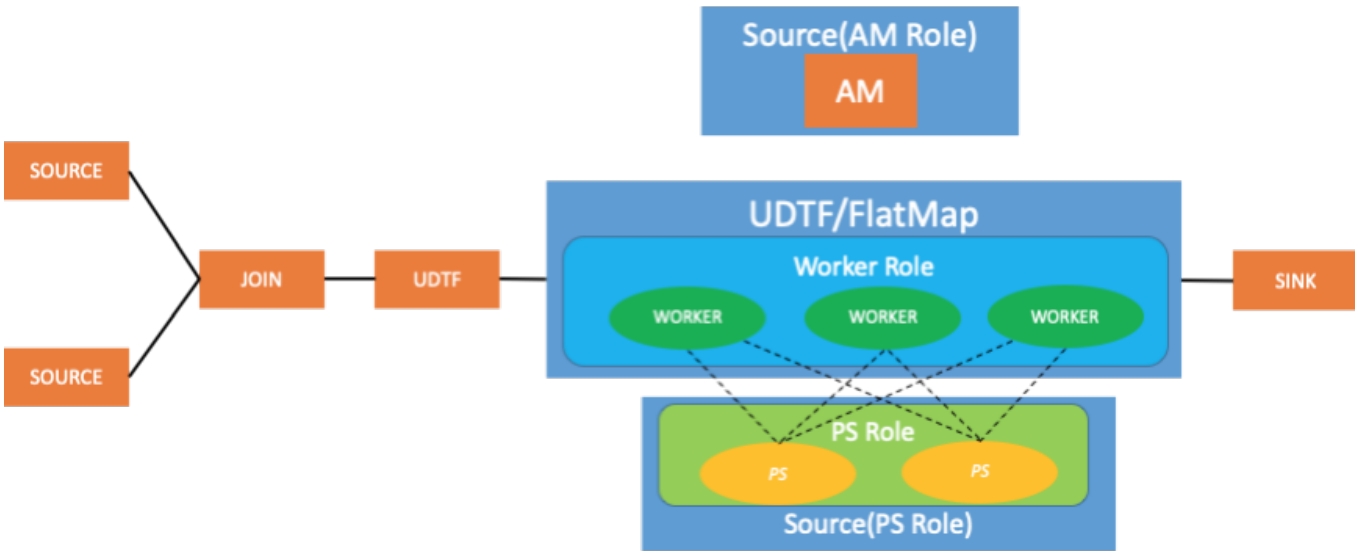
TensorFlow Batch 训练运行模式



Batch 模式下，样本数据可以是放在 HDFS 上的，对于 Flink 作业而言，它会起一个source 的 operator，然后 TensorFlow 的 work 角色就会启动。如上图所示，如果 worker 的角色有三个节点，那么 source 的并行度就会设为 3。同理下面 ps 角色有 2 个，所以 ps source 节点就会设为 2。

而 Application Manager 和别的角色并没有数据交换，所以 Application Manager 是单独的一个节点，因此它的 source 节点并行度始终为 1。这样 Flink 作业上启动了三个 worker 和两个 ps 节点，worker 和 ps 之间的通讯是通过原始的 TensorFlow 的 GRPC 通讯来实现的，并不是走 Flink 的通信机制。

TensorFlow stream 训练运行模式

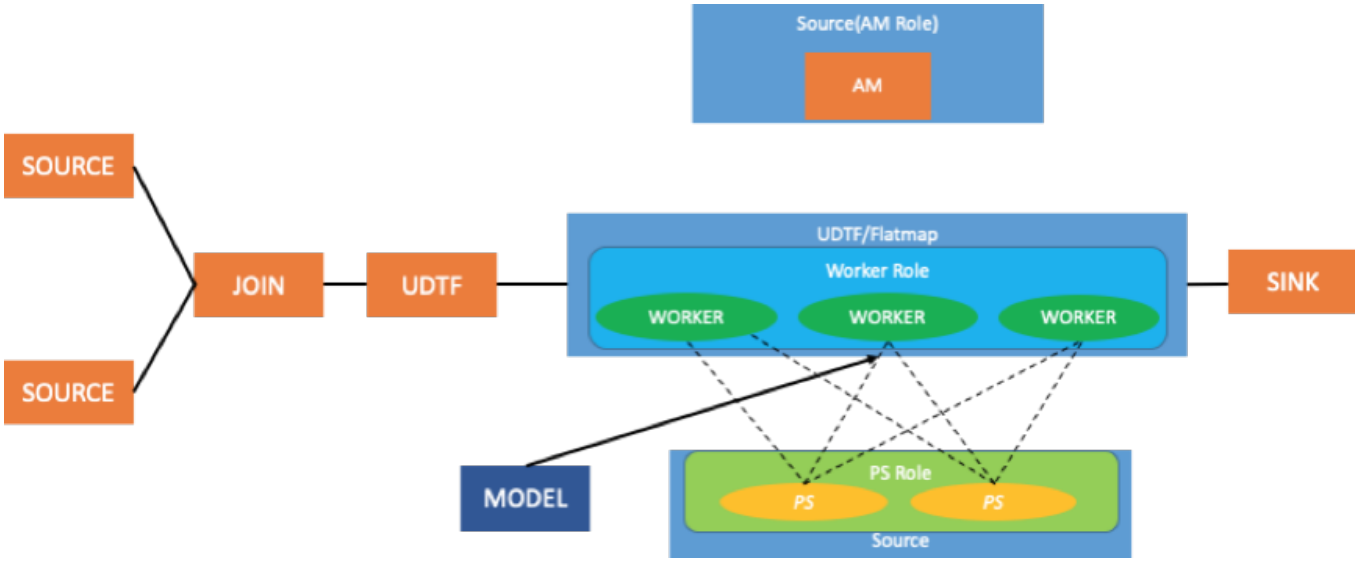


如上图所示，前面有两个 source operator，然后接 join operator，把两份数据合并为一份数据，再加自定义处理的节点，生成样本数据。在 stream 模式下，worker 的角色是通过 UDTF 或者 flatmap 来实现的。

同时，TensorFlow worker node 有3 个，所以 flatmap 和 UDTF 相对应的 operator 的并行度也为 3， 由于ps 角色并不去读取数据，所以是通过 flink source operator 来实现。

下面我们再讲一下，如果已经训练好的模型，如何去支持实时的预测。

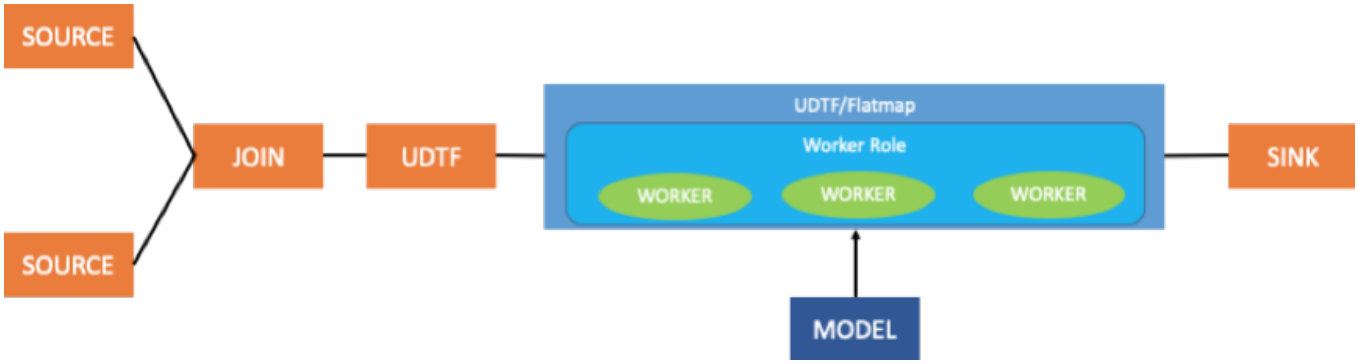
使用 Python 进行预测



使用 Python 进行预测流程如图所示，如果 TensorFlow 的模型是分布式训练出来的模型，并且这个模型非常大，比如说单机放不下的情况，一般出现在推荐和搜索的场景下。那么实时预测和实时训练原理相同，唯一不同的地方是多了一个加载模型的过程。

在预测的情况下，通过读取模型，将所有的参数加载到 ps 里面去，然后上游的数据还是经过和训练时候一样的处理形式，数据流入到 worker 这样一个角色中去进行处理，将预测的分数再写回到 flink operator，并且发送到下游 operator。

使用 Java 进行预测



如图所示，模型单机进行预测时就没必要再去起 ps 节点，单个 worker 就可以装下整个模型进行预测，尤其是使用 TensorFlow 导出 save model。同时，因为 saved model 格式包含了整个深度学习预测的全部计算逻辑和输入输出，所以不需要运行 Python 的代码就可以进行预测。

此外，还有一种方式可以进行预测。前面 source、join、UDTF 都是对数据进行加工处理变成预测模型可以识别的数据格式，在这种情况下，可以直接在 Java 进程里面通过 TensorFlow Java API，将训练好的模型 load 到内存里，这时会发现并不需要 ps 角色， worker 角色也都是 Java 进程，并不是 Python 的进程，所以我们可以直接在 Java 进程内进行预测，并且可以将预测结果继续发给 Flink 的下游。

总结

在本文中，我们讲解了 flink-ai-extended 原理，以及 Flink 结合 TensorFlow 如何进行模型训练和预测。希望通过本文大分享，大家能够使用 flink-ai-extended， 通过 Flink 作业去支持模型训练和模型的预测。