30 | 流处理平台：Flink是如何快速识别用户兴趣，实现实时推荐的？

你好，我是王喆。

刚刚结束的 2020 年双 11 活动，让技术圈出现了一个非常劲爆的新闻，就是阿里基于 Flink，实现了数据的批流一体处理，每秒能够处理 40 亿条的巨量数据。这也是业界首次在这么大规模的数据洪峰之上，实现数据流的实时处理。

也正是因为实时数据流处理功能的实现，让阿里的推荐系统引擎能够在双 11 期间做出更快速的反应，实时抓住用户的兴趣，给出更准确的推荐。

这节课，我就带你揭开阿里使用流处理平台 Flink 的面纱，来重点解决这 3 个问题：

1. 为什么说实时性是影响推荐系统效果的关键因素？
2. 到底什么是批流一体的数据处理体系？
3. 业界流行的 Flink 到底是怎么实现数据流处理的？

为什么实时性是影响推荐系统效果的关键因素？

周星驰的电影《功夫》里有一句著名的台词“天下武功，无坚不摧，唯快不破”。如果说推荐模型的架构是那把“无坚不摧”的“玄铁重剑”，那么推荐系统的实时性就是“唯快不破”的“柳叶飞刀”。那这把柳叶飞刀到底是怎么发挥作用的呢？我说个切身的场景你就明白了。

假设，你正在手机上刷抖音，你刚刚看了一个精彩的足球进球视频，感觉意犹未尽，还想看更多这样的视频。这个时候，抖音的推荐系统

肯定不会让你失望，它很快会接收到“你观看了精彩进球视频”的信号，快速地抓到你的兴趣点，然后，它会迅速做出反馈，给你推送更多类似的视频。

那我们也可以试想一下，如果抖音的推荐系统实时性不够的话，会发生什么呢？可能是你看完了精彩进球的视频之后，推荐系统还跟什么都没发生一样，依然按部就班地给你推荐一些原本就设定好的不相关视频。如果是这样的话，抖音又怎么能让你欲罢不能，刷了又想刷呢？

这个例子就充分说明了，**推荐系统只有拥有实时抓住用户新兴趣点的能力，才能让你的用户“离不开你”**。

什么是批流一体的数据处理体系？

那作为推荐系统工程师，我们就要思考，到底怎样才能实现用户兴趣的快速提取呢？这就不得不提到推荐系统的数据体系。

我们之前讲的数据处理，无论是数据的预处理，还是特征工程，大部分是在 Spark 平台上完成的。Spark 平台的特点是，它处理的数据都是已经落盘的数据。也就是说，这些数据要么是在硬盘上，要么是在分布式的文件系统上，然后才会被批量地载入到 Spark 平台上进行运算处理，这种批量处理大数据的架构就叫做批处理大数据架构。它的整体结构图如图 1 所示。

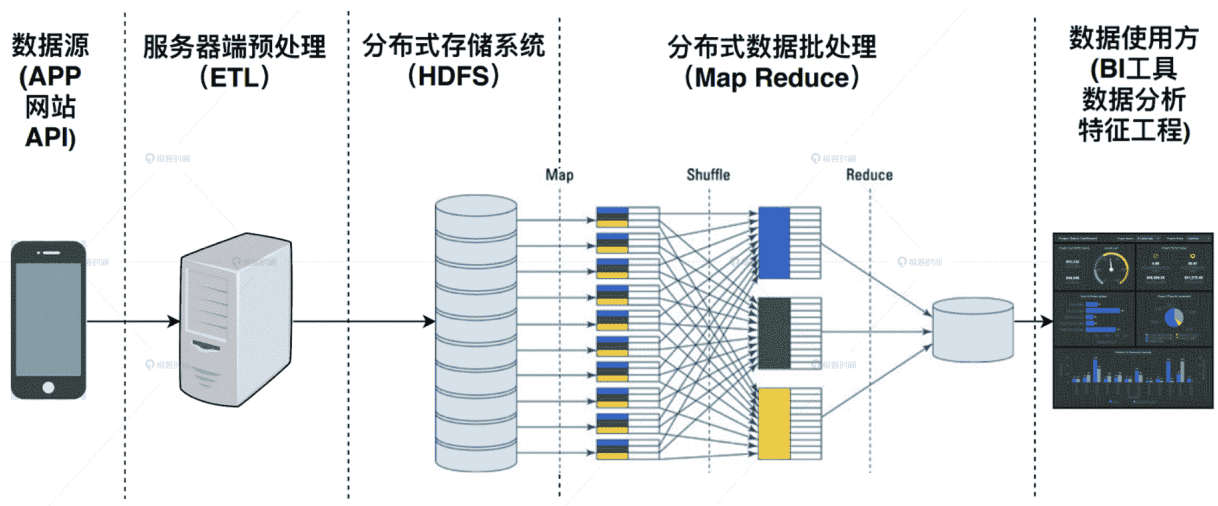

图1 传统批处理大数据架构

但批处理架构的特点就是慢，数据从产生到落盘，再到被 Spark 平台重新读取处理，往往要经历几十分钟甚至几小时的延迟。如果推荐系统是建立在这样的数据处理架构上，还有可能实时地抓住用户的新兴趣点吗？肯定是没有希望了。

那怎么办呢？我们能不能在数据产生之后就立马处理它，而不是等到它落盘后再重新处理它呢？当然是可以的，这种在数据产生后就直接对数据流进行处理的架构，就叫做流处理大数据架构。

我们从图 2 的流处理架构示意图中可以看到，它和批处理大数据架构相比，不仅用流处理平台替换掉了分布式批处理 Map Reduce 计算平台，而且在数据源与计算平台之间，也不再有存储系统这一层。这就大大提高了数据处理的速度，让数据的延迟可以降低到几分钟级别，甚至一分钟以内，这也让实时推荐成为了可能。

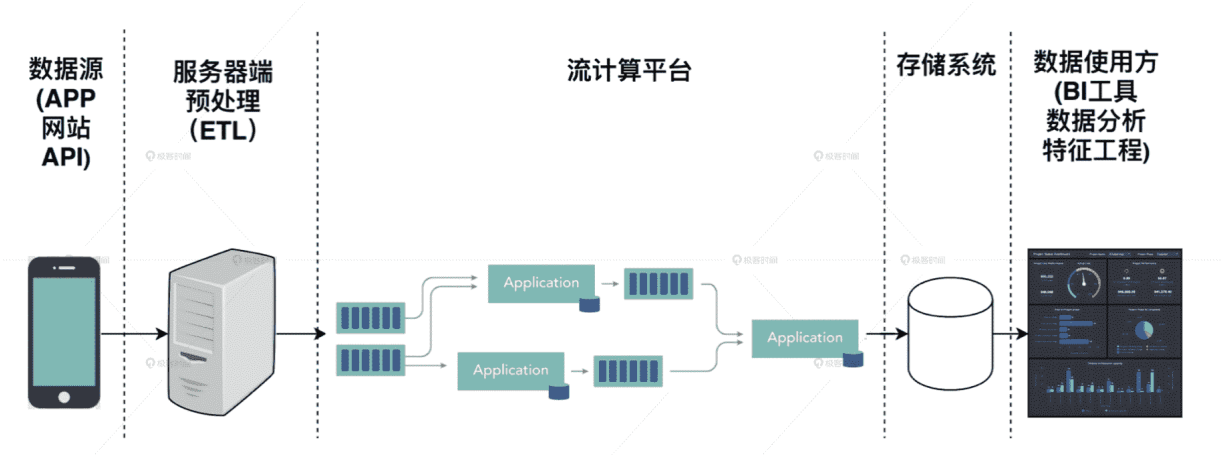

图2 流处理大数据架构

但是，流处理平台也不是十全十美的。由于流处理平台是对数据流进行直接处理，它没有办法进行长时间段的历史数据的全量处理，这就让流处理平台无法应用在历史特征的提取，模型的训练样本生成这样非常重要的领域。

那是不是说，根本就没有能够同时具有批处理、流处理优势的解决方案吗？当然是有的，这就是我们在一开始说的，批流一体的大数据架构，其中最有代表性的就是 Flink。

如下图 3 所示，**批流一体的大数据架构最重要的特点，就是在流处理架构的基础上添加了数据重播的功能。**

我们怎么理解这个数据重播功能呢？它指的是在数据落盘之后，还可以利用流处理平台同样的代码，进行落盘数据的处理，这就相当于进行了一遍重播。这样不就实现了离线环境下的数据批处理了吗？而且由于流处理和批处理使用的是一套代码，因此完美保证了代码维护的一致性，是近乎完美的数据流解决方案。

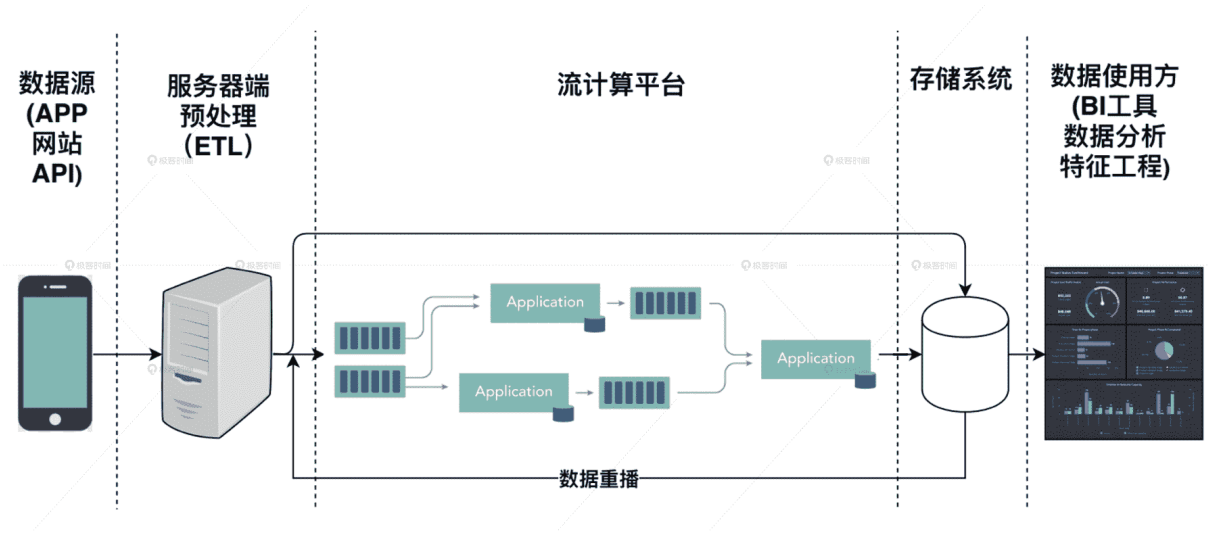

图3 批流一体大数据架构

既然批流一体的大数据架构这么完美，为什么我们很少听说有实现这套方案的公司呢？以我个人的实践经验来看，这主要是因为它实现起来有下面两个难点。

1. 大批成熟的互联网公司已经在 Spark 等批处理平台上，构建起了整套的数据体系，要想完全迁移到批流一体的数据体系上，有着非常沉重的技术负担。
2. 批流一体的解决方案还很理想化，因为我们在实际处理特征的时候，很难让批处理和流处理完全共享一套代码。

比如，我们在流处理中可以很方便地计算出点击量、曝光量这类方便累计的指标，但如果遇到比较复杂的特征，像是用户过去一个月的平均访问时长，用户观看视频的进度百分比等等，这些指标就很难在流处理中计算得到了。这是因为计算这类特征所需的数据时间跨度大，计算复杂，流处理难以实现。

因此，**在对待流处理平台时，我们的态度应该是，取其所长**。更具体点来说就是，在需要实时计算的地方发挥它的长处，但也没有必要过

于理想主义，强调一切应用都应该批流一体，这反而会为我们增加过多的技术负担。

Flink 是如何处理数据流的？

现在，我们已经清楚流处理平台的特点和优势了，但 Flink 平台到底是怎么进行流数据处理的呢？

我们先来认识 Flink 中两个最重要的概念，数据流（DataStream）和窗口（Window）。数据流其实就是消息队列，从网站、APP 这些客户端中产生的数据，被发送到服务器端的时候，就是一个数据消息队列，而流处理平台就是要对这个消息队列进行实时处理。

就像下图 4 所示的那样，上面是来自三个用户的数据，其中一个一个紫色的点就是一条条数据，所有紫色的点按时间排列就形成了一个消息队列。

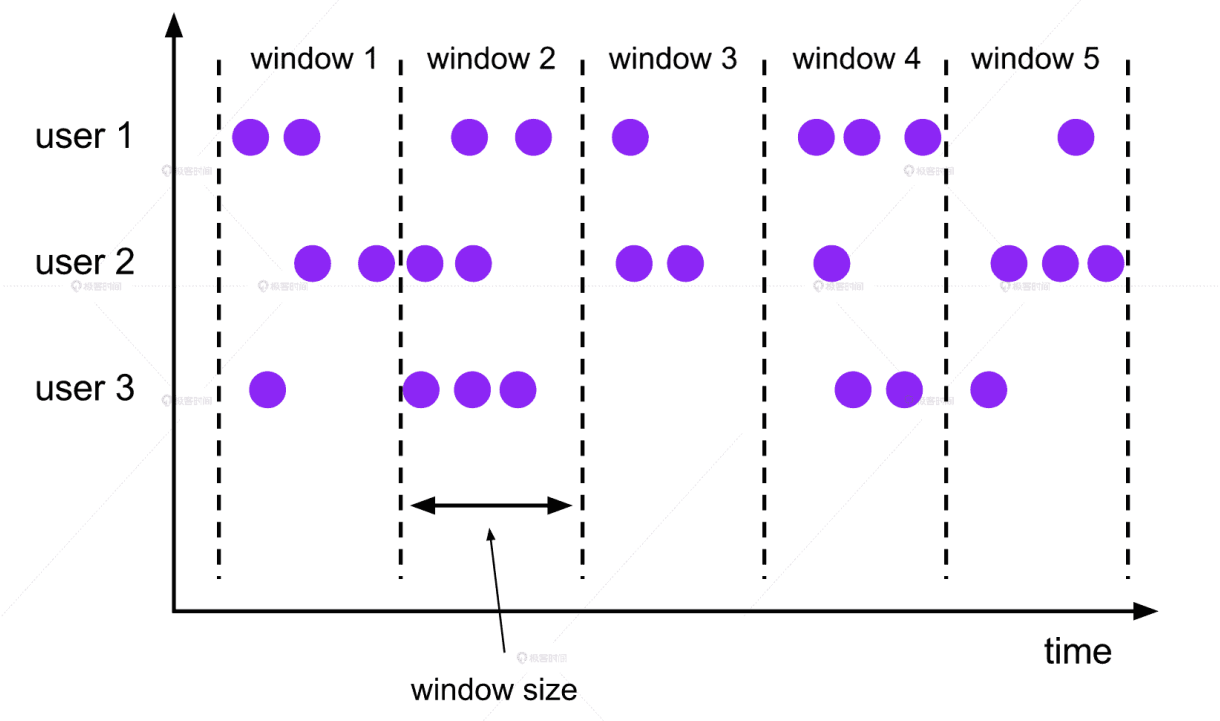

图4 数据流和窗口

知道了什么是消息队列，Flink 会怎么处理这个消息队列里的数据呢？答案很简单，就是随着时间的流失，按照时间窗口来依次处理每个时间窗口内的数据。

比如图 4 中的数据流就被分割成了 5 个时间窗口，每个窗口的长度假设是 5 分钟，这意味着每积攒够 5 分钟的数据，Flink 就会把缓存在内存中的这 5 分钟数据进行一次批处理。这样，我们就可以算出数据流中涉及物品的最新 CTR，并且根据用户最新点击的物品来更新用户的兴趣向量，记录特定物品曝光给用户的次数等等。

除了上面例子中的固定窗口以外，Flink 还提供了多种不同的窗口类型，滑动窗口（Sliding Window）也是我们经常会用到的。

滑动窗口的特点是在两个窗口之间留有重叠的部分，Flink 在移动窗口的时候，不是移动 window size 这个长度，而是移动 window slide 这个长度，window slide 的长度要小于 window size。因此，窗口内部的数据不仅包含了数据流中新进入的 window slide 长度的数据，还包含了上一个窗口的老数据，这部分数据的长度是 window size-window slide。

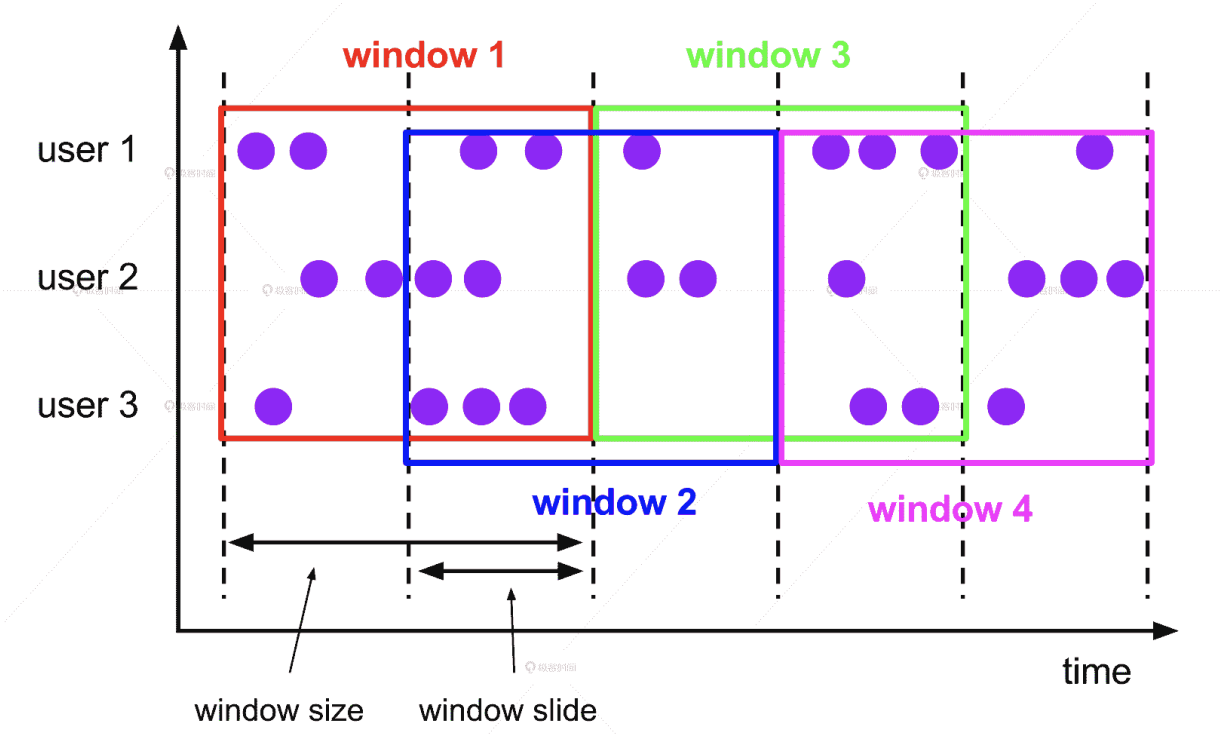

图5 Flink中的滑动窗口

那滑动窗口这种方式有什么用呢？它最典型的用处就是做一些数据的 JOIN 操作。比如我们往往需要通过 JOIN 连接一个物品的曝光数据和点击数据，以此来计算 CTR，但是你要知道，曝光数据肯定是在点击数据之前到达 Flink 的。

那如果在分窗的时候，恰好把曝光数据和点击数据分割在了两个窗口怎么办呢？那点击数据就不可能找到相应的曝光数据了。这个时候，只要我们使用滑动窗口，这个问题就迎刃而解了。因为两个窗口重叠的部分给我们留了足够的余量来进行数据 JOIN，避免数据的遗漏。

事实上，除了固定窗口和滑动窗口，Flink 还提供了更丰富的窗口操作，比如基于会话的 Session Window，全局性的 Global Window。除此之外，Flink 还具有数据流 JOIN，状态保存特性 state 等众多非常有价值的操作，想继续学习的同学可以在课后参考 Flink 的官方文

档 。我们这节课，只要清楚 Flink 的核心概念数据流和时间窗口就可以了，因为它反映了流处理平台最核心的特点。

Flink 数据流处理实践

接下来，就又到了实践的环节。我们要继续在 SparrowRecsys 项目上利用 Flink 实现一个特征更新的应用。因为没有真实的数据流环境，所以我们利用 MoviesLens 的 ratings 表来模拟一个用户评分的数据流，然后基于这个数据流，利用 Flink 的时间窗口操作，来实时地提取出用户最近的评分电影，以此来反映用户的兴趣。

下面就是 SparrowRecsys 中相关的代码（详细代码：com.sparrowrecsys.nearline.flink.RealTimeFeature）。你可以看到，我首先定义了一个评分的数据流 ratingStream，然后在处理 ratingStream 的时候，是把 userId 作为 key 进行处理。

接着，我又利用到了两个函数 timeWindow 和 reduce。利用 timeWindow 函数，我们可以把处理的时间窗口设置成 1s，再利用 reduce 函数，把每个时间窗口到期时触发的操作设置好。

在完成了 reduce 操作后，我们再触发 addSink 函数中添加的操作，进行数据存储、特征更新等操作。

```
DataStream<Rating> ratingStream = inputStream.map(
ratingStream.keyBy(rating -> rating.userId)
        .timeWindow(Time.seconds(1))
        .reduce(
                (ReduceFunction<Rating>) (rating,
                    if (rating.timestamp.compareTo
                        return rating;
                    }else{
                        return t1;
                    }
```

```
                    }
                }
        ).addSink(new SinkFunction<Rating>() {
    @Override
    public void invoke(Rating value, Context conte
        System.out.println("userId:" + value.userI
    }
});
```

看完了这些操作之后，你知道我们应该怎么把用户最近的高分电影评价历史，实时反映到推荐结果上了吗？其实很简单，我们的用户 Embedding 是通过平均用户的高分电影 Embedding 得到的，我们只需要在得到新的高分电影后，实时地更新用户 Embedding 就可以了，然后在推荐过程中，用户的推荐列表自然会发生实时的变化。这就是 SparrowRecsys 基于 Flink 的实时推荐过程。

小结

这节课我们讲了流处理平台 Flink 特点，并且通过 Flink 的实践清楚了，利用流处理平台提高推荐系统实时性的方法。

Flink 是最具代表性的批流一体的大数据平台。它的特点就是，让批处理和流处理共用一套代码，从而既能批量处理已落盘的数据，又能直接处理实时数据流。从理论上来说，是近乎完美的数据流解决方案。

而 Flink 提高推荐系统实时性的原理可以理解为是，用户数据进入数据流，也就是数据消息队列后，会被分割成一定时长的时间窗口，之后 Flink 会按照顺序来依次处理每个时间窗口内的数据，计算出推荐系统需要的特征。这个处理是直接在实时数据流上进行的，所以相比原来基于 Spark 的批处理过程，实时性有了大幅提高。

为了方便你复习，我把这节课的核心概念总结在了下面的表格里，希望你能记住它们。

知识点	要点描述
批处理大数据架构	批量处理已经落盘数据的大数据架构
流处理大数据架构	在数据产生后，直接对数据流进行处理的大数据架构
批流一体大数据架构	融合批处理和流处理架构， 用一套代码同时进行批处理和流处理的架构
Flink的数据流	数据的消息队列
Flink的时间窗口	Flink每次进行数据处理需要等待的数据流时长， 可细分为固定窗口、滑动窗口、会话窗口、全局窗口

至于 Flink 的实时性实践，我们要记住，利用 Flink 我们可以实时地获取到用户刚刚评价过的电影，然后通过实时更新用户 Embedding，就可以实现 SparrowRecsys 的实时推荐了。

课后思考

1. 你觉得实时性是不是对所有推荐系统都非常重要？比如对于抖音、快手这类短视频应用，还有优酷、Netflix 这类长视频应用，实时性对哪个更重要一些？为什么？
2. Flink 要加强的往往是数据的实时性，特征的实时性，你觉得模型训练的实时性重要吗？模型训练的实时性发挥的作用和特征实时性有什么不同呢？

期待在留言区看到你对 Flink 的思考和疑惑，我们下节课见！