

## 21-深入对比Spark与Flink：帮你系统设计两开花

你好，我是蔡元楠。

今天我要与你分享的主题是“深入对比Spark与Flink”。

相信通过这一模块前9讲的学习，你对Spark已经有了基本的认识。现在，我们先来回顾整个模块，理清一下思路。

首先，从MapReduce框架存在的问题入手，我们知道了Spark的主要优点，比如用内存运算来提高性能；提供很多High-level API；开发者无需用map和reduce两个操作实现复杂逻辑；支持流处理等等。

接下来，我们学习了Spark的数据抽象——RDD。RDD是整个Spark的核心概念，所有的新API在底层都是基于RDD实现的。但是RDD是否就是完美无缺的呢？显然不是，它还是很底层，不方便开发者使用，而且用RDD API写的应用程序需要大量的人工调优来提高性能。

Spark SQL提供的DataFrame/DataSet API就解决了这个问题，它提供类似SQL的查询接口，把数据看成关系型数据库的表，提升了熟悉关系型数据库的开发者的工作效率。这部分内容都是专注于数据的批处理，那么我们很自然地就过渡到下一个问题：Spark是怎样支持流处理的呢？

那就讲到了Spark Streaming和新的Structured Streaming，这是Spark的流处理组件，其中Structured Streaming也可以使用DataSet/DataFrame API，这就实现了Spark批流处理的统一。

通过这个简单的回顾我们发现，Spark的发布，和之后各个版本新功能的发布，并不是开发人员拍脑袋的决定，每个新版本发布的功能都是在解决旧功能的问题。在如此多的开源工作者的努力下，Spark生态系统才有今天的规模，成为了当前最流行的大数据处理框架之一。

在开篇词中我就提到过，我希望你能通过这个专栏建立自己的批判性思维，遇到一个新的技术，多问为什么，而不是盲目的接受和学习。只有这样我们才能不随波逐流，成为这个百花齐放的技术时代的弄潮儿。

所以，这里我想问你一个问题，Spark有什么缺点？

这个缺点我们之前已经提到过一个——无论是Spark Streaming还是Structured Streaming，Spark流处理的实时性还不够，所以无法用在一些对实时性要求很高的流处理场景中。

这是因为Spark的流处理是基于所谓微批处理（Micro-batch processing）的思想，即它把流处理看作是批处理的一种特殊形式，每次接收到一个时间间隔的数据才会去处理，所以天生很难在实时性上有所提升。

虽然在Spark 2.3中提出了连续处理模型（Continuous Processing Model），但是现在只支持很有限的功能，并不能在大的项目中使用。Spark还需要做出很大的努力才能改进现有的流处理模型。

想要在流处理的实时性上提升，就不能继续用微批处理的模式，而要想办法实现真正的流处理，即每当有一条数据输入就立刻处理，不做等待。那么当今时代有没有这样的流处理框架呢？

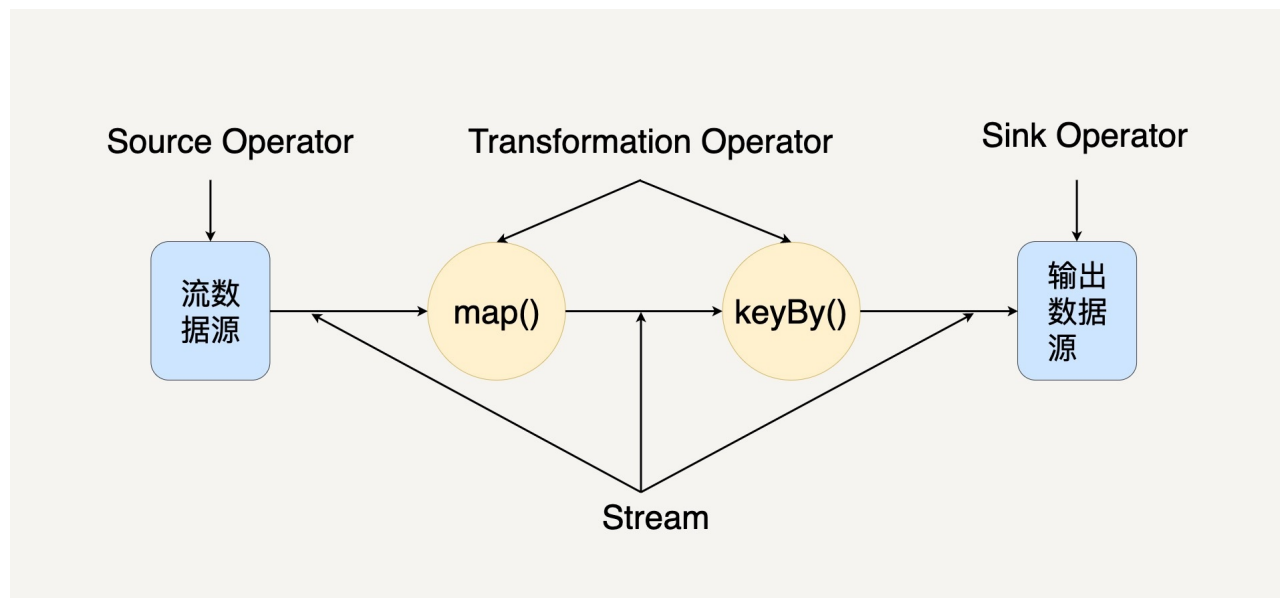
Apache Flink就是其中的翘楚。它采用了基于操作符（Operator）的连续流模型，可以做到微秒级别的延迟。今天我就带你一起了解一下这个流行的数据处理平台，并将Flink与Spark做深入对比，方便你在今后的实际项目中做出选择。

## Flink核心模型简介

Flink中最核心的数据结构是Stream，它代表一个运行在多个分区上的并行流。

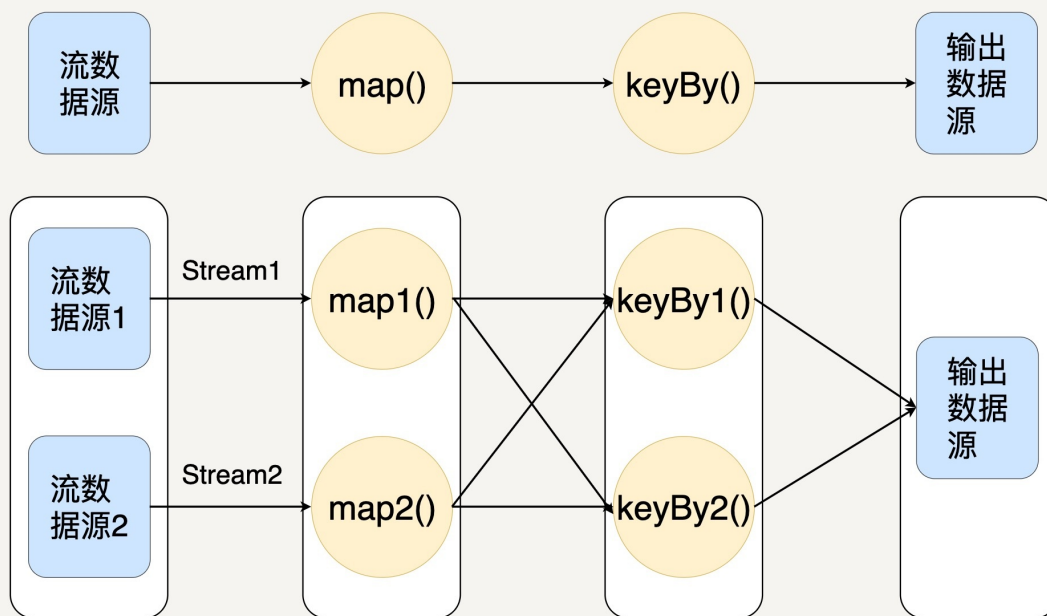
在Stream上同样可以进行各种转换操作（Transformation）。与Spark的RDD不同的是，Stream代表一个数据流而不是静态数据的集合。所以，它包含的数据是随着时间增长而变化的。而且Stream上的转换操作都是逐条进行的，即每当有新的数据进来，整个流程都会被执行并更新结果。这样的基本处理模式决定了Flink会比Spark Streaming有更高的流处理延迟性。

当一个Flink程序被执行的时候，它会被映射为Streaming Dataflow，下图就是一个Streaming Dataflow的示意图。



在图中，你可以看出Streaming Dataflow包括Stream和Operator（操作符）。转换操作符把一个或多个Stream转换成多个Stream。每个Dataflow都有一个输入数据源（Source）和输出数据源（Sink）。与Spark的RDD转换图类似，Streaming Dataflow也会被组合成一个有向无环图去执行。

在Flink中，程序天生是并行和分布式的。一个Stream可以包含多个分区（Stream Partitions），一个操作符可以被分成多个操作符子任务，每一个子任务是在不同的线程或者不同的机器节点中独立执行的。如下图所示：

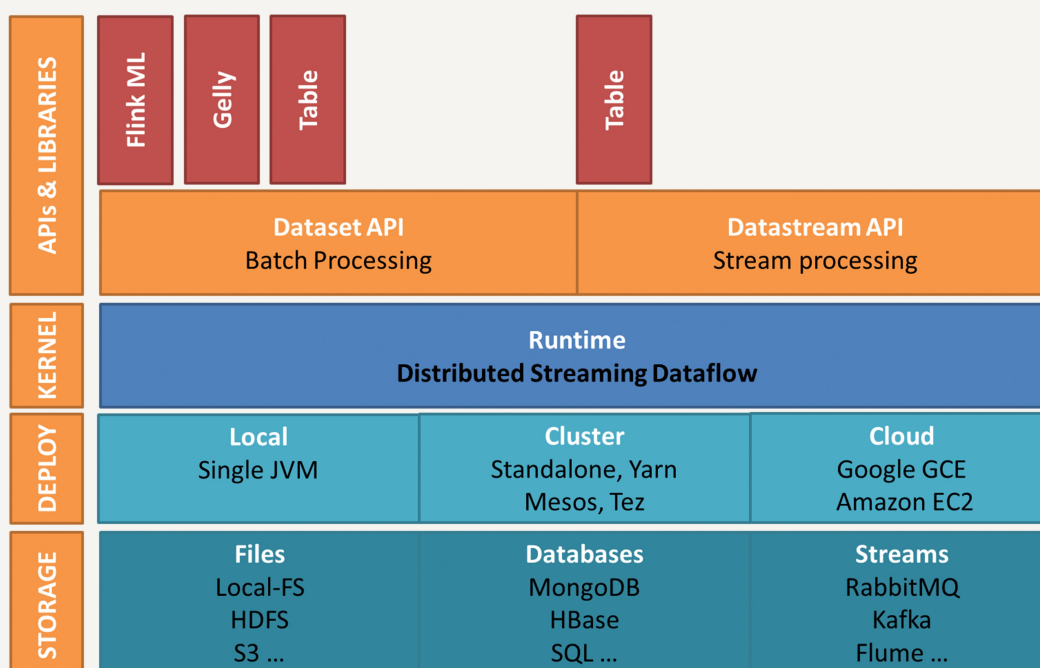


从上图你可以看出，Stream在操作符之间传输数据的形式有两种：一对一和重新分布。

- 一对一（One-to-one）：Stream维护着分区以及元素的顺序，比如上图从输入数据源到map间。这意味着map操作符的子任务处理的数据和输入数据源子任务生产的元素的数据相同。你有没有发现，它与RDD的窄依赖类似。
- 重新分布（Redistributing）：Stream中数据的分区会发生变化，比如上图中map与keyBy之间。操作符的每一个子任务把数据发送到不同的目标子任务。

## Flink的架构

当前版本Flink的架构如下图所示。



我们可以看到，这个架构和[第12讲](#)中介绍的Spark架构比较类似，都分为四层：存储层、部署层、核心处理引擎、high-level的API和库。

从存储层来看，Flink同样兼容多种主流文件系统如HDFS、Amazon S3，多种数据库如HBase和多种数据流如Kafka和Flume。

从部署层来看，Flink不仅支持本地运行，还能在独立集群或者在被 YARN 或 Mesos 管理的集群上运行，也能部署在云端。

核心处理引擎就是我们刚才提到的分布式Streaming Dataflow，所有的高级API及应用库都会被翻译成包含Stream和Operator的Dataflow来执行。

Flink提供的两个核心API就是DataSet API和DataStream API。你没看错，名字和Spark的DataSet、DataFrame非常相似。顾名思义，DataSet代表有界的数据集，而DataStream代表流数据。所以，DataSet API是用来做批处理的，而DataStream API是做流处理的。

也许你会问，Flink这样基于流的模型是怎样支持批处理的？在内部，DataSet其实也用Stream表示，静态的有界数据也可以被看作是特殊的流数据，而且DataSet与DataStream可以无缝切换。所以，Flink的核心是DataStream。

DataSet和DataStream都支持各种基本的转换操作如map、filter、count、groupBy等，让我们来看一个用DataStream实现的统计词频例子。

```
public class WindowWordCount {
    public static void main(String[] args) throws Exception {
        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

        DataStream<Tuple2<String, Integer>> dataStream = env
            .socketTextStream("localhost", 9999)
            .flatMap(new Splitter())
            .keyBy(0)
            .timeWindow(Time.seconds(5))
            .sum(1);

        dataStream.print();
        env.execute("Window WordCount");
    }

    public static class Splitter implements FlatMapFunction<String, Tuple2<String, Integer>> {
        @Override
        public void flatMap(String sentence, Collector<Tuple2<String, Integer>> out) {
            for (String word: sentence.split(" ")) {
                out.collect(new Tuple2<String, Integer>(word, 1));
            }
        }
    }
}
```

这里我是用Java来示范的，因为Flink就是用Java开发的，所以它对Java有原生的支持。此外，也可以用Scala来开发Flink程序，在1.0版本后更是支持了Python。

在这个例子中，我们首先创建了一个Splitter类，来把输入的句子拆分成（词语，1）的对。在主程序中用

StreamExecutionEnvironment创建DataStream，来接收本地Web Socket的文本流，并进行了4步操作。

1. 用flatMap把输入文本拆分成（词语，1）的对；
2. 用keyBy把相同的词语分配到相同的分区；
3. 设好5秒的时间窗口；
4. 对词语的出现频率用sum求和。

可以看出，DataStream的使用方法和RDD比较相似，都是把程序拆分成一系列的转换操作并分布式地执行。

在DataSet和DataStream之上，有更高层次的Table API。Table API和Spark SQL的思想类似，是关系型的API，用户可以像操作SQL数据库表一样的操作数据，而不需要通过写Java代码、操作DataStream/DataSet的方式进行数据处理，更不需要手动优化代码的执行逻辑。

此外，Table API同样统一了Flink的批处理和流处理。

## Flink和Spark对比

通过前面的学习，我们了解到，Spark和Flink都支持批处理和流处理，接下来让我们对这两种流行的数据处理框架在各方面进行对比。

首先，这两个数据处理框架有很多相同点。

- 都基于内存计算；
- 都有统一的批处理和流处理API，都支持类似SQL的编程接口；
- 都支持很多相同的转换操作，编程都是用类似于Scala Collection API的函数式编程模式；
- 都有完善的错误恢复机制；
- 都支持Exactly once的语义一致性。

当然，它们的不同点也是相当明显，我们可以从4个不同的角度来看。

**从流处理的角度来讲**，Spark基于微批量处理，把流数据看成是一个个小的批处理数据块分别处理，所以延迟性只能做到秒级。而Flink基于每个事件处理，每当有新的数据输入都会立刻处理，是真正的流式计算，支持毫秒级计算。由于相同的原因，Spark只支持基于时间的窗口操作（处理时间或者事件时间），而Flink支持的窗口操作则非常灵活，不仅支持时间窗口，还支持基于数据本身的窗口，开发者可以自由定义想要的窗口操作。

**从SQL功能的角度来讲**，Spark和Flink分别提供SparkSQL和Table API提供SQL交互支持。两者相比较，Spark对SQL支持更好，相应的优化、扩展和性能更好，而Flink在SQL支持方面还有很大提升空间。

**从迭代计算的角度来讲**，Spark对机器学习的支持很好，因为可以在内存中缓存中间计算结果来加速机器学习算法的运行。但是大部分机器学习算法其实是一个有环的数据流，在Spark中，却是用无环图来表示。而Flink支持在运行时间中的有环数据流，从而可以更有效的对机器学习算法进行运算。

**从相应的生态系统角度来讲**，Spark的社区无疑更加活跃。Spark可以说有着Apache旗下最多的开源贡献

者，而且有很多不同的库来用在不同场景。而Flink由于较新，现阶段的开源社区不如Spark活跃，各种库的功能也不如Spark全面。但是Flink还在不断发展，各种功能也在逐渐完善。

## 小结

今天我们从Spark存在的一个缺点——无法高效应对低延迟的流处理场景入手，一起学习了另一个主流流数据处理框架Flink，还对比了这两个框架的异同，相信现在你对两个框架肯定有了更多的认识。

我经常被问到的一个问题是：Spark和Flink到底该选哪一个？对于这个问题，我们还是要分一下场景。

对于以下场景，你可以选择Spark。

1. 数据量非常大而且逻辑复杂的批数据处理，并且对计算效率有较高要求（比如用大数据分析来构建推荐系统进行个性化推荐、广告定点投放等）；
2. 基于历史数据的交互式查询，要求响应较快；
3. 基于实时数据流的数据处理，延迟性要求在在数百毫秒到数秒之间。

Spark完美满足这些场景的需求，而且它可以一站式解决这些问题，无需用别的数据处理平台。

由于Flink是为了提升流处理而创建的平台，所以它适用于各种需要非常低延迟（微秒到毫秒级）的实时数据处理场景，比如实时日志报表分析。

而且Flink用流处理去模拟批处理的思想，比Spark用批处理去模拟流处理的思想扩展性更好，所以我相信将来Flink会发展的越来越好，生态和社区各方面追上Spark。比如，阿里巴巴就基于Flink构建了公司范围内全平台使用的数据处理平台Blink，美团、饿了么等公司也都接受Flink作为数据处理解决方案。

可以说，Spark和Flink都在某种程度上统一了批处理和流处理，但也都有一些不足。下一模块中，让我们一起来学习一个全新的、完全统流处理的数据处理平台——Apache Beam，到时候我们会对Spark的优缺点有更加深入的认识。

## 思考题

除了高延迟的流处理这一缺点外，你认为Spark还有什么不足？可以怎样改进？

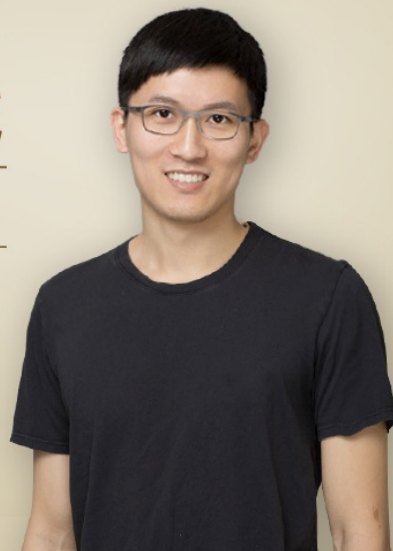
欢迎你把答案写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

# 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。

## 精选留言：

- leben krieg 2019-06-05 19:44:47  
老师能详细解释一下这句话吗？  
“由于相同的原因，Spark 只支持基于时间的窗口操作（处理时间或者事件时间），而 Flink 支持的窗口操作则非常灵活，不仅支持时间窗口，还支持基于数据本身的窗口，开发者可以自定义想要的窗口操作。”
- cricket1981 2019-06-05 09:01:17  
spark根据算子依赖类型将计算过程划分成多个stage，只有上一个stage全部完成才能进入下一个stage，而flink无此限制。