toc: true title: 《从0到1学习Flink》—— Data Source 介绍 date: 2018-10-28 tags:

- Flink
- 大数据
- 流式计算

前言

Data Sources 是什么呢？就字面意思其实就可以知道：数据来源。

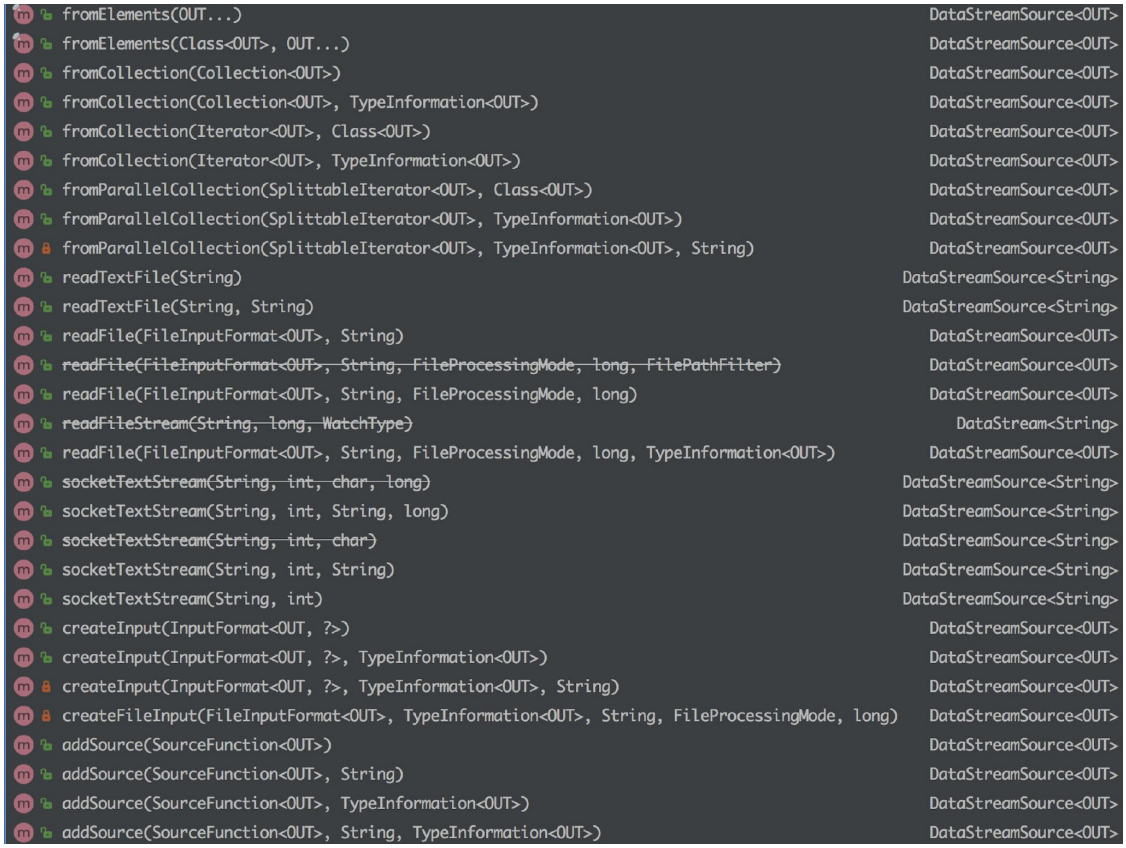
Flink 做为一款流式计算框架，它可用来做批处理，即处理静态的数据集、历史的数据集；也可以用来做流处理，即实时的处理些实时数据流，实时的产生数据流结果，只要数据源源不断的过来，Flink 就能够一直计算下去，这个 Data Sources 就是数据的来源地。

Flink 中你可以使用 `StreamExecutionEnvironment.addSource(sourceFunction)` 来为你的程序添加数据来源。

Flink 已经提供了若干实现好了的 source functions，当然你也可以通过实现 SourceFunction 来自定义非并行的 source 或者实现 ParallelSourceFunction 接口或者扩展 RichParallelSourceFunction 来自定义并行的 source，

Flink

StreamExecutionEnvironment 中可以使用以下几个已实现的 stream sources，



总的来说可以分为下面几大类：

基于集合

1、fromCollection(Collection) - 从 Java 的 Java.util.Collection 创建数据流。集合中的所有元素类型必须相同。

2、fromCollection(Iterator, Class) - 从一个迭代器中创建数据流。Class 指定了该迭代器返回元素的类型。

3、fromElements(T ...) - 从给定的对象序列中创建数据流。所有对象类型必须相同。

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<Event> input = env.fromElements(
    new Event(1, "barfoo", 1.0),
    new Event(2, "start", 2.0),
    new Event(3, "foobar", 3.0),
    ...
);
```

4、fromParallelCollection(SplittableIterator, Class) - 从一个迭代器中创建并行数据流。Class 指定了该迭代器返回元素的类型。

5、generateSequence(from, to) - 创建一个生成指定区间范围内的数字序列的并行数据流。

基于文件

1、readTextFile(path) - 读取文本文件，即符合 TextInputFormat 规范的文件，并将其作为字符串返回。

```
final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<String> text = env.readTextFile("file:///path/to/file");
```

2、readFile(fileInputFormat, path) - 根据指定的文件输入格式读取文件（一次）。

3、readFile(fileInputFormat, path, watchType, interval, pathFilter, typeInfo) - 这是上面两个方法内部调用的方法。它根据给定的 fileInputFormat 和读取路径读取文件。根据提供的 watchType，这个 source 可以定期（每隔 interval 毫秒）监测给定路径的新数据（FileProcessingMode.PROCESS_CONTINUOUSLY），或者处理一次路径对应文件的数据并退出（FileProcessingMode.PROCESS_ONCE）。你可以通过 pathFilter 进一步排除掉需要处理的文件。

```
final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<MyEvent> stream = env.readFile(
        myFormat, myFilePath,
FileProcessingMode.PROCESS_CONTINUOUSLY, 100,
        FilePathFilter.createDefaultFilter(), typeInfo);
```

实现:

在具体实现上，Flink 把文件读取过程分为两个子任务，即目录监控和数据读取。每个子任务都由单独的实体实现。目录监控由单个非并行（并行度为1）的任务执行，而数据读取由并行运行的多个任务执行。后者的并行性等于作业的并行性。单个目录监控任务的作用是扫描目录（根据 watchType 定期扫描或仅扫描一次），查找要处理的文件并把文件分割成切分片（splits），然后将这些切分片分配给下游 reader。reader 负责读取数据。每个切分片只能由一个 reader 读取，但一个 reader 可以逐个读取多个切分片。

重要注意：

如果 watchType 设置为 FileProcessingMode.PROCESS_CONTINUOUSLY，则当文件被修改时，其内容将被重新处理。这会打破“exactly-once”语义，因为在文件末尾附加数据将导致其所有内容被重新处理。

如果 watchType 设置为 FileProcessingMode.PROCESS_ONCE，则 source 仅扫描路径一次然后退出，而不等待 reader 完成文件内容的读取。当然 reader 会继续阅读，直到读取所有的文件内容。关闭 source 后就不会再有检查点。这可能导致节点故障后的恢复速度较慢，因为该作业将从最后一个检查点恢复读取。

基于 Socket：

socketTextStream(String hostname, int port) - 从 socket 读取。元素可以用分隔符切分。

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<Tuple2<String, Integer>> dataStream = env
        .socketTextStream("localhost", 9999) // 监听 localhost 的
9999 端口过来的数据
        .flatMap(new Splitter())
        .keyBy(0)
        .timeWindow(Time.seconds(5))
        .sum(1);
```

这个在 《从0到1学习Flink》—— Mac 上搭建 Flink 1.6.0 环境并构建运行简单程序入门 文章里用的就是基于 Socket 的 Word Count 程序。

自定义：

addSource - 添加一个新的 source function。例如，你可以 addSource(new FlinkKafkaConsumer011<>(...)) 以从 Apache Kafka 读取数据

说下上面几种的特点吧：

1、基于集合：有界数据集，更偏向于本地测试用

2、基于文件：适合监听文件修改并读取其内容

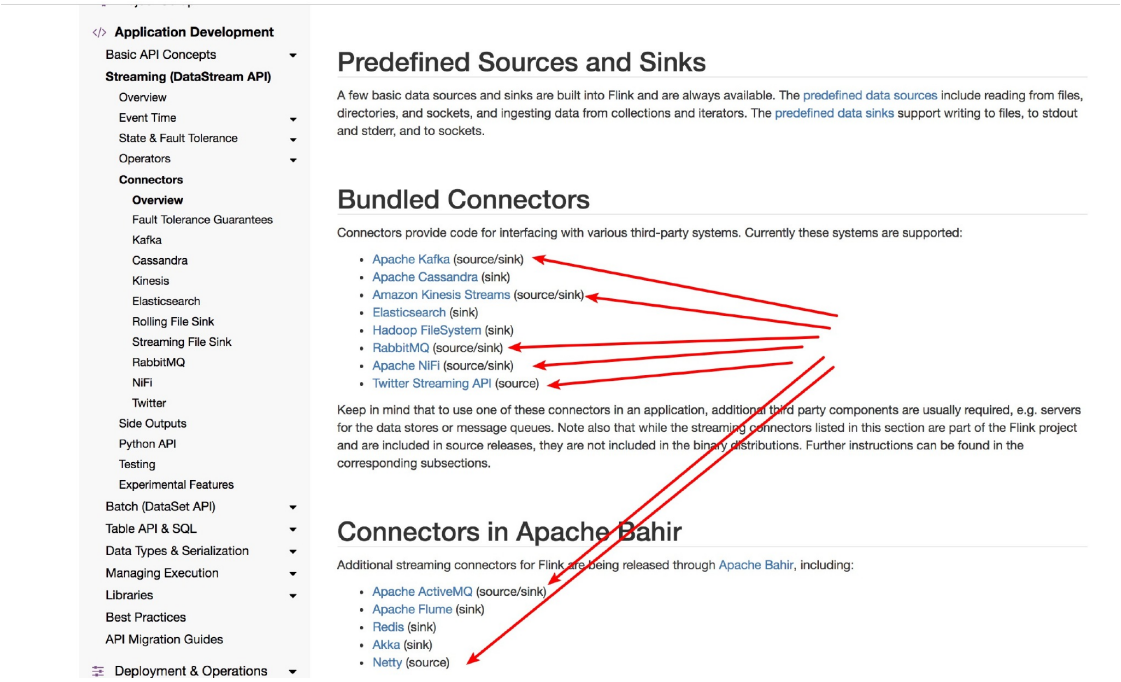
3、基于 Socket：监听主机的 host port，从 Socket 中获取数据

4、自定义 addSource：大多数的场景数据都是无界的，会源源不断的过来。比如去消费 Kafka 某个 topic 上的数据，这时候就需要用到这个 addSource，可能因为用的比较多的原因吧，Flink 直接提供了 FlinkKafkaConsumer011 等类可供你直接使用。你可以去看看 FlinkKafkaConsumerBase 这个基础类，它是 Flink Kafka 消费的最根本的类。

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<KafkaEvent> input = env
        .addSource(
            new FlinkKafkaConsumer011<>(
                parameterTool.getRequired("input-topic"), //从参数中
获取传进来的 topic
                new KafkaEventSchema(),
                parameterTool.getProperties())
            .assignTimestampsAndWatermarks(new
CustomWatermarkExtractor()));
```

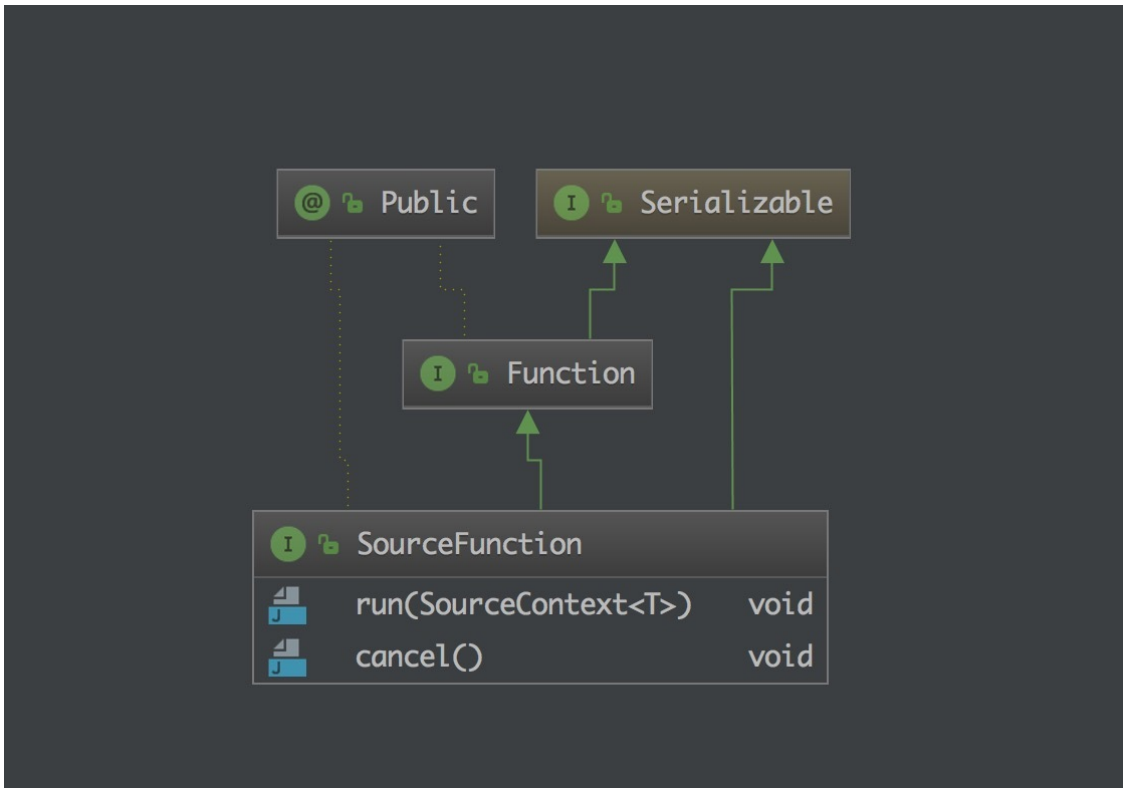
Flink 目前支持如下图里面常见的 Source：



如果你想自己自定义自己的 Source 呢?

那么你就需要去了解一下 SourceFunction 接口了，它是所有 stream source 的根接口，它继承自一个标记接口（空接口）Function。

SourceFunction 定义了两个接口方法：



1、run： 启动一个 source，即对接一个外部数据源然后 emit 元素形成 stream（大部分情况下会通过在该方法里运行一个 while 循环的形式来产生 stream）。

2、cancel： 取消一个 source，也即将 run 中的循环 emit 元素的行为终止。

正常情况下，一个 SourceFunction 实现这两个接口方法就可以了。其实这两个接口方法也固定了一种实现模板。

比如，实现一个 XXXSourceFunction，那么大致的模板是这样的：（直接拿 FLink 源码的实例给你看看）

```
*/
public class HBaseWriteStreamExample {

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

        // data stream with random numbers
        DataStream<String> dataStream = env.addSource(new SourceFunction<String>() {
            private static final long serialVersionUID = 1L;

            private volatile boolean isRunning = true;

            @Override
            public void run(SourceContext<String> out) throws Exception {
                while (isRunning) {
                    out.collect(String.valueOf(Math.floor(Math.random() * 100)));
                }

            }

            @Override
            public void cancel() { isRunning = false; }
        });
        dataStream.writeUsingOutputFormat(new HBaseOutputFormat());

        env.execute();
    }

    /**
```

最后

本文主要讲了下 Flink 的常见 Source 有哪些并且简单的提了下如何自定义 Source。

关注我

转载请务必注明原创地址为：http://www.54tianzhisheng.cn/2018/10/28/flink-sources/

另外我自己整理了些 Flink 的学习资料，目前已经全部放到微信公众号了。你可以加我的微信：zhisheng_tian，然后回复关键字：Flink 即可无条件获取到。

Github 代码仓库

https://github.com/zhisheng17/flink-learning/

以后这个项目的所有代码都将放在这个仓库里，包含了自己学习 flink 的一些 demo 和博客

相关文章

1、《从0到1学习Flink》—— Apache Flink 介绍

2、《从0到1学习Flink》—— Mac 上搭建 Flink 1.6.0 环境并构建运行简单程序入门

3、《从0到1学习Flink》—— Flink 配置文件详解

4、《从0到1学习Flink》—— Data Source 介绍

5、《从0到1学习Flink》—— 如何自定义 Data Source ？

6、《从0到1学习Flink》—— Data Sink 介绍

7、《从0到1学习Flink》—— 如何自定义 Data Sink？

8、《从0到1学习Flink》—— Flink Data transformation(转换)

9、《从0到1学习Flink》—— 介绍Flink中的Stream Windows

10、《从0到1学习Flink》—— Flink 中的几种 Time 详解

11、《从0到1学习Flink》—— Flink 写入数据到 ElasticSearch