Spark Streaming

技术内幕及源码剖析

王家林 夏 阳 编著

- 彻底解密Spark Streaming**架构设计**
- 深入挖掘Spark Streaming**运行机制**
- 逐行剖析Spark Streaming**框架源码**
- 手把手讲解Spark Streaming**性能调优**

Spark Streaming 技术内幕及源码剖析

王家林 夏 阳 编著

清華大學出版社
北京

内 容 简 介

本书以大数据处理引擎 Spark 的稳定版本 1.6.x 为基础，从应用案例、原理、源码、流程、调优等多个角度剖析 Spark 上的实时计算框架 Spark Streaming。在勾勒出 Spark Streaming 架构轮廓的基础上，从基本源码开始进行剖析，由浅入深地引导已具有 Spark 和 Spark Streaming 基础技术知识的读者进行 Spark Streaming 的进阶学习，理解 Spark Streaming 的原理和运行机制，为流数据处理的决策和应用提供了技术参考；结合 Spark Streaming 的深入应用的需要，对 Spark Streaming 的性能调优进行了分析，也对 Spark Streaming 功能的改造和扩展提供了指导。

本书适合大数据领域 CTO、架构师、高级软件工程师，尤其是 Spark 领域已有 Spark Streaming 基础知识的从业人员阅读，也可供需要深入学习 Spark、Spark Streaming 的高校研究生和高年级本科生参考。

图书在版编目(CIP)数据

Spark Streaming技术内幕及源码剖析 / 王家林，夏阳编著. — 北京：清华大学出版社，2017
ISBN 978-7-302-46491-4

Ⅰ. ①S… Ⅱ. ①王… ②夏… Ⅲ. ①数据处理软件 Ⅳ. ①TP274

中国版本图书馆 CIP 数据核字（2017）第 025588 号

责任编辑： 袁金敏 战晓雷
封面设计： 刘新新
责任校对： 徐俊伟
责任印制： 李红英

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 清华大学印刷厂
经　　销： 全国新华书店
开　　本： 185mm×230mm　　**印　　张：** 16.25　　**字　　数：** 264 千字
版　　次： 2017 年 5 月第 1 版　　**印　　次：** 2017 年 5 月第 1 次印刷
印　　数： 1 ～ 3000
定　　价： 49.00 元

产品编号：072312-01

前 言

大数据浪潮汹涌来袭，这绝不仅仅是信息技术领域的革命，更是在全球范围引领社会变革的机遇。大数据的集群计算开源软件Spark在大数据计算平台应用领域日益凸显其重要地位。如果大数据技术领域从业人员的技术水平仍停留在只知使用开源软件，而不从开源软件的原理、架构上去理解，不到源码中去体会细节，则难以从根本上彻底解决现实工作中遇到的技术问题，更难以胜任大数据领域的技术创新工作。

可以预见，大数据的处理将越来越强调实时处理。Spark Streaming是建立在Spark上的实时计算框架，在Spark的各子框架中处于举足轻重的地位。彻底掌握 Spark Streaming的同时，也能加深对Spark Core技术的理解和掌握，还能具备开发同样高端的Spark应用程序的实力。对于有志向的Spark学习进阶者来说，深入了解Spark Streaming的源码是提高核心竞争力的捷径。

本书不仅对Spark Streaming的API做总结性介绍，而且重点针对Spark 1.6.x的Spark Streaming进行源码剖析。该书的开始部分对Spark的基本原理有一些阐述，但主要是彻底深入剖析Spark Streaming的内部原理。

读源码的人都怕自己走进大量源码的迷宫。为了提高源码学习效率，本书在剖析源码前，会对源码实现的功能的大致原理和流程轮廓进行介绍。书中有方便源码剖析的流程图，这对于理解和掌握Spark Streaming的各个功能非常重要。读者看到复杂的流程图时不一定要立刻全部理解掌握，但可以在源码学习过程中经常回过头来对照流程图以加深印象。

为了在书的页面内清晰展示复杂的流程图，书中绝大多数流程图采取了从上至下的树状结构来体现调用关系。每个方框中注明了类和方法，被其调用的类的方法会在下一行从左至右依次显示，调用和被调用的类方法间用有向线连接。有些方框上部会给出类的成员

变量，其类型就是方框中指明的类。粗箭头不是表示调用关系，而是表示传递消息。

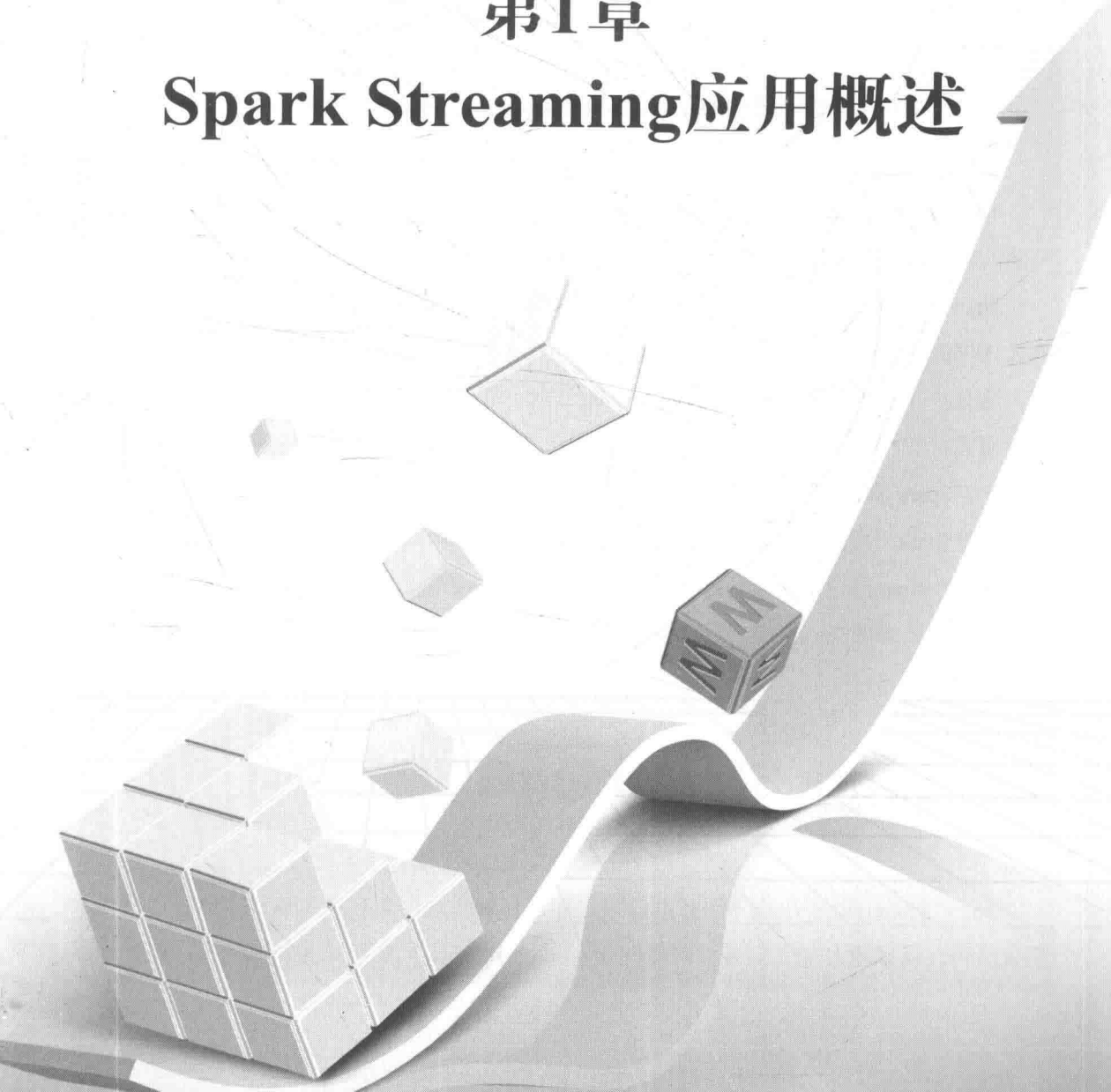
源码剖析过程中，源码中关键的类名、方法名、注释会以粗体显示，使读者清楚重点。读者应以粗体部分为重点进行阅读，其他部分可以粗略浏览。有些源码篇幅过大，可能会省略其中的部分代码，以突出当前读者需要阅读的源码主体。

王家林　夏　阳

2017年2月27日于北京

目 录

第1章
Spark Streaming应用概述



Spark是一个类似于Hadoop的MapReduce的分布式计算框架，其核心是弹性分布式数据集（Resilient Distributed Dataset，RDD），提供了比MapReduce更丰富的模型，可以在内存中快速对数据集进行多次迭代，以支持复杂的数据挖掘算法和图形计算算法。

Spark包含Spark Core、Spark SQL、Spark Streaming、MLlib、GraphX等主要部件，如图1-1所示。

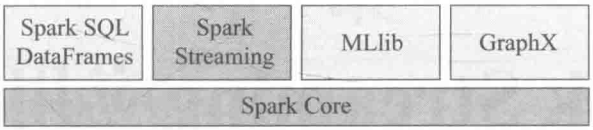


图1-1 Spark的主要部件

其中的Spark Core包含Spark的基本功能，是核心的计算引擎，并定义了RDD的API。其他的Spark部件都是构建在Spark Core之上的。

Spark Streaming是建立在Spark上的实时计算框架，提供了丰富的API，以支持Spark处理大规模流式数据。

这是一个流处理的时代。Spark十分强大的一个重要原因在于，它的流式处理可以在线使用图计算、机器学习或者SparkR的成果，这得益于Spark一体化、多元化的基础架构设计。在Spark Streaming中可以调用Spark SQL、MLlib、GraphX等其他子框架，无须任何设置。这是Spark无可匹敌之处。但Spark的应用中，Spark Streaming也是很容易出问题的。

Spark Streaming与其他子框架不同之处在于，它更像是Spark Core之上的一个复杂应用程序。如果要做Spark的定制开发，Spark Streaming的剖析也会提供最好的参考。

1.1 Spark Streaming应用案例

Spark Streaming应用程序运行的时候，往往在短时间内会产生大量日志信息，不利于研究分析。可以通过加大批处理时间间隔（batch interval）来降低批处理频率，减少日志

信息量，以便看清楚各个环节。

下面从一个Spark Streaming应用程序的开发入手，观察运行过程，以增强感性认识。以下是一个广告点击的在线黑名单过滤的Spark Streaming应用程序，程序中有详细注释，以方便初次接触Spark Streaming的读者理解。

源码1-1 OnlineBlackListFilter

```
package com.dt.spark.streaming

import org.apache.spark.SparkConf
import org.apache.spark.streaming.StreamingContext
import org.apache.spark.streaming.Seconds

object OnlineBlackListFilter {
    def main(args: Array[String]){
      /**
       * 第1步：创建Spark的配置对象SparkConf，设置Spark程序的运行时的配置信息，
       * 例如，通过setMaster来设置程序要链接的Spark集群的Master的URL，如果设置
       * 为local，则代表Spark程序在本地运行，特别适合于机器配置条件非常差（例如
       * 只有1GB内存）的初学者
       */
      // 创建SparkConf对象
      val conf = new SparkConf()
      // 设置应用程序的名称，在程序运行的监控界面可以看到名称
      conf.setAppName("OnlineBlackListFilter")
      // 此时，程序在Spark集群
      conf.setMaster("spark://Master:7077")
```

```
val ssc = new StreamingContext(conf, Seconds(30))

/**
 * 黑名单数据准备。实际上黑名单一般都是动态的，例如在Redis或者数据库中
 * 黑名单的生成往往有复杂的业务逻辑，具体情况算法不同
 * 但是在Spark Streaming进行处理的时候每次都能够访问完整的信息
 */
val blackList = Array(("Spy", true),("Cheater", true))
val blackListRDD = ssc.sparkContext.parallelize(blackList, 8)

val adsClickStream = ssc.socketTextStream("Master", 9999)

/**
 * 此处模拟的广告点击的每条数据的格式为：time、name
 * 此处map操作的结果是name、（time，name）的格式
 */
val adsClickStreamFormatted = adsClickStream.map { ads => (ads.split("")(1), ads) }
adsClickStreamFormatted.transform(userClickRDD => {
  // 通过leftOuterJoin操作既保留了左侧用户广告点击内容的RDD的所有内容，
  // 又获得了相应点击内容是否在黑名单中
val joinedBlackListRDD = userClickRDD.leftOuterJoin(blackListRDD)

  /**
   * 进行filter过滤的时候，其输入元素是一个元组：（name,((time,name), boolean)）
   * 其中，第一个元素是黑名单的名称
```

```
        * 第二元素的第二个元素boolean是进行leftOuterJoin的时候是否存在的值。
        * 如果存在，表明当前广告点击是黑名单，需要过滤掉，否则是有效点击内容
        */
      val validClicked = joinedBlackListRDD.filter(joinedItem => {
        if(joinedItem._2._2.getOrElse(false))
        {
          false
        } else {
          true
        }

      })

      validClicked.map(validClick => {validClick._2._1})
    }).print

    ssc.start()
    ssc.awaitTermination()

  }
}
```

此程序接收Socket信息，过滤掉其中名称为Spy、Cheater的信息，并打印输出。

把程序的批处理时间间隔设置从30s改成300s：

```
val ssc = new StreamingContext(conf, Seconds(300))
```

然后重新生成一下jar包。

Spark集群有5台机器：Master、Worker1、Worker2、Worker3、Worker4。

启动Spark的History Server。

打开数据发送的端口：

```
nc -lk 9999
```

用spark-submit运行前面生成的jar包。

在数据发送端口输入若干数据，例如：

```
1375864674543 Tom
1375864674553 Spy
1375864674571 Andy
1375864688436 Cheater
1375864784240 Kelvin
1375864853892 Steven
1375864979347 John
```

每行第一项为时刻的毫秒数，第二项是程序中要过滤的名称。

打开浏览器，看History Server的日志信息，如图1-2所示。

图中按时间顺序显示了曾经运行过的应用程序，第一列是App ID，有各应用程序执行信息的链接。

单击最新的应用，看目前运行的应用程序中有些什么Job，如图1-3所示。

这样一个Spark Streaming应用程序运行时总共有5个Job。

观察这些Job的内容，可以揭示一些现象。

Job 0不体现应用程序的业务逻辑代码，如图1-4所示。其实此Job是Spark Streaming出于对后面计算的负载均衡的考虑而产生的。

图1-2 History Server日志信息示例

图1-3 Spark Jobs页面示例

图1-4 Details for Job 0示例

Job 0包含Stage 0、Stage 1。随便看一个Stage，比如Stage 0，看看其中的Aggregated Metrics by Executor部分，如图1-5所示。

图1-5 Aggregated Metrics by Executor页面示例

因为是分布式环境做负载均衡，所以Job 0 的Stage 1是在4个Worker的Executor上运行。

Job 1的运行时间比较长，耗时1.5min，如图1-6所示。

图1-6 页面示例：Details for Job 1

单击Stage 2的链接，看看Aggregated Metrics By Executor部分，如图1-7所示。

图1-7 页面示例：Aggregated Metrics by Executor

可以知道，Stage 2只在Worker4上的一个Executor执行，而且执行了1.5min。

从业务处理的角度看，此前发送了很少的数据，这里却显示有一个运行1.5min的任务。这个任务是做什么呢？

从DAG Visualization部分可以知道此Job实际就是启动了一个接收数据的接收器（Receiver），如图1-8所示。

原来Receiver是通过一个Job来启动的。

Tasks部分如图1-9所示。

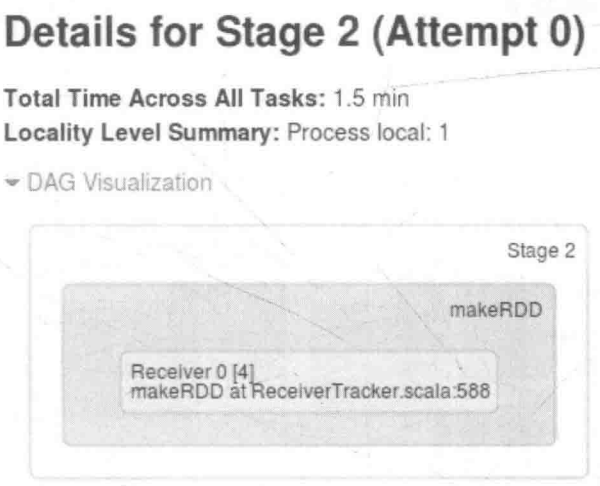


图1-8 页面示例：Details for Stage 2

图1-9 页面示例：Tasks

只有一个Worker运行此Job，用于接收数据。

Spark Streaming应用程序启动后，自己会启动一些Job。默认情况是启动一个Job接收数据，为后续处理做准备。

Locality Level是PROCESS_LOCAL。所以，默认情况下，数据接收不会使用磁盘，而是直接使用内存中的数据。

从Job 2的Details可以发现程序的主要业务逻辑，体现在Stage 3、Stage 4、Stage 5中，如图1-10所示。

仔细观察Stage 3、Stage 4，可以知道这两个Stage都是用4个Executor执行的，所有数据处理是在4台机器上进行的，如图1-11所示。

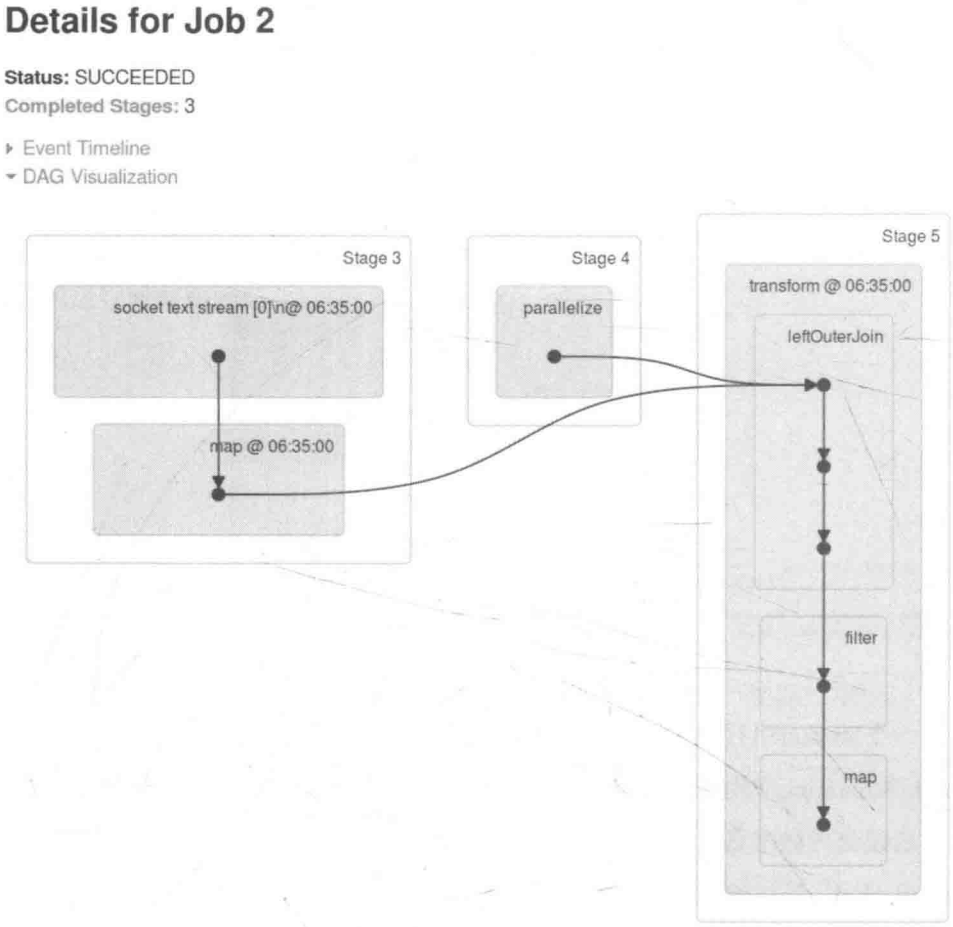


图1-10 页面示例：Details for Job 2

Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Shuffle Write Size / Records
0	Worker1:53116	2 s	1	0	1	152.0 B / 1
1	Worker4:34159	8 s	3	0	3	466.0 B / 3
2	Worker2:39749	4 s	2	0	2	309.0 B / 2
3	Worker3:50215	6 s	2	0	2	314.0 B / 2

图1-11 页面示例：Aggregated Metrics by Executor

Stage 5只在Worker4上，这是因为这个Stage有Shuffle操作。

Job 3有Stage 6、Stage 7、Stage 8，其中Stage 6、Stage 7显示灰色，说明被跳过，如图1-12所示。