云计算与大数据实验教材系列

Spark

案例与实验教程

主编　袁景凌　熊盛武　饶文碧

WUHAN UNIVERSITY PRESS
武汉大学出版社

内容提要

本书是一本大数据和分布式计算领域入门阶段的实验教材，结合实例介绍了Spark基本概念、开发环境、基础案例、进阶实践、性能调优等内容。每章由知识要点和案例实践两部分组成，内容由浅到深，循序渐进。其中，知识要点主要是对学生实验过程中碰到的概念和原理进行了简单的介绍和讨论。案例实践部分选取了典型的应用实例，帮助学生通过实践将理论知识与实际应用有机结合，解决学以致用的问题。

本书最大的特色是把知识和案例相结合，基于Spark分布式计算平台，针对每章内容设计了大量的实践案例，帮助学生更好地理解和运用Spark的相关知识。

■ 责任编辑 / 张　欣
■ 责任校对 / 汪欣怡
■ 版式设计 / 马　佳
■ 封面设计 / 罗　π

ISBN 978-7-307-12842-2

定价: 26.00元

云计算与大数据实验教材系列

Spark

案例与实验教程

主编　袁景凌　熊盛武　饶文碧

WUHAN UNIVERSITY PRESS
武汉大学出版社

图书在版编目(CIP)数据

Spark 案例与实验教程/袁景凌,熊盛武,饶文碧主编. —武汉:武汉大学出版社,2017.4

云计算与大数据实验教材系列

ISBN 978-7-307-12842-2

Ⅰ.S… Ⅱ.①袁… ②熊… ③饶… Ⅲ.数据处理软件—教材 Ⅳ.TP274

中国版本图书馆 CIP 数据核字(2017)第 025306 号

责任编辑:张 欣 责任校对:汪欣怡 版式设计:马 佳

出版发行:**武汉大学出版社** (430072 武昌 珞珈山)

(电子邮件:cbs22@whu.edu.cn 网址:www.wdp.com.cn)

印刷:湖北民政印刷厂

开本:787×1092 1/16 印张:9.75 字数:230 千字 插页:1

版次:2017 年 4 月第 1 版 2017 年 4 月第 1 次印刷

ISBN 978-7-307-12842-2 定价:26.00 元

前　言

Spark 是 UC Berkeley AMP Lab（加州大学伯克利分校的 AMP 实验室）所开源的类 Hadoop MapReduce 的通用并行计算框架，同时也是最活跃、最热门的大数据计算平台。和 Hadoop 相比，Spark 具有许多优势，它可以将计算的中间结果保存到内存中，从而不需要读写 HDFS，这种基于内存式的计算方式能更好地适用于数据挖掘、机器学习等需要多次迭代的算法。Spark 通过 RDD 成功的构建了一体化、多元化的大数据处理体系，使得 Spark 在性能上有较强的扩展性。而且在“one stack to rule them all”的思想下构建的 Spark 生态圈，包括 Spark SQL、Spark Streaming、MLlib、GraphX 四大子框架，更使 Spark 在大数据计算领域具有得天独厚的优势。

本书目的：

本书的主要目的是介绍如何使用 Spark 进行大数据处理。Spark 当下已成为 Apache 基金会的顶级开源项目，拥有着庞大的社区支持，生态系统日益完善，技术也逐渐走向成熟。

随着 Spark 技术的不断成熟，它已被广泛应用于阿里巴巴、百度、网易、英特尔等各大公司的生产环境中，Spark 大数据技术正在如火如荼地发展，许多开发者已经了解 Spark，并且开始使用 Spark，但是关于 Spark 及其开发案例的中文资料比较匮乏，相关书籍也比较少，很多 Spark 初学者和开发人员主要的学习方式仍然限于阅读有限的官方文档、源码，以及网络上零散的博客或文章，使得学习效率较慢。本书也正是为了解决上述问题而着意编写。

一般对于初学者来说，学习一门新技术的难点在于两个方面：一是对理论知识的理解，二是如何将理论运用到实践中去。鉴于此，本书在撰写上采用了理论和案例相结合的方式，系统地介绍了 Spark 方面的知识。各章知识间，内容由浅到深，循序渐进，从最基本的 Spark 环境的安装与配置，到 Spark RDD 算子的基本操作，再到 Spark 基础实践中典型案例的实例剖析，最后到 Spark 生态圈，四个子框架的讲解与实践，贯穿整个 Spark 知识系统，从而可以帮助读者更好地理解和运用 Spark 的相关知识。

本书内容：

第 1 章：Spark 简介，内容包括介绍 Spark 的基本概念、Spark 生态圈以及如何安装和配置 Spark 开发环境。

第 2 章：Spark RDD 算子的概念及基本操作，内容包括 Spark 抽象基石 RDD 的理解、RDD 创建及保存、RDD 的两种基本操作模式 Transformation 和 Action、常用的 Key/Value 操作、RDD API 案例实践、IDE 开发环境的搭建。

第 3 章：Spark 基础实践，以七个基础实践案例：WordCount、Top K、中位数求取、

倒排索引、CountOnce、倾斜连接、股票趋势预测来讲解开发 Spark 程序的一般步骤。

第 4 章：Spark 进阶实践，内容包括 Spark 生态圈中的 Spark SQL、Spark Streaming 流式计算框架、GraphX 图计算框架、Spark MLlib 机器学习库，通过知识要点和案例结合的方式使读者更容易理解。

第 5 章：Spark 性能调优，内容包括 Spark 配置参数详解、动手实践调整 Spark 配置参数，优化 Spark 性能。

本书特点：

本书避免了纯粹的理论介绍。

本书知识点的讲解都有相应的案例，加深读者兴趣与理解。

本书对整个 Spark 生态系统有较全面的介绍。

本书中的很多实践都值得借鉴，部分案例可直接使用。

本书保留了大部分源码，初学者可以根据源码动手实践。

本书目标读者：

对大数据和分布式计算感兴趣的，想要学习 Spark，但对 Spark 原理不了解，不知如何下手的人。

了解 Spark，但希望通过案例实践加深对 Spark 的认知的读者。

开设相关课程的高校本科生和研究生。

本书由袁景凌、饶文碧、熊盛武撰写，研究生刘东领、秦凤参加了实验测试及编写。

编　者

2017 年 1 月

目　录

第1章　Spark简介

本章主要介绍Spark大数据计算框架、架构、计算模型和数据管理策略及Spark在工业界的应用。围绕Spark的BDAS项目及其子项目进行了简要介绍。目前，Spark生态系统已经发展成为一个包含多个子项目的集合，其中包含SparkSQL、Spark Streaming、GraphX、MLlib等子项目，本节只进行简要介绍，后续章节再详细阐述。

1.1　知识要点

1.1.1　Spark概述

Spark是基于内存计算的大数据并行计算框架。Spark基于内存计算，提高了在大数据环境下数据处理的实时性，同时保证了高容错性和高可伸缩性，允许用户将Spark部署在大量廉价硬件之上，形成集群。

Spark于2009年诞生于加州大学伯克利分校AMP Lab。目前，已经成为Apache软件基金会旗下的顶级开源项目。下面是Spark的发展历程。

1. Spark的历史与发展

- 2009年：Spark诞生于AMP Lab。
- 2010年：开源。
- 2013年6月：Apache孵化器项目。
- 2014年2月：Apache顶级项目。
- 2014年2月：Cloudera宣称加大Spark框架的投入来取代MapReduce。
- 2014年4月：大数据公司MapR投入Spark阵营，Apache Mahout放弃MapReduce，将使用Spark作为计算引擎。
- 2014年5月：Pivotal Hadoop集成Spark全栈。
- 2014年5月30日：Spark 1.0.0发布。
- 2014年6月：Spark 2014峰会在旧金山召开。
- 2014年7月：Hive on Spark项目启动。

目前AMPLab和Databricks负责整个项目的开发维护，很多大公司如Yahoo、Intel等参与到Spark的开发中，同时很多开源爱好者积极参与Spark的更新与维护。

AMP Lab开发以Spark为核心的BDAS时提出的目标是：one stack to rule them all，也就是说在一套软件栈内完成各种大数据分析任务。相对于MapReduce上的批量计算、迭代型计算以及基于Hive的SQL查询，Spark可以带来上百倍的性能提升。目前Spark的生

态系统日趋完善，Spark SQL 的发布、Hive on Spark 项目的启动以及大量大数据公司对 Spark 全栈的支持，让 Spark 的数据分析范式更加丰富。

2. Spark 之与 Hadoop

更准确地说，Spark 是一个计算框架，而 Hadoop 中包含计算框架 MapReduce 和分布式文件系统 HDFS，Hadoop 更广泛地说还包括在其生态系统上的其他系统，如 Hbase、Hive 等。

Spark 是 MapReduce 的替代方案，而且兼容 HDFS、Hive 等分布式存储层，可融入 Hadoop 的生态系统，以弥补缺失 MapReduce 的不足。

Spark 相比 Hadoop MapReduce 的优势如下：

① 中间结果输出

基于 MapReduce 的计算引擎通常会将中间结果输出到磁盘上，进行存储和容错。出于任务管道承接的考虑，当一些查询翻译到 MapReduce 任务时，往往会产生多个 Stage，而这些串联的 Stage 又依赖于底层文件系统(如 HDFS)来存储每一个 Stage 的输出结果。Spark 将执行模型抽象为通用的有向无环图执行计划(DAG)，这可以将多 Stage 的任务串联或者并行执行，而无须将 Stage 中间结果输出到 HDFS 中，类似的引擎包括 Dryad、Tez。

② 数据格式和内存布局

由于 MapReduce Schema on Read 处理方式会引起较大的处理开销。Spark 抽象出分布式内存存储结构弹性分布式数据集 RDD，进行数据的存储。RDD 能支持粗粒度写操作，但对于读取操作，RDD 可以精确到每条记录，这使得 RDD 可以用来作为分布式索引。Spark 的特性是能够控制数据在不同节点上的分区，用户可以自定义分区策略，如 Hash 分区等。Shark 和 Spark SQL 在 Spark 的基础之上实现了列存储和列存储压缩。

③ 执行策略

MapReduce 在数据 Shuffle 之前花费了大量的时间来排序，Spark 则可减轻上述问题带来的开销。因为 Spark 任务在 Shuffle 中不是所有情景都需要排序，所以支持基于 Hash 的分布式聚合，调度中采用更为通用的任务执行计划图(DAG)，每一轮次的输出结果在内存缓存。

④ 任务调度的开销

传统的 MapReduce 系统，如 Hadoop，是为了运行长达数小时的批量作业而设计的，在某些极端情况下，提交一个任务的延迟非常高。Spark 采用了事件驱动的类库 AKKA 来启动任务，通过线程池复用线程来避免进程或线程启动和切换开销。

3. Spark 能带来什么

① 打造全栈多计算范式的高效数据流水线

Spark 支持复杂查询，在简单的“map”及“reduce”操作之外，Spark 还支持 SQL 查询、流式计算、机器学习和图算法。同时，用户可以在同一个工作流中无缝搭配这些计算范式。

② 轻量级快速处理

Spark 1.0 核心代码只有 4 万行。这是由于 Scala 语言的简洁和丰富的表达力，以及 Spark 充分利用和集成 Hadoop 等其他第三方组件，同时着眼于大数据处理，数据处理速度是至关重要的，Spark 通过将中间结果缓存在内存减少磁盘 I/O 来达到性能的提升。

③ 易于使用

Spark 支持多语言，Spark 支持通过 Scala、Java 及 Python 编写程序，这允许开发者在自己熟悉的语言环境下进行工作。它自带了 80 多个算子，同时允许在 Shell 中进行交互式计算。用户可以利用 Spark 像书写单机程序一样书写分布式程序，轻松利用 Spark 搭建大数据内存计算平台并充分利用内存计算，实现海量数据的实时处理。

④ 与 HDFS 等存储层兼容

Spark 可以独立运行，除了可以运行在当下的 YARN 等集群管理系统之外，它还可以读取已有的任何 Hadoop 数据。这是个非常大的优势，它可以运行在任何 Hadoop 数据源上，如 Hive、HBase、HDFS 等。这个特性让用户可以轻易迁移已有的持久化层数据。

⑤ 社区活跃度高

Spark 起源于 2009 年，当下已有超过 50 个机构、260 个工程师贡献过代码。开源系统的发展不应只看一时之快，更重要的是支持一个活跃的社区和强大的生态系统。同时我们也应该看到 Spark 并不是完美的，RDD 模型适合的是粗粒度的全局数据并行计算。不适合细粒度的、需要异步更新的计算。对于一些计算需求，如果要针对特定工作负载达到最优性能，还是需要使用一些其他的大数据系统。例如，图计算领域的 GraphLab 在特定计算负载性能上优于 GraphX，流计算中的 Storm 在实时性要求很高的场合要比 SparkStreaming 更胜一筹。

随着 Spark 发展势头日趋迅猛，它已被广泛应用于 Yahoo、Twitter、阿里巴巴、百度、网易、英特尔等各大公司的生产环境中。

1.1.2 Spark 生态系统

目前，Spark 已经发展成为包含众多子项目的大数据计算平台。伯克利将 Spark 的整个生态系统称为伯克利数据分析栈(BDAS)。其核心框架是 Spark，同时 BDAS 涵盖支持结构化数据 SQL 查询与分析的查询引擎 Spark SQL 和 Shark，提供机器学习功能的系统 MLbase 及底层的分布式机器学习库 MLlib、并行图计算框架 GraphX、流计算框架 Spark Streaming、采样近似计算查询引擎 BlinkDB、内存分布式文件系统 Tachyon、资源管理框架 Mesos 等子项目。这些子项目在 Spark 上层提供了更高层、更丰富的计算范式，图 1-1 为 BDAS 的项目结构图。

下面对 BDAS 的各个子项目进行更详细的介绍。

(1)Spark 是整个 BDAS 的核心组件，是一个大数据分布式编程框架，不仅实现了 MapReduce 的算子 map 函数和 reduce 函数及计算模型，还提供了更为丰富的算子，如 filter、join、groupByKey 等。Spark 将分布式数据抽象为弹性分布式数据集(RDD)，实现了应用任务调度、RPC、序列化和压缩，并为运行在其上的上层组件提供 API。其底层采用 Scala 这种函数式语言书写而成，并且所提供的 API 深度借鉴了 Scala 函数式的编程思想，提供与 Scala 类似的编程接口。图 1-2 为 Spark 的处理流程(主要对象为 RDD)。

Spark 将数据在分布式环境下分区，然后将作业转化为有向无环图(DAG)，并分阶段进行 DAG 的调度和任务的分布式并行处理。

(2)Shark 是构建在 Spark 和 Hive 基础之上的数据仓库。目前，Shark 已经完成学术使

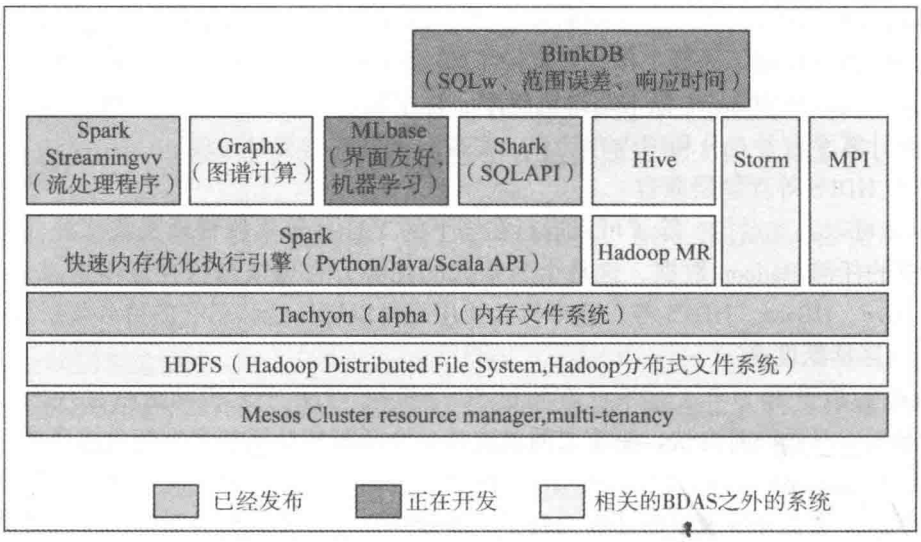

图 1-1　伯克利数据分析栈(BDAS)项目结构图

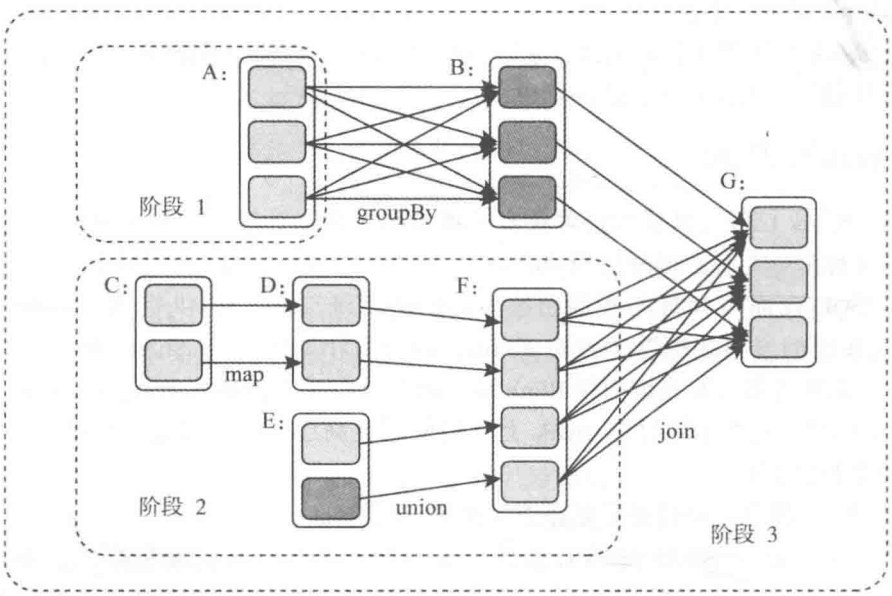

图 1-2　Spark 的任务处理流程图

命，终止开发，但其架构和原理仍具有借鉴意义。它提供了能够查询 Hive 中所存储数据的一套 SQL 接口，兼容现有的 Hive QL 语法。这样，熟悉 Hive QL 或者 SQL 的用户可以基于 Shark 进行快速的 Ad-Hoc、Reporting 等类型的 SQL 查询。Shark 底层复用 Hive 的解析器、优化器以及元数据存储和序列化接口。Shark 会将 Hive QL 编译转化为一组 Spark 任务，进行分布式运算。

(3)Spark SQL 提供在大数据上的 SQL 查询功能，类似于 Shark 在整个生态系统的角色，它们可以统称为 SQL on Spark。之前，Shark 的查询编译和优化器依赖于 Hive，使得 Shark 不得不维护一套 Hive 分支，而 Spark SQL 使用 Catalyst 做查询解析和优化器，并在底层使用 Spark 作为执行引擎实现 SQL 的 Operator。用户可以在 Spark 上直接书写 SQL，相当于为 Spark 扩充了一套 SQL 算子，这无疑更加丰富了 Spark 的算子和功能，同时 Spark SQL 不断兼容不同的持久化存储(如 HDFS、Hive 等)，为其发展奠定了广阔的空间。

(4)Spark Streaming 通过将流数据按指定时间片累积为 RDD，然后将每个 RDD 进行批处理，进而实现大规模的流数据处理。其吞吐量能够超越现有主流处理框架 Storm，并提供丰富的 API 用于流数据计算。

(5)GraphX 基于 BSP 模型，在 Spark 之上封装类似 Pregel 的接口，进行大规模同步全局的图计算，尤其是当用户进行多轮迭代时，基于 Spark 内存计算的优势尤为明显。

(6)Tachyon 是一个分布式内存文件系统，可以理解为内存中的 HDFS。为了提供更高的性能，将数据存储剥离 Java Heap。用户可以基于 Tachyon 实现 RDD 或者文件的跨应用共享，并提供高容错机制，保证数据的可靠性。

(7)Mesos 是一个资源管理框架，提供类似于 YARN 的功能。用户可以在其中插件式地运行 Spark、MapReduce、Tez 等计算框架的任务。Mesos 会对资源和任务进行隔离，并实现高效的资源任务调度。(注：Spark 自带的资源管理框架是 Standalone)

(8)BlinkDB 是一个用于在海量数据上进行交互式 SQL 的近似查询引擎。它允许用户通过在查询准确性和查询响应时间之间做出权衡，完成近似查询。其数据的精度被控制在允许的误差范围内。为了达到这个目标，BlinkDB 的核心思想是：通过一个自适应优化框架，随着时间的推移，从原始数据建立并维护一组多维样本；通过一个动态样本选择策略，选择一个适当大小的示例，然后基于查询的准确性和响应时间满足用户查询需求。

1.1.3 Spark 架构

从上文介绍可以看出，Spark 是整个 BDAS 的核心。生态系统中的各个组件通过 Spark 来实现对分布式并行任务处理的程序支持。

1. Spark 代码架构

图 1-3 展示了 Spark-1.0 的代码结构和代码量(不包含 Test 和 Sample 代码)，学生可以通过代码架构对 Spark 的整体组件有一个初步了解，正是这些代码模块构成了 Spark 架构中的各个组件，同时学生可以通过代码模块的脉络阅读与剖析源码，这对于了解 Spark 的架构和实现细节都是很有帮助的。

下面对图 1-3 中的各模块进行简要介绍：

- scheduler：文件夹中含有负责整体的 Spark 应用、任务调度的代码。
- broadcast：含有 Broadcast(广播变量)的实现代码，API 中是 Java 和 Python API 的实现。
- deploy：含有 Spark 部署与启动运行的代码。
- common：不是一个文件夹，而是代表 Spark 通用的类和逻辑实现，有 5000 行代码。
- metrics：是运行时状态监控逻辑代码，Executor 中含有 Worker 节点负责计算的逻辑代码。

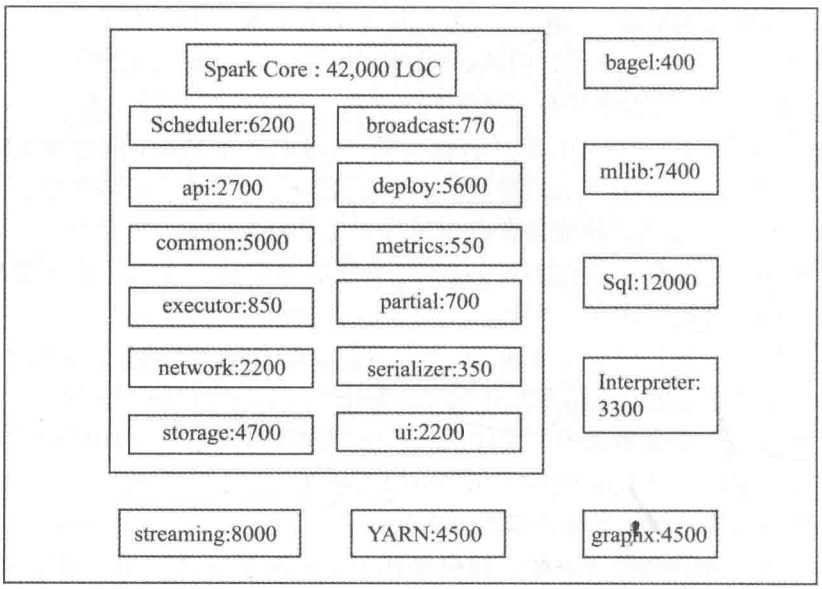

图 1-3　Spark 代码结构和代码量

- partial：含有近似评估代码。
- network：含有集群通信模块代码。
- serializer：含有序列化模块的代码。
- storage：含有存储模块的代码。
- ui：含有监控界面的代码逻辑。其他的代码模块分别是对 Spark 生态系统中其他组件的实现。
- streaming：是 Spark Streaming 的实现代码。
- YARN：是 Spark on YARN 的部分实现代码。
- graphx：含有 GraphX 实现代码。
- interpreter：代码交互式 Shell 的代码量为 3300 行。
- mllib：代表 MLlib 算法实现的代码量。
- sql：代表 Spark SQL 的代码量。

2. Spark 的架构

Spark 架构采用了分布式计算中的 Master-Slave 模型。Master 是对应集群中的含有 Master 进程的节点，Slave 是集群中含有 Worker 进程的节点。Master 作为整个集群的控制器，负责整个集群的正常运行；Worker 相当于是计算节点，接收主节点命令与进行状态汇报；Executor 负责任务的执行；Client 作为用户的客户端负责提交应用，Driver 负责控制一个应用的执行，如图 1-4 所示。

Spark 集群部署后，需要在主节点和从节点分别启动 Master 进程和 Worker 进程，对整个集群进行控制。在一个 Spark 应用的执行过程中，Driver 和 Worker 是两个重要角色。Driver 程序是应用逻辑执行的起点，负责作业的调度，即 Task 任务的分发，而多个

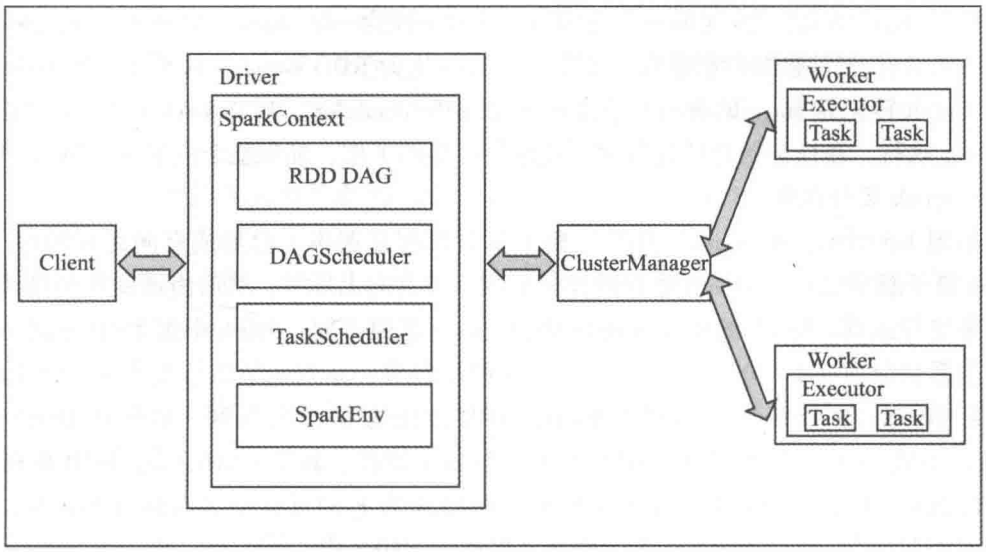

图 1-4 Spark 框架图

Worker 用来管理计算节点和创建 Executor 并行处理任务。在执行阶段，Driver 会将 Task 和 Task 所依赖的 file 和 jar 序列化后传递给对应的 Worker 机器，同时 Executor 对相应数据分区的任务进行处理。

下面详细介绍 Spark 的架构中的基本组件。

- ClusterManager：在 Standalone 模式中即为 Master(主节点)，控制整个集群，监控 Worker。在 YARN 模式中为资源管理器。
- Worker：从节点，负责控制计算节点，启动 Executor 或 Driver。在 YARN 模式中为 NodeManager，负责计算节点的控制。
- Driver：运行 Application 的 main()函数并创建 SparkContext。
- Executor：执行器，在 worker node 上执行任务的组件、用于启动线程池运行任务。每个 Application 拥有独立的一组 Executors。
- SparkContext：整个应用的上下文，控制应用的生命周期。
- RDD：Spark 的基本计算单元，一组 RDD 可形成执行的有向无环图 RDD Graph。
- DAG Scheduler：根据作业(Job)构建基于 Stage 的 DAG，并提交 Stage 给 TaskScheduler。
- TaskScheduler：将任务(Task)分发给 Executor 执行。
- SparkEnv：线程级别的上下文，存储运行时的重要组件的引用。
- SparkEnv 内创建并包含如下一些重要组件的引用。
- MapOutPutTracker：负责 Shuffle 元信息的存储。
- BroadcastManager：负责广播变量的控制与元信息的存储。
- BlockManager：负责存储管理、创建和查找块。
- MetricsSystem：监控运行时性能指标信息。

- SparkConf：负责存储配置信息。

Spark 的整体流程为：Client 提交应用，Master 找到一个 Worker 启动 Driver，Driver 向 Master 或者资源管理器申请资源，之后将应用转化为 RDD Graph，再由 DAGScheduler 将 RDD Graph 转化为 Stage 的有向无环图提交给 TaskScheduler，由 TaskScheduler 提交任务给 Executor 执行。在任务执行的过程中，其他组件协同工作，确保整个应用顺利执行。

3. Spark 运行逻辑

如图 1-5 所示，在 Spark 应用中，整个执行流程在逻辑上会形成有向无环图(DAG)。Action 算子触发之后，将所有累积的算子形成一个有向无环图，然后由调度器调度该图上的任务进行运算。Spark 的调度方式与 MapReduce 有所不同。Spark 根据 RDD 之间不同的依赖关系切分形成不同的阶段(Stage)，一个阶段包含一系列函数执行流水线。图中的 A、B、C、D、E、F 分别代表不同的 RDD，RDD 内的方框代表分区。数据从 HDFS 输入 Spark，形成 RDD A 和 RDD C，RDD C 上执行 map 操作，转换为 RDD D，RDD B 和 RDD E 执行 join 操作，转换为 F，而在 B 和 E 连接转化为 F 的过程中又会执行 Shuffle，最后 RDD F 通过函数 saveAsSequenceFile 输出并保存到 HDFS 中。

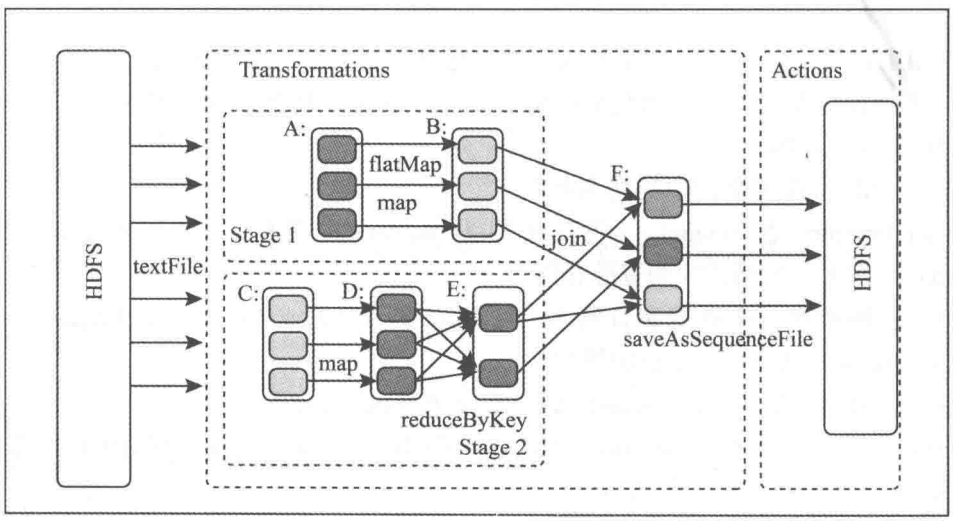

图 1-5　Spark 执行有向无环图

1.2　案例实践

Spark 的安装简便，学生可以在官网上下载到最新的软件包，网址为 http：//spark.apache.org/。Spark 最早是为了在 Linux 平台上使用而开发的，在生产环境中也是部署在 Linux 平台上，但是 Spark 在 UNIX、Windows 和 Mac OS X 系统上也运行良好。不过，在 Windows 上运行 Spark 稍显复杂，必须先安装 Cygwin 以模拟 Linux 环境，才能安装 Spark。

由于 Spark 主要使用 HDFS 充当持久化层，所以完整地使用 Spark 需要预先安装

Hadoop。下面介绍 Spark 集群的安装和部署。

Spark 在生产环境中，主要部署在安装有 Linux 系统的集群中。在 Linux 系统中安装 Spark 需要预先安装 JDK、Scala 等所需的依赖。由于 Spark 是计算框架，所以需要预先在集群内搭建好存储数据的持久化层，如 HDFS、Hive、Cassandra 等。最后用户就可以通过启动脚本运行应用了。

1. 在 Linux 集群上安装与配置 Spark

(1)安装 JDK

本例以 JDK 1.8 为例，学生可以在 Oracle JDK 官网下载相应的 JDK 版本，官网地址为：http://www.oracle.com/technetwork/java/javase/downloads/index.html。

- 解压安装包

Linux 系统下，下载 tar.gz 安装包 jdk-8u60-linux-x64.tar.gz，将其解压到自定义目录/usr/cx/下，执行如下命令：

```
tar-zxvf /usr/software/jdk-8u60-linux-x64.tar.gz-C /usr/cx
```

- 配置环境变量

```
vi ~/.bashrc JAVA_HOME=/usr/cx/jdk1.8.0_60 PATH=$JAVA_HOME/bin:$PATH
CLASSPATH=.:$JAVA_HOME/jre/lib/rt.jar:$JAVA_HOME/jre/lib/tools.jar
export JAVA_HOME PATH CLASSPATH
```

保存退出，更新环境变量，执行如下命令：

```
source ~/.bashrc
```

(2)配置 SSH 免密码登录

在集群管理和配置中有很多工具可以使用。例如，可以采用 pssh 等 Linux 工具在集群中分发与复制文件，也可以自己书写 Shell、Python 的脚本分发包。

Spark 的 Master 节点向 Worker 节点发命令需要通过 ssh 进行发送，通常情况下不希望 Master 每发送一次命令就输入一次密码，因此需要实现 Master 无密码登录到所有 Worker。

Master 作为客户端，要实现无密码公钥认证，连接到服务端 Worker。需要在 Master 上生成一个密钥对，包括一个公钥和一个私钥，然后将公钥复制到 Worker 上。当 Master 通过 ssh 连接 Woker 时，Worker 就会生成一个随机数并用 Master 的公钥对随机数进行加密，发送给 Master。Master 收到加密数之后再用私钥进行解密，并将解密数回传给 Worker，Worker 确认解密数无误之后，允许 Master 进行连接。这就是一个公钥认证过程，其间不需要手工输入密码，主要过程是将 Master 节点公钥复制到 Worker 节点上。下面介绍如何配置 Master 与 Worker 之间的 SSH 免密码登录。