前　言

近年来，大数据凭借其数据量大、数据类型多样、产生与处理速度快、价值高的“4V”特性成为学术界和工业界的研究热点。由于传统软件难以在可接受的时间范围内处理大数据，所以学术界和工业界研发了许多分布式的大数据系统来解决大规模数据的存储、处理、分析和挖掘等问题。

关于 Apach Spark

2003—2006 年，Google 在计算机系统领域会议 SOSP/OSDI 上发表了 *Google File System*、*MapReduce*、*BigTable* 3 篇重量级的系统论文，分别讨论了大规模数据如何存储、处理及结构化组织。之后，Apache Hadoop 社区对这些论文进行了开源实现，开发了 HDFS 分布式文件系统、Hadoop MapReduce 大数据处理框架和 HBase 分布式 Key-Value 数据库，大大降低了大数据应用开发、运行及数据存储管理的难度。这些系统被广泛应用于互联网、电信、金融、科研等领域，以进行大规模数据存储与处理。

对于大数据处理框架来说，MapReduce 成功地将函数式编程思想引入分布式数据处理中，仅仅用两个函数（map() 和 reduce()）就解决了一大类的大数据处理问题，而且不要求用户熟悉分布式文件系统。然而，随着大数据应用越来越多，处理性能的要求越来越高，MapReduce 的一些缺点也显现出来。例如，MapReduce 编程模型的表达能力较弱，仅使用 map() 和 reduce() 两个函数难以实现复杂的数据操作；处理流程固定，不容易实现迭代计算；基于磁盘进行数据传递，效率较低。这些缺点导致使用 MapReduce 开发效率较低、执行复杂的数据处理任务的性能也不高。

为了解决这些问题，微软在 2008—2009 年研发了 Dryad/DryadLINQ，其中 Dryad 类似 MapReduce，但以有向无环图（Directed Acycline Graph，DAG）形式的数据流取代了 MapReduce 固定的 map-reduce 两阶段数据流，处理流程更通用，支持复杂的数据处理任

务。DryadLINQ 为 Dryad 提供了高层编程语言，将更多的函数式编程思想（来源于 C# 的 LINQ）引入编程模型中，表达能力更强，如容易实现 join() 等操作。然而，由于 Dryad/DryadLINQ 在当时没有开源，所以没有得到大规模使用。

鉴于 MapReduce、Dryad 等框架存在一些问题，2012 年，UC Berkeley 的 AMPLab 研发并开源了新的大数据处理框架 Spark。其核心思想包括两方面：一方面对大数据处理框架的输入 / 输出、中间数据进行建模，将这些数据抽象为统一的数据结构，命名为弹性分布式数据集（Resilient Distributed Dataset，RDD），并在此数据结构上构建了一系列通用的数据操作，使得用户可以简单地实现复杂的数据处理流程；另一方面采用基于内存的数据聚合、数据缓存等机制来加速应用执行，尤其适用于迭代和交互式应用。Spark 采用 EPFL 大学研发的函数式编程语言 Scala 实现，并且提供了 Scala、Java、Python、R 四种语言的接口，以方便开发者使用熟悉的语言进行大数据应用开发。

经过多年的发展，Spark 也与 Hadoop 一样构建了完整的生态系统。Apache Spark 生态系统以 Spark 处理框架为核心，在上层构建了面向 SQL 语言的 Spark SQL 框架、面向大规模图计算的 GraphX 框架、面向大规模机器学习的 MLlib 框架及算法库，以及面向流处理的 Spark Streaming 框架；在下层，Spark 及其关联社区也推出了相关存储系统，如基于内存的分布式文件系统 Alluxio、支持 ACID 事务的数据湖系统 Delta Lake 等。由于整个 Spark 生态系统中包含的系统和框架众多，本书主要关注生态系统中核心的 Spark 处理框架本身。下面介绍本书的一些基本信息。

本书的写作目的及面向的读者

本书的写作目的是以 Spark 框架为核心，总结大数据处理框架的设计和实现原理，包括框架设计中面临的基本问题、可能的解决方案、Spark 采用的方案、具体实现方法，以及优缺点等。与机器学习等领域有成熟的理论模型不同，大数据处理框架并没有一个完整的理论模型，但是具有一些系统设计模型及设计方案，如编程模型、逻辑处理流程、物理执行计划及并行化方案等。这些模型及方案是研究人员和工程师在实践中经过不断探索和改进得到的实践经验。本书的目的是将这些宝贵经验抽象总结为系统设计模型和方案，帮助读者从理论层和实现层深入理解大数据处理框架，具体体现如下。

（1）帮助读者理解 Spark 应用，以及与 Spark 关联的上下层框架。Spark 等大数据处理框架只将数据接口和操作接口暴露给用户，用户一般不了解应用程序是如何被转化为可

分布执行任务的，以及任务是如何执行的。本书详细总结了 Spark 框架将应用程序转化为逻辑处理流程，并进一步转化为物理执行计划的一般过程。理解这个过程将帮助读者理解更复杂的 Spark 应用，如在第 5 章中介绍的迭代型机器学习应用和图计算应用。同时，理解 Spark 的设计与实现原理也有助于理解 Spark 生态圈中的上下层框架之间的关系，如 Spark SQL、MLlib、GraphX 是如何利用 Spark 框架来执行 SQL、机器学习和图计算任务的。

（2）帮助读者开发性能更好、可靠性更高的大数据应用。用户在运行大数据应用时，经常面临应用执行效率低下、无响应、I/O 异常、内存溢出等性能和可靠性问题。本书讲述了与 Spark 框架性能和可靠性相关的 Shuffle 机制、数据缓存机制、错误容忍机制、内存管理机制等。理解 Spark 应用性能和可靠性的影响因素，帮助读者在运行 Spark 应用时进行参数调优，有助于更好地利用数据缓存来提升应用性能，同时合理利用数据持久化机制来减少执行错误、内存溢出等可靠性问题。

（3）帮助读者对 Spark 等大数据框架进行进一步优化和改进。在实际使用大数据处理框架的过程中，开发者经常会因为软硬件环境、数据特征、应用逻辑、执行性能的特殊需求，需要对 Spark 框架的部分功能进行优化改进。例如，在内存较小的集群上运行需要对内存管理进行改进，经常出现网络阻塞时需要对 Shuffle 机制进行改进，针对一些特殊应用需要开发新的数据操作等。要对 Spark 框架进行改进，不仅需要非常了解 Spark 框架的设计和实现原理，还需要清楚改进后可能出现的正确性和可靠性问题。本书对 Spark 设计和实现过程中的问题挑战、解决方案、优缺点的阐述将帮助开发者和研究人员对框架进行优化改进。

因此，我们相信，本书对于大数据处理框架的用户、开发者和研究人员都会有一定的帮助。

本书的主要内容

本书主要介绍 Apache Spark 大数据处理框架的设计和实现原理，附带讨论与 Hadoop MapReduce 等框架的优缺点对比。全书分 9 章，主要包含以下四部分内容。

第一部分　大数据处理框架的基础知识。第 1 章介绍大数据处理框架的基本概念、系统架构，以及与其相关的研究工作。第 2 章概览 Spark 的系统架构、编程模型，并以一个典型的 Spark 应用为例概述 Spark 应用的执行流程。Spark 初学者可以直接从第 2 章开始阅读。

第二部分　Spark 大数据处理框架的核心理论。该部分包括两章，主要介绍 Spark 如何将应用程序转化为可以分布执行的计算任务。其中，第 3 章介绍 Spark 将应用程序转化为逻辑处理流程的一般过程及方法，并对常用的数据操作进行详细分析。第 4 章讨论 Spark 将逻辑处理流程转化为物理执行计划的一般过程及方法，以及常用的数据操作形成的计算任务。

第三部分　典型的 Spark 应用。第 5 章介绍复杂的迭代型 Spark 应用，包括迭代型机器学习应用和图计算应用，以及这些应用的并行化方法。这些应用也进一步验证了 Spark 框架的通用性。

第四部分　大数据处理框架性能和可靠性保障机制。该部分主要探究 Spark 框架性能和可靠性相关的技术，其中包括第 6 章的 Shuffle 机制、第 7 章的数据缓存机制、第 8 章的错误容忍机制及第 9 章的内存管理机制。

本书特点

本书注重设计原理和实现方案的阐述，写作方式考虑了技术深度、研究价值、易读性等因素，具体特点如下。

（1）采用问题驱动的阐述方式。本书在章节开头或者中间引入 Spark 设计和实现过程中面临的挑战性问题（Problems），然后将这些问题拆分为子问题逐步深入讨论可能的解决方案（Potential Solutions），最后介绍为什么 Spark 会使用当前的解决方案（Why）及具体是如何实现的（How）。这种表述方式可以让读者“知其然”，并且“知其所以然”，也就是让读者既从使用者角度又从设计者角度来理解大数据处理框架中的基本问题、基本设计方法及实现思想。

（2）强调基本原理的阐述。本书着重介绍 Spark 设计和实现的基本原理，不具体展示代码实现的细节。这样做的第一个原因是，其基本原理是对代码的高层抽象，对大数据框架的用户、开发者和研究人员来说，原理更重要、更容易理解。第二个原因是，代码在不断优化更新，而基本原理比较稳定。本书也可以看作从 Spark 代码中抽象出概要设计和详细设计。

（3）图文并茂，容易理解。本书将复杂的设计和实现逻辑抽象为图例，并通过文字描述和实例来方便读者理解复杂的设计方案和执行过程。对于一些复杂的问题，本书还进行

了分类讨论，使技术内容更具有逻辑性、更容易被理解。

（4）具有一定的学术研究价值。本书讨论了 Spark 设计与实现过程中存在的一些挑战性问题及当前解决方案的不足，同时也探讨了一些相关的研究工作，并在必要时与 Hadoop 进行了对比。这些讨论将有助于读者研究和优化改进大数据处理框架。

Spark 版本与 API 问题

本书以 Spark 2.4.3 版和 RDD API 为基础进行编写。实际上，从 Spark 2.0 版到未来的 Spark 3.0 版，Spark 社区推荐用户使用 DataSet、DataFrame 等面向结构化数据的高层 API（Structured API）来替代底层的 RDD API，因为这些高层 API 包含更多的数据类型信息（Schema），支持 SQL 操作，并且可以利用经过高度优化的 Spark SQL 引擎来执行。然而，由于以下几个原因，本书使用 RDD API 来探讨 Spark 的设计原理和实现。

（1）RDD API 更基础，更适合分析大数据处理框架中的一些基本问题和原理。相比 DataSet、DataFrame 数据结构，RDD API 的数据结构更简单和通用，更适合用来展示一些基本概念和原理，如数据分区方法、每个数据操作的执行过程（生成什么样的 RDD API、建立什么样的数据依赖关系），以及执行阶段划分方法、任务划分方法等。而且，RDD API 包含更多的数据操作类型，可以用来展示更丰富的数据依赖关系、逻辑处理流程等。

（2）学习 RDD API 及其执行原理，帮助读者理解高层 API 和 Spark SQL 执行引擎。由于 Spark SQL 执行引擎采用了许多数据库、分布式系统、编译等领域的优化技术，其将应用程序（使用 DataSet/DataFrame API 编写）转化为物理执行计划的过程比较复杂。本书讲解了“**RDD API 应用程序—逻辑处理流程—物理执行计划**”的基本转化过程，帮助读者进一步学习和理解更复杂的 Spark SQL 转化过程。读者可以参阅 Bill Chambers 和 Matei Zaharia 合著的 *Spark*: *The Definitive Guide*，学习 DataSet、DataFrame API 的具体使用方法，也可以参阅朱锋、张韶全、黄明合著的《Spark SQL 内核剖析》，进一步学习 Spark SQL 引擎的执行过程和具体实现。

（3）上层框架（如 MLlib、GraphX）中有很多代码使用 RDD API，学习原生的 RDD API 有助于在分布式层面理解数据和计算的抽象表达，也有助于理解图计算应用和机器学习应用中的数据分区（如边划分、点划分），以及计算任务划分方法（如数据并行等）。

另外，在未来的 Spark 3.0 版本中还会有一些新的特性，如支持 GPU 调度、SQL 自适

应查询执行等。如果有机会，我们会在本书的下一版中探讨更多关于 Spark SQL 执行引擎的设计原理，以及这些新特性。

与 SparkInternals 技术文档的关系

有些读者可能看过我们在 2015 年撰写并在 GitHub 上公开的 SparkInternals 技术文档。该文档介绍了 Spark 1.0 与 Spark 2.0 版本的设计原理和实现，迄今已收到 4000 多颗 stars、1500 次 forks，也被翻译为英文、日文和泰文，受到很多大数据工程师和研究人员的关注。

不过，SparkInternals 技术文档中总结的设计原理不够完整和深入，而且后几章中涉及的实现细节也较多（包含一些实现细节代码）。由于这本书的目的是总结设计原理和实现，所以并没有使用太多 SparkInternals 技术文档中的内容，而是按照“**问题－解决方案－ Spark 采用的方案－实现－优缺点**”的逻辑重新撰写了相关章节，只是使用了 SparkInternals 技术文档中的部分图例。当然，如果读者想要了解一些实现细节和代码，也可将 SparkInternals 技术文档作为本书的补充资料。

我们在 GitHub 上建立了一个名为 ApacheSparkBook 的公开项目，将本书设计的示例代码和高清图片放到了项目中（项目地址为 https://github.com/JerryLead/ApacheSparkBook）。读者可以在项目中进行提问，方便我们解答问题或勘误。

由于作者水平和经验有限，本书的错漏和不足之处欢迎广大读者朋友批评指正，可以将意见提交到 GitHub 项目中或者通过电子邮件（csxulijie@gmail.com）联系作者。

在本书写作过程中，作者得到了所在单位（中国科学院软件研究所）诸多老师的关注和支持，在此感谢魏峻、武延军、王伟等老师提供的科研工作环境。作者的研究工作受到国家自然科学基金（61802377），以及中国科学院青年创新促进会的项目支持。

感谢参与本书初稿讨论和审阅的各位朋友，包括亚马逊的纪树平博士、腾讯的朱锋博士、阿里巴巴的沈雯婷、中国科学院软件研究所的李慧、王栋、叶星彤、康锴等同学，以及 Databricks 的 Xiao Li 博士及其团队。同时，也感谢广大读者对 GitHub 上 SparkInternals 技术文档的支持和反馈意见。

电子工业出版社的孙学瑛编辑及其团队在本书的审校、排版、出版发行过程中付出了巨大努力，在此表示由衷感谢！

最后，感谢家人及朋友对作者一如既往的支持。