Python大数据处理库

PySpark实战

汪 明 著

清華大學出版社
北 京

内 容 简 介

我国提出新基建概念，要加快大数据中心、人工智能等新型基础设施的建设进度，这无疑需要更多的大数据人才。PySpark 可以对大数据进行分布式处理，降低大数据学习门槛，本书正是一本 PySpark 入门教材，适合有一定 Python 基础的读者学习使用。

本书分为 7 章，第 1 章介绍大数据的基本概念、常用的大数据分析工具；第 2 章介绍 Spark 作为大数据处理的特点和算法；第 3 章介绍 Spark 实战环境的搭建，涉及 Windows 和 Linux 操作系统；第 4 章介绍如何灵活应用 PySpark 对数据进行操作；第 5 章介绍 PySpark ETL 处理，涉及 PySpark 读取数据、对数据进行统计分析等数据处理相关内容；第 6 章介绍 PySpark 如何利用 MLlib 库进行分布式机器学习（Titanic 幸存者预测）；第 7 章介绍一个 PySpark 和 Kafka 结合的实时项目。

本书内容全面、示例丰富，可作为广大 PySpark 入门读者必备的参考书，同时能作为大中专院校师生的教学参考书，也可作为高等院校计算机及相关专业的大数据技术教材使用。

图书在版编目（CIP）数据

Python 大数据处理库 PySpark 实战 / 汪明著.—北京：清华大学出版社，2021.3
ISBN 978-7-302-57508-5

Ⅰ. ①P… Ⅱ. ①汪… Ⅲ. ①数据处理 Ⅳ.①TP274

中国版本图书馆 CIP 数据核字（2021）第 026655 号

责任编辑：夏毓彦
封面设计：王　翔
责任校对：闫秀华
责任印制：杨　艳

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：三河市龙大印装有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张：**20　　**字　　数：**512 千字
版　　次：2021 年 3 月第 1 版　　**印　　次：**2021 年 3 月第 1 次印刷
定　　价：79.00 元

产品编号：086669-01

前　言

PySpark 是 Apache Spark 为 Python 开发人员提供的编程 API 接口，以便开发人员用 Python 语言对大数据进行分布式处理，可降低大数据处理的门槛。

PySpark 优势有哪些？首先 PySpark 是基于 Python 语言的，简单易学。其次，PySpark 可以非常方便地对大数据进行处理，其中可用 SQL 方便地从 Hadoop、Hive 及其他文件系统中读取数据并进行统计分析。最后，PySpark 编写的大数据处理程序，容易维护，且部署方便。

PySpark 可以从多种数据源中读取数据，并可以对数据进行统计分析和处理，其中包括批处理、流处理、图计算和机器学习模型构建等。它还可以将数据处理的结果持久化到多种文件系统中，为大数据 UI 展现提供数据支持。PySpark 比 Java/Scala 更容易学习，借助 IDE 开发工具，可以非常方便地进行代码编写和调试。

如果你对大数据处理有一定兴趣，了解基本的编程知识，立志构建大数据处理的相关应用，那么本书将适合你。本书作为 PySpark 的入门教材，由浅入深地对 PySpark 大数据处理方法进行介绍，特别对常用的操作、ETL 处理和机器学习进行详细的说明，最后结合实战项目将各个知识点有机整合，做到理论联系实际。

本书特点

（1）理论联系实际，先从大数据基本概念出发，然后对 Hadoop 生态、Spark 架构和部署方式等知识点进行讲解，并结合代码进行阐述，最后通过一个实战项目来说明如何从头到尾搭建一个实时的大数据处理演示程序。

（2）深入浅出、轻松易学，以实例为主线，激发读者的阅读兴趣，让读者能够真正学习到 PySpark 最实用、最前沿的技术。

（3）技术新颖、与时俱进，结合时下最热门的技术，如 Spark、Python 和机器学习等，让读者在学习 PySpark 的同时，熟悉更多相关的先进技术。

（4）贴心提醒，本书根据需要在各章使用了很多“注意”小栏目，让读者可以在学习过程中更轻松地理解相关知识点及概念。

源码下载

本书配套的源码，请用微信扫描右边二维码获取（可以击页面上的“推送到我的邮箱”，填入自己的邮箱，到邮箱中下载）。如果阅读中存在疑问，请联系 booksaga@163.com，邮件主题为“Python 大数据处理库 PySpark 实战”。

本书运行环境说明

本书使用的系统为 Windows 7 宿主操作系统上安装 VMware Workstation 15.5，再安装 CentOS 7。PySpark 运行环境搭建在 CentOS 7 上。读者学习本书需要有 CentOS 7 系统管理的基础知识。

本书读者

- 有一定Python编程基础的初学者
- 大数据处理与分析人员
- 从事后端开发，对大数据开发有兴趣的人员
- 想用Python构建大数据处理应用的人员
- 想要掌握大数据处理技术的高等院校师生
- 大数据技术培训学校的师生

本书作者

汪明，硕士，毕业于中国矿业大学，徐州软件协会副理事长，某创业公司合伙人。从事软件行业十余年，发表论文数十篇。著有图书《TypeScript 实战》《Go 并发编程实战》。

著者
2021 年 1 月

目　　录

第 1 章　大数据时代 1

1.1　什么是大数据 1
1.1.1　大数据的特点 2
1.1.2　大数据的发展趋势 3
1.2　大数据下的分析工具 4
1.2.1　Hadoop 5
1.2.2　Hive 6
1.2.3　HBase 6
1.2.4　Apache Phoenix 7
1.2.5　Apache Drill 7
1.2.6　Apache Hudi 7
1.2.7　Apache Kylin 8
1.2.8　Apache Presto 8
1.2.9　ClickHouse 8
1.2.10　Apache Spark 9
1.2.11　Apache Flink 10
1.2.12　Apache Storm 10
1.2.13　Apache Druid 10
1.2.14　Apache Kafka 11
1.2.15　TensorFlow 11
1.2.16　PyTorch 12
1.2.17　Apache Superset 12
1.2.18　Elasticsearch 12
1.2.19　Jupyter Notebook 13
1.2.20　Apache Zeppelin 13
1.3　小结 14

第 2 章　大数据的瑞士军刀——Spark 15

2.1　Hadoop 与生态系统 15
2.1.1　Hadoop 概述 15

2.1.2 HDFS 体系结构 19
2.1.3 Hadoop 生态系统 20
2.2 Spark 与 Hadoop 23
2.2.1 Apache Spark 概述 23
2.2.2 Spark 和 Hadoop 比较 24
2.3 Spark 核心概念 25
2.3.1 Spark 软件栈 25
2.3.2 Spark 运行架构 26
2.3.3 Spark 部署模式 27
2.4 Spark 基本操作 29
2.5 SQL in Spark 33
2.6 Spark 与机器学习 33
2.6.1 决策树算法 35
2.6.2 贝叶斯算法 36
2.6.3 支持向量机算法 36
2.6.4 随机森林算法 37
2.6.5 人工神经网络算法 38
2.6.6 关联规则算法 39
2.6.7 线性回归算法 40
2.6.8 KNN 算法 40
2.6.9 K-Means 算法 41
2.7 小结 42

第 3 章 Spark 实战环境设定 43

3.1 建立 Spark 环境前提 43
3.1.1 CentOS 7 安装 45
3.1.2 FinalShell 安装 55
3.1.3 PuTTY 安装 58
3.1.4 JDK 安装 60
3.1.5 Python 安装 63
3.1.6 Visual Studio Code 安装 64
3.1.7 PyCharm 安装 65
3.2 一分钟建立 Spark 环境 66
3.2.1 Linux 搭建 Spark 环境 66
3.2.2 Windows 搭建 Spark 环境 69
3.3 建立 Hadoop 集群 79
3.3.1 CentOS 配置 79
3.3.2 Hadoop 伪分布模式安装 81

3.3.3 Hadoop 完全分布模式安装 87
3.4 安装与配置 Spark 集群 93
3.5 安装与配置 Hive 99
3.5.1 Hive 安装 99
3.5.2 Hive 与 Spark 集成 108
3.6 打造交互式 Spark 环境 110
3.6.1 Spark Shell 111
3.6.2 PySpark 112
3.6.3 Jupyter Notebook 安装 112
3.7 小结 118

第 4 章 活用 PySpark 119

4.1 Python 语法复习 119
4.1.1 Python 基础语法 120
4.1.2 Python 变量类型 124
4.1.3 Python 运算符 135
4.1.4 Python 控制语句 139
4.1.5 Python 函数 143
4.1.6 Python 模块和包 149
4.1.7 Python 面向对象 154
4.1.8 Python 异常处理 157
4.1.9 Python JSON 处理 159
4.1.10 Python 日期处理 160
4.2 用 PySpark 建立第一个 Spark RDD 161
4.2.1 PySpark Shell 建立 RDD 163
4.2.2 VSCode 编程建立 RDD 165
4.2.3 Jupyter 编程建立 RDD 167
4.3 RDD 的操作与观察 168
4.3.1 first 操作 169
4.3.2 max 操作 169
4.3.3 sum 操作 170
4.3.4 take 操作 171
4.3.5 top 操作 172
4.3.6 count 操作 172
4.3.7 collect 操作 173
4.3.8 collectAsMap 操作 174
4.3.9 countByKey 操作 175
4.3.10 countByValue 操作 175

4.3.11 glom 操作 176
4.3.12 coalesce 操作 177
4.3.13 combineByKey 操作 178
4.3.14 distinct 操作 179
4.3.15 filter 操作 180
4.3.16 flatMap 操作 181
4.3.17 flatMapValues 操作 181
4.3.18 fold 操作 182
4.3.19 foldByKey 操作 183
4.3.20 foreach 操作 184
4.3.21 foreachPartition 操作 185
4.3.22 map 操作 186
4.3.23 mapPartitions 操作 187
4.3.24 mapPartitionsWithIndex 操作 187
4.3.25 mapValues 操作 188
4.3.26 groupBy 操作 189
4.3.27 groupByKey 操作 190
4.3.28 keyBy 操作 191
4.3.29 keys 操作 192
4.3.30 zip 操作 193
4.3.31 zipWithIndex 操作 194
4.3.32 values 操作 194
4.3.33 union 操作 195
4.3.34 takeOrdered 操作 196
4.3.35 takeSample 操作 197
4.3.36 subtract 操作 198
4.3.37 subtractByKey 操作 198
4.3.38 stats 操作 199
4.3.39 sortBy 操作 200
4.3.40 sortByKey 操作 201
4.3.41 sample 操作 202
4.3.42 repartition 操作 203
4.3.43 reduce 操作 204
4.3.44 reduceByKey 操作 205
4.3.45 randomSplit 操作 206
4.3.46 lookup 操作 207
4.3.47 join 操作 208
4.3.48 intersection 操作 209
4.3.49 fullOuterJoin 操作 210

4.3.50 leftOuterJoin 与 rightOuterJoin 操作 211
4.3.51 aggregate 操作 212
4.3.52 aggregateByKey 操作 215
4.3.53 cartesian 操作 217
4.3.54 cache 操作 218
4.3.55 saveAsTextFile 操作 218
4.4 共享变数 220
4.4.1 广播变量 220
4.4.2 累加器 221
4.5 DataFrames 与 Spark SQL 223
4.5.1 DataFrame 建立 223
4.5.2 Spark SQL 基本用法 228
4.5.3 DataFrame 基本操作 231
4.6 撰写第一个 Spark 程序 245
4.7 提交你的 Spark 程序 246
4.8 小结 248

第 5 章 PySpark ETL 实战 249

5.1 认识资料单元格式 249
5.2 观察资料 255
5.3 选择、筛选与聚合 267
5.4 存储数据 269
5.5 Spark 存储数据到 SQL Server 272
5.6 小结 275

第 6 章 PySpark 分布式机器学习 276

6.1 认识数据格式 277
6.2 描述统计 280
6.3 资料清理与变形 284
6.4 认识 Pipeline 288
6.5 逻辑回归原理与应用 290
6.5.1 逻辑回归基本原理 290
6.5.2 逻辑回归应用示例：Titanic 幸存者预测 291
6.6 决策树原理与应用 295
6.6.1 决策树基本原理 295
6.6.2 决策树应用示例：Titanic 幸存者预测 296
6.7 小结 299

第 7 章 实战：PySpark+Kafka 实时项目 301

7.1 Kafka 和 Flask 环境搭建 301
7.2 代码实现 303
7.3 小结 310

第 1 章

大数据时代

当前人类已经步入大数据时代，大数据已经逐步渗透到我们的日常生活中，涉及各行各业，包括零售行业、物流行业、教育行业、金融行业、交通行业和制造行业等。一般来说，大数据的基本任务有数据的存储、计算、查询分析和挖掘，而这些任务往往需要多台计算机共同调度才能完成。

大数据时代，数据已经变成了一种生产资料，其价值也提升到新的高度。随着各种行业的数据化，使得数据逐步形成数据资产，利用大数据技术可以更好地让数据资产价值化。当前越来越多的企业管理决策都转变成以数据为驱动的大数据辅助决策。本章作为开篇，主要介绍大数据的基本概念，重点介绍大数据的常见分析工具。

本章主要涉及的知识点有：

- 大数据概述：介绍大数据相关的概念和特征。
- 大数据分析工具：介绍当前大数据生态系统下的常见分析工具。

1.1 什么是大数据

根据百度百科显示，最早提出大数据时代到来的是全球知名咨询公司麦肯锡，麦肯锡称：“数据，已经渗透到当今每一个行业和业务职能领域，成为重要的生产因素。人们对于海量数据的挖掘和运用，预示着新一波生产率增长和消费者盈余浪潮的到来。”

大数据（Big Data），是一个 IT 行业术语，是指无法在一定时间范围内用单机软件工具进行捕捉、管理和处理的数据集合，它需要使用分布式模式才能处理，其数据具有数量大、来源广、价值密度低等特征。

至于什么数据量算得上大数据，这个也没有一定的标准，一般来说，单机难以处理的数

据量，就可以称得上大数据。

大数据和人工智能往往关系密切，人工智能算法必须依据数据才能构建合适的模型，以便用于预测和智能决策。当前，大数据技术已经在医药、电信、金融、安全监管、环保等领域广泛使用。

大数据时代，分布式的数据存储和查询模式可以对全量数据进行处理。举例来说，以前 DNA 和指纹数据库的建立，由于信息技术水平的限制，只能重点采集并存储部分人口的 DNA 和指纹数据，这种限制对于很多案件的侦破是非常不利的。

而当我们步入大数据时代后，从理论上来讲，采集并存储全球人口的DNA和指纹信息是可行的。因此，建立全量的 DNA 和指纹数据库，这对 DNA 和指纹数据的比对工作来说，具有非常大的价值。

以前我们研究问题，主要研究几个要素之间的因果关系，例如通过经验、观察实验和数学等理论推导出一些公式，用于指导生产和生活。而在大数据时代，更多的是对几个要素之间相关性进行分析。例如，通过对电商平台上的购买行为进行分析，可以对用户进行画像，并根据用户的历史购买记录，来智能推荐他可能感兴趣的商品，这种分析对提升成单率来说至关重要。

基于大数据的推荐系统，可能比你自己都要了解你自己。这也是在大数据时代人类越来越关心个人隐私信息的安全问题的原因。

相关性分析是寻找因果关系的利器。可以说，相关分析和因果分析是互相促进的。如果多个因素之间有明显的相关性，那么就可以进一步研究其因果关系。

大数据的价值就在于从海量数据中，通过机器学习算法自动搜寻多个因素之间的相关性，这些相关性可以大大减少人工搜寻的时间。换句话说，人工从海量数据中往往很难发现多个因素之间的相关性，而这恰恰是机器学习比较擅长的领域。

1.1.1 大数据的特点

一般来说，大数据具有如下几个特点。

1．Volume（大量）

大数据场景下，对数据的采集、计算和存储所涉及的数量是非常庞大的，数据量往往多到单台计算机无法处理和存储，必须借助多台计算机构建的集群来分布式处理和存储。

分布式存储要保证数据存储的安全性。如果某一个节点上的数据损坏，那么必须从其他节点上对损坏节点上的数据进行自动修复，这个过程中就需要数据的副本，同一份数据会复制多份，并分布式存储到不同的节点上。

如果不借助大数据工具，自己实现一个分布式文件系统，那么其工作量非常大。因此，对于大数据的处理和存储来说，更好的方案就是选择一款开源的分布式文件系统。

2．Velocity（高速）

以前由于数据采集手段落后、数据存储空间横向扩展困难，不能存储海量的数据，因此只会采集一些重要的数据，如财务数据、生产数据等。这就导致了高层管理人员在决策时，

缺乏完整、统一的宏观数据作为数据支撑。

在大数据时代，由于数据采集手段多样、数据可以分布式存储，因此当前很多企业都会尽可能地存储数据，其中不少企业中都有传感器或者视频探头，它们会产生大量的数据，形成一个数据流，这些数据流的产生都是非常迅速的，因此分析这些数据的软件系统必须做到高效地采集、处理和存储这些高速生成的数据。

一般来说，大数据系统可以借助分布式集群构建的强大计算力，对海量数据进行快速处理。若处理数据的响应时间能到秒级，甚至毫秒级，那么其价值将非常大。实时大数据的处理，这也是目前众多大数据工具追求的一个重要能力。

3．Variety（多样）

生物具有多样性，动物有哺乳动物、鸟类和冷血动物等，植物有苔藓植物、蕨类植物和种子植物等。多样的生物只有和谐相处，才是可持续发展之道。

同样地，数据的载体也是多种多样，一般来说，可以分为结构化数据、非结构化数据和半结构化数据。其中很多业务数据都属于结构化数据，而是视频、音频和图像等都可划分为非结构化数据。在大数据时代下，非结构化数据从数量上来说占了大部分。因此，对视频、音频、图像和自然语言等非结构化数据的处理，也是当前大数据工具要攻克的重点。

4．Value（低价值密度）

大数据首先是数据量庞大，一般来说，都是PB级别的。但在特定场景下，真正有用的数据可能较少，即数据价值密度相对较低。从大数据中挖掘出有用的价值，如大海捞针一般。

举例来说，交通部门为了更好地对道路交通安全进行监管，在重点的路口都设有违法抓拍系统，会对每辆车进行拍照，这个数据量非常巨大，其中有交通违法行为的车辆照片并不多，可以说是万里挑一。因此这个价值密度相对低，但是存储这些数据非常重要，其中某一些图片资料对于协助破案来说会起到至关重要的作用。

5．Veracity（真实性）

大数据场景下，由于数据来源的多样性，互相可以验证，因此数据的真实性往往比较高。这里说的真实性，是指数据的准确性和及时性。数据的真实性也是大数据可以形成数据资产的一个重要前提，只有真实、可信的数据才能挖掘出有用的价值。

大数据由于具有如上的特点，这就对大数据的信息化软件提出了非常高的要求。一般的软件系统是无法很好的处理大数据的。从技术上看，大数据与云计算密不可分。大数据无法用单台计算机进行存储和处理，而必须采用分布式架构，即必须依托云计算提供的分布式存储和计算能力。

1.1.2　大数据的发展趋势

大数据目前有如下几点发展趋势。

1. 大数据是一种生产资料

目前人类已经步入数字经济时代，大数据是非常重要的一种生产资料，与土地、石油等

资源作为重要的生产资料类似，数字经济时代以大数据作为最基础也是最重要的生产资料。

在大数据时代，信息的载体是数据。对于数据的分析与挖掘来说，其实质是生产各类信息产品，这些信息产品可以看作是一种数字商品，是可以产生实际价值的资产。若将大数据比作土地，那么基于大数据分析和挖掘出的信息产品，就好比在土地上种植出来的各种农产品。

2. 与物联网和5G的融合

大数据的基础是数据，而产生数据的源头更多是来自物联网和 5G。物联网、移动互联网和 5G 等新兴技术，将进一步助力大数据的发展，让大数据为企业管理决策和政府决策提供更大的价值。特别是 5G 技术的推广，将进一步提升大数据的应用。

3. 大数据理论的突破

随着 5G 的发展，大数据很可能爆发新一轮的技术革命。人类处理信息往往借助视频、图像和声音（语言），因此大数据技术目前正在与机器学习、人工智能等相关技术进行深度结合，在视频、图像和语音的处理上，必须在理论上继续突破，才可能实现科学技术上的突破。视频中的行为检测、图像物体识别和语音识别等应用会产生极大的经济效益和社会效益。

4. 数据公开和标准化

数据作为一种重要的资产，只有流动起来才能更好地发挥价值。就像河里的水一样，只有流到田间地头对庄家进行灌溉，才能生产出农产品。数据在流转的过程中，数据的标准化非常重要，这样才能打破信息孤岛，从而更好地让数据产生价值。

5. 数据安全

大数据中涉及各类数据，其中难免有敏感的数据，数据在流转过程中，如何对敏感数据进行加密和脱敏，这将至关重要。因此，大数据应用必须充分考虑数据安全的问题。

1.2 大数据下的分析工具

大数据技术首先需要解决的问题是如何高效、安全地存储；其次是如何高效、及时地处理海量的数据，并返回有价值的信息；最后是如何通过机器学习算法，从海量数据中挖掘出潜在的价值，并构建模型，以用于预测预警。

可以说，当今大数据的基石，来源于谷歌公司的三篇论文，这三篇论文主要阐述了谷歌公司对于大数据问题的解决方案。这三篇论文分别是：

- *Google File System*
- *Google MapReduce*
- *Google BigTable*

其中，*Google File System* 主要解决大数据分布式存储的问题，*Google MapReduce* 主要解

决大数据分布式计算的问题，*Google Bigtable* 主要解决大数据分布式查询的问题。

当前大数据工具的蓬勃发展，或多或少都受到上述论文的启发，不少社区根据论文的相关原理，实现了最早一批的开源大数据工具，如 Hadoop 和 HBase。

但值得注意的是，当前每个大数据工具都专注于解决大数据领域的特定问题，很少有一种大数据工具可以一站式解决所有的大数据问题。因此，一般来说，大数据应用需要多种大数据工具相互配合，才能解决大数据相关的业务问题。

大数据工具非常多，据不完全统计，大约有一百多种，但常用的只有 10 多种，这些大数据工具重点解决的大数据领域各不相同，现列举如下：

- 分布式存储：主要包含Hadoop HDFS和Kafka等。
- 分布式计算：包括批处理和流计算，主要包含Hadoop MapReduce、Spark和Flink等。
- 分布式查询：主要包括Hive、HBase、Kylin、Impala等。
- 分布式挖掘：主要包括Spark ML和Alink等。

据中国信通院企业采购大数据软件调研报告来看，86.6%的企业选择基于开源软件构建自己的大数据处理业务，因此学习和掌握几种常用的开源大数据工具，对于大数据开发人员来说至关重要。下面重点介绍大数据常用的几种分析工具。

1.2.1　Hadoop

Hadoop 是一个由 Apache 基金会开发的分布式系统基础架构，源于论文 *Google File System*。Hadoop 工具可以让用户在不了解分布式底层细节的情况下，开发分布式程序，从而大大降低大数据程序的开发难度。它可以充分利用计算机集群构建的大容量、高计算能力来对大数据进行存储和运算。

在大数据刚兴起之时，Hadoop 可能是最早的大数据工具，它也是早期大数据技术的代名词。时至今日，虽然大数据工具种类繁多，但是不少工具的底层分布式文件系统还是基于 Hadoop 的 HDFS（Hadoop Distributed File System）。

Hadoop 框架前期最核心的组件有两个，即 HDFS 和 MapReduce。后期又加入了 YARN 组件，用于资源调度。其中 HDFS 为海量的数据提供了存储，而 MapReduce 则为海量的数据提供了分布式计算能力。

HDFS 有高容错性的特点，且支持在低廉的硬件上进行部署，而且 Hadoop 访问数据的时候，具有很高的吞吐量，适合那些有着超大数据集的应用程序。

可以说，Hadoop 工具是专为离线和大规模数据分析而设计的，但它并不适合对几个记录随机读写的在线事务处理模式。Hadoop 软件是免费开源的，官网地址为 http://hadoop.apache.org。

注　意

目前来说，Hadoop 工具不支持数据的部分 update 操作，因此不能像关系型数据库那样，可以用 SQL 来更新部分数据。

1.2.2 Hive

对于数据的查询和操作，一般开发人员熟悉的是 SQL 语句，但是 Hadoop 不支持 SQL 对数据的操作，而是需要用 API 来进行操作，这个对于很多开发人员来说并不友好。因此，很多开发人员期盼能用 SQL 语句来查询 Hadoop 中的分布式数据。

Hive 工具基于 Hadoop 组件，可以看作是一个数据仓库分析系统，Hive 提供了丰富的 SQL 查询方式来分析存储在 Hadoop 分布式文件系统中的数据。

Hive 可以将结构化的数据文件映射为一张数据表，这样就可以利用 SQL 来查询数据。本质上，Hive 是一个翻译器，可以将 SQL 语句翻译为 MapReduce 任务运行。

Hive SQL 使不熟悉 MapReduce 的开发人员可以很方便地利用 SQL 语言进行数据的查询、汇总和统计分析。但是 Hive SQL 与关系型数据库的 SQL 略有不同，虽然它能支持绝大多数的语句，如 DDL、DML 以及常见的聚合函数、连接查询和条件查询等。

Hive 还支持 UDF（User-Defined Function，用户定义函数），也可以实现对 map 和 reduce 函数的定制，为数据操作提供了良好的伸缩性和可扩展性。Hive 不适合用于联机事务处理，也不适合实时查询功能。它最适合应用在基于大量不可变数据的批处理作业。Hive 的特点包括：可伸缩、可扩展、容错、输入格式的松散耦合。Hive 最佳使用场合是大数据集的批处理作业，例如，网络日志分析。官网地址为 http://hive.apache.org。

注 意

目前来说，Hive 中的 SQL 支持度有限，只支持部分常用的 SQL 语句，且不适合 update 操作去更新部分数据，即不适合基于行级的数据更新操作。

1.2.3 HBase

HBase 工具是一个分布式的、面向列的开源数据库，该技术来源于 *Google BigTable* 的论文。它在 Hadoop 之上提供了类似于 Google BigTable 的能力。HBase 不同于一般的关系数据库，它是一个适合存储非结构化数据的数据库，且采用了基于列而不是基于行的数据存储模式。

HBase 在很多大型互联网公司得到应用，如阿里、京东、小米和 Facebook 等。Facebook 用 HBase 存储在线消息，每天数据量近百亿；小米公司的米聊历史数据和消息推送等多个重要应用系统都建立在 HBase 之上。

HBase 适用场景有：

- 密集型写应用：写入量巨大，而相对读数量较小的应用，比如消息系统的历史消息，游戏的日志等。
- 查询逻辑简单的应用：HBase只支持基于rowkey的查询，而像SQL中的join等查询语句，它并不支持。
- 对性能和可靠性要求非常高的应用：由于HBase本身没有单点故障，可用性非常高。它支持在线扩展节点，即使应用系统的数据在一段时间内呈井喷式增长，也可以通过横向扩展来满足功能要求。

HBase 读取速度快得益于内部使用了 LSM 树型结构，而不是 B 或 B+树。一般来说，磁盘的顺序读取速度很快，但相对而言，寻找磁道的速度就要慢很多。HBase 的存储结构决定了读取任意数量的记录不会引发额外的寻道开销。官网地址为 http://hbase.apache.org。

注　意
目前来说，HBase 不能基于 SQL 来查询数据，需要使用 API。

1.2.4　Apache Phoenix

前面提到，Hive 是构建在 Hadoop 之上，可以用 SQL 对 Hadoop 中的数据进行查询和统计分析。同样地，HBase 原生也不支持用 SQL 进行数据查询，因此使用起来不方便，比较费力。

Apache Phoenix 是构建在 HBase 数据库之上的一个 SQL 翻译层。它本身用 Java 语言开发，可作为 HBase 内嵌的 JDBC 驱动。Apache Phoenix 引擎会将 SQL 语句翻译为一个或多个 HBase 扫描任务，并编排执行以生成标准的 JDBC 结果集。

Apache Phoenix 提供了用 SQL 对 HBase 数据库进行查询操作的能力，并支持标准 SQL 中大部分特性，其中包括条件运算、分组、分页等语法，因此降低了开发人员操作 HBase 当中的数据的难度，提高了开发效率。官网地址为 http://phoenix.apache.org。

1.2.5　Apache Drill

Apache Drill 是一个开源的、低延迟的分布式海量数据查询引擎，使用 ANSI SQL 兼容语法，支持本地文件、HDFS、HBase、MongoDB 等后端存储，支持 Parquet、JSON 和 CSV 等数据格式。本质上 Apache Drill 是一个分布式的大规模并行处理查询层。

它是 Google Dremel 工具的开源实现，而 Dremel 是 Google 的交互式数据分析系统，性能非常强悍，可以处理 PB 级别的数据。

Google 开发的 Dremel 工具，在处理 PB 级数据时，可将处理时间缩短到秒级，从而作为 MapReduce 的有力补充。它作为 Google BigQuery 的报表引擎，获得了很大的成功。Apache Drill 的官网地址为 https://drill.apache.org。

1.2.6　Apache Hudi

Apache Hudi 代表 Hadoop Upserts and Incrementals，由 Uber 开发并开源。它基于 HDFS 数据存储系统之上，提供了两种流原语：插入更新和增量拉取。它可以很好地弥补 Hadoop 和 Hive 对部分数据更新的不足。

Apache Hudi 工具提供两个核心功能：首先支持行级别的数据更新，这样可以迅速地更新历史数据；其次是仅对增量数据的查询。Apache Hudi 提供了对 Hive、Presto 和 Spark 的支持，可以直接使用这些组件对 Hudi 管理的数据进行查询。

Hudi 的主要目的是高效减少数据摄取过程中的延迟。HDFS 上的分析数据集通过两种类型的表提供服务：

- 读优化表（Read Optimized Table）：通过列式存储提高查询性能。
- 近实时表（Near-Real-Time Table）：基于行的存储和列式存储的组合提供近实时查询。

Hudi 的官网地址为 https://hudi.apache.org。

1.2.7 Apache Kylin

Apache Kylin 是数据平台上的一个开源 OLAP 引擎。它采用多维立方体预计算技术，可以将某些场景下的大数据 SQL 查询速度提升到亚秒级别。值得一提的是，Apache Kylin 是国人主导的第一个 Apache 顶级开源项目，在开源社区有较大的影响力。

它的查询速度如此之快，是基于预先计算尽量多的聚合结果，在查询时应该尽量利用预先计算的结果得出查询结果，从而避免直接扫描超大的原始记录。

使用 Apache Kylin 主要分为三步：

（1）首先，定义数据集上的一个星形或雪花形模型。
（2）其次，在定义的数据表上构建多维立方。
（3）最后，使用 SQL 进行查询。

Apache Kylin 的官网地址为 http://kylin.apache.org。

1.2.8 Apache Presto

Apache Presto 是一个开源的分布式 SQL 查询引擎，适用于交互式分析查询，数据量支持 GB 到 PB 字节。它的设计和编写完全是为了解决像 Facebook 这样规模的商业数据仓库的交互式分析和处理速度的问题。

Presto 支持多种数据存储系统，包括 Hive、Cassandra 和各类关系数据库。Presto 查询可以将多个数据源的数据进行合并，可以跨越整个组织进行分析。

国内的京东和国外的 Facebook 都使用 Presto 进行交互式查询。Apache Presto 的官网地址为 https://prestodb.io。

1.2.9 ClickHouse

Yandex 在 2016 年 6 月 15 日开源了一个用于数据分析的数据库，名字叫作 ClickHouse，这个列式存储数据库的性能要超过很多流行的商业 MPP 数据库软件，例如 Vertica。目前国内社区火热，各个大厂纷纷使用。

今日头条用 ClickHouse 做用户行为分析，内部一共几千个 ClickHouse 节点，单集群最大 1200 节点，总数据量几十 PB，日增原始数据 300TB 左右。腾讯用 ClickHouse 做游戏数据分析，并且为之建立了一整套监控运维体系。

携程从 2018 年 7 月份开始接入试用，目前 80%的业务都跑在 ClickHouse 上。每天数据增量十多亿，近百万次查询请求。快手也在使用 ClickHouse，每天新增 200TB，90%查询低于 3 秒。

Clickhouse 的具体特点如下；

- 面向列的DBMS。
- 数据高效压缩。
- 磁盘存储的数据。
- 多核并行处理。
- 在多个服务器上分布式处理。
- SQL语法支持。
- 向量化引擎。
- 实时数据更新。
- 索引支持。
- 适合在线查询。
- 支持近似预估计算。
- 支持嵌套的数据结构。
- 数据复制和对数据完整性的支持。

ClickHouse 的官网地址为 https://clickhouse.tech。

1.2.10　Apache Spark

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。它是加州大学伯克利分校的 AMP 实验室开源的通用并行框架。它不同于 Hadoop MapReduce，计算任务中间输出结果可以保存在内存中，而不再需要读写 HDFS，因此 Spark 计算速度更快，也能更好地适用于机器学习等需要迭代的算法。

Apache Spark 是由 Scala 语言开发的，可以与 Java 程序一起使用。它能够像操作本地集合对象一样轻松地操作分布式数据集。它具有运行速度快、易用性好、通用性强和随处运行等特点。

Apache Spark 提供了 Java、Scala、Python 以及 R 语言的 API。还支持更高级的工具，如 Spark SQL、Spark Streaming、Spark MLlib 和 Spark GraphX 等。

Apache Spark 主要有如下几个特点：

- 非常快的计算速度：它主要在内存中计算，因此在需要反复迭代的算法上，优势非常明显，比 Hadoop快100倍。
- 易用性：它大概提供了80多个高级运算符，包括各种转换、聚合等操作。这相对于Hadoop组件中提供的map和reduce两大类操作来说，丰富了很多，因此可以更好地适应复杂数据的逻辑处理。
- 通用性：它除了自身不带数据存储外，其他大数据常见的业务需求，比如批处理、流计算、图计算和机器学习等都有对应的组件。因此，开发者通过Spark提供的各类组件，如Spark SQL、Spark Streaming、Spark MLlib和Spark GraphX等，可以在同一个应用程序中无缝组合使用这些库。
- 支持多种资源管理器：它支持Hadoop YARN、Apache Mesos，以及Spark自带的Standalone集群管理器。

Spark 的官网地址为 http://spark.apache.org。

1.2.11 Apache Flink

Apache Flink 是一个计算框架和分布式处理引擎，用于对无界和有界数据流进行有状态计算，该框架完全由 Java 语言开发，也是国内阿里巴巴主推的一款大数据工具。其针对数据流的分布式计算提供了数据分布、数据通信以及容错机制等功能。基于流执行引擎，Flink 提供了诸多更高抽象层的 API，以便用户编写分布式任务。

- DataSet API：对静态数据进行批处理操作，将静态数据抽象成分布式的数据集，用户可以方便地使用Flink提供的各种操作符，对分布式数据集进行处理。
- DataStream API：对数据流进行流处理操作，将流式的数据抽象成分布式的数据流，用户可以方便地对分布式数据流进行各种操作。
- Table API：对结构化数据进行查询操作，将结构化数据抽象成关系表，并通过类SQL的语句对关系表进行各种查询操作。

Apache Flink 的官网地址为 http://flink.apache.org。

1.2.12 Apache Storm

Apache Storm 是开源的分布式实时计算系统，擅长处理海量数据，适用于数据实时处理而非批处理。Hadoop 或者 Hive 是大数据中进行批处理使用较为广泛的工具，这也是 Hadoop 或者 Hive 的强项。但是 Hadoop MapReduce 并不擅长实时计算，这也是业界一致的共识。

当前很多业务对于实时计算的需求越来越强烈，这也是 Storm 推出的一个重要原因。Apache Storm 的官网地址为 http://storm.apache.org。

注 意

随着 Spark 和 Flink 对于流数据的处理能力的增强，目前不少实时大数据处理分析都从 Storm 迁移到 Spark 和 Flink 上，从而降低了维护成本。

1.2.13 Apache Druid

Apache Druid 是一个分布式的、支持实时多维 OLAP 分析的数据处理系统。它既支持高速的数据实时摄入处理，也支持实时且灵活的多维数据分析查询。因此它最常用的大数据场景就是灵活快速的多维 OLAP 分析。

另外，它支持根据时间戳对数据进行预聚合摄入和聚合分析，因此也经常用于对时序数据进行处理分析。

Apache Druid 的主要特性如下：

- 亚秒响应的交互式查询，支持较高并发。
- 支持实时导入，导入即可被查询，支持高并发导入。
- 采用分布式shared-nothing的架构，可以扩展到PB级。

- 数据查询支持SQL。

Apache Druid 的官网地址为 http://druid.apache.org。

注　意

目前 Apache Druid 不支持精确去重，不支持 Join 和根据主键进行单条记录更新。同时，需要与阿里开源的 Druid 数据库连接池区别开来。

1.2.14　Apache Kafka

Apache Kafka 是一个开源流处理平台，它的目标是为处理实时数据提供一个统一、高通量、低等待的平台。Apache Kafka 最初由 LinkedIn 开发，并于 2011 年初开源。2012 年 10 月从 Apache Incubator 毕业。在非常多的实时大数据项目中，都能见到 Apache Kafka 的身影。

Apache Kafka 不仅仅是一个消息系统，主要用于如下场景：

- 发布和订阅：类似一个消息系统，可以读写流式数据。
- 流数据处理：编写可扩展的流处理应用程序，可用于实时事件响应的场景。
- 数据存储：安全地将流式的数据存储在一个分布式、有副本备份且容错的集群。

它的一个重要特点是可以作为连接各个子系统的数据管道，从而构建实时的数据管道和流式应用。它支持水平扩展，且具有高可用、速度快的优点，已经运行在成千上万家公司的生产环境中。Apache Kafka 的官网地址为 http://kafka.apache.org。

注　意

某种程度上来说，很多实时大数据系统已经离不开 Kafka，它充当一个内存数据库，可以快速地读写流数据。

1.2.15　TensorFlow

TensorFlow 最初由谷歌公司开发，用于机器学习和深度神经网络方面的研究，它是一个端到端开源机器学习平台。它拥有一个全面而灵活的生态系统，包含各种工具、库和社区资源，可助力研究人员推动先进的机器学习技术的发展，并使开发者能够轻松地构建和部署由机器学习提供支持的应用。

TensorFlow 的特征如下：

- 轻松地构建模型：它可以使用API轻松地构建和训练机器学习模型，这使得我们能够快速迭代模型并轻松地调试模型。
- 随时随地进行可靠的机器学习生产：它可以在CPU和GPU上运行，可以运行在台式机、服务器和手机移动端等设备上。无论使用哪种语言，都可以在云端、本地、浏览器中或设备上轻松地训练和部署模型。
- 强大的研究实验：现在科学家可以用它尝试新的算法，产品团队则用它来训练和使用计算模型，并直接提供给在线用户。

TensorFlow 的官网地址为 https://tensorflow.google.cn。

1.2.16 PyTorch

PyTorch 的前身是 Torch，其底层和 Torch 框架一样，但是使用 Python 重新写了很多内容，不仅更加灵活，支持动态图，而且还提供了 Python 接口。

它是一个以 Python 优先的深度学习框架，能够实现强大的 GPU 加速，同时还支持动态神经网络，这是很多主流深度学习框架比如 TensorFlow 等都不支持的。

PyTorch 是相当简洁且高效的框架，从操作上来说，非常符合我们的使用习惯，这能让用户尽可能地专注于实现自己的想法，而不是算法本身。PyTorch 是基于 Python 的，因此入门也更加简单。PyTorch 的官网地址为 https://pytorch.org。

1.2.17 Apache Superset

前面介绍的大数据工具，主要涉及大数据的存储、计算和查询，也涉及大数据的机器学习。但是这些数据的查询和挖掘结果如何直观地通过图表展现到 UI 上，以辅助更多的业务人员进行决策使用，这也是一个非常重要的课题。

没有可视化工具，再好的数据分析也不完美，可以说，数据可视化是大数据的最后一公里，因此至关重要。

Apache Superset 是由 Airbnb 开源的数据可视化工具，目前属于 Apache 孵化器项目，主要用于数据可视化工作。分析人员可以不用直接写 SQL 语句，而是通过选择指标、分组条件和过滤条件，即可绘制图表，这无疑降低了它的使用难度。它在可视化方面做得很出色，是开源领域中的佼佼者，即使与很多商用的数据分析工具相比，也毫不逊色。

Apache Superset 的官网地址为 http://superset.apache.org。

注　意
与 Apache Superset 类似的还有开源的 Metabase，官网地址为 https://www.metabase.com。

1.2.18 Elasticsearch

Elasticsearch 是一个开源的、分布式的、提供 Restful API 的搜索和数据分析引擎，它的底层是开源库 Apache Lucene。它使用 Java 编写，内部采用 Lucene 做索引与搜索。它的目标是使全文检索变得更加简单。

Elasticsearch 具有如下特征：

- 一个分布式的实时文档存储。
- 一个分布式实时分析搜索引擎。
- 能横向扩展，支持PB级别的数据。

由于 Elasticsearch 的功能强大和使用简单，维基百科、卫报、Stack Overflow、GitHub 等都纷纷采用它来做搜索。现在，Elasticsearch 已成为全文搜索领域的主流软件之一。Elasticsearch 的官网地址为 https://www.elastic.co/cn/。

注　意

Elasticsearch 中涉及的数据可以用 Kibana 实现数据可视化分析。

1.2.19　Jupyter Notebook

Jupyter Notebook 是一个功能非常强大的 Web 工具，不仅仅用于大数据分析。它的前身是 IPython Notebook，是一个基于 Web 的交互式笔记本，支持 40 多种编程语言。

它的本质是一个 Web 应用程序，便于创建和共享程序文档，支持实时代码、数学方程、可视化和 Markdown。它主要用于数据清理和转换、数值模拟、统计建模、机器学习等。Jupyter Notebook 的官网地址为 https://jupyter.org，官网界面如图 1.1 所示。

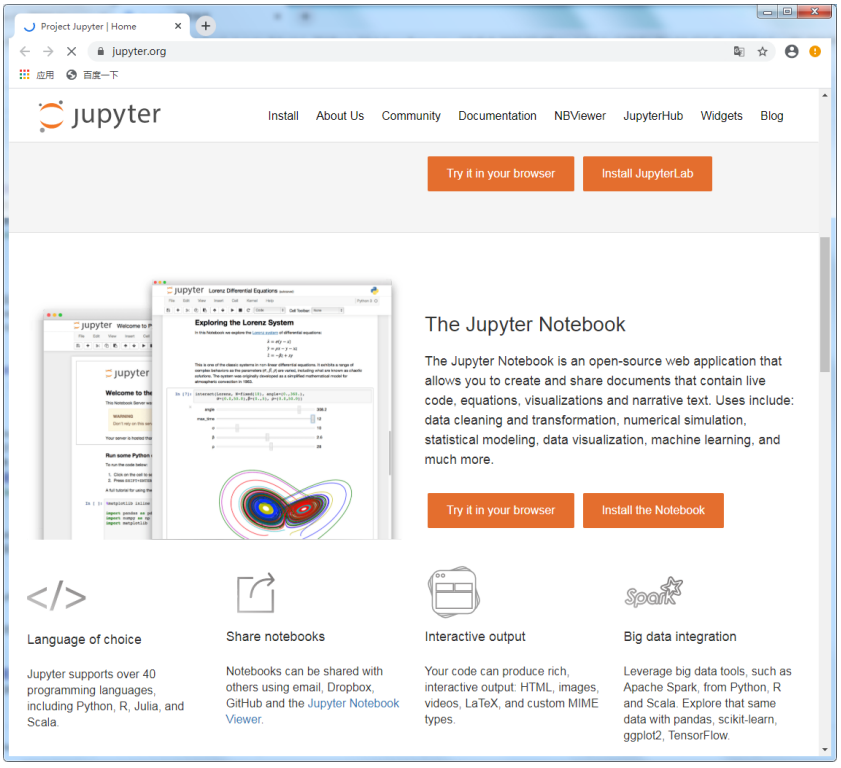

图 1.1　Jupyter Notebook 官网界面

1.2.20　Apache Zeppelin

Apache Zeppelin 和 Jupyter Notebook 类似，是一个提供交互式数据分析且基于 Web 的笔记本。它基于 Web 的开源框架，使交互式数据分析变得可行的。Zeppelin 提供了数据分析、数据可视化等功能。

借助 Apache Zeppelin，开发人员可以构建数据驱动的、可交互且可协作的精美的在线文档，并且支持多种语言，包括 Apache Spark、PySpark、Spark SQL、Hive、Markdown、Shell 等。Apache Zeppelin 主要功能有：

- 数据提取。
- 数据挖掘。
- 数据分析。
- 数据可视化展示。

Apache Zeppelin 的官网地址为 http://zeppelin.apache.org。

大数据工具发展非常快，有很多工具可能都没有听说过。就当前的行情来看，Apache Spark 基本上已经成为批处理领域的佼佼者，且最新版本在流处理上也做得不错，是实现流批一体化数据处理很好的框架。

需要大数据分析与处理的公司，应该根据自身的人员技能结构和业务需求，选择合适的大数据工具。虽然大数据工具非常之多，但是 Hadoop、HBase、Kafka、Spark 或 Flink 几乎是必备的大数据工具。

1.3 小结

本章主要介绍了大数据的相关概念和工具。可以说大数据技术已经深入到人类日常生活的方方面面，特别是交通、金融、电信、教育和安全监管等多个领域。

大数据工具繁多，当前并没有一个一站式的大数据解决方案，不同大数据工具需要有机配合才能构建出一个完整的大数据应用。毫无疑问，虽然大数据工具如此之多，但是常用的大数据工具中，Hadoop、HBase、Kafka、Spark 或 Flink 是不可或缺的工具。

第 2 章

大数据的瑞士军刀——Spark

上一章主要介绍了大数据的相关概念，以及大数据和人类日常生活的关系，并介绍了几种大数据分析工具。本章将重点介绍被誉为大数据处理的瑞士军刀——Spark。

本章主要涉及的知识点有：

- Hadoop及其生态系统：了解Hadoop的由来以及Hadoop生态系统。
- Spark的核心概念：掌握Spark的基本概念和架构。
- Spark基本操作：了解Spark的几种常见操作。
- SQL in Spark概述：了解Spark相关数据统计可以用SQL来操作。
- Spark与机器学习：了解Spark MLlib库中的几种机器学习算法。

2.1 Hadoop 与生态系统

Hadoop 不是一个简单的工具，它有自己的生态体系，这就是本节将要介绍的内容。

2.1.1 Hadoop 概述

如果读者之前听说过大数据，那么几乎都应该听过 Hadoop 这个词。在大数据领域，Hadoop 可以说是一个绕不开的话题。那么什么是 Hadoop 呢？

Hadoop 是一个开源的大数据软件框架，主要用于分布式数据存储和大数据集处理。Hadoop 在大数据领域使用广泛，其中一个重要的原因是开源，这就意味着使用 Hadoop 的成本很低，软件本身是免费的。另一方面，还可以研究其内部的实现原理，并根据自身的业务需求，进行代码层面的定制。

2003 年，谷歌公司推出了 Nutch 项目，用来处理数十亿次的网页搜索和用于数百万网页的索引。2003 年 10 月，谷歌公司公开发表了 GFS（Google File System）论文。2004 年，谷歌公司又公开发表了关于 MapReduce 细节的论文。2005 年，Nutch 项目使用 GFS 和 MapReduce 来执行操作，性能飙升。

2006 年，计算机科学家 Doug Cutting 和 Mike Cafarella 共同开发出 Hadoop。2006 年 2 月，Doug Cutting 为了让 Hadoop 有更好的发展，加入雅虎。此时 Hadoop 在雅虎公司经过专业团队的打造，已经变成一个可在 Web 规模上运行的企业级大数据系统。2007 年，雅虎公司开始在 100 个节点的集群上使用 Hadoop。

到 2008 年 1 月，Hadoop 成为 Apache 顶级项目，并迎来了它的快速发展期，这无疑也证实了 Hadoop 的价值。此时，除了雅虎公司，世界上还有许多其他公司也使用 Hadoop 来构建大数据系统，如《纽约时报》和 Facebook。2008 年 4 月，Hadoop 打破了一项世界纪录，成为对 1T 数据进行排序的最快系统。这个排序系统运行在 910 个节点的集群上，在 209 秒内完成对 1TB 的数据的排序。此消息也让 Hadoop 名声大噪。

2011 年 12 月，Apache Hadoop 发布了 1.0 版本。2013 年 8 月，Apache Hadoop 发布了 2.0.6 版本，其中 Hadoop2.x 对 Hadoop1.x 重构较多。2017 年 6 月，Apache Hadoop3.0.0-alpha4 版本发布。截至到本书撰写时，最新的 Hadoop 版本为 3.2.1。

根据官方的介绍，Apache Hadoop 3.2.0 是 Hadoop 3.x 系列中最大的一个版本，带来了许多新功能和 1000 多个修改，通过 Hadoop 3.0.0 的云连接器的增强功能进一步丰富了平台，并服务于深度学习用例和长期运行的应用。

目前 Apache Hadoop 已经部署在很多大型公司的生产环境中，例如 Adobe、AWS、Apple、Cloudera、eBay、Facebook、Google、Hortonworks、IBM、Intel、LinkedIn、Microsoft、Netflix 和 Teradata 等企业。此外，Apache Hadoop 还促进了 Spark、HBase 和 Hive 等相关项目的发展。

经过大量企业的生产环境验证，Hadoop 可以在具有数千个节点的分布式系统上稳定运行。它的分布式文件系统不但提供了节点间进行数据快速传输的能力，还允许系统在个别节点出现故障时，保证整个系统可以继续运行。

注 意

一般来说，在非高可用架构下，如果 Hadoop 集群中的 NameNode 节点出现故障，那么整个 Hadoop 系统将无法提供服务。

一般来说，Hadoop 的定义有狭义和广义之分。从狭义上来说，Hadoop 就是单独指代 Hadoop 这个软件。而从广义上来说，Hadoop 指代大数据的一个生态圈，包括很多其他的大数据软件，比如 HBase、Hive、Spark、Sqoop 和 Flume 等。

一般我们所说的 Hadoop，指的是 Hadoop 这个软件，即狭义的概念。当提到 Hadoop 生态系统或者生态圈的时候，往往指的是广义的 Hadoop 概念。

注 意

目前而言，Hadoop 主要有三个发行版本：Apache Hadoop、Cloudera 版本（简称 CDH）和 Hortonworks 版本（简称 HDP）。

前面提到，Hadoop 当前最新版本为 3.2.1，其中 Hadoop2.x 和 Hadoop1.x 差异比较大，而 Hadoop3.x 和 Hadoop2.x 从架构上区别并不大，更多是性能的提升以及整合了许多重要的增强功能。

Hadoop1.x 作为第一个大版本，起初的设计并不完善，因此 Hadoop1.x 中的 HDFS 和 MapReduce 在高可用、扩展性等方面存在不足。如 NameNode 存在单点故障，且内存受限，影响扩展性，难以应用于在线场景。

另外，Hadoop1.x 难以支持除 MapReduce 之外的计算框架，如 Spark 等。Hadoop2.x 和 Hadoop1.x 在架构上的区别如图 2.1 所示。

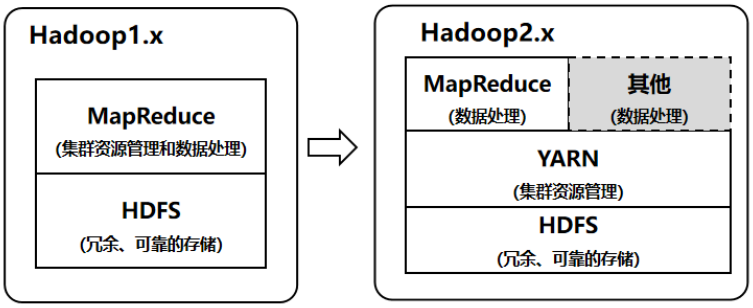

图 2.1　Hadoop2.x 和 Hadoop1.x 架构区别图

从图 2.1 中可以看出，Hadoop1.x 主要由 HDFS（Hadoop Distributed File System）和 MapReduce 两个组件组成，其中 MapReduce 组件除了负责数据处理外，还负责集群的资源管理。而 Hadoop2.x 由 HDFS、MapReduce 和 YARN 三个组件组成，MapReduce 只负责数据处理，且运行在 YARN 之上，YARN 负责集群资源调度。这样单独分离出来的 YARN 组件还可以作为其他数据处理框架的集群资源管理。

Hadoop2.x 的主要组件说明如下：

- HDFS：分布式文件系统，提供对应用程序数据的高吞吐量、高伸缩性、高容错性的访问。它是Hadoop体系中数据存储管理的基础，也是一个高度容错的系统，能检测和应对硬件故障，可在低配置的硬件上运行。
- YARN：用于任务调度和集群资源管理。
- MapReduce：基于YARN的大型数据集并行处理系统，是一种分布式计算模型，用于进行大数据量的分布式计算。

相对于之前的主要发布版本 Hadoop2.x，Apache Hadoop3.x 整合许多重要的增强功能。Hadoop3.x 是一个可用版本，提供了稳定性和高质量的 API，可以用于实际的产品开发。下面介绍 Hadoop3.x 的主要变化：

- 最低Java版本变为JDK 1.8：所有Hadoop的jar都是基于JDK 1.8进行编译的。
- HDFS支持纠删码：纠删码（erasure coding）是一种比副本存储更节省存储空间的数据持久化存储方法。
- YARN时间线服务增强：提高时间线服务的可扩展性、可靠性。

- 重写Shell脚本：修补了许多长期存在的bug，并增加了一些新的特性。
- 覆盖客户端的jar：将Hadoop的依赖隔离在单一jar包中，从而达到避免依赖渗透到应用程序的类路径中的问题，避免包冲突。
- MapReduce任务级本地优化：添加了映射输出收集器的本地化实现的支持，可以带来30%的性能提升。
- 支持2个以上的NameNode：通过多个NameNode来提供更高的容错性。
- 数据节点内置平衡器。
- YARN增强：YARN资源模型已经被一般化，可以支持用户自定义的可计算资源类型，而不仅仅是CPU和内存。

前面对 Hadoop 进行了简要的介绍，下面将介绍为什么要学习 Hadoop，或者说 Hadoop 能解决大数据什么问题。

- 大数据存储

首先，大数据要解决的问题是如何方便地存取海量的数据，而 Hadoop 的 HDFS 组件可以解决这个问题。HDFS 以分布式方式存储数据，并将每个文件存储为块（block）。块是文件系统中最小的数据单元。

假设有一个 512MB 大小的文件需要存储，由于 HDFS 默认创建数据块大小是 128MB，因此 HDFS 将文件数据分成 4 个块（512/128=4），并将其存储在不同的数据节点上。同时为了保证可靠性，还会复制块数据到不同数据节点上。因此，Hadoop 拆分数据的模式，可以胜任大数据的分布式存储。

- 可扩展性

其次，大数据需要解决的问题就是需要进行资源扩展，比如通过增加服务器节点来提升存储空间和计算资源。Hadoop 采取主从架构，支持横向扩展的方式来扩展资源，当存储空间或者计算资源不够的情况下，用户可以向 HDFS 集群中添加额外的数据节点（服务器）来解决。

- 存储各种数据

再次，大数据需要解决存储各种数据的问题，大数据系统中很大一部分都是非结构化数据，结构化的数据可能只占很少的比例。HDFS 可以存储所有类型的数据，包括结构化、半结构化和非结构化数据。Hadoop 适合一次写入、多次读取的业务场景。

- 数据处理问题

大数据还有一个重要的问题就是如何对数据进行分布式计算。一般来说，传统的应用程序都是拉取数据，应用程序是固定的，将需要处理的数据从存储的地方移动到计算程序所在的计算机上，即移动数据。

大数据计算往往需要处理的数据量非常大，而程序一般比较小，因此在这种情况下，更好的解决办法是移动程序到各个数据节点上。Hadoop 就是将计算移到数据节点上，而不是将数据移到计算程序所在的节点上。

Hadoop 的主要优势如下：

- 可扩展性：通过添加数据节点，可以扩展系统以处理更多数据。
- 灵活性：可以存储任意多的数据，且数据支持各种类型。
- 低成本：开源免费，且可以运行在低廉的硬件上。
- 容错机制：如果某个数据节点宕机，则作业将自动重定向到其他节点。
- 计算能力：数据节点越多，处理数据的能力就越强。

Hadoop 的缺点如下：

- 安全问题：Hadoop数据并没有加密，因此如果数据需要在互联网上传输，则存在数据泄漏的风险。
- 小文件问题：Hadoop缺乏有效支持随机读取小文件的能力，因此不适合小文件的存储。

注　意
在生产环境中，Hadoop 在选择版本时，应该优先选择最新的稳定版本。

2.1.2　HDFS 体系结构

HDFS 支持跨多台服务器进行数据存储，且数据会自动复制到不同的数据节点上，以防止数据丢失。HDFS 采用主从（Master/Slave）架构。

一般来说，一个 HDFS 集群是由一个 NameNode 节点（主节点）和多个 DataNode 节点（从节点）组成。其中，NameNode 是一个中心服务器，负责管理文件系统的各种元数据（Metadata）和客户端对文件的访问。DataNode 在集群中一般负责管理节点上的数据读取和计算。Hadoop 的架构设计图如图 2.2 所示。

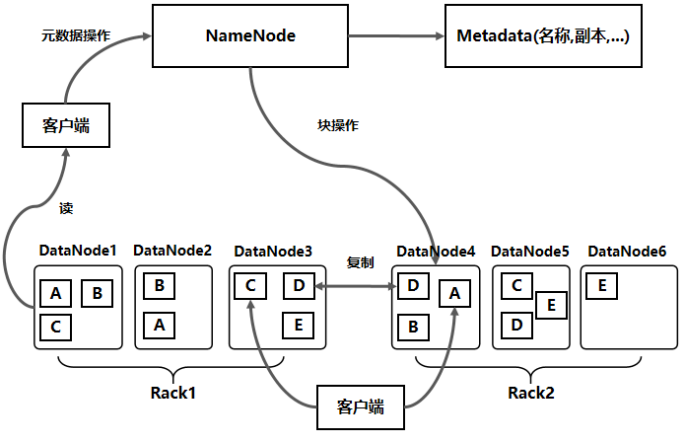

图 2.2　HDFS 架构图

- NameNode：存储文件系统的元数据，即文件名、文件块信息、块位置、权限等，也管理数据节点。

- DataNode：存储实际业务数据（文件块）的从节点，根据NameNode指令为客户端读/写请求提供服务。

NameNode 负责管理 block 的复制，它周期性地接收 HDFS 集群中所有 DataNode 的心跳数据包（heartbeats）和 block 报告。心跳包表示 DataNode 正常工作，block 报告描述了该 DataNode 上所有的 block 组成的列表，并根据需要更新 NameNode 上的状态信息。

当文件读取时，客户端向 NameNode 节点发起文件读取的请求。NameNode 返回文件存储的 block 块信息及其 block 块所在 DataNode 的信息。客户端根据这些信息，即可到具体的 DataNode 上进行文件读取。

由于数据流分散在 HDFS 集群中的所有 DataNode 节点上，且 NameNode 只响应块位置的请求（存储在内存中，速度很快），而无须响应数据请求，所以这种设计能适应大量的并发访问。

当文件写入时，客户端向 NameNode 节点发起文件写入的请求。NameNode 根据文件大小和文件块配置的情况，返回给客户端它所管理部分 DataNode 的信息。客户端将文件划分为多个 block 块，并根据 DataNode 的地址信息，按顺序写入到每一个 DataNode 块中。HDFS 中的文件默认规则是一次写、多次读，并且严格要求在任何时候只有一个写操作（writer）。

注 意

除了最后一个 block，所有的 block 大小都是一样的（128MB）。当一个 1MB 的文件存储在一个 128MB 的 block 中时，文件只使用 1MB 的磁盘空间，而不是 128MB。

备份数据的存放是 HDFS 可靠性和性能的关键。HDFS 采用一种称为 Rack-Aware 的策略来决定备份数据的存放。通过 Rack Awareness 过程，NameNode 给每个 DataNode 分配 Rack Id。比如，DataNode1 属于 Rack1，DataNode4 属于 Rack2。

HDFS 在默认情况下，一个 block 会有 3 个备份，一个在 NameNode 指定的 DataNode 上（假如是 Rack1 下的 DataNode1），一个在指定 DataNode 非同一 Rack 的 DataNode 上（假如是 Rack2 下的 DataNode4），一个在指定 DataNode 同一 Rack 的 DataNode 上（假如是 Rack1 下的 DataNode2）。这种策略综合考虑了同一 Rack 失效，以及不同 Rack 之间数据复制的性能问题。

在读取副本数据时，为了降低带宽消耗和读取延时，HDFS 会尽量读取最近的副本。如果在同一个 Rack 上有一个副本，那么就读该副本。

注 意

Hadoop 服务启动后先进入安全模式，此时系统中的内容不允许修改和删除，直到安全模式结束。

2.1.3 Hadoop 生态系统

Hadoop 可以说是奠定了大数据开源解决方案的基础，因此，不少大数据工具都被纳入 Hadoop 生态系统。随着 Hadoop 与各种各样的 Apache 开源大数据项目的融合，Hadoop 生态

系统也在不断地变化。

Hadoop 正在不断地将其核心组件 HDFS 和 YARN 扩展为一个更为复杂的开源大数据系统，也就是 Hadoop 生态系统。图 2.3 给出了 Hadoop 生态系统中的工具集，其中包含形成 Hadoop 生态系统的各种功能。

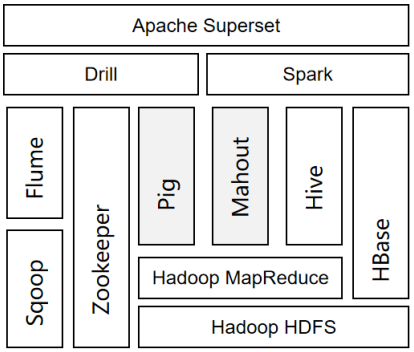

图 2.3　Hadoop 生态系统图

- Hadoop HDFS

HDFS 是一个容错、分布式、水平可扩展的存储系统，可跨多个服务器工作。它可以用作 Hadoop 集群的一部分，也可以用作独立的通用文件系统，它是很多大数据工具的默认数据存储系统，起到非常重要的作用。

另外，Hadoop 是开源的，这意味着一个组织可以运行这个文件系统来处理数 PB 级别的数据，而无须支付软件成本。

- Hadoop MapReduce

MapReduce 是一个分布式编程框架，它可以以并行方式处理 TB 级别的数据。因此非常适合处理离线的数据，而且非常稳定。

- Flume

Flume 是一个分布式的海量日志采集、聚合和传输的系统，它支持在日志系统中定制各类数据发送方，用于收集数据。同时，Flume 提供对数据进行简单处理，并具有写入各种数据存储系统的能力。

- Sqoop

Sqoop 支持 HDFS、Hive、HBase 与关系型数据库之间的批量数据双向传输（导入/导出）。与 Flume 不同，Sqoop 在结构化数据的传输上操作更加方便。

- Pig

Pig 是一个基于 Hadoop 的大数据分析平台，它提供的 SQL-like 语言叫 Pig Latin（其实并不好用），该语言的编译器会把类 SQL 的数据分析请求，转换为一系列经过优化处理的 MapReduce 运算。Pig 为复杂的海量数据并行计算提供了一个简单的操作和编程接口。

注 意

目前 Pig 用得也比较少，更多的是通过 Hive SQL、Spark SQL 或者 Flink SQL 来编写数据分析任务。

- Mahout

Mahout 是在 MapReduce 之上实现的一套可扩展的机器学习库。不过当前随着 Spark 和 Flink 等软件的流行，Mahout 逐步被其他机器学习库所替代。

- Hive

Hive 是一个 SQL 翻译器，可以基于类似 SQL 语言的 HiveQL 来编写查询。Hive 可以将 HDFS 和 HBase 中的数据集映射到表上。虽然 Hive 对于一些复杂 SQL 还不能很好地支持，但是常用的数据查询任务基本都可以用 SQL 来解决，这让开发人员只需用 SQL 就可以完成 MapReduce 作业。

- HBase

HBase 是一个 NoSQL 分布式的面向列的数据库，它运行在 HDFS 之上，可以对 HDFS 执行随机读/写操作，它是 Google Big Table 的开源实现。HBase 能够近实时地存储和检索随机数据。这就很好地弥补了 Hadoop 在实时应用上的不足。

- ZooKeeper

ZooKeeper 是一个开源的分布式应用程序协调服务，是 Google Chubby 的一个开源实现。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步和组服务等。

- Spark

Spark 是专为大规模数据处理而设计的、快速的通用计算引擎。它目前已经可以对大数据中的批处理和流处理进行处理，且 Spark 计算速度比 Hadoop MapReduce 要更快。

- Drill

Drill 是一个用于 Hadoop 和 NoSQL 数据库的低延迟 SQL 查询引擎，它支持 Parquet、JSON 或 XML 等数据格式。Drill 响应速度可达到亚秒级，适合交互式数据分析。

- Apache Superset

Superset 是一个开源的可视化工具，可以接入多种数据源。开发人员或者业务人员可以借助它快速构建出美观的管理面板。

注 意

Hadoop 生态系统有不同的版本，是一个不断发展的工具集，并没有一个统一的标准。

2.2　Spark 与 Hadoop

在当前的大数据领域，Apache Spark 无疑占有重要的位置，特别是新版本的 Spark 在流处理方面的加强，使它可以适用实时数据分析的场景。

在 Spark 出现之前，想要在一个组织内同时完成多种大数据分析任务，必须部署多套大数据工具，比如离线分析用 Hadoop MapReduce，查询数据用 Hive，流数据处理用 Storm，而机器学习用 Mahout。

在这种情况下，一方面增加了大数据系统开发的难度，需要有不同技能的人员共同协作才能完成，这也会导致系统的运维变得复杂。另一方面，由于不同大数据工具之间需要互相传递数据，而对于数据的格式可能要求不同，因此需要在多个系统间进行数据格式转换和传输工作，这个过程既费时又费力。

Spark 软件是一个“One Stack to rule them all”的大数据计算框架，它的目标是用一个技术栈完美地解决大数据领域的各种计算任务。Spark 官方对 Spark 的定义是：通用的大数据快速处理引擎。

从某种程度上来说，Spark 和 Hadoop 软件的组合，是未来大数据领域性价比最高的组合，也是最有发展前景的组合。

2.2.1　Apache Spark 概述

2009 年，Spark 诞生于伯克利大学的 AMPLab 实验室，AMP 是 Algorithms、Machines 与 People 的缩写。一开始 Spark 只是一个学术上的实验性项目，代码量并不多，可以称得上是一个轻量级的框架。

2010 年，伯克利大学正式开源了 Spark 项目。2013 年，Spark 成为 Apache 基金会下的项目，进入高速发展期。同年，Apache Spark 的开发团队成立了 Databricks 公司。

2014 年，Spark 仅在一年左右的时间，就以非常快的速度成为 Apache 的顶级项目。从 2015 年开始，Spark 在国内大数据领域变得异常火爆，很多大型的公司开始使用 Spark 来替代 Hadoop MapReduce、Hive 和 Storm 等传统的大数据计算框架。

现在已经有很多公司在生产环境下使用 Apache Spark 作为大数据的计算框架，包括 eBay、雅虎、阿里、百度、腾讯、网易、京东、华为、优酷、搜狗和大众点评等。与此同时，Spark 也获得了多个世界级 IT 厂商的支持，其中包括 IBM 和 Intel 等。

可以说，Spark 用 Spark RDD、Spark SQL、Spark Streaming、Spark MLlib 和 Spark GraphX 成功解决了大数据领域中，离线批处理、交互式查询、实时流计算、机器学习与图计算等最常见的计算问题。

2.2.2 Spark 和 Hadoop 比较

1．实现语言不同

Apache Spark 框架是用 Scala 语言编写。Scala 是一门多范式（Multi-Paradigm）的编程语言，设计初衷是要集成面向对象编程和函数式编程的各种特性。Scala 运行在 Java 虚拟机上，并兼容现有的 Java 程序。Scala 源代码被编译成 Java 字节码，所以它可以运行在 JVM 之上，并可以调用现有的 Java 类库。

而 Hadoop 是由 Java 语言开发的。Java 是一门面向对象的编程语言，具有功能强大和简单易用的特征。Java 具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程等特点。可以说，作为一个大数据从业人员，Java 语言几乎是必会的一门语言。

2．数据计算方式不同

Apache Spark 最重要的特点是基于内存进行计算，因此计算的速度可以达到 Hadoop MapReduce 或 Hive 的数十倍，甚至上百倍。很多应用程序，为了提升程序的响应速度，常用的方法就是将数据在内存中进行缓存。一般来说，Apache Spark 对计算机的内存要求比较高。

通常来说，Apache Spark 中 RDD 存放在内存中，如果内存不够存放数据，会同时使用磁盘存储数据。因此，为了提升 Spark 对数据的计算速度，应该尽可能让计算机的内存足够大，这样可以防止数据缓存到磁盘上。

而 Hadoop MapReduce 是从 HDFS 中读取数据的，通过 MapReduce 将中间结果写入 HDFS，然后再重新从 HDFS 读取数据进行 MapReduce 操作，再回写到 HDFS 中，这个过程涉及多次磁盘 IO 操作，因此，计算速度相对来说比较慢。

3．使用场景不同

Apache Spark 只是一个计算分析框架，虽然可以在一套软件栈内完成各种大数据分析任务，但是它并没有提供分布式文件系统，因此必须和其他的分布式文件系统进行集成才能运作。

Spark 是专门用来对分布式数据进行计算处理，它本身并不能存储数据。可以说，Spark 是大数据处理的瑞士军刀，支持多种类型的数据文件，如 HDFS、HBase 和各种关系型数据库，可以同时支持批处理和流数据处理。

而 Hadoop 主要由 HDFS、MapReduce 和 YARN 构成。其中 HDFS 作为分布式数据存储，这也是离线数据存储的不二选择。另外，借助 MapReduce 可以很好地进行离线数据批处理，而且非常稳定，对于实时性要求不高的批处理任务，用 MapReduce 也是一个不错的选择。

4．实现原理不同

在 Apache Spark 中，用户提交的任务称为 Application，一个 Application 对应一个 SparkContext。一个 Application 中存在多个 Job，每触发一次 Action 操作就会产生一个 Job。这些 Job 可以并行或串行执行，每一个 Job 中有多个 Stage，每一个 Stage 里面有多个 Task，

由 TaskScheduler 分发到各个 Executor 中执行。Executor 的生命周期和 Application 一样，即使没有 Job 运行也是存在的，所以 Task 可以快速启动读取内存进行计算。

另外，在 Spark 中每一个 Job 可以包含多个 RDD 转换算子，在调度时可以生成多个 Stage，借助 Spark 框架中提供的转换算子和 RDD 操作算子，可以实现很多复杂的数据计算操作，而这些复杂的操作在 Hadoop 中原生是不支持的。

在 Hadoop 中，一个作业称为一个 Job，Job 里面分为 Map Task 和 Reduce Task，每个 Task 都在自己的进程中运行，当 Task 结束时，进程也会随之结束。

注　意

Spark 虽然号称是通用的大数据快速处理引擎，但是目前还不能替换 Hadoop，因为 Spark 并没有提供分布式文件系统。

2.3　Spark 核心概念

2.3.1　Spark 软件栈

Apache Spark 是一个快速通用的分布式计算平台，Spark 支持更多的计算模式，包括交互式查询和流数据处理。在处理大规模数据集时，Spark 的一个核心优势是内存计算，因而处理速度更快。

Spark 在统一的框架下，一站式提供批处理、迭代计算、交互式查询、流数据处理、资源调度、机器学习以及图计算。Spark 支持多种编程语言，包括 Scala、Java、Python 和 R 等。图 2.4 给出了 Spark 的软件栈架构图。

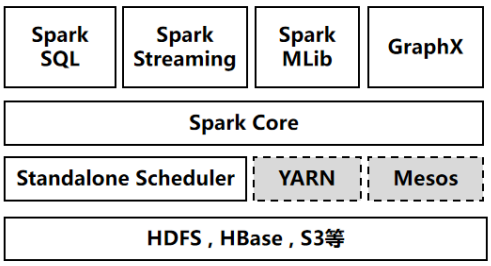

图 2.4　Spark 的软件栈架构图

Spark 软件栈核心组件如下：

- Spark Core：Spark Core包含Spark的基本功能，包含任务调度、内存管理和容错机制等，内部定义了RDD（弹性分布式数据集），提供了很多API来创建和操作这些RDD，为其他组件提供底层的服务。
- Spark SQL：Spark SQL可以处理结构化数据的查询分析，对于HDFS、HBase等多种数据源中的数据，可以用Spark SQL来进行数据分析。

- Spark Streaming：Spark Streaming是实时数据流处理组件，类似Storm。Spark Streaming提供了API来操作实时流数据。一般需要配合消息队列Kafka，来接收数据做实时统计分析。
- Spark Mllib：MLlib是一个包含通用机器学习功能的包，是Machine Learning Lib的缩写，主要包括分类、聚类和回归等算法，还包括模型评估和数据导入。MLlib提供的机器学习算法库，支持集群上的横向扩展。
- Spark GraphX：GraphX是专门处理图的库，如社交网络图的计算。与Spark Streaming和Spark SQL一样，也提供了RDD API。它提供了各种图的操作和常用的图算法。

Spark 提供了一站式的软件栈，因此只要掌握 Spark 这一个工具，就可以编写不同场景的大数据处理应用程序。

2.3.2 Spark 运行架构

Spark 的架构示意图如图 2.5 所示，下面将分别介绍各核心组件。

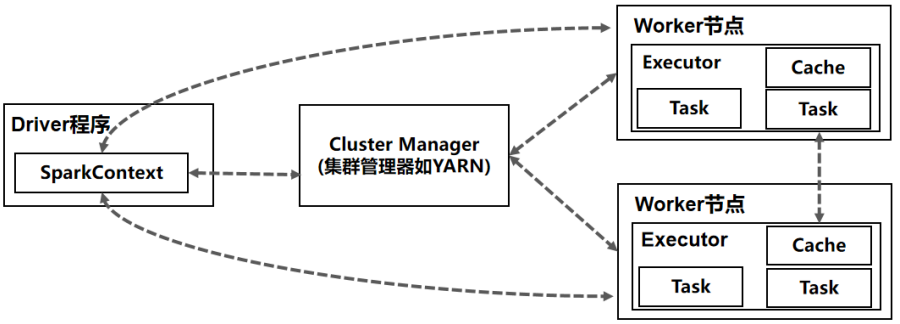

图 2.5　Spark 的架构示意图

Spark 中的重要概念如下：

- Application：提交一个作业就是一个Application，一个Application只有一个SparkContext。由群集上的驱动程序和执行程序组成。
- Driver程序：一个Spark作业运行时会启动一个Driver进程，也是作业的主进程，负责作业的解析、生成Stage和调度Task到Executor上执行。Driver程序运行应用程序的main函数，并创建SparkContext进程。
- Cluster Manager：Cluster Manager是用于获取群集资源的外部服务（如Standalone、YARN或Mesos），在Standalone模式中即为Master（主节点）。Master是集群的领导者，负责管理集群的资源，接收Client提交上来的作业，以及向Worker节点发送命令。在YARN模式中为资源管理器。
- Worker：Worker节点是集群中的Worker，执行Master发送来的命令来具体分配资源，并在这些资源上执行任务Task。在YARN模式中为NodeManager，负责计算节点的控制。
- Executor：Executor是真正执行作业Task的地方。Executor分布在集群的Worker节点

上，每个Executor接收Driver命令来加载和运行Task。一个Executor可以执行一个到多个Task。多个Task之间可以互相通信。

- SparkContext：SparkContext是程序运行调度的核心，由调度器DAGScheduler划分程序的各个阶段，调度器TaskScheduler划分每个阶段的具体任务。SchedulerBankend管理整个集群中为正在运行的程序分配计算资源的Executor。SparkContext是Spark程序入口。
- DAGScheduler：负责高层调度，划分Stage，并生成程序运行的有向无环图。
- TaskScheduler：负责具体Stage内部的底层调度、具体Task的调度和容错等。
- Job：Job是工作单元，每个Action算子都会触发一次Job，一个Job可能包含一个或多个Stage。
- Stage：Stage用来计算中间结果的Tasksets。Tasksets中的Task逻辑对于同一RDD内的不同Partition都一样。Stage在Shuffle的时候产生，如果下一个Stage要用到上一个Stage的全部数据，则要等上一个Stage全部执行完才能开始。Stage有两种：ShuffleMapStage和ResultStage。除了最后一个Stage是ResultStage外，其他的Stage都是ShuffleMapStage。ShuffleMapStage会产生中间结果，以文件的方式保存在集群里，Stage经常被不同的Job共享，前提是这些Job重用了同一个RDD。
- Task：Task是任务执行的工作单位，每个Task会被发送到一个Worker节点上，每个Task对应RDD的一个Partition。
- Taskset：划分的Stage会转换成一组相关联的任务集。
- RDD：RDD指弹性分布数据集，它是不可变的、Lazy级别的、粗粒度的数据集合，包含一个或多个数据分片，即Partition。
- DAG：DAG（Directed Acyclic Graph）指有向无环图。Spark实现了DAG计算模型，DAG计算模型是指将一个计算任务按照计算规则分解为若干子任务，这些子任务之间根据逻辑关系构建成有向无环图。
- 算子：Spark中两种算子：Transformation和Action。Transformation算子会由DAGScheduler划分到pipeline中，是Lazy级别的，它不会触发任务的执行。而Action算子会触发Job来执行pipeline中的运算。
- 窄依赖：窄依赖（Narrow Dependency）指父RDD的分区只对应一个子RDD的分区。如果子RDD只有部分分区数据损坏或者丢失，只需要从对应的父RDD重新计算即可恢复。
- 宽依赖：宽依赖（Shuffle Dependency）指子RDD分区依赖父RDD的所有分区。如果子RDD部分分区甚至全部分区数据损坏或丢失，则需要从所有父RDD上重新进行计算。相对窄依赖而言，数据处理的成本更高，所以应尽量避免宽依赖的使用。
- Lineage：每个RDD都会记录自己依赖的父RDD信息，一旦出现数据损坏或者丢失，将从父RDD迅速重新恢复。

2.3.3　Spark 部署模式

Spark 有多种部署模式（Deploy Mode）。不同部署模式下，驱动程序 Driver 进程的运行位置是不同的。在 Cluster 模式下，在集群内部启动驱动程序。在 Client 模式下，在群集外部

启动驱动程序。

1．Local模式

本地采用多线程的方式执行，主要用于开发测试。下面的语句则是用spark-submit命令提交 jar 包定义的任务，以 Local 模式执行：

```
./bin/spark-submit \
--class com.myspark.Job.WordCount \
--master local[*] \
/root/sparkjar/spark-demo-1.0.jar
```

本地提交时，本地文件路径前缀为 file:///。其中 local[*]表示启动与本地 Worker 机器的 CPU 个数一样的线程数来运行 Spark 任务。

2．Spark on YARN模式

在 Spark on YARN 模式下，每个 Spark Executor 作为一个 YARN Container 在运行，同时支持多个任务在同一个 Container 中运行，极大地节省了任务的启动时间。Spark on YARN 有两种模式，分别为 yarn-client 和 yarn-cluster 模式。

yarn-cluster 模式下，Driver 运行在 Application Master 中，即运行在集群中的某个节点上，节点的选择由 YARN 进行调度，作业的日志在 YARN 的 Web UI 中查看，适合大数据量、非交互式的场景。

这种模式下，当用户提交作业之后，就可以关闭 Client 程序，作业会继续在 YARN 上运行。由于它的计算结果不会在 Client 端显示，因此这种模式不适合交互类型的作业。

下面的语句则是用 spark-submit 命令提交 jar 包定义的任务，以 yarn-cluster 模式执行：

```
./bin/spark-submit \
--class com.myspark.Job.WordCount  \
--master yarn \
--deploy-mode cluster \
/root/sparkjar/spark-demo-1.0.jar
```

而 yarn-client 模式下，Driver 运行 Client 端，会和请求的 YARN Container 通信来调度它们工作，也就是说 Client 在计算期间不能关闭，且计算结果会返回到 Client 端，因此适合交互类型的作业。

下面的语句则是用 spark-submit 命令提交 jar 包定义的任务，以 yarn-client 模式执行：

```
./bin/spark-submit \
--class com.myspark.Job.WordCount  \
--master yarn \
--deploy-mode client \
/root/sparkjar/spark-demo-1.0.jar
```

这两种提交方式的区别如下：

（1）yarn-cluster 的 Driver 是在集群的某一台 Node Manager 节点上运行，而 yarn-client

的 Driver 运行在 Client 上。

（2）yarn-cluster 在提交应用之后，可以关闭 client，而 yarn-client 则不可以。

（3）应用场景不同，yarn-cluster 适合生产环境，而 yarn-client 适合交互和调试。

3．Standalone模式

Standalone 模式和 Local 模式类似，但 Standalone 的分布式调度器是 Spark 提供的，如果一个集群是 Standalone 模式的话，那么需要在每台机器上部署 Spark。

下面的语句则是用 spark-submit 命令提交 jar 包定义的任务，以 Standalone 模式执行：

```
./bin/spark-submit  \
--class com.myspark.Job.WordCount \
-master spark://192.168.1.71:7077 \
-executor-memory 16G \
-total-executor-cores 16 \
/root/sparkjar/spark-demo-1.0.jar
```

在 yarn-client 和 yarn-cluster 模式下提交作业时，可以不启动 Spark 集群，因为相关的 JVM 环境由 YARN 管理。而 Standalone 模式提交作业时，Spark 集群必须启动，因为它运行依赖的调度器来自 Spark 集群本身。

注　意
spark-submit 提交 jar 包到 YARN 上时，输入路径和输出路径都必须是 HDFS 的路径，即 hdfs://ip:port/，否则报错。

2.4　Spark 基本操作

Spark 对内存数据的抽象，即为 RDD。RDD 是一种分布式、多分区、只读的数组，Spark 相关操作都是基于 RDD 进行的。Spark 可以将 HDFS 块文件转换成 RDD，也可以由一个或多个 RDD 转换成新的 RDD。基于这些特性，RDD 在分布式环境下能够被高效地并行处理。

PySpark 是 Apache Spark 社区发布的一个工具，让熟悉 Python 的开发人员可以方便地使用 Spark 组件。PySpark 借助 Py4j 库，可以使用 Python 编程语言处理 RDD。

PySpark 的基本数据流架构如图 2.6 所示。

由图 2.6 可知，PySpark 首先利用 Python 创建 Spark Context 对象，然后用 Socket 与 JVM 上的 Spark Context 通信，当然，这个过程需要借助 Py4J 库。JVM 上的 Spark Context 负责与集群上的 Spark Worker 节点进行交互。

在 PySpark 中，对 RDD 提供了 Transformation 和 Action 两种操作类型。Transformation 操作非常丰富，采用延迟执行的方式，在逻辑上定义了 RDD 的依赖关系和计算逻辑，但并不会真正触发执行动作，只有等到 Action 操作才会触发真正执行操作。Action 操作常用于最终结果的输出。

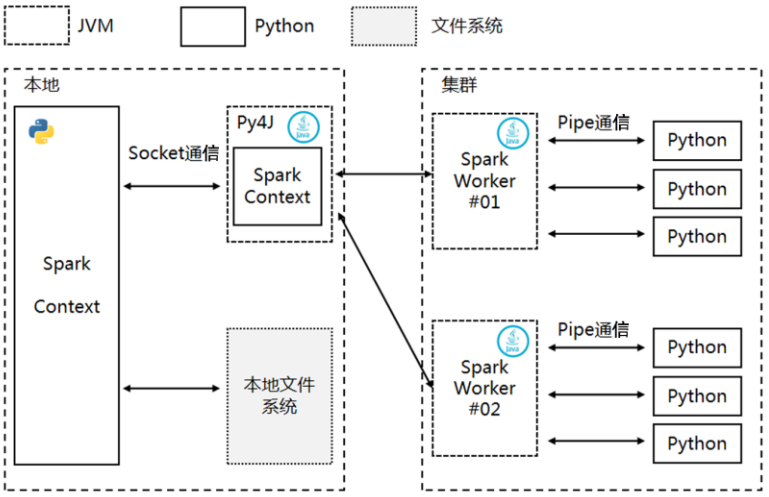

图 2.6　PySpark 的数据流架构示意图

常用的 Transformation 操作及其描述如下：

- map：map 接收一个处理函数并行处理源RDD中的每个元素，返回与源RDD元素一一对应的新RDD。
- filter：filter并行处理源RDD中的每个元素，接受一个函数，并根据定义的规则对RDD中的每个元素进行过滤处理，返回处理结果为true的元素重新组成新的RDD。
- flatMap：flatMap是map和flatten的组合操作，与map函数相似，不过map函数返回的新RDD包含的元素可能是嵌套类型。
- mapPartitions：与map函数应用于RDD中的每个元素不同，mapPartitions应用于RDD中的每个分区。mapPartitions函数接受的参数为一个函数，该函数的参数为每个分区的迭代器，返回值为每个分区元素处理之后组成的新的迭代器，该函数会作用于分区中的每一个元素。
- mapPartitionsWithIndex：作用与mapPartitions函数相同，只是接受的参数（一个函数）需要传入两个参数，分区的索引作为第一个参数传入，按照分区的索引对分区中元素进行处理。
- Union：将两个RDD进行合并，返回结果为RDD中元素（不去重）。
- Intersection：对两个RDD进行取交集运算，返回结果为RDD无重复元素。
- Distinct：对RDD中元素去重。
- groupByKey：在键值对（K-V）类型的RDD中按Key分组，将相同Key的元素聚集到同一个分区内，此函数不能接受函数作为参数，只接受一个可选参数，即任务数。
- reduceByKey：对K-V类型的RDD按Key分组，它接受两个参数，第一个参数为处理函数，第二个参数为可选参数，用于设置reduce的任务数。reduceByKey函数能够在RDD分区本地提前进行聚合运算，这有效减少了shuffle过程传输的数据量。相对于groupByKey函数更简洁高效。
- aggregateByKey：对K-V类型的RDD按Key分组进行reduce计算，可接受三个参数，第

一个参数是初始化值，第二个参数是分区内处理函数，第三个参数是分区间处理函数。

- sortByKey：对K-V类型的RDD内部元素按照Key进行排序，排序过程会涉及Shuffle操作。
- join：对K-V类型的RDD进行关联操作，它只能处理两个RDD之间的关联，超过两个RDD关联需要多次使用join函数。另外，join操作只会关联出具有相同Key的元素，相当于SQL语句中的inner join。
- cogroup：对K-V类型的RDD进行关联，cogroup在处理多个RDD的关联上比join更加优雅，它可以同时传入多个RDD作为参数进行关联。
- coalesce：对RDD重新分区，将RDD中的分区数减小到参数numPartitions个，不会产生shuffle。在较大的数据集中使用filer等过滤操作后可能会产生多个大小不等的中间结果数据文件，重新分区并减小分区可以提高作业的执行效率，是Spark中常用的一种优化手段。
- repartition：对RDD重新分区，接受一个参数，即numPartitions分区数，它是coalesce函数设置shuffle为true的一种实现形式。

常用的 Action 操作及其描述如下：

- reduce：处理RDD两两之间元素的聚集操作。
- collect：返回RDD中所有数据元素。
- Count：返回RDD中元素个数。
- First：返回RDD中的第一个元素。
- Take：返回RDD中的前N个元素。
- saveAsTextFile：将RDD写入文本文件，保存至本地文件系统或者HDFS中。
- saveAsSequenceFile：将K-V类型的RDD写入Sequence File文件，保存至本地文件系统或者HDFS中。
- countByKey：返回K-V类型的RDD，这个RDD中数据为每个Key包含的元素个数。
- Foreach：遍历RDD中所有元素，接受参数为一个函数，常用操作是传入println函数打印所有元素。

从 HDFS 文件生成 RDD，经过 map、filter、join 等多次 Transformation 操作，最终调用 saveAsTextFile 操作，将结果输出到 HDFS 中，并以文件形式保存。RDD 操作的基本流程示意图如图 2.7 所示。

在 PySpark 中，RDD 可以缓存到内存或者磁盘上，提供缓存的主要目的是减少同一数据集被多次使用的网络传输次数，提高 PySpark 的计算性能。PySpark 提供对 RDD 的多种缓存级别，可以满足不同场景对 RDD 的使用需求。RDD 的缓存具有容错性，如果有分区丢失，可以通过系统自动重新计算。

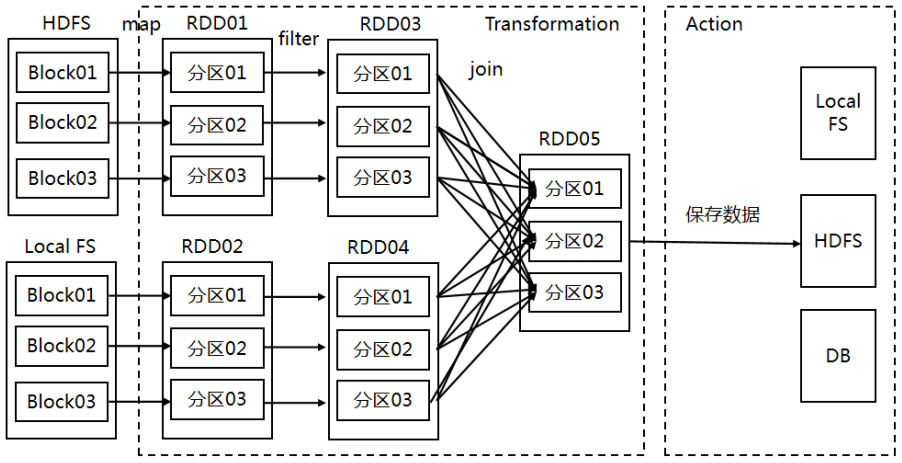

图 2.7　RDD 操作的基本流程示意图

在代码中可以使用 persist()方法或 cache()方法缓存 RDD。cache()方法默认将 RDD 缓存到内存中，cache()方法和 persist()方法都可以用 unpersist()方法来取消 RDD 缓存。具体如下所示：

```
rdd= sc.textFile("hdfs://data/hadoop/test.text")
rdd.cache() // 缓存 RDD 到内存
```

或者

```
rdd.persist(StorageLevel.MEMORY_ONLY)
rdd.unpersist() // 取消缓存
```

Spark 各缓存级别及其描述：

- MEMORY_ONLY：RDD仅缓存一份到内存，此为默认级别。
- MEMORY_ONLY_2：将RDD分别缓存在集群的两个节点上，RDD在集群内存中保存两份。
- MEMORY_ONLY_SER：将RDD以Java序列化对象的方式缓存到内存中，有效减少了RDD在内存中占用的空间，不过读取时会消耗更多的CPU资源。
- DISK_ONLY：RDD仅缓存一份到磁盘。
- MEMORY_AND_DISK：RDD仅缓存一份到内存，当内存中空间不足时，会将部分RDD分区缓存到磁盘。
- MEMORY_AND_DISK_2：将RDD分别缓存在集群的两个节点上，当内存中空间不足时，会将部分RDD分区缓存到磁盘，RDD在集群内存中保存两份。
- MEMORY_AND_DISK_SER：将RDD以Java序列化对象的方式缓存到内存中，当内存中空间不足时，会将部分RDD分区缓存到磁盘，有效减少了RDD在内存中占用的空间，不过读取时会消耗更多的CPU资源。
- OFF_HEAP：将RDD以序列化的方式缓存到JVM之外的存储空间中，与其他缓存模式相比，减少了JVM垃圾回收开销。

注　意

Spark 的操作还有很多，具体可以参考 Spark 官网的相关文档。这里只是给出了基本操作的语言描述，因此会比较难于理解，后续会详细通过示例来说明每个操作的具体用法。

2.5　SQL in Spark

Spark SQL 的前身是 Shark，即 Hive on Spark，本质上是通过 Hive 的 HQL 进行解析，把 HQL 翻译成 Spark 上对应的 RDD 操作，然后通过 Hive 的 Metadata 获取数据库里的表信息，最后获取相关数据并放到 Spark 上进行运算。

Spark SQL 支持大量不同的数据源，包括 Hive、JSON、Parquet、JDBC 等，允许开发人员直接用 SQL 来处理数据。它的一个重要特点是能够统一处理二维表和 RDD，这就使得开发人员可以使用 SQL 进行更复杂的数据分析。

Spark SQL 的特点如下：

- SchemaRDD：引入了新的RDD类型SchemaRDD，可以像传统数据库定义表一样来定义RDD。SchemaRDD由列数据类型和行对象构成。它与RDD可以互相转换。
- 多数据源支持：可以混合使用不同类型的数据，包括Hadoop、Hive、JSON、Parquet和JDBC等。
- 内存列存储：Spark SQL的表数据在内存中采用内存列存储（In-Memory Columnar Storage），这样计算的速度更快。
- 字节码生成技术：Spark 1.1.0版本后在Catalyst模块的Expressions增加了Codegen模块，使用动态字节码生成技术，对匹配的表达式采用特定的代码动态编译。另外，对SQL表达式也做了优化。因此，效率进一步提升。

2.6　Spark 与机器学习

如何让计算机更加智能化，从计算机诞生之时，就是不少计算机科学家的梦想。在智能计算领域，先后提出人工智能（Artificial Intelligence）、数据挖掘（Data Mining）、机器学习（Machine Learning）和深度学习（Deep Learning），这几块都有各自的专有内容，同时也有交集。

根据百度百科上的定义，机器学习是一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科。专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构，并不断改善自身的性能。

机器学习是人工智能的核心，是使计算机具有智能的根本途径。它是当前计算机领域的研究热点。最近，我国提出新型基础设施建设（新基建）主要包括 5G 基站建设、特高压、城际高速铁路和城市轨道交通、新能源汽车充电桩、大数据中心、人工智能、工业互联网七大

领域，提供数字转型、智能升级、融合创新等服务的基础设施体系。

Spark 当中也专门提供了机器学习算法库 MLlib（Machine Learning Library）。MLlib 中已经包含了一些通用的学习算法，具体罗列如下：

- 分类（Classification）算法
  - Logistic regression
  - Decision tree classifier
  - Random forest classifier
  - Gradient-boosted tree classifier
  - Multilayer perceptron classifier
  - Linear Support Vector Machine
  - One-vs-Rest classifier
  - Naive Bayes
  - Factorization machines classifier
- 回归Regression算法
  - Linear regression
  - Generalized linear regression
  - Decision tree regression
  - Random forest regression
  - Gradient-boosted tree regression
  - Survival regression
  - Isotonic regression
  - Factorization machines regressor
- 聚类Clustering
  - K-means
  - Latent Dirichlet allocation (LDA)
  - Bisecting k-means
  - Gaussian Mixture Model (GMM)
  - Power Iteration Clustering (PIC)
- 协同过滤（Collaborative Filtering）
- 关联规则（Frequent Pattern Mining）
- 降维（Dimensionality Reduction）
  - Singular value decomposition (SVD)
  - Principal component analysis (PCA)

MLlib 算法库除了提供基本的机器学习算法外，还提供了如下工具：

- 特征化（Featurization）：特征提取、变换、降维和选择。
- 管道（Pipelines）：用于构建、评估和调整ML管道的工具。
- 持久性（Persistence）：保存和加载算法、模型和管道。

- 通用工具（Utilities）：线性代数、统计信息、数据处理等。

算法中有分类和回归之分，其实本质上分类模型和回归模型是一样的。分类模型是将回归模型的输出离散化，比如预测某个上市公司财务是否良好，用1代表好，用0代表不好。而回归模型用于处理连续的值，比如预测一个公司明年的产量。下面简要介绍常用的机器学习算法的基本概念以及用法。

机器学习应用于实际项目中，其一般的流程如图 2.8 所示。它强调将机器学习用来解决实际的商业问题，它是一个不断迭代的过程。

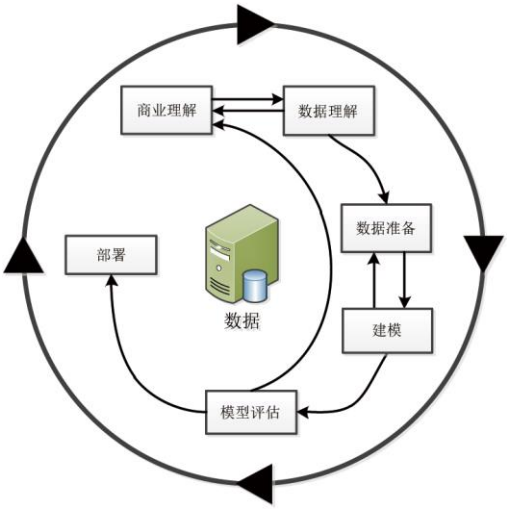

图 2.8　机器学习流程示意图

2.6.1　决策树算法

决策树（Decision Tree）充分利用了树形模型，根节点到一个叶节点是一条分类的路径规则，每个叶子节点象征一个判断类别。决策树有 2 类：

- 分类树：对离散变量做决策树。
- 回归树：对连续变量做决策树。

先将样本分成不同的子集，再进行分割递推，直至每个子集得到同类型的样本，从根节点开始测试，到子树再到叶子节点，即可得出预测类别。此方法的特点是结构简单、处理数据效率较高。

决策树优点如下：

- 速度快，计算量相对较小，且容易转化成分类规则。
- 准确性高，挖掘出来的分类规则准确性高，非常容易理解。
- 可以处理连续和分类数据。
- 不需要任何领域知识和参数假设。
- 适合高维数据。

决策树缺点如下：

- 容易产生过拟合的情况。
- 忽略属性之间的相关性。

下面给出一个决策树示意图，如图 2.9 所示。

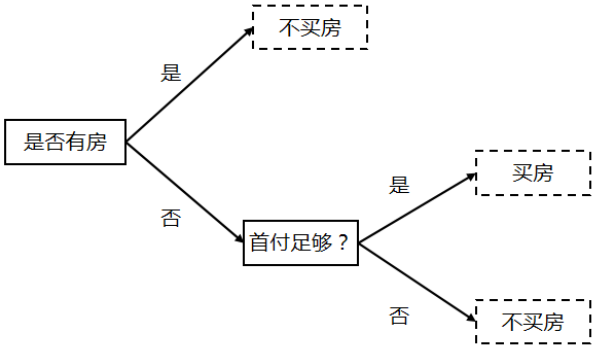

图 2.9　决策树示意图

2.6.2　贝叶斯算法

贝叶斯（Naive Bayes）算法是一种分类算法，也叫作朴素贝叶斯算法。它不是单一算法，而是一系列算法，它们都有一个共同的原则，即被分类的每个特征都与任何其他特征的值无关。

贝叶斯算法的优点如下：

- 计算过程简单、速度快。
- 适用多分类问题。
- 在分布独立这个假设成立的情况下，预测效果非常好，且需要的样本数据也更少。
- 可以处理连续和分类数据。

朴素贝叶斯分类器认为这些“特征”中的每一个都独立地贡献概率，而不管特征之间的任何相关性。然而，特征并不总是独立的，这通常被视为朴素贝叶斯算法的缺点。

2.6.3　支持向量机算法

支持向量机 SVM（Support Vector Machine）算法，首先利用一种变换将空间高维化，当然这种变换是非线性的，然后在新的复杂空间取最优线性分类面。

SVM 是统计学领域中一个代表性算法，它与传统的思维方法很不同，它输入空间数据、通过维度变换从而将问题简化，使问题归结为线性可分的经典问题。

SVM 算法优点如下：

- 可以向高维空间映射。

- 可以解决非线性的分类问题。
- 分类思想很简单，就是将样本与决策面的间隔最大化。
- 分类效果较好。

SVM 算法缺点如下：

- 在大规模样本上难以进行模型训练。
- 难于解决多分类问题。
- 对缺失数据敏感。
- 对参数和核函数的选择敏感。
- 模型类似一个黑盒，评估结果难以解释。

下面给出一个 SVM 的示意图，如图 2.10 所示。

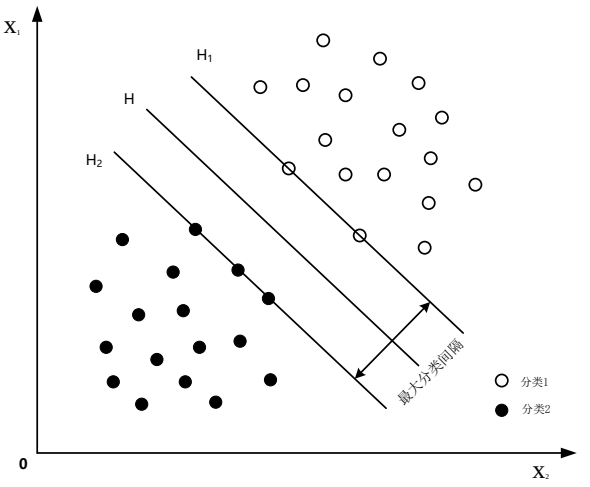

图 2.10　SVM 示意图

2.6.4　随机森林算法

由于传统的分类模型往往精度不高，且容易出现过拟合问题。因此，很多学者通过聚集多个模型来提高预测精度，这些方法称为组合或分类器组合方法。

这些方法的一个共同特征是：为第 k 棵决策树生成随机向量 Θ_k，且 Θ_k 独立同分布于前面的随机向量 $\Theta_1,\Theta_2,\cdots,\Theta_{k-1}$。利用训练集和随机向量 Θ_k 生成一棵决策树，得到分类模型 $h(\mathbf{X},\Theta_k)$，其中 X 为输入变量（自变量）。在生成许多决策树后，通过投票方法或取平均值作为最后结果，我们称这种方法为随机森林方法。随机森林是一种非线性建模工具，是目前数据挖掘、生物信息学等领域最热门的前沿研究方法之一。

随机森林（Random Forest）分类是由很多决策树分类模型 $\{h(\mathbf{X},\Theta_k),k=1,2,\cdots\}$ 组成的组合分类模型，且参数集 $\{\Theta_k\}$ 是独立同分布的随机向量，在给定自变量 $\mathbf{X}$ 下，每个决策树分类模型都有相应的投票权来选择最优的分类结果。

随机森林分类的基本思想是：首先利用 bootstrap 抽样从原始训练集抽取 k 个样本，且每

个样本的样本容量都与原始训练集一样；其次对 k 个样本分别建立 k 个决策树模型，得到 k 种分类结果；最后根据 k 种分类结果，对每个记录进行投票决定其最终分类，其示意图如图 2.11 所示。

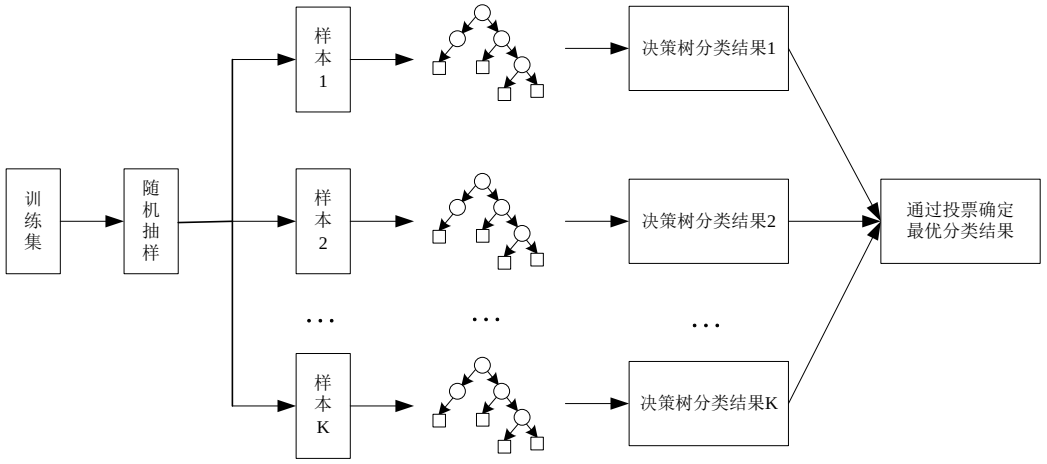

图 2.11　随机森林示意图

随机森林算法优点如下：

- 相对其他算法来说，评估效果良好。
- 能够处理高维度的数据，且不用做特征选择。
- 能够给出哪些特征比较重要。
- 模型泛化能力强。
- 训练速度快。
- 对于不平衡的数据集来说，可以平衡误差。
- 部分特征遗失的数据，仍可确保评估准确度。
- 能够解决分类与回归问题。

随机森林算法缺点如下：

- 小数据或者低维数据，评估效果可能比较差。
- 训练模型相对比较慢。
- 对参数选择敏感。
- 模型类似一个黑盒，评估结果难以解释。

2.6.5　人工神经网络算法

人工神经网络是由多个神经元互相连接而成，组成异常复杂的网络，形似一个网络。每个神经元有数值量的输入和输出，形式可以为实数或线性组合函数。

当网络判断错误时，通过学习使其减少犯同样错误的可能性。此方法有很强的泛化能力和非线性映射能力，可以对信息量少的数据进行模型处理。

人工神经网络算法优点如下：

- 非线性映射能力强，数学理论上证明三层的神经网络能够以任意精度逼近任何非线性连续函数。
- 自学习和自适应能力强。
- 模型泛化能力强。
- 模型具备一定的容错能力。

人工神经网络算法缺点如下：

- 局部极小化问题。
- 训练模型相对比较慢。
- 对网络结构选择敏感。
- 模型类似一个黑盒，评估结果难以解释。

下面给出一个神经网络模型示意图，如图 2.12 所示。

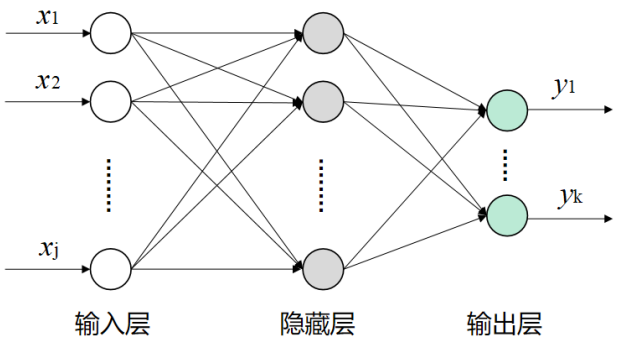

图 2.12　神经网络模型示意图

2.6.6　关联规则算法

关联规则是用规则去描述两个变量或多个变量之间的关系，是客观反映数据本身性质的方法。它是机器学习的一大类任务，可分为 2 个阶段，先从资料集中找到高频项目组，再去研究它们的关联规则。其得到的分析结果即是对变量间规律的总结。

其中经典的商业案例，就是啤酒和尿布的故事。经过对超市购物记录的统计分析，发现男士在购买尿布的时候，非常有可能会一同购买啤酒，虽然不知道具体原因，但是实际的统计结果肯定隐含着某种内在联系，因此可以优化货物摆放位置，将尿布和啤酒放一起进行摆放，后期发现二者的销量都有提升。关联规则示例如图 2.13 所示。

概率	重要性	规则
1.000	0.368	毛巾 -> 牙刷
1.000	0.368	毛巾，牙膏 -> 牙刷
1.000	0.192	毛巾 -> 牙膏
1.000	0.192	毛巾，牙刷 -> 牙膏
1.000	0.352	牙刷 -> 牙膏
1.000	0.192	记事本 -> 墨水
1.000	0.192	记事本，钢笔 -> 墨水
1.000	0.368	记事本，墨水 -> 钢笔
1.000	0.368	记事本 -> 钢笔
1.000	0.352	钢笔 -> 墨水
0.667	0.477	牙膏 -> 牙刷
0.667	0.477	墨水 -> 钢笔
0.500	0.477	牙刷，牙膏 -> 毛巾
0.500	0.477	钢笔 -> 记事本
0.500	0.477	牙刷 -> 毛巾
0.500	0.477	钢笔，墨水 -> 记事本
0.333	0.301	牙膏 -> 毛巾
0.333	0.301	墨水 -> 记事本

图 2.13　关联规则示例图

2.6.7　线性回归算法

线性回归（Linear Regression）算法，可以从最小二乘法开始讲起。最小二乘法是一种数学优化技术，它通过最小化误差的平方来寻找数据的最佳函数表达式。利用最小二乘法可以简便地求得未知的数据，并使这些求得的数据与实际数据之间误差的平方和最小。

线性回归算法能用一个方程式较为精确地描述数据之间的关系，因此可以非常方便地预测值。

它的优点如下：

- 算法简单，对于小规模数据来说，建立的数据关系很有效。
- 回归模型容易理解，结果具有很好的可解释性，有利于决策。
- 可用于分类和回归。

它的缺点如下：

- 对于非线性数据难以处理。
- 在高维度的数据上难以保证预测效果。

下面给出一个线性回归示意图，如图 2.14 所示。

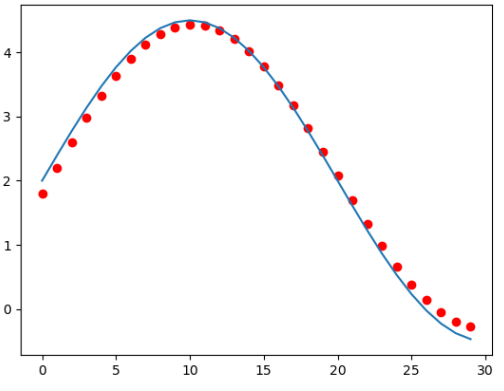

图 2.14　线性回归示意图

2.6.8　KNN 算法

KNN（K-Nearest Neighbor）算法，即 K 最近邻算法，是机器学习分类算法中最简单的方法之一。所谓K最近邻，说的是每个样本都可以用它最接近的 K 个邻近值来代表。

算法的描述如下：

（1）计算测试数据与各个训练数据之间的距离。
（2）按照距离的递增关系进行排序。
（3）选取距离最小的 K 个点。
（4）确定前 K 个点所在类别的出现频率。
（5）返回前 K 个点中出现频率最高的类别，作为测试数据的预测分类。

KNN 的主要优点有：

- 理论成熟，思想简单，既可以用来做分类，也可以用来做回归。
- 可用于非线性分类。
- 训练时间复杂度比支持向量机之类的算法低。
- 对数据没有假设，准确度高。
- 对异常点不敏感。

KNN 缺点如下：

- 参数K值需要预先设定，而不能自适应。

下面给出一个 KNN 算法示意图，如图 2.15 所示。

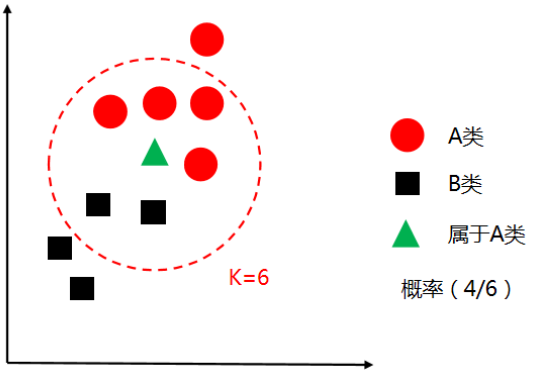

图 2.15　KNN 算法示意图

2.6.9　K-Means 算法

K-Means 算法，即 K 均值聚类算法，是一种迭代求解的聚类分析算法。其步骤是：预先将数据分为 K 组，则随机选取 K 个对象作为初始的聚类中心，然后计算每个对象与各个子聚类中心之间的距离，把每个对象分配给距离它最近的聚类中心。

聚类中心以及分配给它们的对象就代表一个聚类。每分配一个样本，聚类的聚类中心会根据聚类中现有的对象被重新计算。这个过程将不断重复，直到满足某个终止条件。

K-Means 的优点如下：

- 算法速度快。

- 算法简单，易于理解。

K-Means 的缺点如下：

- 分组K值需要给定。
- 对K值的选取比较敏感。

下面给出一个 K-Means 的算法示意图，如图 2.16 所示。

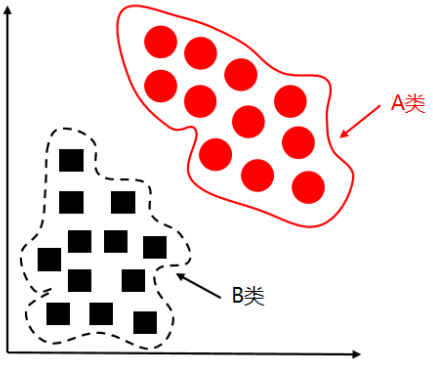

图 2.16　K-Means 的算法示意图

2.7　小结

本章对 Spark 相关概念和内部结构体系进行了介绍，并与 Hadoop 进行了对比。其中涉及 Spark 生态系统、Spark 运行架构和部署模式。

同时也对 Spark 中的 RDD 的基本操作进行了简要说明，阐述了 SQL 语句在 Spark 当中的支持，最后介绍了 Spark 机器学习相关的知识。