第5章 Spark

导学

Spark是一个围绕速度、易用性和复杂分析构建的大数据处理框架，在近两年内已发展成为大数据处理领域最炙手可热的开源项目。

了解：Spark的发展现状；Spark的应用场景与Spark的医学应用案例。

掌握：Spark的概念；Spark有哪些优点(对比Hadoop)；Spark速度比Hadoop快的原因；Spark生态系统的组成；Spark生态系统中的Runtime、Spark SQL、MLlib、GraphX、Spark Streaming的概念。

在大数据领域，Apache Spark(以下简称Spark)通用并行分布式计算框架越来越受人瞩目。Spark适合各种迭代算法和交互式数据分析，能够提升大数据处理的实时性和准确性，能够更快速地进行数据分析。

5.1 Spark平台

Spark和Hadoop都属于大数据的框架平台，而Spark是Hadoop的后继产品。由于Hadoop设计上只适合离线数据的计算以及在实时查询和迭代计算上的不足，已经不能满足日益增长的大数据业务需求。因而Spark应运而生，Spark具有可伸缩、在线处理、基于内存计算等特点，并可以直接读写Hadoop上任何格式的数据。

5.1.1 Spark简介

Spark是一个开源的通用并行分布式计算框架。2009年由加州大学伯克利分校的AMP实验室开发，是当前大数据领域最活跃的开源项目之一。Spark也称为快数据，与Hadoop的传统计算方式MapReduce相比，效率至少提高100倍。比如通过对比逻辑回归算法在Hadoop和Spark上的运行时间，可以看出Spark的效率有很大的提升，如图5-1所示。

Spark编程非常高效、简洁，支持多种语言的API，如Scala、Java、Python等。例如在基

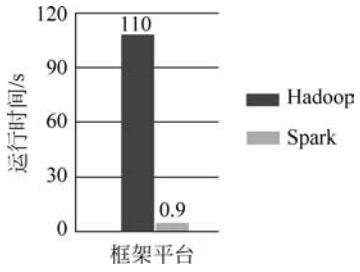


图 5-1 逻辑回归算法在 Hadoop 和 Spark 上的运行时间对比

于 MapReduce 开发的 WordCount 示例程序中，用户需要重写 Map 类和 Reduce 类，虽然 MapReduce 类似八股文的程序编写模式极大地简化了并行程序开发过程，但是程序代码至少几十行。若基于 Spark 开发同样的 WordCount 程序，仅需短短的几行代码，就可以对单词个数进行统计。

5.1.2 Spark 发展

Spark 的发展速度非常迅速。2009 年，Spark 诞生；2010 年，Spark 正式开源；2013 年成为 Apache 基金项目；2014 年成为 Apache 基金的顶级项目，整个过程不到五年时间。

从 2013 年 6 月到 2014 年 6 月，Spark 的开发人员从原来的 68 位增长到 255 位，参与开发的公司也从 17 家上升到 50 家。在这 50 家公司中，有来自中国的阿里巴巴、百度、网易、腾讯、搜狐等公司。当然，代码库的代码行也从原来的 63 000 行增加到 75 000 行。图 5-2 所示为截至 2014 年 Spark 的开发人员数量每年的增长曲线。

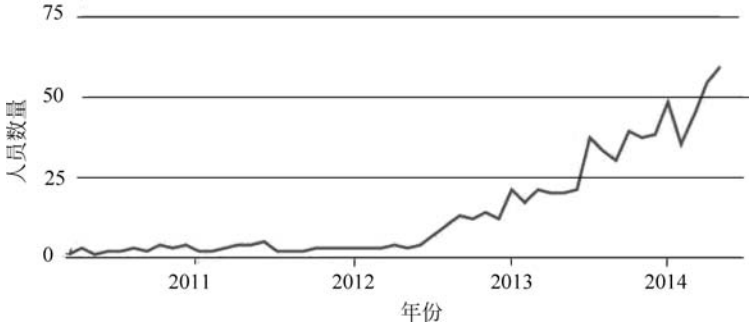


图 5-2 Spark 的开发人员数量每个月的增长曲线

Spark 广泛应用在国内外各大公司，比如国外的谷歌、亚马逊、雅虎、微软和国内的百度、腾讯、爱奇艺、阿里等公司。近两年，Spark 在中国的发展达到了一个前所未有的状态和高度。其中阿里巴巴的搜索和广告业务，最初使用 Mahout 和 MapReduce 来解决复杂的机器学习问题，但是在效率和代码维护方面并不理想，现已转向 Spark 框架。淘宝技术团队使用 Spark 实现了多次迭代的机器学习算法和一些高计算复杂度的算法，并将其运用在推荐系统上；同时还利用 Spark 中的一系列组件解决了基于最大连通图的社区发现、基于三角形计数的关系衡量、基于随机游走的用户属性传播等许多生产问题。此外，腾讯也是最早使用 Spark 的应用之一。借助 Spark 快速迭代的优势，腾讯提出了大数据精准推荐，并采用“数据-算法-系统”这套技术方案支持每天上百亿的请求量。随着各行业数据量的与日俱增，相信 Spark 会应用到越来越多的生产场景中去。

5.1.3 Spark 的优点

与 Hadoop 相比，Spark 真正的优势在于速度，除了速度之外，Spark 还有很多的优点，如表 5-1 所示。

表 5-1 Hadoop 与 Spark 的对比

对比项	Hadoop	Spark
工作方式	非在线、静态	在线、动态
处理速度	高延迟	比 Hadoop 快数十倍至上百倍
兼容性	开发语言：JAVA 语言 最好在 Linux 系统下搭建，对 Windows 的兼容性不好。	开发语言：以 Scala 为主的多语言 对 Linux 和 Windows 等操作系统的兼容性都非常好
存储方式	磁盘	既可以在内存上存储，也可以在磁盘上存储。
操作类型	只提供 Map 和 Reduce 两个操作，表达力欠缺。	提供很多转换和动作，很多基本操作如 Join、Group By 已经在 RDD 转换和动作中实现。
数据处理	只适用数据的批处理，实时处理非常差。	除了能够提供交互式实时查询外，还可以进行图处理、流式计算和反复迭代的机器学习等。
逻辑性	处理逻辑隐藏在代码细节中，没有整体逻辑。	代码不包含具体操作的实现细节，逻辑更清晰。
抽象层次	抽象层次低，需要手工编写代码来完成。	Spark 的 API 更强大，抽象层次更高。
可测试性	不容易	容易

5.1.4 Spark 速度比 Hadoop 快的原因

1. Hadoop 数据抽取运算模型

使用 Hadoop 处理一些问题诸如迭代式计算，每次对磁盘和网络的开销相当大。尤其每一次迭代计算都将结果写到磁盘以后再读回来，另外计算的中间结果还需要三个备份。Hadoop 中的数据传送与共享、串行方式、复制以及磁盘 I/O 等因素都使得 Hadoop 集群在低延迟、实时计算方面表现有待改进。Hadoop 的数据抽取运算模型如图 5-3 所示。

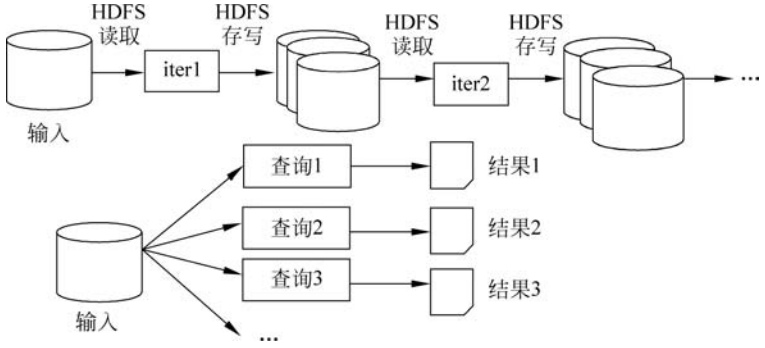


图 5-3 Hadoop 数据抽取运算模型

从图 5-3 中可以看出，Hadoop 中数据的抽取运算是基于磁盘的，中间结果也存储在磁盘上。所以，MapReduce 运算伴随着大量的磁盘的 I/O 操作，运算速度严重受到了限制。

2. Spark 数据抽取运算模型

Spark 使用内存(RAM)代替了传统 HDFS 存储中间结果，Spark 的数据抽取运算模型如图 5-4 所示。

从图 5-4 中可以看出，Spark 这种内存型计算框架比较适合各种迭代算法和交互式数

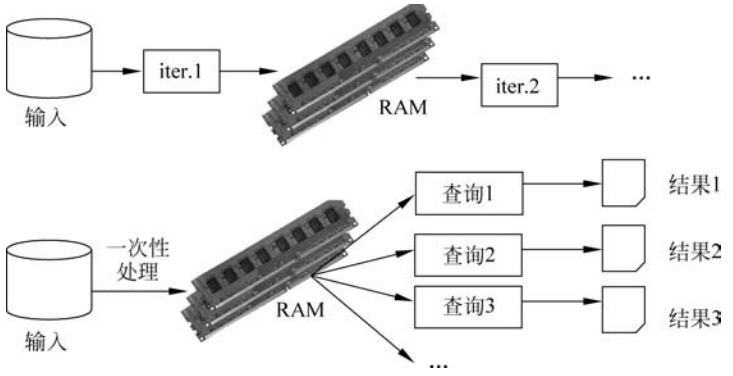


图 5-4 Spark 数据抽取运算模型

据分析。可每次将操作过程中的中间结果存入内存中，下次操作直接从内存中读取，省去了大量的磁盘 I/O 操作，效率也随之大幅提升。

5.2 Spark 生态系统

Spark 整个生态系统分为三层，如图 5-5 所示。

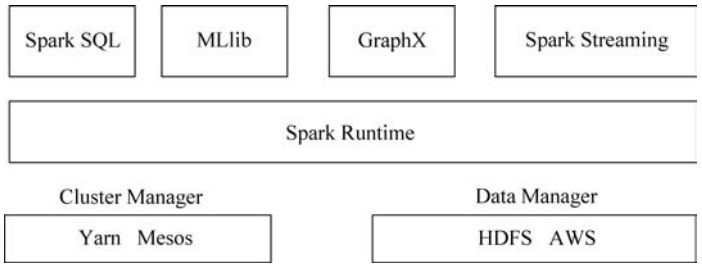


图 5-5 Spark 生态系统组成

从底向上分别为：

(1) 底层的 Cluster Manager 和 Data Manager：Cluster Manager 负责集群的资源管理；Data Manager 负责集群的数据管理。

(2) 中间层的 Spark Runtime，即 Spark 内核。它包括 Spark 的最基本、最核心的功能和基本分布式算子。

(3) 最上层为四个专门用于处理特定场景的 Spark 高层模块：Spark SQL、MLlib、GraphX 和 Spark Streaming，这四个模块基于 Spark RDD 进行了专门的封装和定制，可以无缝结合，互相配合。

5.2.1 底层的群集管理器和数据管理器

Cluster Manager 负责集群的资源管理；Data Manager 负责集群的数据管理。

1. 集群的资源管理可以选择 YARN、Mesos 等

Mesos 是 Apache 下的开源分布式资源管理框架，它被称为分布式系统的内核。Mesos

根据资源利用率和资源占用情况，在整个数据中心内进行任务的调度，提供类似于 YARN 的功能。Mesos 内核运行在每个机器上，可以通过数据中心和云环境向应用程序（Hadoop、Spark 等）提供资源管理和资源负载的 API 接口。

2. 集群的数据管理则可以选择 HDFS、AWS 等

Spark 支持两种分布式存储系统：HDFS 和 AWS。亚马逊云计算服务 AWS（Amazon Web Services，AWS）提供全球计算、存储、数据库、分析、应用程序和部署服务；AWS 提供的云服务中支持使用 Spark 集群进行大数据分析。Spark 对文件系统的读取和写入功能是 Spark 自己提供的，借助 Mesos 分布式实现。

5.2.2 中间层的 Spark Runtime

Spark Runtime 包含 Spark 的基本功能，这些功能主要包括任务调度、内存管理、故障恢复以及和存储系统的交互等。Spark 的一切操作都是基于 RDD 实现的，RDD 是 Spark 中最核心的模块和类，也是 Spark 设计的精华所在。

1. RDD 的概念

RDD（Resilient Distributed Datasets，RDD）即弹性分布式数据集，可以简单地把 RDD 理解成一个提供了许多操作接口的数据集合，和一般数据集不同的是，其实际数据分布存储在磁盘和内存中。

对开发者而言，RDD 可以看作是 Spark 中的一个对象，它本身运行于内存中。如读文件、写文件、数据之间的依赖、Key-Value 类型的 Map 数据都可以看作 RDD。RDD 是一个大的集合，将所有数据都加载到内存中，方便进行多次重用。

2. RDD 的操作类型

RDD 提供了丰富的编程接口来操作数据集合，一种是 Transformation 操作，另一种是 Action 操作。

（1）Transformation 的返回值是一个 RDD，如 Map、Filter、Union 等操作。它可以理解为一个领取任务的过程。如果只提交 Transformation 是不会提交任务来执行的，任务只有在 Action 提交时才会被触发。

（2）Action 返回的结果把 RDD 持久化起来，是一个真正触发执行的过程。它将规划以任务（Job）的形式提交给计算引擎，由计算引擎将其转换为多个 Task，然后分发到相应的计算结点，开始真正的处理过程。

Spark 的计算发生在 RDD 的 Action 操作，而对 Action 之前的所有 Transformation，Spark 只是记录下 RDD 生成的轨迹，而不会触发真正的计算。

Spark 内核会在需要计算发生的时刻绘制一张关于计算路径的有向无环图（Directed Acyclic Graph，简称 DAG）。举个例子，在图 5-6 中，从输入中逻辑上生成 A 和 C 两个

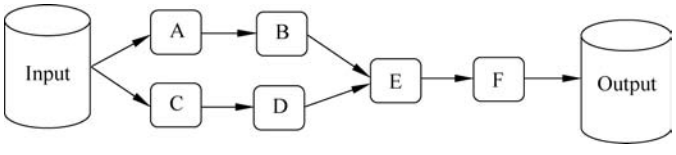


图 5-6 有向无环图 DAG 的生成

RDD，经过一系列 Transformation 操作，逻辑上生成了 F。注意，这时候计算没有发生，Spark 内核只是记录了 RDD 的生成和依赖关系。当 F 要进行输出（进行了 Action 操作）时，Spark 会根据 RDD 的依赖生成 DAG，并从起点开始真正的计算。

5.2.3 高层的应用模块

1. Spark SQL

Spark SQL 作为 Spark 大数据框架的一部分，主要用于结构化数据处理和对 Spark 数据执行类 SQL 的查询，并且与 Spark 生态的其他模块无缝结合。Spark SQL 兼容 SQL、Hive、JSON、JDBC 和 ODBC 等操作。Spark SQL 的前身是 Shark，而 Shark 的前身是 Hive。Shark 比 Hive 在性能上要高出一到两个数量级，而 Spark SQL 比 Shark 在性能上又要高出一到两个数量级。

2. MLlib

MLlib 是一个分布式机器学习库，即在 Spark 平台上对一些常用的机器学习算法进行了分布式实现，随着版本的更新，它也在不断扩充新的算法。MLlib 支持多种分布式机器学习算法，如分类、回归、聚类等。MLlib 已经实现的算法如表 5-2 所示。

表 5-2 MLlib 已经实现的算法

算　法	功　能
Classilication/Clustenng/Regressionilree	分类算法、回归算法、决策树、聚类算法
Optimization	核心算法的优化方法实现
Stat	基础统计
Feature	预处理
Evaluation	算法效果衡量
Linalg	基础线性代数运算支持
Recommendation	推荐算法

3. GraphX

GraphX 是构建于 Spark 上的图计算模型，GraphX 利用 Spark 框架提供的内存缓存 RDD、DAG 和基于数据依赖的容错等特性，实现高效健壮的图计算框架。GraphX 的出现，使得 Spark 生态系统在大图处理和计算领域得到了更加的完善和丰富，同时其与 Spark 生态系统其他组件进行很好的融合，以及强大的图数据处理能力，使其广泛地应用在多种大图处理的场景中。

GraphX 实现了很多能够在分布式集群上运行的并行图计算算法，而且还拥有丰富的 API 接口。因为图的规模大到一定的程度之后，需要将算法并行化，以方便其在分布式集群上进行大规模处理。GraphX 优势就是提升了数据处理的吞吐量和规模。

4. Spark Streaming

Spark Streaming 是 Spark 系统中用于实时处理流数据的分布式流处理框架，扩展了 Spark 流式大数据处理能力。流数据的例子有生产环境的 Web 服务器生成的日志文件，用户向一个 Web 服务请求包含状态更新的消息。

Spark Streaming 将数据流以时间片为单位进行分割形成 RDD，能够以相对较小的时间间隔对流数据进行处理。Spark Streaming 还能够和其余 Spark 生态的模块，如 Spark

SQL、GraphX、MLlib 等进行无缝的集成，以便联合完成基于实时流数据处理的复杂任务。

如果要用一句话来概括 Spark Streaming 的处理思路的话，那就是“将连续的数据持久化、离散化、然后进行批量处理”。

(1) 数据持久化

将从网络上接收到的数据先暂时存储下来，为事件处理出错时的事件重演提供可能。

(2) 数据离散化

数据源源不断的涌进，永远没有尽头。既然不能穷尽，那么就将其按时间分片。比如采用一分钟为时间间隔，那么在连续的一分钟内收集到的数据就集中存储在一起。

(3) 批量处理

将持久化下来的数据分批进行处理，处理机制套用之前的 RDD 模式。

5.3 Spark 的实现方法

5.3.1 Spark 环境配置

Spark 环境配置如图 5-7 和图 5-8 所示。

图 5-7 Spark 环境配置软件

```
@rem 环境变量的配置和初始化:
set JAVA_HOME=%~dp0%jdk1.8.0_231
set HADOOP_HOME=%~dp0%hadoop2.7.2
set SPARK_HOME=%~dp0spark-2.4.4-bin-hadoop2.7
set path=%~dp0python37-64;%path%
set pa=%~dp0%
```

图 5-8 Spark 环境变量配置

5.3.2 Spark 操作实例

测试文件 test.txt 内容，测试代码如图 5-9 所示。

```
These examples give a quick
overview of the Spark API.
Spark is built on the
concept of distributed datasets,
which contain arbitrary
Java or Python objects.
```

```
from __future__ import print_function
import sys
from operator import add
from pyspark.sql import SparkSession
if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: wordcount <file>", file=sys.stderr)
        exit(-1)
    spark = SparkSession.builder.appName("PythonWordCount").getOrCreate()
    lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
    counts = lines.flatMap(lambda x: x.split(' ')).map(lambda x: (x, 1)).reduceByKey(add)
    output = counts.collect()
    for (word, count) in output:
        print("%s: %i" % (word, count))
    spark.stop()
```

图 5-9 测试代码

5.4 Spark 在医学领域的应用

5.4.1 Spark 在医学领域的应用场景

Spark 能够满足医学信息处理中以交互式查询和迭代计算为代表的统计分析、数据挖掘、图形计算等各种数据处理需求，可用于临床转化医学研究、基于海量原始数据的实时卫生统计和辅助决策、文献挖掘、流行病预警和预测等。

Spark 可以解决医学大数据计算中的批处理、交互查询及流式计算等核心问题。Spark 的优势不仅体现性能的提升，Spark 框架还为批处理(Spark Core)、SQL 查询(Spark SQL)、流式计算(Spark Streaming)、机器学习(MLlib)、图计算(GraphX)等提供一个统一的数据处理平台，这相对于使用 Hadoop 有很大优势。表 5-3 列举了 Spark 在应用方面与其他大数据框架的对比。

表 5-3 Spark 的应用

应用场景	成熟的框架	Spark	时间对比
复杂的批量数据处理	MapReduce(Hive)	Spark Runtime	小时级，分钟级
基于历史数据的交互式查询	MapReduce	Spark SQL	分钟级，秒级
基于实时数据流的数据处理	Storm	Spark Streaming	秒级，秒级
基于历史数据的数据挖掘	Mahout	Spark MLlib	分钟级，秒级
基于增量数据的机器学习	无	Spark Streaming+ MLlib	分钟级
基于图计算的数据处理	无	Spark GraphX	分钟级

5.4.2 使用 Scala 语言开发 Spark 医学应用程序

本节将从实例出发，介绍如何使用 Scala 语言(Spark 框架的开发语言)开发 Spark 应用程序并且将其运行在 Spark 集群环境中。

ID	年龄
1	22
2	20
3	18
4	38
5	75
6	64
7	58
8	88
9	67
10	16
11	20
12	45

图 5-10 测试数据格式预览

医学应用案例描述：假设需要统计一个 1000 万患有糖尿病患者的平均年龄。假设这些年龄信息都存储在一个文件里，并且该文件的格式如下，第一列是 ID，第二列是年龄。测试数据格式预览如图 5-10 所示。

针对以上应用案例，使用 Scala 语言开发 Spark 应用程序，操作步骤如下：

(1) 在 Windows 环境下搭建 Spark 平台

Spark 平台的搭建主要包括四个步骤，分别是 JDK 的安装和配置、Scala 的安装、Intellij IDE 的安装和配置、Spark 的安装。详细的安装和调试方法可以参照网址：http://blog.csdn.net/ZHAOLEI5911/article/details/53138493。

(2) 启动 Intellij IDE(Scala 语言编辑器)，选择 Scala，新建工程，如图 5-11 所示。然后在工程目录下创建一个 lib 文件夹，并且把 Spark 安装包下的 spark-assembly.jar 包复制到 lib 目录下，如图 5-12 所示。

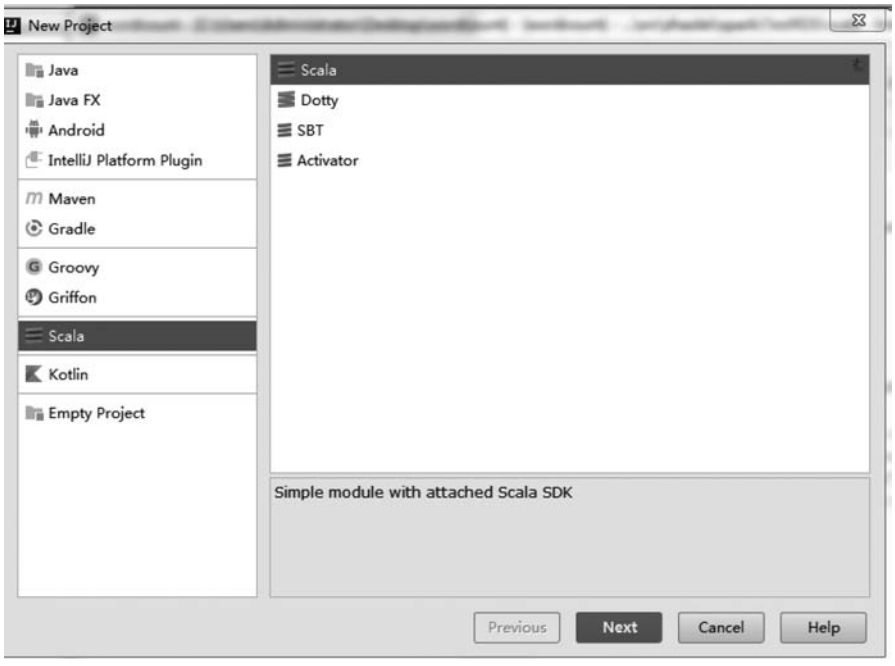
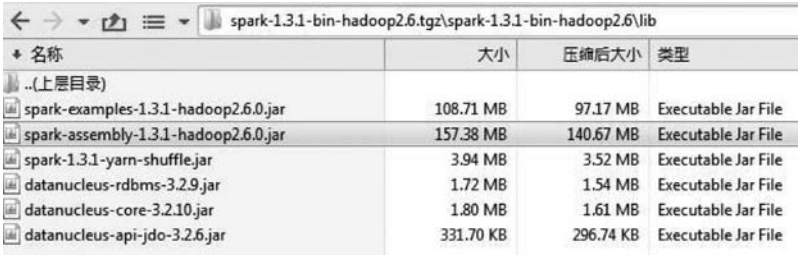


图 5-11 新建工程



名称	大小	压缩后大小	类型
..(上层目录)			
spark-examples-1.3.1-hadoop2.6.0.jar	108.71 MB	97.17 MB	Executable Jar File
spark-assembly-1.3.1-hadoop2.6.0.jar	157.38 MB	140.67 MB	Executable Jar File
spark-1.3.1-yarn-shuffle.jar	3.94 MB	3.52 MB	Executable Jar File
datanucleus-rdbms-3.2.9.jar	1.72 MB	1.54 MB	Executable Jar File
datanucleus-core-3.2.10.jar	1.80 MB	1.61 MB	Executable Jar File
datanucleus-api-jdo-3.2.6.jar	331.70 KB	296.74 KB	Executable Jar File

图 5-12 Spark 开发 jar 包

并且添加该 jar 包到工程的 classpath 中，并配置工程使用刚刚安装的 Scala2.10.4 版本，工程目录结构如图 5-13 所示。

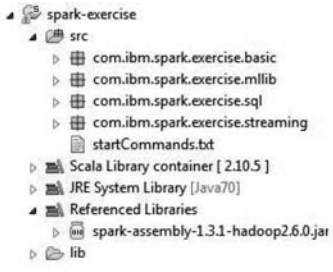


图 5-13 添加 jar 包到 classpath 中

(3) 用 Scala 语言编写一个生成 1000 万患者年龄数据的程序文件，源程序如图 5-14 所示。

(4) 案例分析与编程实现

案例分析：要计算 1000 万患者的平均年龄，那么首先需要对源文件对应的 RDD 进行处理，也就是将它转化成一个只包含年龄信息的 RDD，其次是计算元素个数即为总人数，然后把所有年龄数加起来即为总年龄，最后平均年龄＝总年龄/人数。对于第一步需要使用 Map 算子把源文件对应的 RDD 映射成一个新的只包含年龄数据的 RDD，需要对在 Map 算子的传入函数中使用 Split 方法，得到数组后只取第二个元素即

```
import java.io.FileWriter
import java.io.File
import scala.util.Random

object SampleDataFileGenerator {

def main(args:Array[String]) {
val writer = new FileWriter(new File("C: \\sample_age_data.txt"),false)
val rand = new Random()
for ( i <- 1 to 10000000) {
writer.write( i + " " + rand.nextInt(100))
writer.write(System.getProperty("line.separator"))
}
writer.flush()
writer.close()
}
}
```

图 5-14　年龄信息文件生成类源码

为年龄信息；第二步计算数据元素总数需要对于第一步映射的结果 RDD，使用 Count 算子；第三步则是使用 Reduce 算子对只包含年龄信息的 RDD 的所有元素用加法求和；最后使用除法计算平均年龄即可。

编程实现：由于本例输出结果很简单，所以在控制台输出即可，用 Scala 语言编写 AvgAgeCalculator 类来实现平均年龄的计算，源码如图 5-15 所示。

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
object AvgAgeCalculator {
 def main(args:Array[String]) {
 if (args.length < 1){
 println("Usage:AvgAgeCalculator datafile")
 System.exit(1)
 }
 val conf = new SparkConf().setAppName("Spark Exercise:Average Age Calculator")
 val sc = new SparkContext(conf)
 val dataFile = sc.textFile(args(0), 5);
 val count = dataFile.count()
 val ageData = dataFile.map(line => line.split(" ")(1))
 val totalAge = ageData.map(age => Integer.parseInt(
                                String.valueOf(age))).collect().reduce((a,b) => a+b)
 println("Total Age:" + totalAge + ";Number of People:" + count )
 val avgAge : Double = totalAge.toDouble / count.toDouble
 println("Average Age is " + avgAge)
 }
}
```

图 5-15　AvgAgeCalculator 类源码

(5) 提交到集群执行

要执行本实例的程序，需要将刚刚生成的年龄信息文件上传到 HDFS 上，假设刚才已经在目标机器上执行生成年龄信息文件的 Scala 类，并且文件被生成到了/home/fams 目录下。需要运行一下 HDFS 命令把文件复制到 HDFS 的/user/fams 目录下。

将年龄信息文件复制到 HDFS 目录下的命令为：

```
hdfs dfs -copyFromLocal /home/fams /user/fams.
```

在控制台执行 AvgAgeCalculator 类的命令如图 5-16 所示。

```
./spark-submit \
--class com.ibm.spark.exercise.basic.AvgAgeCalculator \
--master spark://hadoop036166:7077 \
--num-executors 3 \
--driver-memory 6g \
--executor-memory 2g \
--executor-cores 2 \
/home/fams/sparkexercise.jar \
hdfs://hadoop036166:9000/user/fams/inputfiles/sample_age_data.txt
```

图 5-16　执行 AvgAgeCalculator 类的命令

执行完 AvgAgeCalculator 类的命令后，就可以在控制台看到程序的输出结果，如图 5-17 所示。

```
Total Age:494920921;Number of People:10000000
Average Age is 49.4920921_
```

图 5-17 程序的输出结果

通过本例题，目的是让大家对如何使用 Scala 语言编写 Spark 应用程序进行简单的了解。当然在处理实际问题时，情况可能比本例题复杂很多，但是解决问题的基本思想是一致的。在碰到实际问题的时候，首先要对源数据结构格式等进行分析，然后确定如何去使用 Spark 提供的算子对数据进行转化，最终根据实际需求选择合适的算法操作数据并计算结果。

本章小结

Spark 作为一个开源的大数据处理平台，以其内存计算、可伸缩及高效的容错特性，与分布式文件存储系统、分布式数据库结合使用，配合其丰富的生态系统，解决了数据增长和处理性能需求之间存在的瓶颈问题。可以预见 Spark 在医学领域中具有广阔的应用前景，将在医学大数据处理分析中得到更广泛和深入的应用。

【关键词注释】

1. 迭代：是重复反馈过程的活动，其目的通常是为了逼近所需目标或结果。每一次对过程的重复称为一次“迭代”，而每一次迭代得到的结果会作为下一次迭代的初始值。

2. 流数据：流数据是一组顺序、大量、快速、连续到达的数据序列。一般情况下，数据流可被视为一个随时间延续而无限增长的动态数据集合。应用于网络监控、传感器网络、航空航天、气象测控和金融服务等领域。

3. R 语言：用于统计分析、绘图的语言和操作环境。R 语言是一个自由、免费、源代码开放的软件，它是一个用于统计计算和统计制图的优秀工具。

4. 逻辑回归：是一种广义的线性回归分析模型，常用于数据挖掘、疾病自动诊断、经济预测等领域。例如探讨引发疾病的危险因素，并根据危险因素预测疾病发生的概率等。

5. Python 语言：是一种面向对象、解释型计算机程序设计语言。Python 具有丰富和强大的库。它常被昵称为胶水语言，能够把用其他语言制作的各种模块(尤其是 C/C++)很轻松地联结在一起。

6. JSON：(JavaScript Object Notation，JSON)是一种轻量级的数据交换格式。JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯(包括 C、C++、C#、Java、JavaScript、Perl、Python 等)。这些特性使 JSON 成为理想的数据交换语言。

7. JDBC：(Java Data Base Connectivity，Java 数据库连接)是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一访问，它由一组用 Java 语言编写的类和接口组成。

8. ODBC：(Open Database Connectivity，开放数据库连接)是微软公司开放服务结构中有关数据库的一个组成部分，它建立了一组规范，并提供了一组对数据库访问的标准 API。这些 API 利用 SQL 来完成其大部分任务。ODBC 本身也提供了对 SQL 语言的支持，用户可以直接将 SQL 语句送给 ODBC。

9. 基于最大连通图：把图的所有结点用最少的边将其连接起来的子图，所以极大连通子图不为 1，因此可以说最大连通子图是一个累赘概念，因为任何一个极大连通子图，其实都可以叫作最大连通子图。

10. 持久化：把数据(如内存中的对象)保存到可永久保存的存储设备中(如磁盘)。持久化的主要应用是将内存中的对象存储在数据库中，或者存储在磁盘文件、XML 数据文件中，等等。

11. iter：迭代器(iterator)有时又称游标(cursor)，是程序设计的软件设计模式，可在容器(container，例如链表或阵列)上遍访的接口，设计人员无须关心容器的内容。

12. 算子：是一个函数空间到函数空间上的映射 O：X→X。广义地讲，对任何函数进行某一项操作都可以认为是一个算子，甚至包括求幂次、开方都可以认为是一个算子，只是有的算子用了一个符号来代替它所要进行的运算罢了，它和 f(x)的 f 没区别，它甚至和加减乘除的基本运算符号都没有区别，只是它可以对单对象操作罢了(有的符号比如大于、小于号要对多对象操作)。

习题 5

一、填空题

1. Spark 大数据框架适合各种_____算法和交互式数据分析，能够提升大数据处理的实时性和准确性。

2. _____也称为快数据，与 Hadoop 的传统计算方式 MapReduce 相比，效率至少提高 100 倍。

3. _____语言是 Spark 框架的开发语言，是一种类似 Java 的编程语言。

4. Spark 是当前流行的_____大数据处理框架，具有快速、通用、简单等特点。

5. 与 Hadoop 相比，Spark 真正的优势在于_____。

6. Spark 使用_____代替了传统 HDFS 存储中间结果。

7. Spark 整个生态系统分为三层，底层的_____负责集群的资源管理。

8. Spark 整个生态系统分为三层，底层的_____负责集群的数据管理。

9. Spark 整个生态系统分为三层，中间层的_____包括 Spark 的最基本、最核心的功能和基本分布式算子。

10. RDD(Resilient Distributed Datasets，弹性分布式数据集)即_____。

11. 对开发者而言，_____可以看作是 Spark 中的一个对象，它本身运行于内存中。它是一个大的集合，将所有数据都加载到内存中，方便进行多次重用。

12. RDD 提供了丰富的编程接口来操作数据集合，一种是_____操作，另一种是 Action 操作。

13. RDD 的_____操作返回的结果把 RDD 持久化起来，是一个真正触发执行的过程。

14. Spark 内核会在需要计算发生的时刻绘制一张关于计算路径的_____，简称 DAG。

15. _____作为 Spark 大数据框架的一部分，主要用于结构化数据处理和对 Spark 数据执行类 SQL 的查询。

16. _____是一个分布式机器学习库，即在 Spark 平台上对一些常用的机器学习算法进行了分布式实现。

17. _____是构建于 Spark 上的图计算模型，它利用 Spark 框架提供的内存缓存 RDD、DAG 和基于数据依赖的容错等特性，实现高效健壮的图计算框架。

18. _____是 Spark 系统中用于处理流数据的分布式流处理框架，扩展了 Spark 流式大数据处理能力。

19. Spark Streaming 将数据流以时间片为单位进行分割形成_____，能够以相对较小的时间间隔对流数据进行处理。

20. Spark Streaming 还能够和其余 Spark 生态的模块进行无缝的集成，以便联合完成基于_____处理的复杂任务。

二、简答题

1. 简述什么是 Spark。

2. 与 Hadoop 进行比较，Spark 在工作方式、处理速度、存储方式和兼容性等方面有哪些优点。

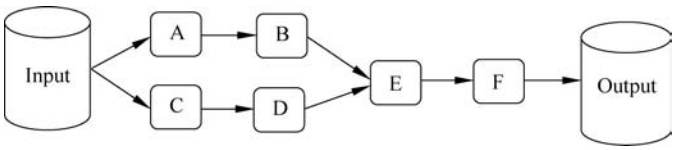
3. 从数据抽取运算模型进行分解，分析 Spark 速度比 Hadoop 快的原因。

4. 简述 Spark 整个生态系统分为哪三层。

5. 简述什么是 RDD。

6. 简述什么是 RDD 的 Transformation 操作和 Action 操作。

7. 通过下面的图，简述什么是 DAG，DAG 是如何生成的。



8. 简述什么是 Spark SQL

9. 简述什么是 GraphX。

10. 简述什么是 Spark Streaming 的数据持久化、离散化和批量处理。