《大数据技术原理与应用》

http://www.icourse163.org/course/XMU-1002335004

中国大学MOOC　2017年秋季学期

第10讲 Spark

林子雨

厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

主页：http://www.cs.xmu.edu.cn/linziyu

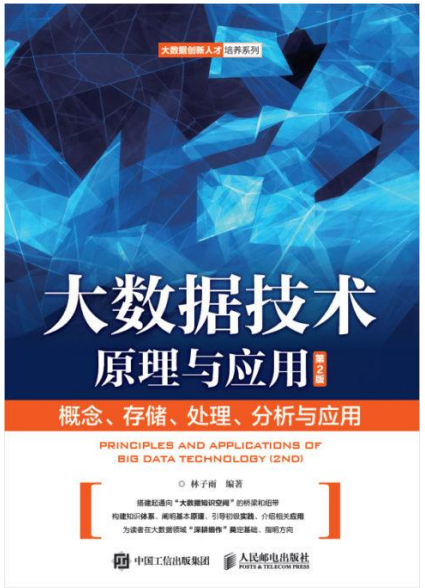
中国大学MOOC《大数据技术原理与应用》课程地址:
http://www.icourse163.org/course/XMU-1002335004

欢迎访问教材官网获取教学资源

《大数据技术原理与应用——大数据概念、存储、处理、分析与应用》

教材官网：http://dblab.xmu.edu.cn/post/bigdata

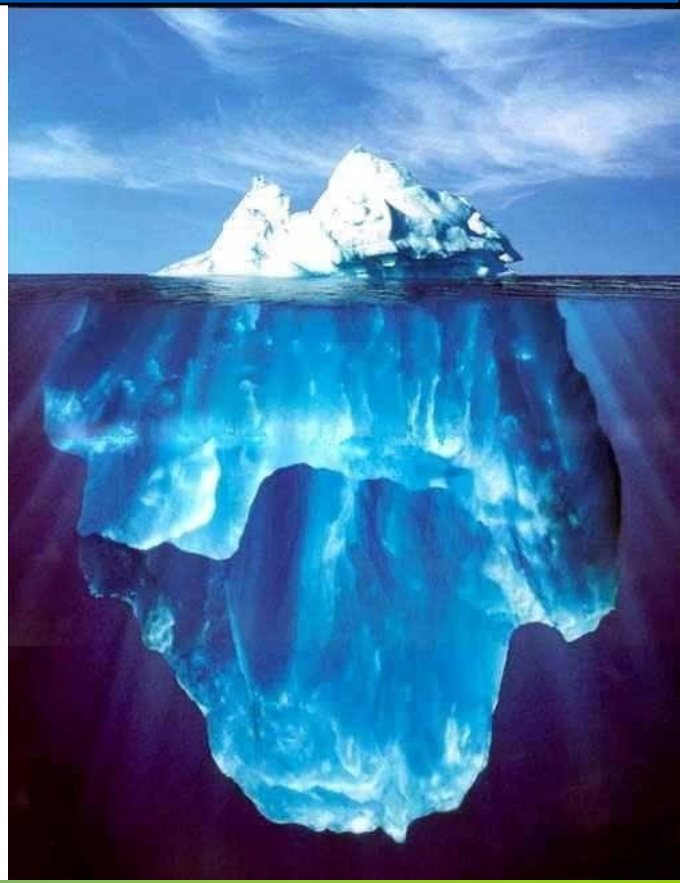
厦门大学 林子雨编著，人民邮电出版社，2017年1月第2版
ISBN:978-7-115-44330-4



- 国内高校第一本系统介绍大数据知识专业教材
- 京东、当当等各大网店畅销书籍
- 大数据入门教材精品
- 国内多所高校采用本教材开课
- 配套目前国内高校最完备的课程公共服务平台
- 福建省精品在线开放课程

提纲

- **10.1 Spark概述**
- **10.2 Spark生态系统**
- **10.3 Spark运行架构**
- **10.4 Spark SQL**
- **10.5 Spark的部署和应用方式**
- **10.6 Spark编程实践**



本PPT是如下教材的配套讲义：
《大数据技术原理与应用——概念、存储、处理、分析与应用》（2017年1月第2版）
厦门大学 林子雨 编著，人民邮电出版社
ISBN:978-7-115-44330-4

欢迎访问《大数据技术原理与应用》教材官方网站，免费获取教材配套资源：
http://dblab.xmu.edu.cn/post/bigdata

10.1 Spark概述

10.1.1 Spark简介
10.1.2 Scala简介
10.1.3 Spark与Hadoop的比较

10.1.1 Spark简介

•Spark最初由美国加州伯克利大学（UCBerkeley）的AMP实验室于2009年开发，是基于内存计算的大数据并行计算框架，可用于构建大型的、低延迟的数据分析应用程序

•2013年Spark加入Apache孵化器项目后发展迅猛，如今已成为Apache软件基金会最重要的三大分布式计算系统开源项目之一（Hadoop、Spark、Storm）

•Spark在2014年打破了Hadoop保持的基准排序纪录
- •Spark/206个节点/23分钟/100TB数据
- •Hadoop/2000个节点/72分钟/100TB数据
- •Spark用十分之一的计算资源，获得了比Hadoop快3倍的速度

10.1.1 Spark简介

Spark具有如下几个主要特点：

•运行速度快：使用DAG执行引擎以支持循环数据流与内存计算
•容易使用：支持使用Scala、Java、Python和R语言进行编程，可以通过Spark Shell进行交互式编程
•通用性：Spark提供了完整而强大的技术栈，包括SQL查询、流式计算、机器学习和图算法组件
•运行模式多样：可运行于独立的集群模式中，可运行于Hadoop中，也可运行于Amazon EC2等云环境中，并且可以访问HDFS、Cassandra、HBase、Hive等多种数据源

10.1.1 Spark简介

Spark如今已吸引了国内外各大公司的注意，如腾讯、淘宝、百度、亚马逊等公司均不同程度地使用了Spark来构建大数据分析应用，并应用到实际的生产环境中

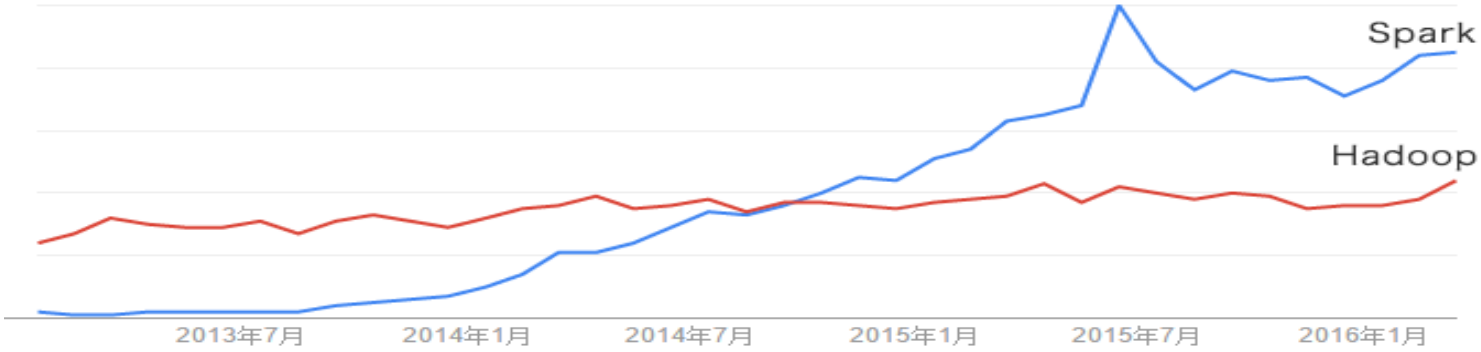


图10-1 谷歌趋势：Spark与Hadoop对比

10.1.2 Scala简介

Scala是一门现代的多范式编程语言，运行于Java平台（JVM，Java 虚拟机），并兼容现有的Java程序

Scala的特性：

•Scala具备强大的并发性，支持函数式编程，可以更好地支持分布式系统

•Scala语法简洁，能提供优雅的API

Scala兼容Java，运行速度快，且能融合到Hadoop生态圈中

Scala是Spark的主要编程语言，但Spark还支持Java、Python、R作为编程语言

Scala的优势是提供了REPL（Read-Eval-Print Loop，交互式解释器），提高程序开发效率

10.1.3 Spark与Hadoop的对比

Hadoop存在如下一些缺点：

- 表达能力有限
- 磁盘IO开销大
- 延迟高
  - 任务之间的衔接涉及IO开销
  - 在前一个任务执行完成之前，其他任务就无法开始，难以胜任复杂、多阶段的计算任务

10.1.3 Spark与Hadoop的对比

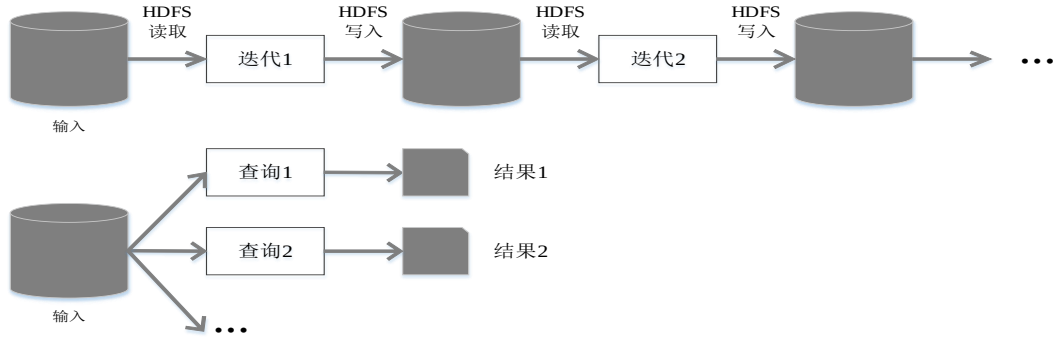
Spark在借鉴Hadoop MapReduce优点的同时，很好地解决了MapReduce所面临的问题

相比于Hadoop MapReduce，Spark主要具有如下优点：

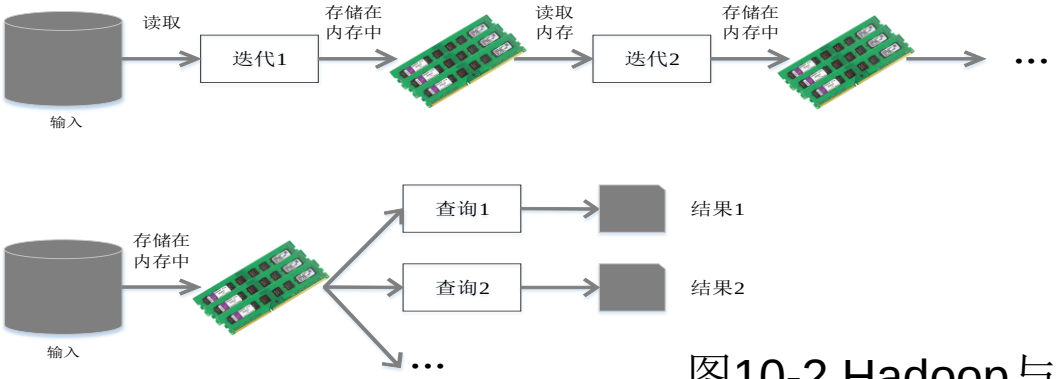
- Spark的计算模式也属于MapReduce，但不局限于Map和Reduce操作，还提供了多种数据集操作类型，编程模型比Hadoop MapReduce更灵活
- Spark提供了内存计算，可将中间结果放到内存中，对于迭代运算效率更高

Spark基于DAG的任务调度执行机制，要优于Hadoop MapReduce的迭代执行机制

10.1.3 Spark与Hadoop的对比



(a) Hadoop MapReduce执行流程



(b) Spark执行流程

图10-2 Hadoop与Spark的执行流程对比

10.1.3 Spark与Hadoop的对比

- 使用Hadoop进行迭代计算非常耗资源
- Spark将数据载入内存后，之后的迭代计算都可以直接使用内存中的中间结果作运算，避免了从磁盘中频繁读取数据

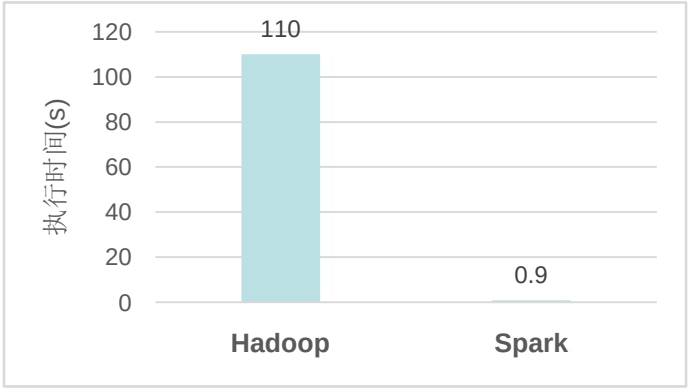


图10-3 Hadoop与Spark执行逻辑回归的时间对比

10.2 Spark生态系统

在实际应用中，大数据处理主要包括以下三个类型：

- 复杂的批量数据处理：通常时间跨度在数十分钟到数小时之间
- 基于历史数据的交互式查询：通常时间跨度在数十秒到数分钟之间
- 基于实时数据流的数据处理：通常时间跨度在数百毫秒到数秒之间

当同时存在以上三种场景时，就需要同时部署三种不同的软件

- 比如: MapReduce / Impala / Storm

这样做难免会带来一些问题：

- 不同场景之间输入输出数据无法做到无缝共享，通常需要进行数据格式的转换
- 不同的软件需要不同的开发和维护团队，带来了较高的使用成本
- 比较难以对同一个集群中的各个系统进行统一的资源协调和分配

10.2 Spark生态系统

- Spark的设计遵循“一个软件栈满足不同应用场景”的理念，逐渐形成了一套完整的生态系统
- 既能够提供内存计算框架，也可以支持SQL即席查询、实时流式计算、机器学习和图计算等
- Spark可以部署在资源管理器YARN之上，提供一站式的大数据解决方案
- 因此，Spark所提供的生态系统足以应对上述三种场景，即同时支持批处理、交互式查询和流数据处理

10.2 Spark生态系统

Spark生态系统已经成为伯克利数据分析软件栈BDAS（Berkeley Data Analytics Stack）的重要组成部分

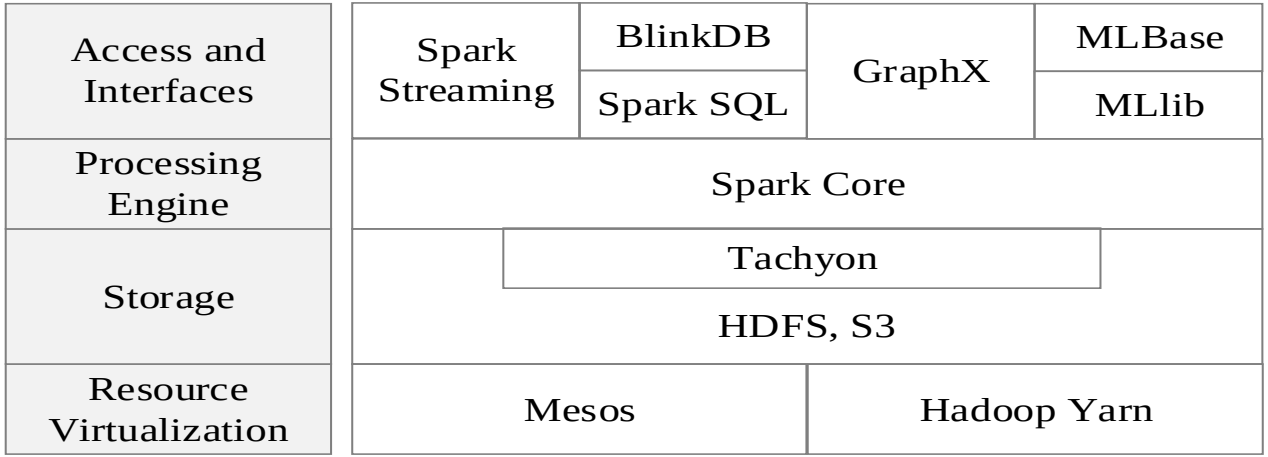


图10-4 BDAS架构

Spark的生态系统主要包含了Spark Core、Spark SQL、Spark Streaming、MLLib和GraphX 等组件

10.2 Spark生态系统

表1 Spark生态系统组件的应用场景

应用场景	时间跨度	其他框架	Spark生态系统中的组件
复杂的批量数据处理	小时级	MapReduce、Hive	Spark
基于历史数据的交互式查询	分钟级、秒级	Impala、Dremel、Drill	Spark SQL
基于实时数据流的数据处理	毫秒、秒级	Storm、S4	Spark Streaming
基于历史数据的数据挖掘	-	Mahout	MLlib
图结构数据的处理	-	Pregel、Hama	GraphX

10.3 Spark运行架构

10.3.1 基本概念
10.3.2 架构设计
10.3.3 Spark运行基本流程
10.3.4 Spark运行原理

10.3.1 基本概念

- RDD：是Resillient Distributed Dataset（弹性分布式数据集）的简称，是分布式内存的一个抽象概念，提供了一种高度受限的共享内存模型
- DAG：是Directed Acyclic Graph（有向无环图）的简称，反映RDD之间的依赖关系
- Executor：是运行在工作节点（WorkerNode）的一个进程，负责运行Task
- Application：用户编写的Spark应用程序
- Task：运行在Executor上的工作单元
- Job：一个Job包含多个RDD及作用于相应RDD上的各种操作
- Stage：是Job的基本调度单位，一个Job会分为多组Task，每组Task被称为Stage，或者也被称为TaskSet，代表了一组关联的、相互之间没有Shuffle依赖关系的任务组成的任务集

10.3.2 架构设计

•Spark运行架构包括集群资源管理器（Cluster Manager）、运行作业任务的工作节点（Worker Node）、每个应用的任务控制节点（Driver）和每个工作节点上负责具体任务的执行进程（Executor）

•资源管理器可以自带或Mesos或YARN

与Hadoop MapReduce计算框架相比，Spark所采用的Executor有两个优点：

•一是利用多线程来执行具体的任务，减少任务的启动开销

•二是Executor中有一个BlockManager存储模块，会将内存和磁盘共同作为存储设备，有效减少IO开销

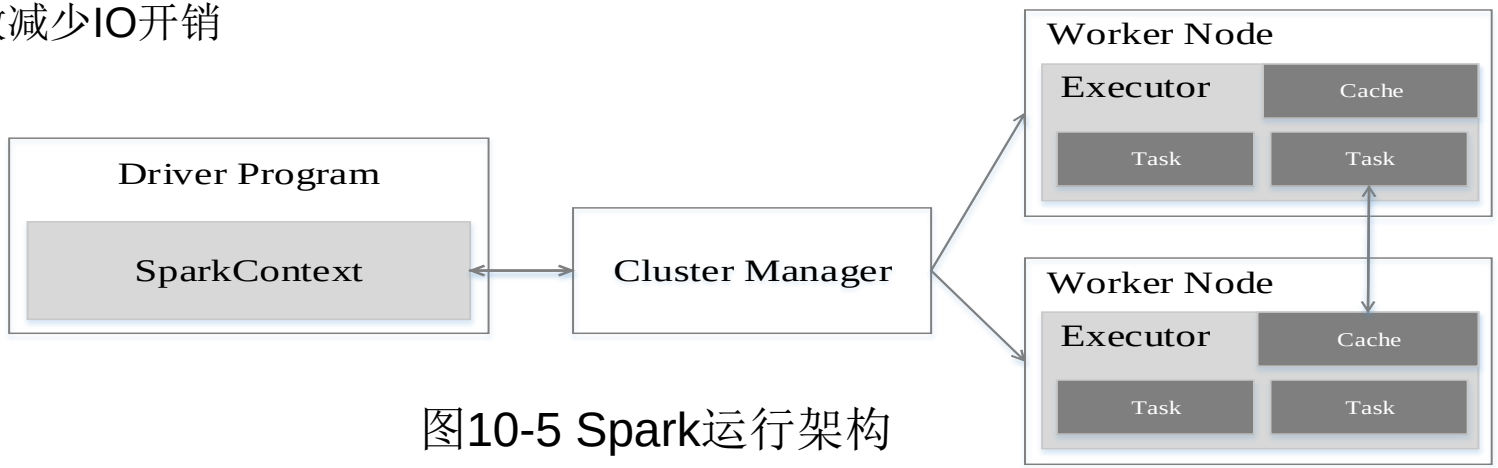


图10-5 Spark运行架构

10.3.2 架构设计

•一个Application由一个Driver和若干个Job构成，一个Job由多个Stage构成，一个Stage由多个没有Shuffle关系的Task组成

•当执行一个Application时，Driver会向集群管理器申请资源，启动Executor，并向Executor发送应用程序代码和文件，然后在Executor上执行Task，运行结束后，执行结果会返回给Driver，或者写到HDFS或者其他数据库中

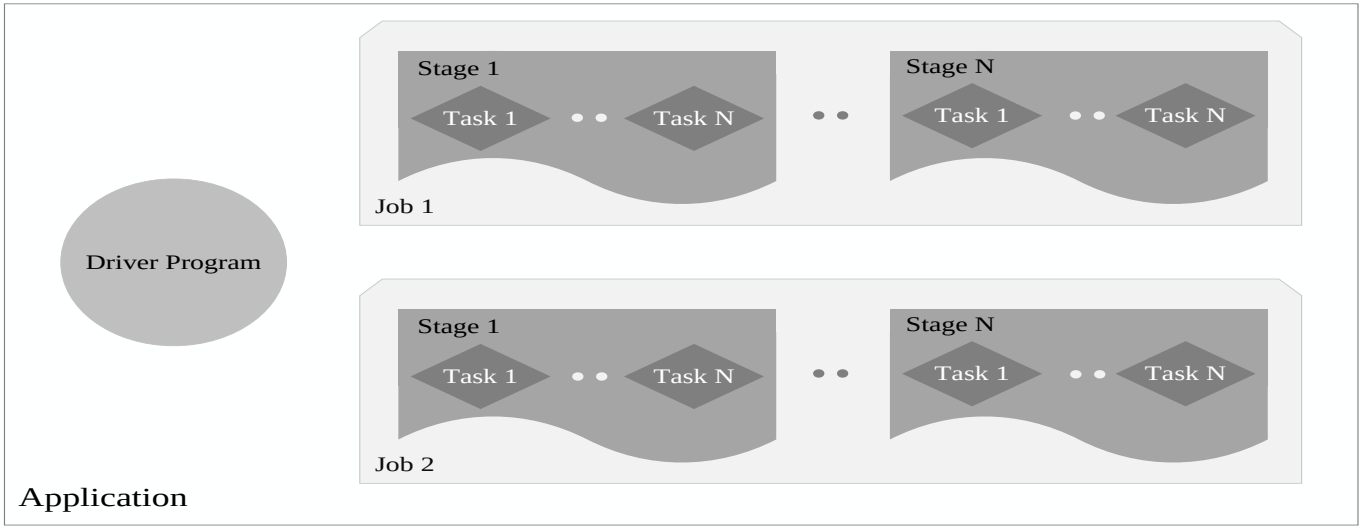


图10-6 Spark中各种概念之间的相互关系

10.3.3 Spark运行基本流程

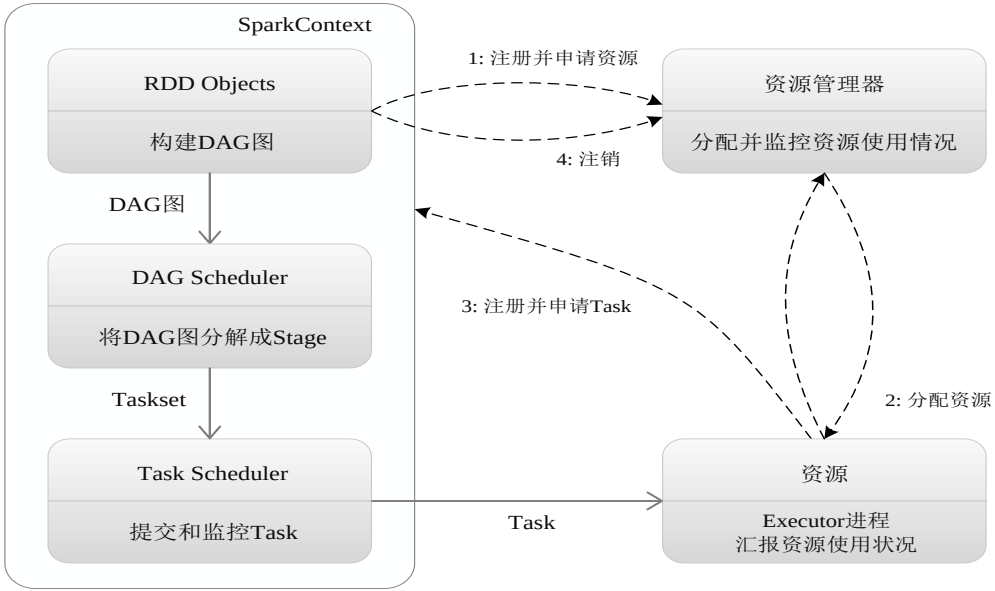


图10-7 Spark运行基本流程图

（1）首先为应用构建起基本的运行环境，即由Driver创建一个SparkContext，进行资源的申请、任务的分配和监控

（2）资源管理器为Executor分配资源，并启动Executor进程

（3）SparkContext根据RDD的依赖关系构建DAG图，DAG图提交给DAGScheduler解析成Stage，然后把一个个TaskSet提交给底层调度器TaskScheduler处理；Executor向SparkContext申请Task，Task Scheduler将Task发放给Executor运行，并提供应用程序代码

（4）Task在Executor上运行，把执行结果反馈给TaskScheduler，然后反馈给DAGScheduler，运行完毕后写入数据并释放所有资源

10.3.3 Spark运行基本流程

总体而言，Spark运行架构具有以下特点：

（1）每个Application都有自己专属的Executor进程，并且该进程在Application运行期间一直驻留。Executor进程以多线程的方式运行Task

（2）Spark运行过程与资源管理器无关，只要能够获取Executor进程并保持通信即可

（3）Task采用了数据本地性和推测执行等优化机制

10.3.4 RDD运行原理

1.设计背景
2.RDD概念
3.RDD特性
4.RDD之间的依赖关系
5.Stage的划分
6.RDD运行过程

10.3.4 RDD运行原理

1.设计背景

- 许多迭代式算法（比如机器学习、图算法等）和交互式数据挖掘工具，共同之处是，不同计算阶段之间会重用中间结果
- 目前的MapReduce框架都是把中间结果写入到HDFS中，带来了大量的数据复制、磁盘IO和序列化开销
- RDD就是为了满足这种需求而出现的，它提供了一个抽象的数据架构，我们不必担心底层数据的分布式特性，只需将具体的应用逻辑表达为一系列转换处理，不同RDD之间的转换操作形成依赖关系，可以实现管道化，避免中间数据存储

10.3.4 RDD运行原理

2.RDD概念

•一个RDD就是一个分布式对象集合，本质上是一个只读的分区记录集合，每个RDD可分成多个分区，每个分区就是一个数据集片段，并且一个RDD的不同分区可以被保存到集群中不同的节点上，从而可以在集群中的不同节点上进行并行计算

•RDD提供了一种高度受限的共享内存模型，即RDD是只读的记录分区的集合，不能直接修改，只能基于稳定的物理存储中的数据集创建RDD，或者通过在其他RDD上执行确定的转换操作（如map、join和group by）而创建得到新的RDD

10.3.4 RDD运行原理

- RDD提供了一组丰富的操作以支持常见的数据运算，分为“动作”（Action）和“转换”（Transformation）两种类型
- RDD提供的转换接口都非常简单，都是类似map、filter、groupBy、join等粗粒度的数据转换操作，而不是针对某个数据项的细粒度修改（不适合网页爬虫）
- 表面上RDD的功能很受限、不够强大，实际上RDD已经被实践证明可以高效地表达许多框架的编程模型（比如MapReduce、SQL、Pregel）
- Spark用Scala语言实现了RDD的API，程序员可以通过调用API实现对RDD的各种操作

10.3.4 RDD运行原理

RDD典型的执行过程如下：

•RDD读入外部数据源进行创建

•RDD经过一系列的转换（Transformation）操作，每一次都会产生不同的RDD，供给下一个转换操作使用

•最后一个RDD经过“动作”操作进行转换，并输出到外部数据源

这一系列处理称为一个Lineage（血缘关系），即DAG拓扑排序的结果

优点：惰性调用、管道化、避免同步等待、不需要保存中间结果、每次操作变得简单

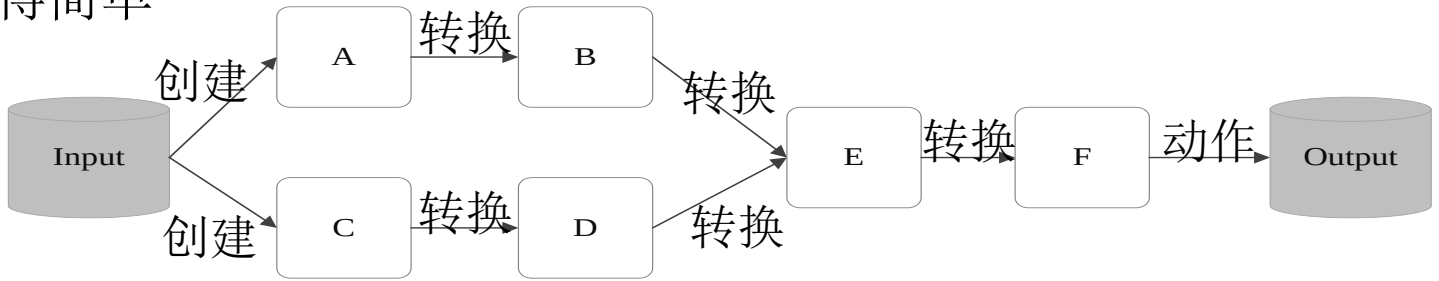


图10-8 RDD执行过程的一个实例

10.3.4 RDD运行原理

3.RDD特性

Spark采用RDD以后能够实现高效计算的原因主要在于：

（1）高效的容错性

- 现有容错机制：数据复制或者记录日志
- RDD：血缘关系、重新计算丢失分区、无需回滚系统、重算过程在不同节点之间并行、只记录粗粒度的操作

（2）中间结果持久化到内存，数据在内存中的多个RDD操作之间进行传递，避免了不必要的读写磁盘开销

（3）存放的数据可以是Java对象，避免了不必要的对象序列化和反序列化

10.3.4 RDD运行原理

4. RDD之间的依赖关系

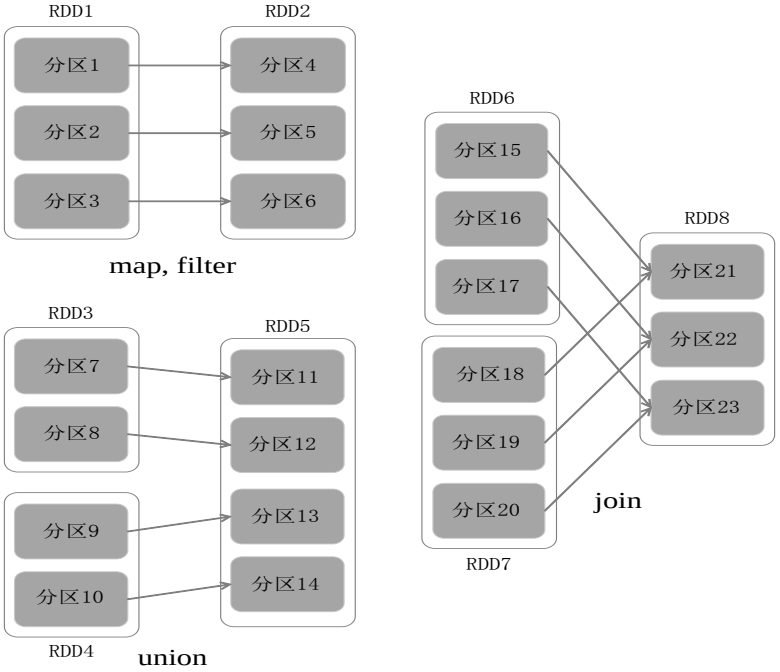


图10-9 窄依赖与宽依赖的区别

- 窄依赖表现为一个父RDD的分区对应于一个子RDD的分区或多个父RDD的分区对应于一个子RDD的分区
- 宽依赖则表现为存在一个父RDD的一个分区对应一个子RDD的多个分区

5.Stage的划分

Spark通过分析各个RDD的依赖关系生成了DAG，再通过分析各个RDD中的分区之间的依赖关系来决定如何划分Stage，具体划分方法是：

- 在DAG中进行反向解析，遇到宽依赖就断开
- 遇到窄依赖就把当前的RDD加入到Stage中
- 将窄依赖尽量划分在同一个Stage中，可以实现流水线计算

10.3.4 RDD运行原理

5.Stage的划分

被分成三个Stage，在Stage2中，从map到union都是窄依赖，这两步操作可以形成一个流水线操作

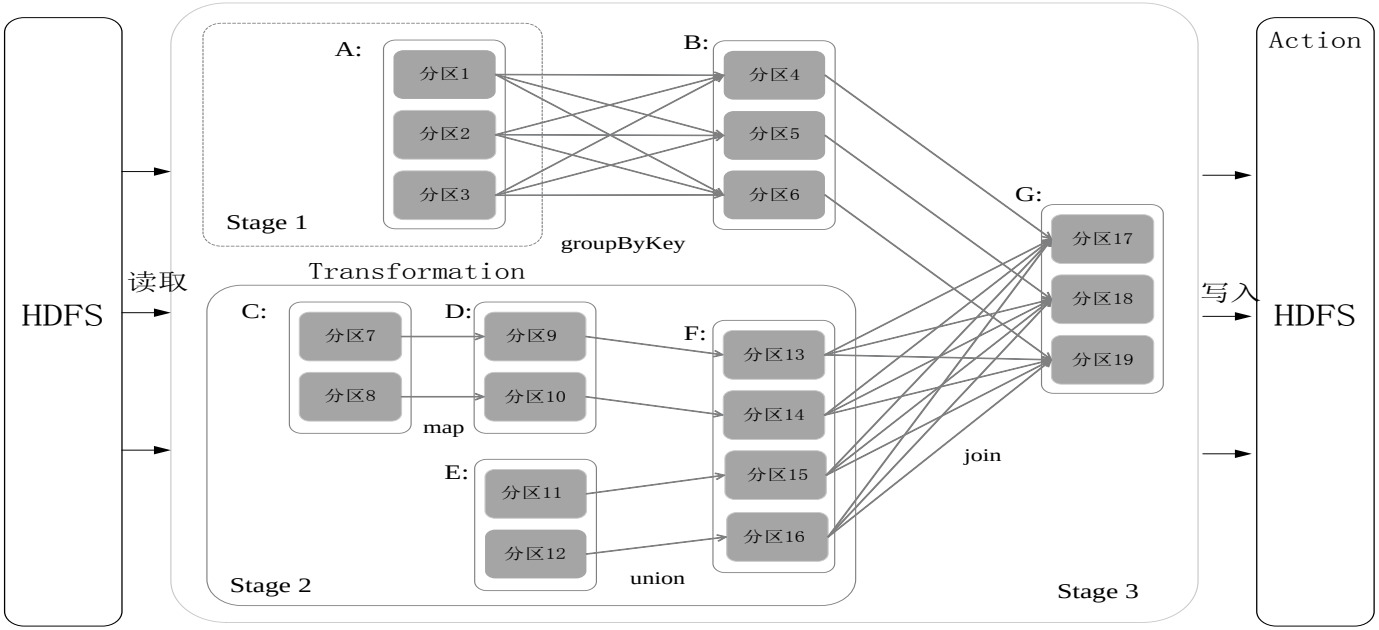


图10-10根据RDD分区的依赖关系划分Stage

流水线操作实例

分区7通过map操作生成的分区9，可以不用等待分区8到分区10这个map操作的计算结束，而是继续进行union操作，得到分区13，这样流水线执行大大提高了计算的效率

5.Stage的划分

Stage的类型包括两种：ShuffleMapStage和ResultStage，具体如下：

（1）ShuffleMapStage：不是最终的Stage，在它之后还有其他Stage，所以，它的输出一定需要经过Shuffle过程，并作为后续Stage的输入；这种Stage是以Shuffle为输出边界，其输入边界可以是从外部获取数据，也可以是另一个ShuffleMapStage的输出，其输出可以是另一个Stage的开始；在一个Job里可能有该类型的Stage，也可能没有该类型Stage；

（2）ResultStage：最终的Stage，没有输出，而是直接产生结果或存储。这种Stage是直接输出结果，其输入边界可以是从外部获取数据，也可以是另一个ShuffleMapStage的输出。在一个Job里必定有该类型Stage。

因此，一个Job含有一个或多个Stage，其中至少含有一个ResultStage。

6.RDD运行过程

通过上述对RDD概念、依赖关系和Stage划分的介绍，结合之前介绍的Spark运行基本流程，再总结一下RDD在Spark架构中的运行过程：

（1）创建RDD对象；

（2）SparkContext负责计算RDD之间的依赖关系，构建DAG；

（3）DAGScheduler负责把DAG图分解成多个Stage，每个Stage中包含了多个Task，每个Task会被TaskScheduler分发给各个WorkerNode上的Executor去执行。

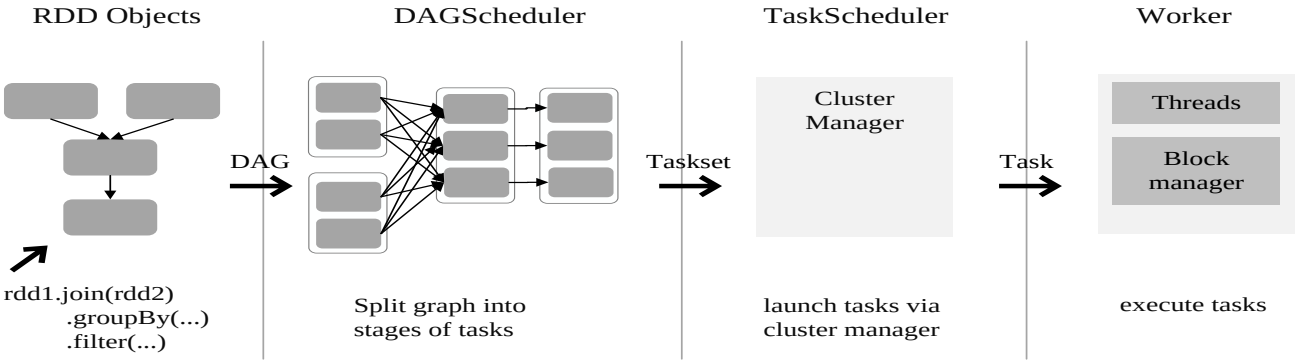


图10-11 RDD在Spark中的运行过程

10.4 Spark SQL

10.4.1 从Shark说起

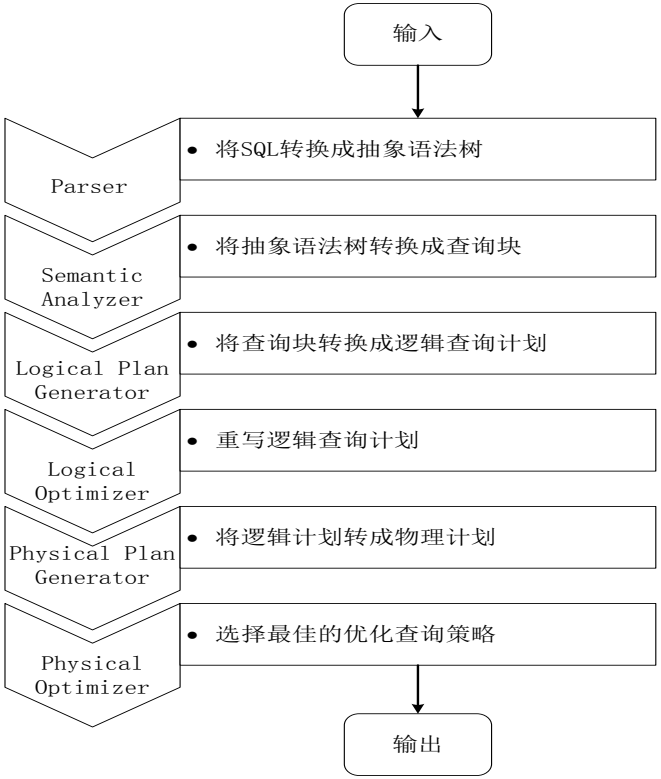
10.4.2 Spark SQL设计

10.4.1 从Shark说起

•Shark即Hive on Spark，为了实现与Hive兼容，Shark在HiveQL方面重用了Hive中HiveQL的解析、逻辑执行计划翻译、执行计划优化等逻辑，可以近似认为仅将物理执行计划从MapReduce作业替换成了Spark作业，通过Hive的HiveQL解析，把HiveQL翻译成Spark上的RDD操作。

•Shark的设计导致了两个问题：

- •一是执行计划优化完全依赖于Hive，不方便添加新的优化策略；
- •二是因为Spark是线程级并行，而MapReduce是进程级并行，因此，Spark在兼容Hive的实现上存在线程安全问题，导致Shark不得不使用另外一套独立维护的打了补丁的Hive源码分支



Hive中SQL查询的MapReduce作业转化过程

10.4.2 Spark SQL设计

Spark SQL在Hive兼容层面仅依赖HiveQL解析、Hive元数据，也就是说，从HQL被解析成抽象语法树（AST）起，就全部由Spark SQL接管了。Spark SQL执行计划生成和优化都由Catalyst（函数式关系查询优化框架）负责

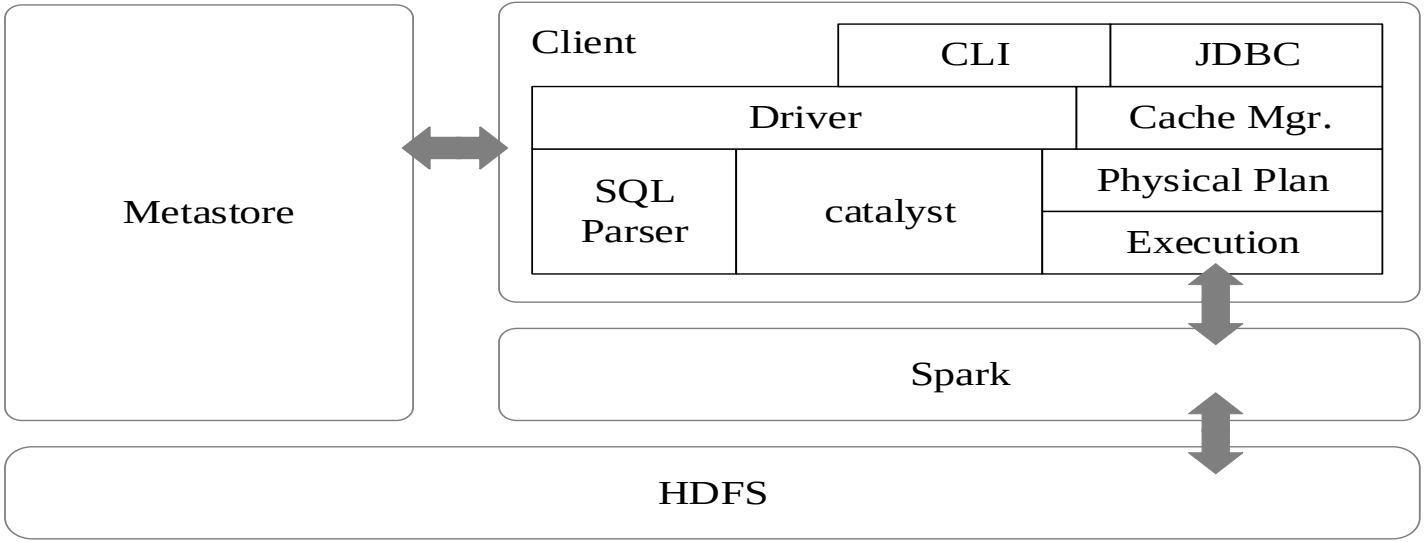


图10-12 Spark SQL架构

10.4.2 Spark SQL设计

•Spark SQL增加了SchemaRDD（即带有Schema信息的RDD），使用户可以在Spark SQL中执行SQL语句，数据既可以来自RDD，也可以是Hive、HDFS、Cassandra等外部数据源，还可以是JSON格式的数据

•Spark SQL目前支持Scala、Java、Python三种语言，支持SQL-92规范

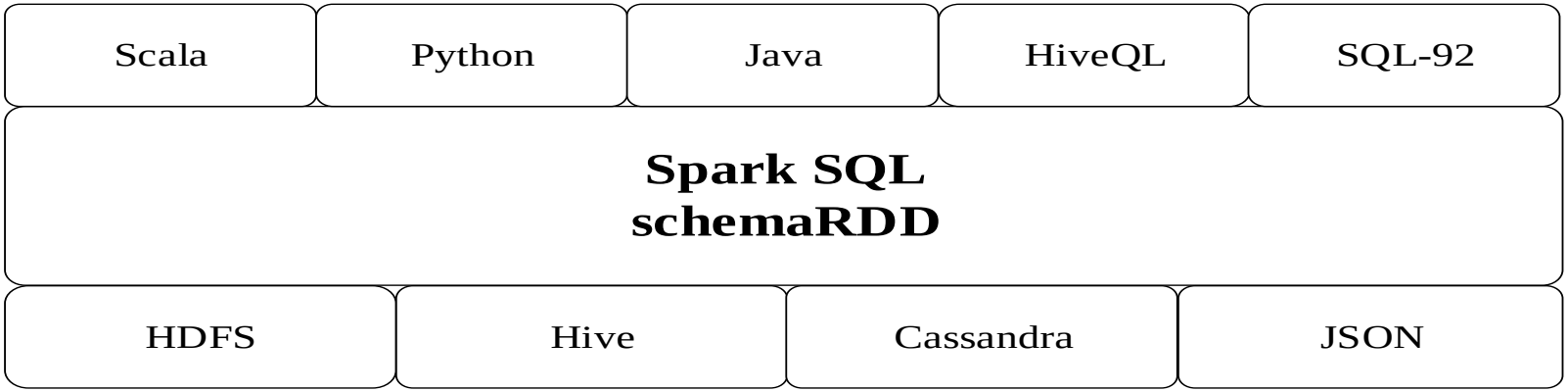


图10-13 Spark SQL支持的数据格式和编程语言

10.5 Spark的部署和应用方式

10.5.1 Spark三种部署方式
10.5.2 从Hadoop+Storm架构转向Spark架构
10.5.3 Hadoop和Spark的统一部署

10.5.1 Spark三种部署方式

Spark支持三种不同类型的部署方式，包括：

- Standalone（类似于MapReduce1.0，slot为资源分配单位）
- Spark on Mesos（和Spark有血缘关系，更好支持Mesos）
- Spark on YARN

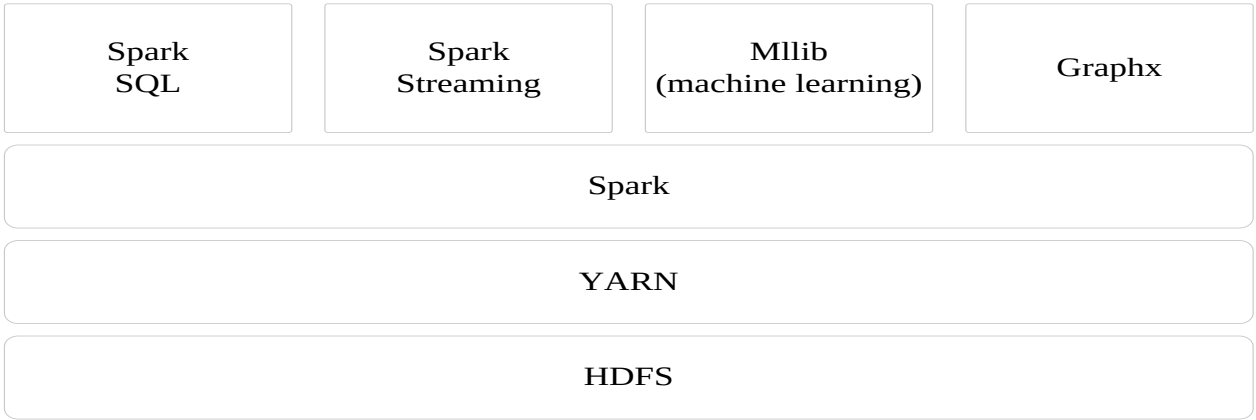
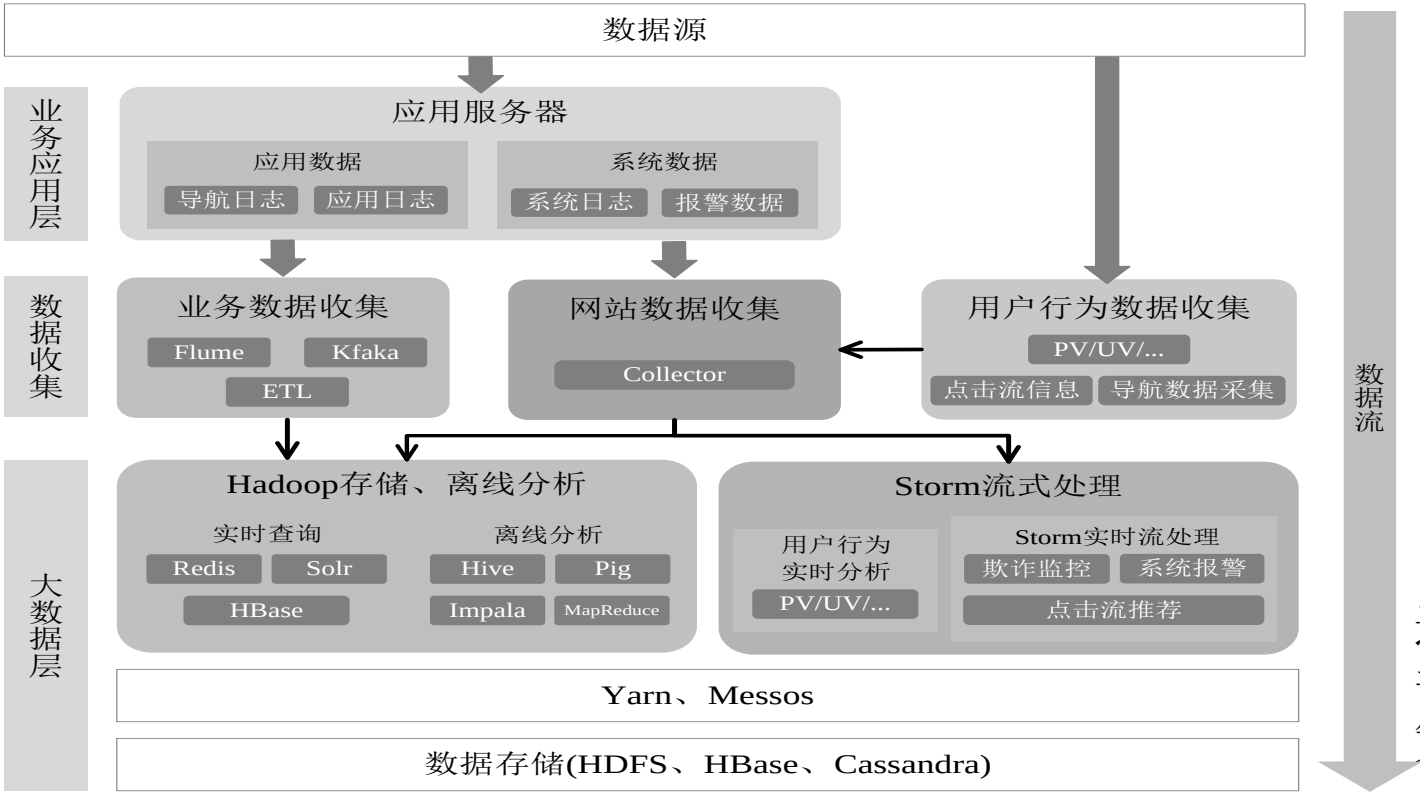


图10-17 Spark on Yarn架构

10.5.2 从Hadoop+Storm架构转向Spark架构



这种架构部署较为繁琐

图10-18 采用Hadoop+Storm部署方式的一个案例

10.5.2 从Hadoop+Storm架构转向Spark架构

用Spark架构具有如下优点：

- 实现一键式安装和配置、线程级别的任务监控和告警
- 降低硬件集群、软件维护、任务监控和应用开发的难度
- 便于做成统一的硬件、计算平台资源池

需要说明的是，Spark Streaming无法实现毫秒级的流计算，因此，对于需要毫秒级实时响应的企业应用而言，仍然需要采用流计算框架（如Storm）

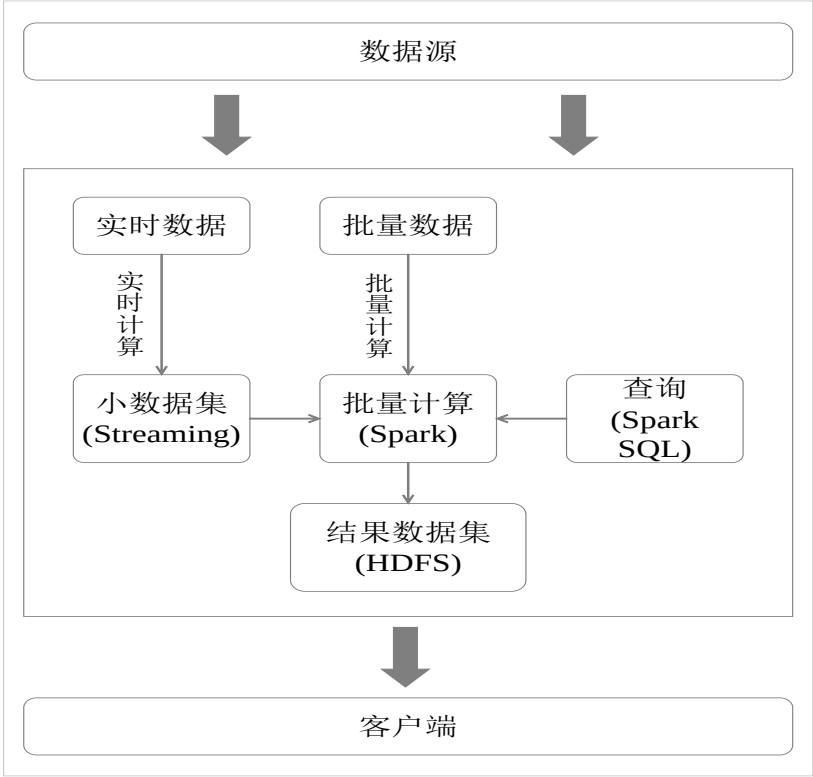


图10-19 用Spark架构满足批处理和流处理需求

10.5.3 Hadoop和Spark的统一部署

- 由于Hadoop生态系统中的一些组件所实现的功能，目前还是无法由Spark取代的，比如，Storm
- 现有的Hadoop组件开发的应用，完全转移到Spark上需要一定的成本

不同的计算框架统一运行在YARN中，可以带来如下好处：

- 计算资源按需伸缩
- 不用负载应用混搭，集群利用率高
- 共享底层存储，避免数据跨集群迁移

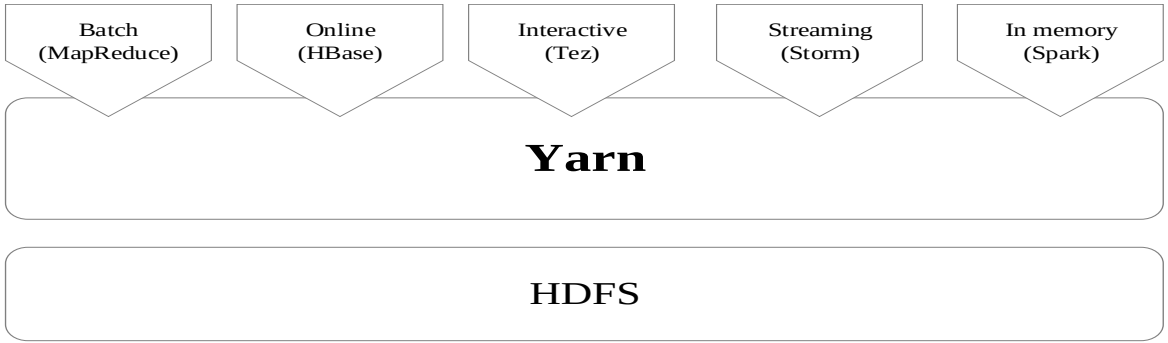


图10-20 Hadoop和Spark的统一部署

10. 6 Spark编程实践

10.6.1 Spark安装
10.6.2 启动Spark Shell
10.6.3 Spark RDD基本操作
10.6.4 Spark应用程序

Spark上机实践详细过程，请参考厦门大学数据库实验室建设的
“高校大数据课程公共服务平台”中的
“大数据课程学生服务站”中的“**学习指南**”栏目：
学生服务站地址：http://dblab.xmu.edu.cn/post/4331/

学习指南栏目中包含了《第十六章Spark学习指南》
http://dblab.xmu.edu.cn/blog/778-2/

扫一扫访问学生服务站

10.6.1 Spark安装

安装Spark之前需要安装Java环境和Hadoop环境。

•下载地址： http://spark.apache.org

进入下载页面后，点击主页右侧的“Download Spark”按钮进入下载页面，下载页面中提供了几个下载选项，主要是Spark release及Package type的选择，如下图所示。第1项Spark release一般默认选择最新的发行版本，如截止至2016年3月份的最新版本为1.6.0。第2项package type则选择“Pre-build with user-provided Hadoop [can use with most Hadoop distributions]”，可适用于多数Hadoop版本。选择好之后，再点击第4项给出的链接就可以下载Spark了。

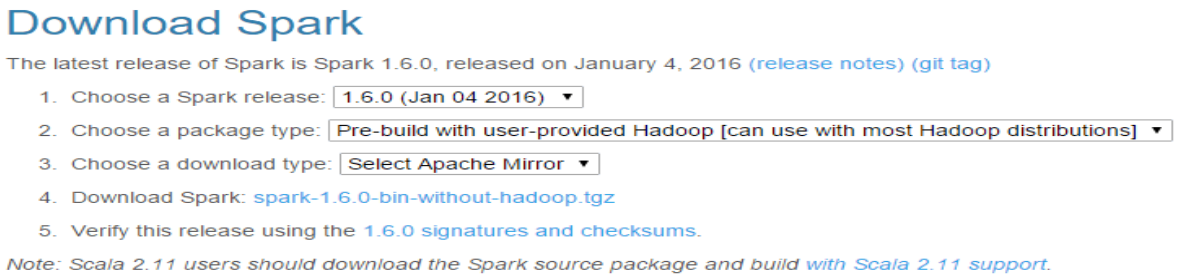


图 Spark下载选项

10.6.1 Spark安装

•解压安装包spark-1.6.0-bin-without-hadoop.tgz至路径 /usr/local：

```
$ sudo tar -zxf ~/下载/spark-1.6.0-bin-without-hadoop.tgz -C /usr/local/
$ cd /usr/local
$ sudo mv ./spark-1.6.0-bin-without-hadoop/ ./spark   # 更改文件夹名
$ sudo chown -R hadoop ./spark                        # 此处的 hadoop 为系统用户名
```

•配置Spark 的Classpath。

```
$ cd /usr/local/spark
$ cp ./conf/spark-env.sh.template ./conf/spark-env.sh   #拷贝配置文件
```

编辑该配置文件，在文件最后面加上如下一行内容：

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

保存配置文件后，就可以启动、运行Spark了。Spark包含多种运行模式：单机模式、伪分布式模式、完全分布式模式。本章使用单机模式运行Spark。若需要使用HDFS中的文件，则在使用Spark前需要启动Hadoop。

10.6.2启动Spark Shell

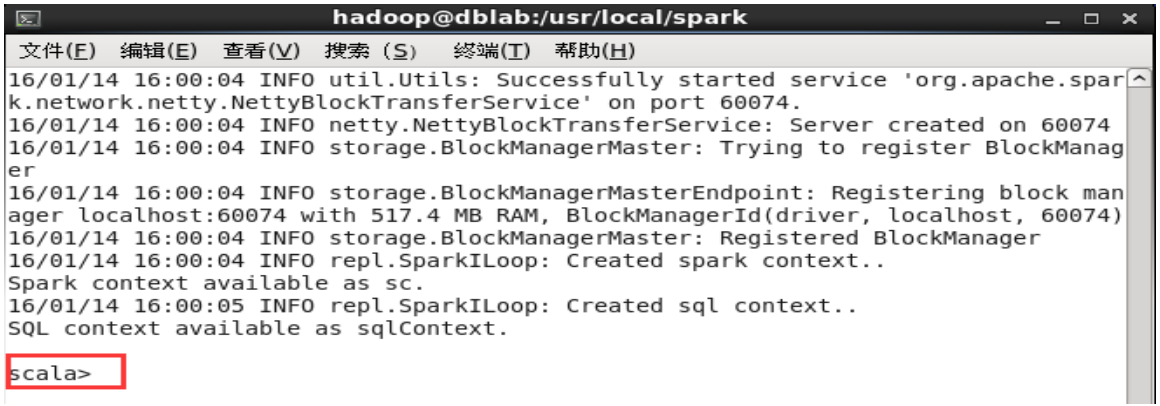
- Spark Shell 提供了简单的方式来学习Spark API
- Spark Shell可以以实时、交互的方式来分析数据
- Spark Shell支持Scala和Python

本章节内容选择使用Scala进行编程实践，了解Scala有助于更好地掌握Spark。

执行如下命令启动Spark Shell：

```
$ ./bin/spark-shell
```

启动Spark Shell成功后在输出信息的末尾可以看到“Scala >”的命令提示符，如下图所示。



10.6.3 Spark RDD基本操作

•Spark的主要操作对象是RDD，RDD可以通过多种方式灵活创建，可通过导入外部数据源建立，或者从其他的RDD转化而来。

•在Spark程序中必须创建一个SparkContext对象，该对象是Spark程序的入口，负责创建RDD、启动任务等。在启动Spark Shell后，该对象会自动创建，可以通过变量sc进行访问。

作为示例，我们选择以Spark安装目录中的“README.md”文件作为数据源新建一个RDD，代码如下：

```
Scala > val textFile = sc.textFile("file:///usr/local/spark/README.md")
                                        // 通过file:前缀指定读取本地文件
```

Spark RDD支持两种类型的操作：
动作（action）：在数据集上进行运算，返回计算值
转换（transformation）： 基于现有的数据集创建一个新的数据集

10.6.3 Spark RDD基本操作

Spark提供了非常丰富的API，下面两表格列出了几个常用的动作、转换API，更详细的API及说明可查阅官方文档。

7-1常用的几个Action API介绍

Action API	说明
count()	返回数据集中的元素个数
collect()	以数组的形式返回数据集中的所有元素
first()	返回数据集中的第一个元素
take(n)	以数组的形式返回数据集中的前n个元素
reduce(func)	通过函数func（输入两个参数并返回一个值）聚合数据集中的元素
foreach(func)	将数据集中的每个元素传递到函数func中运行

7-2常用的几个Transformation API介绍

Transformation API	说明
filter(func)	筛选出满足函数func的元素，并返回一个新的数据集
map(func)	将每个元素传递到函数func中，并将结果返回为一个新的数据集
flatMap(func)	与map()相似，但每个输入元素都可以映射到0或多个输出结果
groupByKey()	应用于(K,V)键值对的数据集时，返回一个新的(K, Iterable<V>)形式的数据集
reduceByKey(func)	应用于(K,V)键值对的数据集时，返回一个新的(K, V)形式的数据集，其中的每个值是将每个key传递到函数func中进行聚合

10.6.3 Spark RDD基本操作

•使用action API - count()可以统计该文本文件的行数，命令如下：

```
Scala > textFile.count()
```

输出结果 Long = 95（ “Long=95”表示该文件共有95行内容）。

•使用transformation API - filter()可以筛选出只包含Spark的行，命令如下：

```
Scala > val linesWithSpark = textFile.filter(line => line.contains("Spark"))
Scala > linesWithSpark.count()
```

第一条命令会返回一个新的RDD；
输出结果Long=17（表示该文件中共有17行内容包含“Spark”）。
也可以在同一条代码中同时使用多个API，连续进行运算，称为链式操作。不仅可以使Spark代码更加简洁，也优化了计算过程。如上述两条代码可合并为如下一行代码：

```
Scala > val linesCountWithSpark
= textFile.filter(line => line.contains("Spark")).count()
```

假设我们只需要得到包含“Spark”的行数，那么存储筛选后的文本数据是多余的，因为这部分数据在计算得到行数后就不再使用到了。Spark基于整个操作链，仅储存、计算所需的数据，提升了运行效率。

10.6.3 Spark RDD基本操作

Spark属于MapReduce计算模型，因此也可以实现MapReduce的计算流程，如实现单词统计，可以使用如下的命令实现：

```
Scala > val wordCounts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey((a, b) => a + b)
Scala > wordCounts.collect() // 输出单词统计结果
// Array[(String, Int)] =
Array((package,1), (For,2), (Programs,1), (processing.,1), (Because,1), (T
```

- 首先使用flatMap()将每一行的文本内容通过空格进行划分为单词；
- 再使用map()将单词映射为(K,V)的键值对，其中K为单词，V为1；
- 最后使用reduceByKey()将相同单词的计数进行相加，最终得到该单词总的出现的次数。

输出结果 Long = 95（“Long=95”表示该文件共有95行内容）。

10.6.4 Spark应用程序

在Spark Shell中进行编程主要是方便对代码进行调试，但需要以逐行代码的方式运行。一般情况下，会选择将调试后代码打包成独立的Spark应用程序，提交到Spark中运行。

采用Scala编写的程序需要使用sbt（Simple Build Tool）进行打包，sbt的安装配置步骤如下：

1. 下载sbt-launch.jar（下载地址 http://pan.baidu.com/s/1eRyFddw ）
2. 将下载后的文件拷贝至安装目录/usr/local/sbt中，命令如下：

```
sudo mkdir /usr/local/sbt      # 创建安装目录
cp ~/下载/sbt-launch.jar /usr/local/sbt
sudo chown -R hadoop /usr/local/sbt  #此处的hadoop为系统当前用户名
```

3. 在安装目录中创建一个Shell脚本文件（文件路径：/usr/local/sbt/sbt）用于启动sbt，脚本文件中的代码如下：

```
#!/bin/bash
SBT_OPTS="-Xms512M -Xmx1536M -Xss1M -XX:+CMSClassUnloadingEnabled -
XX:MaxPermSize=256M"
java $SBT_OPTS -jar `dirname $0`/sbt-launch.jar "$@"
```

4. 保存后，还需要为该Shell脚本文件增加可执行权限，命令如下：

```
chmod u+x /usr/local/sbt/sbt
```

10.6.4Spark应用程序

我们以一个简单的程序为例，介绍如何打包并运行Spark程序，该程序的功能是统计文本文件中包含字母a和字b的各有多少行，具体步骤如下：

1. 创建程序根目录，并创建程序所需的文件夹结构，命令如下：

```
mkdir ~/sparkapp                # 创建程序根目录
mkdir -p ~/sparkapp/src/main/scala   # 创建程序所需的文件夹结构
```

2. 创建一个SimpleApp.scala文件（文件路径：~/sparkapp/src/main/scala/SimpleApp.scala），文件中的代码内容如下：

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object SimpleApp {
  def main(args: Array[String]) {
    val logFile = "file:///usr/local/spark/README.md" // 用于统计的文本文件
    val conf = new SparkConf().setAppName("Simple Application")
    val sc = new SparkContext(conf)
    val logData = sc.textFile(logFile, 2).cache()
    val numAs = logData.filter(line => line.contains("a")).count()
    val numBs = logData.filter(line => line.contains("b")).count()
    println("Lines with a: %s, Lines with b: %s".format(numAs, numBs))
  }
}
```

10.6.4 Spark应用程序

3. 然后创建一个simple.sbt文件（文件路径：~/sparkapp/simple.sbt），用于声明该应用程序的信息以及与Spark的依赖关系，具体内容如下：

```
name := "Simple Project"
version := "1.0"
scalaVersion := "2.10.5"
libraryDependencies += "org.apache.spark" %% "spark-core" % "1.6.0"
```

4. 使用sbt对该应用程序进行打包，命令如下：

```
cd ~/sparkapp
/usr/local/sbt/sbt package
```

打包成功后，会输出程序jar包的位置以及“Done Packaging”的提示，如下图所示。

10.6.4 Spark应用程序

有了最终生成的jar包后，再通过spark-submit就可以提交到Spark中运行了，命令如下：

```
/usr/local/spark/bin/spark-submit --class "SimpleApp"
~/sparkapp/target/scala-2.10/simple-project_2.10-1.0.jar
```

该应用程序的执行结果如下：

Lines with a: 58, Lines with b: 26

附录A：主讲教师林子雨简介

主讲教师：林子雨

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），现为厦门大学计算机科学系助理教授（讲师），曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国计算机学会数据库专业委员会委员，中国计算机学会信息系统专业委员会委员。中国高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度和2017年度厦门大学教学类奖教金获得者。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，并以第一作者身份在《软件学报》《计算机学报》和《计算机研究与发展》等国家重点期刊以及国际学术会议上发表多篇学术论文。作为项目负责人主持的科研项目包括1项国家自然科学青年基金项目(No.61303004)、1项福建省自然科学青年基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，主持的教改课题包括1项2016年福建省教改课题和1项2016年教育部产学协作育人项目，同时，作为课题负责人完成了国家发改委城市信息化重大课题、国家物联网重大应用示范工程区域试点泉州市工作方案、2015泉州市互联网经济调研等课题。中国高校首个“数字教师”提出者和建设者，2009年至今，“数字教师”大平台累计向网络免费发布超过500万字高价值的研究和教学资料，累计网络访问量超过500万次。打造了中国高校大数据教学知名品牌，编著出版了中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》，并成为京东、当当网等网店畅销书籍；建设了国内高校首个大数据课程公共服务平台，为教师教学和学生学习大数据课程提供全方位、一站式服务，年访问量超过100万次。

附录B：《大数据技术原理与应用》教材

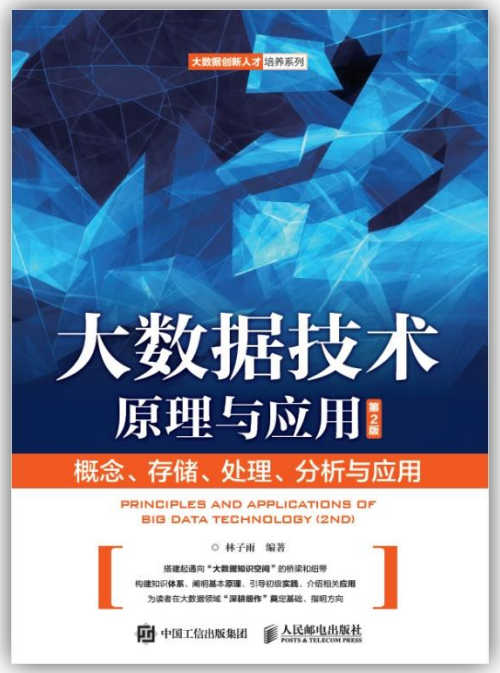
扫一扫访问教材官网

《大数据技术原理与应用——概念、存储、处理、分析与应用（第2版）》，由厦门大学计算机科学系林子雨博士编著，是中国高校第一本系统介绍大数据知识的专业教材。

全书共有15章，系统地论述了大数据的基本概念、大数据处理架构Hadoop、分布式文件系统HDFS、分布式数据 库HBase、NoSQL数据库、云数据库、分布式并行编程模型MapReduce、Spark、流计算、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在Hadoop、HDFS、HBase和MapReduce等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。

欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata



附录C：《大数据基础编程、实验和案例教程》

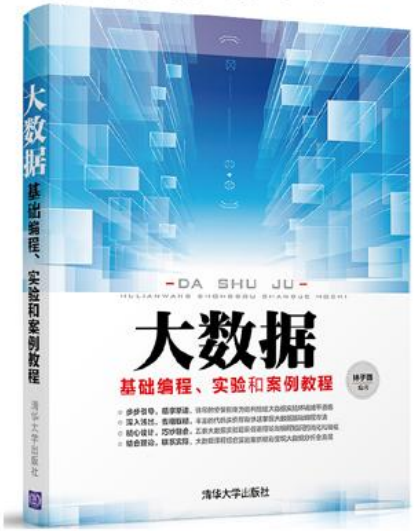
本书是与《大数据技术原理与应用（第2版）》教材配套的唯一指定实验指导书

大数据教材

1+1黄金组合

厦门大学林子雨编著

配套实验指导书



- 步步引导，循序渐进，详尽的安装指南为顺利搭建大数据实验环境铺平道路
- 深入浅出，去粗取精，丰富的代码实例帮助快速掌握大数据基础编程方法
- 精心设计，巧妙融合，五套大数据实验题目促进理论与编程知识的消化和吸收
- 结合理论，联系实际，大数据课程综合实验案例精彩呈现大数据分析全流程

清华大学出版社 ISBN:978-7-302-47209-4

附录D：高校大数据课程公共服务平台

高校大数据课程

公 共 服 务 平 台

http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/

扫一扫访问平台主页

扫一扫观看3分钟FLASH动画宣传片


Thank You!