TURING 图灵原创

- 季昕华、徐羽作序，连城/胡熠/武泽胜/肖磊/靳志辉联袂推荐
- 腾讯专家首次分享Spark最佳实践
- 基于真实数据，用案例分析全面解读大数据应用设计

Spark最佳实践

陈欢 林世飞◎著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵原创

Spark 最佳实践

陈欢 林世飞◎著

人民邮电出版社
北京

图书在版编目（CIP）数据

Spark最佳实践 / 陈欢，林世飞著. -- 北京 : 人民邮电出版社，2016.5
（图灵原创）
ISBN 978-7-115-42228-6

Ⅰ. ①S… Ⅱ. ①陈… ②林… Ⅲ. ①数据处理软件 Ⅳ. ①TP274

中国版本图书馆CIP数据核字(2016)第083128号

内 容 提 要

本书是 Spark 实战指南，全书共分 8 章。前 4 章介绍 Spark 的部署、工作机制和内核，后 4 章分别通过实战项目介绍 Spark SQL、Spark Streaming、Spark GraphX 和 Spark MLlib 功能模块。此外，本书详细介绍了常见的实战问题，比如大数据环境下的配置设置、程序调优等。本书附带的一键安装脚本，更能为初学者提供很大帮助。

本书适合大数据开发、运维等相关从业人员学习参考。

◆ 著　　陈　欢　林世飞
责任编辑　毛倩倩
责任印制　彭志环
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷
◆ 开本：800×1000　1/16
印张：14　　彩插：2
字数：339千字　　2016年5月第1版
印数：1－4 000册　　2016年5月河北第1次印刷

定价：49.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广字第 8052 号

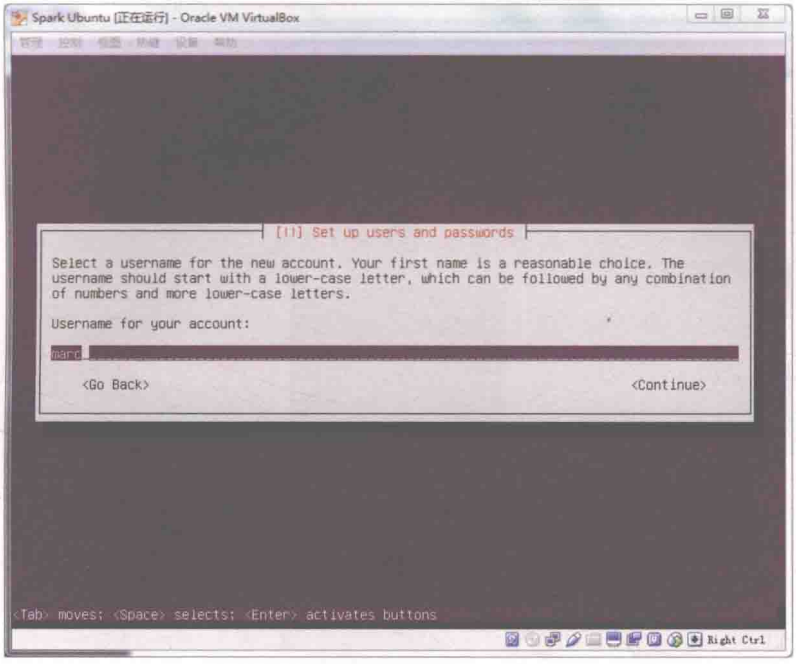

图 2-9　添加账号

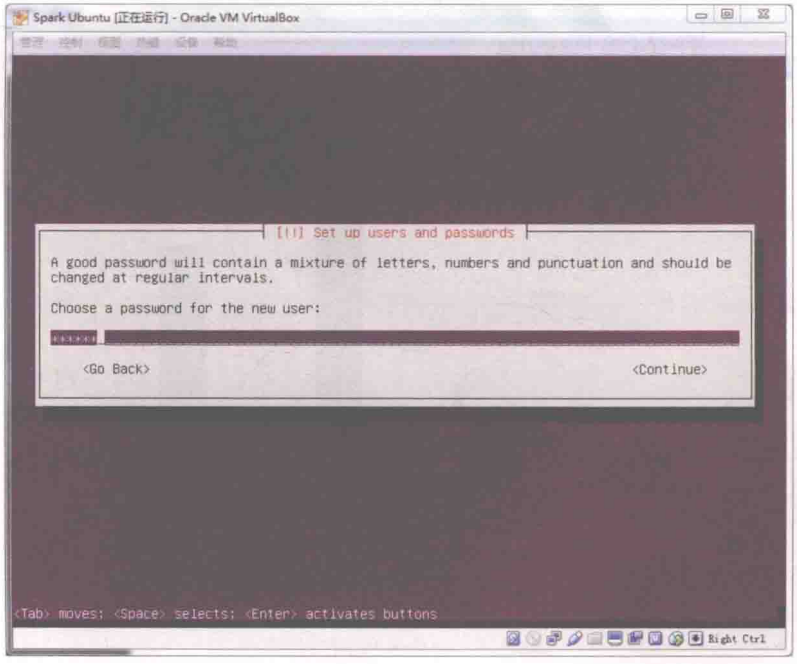

图 2-10　设置密码

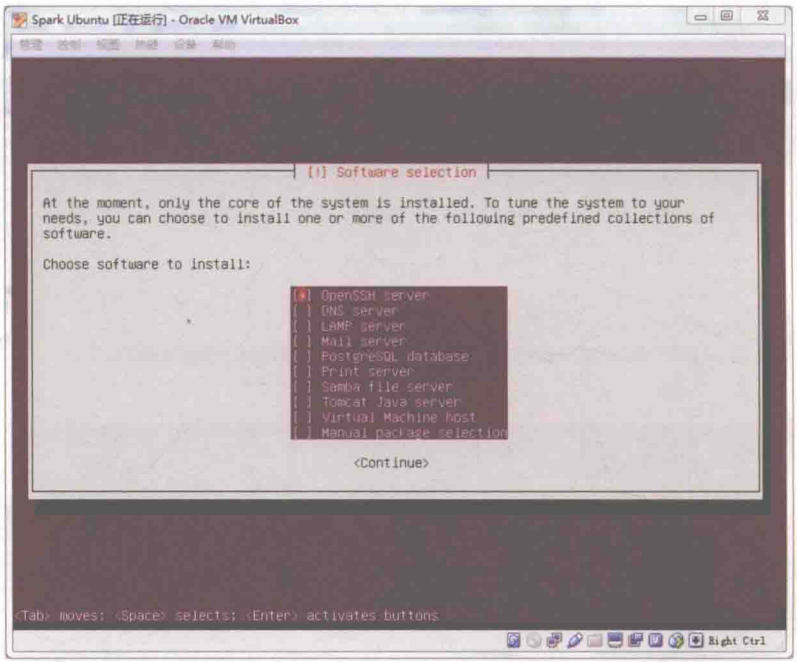

图 2-11　选择 OpenSSH server

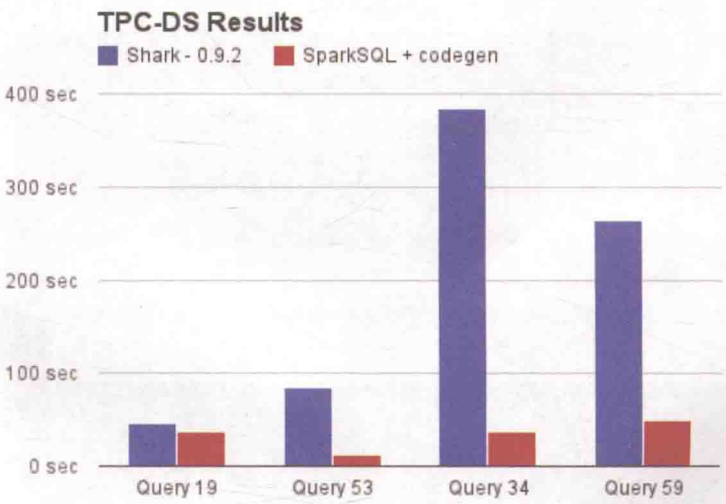

图 5-4　对比性能测试结果

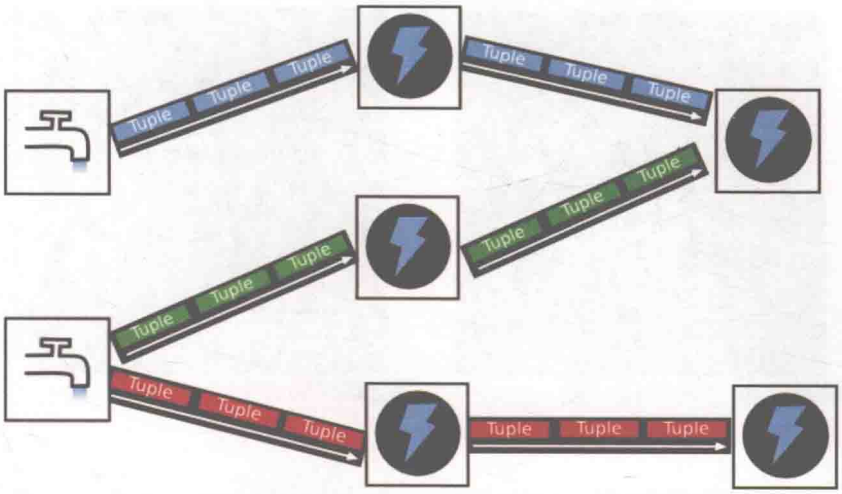

图 6-10　Storm 结构示意图

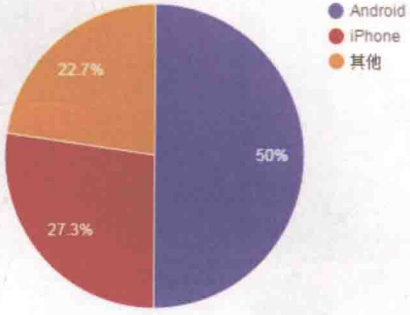

图 6-11　终端类型分布图示例

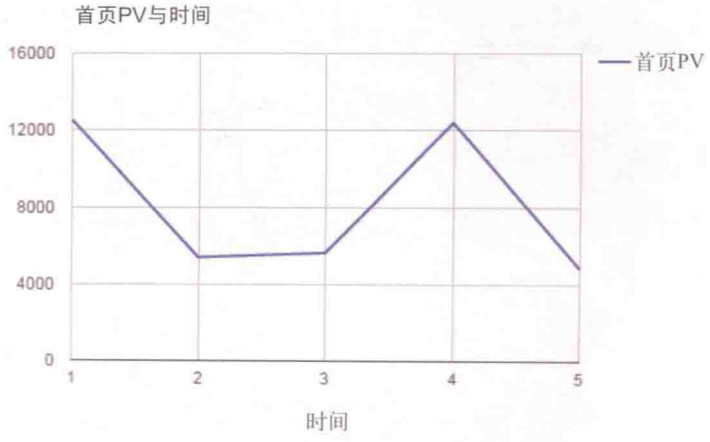

图 6-12　首页 PV 趋势图示例

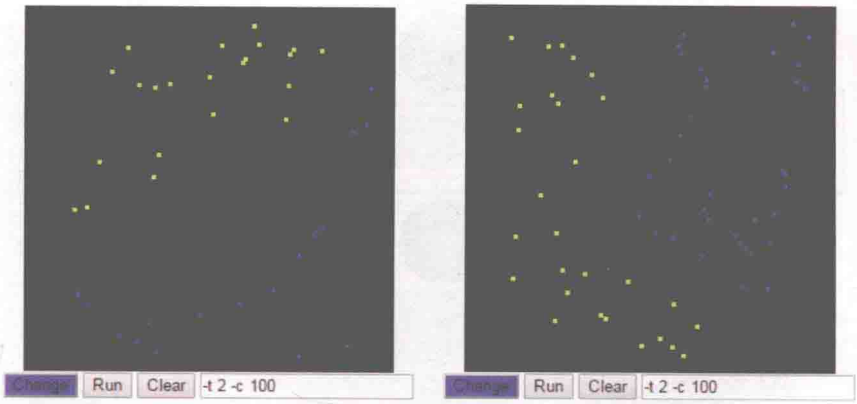

图 8-1　SVM 算法效果演示：学习的对象

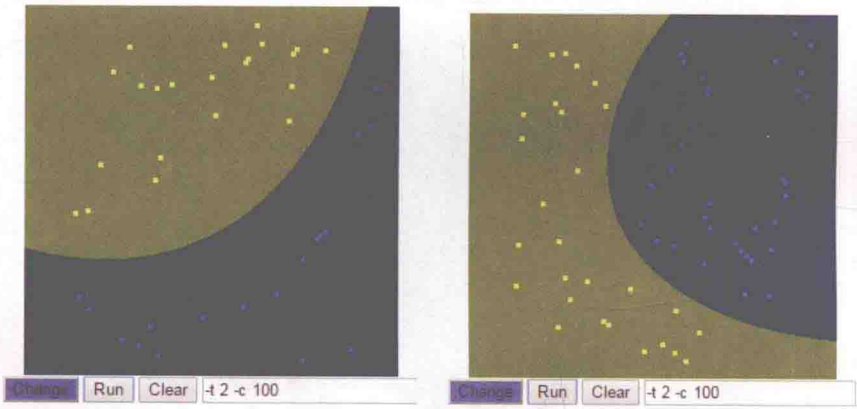

图 8-2　SVM 算法效果演示：学习的效果

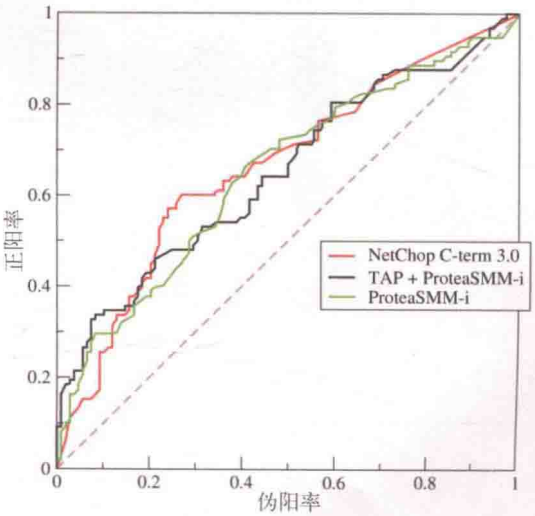

图 8-9　ROC 曲线示例（来自维基百科）

序　一

随着互联网的快速发展，特别是云计算的普及，大数据日益受到人们的重视，最近几年全球数据量以每年约 50%的速度递增。大数据正在事实上改变着我们的思维和生产方式，未来人类社会的精神和物质世界都将构建在数据之上。2015 年 9 月，国务院印发《促进大数据发展行动纲要》，确定了大数据发展的国家顶层设计，大数据与各行各业的结合已是行业未来发展的必然趋势。大数据在中国已经全面生根发芽，大数据创业企业也迎来了一波新的发展机会。

学习大数据技术、使用大数据技术为各行各业服务是每个 IT 人都梦寐以求的事情。在中国，BAT 的大数据员工是最幸运的一群人，因为他们手上拥有最多的数据，能利用海量的服务器资源捣腾海量的数据。

我的老朋友林世飞先生和陈欢先生正是这样的幸运者，他们刚毕业就加入了腾讯，在腾讯多个部门工作学习锻炼过，每天接触的都是上亿用户的数据处理。而现在他们又在腾讯发展最快速的部门——社交与效果广告部，负责大数据处理和分析的相关工作，利用成规模的 Spark 集群来做海量数据的处理，根据用户的画像给用户推荐个性化的广告。除此之外，林世飞和陈欢本身有着爱钻研的性格，他们成立了学习小组，拿着各种大数据技术的“锤子”，敲打着各种内外部的大数据“钉子”，来仔细分析、对比各种技术的差异点以及特性。在此基础上，他们总结成书，从运行原理到实际案例分析，覆盖了流式计算、数据仓库、图计算、机器学习等算法应用。在我看来，这应该是国内第一本兼顾基础，针对 Spark 的典型应用都做了案例讲解的实践书，对于创业公司快速搭建大数据系统，以及科研院校学生的毕业设计的实战项目帮助意义很大。

丰富的大数据实践经验、对大数据深入的思考，这是本书的两大特点，也是林世飞先生和陈欢先生严谨工作、对技术深入研究的体现，希望他们付出的心血能为每一位读者带来价值，使大家深入地了解 Spark 的工作原理、掌握好 Spark 的优势，为日常工作创造更大的价值。

季昕华

UCloud 联合创始人兼 CEO

序　二

从 2003 年开始，谷歌先后发表了关于 GFS、MapReduce 和 BigTable 的 3 份重量级论文。在 Apache 软件基金会和雅虎等互联网公司的支持下，人们在参考谷歌的论文基础上实现了大量的开源服务框架（包括著名的 Hadoop、Hive 等重量级产品），特别是开源项目 Hadoop 的出现揭开了在 IT 行业，特别是互联网行业内大规模使用大数据技术的序幕。随着最近几年海量数据的持续增长和计算机硬件的发展，越来越多的新架构也涌现出来。从 2010 年开始，美国加州大学伯克利分校陆续提出了多份 RDD（Resilient Distributed Dataset，弹性分布式数据集）相关的论文，并随之推出开源的 Spark 框架。对比传统的 Hadoop，拥有深厚学术界背景的 Spark 把以往的 MapReduce、流式计算、机器学习算法等模型全部统一起来，让数据挖掘和机器学习的门槛大大降低，从而加速了大数据技术在各个行业和产品里面的普及。

手机 QQ 浏览器的大数据开发模式同样符合这种演进趋势。2011 年，我们主要基于 MapReduce 模式和 Hive 进行一些海量数据的分析和统计工作，而数据挖掘算法依然沿用传统的 C++和 MPI 框架方式。随着这两年 Spark 版本的不断更新和功能的逐步强大，我们已经开始用 Scala 和 Spark 自带的 MLlib 实现多个数据挖掘模型。由于使用了统一的 RDD 和 MLlib，我们可以快速实现各种算法模型，并在它们之间更灵活地进行切换和对比，这也体现了互联网快速迭代开发的效率和 ABTest 的思维模式。

最近的一年里面，世飞负责的广点通项目和手机QQ浏览器有深入的技术和产品层面的合作，广点通的广告推荐投放在手机浏览器里面也取得了非常好的效果。在工作交流和讨论的过程中，我深深体会到世飞在数据挖掘算法上的深厚功力，以及在 Spark 技术上追求极致的精神。这次有机会拜读世飞和陈欢所著的书稿，最大的体会是与市面上大量的手册型图书不同，本书不仅可以让读者逐步掌握 Spark 的核心概念和流程，而且得以吸取大数据业务特定场景下的实战经验。这些真正经过海量用户考验的行业案例，对于大数据的学习人员来说都传递着非常宝贵的经验。最后，衷心希望本书可以在为每一位读者带来 Spark 底层技术知识的同时，更好地让 Spark 普及到更多的大数据应用场景当中。

徐羽

手机 QQ 浏览器总监、腾讯 T4 技术专家

前　言

近些年，大数据的概念非常火，且许多行业已经开始从大数据中获益，比如广告业。越来越多的企业也正在尝试使用大数据的平台和方法来解决一些现实问题，不过，在这个过程中难免会遇到困难。因此，本书旨在提供一个非常具有可操作性的指引，帮助读者亲身体验大数据的魅力。

Spark 是最近几年开始流行的一种通用分布式大数据计算平台，无论是性能、功能还是易用性，都在众多大数据技术中名列前茅，可以很好地解决大数据领域的许多常见问题。本书循序渐进地介绍 Spark 的基本概念、核心思想、部署、开发，并提供多个典型场景的解决方案，试图让即便是零基础的 Spark 读者也可以从中受益，并让对 Spark 已有所知的读者可以更深入地了解其运行机制及精髓。

没有大数据平台的支持，普通海量数据的计算分析极端低效

笔者 2008 年毕业后即进入腾讯公司工作至今，刚进公司时感觉真是“一夜暴富”，因为名下的服务器有上百台，日常工作除了后台应用程序的开发，还要担负一些运营工作。然而，维护上百台机器上产生的日志文件就是一个非常令人头痛的问题。虽然每台机器都有 TB 级的硬盘空间，但还是经常爆满，我们经常半夜被电话吵醒起来删文件。此外，产品人员经常提出一些数据统计分析的需求，然而数据分布在上百台机器上，统计效率非常低下。

面对这样大量数据的统计，最常用的一种处理模式是，选择一个字段对数据进行散列化，然后按散列值的不同分别存储在单独的 MySQL 数据库表中，而这些 MySQL 节点分布在上百台机器上。需要统计的时候，我们再写个脚本分别去各个 MySQL 节点上取数据，或者简单处理一下，然后再汇总到一台机器上做处理。

这种方法简单易行，但问题很多。最大的问题是计算过程复杂，开发统计脚本的时间以天计，因为原本一个 SQL 就可以搞定的事情，现在需要分多步进行，且中间还需要将大量数据在多个节点上进行传输。比如这个简单的案例，它需要对数据按 IP 进行分布统计，计算过程至少需要 3 步：

(1) 去各个 MySQL 节点上执行 SQL，按 IP 汇总统计；
(2) 数据汇总到一个节点上，导入 MySQL；
(3) 在汇总的 MySQL 节点上进行统计。

而如果数据没有拆分存储，一个 SQL 就可以搞定。这只是一个最简单的例子，如果统计更复杂一些，这样的“计算–汇总–计算”的过程可能需要进行多次。

这种方法带来的第二个问题是散列分布不均匀导致的计算瓶颈和资源浪费。为了方便查数据，散列的规则设计不能太复杂，一般是取某个整数的十进制数值的前 3 位或末 3 位。这样的好处是日常定位数据位置很方便，缺点是数据分布不均匀，总有一些节点会承载非常多的数据，而另外一些节点又承载非常少的数据。承载数据多的机器会成为整体计算的瓶颈，而承载数据少的机器总是空闲，从而造成资源的浪费。

所以整体看来，这种方法的效率非常低下，计算本身耗时虽然不多，但开发过程相当费时，如果逻辑更复杂，甚至需要一周时间。

Hadoop 和 Hive 的组合，可以解决大数据计算的许多常见问题

后来我们引入“Hadoop + Hive”的解决方案，问题改善了很多。数据全部存储在 Hadoop 上，在各节点上均匀分布。在 Hive 上创建表并导入数据，设置好分区，这样虽然数据的存储还是分布式的，但从数据统计角度来看像是一个数据库，而不是之前的上千个数据库表。这样，开发者只需要关注自己的业务逻辑，而不需要关注底层数据的存储、分区等信息，开发效率大大提升。虽然 Hive 的计算效率不是很高，运行时间可能是“脚本 + MySQL”方式的数倍，但一般也只需要几十分钟，而节省的开发成本是以“天”计的，整体效率大大提升。

然而，随着数据量的继续快速上升，以及大家对数据的关注度的提高，Hive 平台上的计算量越来越大，人们对于计算的需求也在变化。具体来讲，一是统计规则越来越复杂，原来几行 SQL 代码可以搞定的统计越来越少，新的统计需求动辄需要上百行 SQL 代码才能完成，而且量越来越多；二是越来越多的计算是 SQL 完成不了的，需要借助机器学习算法或图计算来进行。

这些新需求带来的挑战逐渐让“Hadoop + Hive”方式有些吃不消，一些问题逐渐暴露出来：一是性能不够高，一是简单的 MR 计算模式支持的应用场景非常有限。

Spark 出现之后，性能、功能、易用性都有质的提升

Hadoop 的这些问题暴露之后，业界也有一些针对性的优化方案。但 Spark 走得更远，在参考 MR 的基础上重新实现了一套通用计算框架，性能提升 10~100 倍，尤其是在机器学习等复杂迭代计算的场景下，性能提升最明显。另外，使用场景除了 SQL 之外，Spark 还支持类似 Storm 的实时流式计算和图计算，机器学习库也更丰富，为开发带来了极度的便利。因此，越来越多的技术人员开始使用 Spark 来替代 Hadoop 的 MR 计算，用 Spark SQL 来替代 Hive，但存储依然使用 Hadoop HDFS。

本书会详细介绍 Spark，值得注意的是，Spark 在设计伊始就充分考虑到了用户体验问题。

Spark 使用 Scala 语言开发，支持函数式编程，让代码非常简洁。比如，Hadoop MR 编程中的示例程序 WordCount，在 Hadoop 下用 Java 实现时代码有上百行，而且还需要经过编写代码文件、编译、上传执行等过程，而 Spark 集成的 Scala 语言支持交互式编程，进入交互式模式后直接输入代码即可开始执行，并且代码只有区区几行：

```
text_file = spark.textFile("hdfs://...")
text_file.flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a+b)
```

交互式编程配合函数式编程，使得 Spark 编程非常容易上手。笔者曾经考虑要不要让 Spark 支持 bash 这类脚本语言，但最终觉得 Scala 足矣。本书附录提供了 Scala 语法的简单参考，相信有一定编程基础的读者会非常容易上手。

经常有人会问：为何 Spark 会比 Hadoop 快那么多呢？我也曾反复思考过，在此抛砖引玉了。Spark 和 Hadoop 都是非常优秀的开源项目，代码实现及架构已经非常优秀了，差别应该不在实现层面。大多数人认为 Spark 比 Hadoop 快是因为大量使用内存，正因此，Spark 一开始被称作内存计算，但我认为还有一个更重要的原因，那就是核心数据结构。Spark 使用 RDD（弹性分布式数据集，读者可以将其想象成一个分布式的大数组）作为核心数据结构（相比之下 Hadoop 没有核心数据结构，只有 Map/Reduce 计算模式），在此基础上提供了许多计算函数，类似 Hadoop 下的 Map/Reduce 函数，但数量更多，还可以无限扩充。这样的好处是 Spark 的任务粒度更小，原先在 Hadoop 下一个 Map 或 Reduce 实现的功能，在 Spark 下可能会被拆分成多个 Job。如果把 Hadoop 中的任务比作罐子里面的石头，那么 Spark 的 Job 就是罐子里面的碎石子，因此相同罐子可以装得更多。Spark 这样细粒度的任务调度，再配合上内存的充分利用，最终让性能提升了一个台阶。

还有一个有意思的现象：大数据领域的技术很多是基于 Java 开发的，Hadoop、Hive、HBase 都是，Spark 使用的 Scala 最后也是转化成 JVM 字节码运行。Hadoop 虽然被诟病性能低，但也没有基于 C++的开源版本出现，究其原因，除了大家经常提到的这些开源系统的创始人喜欢 Java，以及 Java 对于跨平台的强有力支持，我猜测可能还有如下几个原因。一是因为 Java 在大型系统开发方面的便利性，有非常多的库可以使用。这样一来，在系统最初的探索阶段可以节省大量人力，而 Spark 使用的 Scala 语言因为支持函数式编程，代码量进一步精简了好几倍，比如 Spark 中的 SQL 优化执行引擎只用 2000 行左右的代码就实现了，而且表现接近于手写代码，完全受益于语言在开发上的便利性。二是因为大数据计算场景下，系统的瓶颈经常不在于计算，而是网络传输和磁盘读写，此时 CPU 反而不是瓶颈，所以无论 Hadoop 和 Spark 都在这方面做了大量优化，语言性能方面的弱势不那么明显了。比如 Hadoop 和 Spark 中最重要的 shuffle 机制，就是为解决网络传输而设计的，而且不断优化，每次优化都可以在整体上提升性能。三是在 Hadoop 和 Spark 这类通用计算平台上都遵循一个理念，就是“宁可移动计算，不要移动数据”，这正是 Java 语言强大的序列化功能发威的时候。当然 C++也可以实现，性能可能会更优，但开发成本可能不是一般小公司能承受得了的，也只有谷歌这样技术、财力雄厚的公司才有可能实现。事实上，Hadoop

的理论基础也来自谷歌的论文。

大数据平台众多，我们要怎么选呢

大数据热门之后出现了非常多的大数据技术，有 Hadoop、Hive、HBase、Flume、Kafka、Lucene、Spark、Storm 等，还有很多 NoSQL 技术可以归入大数据，比如 MongoDB、CouchDB、Cassandra 等。有人列举过大数据技术族谱，其中至少上百种技术。面对如此多的技术，我们往往会产生疑惑：到底该如何选择？这里有一个建议：在充分理解自身需求的基础上，挑选最合适的。

比如要实现这样一个任务：有 10 亿条用户画像数据，里面记录了用户的性别、年龄、地域等信息，输入一个用户包，只包含用户账号，数量平均百万级，最多 1000 万，希望查询这些用户的性别、年龄、地域等属性的分布信息。

面对这样的问题，如果我们贸然找一个开源系统部署上，刚开始的学习成本高不说，即便可以用，最后也不一定能证明这个方案到底是好还是不好。因为从来没有最好的方案，只有当前最合适的方案，不但要跟其他方案比，还要跟问题本身的需求比。比如，一开始就有一些有经验的前辈建议使用 Solr 或 ElasticSearch，这两个系统固然非常强大，但大家都知道是为搜索引擎这样的场景设计的，而不是为这个问题场景设计的，中间难免有些匹配度的问题，不一定最合适。而且，如果不分析一下面对的问题场景，那么在研究 Solr 或 ElasticSearch 时也没有针对性，只能“全面开花”，结果必然费时费力，可能还没有好的结果。要知道，单 Solr 的文档就有几百页，全部看完至少需要一周时间。

因此我们先来分析一下当前的问题。如果所有数据都在数据库中，那么这个计算可以用 SQL 实现（以统计“性别”属性为例，其他与此类似）：

```
select
    性别, count(1)
from
    用户号码包表(1000W) join 画像表(10 亿)
group by
    性别
```

显然计算的瓶颈在于 join 操作，join 之后需要处理的数据量从 10 亿下降到 1000 万，这可以大大降低后续的聚合统计的成本。而对于 join 操作来说，如果两边的数据都已经排好序，那么 join 的算法复杂度只是 $O(n)$，非常低。由于 10 亿条画像数据更新频率很低，我们可以提前做好排序预处理，因此最理想的计算过程可以分为这么几步：

(1) 对输入的用户号码包进行排序；

(2) 排序后的号码包与已经有序的画像数据进行 join，筛选出号码包的属性；

(3) 对筛选后的少量数据进行聚合统计。

有了对问题的理解，我们再简单对比一下各种方案的优劣（使用相同的机器，集群数量为10台），参见下表。

方　案	实现过程	总用时	缺　点
Hive集群 (10台)	(1) 画像数据和号码包都存储为一张表； (2) 对两张表进行join操作，筛选出号码包的属性； (3) 对每个属性进行统计	>1小时	机器资源占用多，维护成本高
Spark SQL (10台)	同Hive	10分钟	
Spark MR (10台)	思路同上，但可以让画像数据提前排好序，并且两张表使用相同的分区策略，这样可以提升join的效率	8分钟	
Bash (单机单核)	(1) 对号码包进行排序； (2) 号码包文件与有序的画像文件进行join，筛选出号码包的属性； (3) 对筛选出的结果按维度分别统计	12分钟	单点
Bash (单机多核)	思路同上，但把大文件拆成小文件，并发执行	3分钟 磁盘替代CPU成为瓶颈，如果有多个物理磁盘，性能可以进一步提升至1分钟	
从零开发一个专有分布式系统 (3~10台)	思路同上，但所有的操作都在内存中，可能需要多台机器	<0.5分钟	开发成本太高 对开发人员要求高
Solr (3台机器)	(1) 设置集群为多shard结构，画像数据提前导入； (2) 新的画像数据作为新的字段导入； (3) 使用join操作和Facet特性对各个维度分别统计	1分钟	没有筛选，每个维度的查询都搜索了所有10亿画像数据 Facet特性必须指定一个UniqueKey，导致号码包导入时需要多处理一个UniqueKey字段

通过分析不难发现，有很多技术可以解决这个问题，连最不起眼的 bash 都可以，而且单机的性能就接近甚至超过 10 台 Spark 集群的性能。如果单独开发一个专有系统，性能更是会得到指数级提升。

然而，通过这些对比依然不足以确定我们的最终方案，因为对于需求本身还有很多疑问，比如系统的访问频率、扩容要求、需求变更，甚至还有团队的研发能力等。如果回答了这些问题，那么最终的决策只是水到渠成的事情。

本书内容

全书共分 8 章，外加一篇附录。前 4 章介绍 Spark 本身，包括部署、工作机制、内核等。全书的重点在第 5 章~第 8 章，每章不但深入浅出地介绍 Spark 的一个功能模块，而且包含一个实战项目，项目利用国内互联网的真实数据为案例，搭建一个产品和平台输出。这些例子每个都可以是一个独立的大项目。这在 BAT 等公司中，动辄都是有几十上百人的项目团队，而笔者基于 Spark 快速搭建了一个原型，为创业者提供了很好的入门示例。此外，书中还详细介绍了各种可能遇到的实战问题，比如大数据环境下的配置设置、程序的调优等。随书带一键安装脚本，以便为初学者提供很多帮助。

- 第 1 章概述大数据的发展状况，以及 Spark 的起源、特点、优势、未来等。
- 第 2 章介绍 Spark 部署和编程。作者首先带领读者在本地单机下体验 Spark 的基本操作，然后部署一个包括 ZooKeeper、Hadoop、Spark 的可实际应用的高可用集群。并且，这一章还介绍了笔者合作开发的一个自动化部署工具，最后介绍了 Spark 编程的基础知识，以及如何打包提交至集群上运行。
- 第 3 章介绍 Spark 底层的工作机制，包括调度管理、内存管理、容错机制、监控管理以及 Spark 程序配置管理，这对理解 Spark 程序的运行非常有帮助。
- 第 4 章深入 Spark 内核，并结合源码，介绍了核心结构 RDD、RDD 对象的 Transformation 和 Action 操作是如何实现的、`SparkContext` 对象及初始化过程、DAG 调度的工作流程。了解这些内容可以帮助读者编写出高质量的 Spark 程序代码。
- 第 5 章介绍 Spark SQL，可以代替 Hive，用于搭建一个企业级的数据仓库。案例基于淘宝的电商数据建立电商数据仓库，并以日常运营工作为例，通过电商数据库分析电商运营中的各类问题。
- 第 6 章介绍 Spark 实时流式计算，类似于 Storm，但吞吐量方面更有优势。案例是基于一个站点的 Web 日志建立一个类似百度统计的实时统计系统，是各种实时系统典型的参考例子。
- 第 7 章介绍 Spark 的图计算。案例基于新浪微博 2000 万的关系链数据，讲解了如果利用图计算来实现社交关系链的挖掘，比如闺蜜的发现、粉丝团伙的发现等。
- 第 8 章介绍 Spark 的机器学习库。案例基于某个搜索引擎的点击日志，建立了一个搜索广告点击率预估系统。广告点击率预估是各家互联网系统的核心系统，公开的实战项目不多。
- 附录为 Scala 语言参考，方便第一次接触 Scala 语言的读者快速上手，体验 Spark 编程。

致谢

本书由陈欢和林世飞一起合作编著，大家年龄相仿，志同道合，牺牲了很多陪家人的时间才完成本书的撰写，非常感谢家人的支持。还要特别感谢腾讯企业云的“攻城之地”活动提供免费服务器，感谢李冠灿为服务器使用提供很多便利，感谢李学凯、余衍炳等人审阅了部分章节，提

供了许多宝贵意见。感谢王流斌在如何使用 KDD2012 数据集方面提供了不少建议。感谢腾讯公司提供的学习和成长环境，以及各位领导对于我们撰写本书的鼓励和支持。感谢 UCloud 的 CEO 季昕华提供宝贵建议，并撰写“推荐序”。成稿期间，图灵出版公司的王军花编辑和毛倩倩编辑给予了极其详细的改进意见，在此表示由衷的感谢。

由于作者水平有限，书中难免有错误和不当之处，欢迎专家和读者给予批评指正，来函请发至 sparkbestpractices@qq.com。

2016 年 2 月

目　录