Spark
亚太研究院

Spark 官方文档中文翻译

快速入门

翻译者　唐海东

Spark 官方文档翻译团成员

前 言

世界上第一个Spark 1.1.0 中文文档问世了！

伴随着大数据相关技术和产业的逐步成熟，继Hadoop之后，Spark技术以集大成的无可比拟的优势，发展迅速，将成为替代Hadoop的下一代云计算、大数据核心技术。

Spark是当今大数据领域最活跃最热门的高效大数据通用计算平台，基于RDD，Spark成功的构建起了一体化、多元化的大数据处理体系，在“One Stack to rule them all”思想的引领下，Spark成功的使用Spark SQL、Spark Streaming、MLLib、GraphX近乎完美的解决了大数据中Batch Processing、Streaming Processing、Ad-hoc Query等三大核心问题，更为美妙的是在Spark中Spark SQL、Spark Streaming、MLLib、GraphX四大子框架和库之间可以无缝的共享数据和操作，这是当今任何大数据平台都无可匹敌的优势。

在实际的生产环境中，世界上已经出现很多一千个以上节点的Spark集群，以eBay为例，eBay的Spark集群节点已经超过2000个，Yahoo 等公司也在大规模的使用Spark，国内的淘宝、腾讯、百度、网易、京东、华为、大众点评、优酷土豆等也在生产环境下深度使用Spark。2014 Spark Summit上的信息，Spark已经获得世界 20 家顶级公司的支持，这些公司中包括Intel、IBM等，同时更重要的是包括了最大的四个Hadoop发行商，都提供了对Spark非常强有力的支持。

与Spark火爆程度形成鲜明对比的是Spark人才的严重稀缺，这一情况在中国尤其严重，这种人才的稀缺，一方面是由于Spark技术在 2013、2014 年才在国内的一些大型企业里面被逐步应用，另一方面是由于匮乏Spark相关的中文资料和系统化的培训。为此，Spark亚太研究院和 51CTO联合推出了“Spark亚太研究院决胜大数据时代 100 期公益大讲堂”，来推动Spark技术在国内的普及及落地。

具体视频信息请参考 http://edu.51cto.com/course/course_id-1659.html

与此同时，为了向Spark学习者提供更为丰富的学习资料，Spark亚太研究院发起并号召，结合网络社区的力量构建了Spark中文文档专家翻译团队，历经 1 个月左右的艰苦努力和反复修改，Spark中文文档V1.1 终于完成。尤其值得一提的是，在此次中文文档的翻译期间，Spark官方团队发布了Spark 1.1.0 版本，为了让学习者了解到最新的内容，Spark中文文档专家翻译团队主动提出基于最新的Spark 1.1.0 版本，更新了所有已完成的翻译内容，在此，我谨代表Spark亚太研究院及广大Spark学习爱好者向专家翻译团队所有成员热情而专业的工作致以深刻的敬意！

当然，作为世界上第一份相对系统的Spark中文文档，不足之处在所难免，大家有任何建议或者意见都可以发邮件到marketing@sparkinchina.com ;同时如果您想加入Spark中文文档翻译团队，也请发邮件到marketing@sparkinchina.com进行申请；Spark中文文档的翻译是一个持续更新的、不断版本迭代的过程，我们会尽全力给大家提供更高质量的Spark中文文档翻译。

最后，也是最重要的，请允许我荣幸的介绍一下我们的Spark中文文档第一个版本翻译的专家团队成员，他们分别是（排名不分先后）:

- 傅智勇, 《快速开始(v1.1.0)》（和唐海东翻译的是同一主题，大家可以对比参考）
- 吴洪泽,《Spark机器学习库 (v1.1.0)》（其中聚类和降维部分是蔡立宇翻译）
- 武扬，《在Yarn上运行Spark (v1.1.0)》《Spark 调优(v1.1.0)》
- 徐骄，《Spark配置(v1.1.0)》《Spark SQL编程指南(v1.1.0)》（Spark SQL和韩保礼翻译的是同一主题，大家可以对比参考）
- 蔡立宇，《Bagel 编程指南(v1.1.0)》
- harli，《Spark 编程指南 (v1.1.0)》
- 吴卓华，《图计算编程指南(1.1.0)》
- 樊登贵，《EC2(v1.1.0)》《Mesos(v1.1.0)》
- 韩保礼，《Spark SQL编程指南(v1.1.0)》（和徐骄翻译的是同一主题，大家可以对比参考）
- 颜军，《文档首页(v1.1.0)》
- Jack Niu，《Spark实时流处理编程指南(v1.1.0)》
- 俞杭军，《sbt-assembly》《使用Maven编译Spark(v1.1.0)》
- 唐海东，《快速开始(v1.1.0)》（和傅智勇翻译的是同一主题，大家可以对比参考）
- 刘亚卿，《硬件配置(v1.1.0)》《Hadoop 第三方发行版(v1.1.0)》《给Spark提交代码(v1.1.0)》
- 耿元振《集群模式概览(v1.1.0)》《监控与相关工具(v1.1.0)》《提交应用程序(v1.1.0)》
- 王庆刚，《Spark作业调度(v1.1.0)》《Spark安全(v1.1.0)》
- 徐敬丽，《Spark Standalone 模式 (v1.1.0)》

另外关于Spark API的翻译正在进行中，敬请大家关注。

Life is short, You need Spark!

Spark亚太研究院院长 王家林
2014 年 10 月

Spark 亚太研究院决胜大数据时代 100 期公益大讲堂

简 介

作为下一代云计算的核心技术，Spark性能超Hadoop百倍，算法实现仅有其 1/10 或 1/100,是可以革命Hadoop的目前唯一替代者，能够做Hadoop做的一切事情，同时速度比Hadoop快了 100 倍以上。目前Spark已经构建了自己的整个大数据处理生态系统，国外一些大型互联网公司已经部署了Spark。甚至连Hadoop的早期主要贡献者Yahoo现在也在多个项目中部署使用Spark；国内的淘宝、优酷土豆、网易、Baidu、腾讯、皮皮网等已经使用Spark技术用于自己的商业生产系统中，国内外的应用开始越来越广泛。Spark正在逐渐走向成熟，并在这个领域扮演更加重要的角色，刚刚结束的 2014 Spark Summit上的信息，Spark已经获得世界 20 家顶级公司的支持，这些公司中包括Intel、IBM等，同时更重要的是包括了最大的四个Hadoop发行商都提供了对非常强有力的支持Spark的支持.

鉴于Spark的巨大价值和潜力，同时由于国内极度缺乏Spark人才，Spark亚太研究院在完成了对Spark源码的彻底研究的同时，不断在实际环境中使用Spark的各种特性的基础之上，推出了Spark亚太研究院决胜大数据时代 100 期公益大讲堂，希望能够帮助大家了解Spark的技术。同时，对Spark人才培养有近一步需求的企业和个人，我们将以公开课和企业内训的方式，来帮助大家进行Spark技能的提升。同样，我们也为企业提供一体化的顾问式服务及Spark一站式项目解决方案和实施方案。

Spark亚太研究院决胜大数据时代 100 期公益大讲堂是国内第一个Spark课程免费线上讲座，每周一期，从 7 月份起，每周四晚 20:00-21:30，与大家不见不散！老师将就Spark内核剖析、源码解读、性能优化及商业实战案例等精彩内容与大家分享，干货不容错过！

时间：从 7 月份起，每周一期，每周四晚 20:00-21:30
形式：腾讯课堂在线直播
学习条件：对云计算大数据感兴趣的技术人员
课程学习地址：http://edu.51cto.com/course/course_id-1659.html

快速入门（翻译者：唐海东）

Quick Start，原文档链接：http://spark.apache.org/docs/latest/quick-start.html

目录

本教程对Spark的使用提供了一个简单介绍。 首先，通过Spark Shell上的交互式操作（Spark Shell 支持Python或Scala两种语言），让我们对Spark 的API有个初步了解；然后演示如何使用Java、 Scala和Python三种语言调用Spark API，编写一个独立的应用程序。 获取更详细的资料,请参阅编程指南。

想要一步一步完成本指南中的教程，你必须从 Spark官方网站下载一个Spark的发行包。虽然在本教程中我们不会使用到HDFS的内容 ,但仍然需要你下载一个Hadoop软件包，任意版本的都可以。

1. 初识 Spark Shell

1.1 基础知识

Spark Shell 既是学习 Spark API 的一个简单方法，也是一个功能强大的交互式数据分析工具。Spark Shell 的执行需要有 Scala 或 Python 运行时环境支持（Scala 是一种运行在 Java 虚拟机上的语言，并支持使用现有的 Java 库）。 要运行 Spark Shell，请在 Spark 安装目录下执行下面的命令：

- **Scala**

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

- **Python**

```
./bin/pyspark
```

Spark的主要概念是分布式数据集(RDD),也就是所谓的弹性分布式数据集(Resilient Distributed Dataset)。 RDD可以由Hadoop的InputFormats来创建（如HDFS文件），也可以从其他RDD 转换而来。 为了让你有个感性的认识，让我们从Spark安装目录中的README文本文件来创建一个RDD：

- **Scala**

```
scala> val textFile = sc.textFile("README.md")

textFile: spark.RDD[String] = spark.MappedRDD@2ee9b6e3
```

- **Python**

```
>>> textFile = sc.textFile("README.md")
```

每个 RDD 都有很多方法或者函数（以下统称为函数），根据这些者函数的返回值，可以将其分为两类：动作函数（*action*）和转换函数（ transformation）； 动作函数通常返回一个或者一组标量数值，而转换函数通常返回一个或者一组新的 RDD。 让我们先从几个动作函数开始，小试伸手：

- **Scala**

```
scala> textFile.count() // 统计 RDD 中文本行数

res0 : Long = 126

scala> textFile.first() // 获取第一行的内容

res1 : String = # Apache Spark
```

- **Python**

```
>>> textFile.count () # 统计RDD中文本行数

126

>>> textFile.first () # 获取第一行的内容

u'# Apache Spark'
```

再来试试转换函数。 过滤器函数（filter)按照指定的过滤条件返回一个新的 RDD，该 RDD 是对文件过滤后的一个子集。

- **Scala**

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

linesWithSpark : spark.RDD[String] = spark.FilteredRDD@7dd4af09
```

- **Python**

```
 >>> linesWithSpark = textFile.filter(lambda line : "Spark" in line )
```

我们也可以将动作函数和转换函数串联起来使用：

- **Scala**

```
scala>textFile.filter(line=>line.contains("Spark")).count() // 多少行包含了单词"Spark"?

res3 : Long = 15
```

- **Python**

```
>>> textFile.filter(lambda line:"Spark" in line).count()     # 多少行包含了单词"Spark"?

15
```

1.2 深入 RDD

RDD 动作和变换函数可以进行一些更复杂的计算。 比方说，我们要找出包含最多单词的行：

- **Scala**

```
scala> textFile.map(line => line.split(" ").size).reduce((a ,b) => if(a>b) a else b )

res4 : Long = 15
```

- **Python**

```
>>> textFile.map(lambda line : len (line.split())).reduce(lambda a, b : a if ( a > b) else b )

15
```

上述示例中 map 函数先是将一个文本行映射为一个整数值（该行的单词数），这个转换会创建一个新的 RDD。 Reduce 函数则是在这个新的 RDD 上查找最大值。 上述代码中 map 和 reduce 函数的参数是 Scala 函数的返回值（闭包），并可以使用任何语言功能或 Scala/ Java 库。我们可以方便地调用其他地方声明的函数 比如我们可以调用 Math.max() 函数，从而使上述代码更容易理解：

- **Scala**

```
scala> import java.lang.Math

import java.lang.Math

scala> textFile.map(line =>line.split(" ").size).reduce((a,b)=> Math.max(a, b
 ))

res5 : Int = 15
```

- **Python**

```
>>> def max ( a , b ):
... if a > b :
...     return a
... else :
...     return b
...
>>> textFile.map(lambda line : len(line.split())).reduce(max)
15
```

目前最常见的数据流模式是 Hadoop 采用的 MapReduce。事实上 Spark 也可以很容易的实现

- **Scala**

```
scala>val wordCounts=textFile.flatMap(line =>line.split(" ")).map(word=>(word
 , 1)).reduceByKey((a ,b)=>a + b)

wordCounts:spark.RDD[( String,Int)]=spark.ShuffledAggregatedRDD@71 f027b8
```

- **Python**

```
>>> wordCounts = textFile.flatMap(lambda line : line.split()).map(lambda word
 : ( word , 1 )).reduceByKey(lambda a , b : a + b )
```

在这里，我们结合使用了 flatMap ， map 和 reduceByKey 三个转换函数计算每个单词在文件中出现的次数，并生成一个 RDD,其类型为（String，Int）。 要想在 Shell 查看最终的结果，我们可以使用一个动作函数 collect：

- **Scala**

```
scala> wordCounts.collect ()

res6 : Array[(String, Int)] = Array ((means , 1 ), (under , 2 ), (this , 3 ),
(Because, 1 ), (Python, 2 ), (agree, 1 ), (cluster, 1 ), ...)
```

- **Python**

```
>>> wordCounts.collect()

[(u'and', 9), (u'A', 1), (u'webpage', 1), (u'README', 1), (u'Note', 1), (u'"lo
cal"', 1), (u'variable', 1), ... ]
```

1.3 缓存

Spark 也支持将数据缓存到一个集群的内存中。 当我们需要重复的访问数据，比如查询一些体量虽小却非常热门的数据集，或者正在运行类似 PageRank 的迭代算法的时候，缓存对于提高应用的性能就至关重要了。一个简单的例子，让我们把 linesWithSpark 数据集进行缓存：

- **Scala**

```
scala > linesWithSpark.cache()

res7 : spark.RDD [String ] = spark.FilteredRDD@17e51082

scala > linesWithSpark.count()

res8 : Long = 15

scala > linesWithSpark.count ()

res9 : Long = 15
```

- **Python**

```
>>> linesWithSpark.cache()

>>> linesWithSpark.count()

15
```

```
>>> linesWithSpark.count()
15
```

虽然用 Spark 浏览和缓存一个只有 100 行的文本文件让人感觉就像高射炮打蚊子，但 Spark 的这些功能，的确可以用于非常大的数据集，那些跨越数十甚至数百个节点的集群。 当然你也可以连接到一个大的集群，使用 Spark Shell 完成如上交互操作. Spark Shell 如何连接到一个集群，请参考编程指南 。

2. 独立的应用程序

现在，假设我们要使用 Spark API 来开发一个独立应用程序。 我们将使用 Scala（需要 SBT 支持），Java（需要 Maven 支持），和 Python 三种语言，一步一步完成一个简单应用程序。

2.1 Scala

接下来，我们将用 Scala 创建一个非常简单的 Spark 应用。由于它非常简单，我们不妨就直接将它命名为简单应用(SimleApp.scala)

```
/* SimpleApp.scala */
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object SimpleApp {
  def main(args: Array[String]) {
    val logFile = "YOUR_SPARK_HOME/README.md" //可以是你系统上任意的英文txt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))
```

```
  }
}
```

这个程序只是统计 Spark Readme 文件中包含字母“a”和字母“b”的行数。上述代码中有一点需要注意，你需要用 Spark 安装位置取代 YOUR_SPARK_HOME。 与前面的运行在 Spark Shell 中例子不同，作为程序开发的一部分，我们需要自己初始化一个 SparkContext。（实际上，Spark Shell 在启动后就为自己初始化了一个 SparkContext）

我们首先创建了一个包含我们的应用程序的信息的SparkConf对象 conf 然后将conf传递给 SparkContext 构造函数。

由于我们的应用程序依赖于 Spark API，所以我们需要在 SBT 配置文件 simple.sbt 中添加这一依赖:

```
name := "Simple Project"

version := "1.0"

scalaVersion := "2.10.4"

libraryDependencies += "org.apache.spark" %% "spark-core" % "1.1.0-SNAPSHOT"
```

为了让 SBT 正常工作 ,我们需要让 SimpleApp.scala 和 simple.sbt 文件所在目录符合 SBT 的标准目录结构。 目录准备就绪后，我们可以创建一个包含应用程序代码的 JAR 包，然后使用 spark-submit 脚本来运行我们的程序。

```
# 程序的目录结构应该跟下面的一样

$ find .

.

./simple.sbt

./src

./src/main

./src/main/scala

./src/main/scala/SimpleApp.scala


# 将你的应用打成一个jar包

$ sbt package

...

[info] Packaging {..}/{..}/target/scala-2.10/simple-project_2.10-1.0.jar
```

```
# 使用 spark-submit 命令来执行你的应用
$ YOUR_SPARK_HOME/bin/spark-submit \
  --class "SimpleApp" \
  --master local[4] \
  target/scala-2.10/simple-project_2.10-1.0.jar
...
Lines with a: 46, Lines with b: 23  # 返回结果：a: 46 行, b: 23 行
```

2.2 Java

我们将使用 Maven 来编译创建的应用程序，并生成相应的 jar 文件，但你也可以使用任何类似的编译系统。

我们将创建一个非常简单的 Spark 应用， SimpleApp.java ：

```
/* SimpleApp.java */
import org.apache.spark.api.java.*;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.function.Function;
public class SimpleApp {
  public static void main(String[] args) {
    String logFile = "YOUR_SPARK_HOME/README.md";//可以是你系统上任意的英文txt文件
    SparkConf conf = new SparkConf().setAppName("Simple Application");
    JavaSparkContext sc = new JavaSparkContext(conf);
    JavaRDD<String> logData = sc.textFile(logFile).cache();
    long numAs = logData.filter(new Function<String, Boolean>() {
      public Boolean call(String s) { return s.contains("a"); }
    }).count();
    long numBs = logData.filter(new Function<String, Boolean>() {
      public Boolean call(String s) { return s.contains("b"); }
    }).count();
```

```
    System.out.println("Lines with a: " + numAs + ", lines with b: " + numBs);
  }
}
```

这个程序只是统计 Spark Readme 文件中包含字母“a”和字母“b”的行数。上述代码中有一点需要注意，那就是你需要用 Spark 安装位置取代 YOUR_SPARK_HOME。 对比前面的 Scala 范例 我们使用了一个对 Java 更加友好的构造函数来初始化 SparkContext。我们生成了 RDD(JavaRDD 类型),然后执行了一系列转换算法。最后，传给 Spark 的作为参数的函数，是通过 spark.api.java.function.Function 的子类实现的。 编程指南中详细描述了个中细节

要运行这个程序，我们需要编写一个 Maven 的 pom.xml 文件。

```
<project>
  <groupId>edu.berkeley</groupId>
  <artifactId>simple-project</artifactId>
  <modelVersion>4.0.0</modelVersion>
  <name>Simple Project</name>
  <packaging>jar</packaging>
  <version>1.0</version>
  <dependencies>
    <dependency> <!-- Spark依赖-->
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.10</artifactId>
      <version>1.1.0-SNAPSHOT</version>
    </dependency>
  </dependencies>
</project>
```

程序文件所在目录还必须符合 Maven 的标准目录结构:

```
$ find .
./pom.xml
./src
./src/main
./src/main/java
```

```
./src/main/java/SimpleApp.java
```

现在我们可以使用 Maven 来对你的应用进行打包、运行

```
# 将你的应用进行打包
$ mvn package
...
[INFO] Building jar: {..}/{..}/target/simple-project-1.0.jar


# 使用spark-submit命令来运行你的应用
$ YOUR_SPARK_HOME/bin/spark-submit \
  --class "SimpleApp" \
  --master local[4] \
  target/simple-project-1.0.jar
...
Lines with a: 46, Lines with b: 23  # 返回结果：a: 46 行, b: 23 行
```

2.3 Python

现在我们来演示如何使用 Python API (PySpark)编写一个独立应用

作为范例我们创建一个简单的 Spark 应用,并命名为 SimpleApp.py

```
"""SimpleApp.py"""
from pyspark import SparkContext


logFile = "YOUR_SPARK_HOME/README.md"  # 可以是你系统上的任意英文txt文件
sc = SparkContext("local", "Simple App")
logData = sc.textFile(logFile).cache()


numAs = logData.filter(lambda s: 'a' in s).count()
numBs = logData.filter(lambda s: 'b' in s).count()
print "Lines with a: %i, lines with b: %i" % (numAs, numBs)
```

这个程序只是统计某个文本文件中包含字母“a”和字母“b”的行数。 上述代码中有一点需要注意，你需要用 Spark 安装位置取代 YOUR_SPARK_HOME。 对比前面的 Scala 范例，我们使用 SparkContext 来创建 RDD。然后将 Python 函数传递给了 Spark。对于应用中使用到的自定义类或者第三方类库，我们可以将他们打成一个.zip 的包，然后通过 --py-files 参数传递给 spark-submit(详细信息，参考 spark-submit --help 命令)。SimpleApp 是在太简单，我们不需要添加任何依赖.

要运行我们的应用，使用脚本 bin/spark-submit:

```
# 使用 spark-submit 命令来运行你的应用

$ YOUR_SPARK_HOME/bin/spark-submit \

  --master local[4] \

  SimpleApp.py

...

Lines with a: 46, Lines with b: 23  //运行结果: a:46 行, b:23 行
```

更上一层

恭喜! 你已成功运行的你的第一个 Spark 应用程序！

- 想要深入了解Spark的API，可以从Spark编程指南着手，或参阅“编程指南”中的其他组件。
- 想要在一个集群上运行的 Spark 应用程序，请参考部署概述 。
- 最后，Spark 安装目录的 examples 目录下包含几个示例程序（有 Scala ， Java ， Python 三种语言的示例）。 您可以运行它们，如下所示：

```
#Scala 及 Java 用户，使用命令 run-example

     ./bin/run-example SparkPi

#Python 用户，使用命令 spark-submit

     ./bin/spark-submit examples/src/main/python/pi.py
```

Spark 亚太研究院

Spark 亚太研究院是中国最专业的一站式大数据 Spark 解决方案供应商和高品质大数据企业级完整培训与服务供应商，以帮助企业规划、架构、部署、开发、培训和使用 Spark 为核心，同时提供 Spark 源码研究和应用技术训练。针对具体 Spark 项目，提供完整而彻底的解决方案。包括 Spark 一站式项目解决方案、Spark 一站式项目实施方案及 Spark 一体化顾问服务。

官网： www.sparkinchina.com

近期活动

- 2014 年亚太地区规格最高的 Spark 技术盛会！
- 面向大数据、云计算开发者、技术爱好者的饕餮盛宴！
- 云集国内外 Spark 技术领军人物及灵魂人物！
- 技术交流、应用分享、源码研究、商业案例探询！

时间：2014 年 12 月 6-7 日
地点：北京珠三角万豪酒店
Spark亚太峰会网址：http://www.sparkinchina.com/meeting/2014yt/default.asp

- 如果你是对 Spark 有浓厚兴趣的初学者，在这里你会有绝佳的入门和实践机会！

- 如果你是 Spark 的应用高手，在这里以“武”会友，和技术大牛们尽情切磋！
- 如果你是对 Spark 有深入独特见解的专家，在这里可以尽情展现你的才华！

比赛时间：
2014 年 9 月 30 日—12 月 3 日
Spark开发者大赛网址：http://www.sparkinchina.com/meeting/2014yt/dhhd.asp

视频课程：

《大数据 Spark 实战高手之路》 国内第一个 Spark 视频系列课程

从零起步，分阶段无任何障碍逐步掌握大数据统一计算平台 Spark，从 Spark 框架编写和开发语言 Scala 开始，到 Spark 企业级开发，再到 Spark 框架源码解析、Spark 与 Hadoop 的融合、商业案例和企业面试，一次性彻底掌握 Spark，成为云计算大数据时代的幸运儿和弄潮儿，笑傲大数据职场和人生！

- 第一阶段：熟练的掌握 Scala 语言
  课程学习地址：http://edu.51cto.com/pack/view/id-124.html
- 第二阶段：精通 Spark 平台本身提供给开发者 API
  课程学习地址：http://edu.51cto.com/pack/view/id-146.html
- 第三阶段：精通 Spark 内核
  课程学习地址：http://edu.51cto.com/pack/view/id-148.html
- 第四阶段：掌握基于 Spark 上的核心框架的使用
  课程学习地址：http://edu.51cto.com/pack/view/id-149.html
- 第五阶段：商业级别大数据中心黄金组合：Hadoop+ Spark
  课程学习地址：http://edu.51cto.com/pack/view/id-150.html
- 第六阶段：Spark 源码完整解析和系统定制
  课程学习地址：http://edu.51cto.com/pack/view/id-151.html

近期公开课：

《决胜大数据时代：Hadoop、Yarn、Spark 企业级最佳实践》

集大数据领域最核心三大技术：Hadoop 方向 50%：掌握生产环境下、源码级别下的 Hadoop 经验，解决性能、集群难点问题；Yarn 方向 20%：掌握最佳的分布式集群资源管理框架，能够轻松使用 Yarn 管理 Hadoop、Spark 等；Spark 方向 30%：未来统一的大数据框架平台，剖析 Spark 架构、内核等核心技术，对未来转向 SPARK 技术，做好技术储备。课程内容落地性强，即解决当下问题，又有助于驾驭未来。
开课时间： 2014 年 10 月 26-28 日北京、2014 年 11 月 1-3 日深圳

咨询电话：4006-998-758

QQ 交流群： 1 群：317540673 （已满）
2 群：297931500

微信公众号：spark-china