亚太研究院

Spark 官方文档翻译

Spark 快速入门

翻译者　傅智勇

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

Spark 快速入门(V1.1.0)

（翻译者：傅智勇）

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

目录

第 1 章 快速开始

本章提供了Spark快速入门介绍。首先，我们会通过Spark交互式Shell（以Python或Scala 模式）来介绍Spark API，然后介绍如何用Java，Scala和Python写单机应用程序。如果想要了解更多，可以参考 编程指导。

参考本指南之前，首先需要从 Spark官网下载已经打包好的Spark发布版本。由于我们不使用HDFS，所以可以下载任意版本的Hadoop包。

1.1 用 Spark Shell 交互式分析

1.1.1 基础知识

Spark Shell 提供了一种简单的方式来学习 Spark API，同时也是一种交互式数据分析的强大工具。目前可以获得 Scala（Scala 是运行在 Java 虚拟机之上，因而可以在 Scala 中很好的调用 Java 库）和 Python 两种语言的 Spark Shell。切换到 Spark 目录下，运行以下命令可以启动 Spark Shell。

Scala:

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

Python:

```
./bin/pyspark
```

Spark 主要抽象是一种称为弹性分布式数据集（RDD）的分布式对象集合。RDD 能够从 Hadoop InputFormats（比如 HDFS 文件）所创建或者转换成其他的 RDD。下面我们以 Spark 源代码包中的 README 文件内容来创建一个新的 RDD：

Scala:

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

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

Python:

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

RDD 有操作和转换两种对象，操作的返回值是一些操作方法，而转换的返回值是指向新的 RDD 引用。接下来，我们会以一些操作方法开始介绍 RDD。

Scala：

```
scala> textFile.count() // 该RDD项目数目

res0: Long = 126

scala> textFile.first() // 该RDD中第一项

res1: String = # Apache Spark
```

Python：

```
>>> textFile.count() # 该RDD项目数目

126

>>> textFile.first() # 该RDD中第一项

u'# Apache Spark'
```

接下来，我们将使用 filter 转换来返回包含文件项子集的一个新 RDD。

Scala：

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

linesWithSpark: spark.RDD[String] = spark.FilteredRDD@7dd4a
f09
```

Python：

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

我们也能将操作和转换链接起来使用，例如：

Scala：

```
scala> textFile.filter(line => line.contains("Spark")).c
ount() // 包含"Spark"的行数

res3: Long = 15
```

Python：

```
>>> textFile.filter(lambda line: "Spark" in line).count()
 # 包含"Spark"的行数

15
```

1.1.2 更多有关 RDD 操作

RDD 操作和转换可以用于更复杂的操作。我们可以通过以下方式获得包括最多字符的行号：

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
```

首先将每行 map 成一个整形值，创建一个新的 RDD。然后在新生成的 RDD 上调用 reduce 来计算出行中最大的字符数。map 和 reduce 的参数值是 Scala 函数字面值(闭包)，可以使用任何语言特征或 Scala/Java 库。例如，我们可以在其他地方简单地调用已经申明的函数。我们将使用 Math.max()函数使这行代码更容易理解：

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)1
```

```
5
```

正如 Hadoop 流行的一种通用 MapReduce 数据流模式一样，Spark 也能够简单地实现 MapReduce 数据流：

Scala:

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

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

Python:

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

这里我们结合了 flatmap, map 和 reduceByKey 转换来计算文件中每个文字出现的次数作为(String, Int)对的 RDD。然后，我们可以在 shell 中通过调用 collect 操作计算出字符出现的次数：

Scala:

```
scala> wordCounts.collect()res6: Array[(String, Int)] = Array((me
ans,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'"local"', 1), (u'variable', 1),
 ...]
```

1.1.3 缓存

Spark 也支持将数据集迁移到集群范围的内存缓存。这在数据被重复访问的时候非常有用，例如当查询一个小的数据集“hot”或当运行一个类似于 PageRank 的迭代算法。我们将 linesWithSpark 数据集标记成已被缓存作为一个例子来说明缓存的使用和它的优势：

Scala:

```
scala> linesWithSpark.cache()res7: spark.RDD[String] = spark.Filter
edRDD@17e51082□scala> linesWithSpark.count()res8: Long = 15□scala>
linesWithSpark.count()res9: Long = 15
```

Python:

```
>>> linesWithSpark.cache()
>>> linesWithSpark.count()

15
>>> linesWithSpark.count()

15
```

也许使用Spark来探索和缓存一个100行的文本文件看似比较愚蠢。但值得注意的是，同样的函数也可以用于大型数据集中，甚至可以跨越几十个或几百个节点。正如编程指南所描述的那样，你也可以通过使用 bin/spark-shell 连到集群中，同样可以以交互式 shell 做到这一点。

1.2 单机应用程序

现在，我们想用 Spark API 来写一个单机应用程序。我们将用 Scala(SBT)，Java(Maven)和 Python 三种语言开始一个简单的应用程序。

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" // 应该是你文件系统的某
个文件

     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, num
Bs))

  }

}
```

Java:

```
/* SimpleApp.java */import org.apache.spark.api.java.*;import org.a
pache.spark.SparkConf;import org.apache.spark.api.java.function.Fun
ction;
public class SimpleApp {
  public static void main(String[] args) {
    String logFile = "YOUR_SPARK_HOME/README.md"; // 应该是你文件系统
的某个文件
    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);
  }

}
```

Python:

```
"""SimpleApp.py"""

from pyspark import SparkContext
logFile = "YOUR_SPARK_HOME/README.md"  # 应该是你文件系统的某个文件

sc = SparkContext("local", "Simple App")

logData = sc.textFile(logFile).cache()□numAs = logData.filter(lambd
a s: 'a' in s).count()numBs = logData.filter(lambda s: 'b' in s).co
```

```
unt()□print "Lines with a: %i, lines with b: %i" % (numAs, numBs)
```

该应用程序仅仅只计算 Spark README 文件中包含“a”的行数和包含“b”的行数。注意，你需要将代码中 YOUR_SPARK_HOME 替换成你本地 Spark 的安装目录。不如 Spark Shell 那么简单，Spark Shell 会自动初始化 SparkContext，而程序里面需要将 SparkContext 初始化代码作为程序的一部分。

我们将包含应用程序相关的信息 SparkConf 对象传递给 SparkContext 构造器。应用程序依赖于 Spark API，因此，我们也需要包含一个 sbt 配置文件 simple.sbt，该配置文件说明了 Spark 是一种依赖，该文件也添加了 Spark 所依赖的存储库。

Scala:

```
name := "Simple Project"□version := "1.0"□scalaVersion := "2.10.4"□
libraryDependencies += "org.apache.spark" %% "spark-core" % "1.0.2"
□resolvers += "Akka Repository" at "http://repo.akka.io/releases/"
```

Java:

```
<project>□  <groupId>edu.berkeley</groupId>□  <artifactId>simple-pr
oject</artifactId>□  <modelVersion>4.0.0</modelVersion>□  <name>Sim
ple Project</name>□  <packaging>jar</packaging>□  <version>1.0</ver
sion>□  <repositories>□    <repository>□      <id>Akka repository</
id>□      <url>http://repo.akka.io/releases</url>□    </repository>
□  </repositories>□  <dependencies>□    <dependency> <!-- Spark 依赖
 -->□      <groupId>org.apache.spark</groupId>□      <artifactId>sp
ark-core_2.10</artifactId>□      <version>1.0.2</version>□    </dep
endency>□  </dependencies></project>
```

Python:

```
name := "Simple Project"□version := "1.0"□scalaVersion := "2.10.4"□
libraryDependencies += "org.apache.spark" %% "spark-core" % "1.0.2"
□resolvers += "Akka Repository" at "http://repo.akka.io/releases/"
```

为了让 sbt 正常工作，我们需要根据指定的目录结构来布局 SimpleApp.scala 和 simple.sbt 文件。一旦位置放对，我们就可以将应用程序代码打包成一个 JAR 包，然后使用 spark-submit 脚本来运行打包好的应用程序。

Scala:

```
# 你目录结构应该如下
```

```
$ find .

.

./simple.sbt

./src

./src/main

./src/main/scala

./src/main/scala/SimpleApp.scala

# 将应用程序打包成 jar 包

$ sbt package

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

# 使用 spark-submit 运行应用程序

$ 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
```

Java:

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

现在，我们可以使用 Maven 来打包该应用程序，并且用 ./bin/spark-submit 来执行该应用程序。

```
# 将该应用程序打包成 jar 包

$ mvn package
...[INFO] Building jar: {..}/{..}/target/simple-project-1.0.jar
# 使用 spark-submit 运行应用程序

$ YOUR_SPARK_HOME/bin/spark-submit \□  --class "SimpleApp" \□  --ma
ster local[4] \□  target/simple-project-1.0.jar□...□Lines with a: 4
6, Lines with b: 23
```

Python:

```
# 使用 spark-submit 运行应用程序

$ YOUR_SPARK_HOME/bin/spark-submit \□  --master local[4] \□  Simple
App.py...Lines with a: 46, Lines with b: 23
```

1.3 下一步去哪

恭喜你，你已经成功运行了第一个 Spark 应用程序！

想要深入了解Spark API，可以查看 Spark变成指南，或者查看“编程指南”菜单栏查看其他的组件。

想要在集群中运行引用程序，可以到 部署概述。

最后，在 Spark examples 目录下（Scala，Java，Python）包括很多例子。有可以用以下方式运行这些例子：

```
./bin/run-example SparkPi

# 对于 Python 例子，直接使用 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

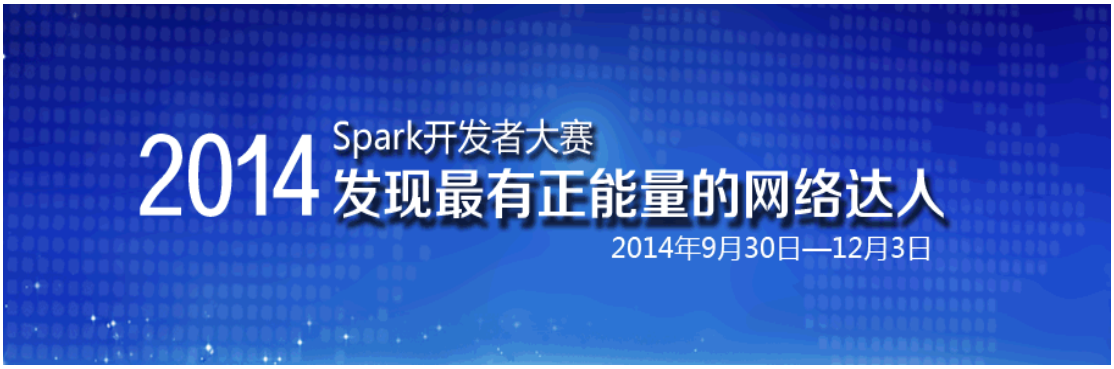
近期活动

- 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