



Spark 官方文档翻译

Spark 调优 (v1.1.0)

翻译者 武扬

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員，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，我们也为企业一体化顾问式服务及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 调优(v1.1.0)

(翻译者：武扬)

Tuning Spark , 原文档链接：<http://spark.apache.org/docs/latest/tuning.html>

目录

1. 数据序列化.....	6
2. 内存调优.....	7
2.1 确定内存的消耗.....	8
2.2 数据结构调优.....	8
2.3 序列化RDD存储.....	8
2.4 垃圾回收调优.....	9
2.4.1 估计垃圾回收的影响.....	9
2.4.2 缓存大小调优.....	9
2.4.3 垃圾回收调优进阶.....	9
3. 其他需要考虑的地方.....	10
3.1 并行度.....	10
3.2 Reduce任务的内存使用.....	11
3.3 广播 (Broadcasting) 较大的变量.....	11
4. 总结.....	11

1. 数据序列化

由于基于 Spark 的计算大多数是在内存中进行的，所以集群中的 CPU、网络带宽、内存等任何资源都可能成为 Spark 程序的瓶颈。通常如果数据正好存在内存中，那网络带宽就会成为瓶颈，但有时你还是需要做一些调优，就像用序列化的方法存储 RDDs 中所说的方法来减少内存使用。这篇指南将主要涉及两个话题：数据序列化，这个对于网络性能来说是关键，同时也能减少内存使用。另外一个话题是内存调优。同时我们还会涉及一些小的话题。

在所有的分布式应用中，序列化对性能至关重要。那些序列化速度慢或者占用大量内存的格式会明显拖慢计算速度。这通常是你为 Spark 应用优化的第一步。Spark 致力于在便捷性（允许你操作任意的 Java 类型）与性能之间取得平衡。所以提供两个序列化的类库：

- Java 序列化: Spark 默认的对象序列化方法是 Java 原生的 `ObjectOutputStream` 框架，你定义的任何类只要实现了 `java.io.Serializable` 的接口都可以正常的使用。你也可以通过继承 `java.io.Externalizable` 类来获得更好的序列化性能。Java 原生序列化方法虽然很方便，但是通常很慢。这导致许多类的序列化格式很大。
- Kryo 序列化: Spark 也可以使用 Kryo 的序列化库(版本号 2)以进行更快的序列化。Kryo 比 Java 原生序列化方法更为高效（可达 10 倍），但是并不支持所有实现了 `Serializable` 接口的类型，为了获得最好的效果，需要将要使用那些类注册后再使用。

在任务初始化的时候，可以通过 `SparkConf` 调用 `conf.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")` 来设置使用 Kryo。这个设置会影响工作节点之间混洗（shuffling）的数据和将 RDDs 序列化到磁盘。需要用户注册是不把 Kryo 作为默认的序列化方法的唯一原因，但是我们还是推荐你在任何网络要求较高的应用中试用一下。

Spark 会在很多常用的核心 Scala 类（均在 [Twitter chill](#) 库中的 `AllScalaRegistrar` 中）中自动包含 Kryo 序列化方法。

用 Kryo 注册你自己定义的类，你需要定义一个公共类继承 `org.apache.spark.serializer.KryoRegistrar` 并且设置 `spark.kryo.registrator` 来指向他，如下所示

```
import com.esotericsoftware.kryo.Kryo

import org.apache.spark.serializer.KryoRegistrar

class MyRegistrar extends KryoRegistrar {
```



```
override def registerClasses(kryo: Kryo) {  
    kryo.register(classOf[MyClass1])  
    kryo.register(classOf[MyClass2])  
}  
  
val conf = new SparkConf().setMaster(...).setAppName(...)  
conf.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")  
  
conf.set("spark.kryo.registrator", "mypackage.MyRegistrator")  
  
val sc = new SparkContext(conf)
```

[Kryo文档](#)中讲述了更多的高级注册选项，例如增加自定义序列化代码。

如果你的对象很大，你可以增加 `spark.kryo.serializer.buffer.mb` 配置选项。默认值为 2，但是这个值要足够大来保存你要序列化的最大的那个对象。

当然，如果你不注册你自定义的类，Kryo 仍然正常工作，但是他会 在每一个对象中保存类的全名，相当浪费空间。

2. 内存调优

内存使用的调优有如下三个注意事项：对象占用内存的大小（你可能会使整个数据匹配内存空间），访问对象的效率，以及垃圾回收的额外空间（如果频繁生成对象）。

默认情况下 Java 对象的访问速度很快，但是比数据本身会多消耗超过 2-5 倍的空间。这是由于如下几个原因造成的：

- 每一个不同的 Java 对象都有一个“对象头（object header）”，大概占用 16 个字节，包含一些如这个类的指针等信息。对于那些几乎不占空间的数据（比如 Int）来说，这比数据本身还要大。
- Java String 类型除了原始字符串数据大概还需要 40 个字节的额外空间（数据保存在 Char 数组中，同时保存数组的长度信息等），同时每一个字符需要两字节的空间，因为 String 类在内部使用 UTF-16 编码。因此，一个 10 字符的字符串至少占用 60 个字节。
- 还有一些像 HashMap 和 LinkedList 这样常见的集合类。使用列表型的数据结构，还有一些对象的封装来访问每一个元素。（例如 Map.Entry）。这个对象不仅仅是一个对象头，还包含指向下一个元素的指针（通常每个元素需要 8 个字节）。

- 对基本类型的集合通常需要“装箱”存储，例如 `java.lang.Integer`。

这一章节将会讨论如何确定你的对象所需的内存，以及如何优化他。可以通过改变数据结构的方法，或者将数据以有序的方式排列。我们还会涉及 Spark 的缓存大小的调优和 Java 的垃圾回收。

2.1 确定内存的消耗

衡量数据所需内存大小最好的方法是生成一个 RDD 并缓存，然后通过 `SparkContext` 的日志可以看到每一个分区的内存消耗。这样可以合计得到 RDD 的总大小。将会看到如下信息：

```
INFO BlockManagerMasterActor: Added rdd_0_1 in memory on mbk.local:50311 (size: 717.5 KB, free: 332.3 MB)
```

这表示 RDD 0 的分区 1 占消耗了 717.5 KB 内存。

2.2 数据结构调优

第一个减少内存消耗的方法是避免使用一些 Java 会增加额外开销的特性，比如基于指针的数据结构和对象的封装。有下面几种方法可以使用：

1. 设计数据结构的时候尽量使用数组和基本元素，避免使用 Java 或者 Scala 的集合类（例如 `HashMap`）。`fastutil` 库为基本类型提供一些高效的集合类于 Java 标准类库兼容。
2. 尽可能避免使用包含大量小对象和指针的嵌套数据结构。
3. 考虑使用数字型的 ID 或者枚举类型代替字符串类型作为键。
4. 如果你的内存不到 32GB，可以设置 JVM 的参数 `-XX:+UseCompressedOops` 将指针大小从 8 字节改为 4 字节。你可以将这个选项写入 `spark-env.sh`。

2.3 序列化 RDD 存储

如果经过优化后对象仍然过大不能有效存储的话，一个更简单的减少内存占用的方法是将他们保存为序列化的形式，可以使用序列化的存储级别，例如 `MEMORY_ONLY_SER`，详细参见 RDD 持久化 API。

2.4 垃圾回收调优

如果你的程序频繁“折腾”RDD们的话，JVM的垃圾回收会成为一个问题（通常如果程序中对RDD只读取一次再做多次操作的话这并不是个问题）。当Java需要为新对象来清除旧对象的时候，会跟踪所有的Java对象来定位不需要的那些。这里主要的问题是 *垃圾回收的代价与Java对象的数量是成正比的*，所以使用含有较少对象的数据结构（例如用Int的数组代替LinkedList）会很大程度减小这个开销。一个更好的方法是将对象用序列化的形式持久化，像前面提到的那样，这样在RDD的每个分区中就只有 一个 对象（一个字节数组）。如果垃圾回收是一个问题的话可以先考虑使用 [缓存序列化](#)中提到的方法。

垃圾回收的问题还可能存在于任务工作用的内存（执行任务所需要的空间）与节点上缓存的 RDD 之间的冲突。我们会讨论如何控制 RDD 缓存大小来缓解这个问题。

2.4.1 估计垃圾回收的影响

对于垃圾回收的调优，第一步要对垃圾回收发生的频率以及消耗的时间进行统计。可以通过增加 `-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps` 到 `SPARK_JAVA_OPTS`环境变量来实现。在后面的Spark任务运行时，就可以在worker的日志中看到每次垃圾回收发生的相关信息。注意这些日志会在集群中的工作节点上看到（在目录中的 `stdout`文件中），而 不是 你的driver程序。

2.4.2 缓存大小调优

对于垃圾回收一个重要的参数是用于缓存RDD的内存大小。Spark默认使用executor 60%的内存(`spark.executor.memory`)来缓存RDD。也就是说 40%的内存可以用于任务执行中的对象创建。

当你任务变慢的时候，如果发现 JVM 频繁进行垃圾回收或者把内存被用尽，那么可以调低这个值以减少内存消耗。如果要调整为 50%，你可以在 SparkConf 中设置 `conf.set("spark.storage.memoryFraction", "0.5")`。通过与缓存的序列化结合，使用较小的缓存可以有效缓解大部分垃圾回收相关的问题。如果你对 Java 垃圾回收的调优还有兴趣的话，可以继续看看下面的内容。

2.4.3 垃圾回收调优进阶

为了更深入的优化垃圾回收，我们首先要对 JVM 的内存管理有一些概念：

Java 堆空间被分为新生代和老年代两个区域。新生代表示保存存活期比较短的对象，而老年代用于存活期比较长的对象。

新生代又进一步分为三个区域 [Eden, Survivor1, Survivor2]。

简单描述一下垃圾回收的过程：当 Eden 区满时，会在 Eden 区发生一次 minor GC，同时 Eden 区域 Survivor1 区仍然存活的对象会被复制到 Survivor2 区。Survivor 区域被交换。如果对象生命力足够长或者 Survivor2 区域满了，就会被转移到老年代。如果老年代区域满了，会引发一次 full GC。

那 Spark 在 GC 调优的目标就是确保只有长生命力的 RDD 保存在老年代，同时新生代有足够空间来存储短生命力的对象。这样可以避免任务执行过程中回收临时对象引起的 full GC。下面这些步骤会有所帮助：

通过 GC 的统计确定是否有过多的垃圾回收。如果任务完成之前有多次 full GC，说明执行任务的内存不足。

- 如果 GC 的统计表示老年代接近满的话，减少缓存的内存占用。可以通过设置 `spark.storage.memoryFraction`。少缓存一些对象总比减慢任务的执行要好！
- 如果 minor GC 发生的次数过多但是 major GC 并不多的话，那最好给 Eden 区分配更多内存。你可以估计每一个任务需要多少内存来设置 Eden 区的大小。如果 Eden 区大小为 E，那么可以通过选项 `-Xmn=4/3*E` 来设置新生代的大小。（扩到 4/3 倍是考虑到 Survivor 区也要占用空间。）
- 举个例子，如果你的任务需要从 HDFS 上读取数据，那么可以通过从 HDFS 上读取数据块的大小来估计任务需要使用的内存。需要注意解压后的块大小可能是原始块大小的 2、3 倍。如果你希望有 3、4 个任务进行工作，并且 HDFS 数据块大小为 64MB，那么我们可以估计 Eden 区的大小为 $4 \times 3 \times 64\text{MB}$ 。
- 使用新设置后注意观察垃圾回收的频率和次数的变化。

我们的实验表明 GC 调优的效果约定于应用本身和可用的内存大小。可以参见 [更多的调优选项](#) 所描述的。从更高层面来说，管理 full GC 的发生频率可以减少额外开销。

3. 其他需要考虑的地方

3.1 并行度

将每一个操作并行度设置的足够高才能充分利用集群的资源。Spark 根据每个文件的大小自动设置 “map” 任务的个数。（你也可以通过参数 `SparkContext.textFile` 等方法来控制），对于分布式的 “reduce” 操作，例如 `groupByKey` 和 `reduceByKey` 会使用那些父 RDD 的最大分区数。你可以将并行度通过第二个参数传入（参见

thespark.PairRDDFunctions 的文档)，或者通过设置 `spark.default.parallelism` 改变默认值。通常情况下，我们推荐你的集群中每个 CPU 核执行 2-3 个任务。

3.2 Reduce 任务的内存使用

有时候出现 OutOfMemory 错误的时候并不是因为 RDD 的内存不足，而是任务中的一个引起内存不足。比如 `groupByKey` 中的一个 reduce 任务过大。Spark 的 shuffle 操作（`sortByKey`，`groupByKey`，`reduceByKey`，`join`，等）会在每一个聚合任务中建立一个 hash 表，通常会很大。最简单解决这个问题的方法是 **增加并行度**，这样可以使每一个任务的输入的集合变小。Spark 在 200ms 高效的支持任务，因为所有的任务都可以重用 worker JVM，启动一个任务的开销很小。所以你可以安全的提高并行度到比集群 cpu 核心数更多的数量。

3.3 广播 (Broadcasting) 较大的变量

使用 SparkContext 中的 [广播](#) 可以极大减小每个序列化任务的大小以及在集群中启动一个 job 的代价。如果你的任务中使用了来自 driver program 的大型对象，（比如一个静态的查找表），那就可以考虑将这个变量进行广播。Spark 打印了 master 上每一个任务的序列化大小，你可以通过这个来确定你的任务是不是过大了。通常任务比 20KB 大的话就值得进行优化了。

4. 总结

这是一小篇关于 Spark 应用调优尤其在于数据序列化与内存调优的指引，涉及到一些你需要知道的重要概念。对于大多数应用，将序列化器改为 Kryo 并且用序列形式方法持久化数据已经可以解决大部分性能问题。可以通过 [Spark mailing list](#) 对其他调优实践来提问。

■ 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、商业案例探讨！

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