亚太研究院

Spark 官方文档翻译

Spark 配置（V1.2.0）

翻译者　徐骄

Spark 官方文档翻译团成员

前 言

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 亚太研究院去年 8 月发起并号召，结合网络社区的力量构建了 Spark 中文文档专家翻译团队，翻译了 Spark 中文文档 V1.1.0 版本。2014 年 12 月，Spark 官方团队发布了 Spark 1.2.0 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又对 Spark 1.2.0 版本进行了部分更新，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 1.2.0 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(v1.2.0)》
- 王宇舟,《Spark 机器学习库 (v1.2.0)》
- 武扬，《在 Yarn 上运行 Spark (v1.2.0)》《Spark 调优(v1.2.0)》
- 徐骄，《Spark 配置(v1.2.0)》《Spark 作业调度(v1.2.0)》
- 蔡立宇，《Bagel 编程指南(v1.2.0)》
- harli，《Spark 编程指南 (v1.2.0)》
- 韩保礼，《Spark SQL 编程指南(v1.2.0)》
- 李丹丹，《文档首页(v1.2.0)》
- 李军，《Spark 实时流处理编程指南(v1.2.0)》
- 俞杭军，《使用 Maven 编译 Spark(v1.2.0)》
- 王之，《给 Spark 提交代码(v1.2.0)》
- Ernest ,《集群模式概览(v1.2.0)》《监控与相关工具(v1.2.0)》《提交应用程序(v1.2.0)》

Life is short, You need Spark!

Spark 亚太研究院院长 王家林
2015 年 2 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 配置（V1.2.0）

（翻译者：徐骄）

Spark Configuration，原文档链接：
http://spark.apache.org/docs/latest/configuration.html

目录

Spark 配置

Spark 提供了对系统进行设置的 3 类配置：

1、 Spark 属性：控制应用程序参数，可使用 SparkConf 对象或者 Java 体系中的属性进行设置；
2、 环境变量 通过 conf/spark-env.sh 文件用于对单独的每台机器进行设置 比如 IP 地址；
3、 日志，通过 log4j.properties 文件对 Spark 日志进行配置。

Spark 属性

Spark 的属性控制着应用程序的设置，且是由每个应用程序单独进行配置的。这些属性可以直接通过传递 SparkConf 对象到 SparkContext 进行设置。SparkConf 允许配置一些通用属性（例如 Master URL 以及应用程序名称），也可以使用 set()方法配置任意的 key-value 对。例如，可以如下所示使用 2 个线程来初始化一个应用程序：

注意，示例中使用 local[2]模式运行，意味着有 2 个线程（表示最小的并行度），这样可以检测出仅存在于分布式上下文中的错误。在 local 模式中可以使用>1 的线程数，例如在 spark streaming 中，可以专门使用 1 个线程以防止任何形式的饥饿问题。

```
val conf = new SparkConf()
             .setMaster("local[2]")
             .setAppName("CountingSheep")
             .set("spark.executor.memory", "1g")
val sc = new SparkContext(conf)
```

动态加载属性

在一些例子中，你可能想避免在 SparkConf 中捆绑某些配置，比如，你希望在不同的 masters 中或以不同的内存量运行同一个应用程序，此时 Spark 允许你像下面这样简单的创建一个空的 SparkConf：

```
val sc = new SparkContext(new SparkConf())
```

然后，可以在运行时给出具体的属性值：

```
./bin/spark-submit --name "My app" --master local[4] --conf spark.shuffle.spi
ll=false --conf "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+Pri
ntGCTimeStamps" myApp.jar
```

Spark shell 和 spark-submit 工具为动态加载配置提供了两种方式，第一种是命令行选项，比如上述命令中的—master。Spark-submit 可以通过--conf 标签接受任意 Spark 属性，但是这些属性标签在启动 Spark 应用程序时用得并不多。运行 ./bin/spark-submit --help 会显示出全部选项列表。

bin/spark-submit 会从 conf/spark-defaults.conf 文件中读取配置选项，该文件中每行由空格分割的一个键值对组成。例如：

```
spark.master            spark://5.6.7.8:7077

spark.executor.memory   512m

spark.eventLog.enabled  true

spark.serializer        org.apache.spark.serializer.KryoSerializer
```

由选项标记或属性文件中设定的值都将被传递给应用程序，并与 SparkConf 中的设定进行合并。SparkConf 中的属性设置拥有更高的优先级，选项标记传递给 spark-submit 或 spark-shell 的设置优先级次之，spark-defaults.conf 文件中的设置优先级为最末。

查看 Spark 属性

应用程序的 Web UI 在 http://<driver>:4040 页面中的“Environment”标签下列出了 Spark 的一些属性。可以通过“Environment”标签的列表方便地检查 Spark 的各个属性是否设置正确，注意，只有通过 spark-defaults.conf 文件或 SparkConf 对象明确设定的值才会被列出。而其他的属性，则使用默认值。

已提供的属性

控制内部设置的大多数属性都有其默认值。下面给出常用选项的设置：

应用程序属性

属性名称	默认值	属性意义
spark.app.name	(none)	应用程序名称。此名称显示在 UI 以及日志数据中。
spark.master	(none)	连接到的集群管理器。
spark.executor.memory	512m	每个 executor 进程可使用的内存大小，设置格式与 JVM 内存设置一样

		(e.g. 512m, 2g)
spark.driver.memory	512m	每个 driver 进程可使用的内存大小，即 SparkContext 初始化的地方 (e.g. 512m, 2g)
spark.driver.maxResultSize	1g	一个 action 操作（如 collect）所对应所有分区的序列化结果大小的上限值。最小 1M，或不限制（设置为 0）。如果结果大小超出限制，作业将中断。此属性值设置过高可能会导致 driver 内存溢出（依据 spark.driver.memory 以及 JVM 上限），设置一个合适的值可以避免 driver 发生内存溢出错误。
spark.serializer	org.apache.spark.serializer.JavaSerializer	序列化对象使用的类。序列化对象将通过网络发送或以某种序列化方式被缓存。Java 默认序列化使用任何可序列化的 Java 对象，但是很慢，所以若对速度有要求，我们建议使用 org.apache.spark.serializer.KryoSerializer 并配置 Kryo 序列化，可以是 org.apache.spark.Serializer 的任意子类。
spark.kryo.classesToRegister	(none)	如果使用了 Kryo 序列化，此值为以逗号分隔的自定义类名。
spark.kryo.registrator	(none)	如果使用了 Kryo 序列化，对 Kryo 进行自定义的注册。它应该被设置成一个继承自 KryoRegistrator 的类。
spark.local.dir	/tmp	Spark 所使用的目录空间，包括 map 的输出文件、要存储在磁盘中的 RDDs。这是一个快速的本地磁盘，可以设置为以逗号分隔的不同磁盘中的多个目录。注意：在 Spark1.0 及其后续版本中，此属性将被集群管理器设置的环境变量 SPARK_LOCAL_DIRS（Standalone，Mesos）或 LOCAL_DIRS（YARN）覆盖。
spark.logConf	false	在 SparkContext 启动的时候记录有效的 SparkConf 信息。

除了上述属性之外，还提供了以下这些在某些场景很实用的属性：

运行时环境

属性名称	默认值	属性意义
spark.executor.extraJavaOptions	(none)	传递给 executor 的额外选项。

		例如，GC 设置或其他日志记录。注意，用这个选项设置 Spark 属性或者堆大小都是非法的。Spark 属性应该由 SparkConf 对象或 spark-submit 脚本使用的 spark-defaults.conf 文件设置。堆大小的设置可以由 spark.executor.memory 设置。
spark.executor.extraClassPath	(none)	附加给 executor 类路径的额外类路径入口，这主要是为了向后兼容旧版本的 Spark ,用户不应该设置这个选项。
spark.executor.extraLibraryPath	(none)	专门设置的库路径，供启动 executor JVM 时使用。
spark.executor.logs.rolling.strategy	(none)	设置 Executor 日志回滚策略，缺省为不可用，可以设置为“time“（基于时间的回滚），或“size“（基于大小的回滚）。前者使用 spark.executor.logs.rolling.time.interval 设置回滚间隔，后者使用 spark.executor.logs.rolling.size.maxBytes 设置回滚的日志文件最大值。
spark.executor.logs.rolling.time.interval	daily	Executor 日志回滚的时间间隔，缺省情况下日志回滚不可用。此属性的有效值为’daily’,’hourly’,’minutely’或任意秒级间隔。
spark.executor.logs.rolling.size.maxBytes	(none)	Executor 日志回滚的日志文件阈值，缺省情况下日志回滚不可用。此属性值的大小单位为字节。
spark.executor.logs.rolling.maxRetainedFiles	(none)	设置系统预保留的最新的日志文件个数。此前的日志文件将被删除。此属性缺省下为不可用。
spark.files.userClassPathFirst	false	（试验中）此选项表示在 Executor 中加载类时，是否给予用户添加的 jar 包更高的优先

		级（比 Spark 自身的 jar 包）。这个设置可以减少 Spark 依赖和用户依赖之间的冲突。目前这个属性还在试验阶段。
spark.python.worker.memory	512m	在聚合阶段每个 python 工作进程使用的内存量，设置格式与 JVM 内存设置一样 (e.g. 512m, 2g)。如果在聚合阶段所使用的内存超出了这个属性值，数据将溢出写到磁盘中。
spark.python.profile	false	在 python 工作进程中开启简报功能，简报可以使用 sc.show_profile()查看，或者在 driver 退出前显示。也可以使用 sc.dump_profiles(path)将其导出到磁盘。
spark.python.profile.dump	(none)	导出简报结果的磁盘目录，将为每个 RDD 单独导出一个简报文件，它们可以一起由 ptats.Stats()加载，如果指定了这个属性，那么简报结果就不会自动显示了。
spark.python.worker.reuse	true	是否允许重用 python 工作进程。如果允许重用，会使用固定的 python 工作进程数 ,通过重用不需要为每一个任务调用一个 python 进程。
spark.executorEnv.[EnvironmentVariableName]	(none)	通过 EnvironmentVariableName 向 Executor 进程添加环境变量，用户可以指定多个此属性以设置多个环境变量。
spark.mesos.executor.home	Driver 端的 SPARK_HOME	设置 Executor 基于 Mesos 环境中 Spark 的安装位置。缺省下 Executor 会使用 driver 的 Spark home 目录，而此目录对 Executor 是不可见的。注意，此属性仅当 spark 二进制包没有用 Spark.executor.uri 进行设置时其作用。
spark.mesos.executor.memoryOverhead	Executor 内存 * 0.07 ,最	这个属性值是 spark.executor.memory 的附属

	小值 384	值，用 MiB（用于计算总的 Mesos 任务内存）指定。384 指 384MiB 的上限，此外，还可以为 7%的硬编码最小上限。此属性的最终值为这两者中较大的。

Shuffle 行为

属性名称	默认值	属性意义
spark.shuffle.consolidateFiles	false	如果为 true ,会在 shuffle 中生成中间文件 ,生成越少的文件可以提升 shuffle 和大量 reduce 任务时文件系统的性能。建议在使用 ext4 或 xfs 文件系统时设置为 true，而对于 ext3 文件系统，若此选项设置为 true 会由于文件系统限制而降低多核（>8）机器的性能。
spark.shuffle.spill	true	如果为 true，通过使数据溢出到磁盘对 reduce 阶段内存的使用进行限制。溢出的阀值由 spark.shulle.meoryFraction 属性设定。
spark.shuffle.spill.compress	true	Shuffle 时是否压缩溢出数据。具体压缩方式由 spark.io.compression.codec 属性设定。
spark.shuffle.memoryFraction	0.2	如果 spark.shuffle.spill 设为 true ,此选项才有意义 ,表示 shuffle 阶段聚合及合组操作所使用的 Java 堆内存占比。在任意时刻，shuffle 所使用的内存中所有 map 的公共大小受限于这个设置，超过这个阀值就溢出到磁盘。如果频繁的溢出，应该考虑以 spark.storage.memoryFraction 选项的设置为代价增大这个值。（后面会介绍 spark.storage.memoryFraction 这个选项）
spark.shuffle.compress	true	是否压缩 map 的输出文件 ,通常选择压缩。具体压缩方式由 spark.io.compression.codec 属性设定。
spark.shuffle.file.buffer.kb	32	每个 shuffle 文件输出流的内存缓冲区大小，单位是 KB。这些缓冲区减少了创建 shuffle 中间文件时的磁盘寻道以及

		系统调用。
spark.reducer.maxMbInFlight	48	此选项设定同时从 reduce 任务中取出的的 Map 输出最大值（单位 MB）。因为要为每一份输出创建一个缓冲区进行接收，这表示每一个 reduce 任务要消耗固定大小的内存，所以，尽量使这个选项的值较小，除非你有大量的内存可用。
spark.shuffle.manager	sort	对数据进行 shuffle 时执行的 shuffle 管理器。目前提供了 2 种 shuffle 执行管理器：sort 和 hash。基于排序的 shuffle 具有更佳的内存效率，spark1.1 中的默认值为 hash，而从 spark1.2 开始，此属性的默认值为 sort。
spark.shuffle.sort.bypassMergeThreshold	200	（高级的）此选项属性用于设置在以下情形下的最多 reduce 分区数 :基于排序的 shuffle 管理器时，若没有 map 端聚合，同时避免数据的归并排序。
spark.shuffle.blockTransferService	netty	在 executor 中传递 shuffle 和缓存块时使用的执行框架。目前提供了 2 种执行框架：netty 和 nio。从 spark 1.2 开始，此属性的默认值为 netty

Spark UI

属性名称	默认值	属性意义
spark.ui.port	4040	应用程序 dashboard 端口，显示内存使用及作业完成情况。
spark.ui.retainedStages	1000	在 GC 前 Spark UI 能保留的 Stage 数目。
spark.ui.retainedJobs	1000	在 GC 前 Spark UI 能保留的 Job 数目。
spark.ui.killEnabled	true	是否允许 Stage 和相应的作业从 Web UI 关闭。
spark.eventLog.enabled	false	是否记录 Spark 事件，这个选项对于应用程序已完成后重构其 Web UI 非常有用。
spark.eventLog.compress	false	当 spark.eventLog.enabled 设置为 true 时，是否压缩日志事件。
spark.eventLog.dir	file:///tmp/spark-events	如果 spark.eventLog.enabled 设置为 true ,此选项表示 Spark 事件日志的基目录。Spark 将在此基目录的基础上为每个应用程序创建

		一个子目录，并在子目录中记录相关的事件。用户可以将其设置为如同 HDFS 目录的统一位置以便服务器读取历史日志文件。

压缩及序列化

属性名称	默认值	属性意义
spark.broadcast.compress	true	在发送广播变量前，是否进行压缩。一般来说，这是个好主意。
spark.rdd.compress	false	是否压缩已序列化的 RDD 分区（如 StorageLevel.MEMORY_ONLY_SER），在消耗部分额外 CPU 的同时节省大量的空间。
spark.io.compression.codec	snappy	压缩中间数据（例如 RDD 分区和 shuffle 输出）使用的编码。默认情况下 Spark 提供 3 种编码：lz4，lzf 以及 snappy。你也可以使用全称类名指定编码，如：org.apache.spark.io.LZ4CompressionCodec,org.apache.spark.io.LZFCompressionCodec,以及 org.apache.spark.io.SnappyCompressionCodec。
spark.io.compression.snappy.block.size	32768	Snappy 压缩使用的块大小（单位 B），使用 Snappy 压缩时降低这个值也会降低 shuffle 使用的内存。
spark.io.compression.lz4.block.size	32768	LZ4 压缩使用的块大小（单位 B），使用 LZ4 压缩时降低这个值也会降低 shuffle 使用的内存。
spark.closure.serializer	org.apache.spark.serializer.JavaSerializer	闭包使用的序列化类。目前只支持 Java Serializer。
spark.serializer.objectStreamReset	100	当使用 org.apache.spark.serializer.JavaSerializer 进行序列化时，序

		列化器将缓存对象以防止写入冗余数据，然而此举也将停止这些对象的 GC。通过调用‘reset’，可以从序列化器中清洗这些对象信息，且允许回收这些旧对象。若要关闭这个周期性重置，将其设置为-1 即可。默认情况下对每 100 个对象重置序列化器。
spark.kryo.referenceTracking	true	使用 Kryo 序列化数据时，是否跟踪对同一对象的引用。如果你的对象图中存在环路，或者包含同一对象的多份拷贝，那么这个设置是非常必要的且高效的。如果你确定你的场景与此无关，可以将此选项设置为 false 以提升性能。
spark.kryo.registrationRequired	false	是否要求 Kryo 注册，如果设置为 true，在序列化一个未注册的类时，Kryo 将抛出异常。如果设置为 false（默认设置），Kryo 将未注册的类名同每个对象一起写入。写类名会引起显著的性能开销，所以开启这个选项可以强制使用户不能忽略类的注册。
spark.kryoserializer.buffer.mb	0.064	Kryo 序列化缓冲区的初始大小（单位 MB）。注意，每个 Worker 的每个核都有一个缓冲区，必要时会增大到 spark.kryoserializer.buffer.max.mb 选项设置的值。
spark.kryoserializer.buffer.max.mb	64	Kryo 序列化所允许的最大缓冲区大小，单位为 MB。此值必须大于即将序列化的任何一个对象。如果在 Kryo 中出现“缓冲区超过限制”，则增大这个值。

执行行为

属性名称	默认值	属性意义
spark.default.parallelism	本地模式：本地机器的核数 Mesos 细粒度模式：8	在用户未设置的情况下，默认跨集群的分布式 shuffle 操作（groupByKey,reduceByKey

	其他：所有 Executor 节点的核数总和或 2	等）使用的任务数。
spark.broadcast.factory	org.apache.spark.broadcast.TorrentBroadcastFactory	广播的实现
spark.broadcast.blockSize	4096	实现广播时，块的每一个分片大小（单位 KB）。值过大将降低并行性（使其变慢），而值过小则会使 BlockManager 产生性能影响。
spark.files.overwrite	false	当目标文件已存在且内容不相同时，是否通过 SparkContext.addFile()覆盖已添加的文件。
spark.files.fetchTimeout	60	当从 Dirver 处通过 SparkContext.addFile()获取已添加文件时是否启用通信超时。
spark.storage.memoryFraction	0.6	Spark 内存缓存使用的 Java 堆占比。这个值不能大于 JVM 中的旧对象的大小，但如果你配置了旧版本的大小值，可以增大这个选项的值。
spark.storage.unrollFraction	0.2	spark.storage.memoryFraction 的占比，用于表示内存中的块。这是在没有足够的存储空间整体的表示新块的情况下通过删除已存在的块而形成的动态分配。
spark.tachyonStore.baseDir	System.getProperty("java.io.tmpdir")	用于存储 RDD 的 Tachyon 文件系统的目录。Tachyon 文件系统的 URL 由 spark.tachyonStore.url 设置。也可以是一个由逗号分隔的目录列表。
spark.storage.memoryMapThreshold	8192	从磁盘读入一个块时 Spark 内存映射的以字节为单位的块大小。这防止了 Spark 在内存中映射出很多小块。一般而言，内存映射对块的关闭有高额开销或使得低于操作系统中页的大小。
spark.tachyonStore.url	tachyon://localhost:19998	Tachyon 文件系统的 URL
spark.cleaner.ttl	(infinite)	Spark 记录元数据（Stage 产生的，任务产生的等）的时间间隔

		（以秒为单位）。周期性的清理可以保证过时的元数据被抛弃。这对于长时间（小时、天）运行 Spark 很实用。注意，超过这个时间间隔的任意物化于内存的 RDD 都将被清理。
spark.hadoop.validateOutputSpecs	true	若设置为 true，则对 saveAsHadoopFile 或其同类操作输出进行验证（如检查输出目录是否已存在）。也可以由于输出目录已存在关闭此选项。建议开启此选项，除非你要兼容于以前的 Spark 版本。可以简单的使用 Hadoop 文件系统的 API 手动的删除输出目录。
spark.hadoop.cloneConf	false	若设置为 true，为每个任务克隆一个新的 Hadoop Configuration 对象。当涉及到 Configuration 线程安全问题时，此属性值需设置为 true。
spark.executor.heartbeatInterval	10000	Executor 向 Driver 返回的心跳信息之间的时间间隔（毫秒）。心跳信息使得 Driver 知道 Executor 是否还存活，并对未完成的任务进行更新。

网络

属性名称	默认值	属性意义
spark.driver.host	(local hostname)	Driver 监听的主机名或 IP 地址，用于与 Executors 以及独立模式中 Master 直接的通信。
spark.driver.port	(random)	Driver 监听的端口号，用于与 Executors 以及独立模式中 Master 直接的通信。
spark.fileserver.port	(random)	Driver 的 HTTP 文件服务器监听端口。
spark.broadcast.port	(random)	Driver 的 HTTP 广播服务器监听端口，与洪流广播无关。
spark.replClassServer.port	(random)	Driver 的 HTTP 类服务器监听端口。只有它与 Spark Shell 相关。
spark.blockManager.port	(random)	所有的块管理器的监听端口。在

		Driver 和 Executor 两端都存在。
spark.executor.port	(random)	Executor 的监听端口，用于与 Driver 通信。
spark.port.maxRetries	16	在放弃绑定到某个端口之前可以尝试的最大次数。
spark.akka.frameSize	10	在控制层通信（序列化任务及结果）所允许的最大消息量，以 MB 为单位。如果你的任务需要返回大量结果给 Driver，可以增大这个值。
spark.akka.threads	4	用于通信的 Actor 线程的数量。在 Driver 有多核时对大规模集群增加此值是非常有用的。
spark.akka.timeout	100	Spark 节点之间的通信超时时间，以秒为单位。
spark.akka.heartbeat.pauses	6000	将其设置为较大的值以禁用内建 akka 中的错误检测器，如果你打算使用这个特性（但是不建议），可以被再次启动。akka 可接受短时间内的心跳信息暂停，这可以用来控制 GC 暂停的敏感性。如果有需要，可以结合 `spark.akka.heartbeat.interval`和 `spark.akka.failure-detector.threshold`进行调优。
spark.akka.failure-detector.threshold	300.0	将其设置为较大的值以禁用内建 akka 中的错误检测器，如果你打算使用这个特性（但是不建议），可以被再次启动。此属性值对应于 akka 的 `akka.remote.transport-failure-detector.threshold`。如果有需要，可以结合 `spark.akka.heartbeat.interval` 和 `spark.akka.heartbeat.pauses` 进行调优。
spark.akka.heartbeat.interval	1000	将其设置为较大的值以禁用内建 akka 中的错误检测器，如果你打算使用这个特性（但是不建议），可以被再次启动。大的时间间隔值（单位秒）将降低网络负载而

		较小的值(~1s)对于 akka 的错误检测器会更有益。如果有需要，可以结合 `spark.akka.failure-detector.threshold` 和 `spark.akka.heartbeat.pauses` 进行调优。敏感性的错误检测器可以真正的快速帮助驱逐流氓 Executor ,然而这通常并非真是 Spark 集群环境下所期望的 GC 暂停和网络延迟。除了启用这个导致大量节点之间交换心脏跳动导致大量的网络延迟。除此之外，开启此特性将导致各节点间大量的心跳信息交换，从而导致网络拥塞。

调度器

属性名称	默认值	属性意义
spark.task.cpus	1	每个任务所分配的核数。
spark.task.maxFailures	4	任务在被撤销前的可失败次数。应该设置为大于 1，允许重复尝试的次数等于这个属性值-1。
spark.scheduler.mode	FIFO	作业的调度模式。可以设置为 FAIR 模式。
spark.cores.max	(not set)	当运行 Standalone 或 Mesos 集群模式时，此属性值表示应用程序请求的跨集群的总核数，而非单台机器核数。如果未设置，则默认将会使用 Standalone 的 spark.deploy.defaultCores 属性值或 Mesos 的所有核数。
spark.mesos.coarse	false	如果设置为 true ,Mesos 集群以粗粒度共享模式运行，这种模式要求每一台机器中中有一个长期存活的 Mesos 任务。
spark.speculation	false	若设置为 true，则执行任务的推测执行。也就是说，当一个阶段中的一个或多个任务运行缓慢的时候，它们将被重新执行。
spark.speculation.interval	100	推测执行的检查时间间隔，以毫秒为单位。

spark.speculation.quantile	0.75	某个特定 Stage 中，开启推测执行前任务必需已完成的百分比。
spark.speculation.multiplier	1.5	任务慢于中值的倍数，通过此属性值判断是否推测执行。
spark.locality.wait	3000	在放弃并启动一个非本地节点前等待启动一个数据本地化任务的毫秒时间。同样的等待时间也将用于多个本地化级别（处理、节点、机架等）。也可以设置 spark.locality.wait.node 自定义每个本地级别的等待时间。如果任务较长且本地化较低可以增大此值。但是默认值在一般情况下可以工作得很好。
spark.locality.wait.process	spark.locality.wait	自定义处理本地化的本地等待，这会使任务在特定的 Executor 处理时访问缓存数据。
spark.locality.wait.node	spark.locality.wait	自定义节点本地化的本地等待。例如，可以设置为 0 跳过节点本地性并立即开始机架本地性（若集群存在机架信息）。
spark.locality.wait.rack	spark.locality.wait	自定义机架本地化的本地等待。
spark.scheduler.revive.interval	1000	调度器恢复一个 Worker 资源以供任务运行的时间间隔，单位：毫秒。
spark.scheduler.minRegisteredResourcesRatio	Mesos & Standalone:0.0; YARN:0.8	开始调度之前，等待的注册资源的最小比例（已注册资源/总资源）（资源是 Yarn 模式中的 Executor 数，Standalone 模式中的 CPU 核数），设置为 0 到 1 之间的一个 Double 数。无论是否达到最小比例，开始调度前等待资源的最长时间由 spark.scheduler.maxRegisteredResourcesWaitingTime 属性值控制。
spark.scheduler.maxRegisteredResourcesWaitingTime	30000	开始调度前等待资源注册的最长时间，单位：毫秒。
spark.localExecution.enabled	false	设定 Spark 是否可以在 Driver 中运行某些作业，如 first()或 take()，而不将作业发往集群。这可以使某些作业快速执行，但

		可能要求将数据的整个分区发送给 Driver。

安全

属性名称	默认值	属性意义
spark.authenticate	false	Spark 是否认证其内部连接。若不是在 YARN 模式运行，此设置请查看 spark.authenticate.secret 属性。
spark.authenticate.secret	None	设置 Spark 组件间的连接认证的密钥，若不在 YARN 模式运行，且已启动 spark.authenticate 属性，则需设置此属性。
spark.core.connection.auth.wait.timeout	30	在超时及放弃认证前的认证连接等待时间。
spark.core.connection.ack.wait.timeout	60	在超时及放弃认证前的应答等待时间。为避免由类似 GC 暂停这样的长时间暂停所导致的超时，可以增大此值。
spark.ui.filters	None	应用于 Spark Web UI 的滤器类名，由逗号分隔。过滤器应该是标准的 javax servlet Filter，每个过滤器的参数可以由 Java 系统属性 spark.<class name of filter>.params='param1=value1,param2=value2'设定。 示例： -Dspark.ui.filters=com.test.filter1 -Dspark.com.test.filter1.params='param1=foo,param2=testing'
spark.acls.enable	false	是否使用 Spark ACLS，如果使用，将检查用户是否具有查看或修改作业的权限。注意，此属性要求用于是已知用户，如果用户为 null，则不会进行权限检查。可以使用 UI 的过滤器验证和设置用户。
spark.ui.view.acls	Empty	对 Spark Web UI 有查看权限的

		用户列表，以逗号分隔。默认只允许启动该 Spark 作业的用户查看。
spark.modify.acls	Empty	对 Spark 作业有修改权限的用户列表，以逗号分隔。默认只允许启动该 Spark 作业的用户修改。例如 kill 该作业。
spark.admin.acls	Empty	对所有 Spark 作业有查看和修改权限的用户/管理员列表 ,以逗号分隔。当你在共享集群上运行作业且有一群管理员或开发人员时 ,可以使用这个属性进行设置。

Spark Streaming

属性名称	默认值	属性意义
spark.streaming.blockInterval	200	时间间隔，此间隔内 Spark Streaming 接收器接收的数据在存入 Spark 前将合并成一个块。
spark.streaming.receiver.maxRate	infinite	每一个接收器将数据推送到块的最大速率（每秒）。将这个属性设置为 0 或负数表示此属性没有限制。
spark.streaming.receiver.writeAheadLogs.enable	false	允许为 receiver 写‘预日志’。所有通过 receiver 接收到的数据将通过预日志进行保存，预日志可以在 driver 掉线后进行数据恢复。
spark.streaming.unpersist	true	Spark Streaming 强制使 RDDs 不持久化于 Spark 内存中，Spark Streaming 接收的原始输入数据也将自动被清空。将其设置为 false ,则原始输入数据及持久化 RDD 因为未被清空而可以在 Streaming 应用程序之外被访问。但这也会造成大量的内存消耗。

集群管理器

Spark 中的每个集群管理器都有附加的配置选项，不同模式的配置分别如下：

YARN

Mesos

Standalone

环境变量

有部分 Spark 属性可以通过环境变量设置，这些环境变量从 Spark 安装目录的 conf/spark-env.sh（或 Windows 中的 conf/spark-env.cmd）中读取。在 Standalone 和 Mesos 模式下，这个文件可以给出机器的固定信息，如主机名。

注意，conf/spark-env.sh 文件在 Spark 安装时并不存在，需要复制 conf/spark-env.sh.tmplate 来创建 spark-env.sh 这个文件。请确保你对该文件进行了复制。

以下变量可以在 spark-env.sh 中设置：

环境变量	意义
JAVA_HOME	Java 的安装位置
PYSPARK_PYTHON	用于 PySpark 执行的 Python 二进制可执行目录
SPARK_LOCAL_IP	机器绑定的 IP 地址
SPARK_PUBLIC_DNS	Spark 程序广播给其它机器的主机名称

除了上述的环境变量之外，还有为 Standalone 集群提供的选项，如每个机器所使用的核数、最大使用内存等。

由于 spark-env.sh 是一个 shell 脚本，所以可编程式的对环境变量进行设置。例如，通过查询某个特定的网络接口 IP 得到 SPARK_LOCAL_IP 的值。

配置日志记录

Spark 使用 log4j 进行日志记录。通过在 conf 目录中添加 log4j.properties 文件进行日志的各种配置。一种方式是通过复制 conf/下已存在的 log4j.properties.template 进行创建。

重载配置目录

若要指定一个不同的配置目录（Spark 默认配置目录为”SPARK_HOME/conf”），可以设置 SPARK_CONF_DIR，spark 将使用此目录下的配置文件（spark-defaults.conf，spark-env.sh，log4j.properties 等）。

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

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

图书：

《大数据 spark 企业级实战》

京东购买官网：http://item.jd.com/11622851.html

当当购买官网：http://product.dangdang.com/23631792.html

亚马逊购买官网：http://www.amazon.cn/dp/B00RMD8KI2/