亚太研究院

Spark 官方文档翻译

Spark 作业调度

(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 作业调度
(v1.2.0)

（翻译者：徐骄）

Job Scheduling，原文档链接：

http://spark.apache.org/docs/latest/job-scheduling.html

目录

1. 概述

Spark 有很多措施能够使得它在计算过程中进行资源调度。首先，就像集群模式概述中讲的那样，每一个 spark 应用（即 sparkContext 的实例）运行一个独立的执行进程。Spark 的集群管理器为跨应用的调度提供处理措施。其次，在每一个 spark 的应用中，不同的线程可能会导致多个作业（spark actions）的同时运行。这一点非常常见如果你的应用跨网络提交请求；比如，shark 的服务端就是这样工作。Spark 在每个 sparkContext 包含了一个公正的调度策略来进行资源调度。

2. 应用间的调度

当 Spark 以集群模式运行的时候，每一个 spark 的应用都会获得一个独立的 jvm 使其可以执行作业跟保存相关数据。如果有多个用户需要共同使用你的集群，Spark 会根据集群的管理器具有多种策略来进行资源分配。

对所有的集群管理器来说最简单的策略就是静态的资源分配。在这种方法中，每一个应用将会被给予系统能够分配的最大的资源并且在整个运行周期内占有这些资源。以下将会介绍 spark 中的 standalone 模式，yarn 模式以及 coarse-grained memos 模式：

Standalone 模式：默认的，当应用被提交到 standalone 模式下，集群会进入 FIFO 排序模式（先进先出），并且每一个应用都会尝试去使用所有可用的节点。你可以通过使用 spark.cores.max 或者修改默认的参数让应用不再使用 spark.deploy.defaultCores 这个参数来设置从而来限制节点的数量。 Spark 除了能够设置 cpu 的核数之外，还可以通过 spark.executor.memory 这个参数来设置应用所使用能够使用的内存。

YARN 模式：可以通过配置 --num-executors 这个属性使 spark 的 YARN 客户端控制集群下执行器的数量，可以使用-executor-memory 与 --executor-cores 这两个属性来限制每个执行器使用的资源。

Mesos 模式：为了使用 Mesos 的静态分区，可以通过设置 spark.mesos.coarse 这个属性为 true，并且有选择性的设置 spark.cores.max 来限制每个应用在所使用的资源，就像 standalone 模式一样。另外还可以通过设置 spark.executor.memory 这个属性来控制执行器的内存使用。

Mesos 模式下还存在一个额外的可以动态分配 CPU 核数选项。在这种模式下，每一个应用仍然会有一个固定并且独立的内存（通过 spark.executor.memory 设置），但是当这个应用不在运行作业的实时，别的应用可以使用这些 cpu。这种模式尤为有效当有大量非活跃的应用的时候，比如单独用户的 shell 会话。然而，这种模式带有不可预测的延时风险，因为当一个应用又开始运行的时候，前面的设置会导致这个应用需要一段时间才能够重新获得它原有的资源。想用这个模式的话，直接用 mesos:// URL 并不用设置 spark.mesos.coarse 为 true。

需要指出的是上面的模式中没有任何一种可以在众多应用中共享内存。如果你想共享内存，我们建议运行一个单独的服务器的应用使其可以通过查询相同的 RDDs 能够提供多个请求。像 Shark 的 JDBC 服务端就是通过这种方式来提供 SQL 的查询。在将来的版本中，基于内存的存储系统例如 Tachyon 将会提供其他的方式来共享 RDDs

动态资源分配

Spark 1.2 介绍了其根据工作负载进行资源动态分配的特性，换言之，如果你的应用稍后不再需要使用或请求当前需求的资源时，会将这些资源还给集群，这个特性在你的 Spark 集群有多个应用共享资源时极其有用。如果分配给一个应用的一部分资源处于闲置状态，这些闲置资源将被返回到集群资源池中并等待其他应用的请求。Spark 中的资源动态分配是执行器的粒度上执行的 通过设置 spark.dynamicAllocation.enable 属性启动动态资源分配。

目前，这个特性仅在 YARN 模式可用，且默认情况下为未启用状态，在后续的 Spark 版本中，将扩展此特性到 Standalone 模式以及 Mesos coarse-grained 模式。值得一提的是，虽然基于 Mesos 的 Spark 在 fine-grained 模式下已经有了一个类似于动态资源共享的概念，但是仍然建议启用动态资源分配，因为这可以使得你的 Mesos 应用在共享集群资源时，能够更有效地利用粗粒度的低延迟调度。

1） 配置

动态资源分配这一特性的所有配置都在 spark.dynamicAllocation.*空间下。要开启动态资源分配属性，需将 spark.dynamicAllocation.enabled 设置为 true，并设置 spark.dynamicAllocation.minExecutors 和 spark.dynamicAllocation.maxExecutors 以提供执行器数量的上下限。

此外，你的应用必须使用一个额外的 shuffle 服务，此服务的目的是保存执行器写出的 shuffle 文件，这样执行器即可安全移除。设置 spark.shuffle.service.enabled 为 true 以以激活此服务，在 YARN 中，这个服务由运行在集群中每个 NodeManager 上的 org.apache.spark.yarn.network.YarnShuffleService 执行。按照以下步骤，启动服务：

1，编译 Spark。（若使用预封装版本跳过此步）

2，定位 spark-<version>-yarn-shuffle.jar。如果是自己编译 Spark，此 jar 包应该在$SPARK_HOME/network/yarn/target/scala-<version>；如果使用预封装版本，此 jar 包应该在 lib 目录下。

3，将 2 中的 jar 添加到集群中所有 NodeManager 的类路径中（classpath）。

4，在每个节点的 yarn-site.xml 文件中，将 spark_shuffle 添加到 yarn.nodemanager.aux-services 属性，并将 yarn.nodemanager.aux-services.spark_shuffle 属性设置为

org.apache.spark.networ.yarn.YarnShuffleService，此外可设置其他 spark.shuffle.service.*的属性。

5，重启集群中所有的 NodeManager。

2） 资源分配策略

在高级应用中，Spark 放弃不再使用的执行器，需要时再请求。由于并没有一种确定的方法预测一个即将被移除的执行器是否在不久之后运行任务，也没法确定一个即将被添加的新执行器是否真的处于闲置状态，因此我们需要一些启发式的方法去决定对执行器进行移除和请求。

A. 执行器请求策略

在启用了动态分配的 Spark 应用中，若存在正等待被调度的被挂起任务时，则请求额外的执行器，这个必然条件意味着已存在的执行器不能同时满足被提交且未完成的所有任务。

Spark 以轮询的方式请求执行器，实际请求在有任务被挂起达到属性 spark.dynamicAllocation.schedulerBacklogTimeout 设置的数秒时被触发，如果挂起任务队列中仍有任务时每隔由属性 spark.dynamicAllocation.sustainedSchedulerBacklogTimeout 设置的秒数再次触发执行器请求。此外，每一轮所请求的执行器数量会在上一轮的基础上以指数递增。举例来说，一个应用在第一轮添加了 1 个执行器，后续轮中即为 2，4，8。

上述请求的执行器以指数递增的原因有 2 点。首先，如果一个应用在开始执行时只需要很少的额外执行器，那么就要慎重地请求执行器，即起始基数较小。其次，如果一个应用确实需要多个执行器，则其应该具备及时提升资源利用率的能力，即指数递增。

B. 执行器移除策略

执行器的移除策略更为简单，Spark 应用会在执行器空闲时间超过属性 spark.dynamicAllocation.executorIdleTimeout 所设置的秒数时移除执行器。要注意的是，在多数情况下，如果仍然有任务被挂起，那执行器就不应该闲置，所以执行器的移除条件和请求条件是互斥的。

3） 执行器优雅的退出

在动态分配之前，任务失败或关联的应用程序退出会导致 Spark 执行器的退出， 在这两种场景中，所有跟执行器相关的状态都不再需要，可以安全的将这些状态废除。然而在动

态分配中，若有执行器强制退出，应用仍然会运行，如果应用程序要访问执行器所保存或写出的状态，就得重新计算状态，这样一来，Spark 就需要一个机制使执行器优雅地退出，即在退出前保存其状态。

这个要求对于 shuffle 来说尤为重要，在 shuffle 期间，Spark 执行器首先将 map 输出写出到本地磁盘，然后在其他执行器需要获取这些输出结果时扮演服务器的角色。如果发生任务滞后（即某些任务比其他同等任务运行更长的时间），动态分配策略将在 shuffle 完成前移除该执行器，在这种情况下，由这个执行器所写出的 shuffle 文件得进行重新计算。

保存 shuffle 文件的解决方法是，使用额外的 shuffle 服务，具体在 Spark 1.2 配置中有说明。这个服务指的是在集群的每个节点上长期运行的一个进程，独立于 Spark 应用和他们的执行器，如果这项服务被激活，spark 执行器会从这个服务中获取 shuffle 文件。这就意味着由执行器写出的 shuffle 状态可能会超出这个执行器的生命周期而继续保留。

除了写 shuffle 文件，执行器也会缓存数据到磁盘或内存中，然而在移除执行器后，其缓存数据变得不再可访问，在 Spark1.2 版本中暂时还没有针对于此的解决方法，在后续发布版中，缓存数据可能会通过一个与借助于额外的 shuffle 服务来保存 shuffle 文件具有类似精髓的堆外存储系统进行保存。

3. 应用内的调度

在一个给定的 spark 应用（即 SparkContext 的实例）中，如果这些作业由不同的线程提交的话，多个平行的作业可以同时运行。在这一节中的“作业”，我们指的是一个 spark 的 action（比如 save 或者 collect 操作）和任何需要评估 action 所运行的任务。Spark 的调度器是完全线程安全的并且可以支持应用为多个请求提供服务（例如为多用户提供查询）。

默认的，Spark 的调度器使用 FIFO 模式运行作业。每一个作业都会被分割成多个“阶段”（例如 map， reduce），并且第一个作业在其阶段中存在任务运行的期间会优先获取所有可以获得的资源，然后是第二个作业获得优先权，以此类推。如果排在作业调度队列最前面的作业不需要使用整个集群，之后的作业可以立即启动，但是如果最前面的作业需要占用绝大多数资源，那么接下来的作业将会明显的被耽搁。

在 Spark 的 0.8 版本之后，可以通过配置来使不同的作业之间能够公平共享资源，在这种模式下，Spark 将不同作业的任务放置到一个循环模式下，从而使得所有的作用能够获得相对平均的集群资源。这就意味着短作业可以在长任务运行的时候立马启动并能够获得一个非常好响应时间而不用去等待长任务去完成。这个模式对于多个用户来说是最好的设置方案。

为了能够使用公平调度器，在配置 SparkContext 的时候可以通过设置 spark.scheduler.mode 为 fair

```
val conf = new SparkConf().setMaster(...).setAppName(...)
```

```
conf.set("spark.scheduler.mode", "FAIR")
val sc = new SparkContext(conf)
```

1）公平调度池

公平调度也能够支持将作业分组到各个池中，并且为每一个池分配不同的调度选项（例如 权重）。这在为一些非常重要的作业生成一个优先级高的池的时候非常有用，或者按用户分池，把一个用户的作业集中起来并通过用户来分配资源而不管这个用户具有多少正在运行的作业。这种方式是 Hadoop Fair scheduler 的仿照。

在没有任何干预的情况下，新提交的作业将会进入到默认的池中，但是作业的池可以通过在提交作业的 Spark Context 线程中添加 spark.scheduler.pool 到 "local property "以设置。可以按如下的方式做

```
// Assuming sc is your SparkContext variable
sc.setLocalProperty("spark.scheduler.pool", "pool1")
```

在设置了这个属性之后，所有通过这个线程（线程中的 RDD.save count collect 等等操作）提交的作业将会使用这个池的名字。这种设置是基于线程的，这使得一个线程能够代表一个用户运行很多的作业变得非常方便。如果你想清除某个线程所关联的池，按照如下调用就可以

```
sc.setLocalProperty("spark.scheduler.pool", null)
```

2）池的默认行为

默认的，每一个池对于集群来说都是公平的（跟默认池中每一个作业对应集群是公平的一样），但是在每一个池中，作业是按照 FIFO 排序运行。比如，如果你想为每一个用户建立一个池，这意味着每一个用户对应集群来说将会等同对待，每一个用户的查询都会按照顺序执行而不是之后的查询获取这个用户以前查询的资源

3）配置池的属性

池的某些特定的属性可以通过配置文件来进行修改，每一个池支持以下三种属性：

- schedulingMode: 这个属性可以为 FIFO 或者是 FAIR，来控制池中队列里面的作业是先出的优先获取资源还是公平的共享资源
- weight:这个可以设置集群中不同池的权重，默认的，所有的池权重都为 1，如果你把某个特定的池的权重设置为 2，他相对与其他的池来说将会获得 2 倍的资源。如果设置一个很高的值例如 1000 ,这使得这个池能够相对应其他的池总是能够优先获取资源，这个权重为 1000 的池总是能够在它的作业活跃的时候执行任务
- minShare: 除了大体上的设置，每一个池可以按照管理员的意愿给予一个最小的分配（例如 cpu 的核数）。公平的调度器总是会在先满足所有活跃的池的最小的分配值然后重新通过权重重新分配剩余的资源。这个 minShare 属性使得每个资源池总是能够获取一部分资源，而不用设置一个相对于其他作业非常高的权重。默认的每一个池的 minShare 这个属性为 0

池的属性可以通过生成一个类似于 conf/fairscheduler.xml.template 的 xml，或者是在 SparkConf 中设置 spark.scheduler.allocation.file

```
conf.set("spark.scheduler.allocation.file", "/path/to/file")
```

这个 XML 的格式为每一个池都会对应一个 <pool> 的元素 ,在里面有不同的元素对应于不同的设置，例如

```
<?xml version="1.0"?>
<allocations>
   <pool name="production">
       <schedulingMode>FAIR</schedulingMode>
       <weight>1</weight>
       <minShare>2</minShare>
    </pool>
   <pool name="test">
       <schedulingMode>FIFO</schedulingMode>
       <weight>2</weight>
       <minShare>3</minShare>
   </pool>
</allocations>
```

在 conf/fairscheduler.xml.template 中有一个完整的例子。需要指出的是不在 xml 文件中配置的任何池的属性都会为默认属性值（scheduling mode FIFO, weight 1, and minShare 0）。

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