打造价值40万Offer的朋友圈
数据人的宝藏朋友圈
欢迎邀请你的同事同学参观

公众号：import_bigdata
知　乎：https://www.zhihu.com/people/wang-zhi-wu-66
CSDN ：https://blog.csdn.net/u013411339
Github：https://github.com/wangzhiwubigdata/God-of-Bigdata

本文已经加入「大数据成神之路PDF版」中提供下载，你可以关注公众号，后台回复：「**PDF**」即可获取。

更多PDF下载可以参考：**《重磅,大数据成神之路PDF可以分类下载啦!》**

Spark重点难点系列：

- **《【Spark重点难点01】你从未深入理解的RDD和关键角色》**
- **《【Spark重点难点02】你以为的Shuffle和真正的Shuffle》**
- **《【Spark重点难点03】你的数据存在哪了?》**
- **《【Spark重点难点04】你的代码跑起来谁说了算？(内存管理)》**

前言

之前我们成功完成了Flink重点难点部分的学习了。很多同学可能还没有意识到,你已经把Flink这个框架中最关键的部分掌握了。Flink的重点难点部分就是我列在这里的部分：

《我把Flink的重点和难点部分更新完了》

这个系列中有一部分是我写的，也有一些是我从Flink的中文社区，各个网站找到的一手资料。如果你对Flink一窍不通，是个初学者，那么可以参考：

- **《193篇文章暴揍Flink，这个合集你需要关注一下》**

截止目前已经有200+的文章，基本上算是涵盖了Flink的方方面面，基本上你可以拿捏这个框架了。

如果你还嫌不够，看这里：

- **《我在B站读大学，大数据专业》**

这在这里总结了B站上讲的非常不错的资源推荐给大家。其中的Flink部分有下面几个：

Flink

再次给清华大佬跪下了。Flink的这个视频我在群里跟很多小伙伴推荐过了。

此视频一出，B站吃瓜群众惊呼：武老师，发生甚么事了？！

> Java版Flink(武老师清华硕士，原IBM-CDL负责人)
> https://www.bilibili.com/video/BV1qy4y1q728

清华大佬不满足于此，竟然给Flink SQL出了单独的视频！

> Flink SQL(武老师清华硕士，原IBM-CDL负责人)
> https://www.bilibili.com/video/BV12k4y1z7LM

相信我，这些资源足够一个新手从0开始并且学习并胜任一些例如实时数仓、业务开发的工作了。

今天我们开更Spark了。

关于Spark的整体学习你可以看这里：

- **《我们在学习Flink的时候，到底在学习什么？》**

大家都知道2019年Flink开源，可能抢了一部分热度，很多公司都开始转向对Flink的研究。事实上Spark在欧洲和北美异常火爆，很多公司的很多任务估计都还在用Spark，并且在离线的批处理上，Spark的稳定程度超出你的想象。

未来在数据开发方向，Spark的重心会转移到Spark SQL，并且官方推荐大家使用DataFrame Based方式开发Spark程序。 其中的Spark Streaming和Structured Streaming可能真的要湮灭在历史的长河中了。

所以我们关于Spark的学习重点可以放在Spark Core、Spark SQL以及SPark Mlib等模块上。另外两块可以通过我推荐的学习资源去学习。

Spark Core

关于RDD你需要知道的

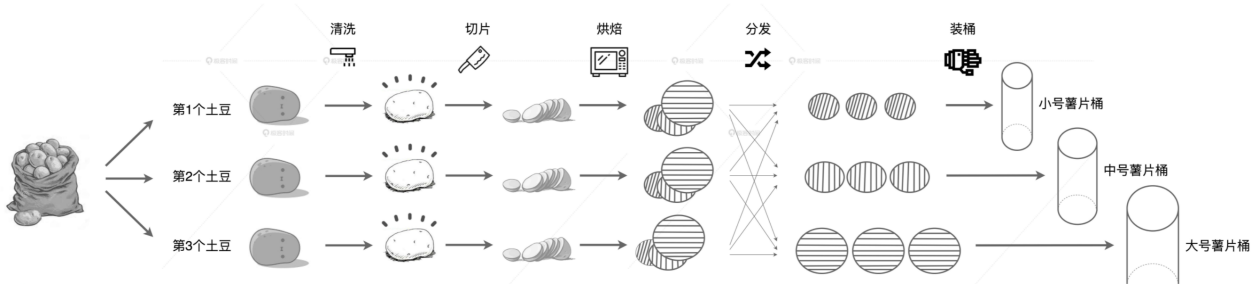
你肯定在网上看到过一大堆的废话了。比如下面这两段：

> RDD 是 Spark 提供的最重要的抽象概念，它是一种有容错机制的特殊数据集合，可以分布在集群的结点上，以函数式操作集合的方式进行各种并行操作。

> 通俗点来讲，可以将 RDD 理解为一个分布式对象集合，本质上是一个只读的分区记录集合。每个 RDD 可以分成多个分区，每个分区就是一个数据集片段。一个 RDD 的不同分区可以保存到集群中的不同结点上，从而可以在集群中的不同结点上进行并行计算。

恕我直言， 这两段废话狗看了都摇头 。你在说什么东西??

如果你看过吴磊老师的《Spark性能调优实战》你对RDD的理解应该会上一个大台阶。



刚从地里挖出来的土豆食材、清洗过后的干净土豆、生薯片、烤熟的薯片，流水线上这些食材的不同形态，就像是 Spark 中 RDD 对于不同数据集合的抽象。

RDD 具有 4 大属性，分别是 partitions 、 partitioner 、 dependencies 和 compute 属性。正因为有了这 4 大属性的存在，让 RDD 具有分布式和容错性这两大最突出的特性。

- partitions: 图中每一颗土豆就是 RDD 中的数据分片，3 颗土豆一起对应的就是 RDD 的 partitions 属性。
- partitioner: 根据尺寸的不同，即食薯片会被划分到不同的数据分片中。像这种数据分片划分规则，对应的就是 RDD 中的 partitioner 属性。
- dependencies: 每种食材形态都依赖于前一种食材，这就像是 RDD 中 dependencies 属性记录的依赖关系
- compute: 不同环节的加工方法，对应的刚好就是 RDD 的 compute 属性。

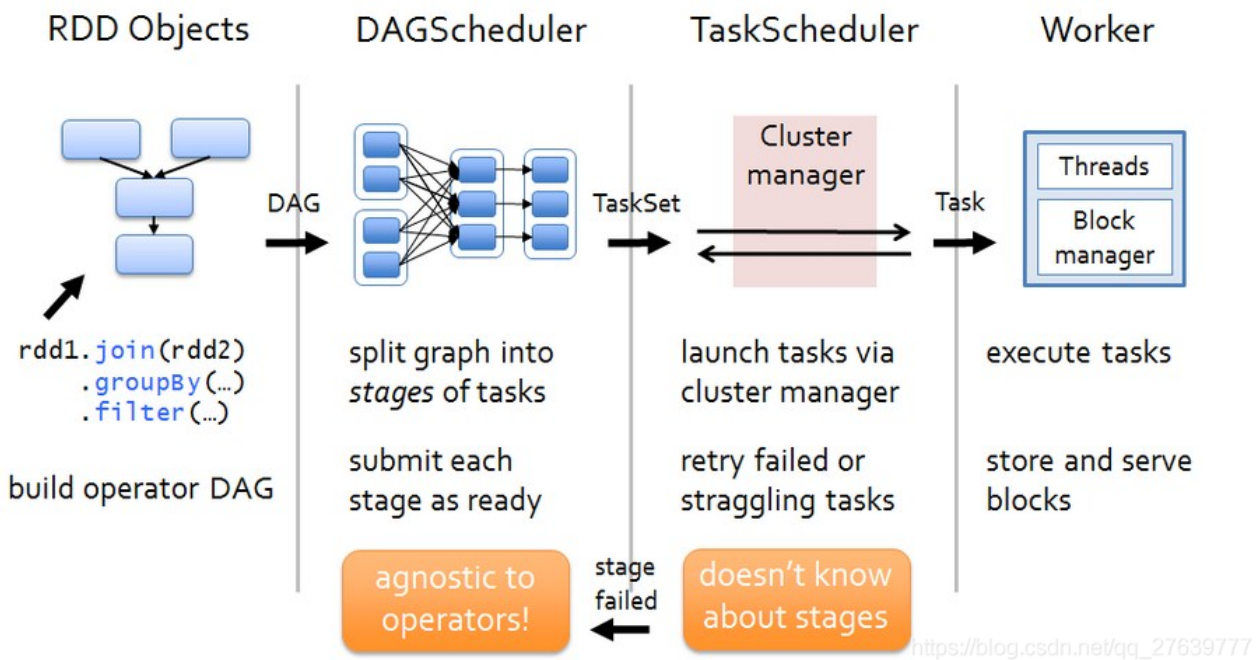
现在你理解RDD了吗？正确理解RDD会对你学习Spark有很深刻的影响、

Spark中的关键角色

DAGScheduler

DAGScheduler是一家公司的总架构师。

DAGScheduler把DAG拆分成很多的Tasks,每组的Tasks都是一个 Sage,解析时是以Shuffle为边界反向解析构建Stage,每当遇到 Shuffle,就会产生新的Stage,然后以一个个TaskSet(每个Stage封装一个TaskSet)的形式提交给底层调度器TaskScheduler。

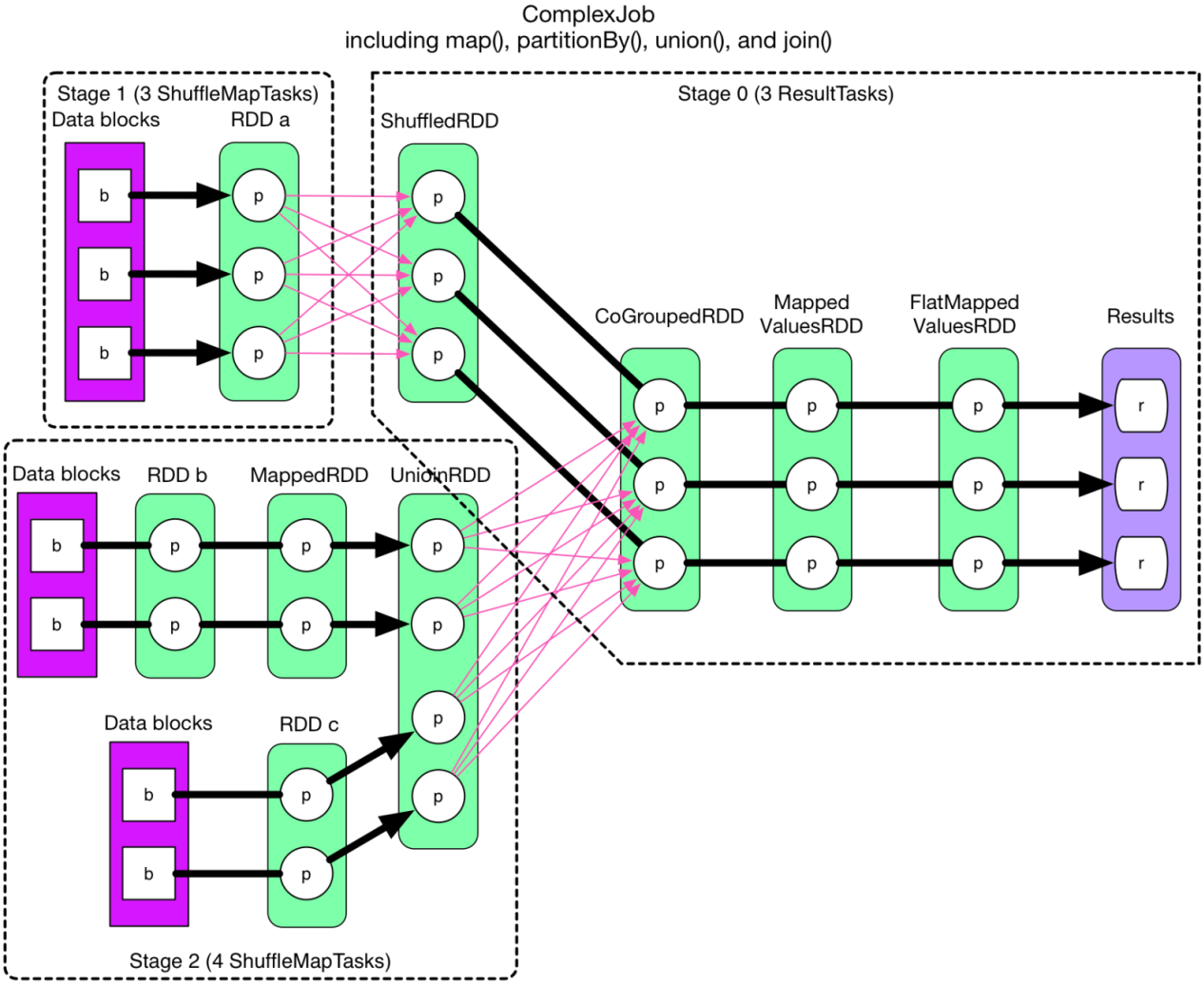


它的三个主要职责：

- 根据用户代码构建DAG;
- 以Shuffle为边界切割Stages;
- 基于Stages创建TaskSets，并将TaskSets提交给TaskScheduler 请求调度。

DAGScheduler划分Stage的原理

Spark在分布式环境下将数据分区,然后将作业转化为DAG, 并分阶段进行 DAG的调度和任务的分布式并行处理。DAG将调度提交给DAGScheduler, DAGScheduler调度时会根据是否需要经过Shuffle过程将Job划分为多个 Stage。



在上图中,RDD a到ShuffledRDD之间,以及UnionRDD到CoGroupedRDD 之间的数据需要经过Shuffle过程,因此RDDa和UnionRDD分别是Stage1跟Stage3和Stage2跟Stage3的划分点。而ShuffledRDD到CoGroupedRDD 之间,以及RDDb到MappedRDD到UnionRDD和RDDc到UnionRDD之间的数据不需要经过Shuffle过程。因此,ShuffledRDD和CoGroupedRDD的依赖是窄依赖,两个RDD属于同一个Stage3,其余RDD划分为2个Stage。Stage1和Stage2是相对独立的,可以并行运行。Stage3则依赖于Stage1和Stage2的运行结果,所以Stage3最后执行。

由此可见,在DAGScheduler调度过程中,Stage阶段换份是依据作业是否有Shuffle过程,也就是存在ShuffleDependency的宽依赖时,需要进行Shuffle,此时才会将作业划分为多个Stage。

SchedulerBackend

SchedulerBackend是一家公司的人力资源总监。

对于集群中可用的计算资源，SchedulerBackend 用一个叫做 ExecutorDataMap 的数据结构，来记录每一个计算节点中 Executors 的资源状态。

这里的 ExecutorDataMap 是一种 HashMap，它的 Key 是标记 Executor 的字符串，Value 是一种叫做 ExecutorData 的数据结构。ExecutorData 用于封装 Executor 的资源状态，如 RPC 地址、主机地址、可用 CPU 核数和满配 CPU 核数等等，它相当于是对 Executor 做的"资源画像"。

SchedulerBackend 可以同时提供多个 WorkerOffer 用于分布式任务调度。WorkerOffer 这个名字起得很传神，Offer 的字面意思是公司给你提供的工作机会，到了 Spark 调度系统的上下文，它就变成了使用硬件资源的机会。

SchedulerBackend 与集群内所有 Executors 中的 ExecutorBackend 保持周期性通信，双方通过 LaunchedExecutor、RemoveExecutor、StatusUpdate 等消息来互通有无、变更可用计算资源。

TaskScheduler

TaskScheduler是一家公司干活的总负责人。

TaskScheduler的核心任务是提交TaskSets到集群运算并汇报结果。

他主要做三件事：

1. 为TaskSet创建和维护一个TaskSetManager，并追踪任务的本地性以及错误信息。
2. 遇到Straggle任务时，会放到其他节点进行重试。
3. 向DAGScheduler汇报执行情况，包括在Shuffle输出丢失的时候报告 fetch failed错误等信息。

每个任务都是自带本地倾向性的，换句话说，每个任务都有自己擅长做的事情。

ExecutorBackend

ExecutorBackend是分公司的人力资源主管。

ExecutorBackend拿到Task任务之后,随即把Task派发给分公司的工人。这些工人,就是Executors线程池中一个又一个的CPU线程,每个线程负责处理一个Task。

每当 Task 处理完毕，这些线程便会通过 ExecutorBackend，向 Driver 端的 SchedulerBackend 发送 StatusUpdate 事件，告知 Task 执行状态。接下来，TaskScheduler 与 SchedulerBackend 通过接力的方式，最终把状态汇报给 DAGScheduler。

直到整个Spark程序中的所有Task执行完毕。一次完整的Spark任务就执行结束了。