法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象学院
  - 新浪微博：小象AI学院

大纲

- Spark产生背景
- Spark工作原理
- Spark Shell

大纲

Spark产生背景

Spark产生背景

- MapReduce局限性
  - 仅支持Map和Reduce两种语义操作
  - 处理效率低，耗费时间长
  - 不适合处理迭代计算、交互式处理、实时流处理等
  - 更多的应用于大规模批处理场景

Spark产生背景

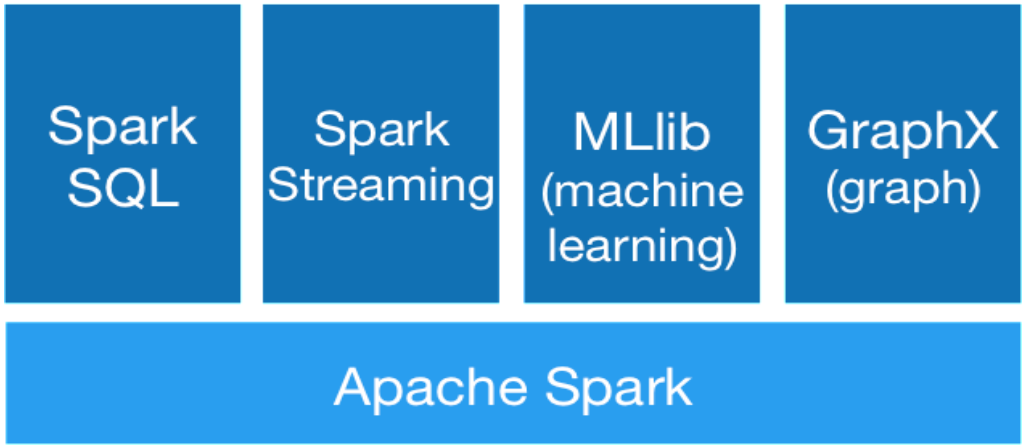
- 计算处理框架种类多，选型复杂
  - 批处理：MapReduce、Hive、Pig
  - 流式计算：Storm
  - 交互式计算：Impala、Presto
  - 机器学习算法：Mahout
- 希望能够简化技术选型，在一个统一的框架下，能够完成批处理、流式计算、交互式计算、机器学习算法等

大纲

Spark工作原理

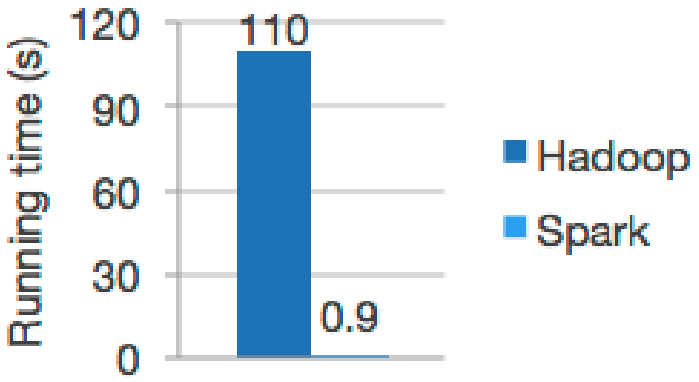
Spark简介

- 由加州大学伯克利分校的AMP实验室开源
- 大规模分布式通用计算引擎
- 具有高吞吐、低延时、通用易扩展、高容错等特点
- 使用Scala语言开发，提供了丰富的开发API，支持Scala、Java、Python、R开发语言
- Spark提供多种运行模式



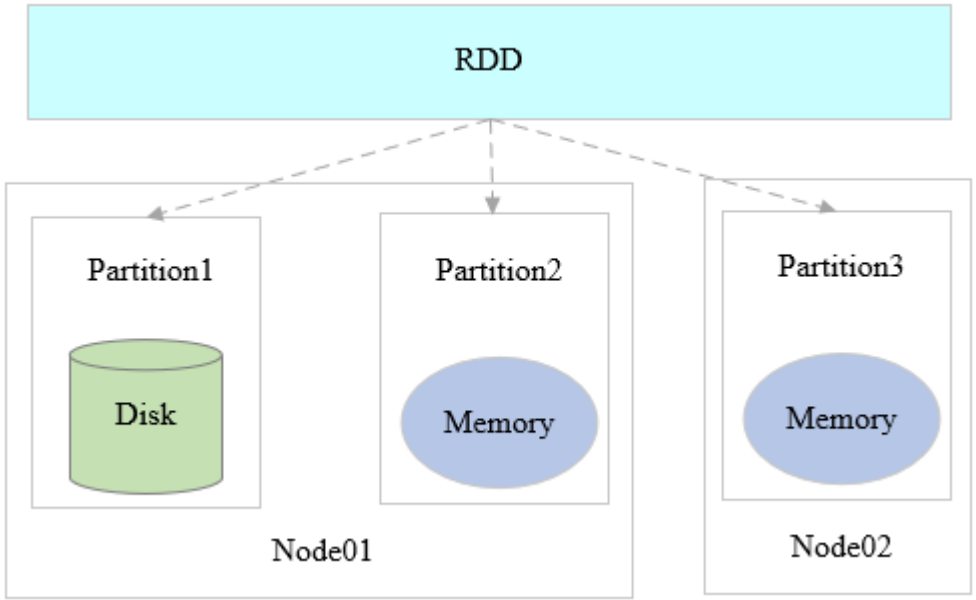
Spark特点

- 计算高效
  - 使用内存计算引擎，提供Cache缓存机制支持迭代计算或多次数据共享，减少数据读取的IO开销
  - DAG引擎，减少多次计算之间中间结果写到HDFS的开销
  - 使用多线程池模型来减少task启动开销，shuffle过程中避免不必要的sort操作以及减少磁盘IO操作
- 通用易用
  - 提供了丰富的开发API，支持Scala、Java、Python、R开发语言
  - 集成批处理、流处理、交互式计算、机器学习算法、图计算
- 运行模式多样
  - Local、Standalone、Yarn、Mesos



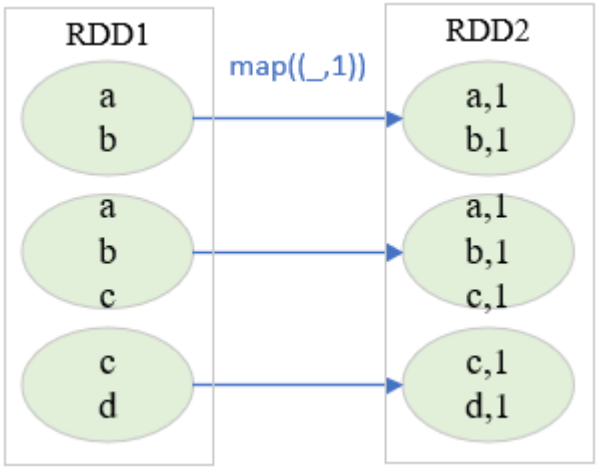
Spark核心概念-RDD

- RDD：Resilient Distributed Datasets弹性分布式数据集
  - Spark基于RDD进行计算
  - 分布在集群中的只读对象集合（由多个Partition构成）
  - 可以存储在磁盘或内存中
  - 可以通过并行转换操作构造
  - 失效后自动重构



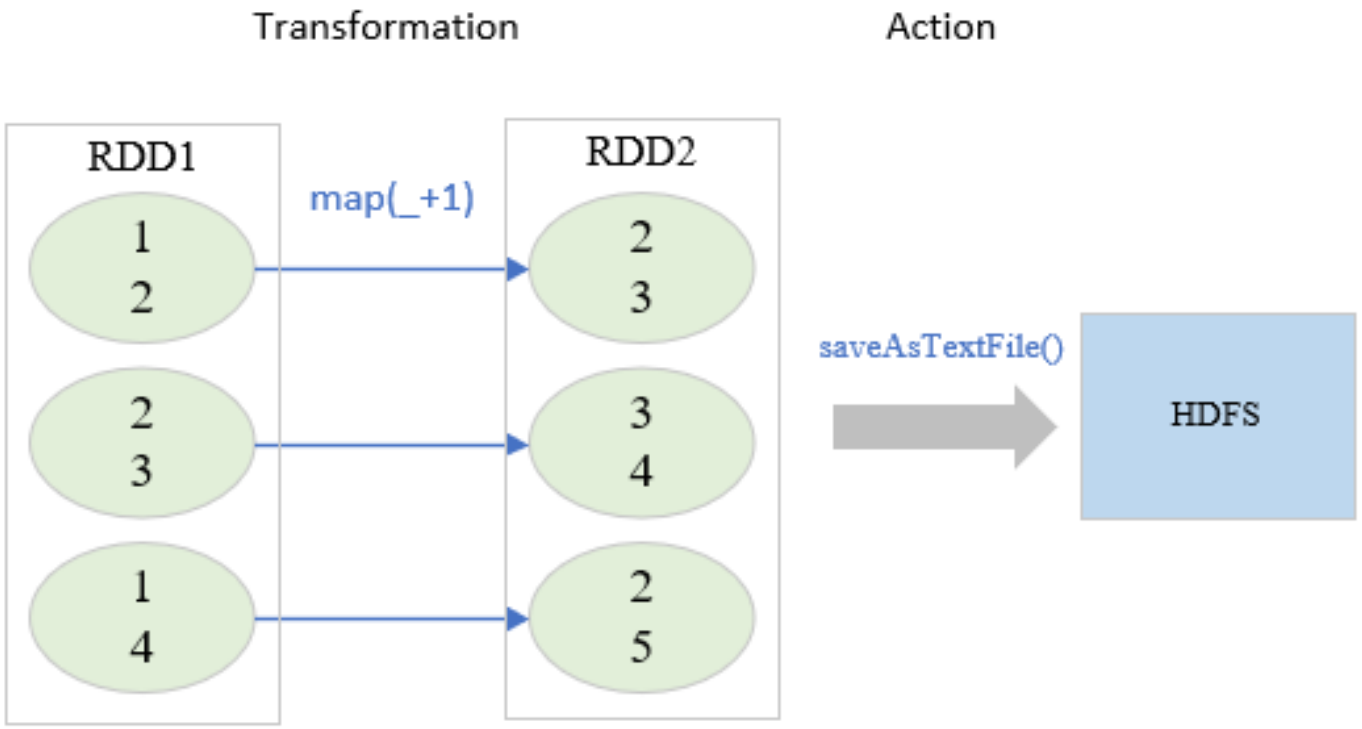
RDD操作（Operator）

- Transformation
  - 将Scala集合或者Hadoop数据集构造一个新的RDD
  - 通过已有的RDD产生新的RDD
  - 只记录转换关系，不触发计算
  - 如：map、filter等
- Action
  - 通过RDD计算得到一个或者一组值
  - 真正触发执行
  - 如：count、collect、saveAsTextFile



RDD操作示例

- rdd1.map(_+1).saveAsTextFile("hdfs://node01:9000/")



RDD常用Transformation

函数名称	描述
map (func)	接收一个处理函数并行处理源RDD中的每个元素，返回与源RDD元素一一对应的新RDD
filter (func)	并行处理源RDD中的每个元素，接收一个处理函数根据定义的规则对RDD中的每个元素进行过滤处理，返回处理结果为true的元素重新组成新的RDD
flatMap (func)	与map函数相似， flatMap是map和flatten的组合操作， map函数返回的新RDD包含的元素可能是嵌套类型，flatMap接收一个处理嵌套类型数据的函数，将嵌套类型的元素展开映射成多个元素组成新的RDD
union (otherDataset)	将两个RDD进行合并，返回结果RDD中元素不去重
distinct ([numTasks]))	对RDD中元素去重
reduceByKey(func, [numTasks])	对KV类型的RDD中按Key分组，接收两个参数，第一个参数为处理函数，第二个参数为可选参数设置reduce的任务数， reduceByKey能够在RDD分区本地提前进行聚合运算，能够有效减少shuffle过程传输的数据量
sortByKey([ascending],[numTasks])	对KV类型的RDD内部元素按照Key排序，排序过程会涉及到Shuffle
join (otherDataset,[numTasks])	对KV类型的RDD关联，只能是两个RDD之间关联，超过两个RDD关联需要使用多次join函数，join函数只会关联出具有相同Key的元素，相当于SQL语句中的inner join
repartition (numPartitions)	对RDD重新分区接收一个参数numPartitions分区数

RDD常用Action

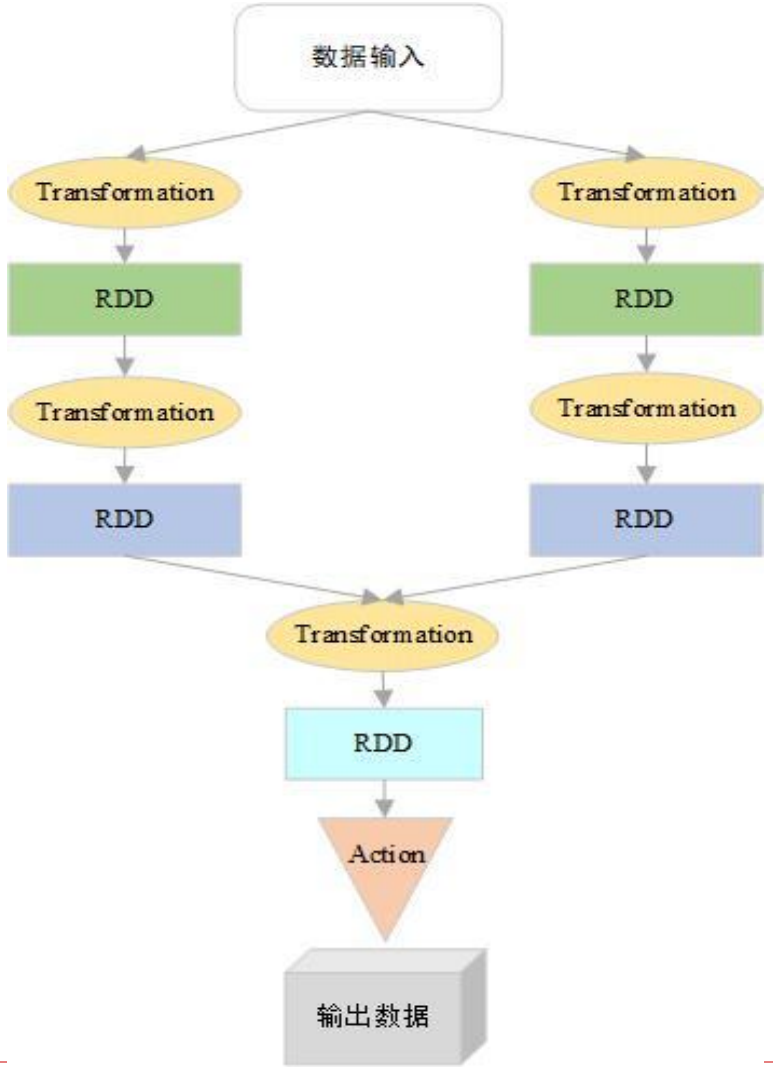
函数名称	描述
reduce(func)	处理RDD两两之间元素的聚集操作
collect()	返回RDD中所有数据元素
count()	返回RDD中元素个数
first()	返回RDD中的第一个元素
take(n)	返回RDD中的前n个元素
saveAsTextFile(path)	将RDD写入到文本文件，保存到本地文件系统或者HDFS中
saveAsSequenceFile(path)	将KV类型的RDD写入到SequenceFile文件保存到本地文件系统或者HDFS
countByKey()	返回KV类型的RDD每个Key有多少个元素
foreach(func)	遍历RDD中所有元素，接收参数为func函数，常用操作是传入println函数打印所有元素

Transformation与Action对比

- 接口定义方式不同
  - Transformation：RDD[X] -> RDD[Y]
  - Action：RDD[X] -> Z
- 执行计算方式不同
  - Transformation采用惰性执行方式，只记录RDD转化关系，不会触发真正计算执行
  - Action真正触发计算执行

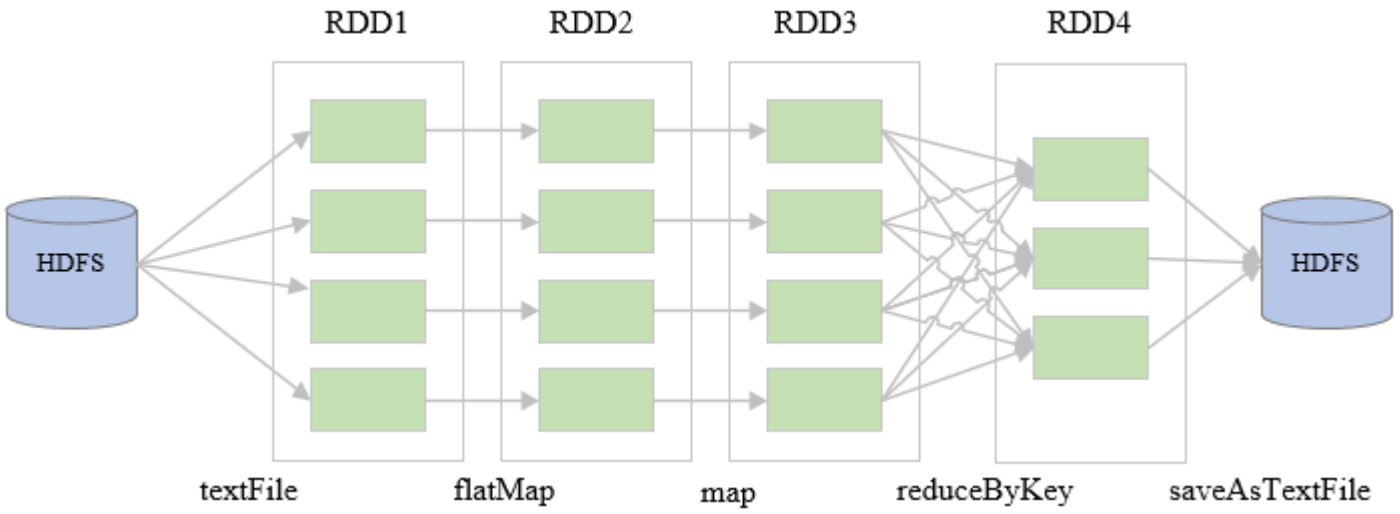
Transformation Lazy Execution

- 惰性执行



程序执行流程

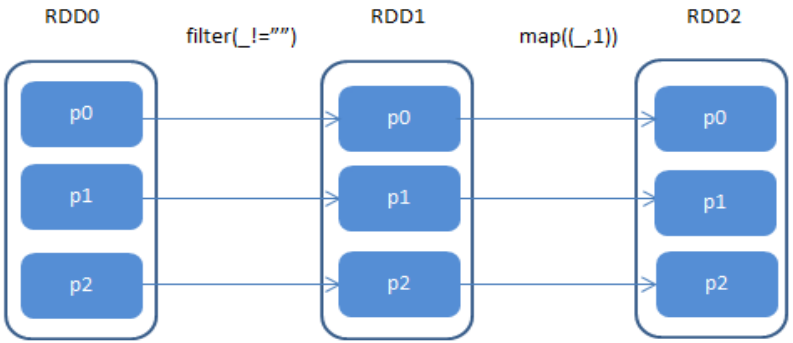
- Spark中的WordCount
  - val rdd1 = sc.textFile("hdfs://192.168.183.101:9000/data/wc/in")
  - val rdd2 = rdd2.flatMap(_.split("\t"))
  - val rdd3= rdd3.map((_,1))
  - val rdd4 = rdd3.reduceByKey(_ + _)
  - rdd4.saveAsTextFile(“hdfs://192.168.183.100:9000/data/wc/out”)



RDD Dependency依赖

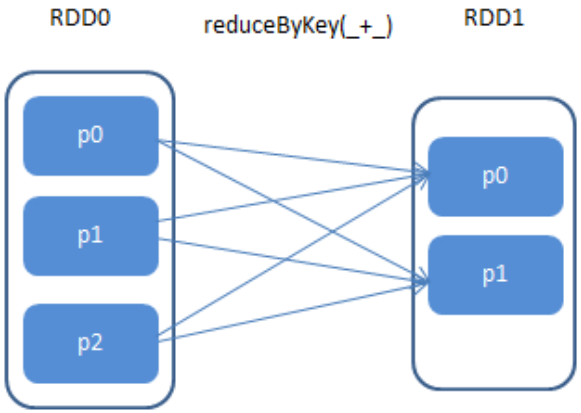
Narrow Dependency窄依赖

- 父RDD中的分区最多只能被一个子RDD的一个分区使用
- 子RDD如果只有部分分区数据丢失或者损坏只需要从对应的父RDD重新计算恢复



Shuffle Dependency宽依赖

- 子RDD分区依赖父RDD所有分区
- 子RDD如果部分分区或者全部分区数据丢失或者损坏需要从所有父RDD重新计算，相对窄依赖付出的代价更高，尽量避免宽依赖的使用



RDD Cache缓存

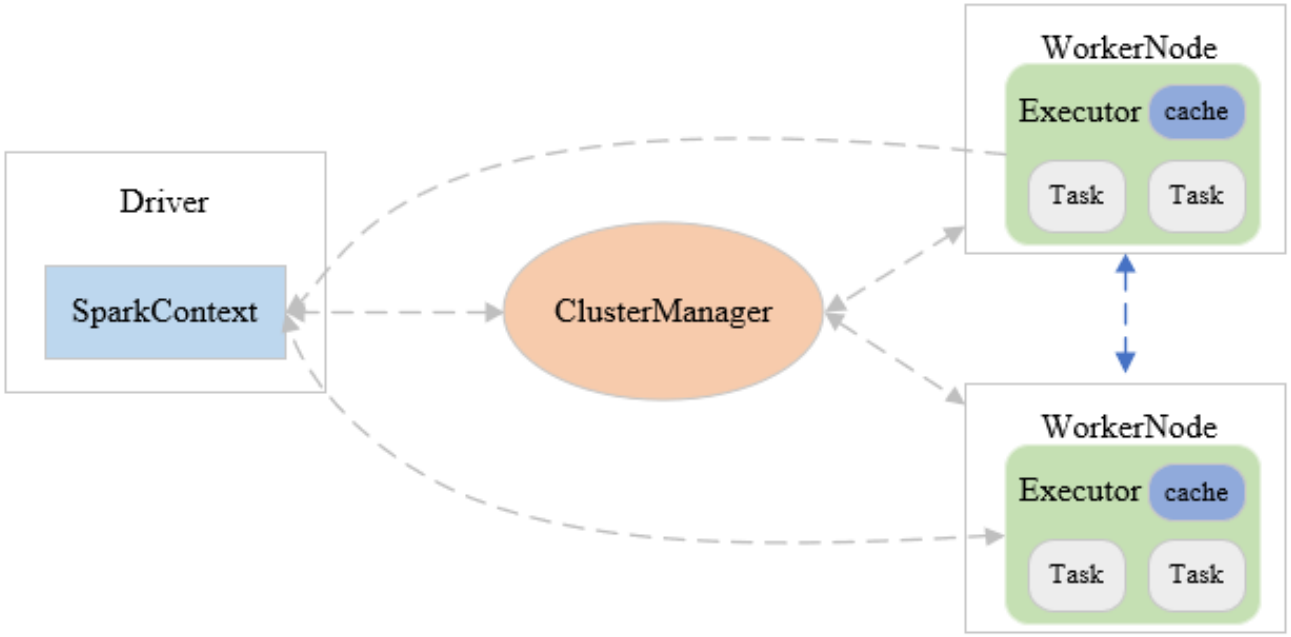
- Spark允许将RDD缓存到内存或磁盘上，方便重用，提高性能
- Spark提供了多种缓存级别
- 用户可以根据实际需求进行调整

```
val rdd = sc.textFile(inputArg)
rdd.cache()//实际上是调用了rdd.persist(StorageLevel.MEMORY_ONLY)
//data.persist(StorageLevel.MEMORY_AND_DISK)
```

缓存级别	描述
MEMORY_ONLY	RDD仅缓存一份到内存，默认级别
MEMORY_ONLY_2	将RDD分别缓存在集群的两个节点上，RDD在集群内存中保存两份
MEMORY_ONLY_SER	将RDD以 Java序列化对象的方式缓存到内存中，有效减少RDD在内存中占用的空间，读取的时候会消耗更多的CPU资源
MEMORY_ONLY_SER_2	将RDD以 Java序列化对象的方式缓存到内存中，将RDD分别缓存在集群的两个节点上，RDD在集群内存中保存两份
DISK_ONLY	RDD仅缓存一份到磁盘中
DISK_ONLY_2	RDD在集群中缓存两份到磁盘中
MEMORY_AND_DISK	RDD仅缓存一份到内存当内存中空间不足时会将部分RDD分区缓存到磁盘
MEMORY_AND_DISK_2	将RDD分别缓存在集群的两个节点上，当内存中空间不足时会将部分RDD分区缓存到磁盘，RDD在集群内存中保存两份
MEMORY_AND_DISK_SER	将RDD以 Java序列化对象的方式缓存到内存中当内存中空间不足时会将部分RDD分区缓存到磁盘，有效减少RDD在内存中占用的空间，读取的时候会消耗更多的CPU资源
MEMORY_AND_DISK_SER_2	将RDD以 Java序列化对象的方式缓存到内存中当内存中空间不足时会将部分RDD分区缓存到磁盘
OFF_HEAP	RDD将以序列化的方式缓存到JVM之外的存储空间Tachyon（Alluxio）中，与其他缓存模式相比减少了JVM垃圾回收开销，Spark执行程序的失败不会导致数据丢失， Spark与Tachyon已经能够很好的兼容，使用方便稳定
NONE	不使用缓存

Spark程序架构

- Driver：一个Spark程序有一个Driver，一个Driver创建一个Spark Context，程序的main函数运行在Driver中。Driver主要负责Spark程序的解析、划分Stage，调度Task到Executor上执行
- SparkContext：加载配置信息，初始化Spark程序运行环境，创建内部的DAGScheduler和TaskScheduler
- Executor：负责执行Driver分发的Task任务，集群中一个节点可以启动多个Executor，每个一个Executor通过多线程运行多个Task任务
- Task：Spark运行的基本单位，一个Task负责处理RDD一个分区的计算逻辑



Spark运行模式

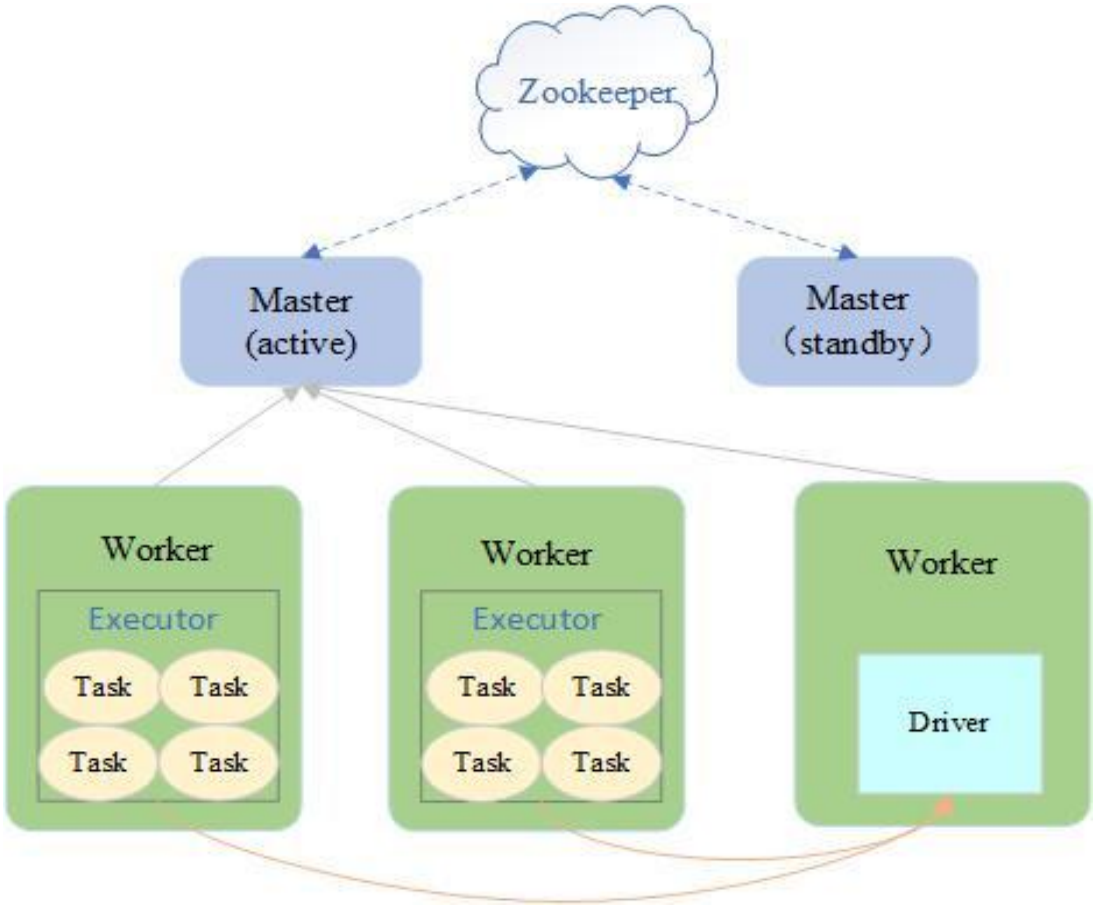
- Local本地模式
  - 单机运行，通常用于测试
- Standalone独立模式
  - Spark集群单独运行
- Yarn/Mesos
  - 运行在其他资源管理系统上，如Yarn、Mesos

Spark Local模式

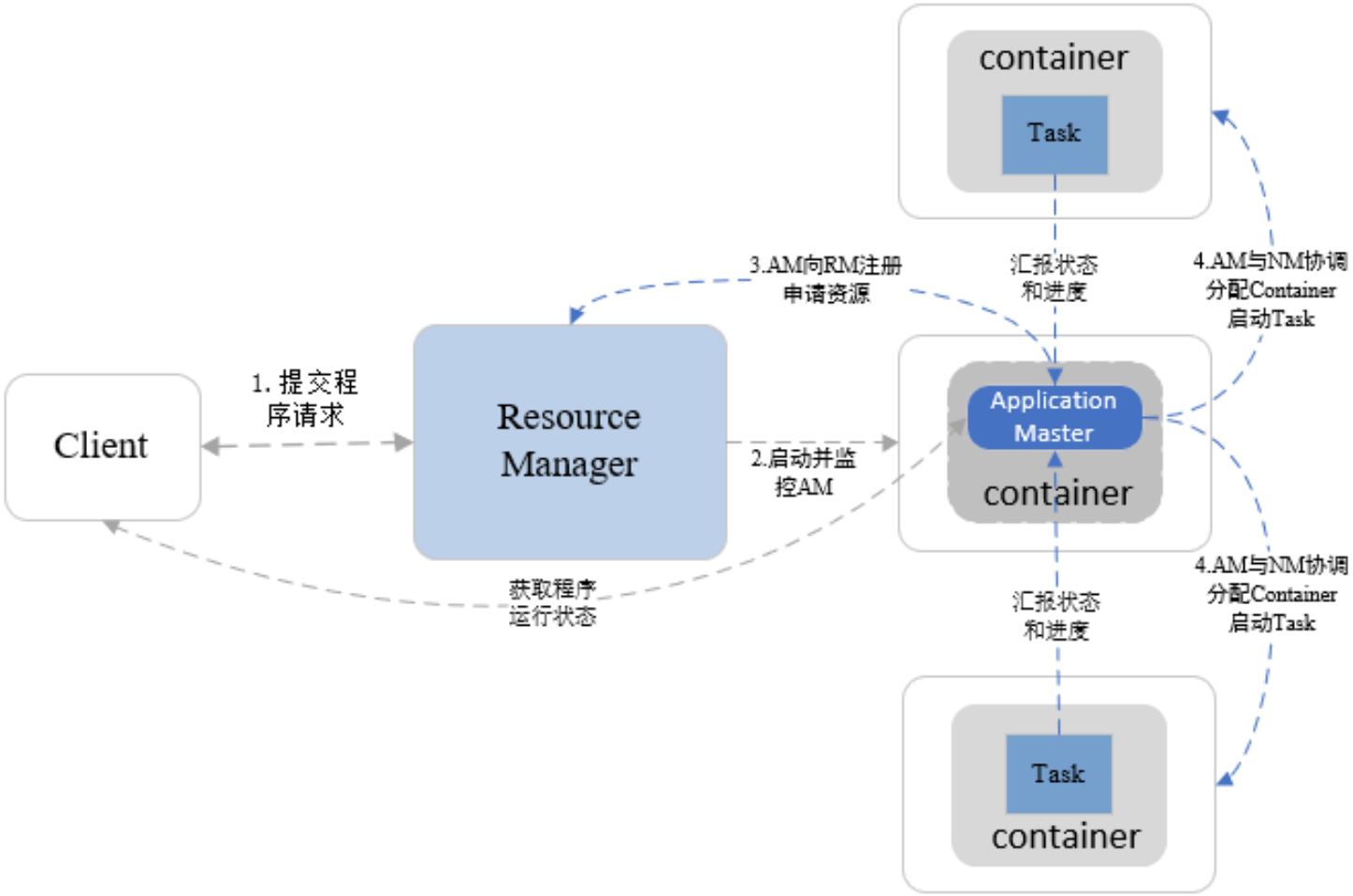
- Local本地模式
  - 将Spark应用以多线程方式，直接运行在本地，方便调试
- 本地模式分类
  - local：只启动一个线程运行executor
  - local[n]：启动n个线程运行executor
  - local[*]：启动跟cpu数目相同的executor

Spark Standalone模式

- Spark集群独立部署

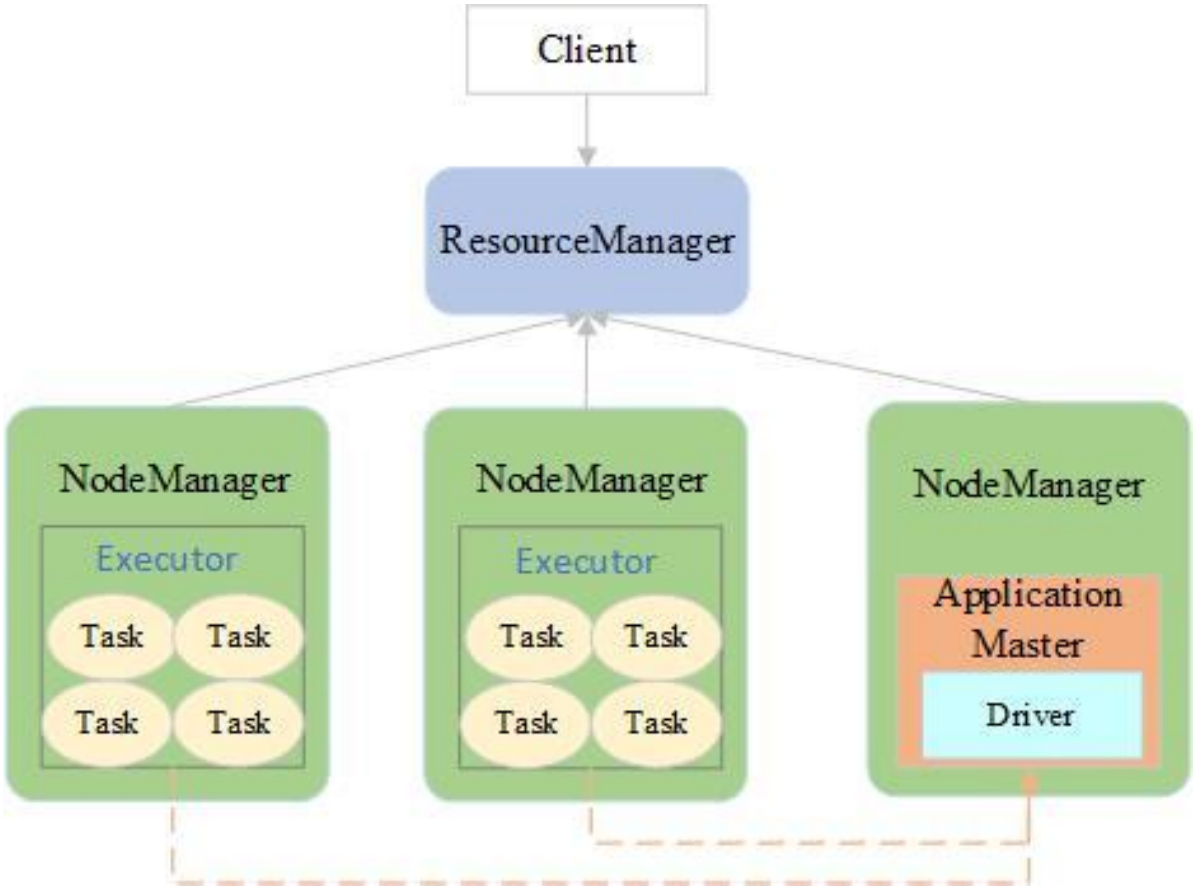


YARN程序运行流程


container
Task
3.AM向RM注册
申请资源
汇报状态
和进度
4.AM与NM协调
分配Container
启动Task
Client
1. 提交程
序请求
Resource
Manager
2.启动并监
控AM
Application
Master
container
获取程序
运行状态
汇报状态
和进度
4.AM与NM协调
分配Container
启动Task
Task
container

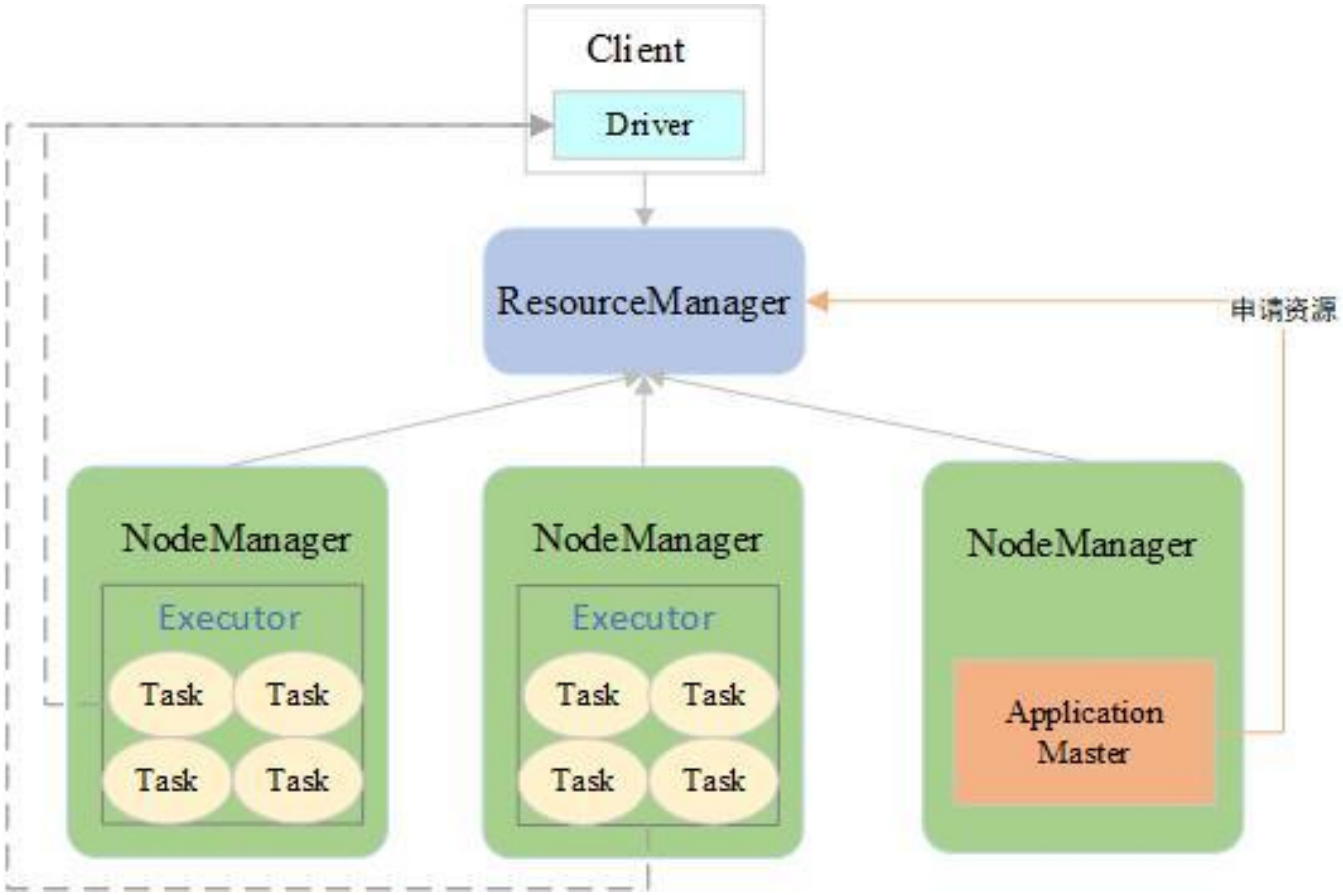
Spark YARN模式

- yarn-cluster

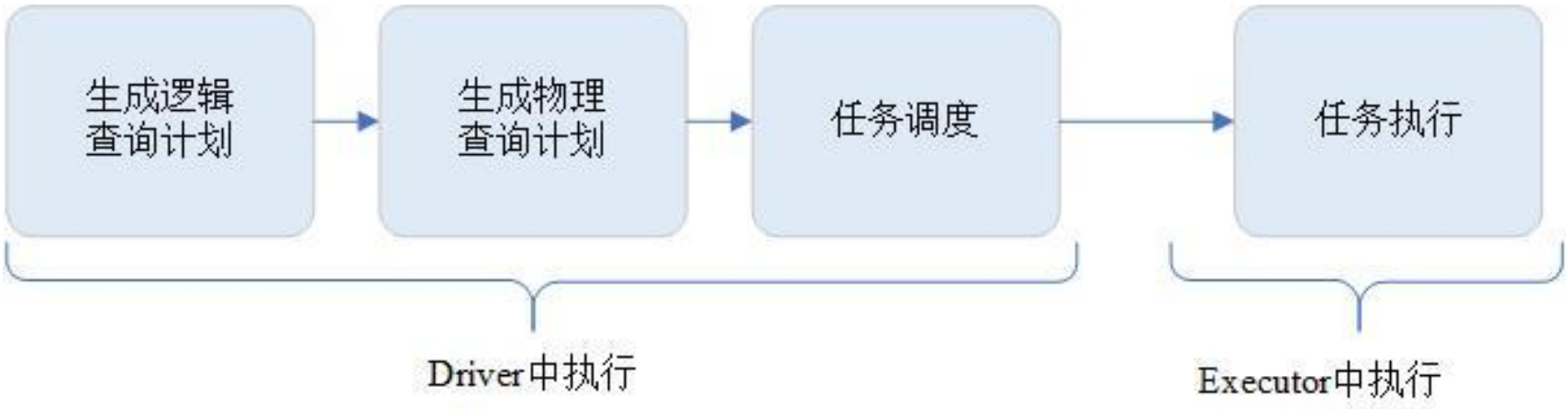


Spark YARN模式

- yarn-client

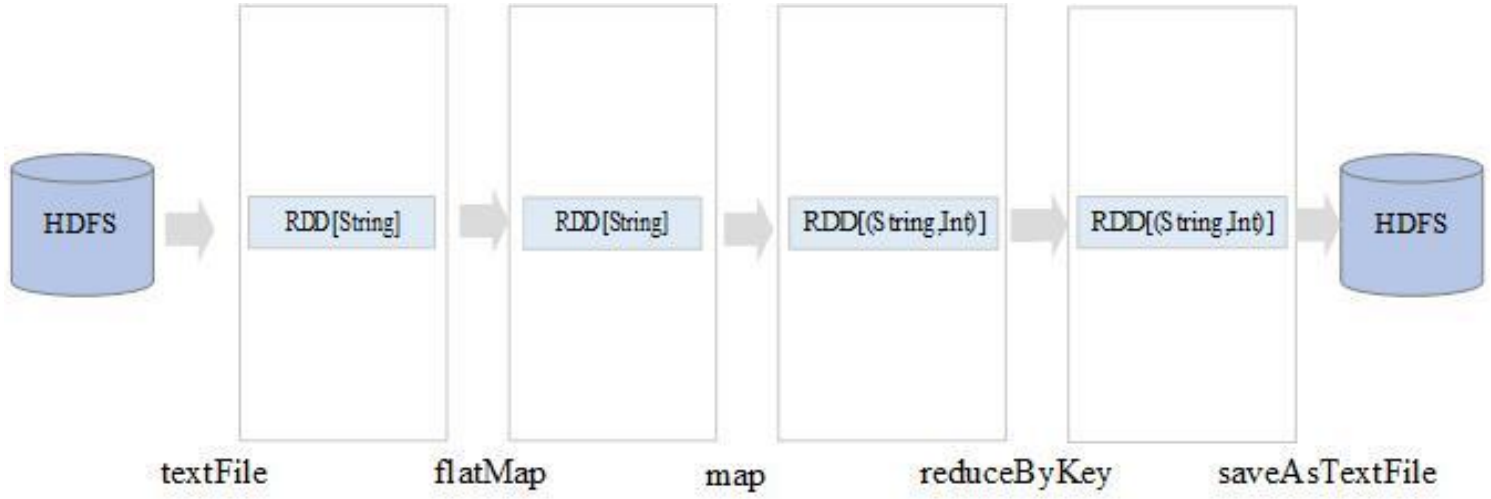


Spark内部执行流程

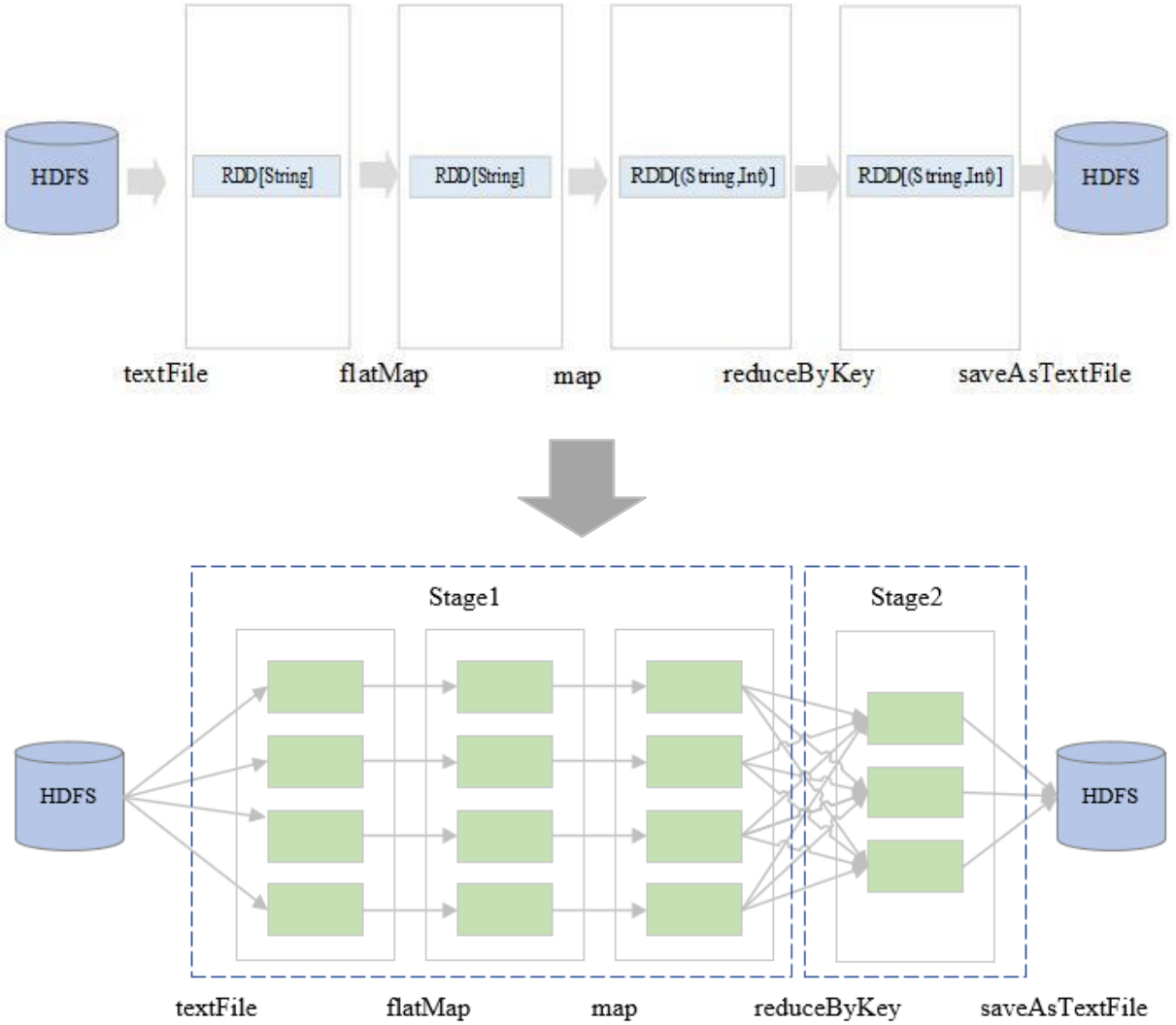


生成逻辑查询计划

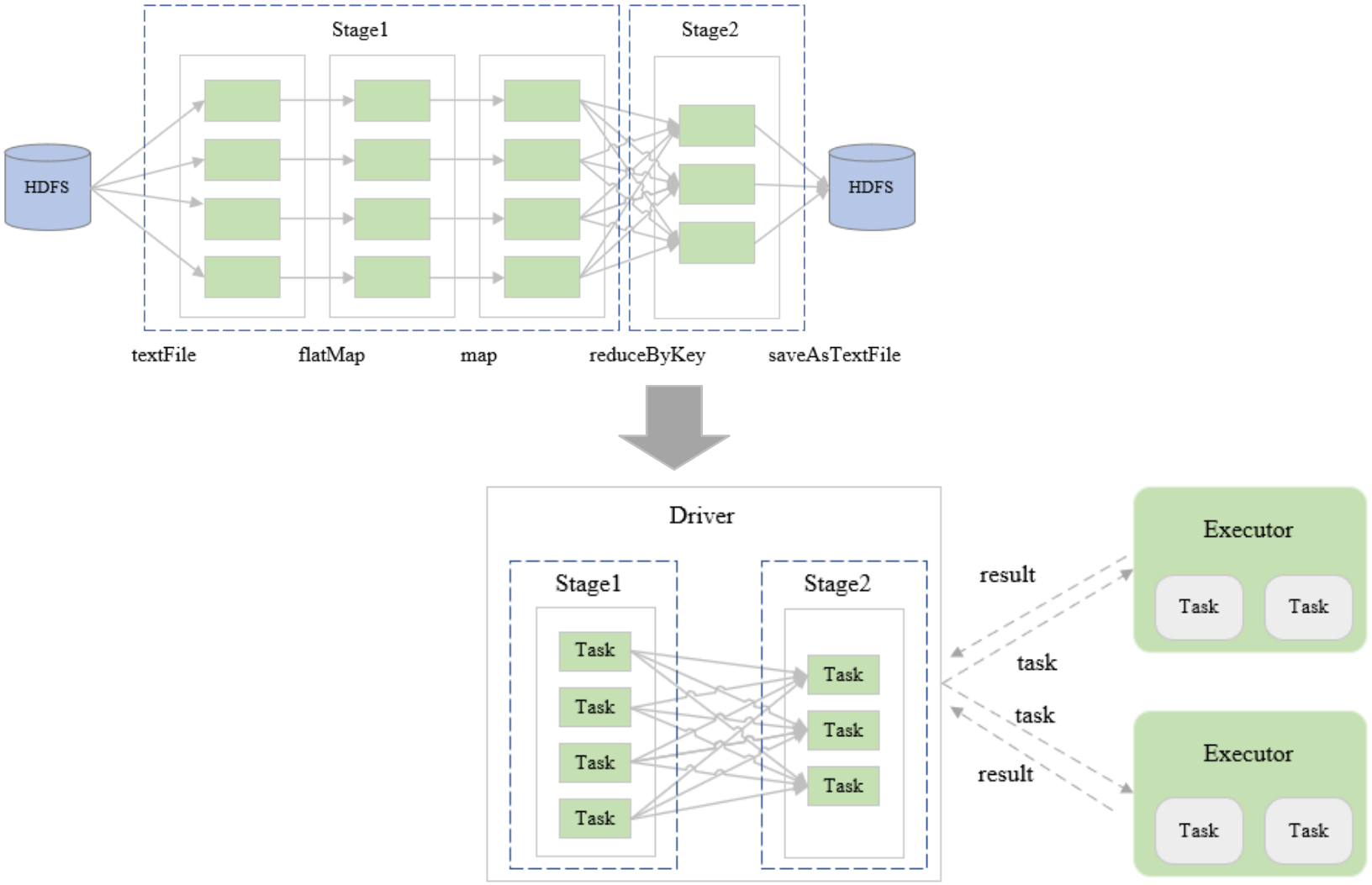
```
sc.textFile(inputArg)                    RDD[String]
    .flatMap(_.split("\t"))              RDD[String]
    .map((_,1))                          RDD[(String,Int)]
    .reduceByKey(_ + _)                  RDD[(String,Int)]
    .saveAsTextFile(outArg)
```



生成物理查询计划

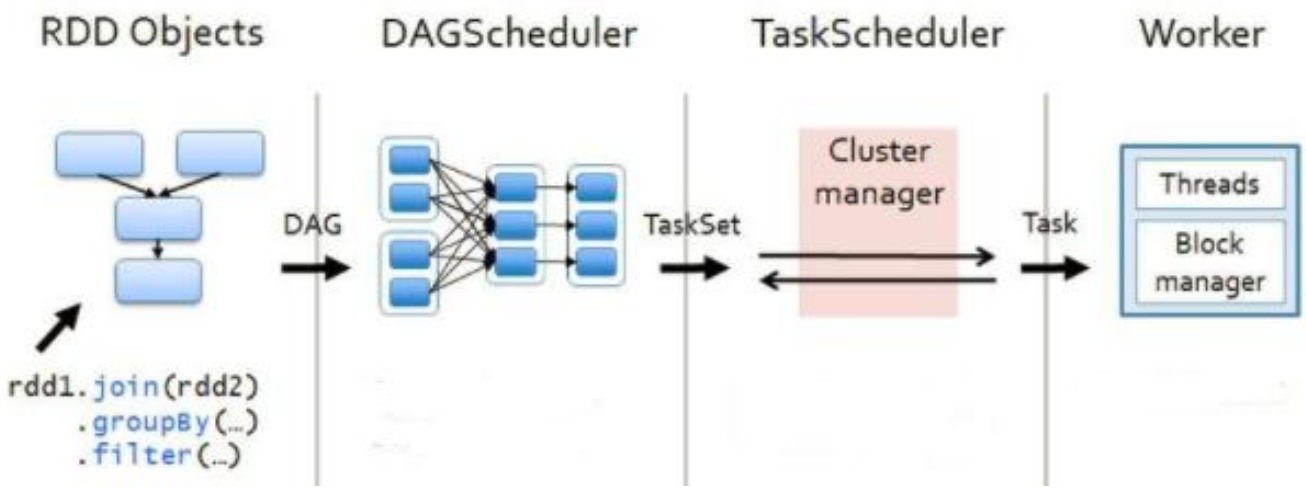

HDFS
RDD[String]
RDD[String]
RDD[(String,Int)]
RDD[(String,Int)]
HDFS
textFile
flatMap
map
reduceByKey
saveAsTextFile
Stage1
Stage2
HDFS
HDFS
textFile
flatMap
map
reduceByKey
saveAsTextFile

任务调度与执行


Stage1
Stage2
HDFS
HDFS
textFile
flatMap
map
reduceByKey
saveAsTextFile
Driver
Stage1
Stage2
Task
Task
Task
Task
Task
Task
Task
result
task
task
result
Executor
Task
Task
Executor
Task
Task

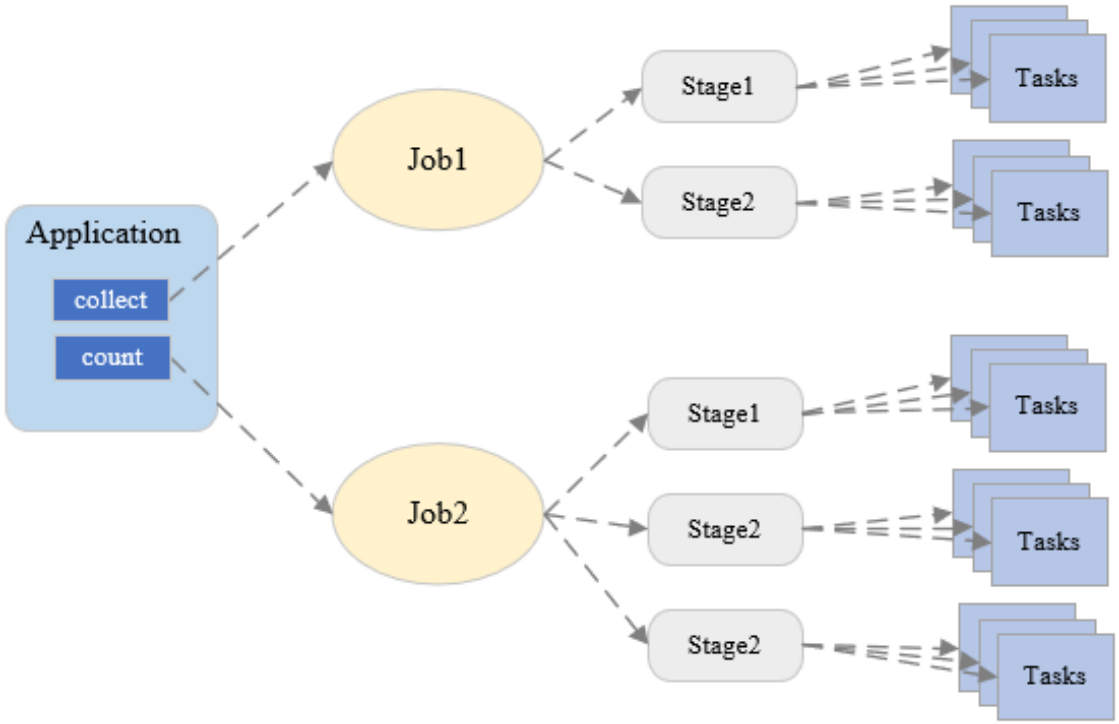
Spark调度模块

- DAG：Directed Acyclic Graph有向无环图
- DAGScheduler
  - 根据计算任务的依赖关系建立DAG
  - 根据依赖关系是否是宽依赖，将DAG划分为不同的Stage阶段
  - 将各个Stage中的Task组成的TaskSet提交到TaskScheduler
- TaskScheduler
  - 负责Application的Job调度
  - 重新提交执行失败的Task
  - 为执行速度慢的Task启动备份任务



Spark任务类型和Job划分

- Spark中task类型
  - ShuffleMapTask：除了最后一个输出Task，其他Task类型都是ShuffleMapTask
  - ResultTask：只有最后一个阶段输出的Task是ResultTask
- Appication中调用一次Action就会生成一个Job



大纲

Spark Shell

Spark Shell

- Spark交互式运行模式
- 边写代码边执行
  - 快速体验Spark
  - 快速学习Spark
- 常用于测试

Spark Shell

- 下载Spark安装包
  - http://spark.apache.org/downloads.html
- 本地模式启动
  - bin/spark-shell 或者bin/spark-shell --master local
  - 自动创建SparkContext
- 演示

Spark Shell

- On Yarn模式启动

- 修改yarn-site.xml添加物理内存与虚拟内存的比率配置项，任务每使用1MB物理内存，最多可使用虚拟内存量，默认是2.1

```
<property>
        <name>yarn.nodemanager.vmem-pmem-ratio</name>
        <value>10</value>
</property>
```

注意：保持所有节点的配置文件内容一致，重启YARN

- 修改spark安装包conf目录下的spark-env.sh，解压的安装包里没有该文件，需要复制spark-env.sh.template模板

添加hadoop配置文件路径：

export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop

- 启动

bin/spark-shell --master yarn --deploy-mode client

注意：spark-shell只能以client的模式运行，Driver要运行在本地

疑问

- 小象问答官网
  - http://wenda.chinahadoop.cn

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：小象学院
- 新浪微博：小象AI学院