https://www.bilibili.com/video/BV174411X7Pk?from=search&seid=16140933637673040363
(https://www.bilibili.com/video/BV174411X7Pk?from=search&seid=16140933637673040363)

https://www.bilibili.com/video/BV11A411L7CK?p=186
(https://www.bilibili.com/video/BV11A411L7CK?p=186)

In [ ]:
```
1
```

In [ ]:
```
1
```

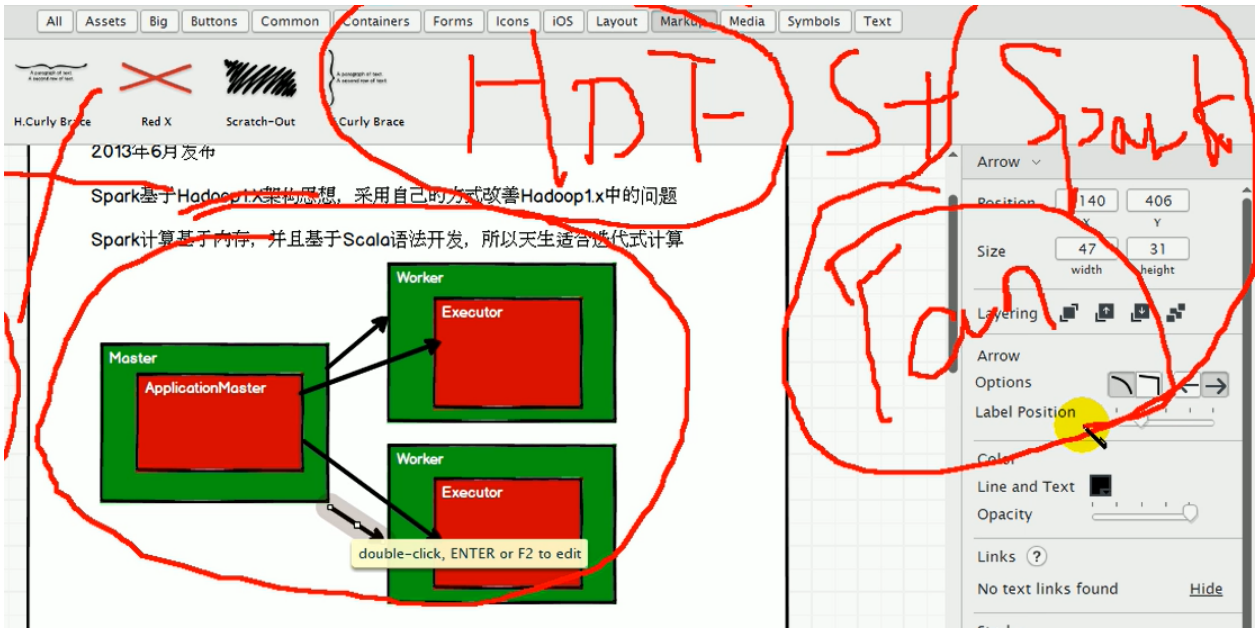
Spark 和 Hadoop 的根本差异是多个作业之间的数据通信问题 ：Spark 多个作业之间数据通信是基于内存，而 Hadoop 是基于磁盘。

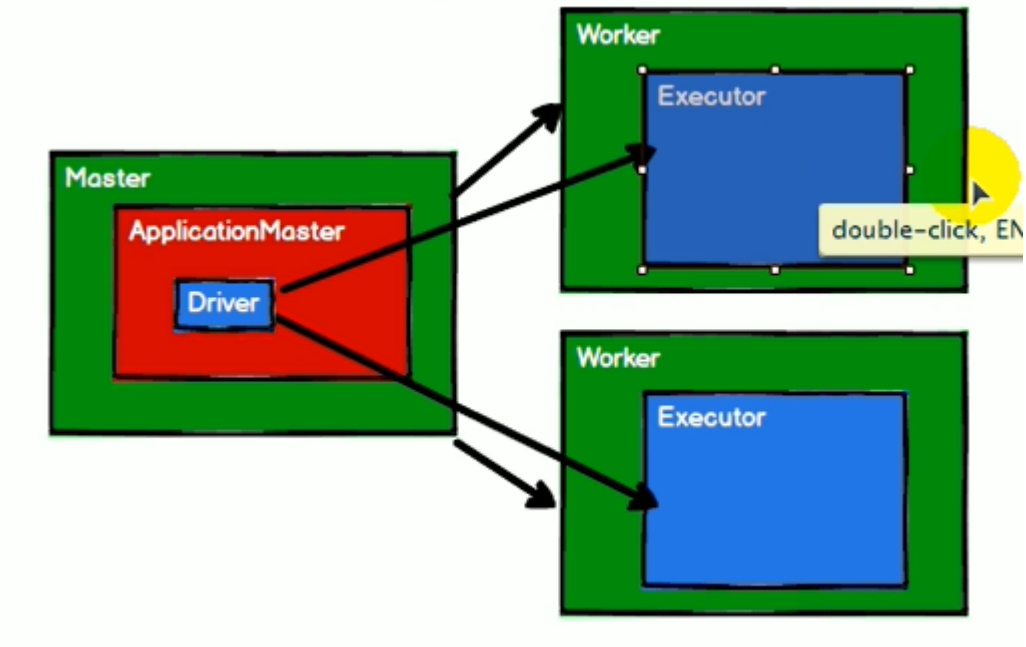
经过上面的比较，我们可以看出在绝大多数的数据计算场景中，Spark 确实会比 MapReduce 更有优势。但是 Spark 是基于内存的，所以在实际的生产环境中，由于内存的限制，可能会由于内存资源不够导致 Job 执行失败，此时，MapReduce 其实是一个更好的选择，所以 Spark 并不能完全替代 MR。

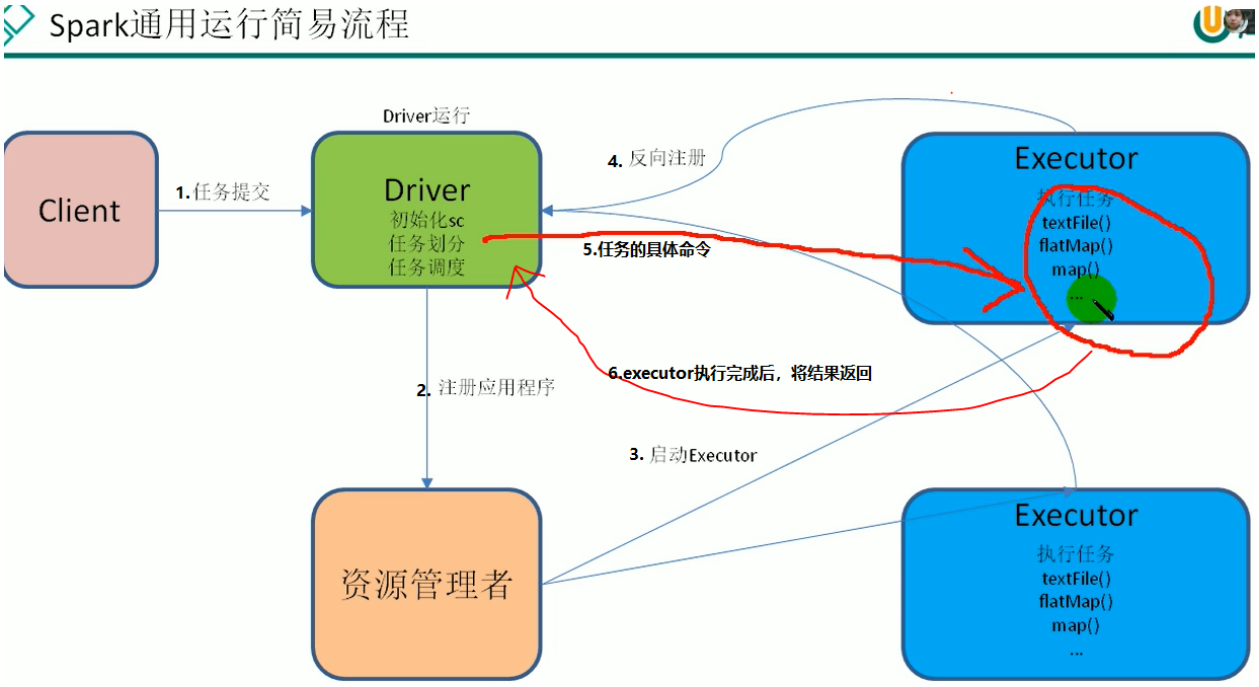
- 存储的弹性：内存与磁盘的自动切换；
- 容错的弹性：数据丢失可以自动恢复；
- 计算的弹性：计算出错重试机制；
- 分片的弹性：可根据需要重新分片。

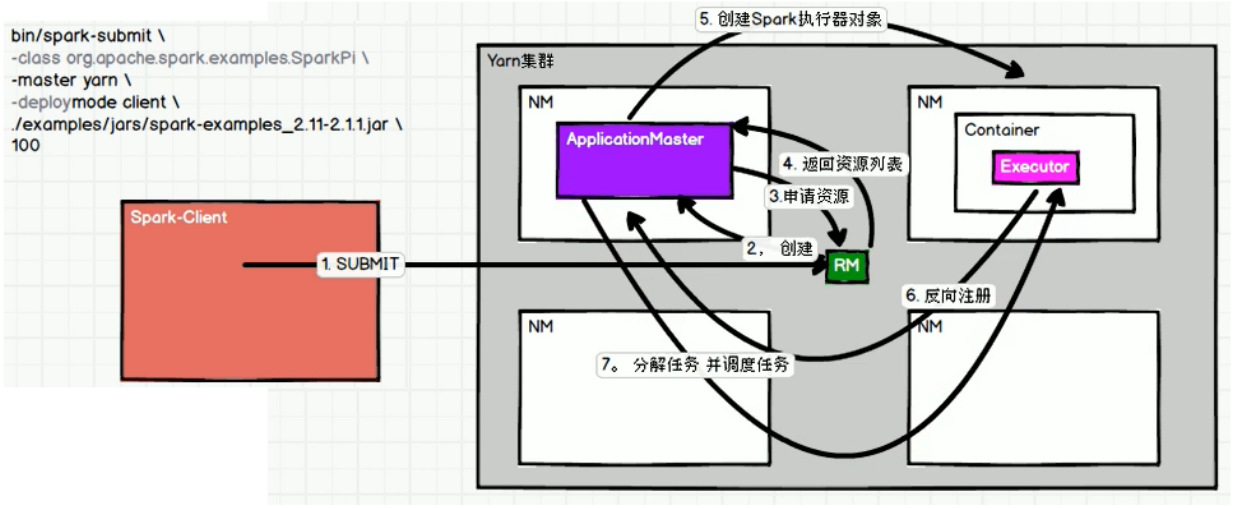
- 分布式：数据存储在大数据集群不同节点上
- 数据集：RDD 封装了计算逻辑，并不保存数据
- 数据抽象：RDD 是一个抽象类，需要子类具体实现
- 不可变：RDD 封装了计算逻辑，是不可以改变的，想要改变，只能产生新的 RDD，在新的 RDD 里面封装计算逻辑
- 可分区、并行计算

In [ ]:
```
1
```

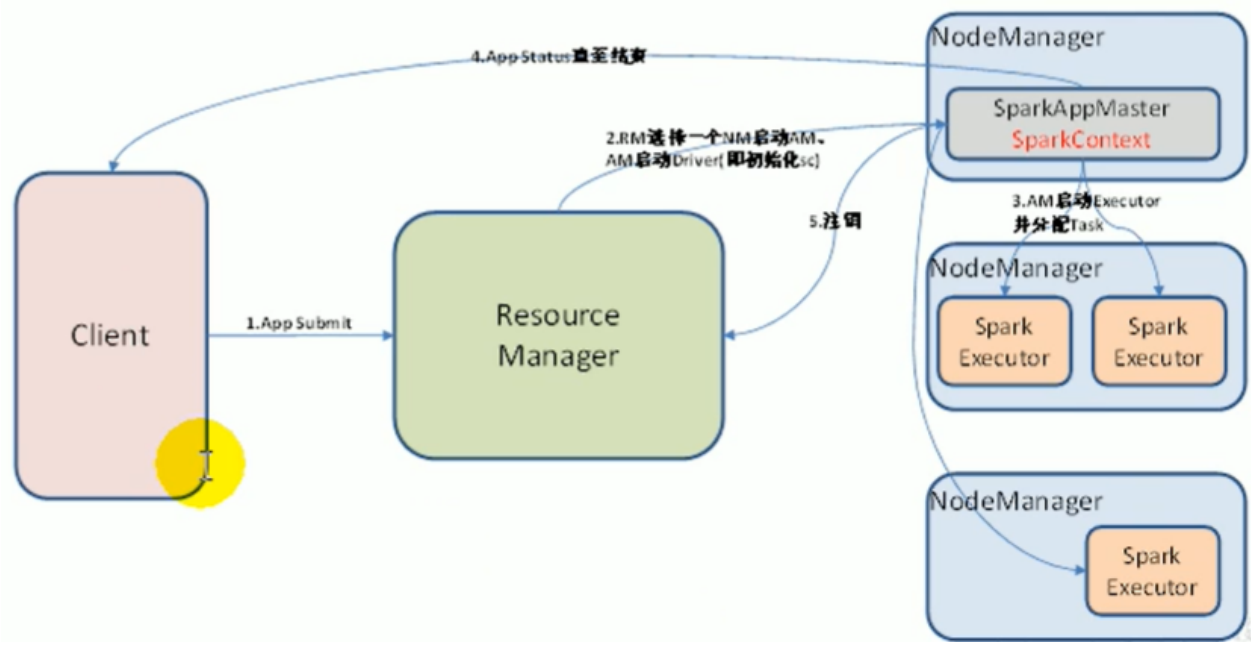

All
Assets
Big
Buttons
Common
Containers
Forms
Icons
iOS
Layout
Media
Symbols
Text
H.Curly Brace
Red X
Scratch-Out
Curly Brace
2013年6月发布
Spark基于Hadoop1.x架构思想，采用自己的方式改善Hadoop1.x中的问题
Spark计算基于内存，并且基于Scala语法开发，所以天生适合迭代式计算
Worker
Executor
Master
ApplicationMaster
Worker
Executor
double-click, ENTER or F2 to edit
Arrow
Position
140
406
Size
47
31
width
height
Layering
Arrow
Options
Label Position
Line and Text
Opacity
Links
No text links found
Hide


Worker
Executor
Master
ApplicationMaster
Driver
Worker
Executor



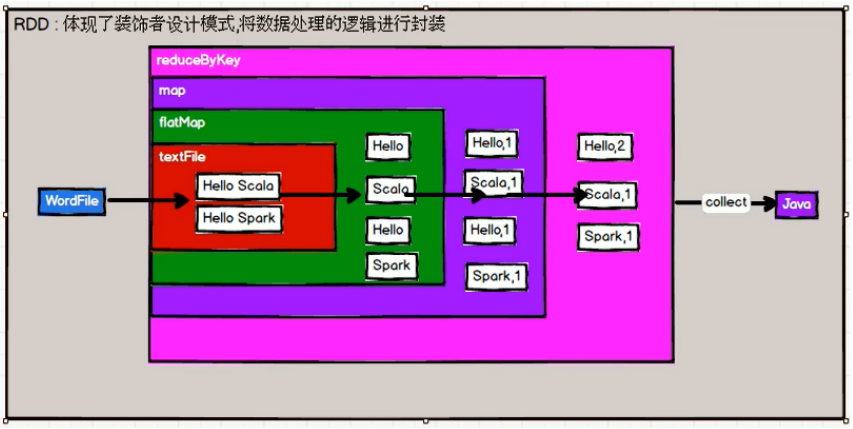


4.返回资源列表--返回可用的NM，applicationmaster所在的NM就不是可用的NM



RDD（Resilient Distributed Dataset）叫做弹性分布式数据集，是 Spark 中最基本的数据抽象。代码中是一个抽象类，它代表一个不可变、可分区、里面的元素可并行计算的集合。

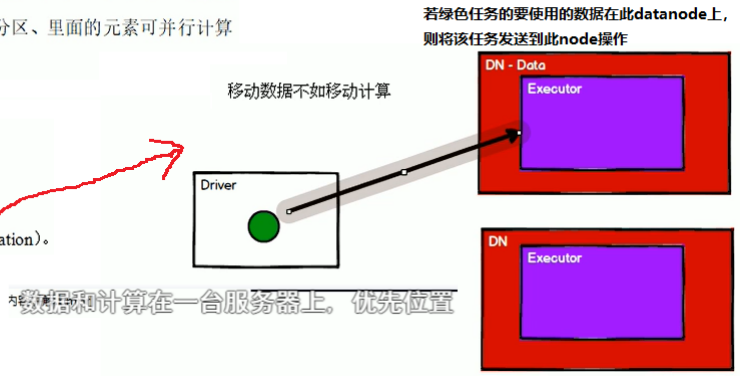
```
sc.textFile('file:///root/tmp/test.txt') \
.flatMap(lambda line: line.split(" ")) \
.map(lambda x: (x, 1)) \
.reduceByKey(lambda a, b: a + b).collect()
```



RDD（Resilient Distributed Dataset）叫做弹性分布式数据集，是 Spark 中最基本的数据（计算）抽象。代码中是一个抽象类，它代表一个不可变、可分区、里面的元素可并行计算的集合。

RDD的属性：

1) 一组分区（Partition），即数据集的基本组成单位;
2) 一个计算每个分区的函数;
3) RDD 之间的依赖关系;
4) 一个 Partitioner，即 RDD 的分片函数;
5) 一个列表，存储存取每个 Partition 的优先位置（preferred location）。



RDD 表示只读的分区的数据集，对 RDD 进行改动，只能通过 RDD 的转换操作，由一个 RDD 得到一个新的 RDD，新的 RDD 包含了从其他 RDD 衍生所必需的信息。RDDs 之间存在依赖，RDD 的执行是按照血缘关系延时计算的。如果血缘关系较长，可以通过持久化 RDD 来切断血缘关系。

算子：

由一个 RDD 转换到另一个 RDD，可以通过丰富的操作算子实现，不再像 MapReduce 那样只能写 map 和 reduce 了

算子：从认知心理学角度，解决问题其实是将 问题的初始状态，通过一系列的操作（Operate）（算子）对问题的状态进行转换，然后达到完成（解决）状态

Spark中的所有的RDD方法都称之为算子，但是分为2大类：转换算子 & 行动算子

转换结构

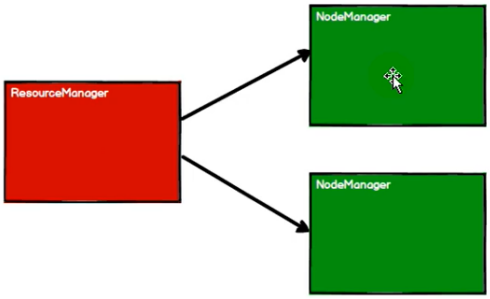
依赖关系：

RDDs 通过操作算子进行转换，转换得到的新 RDD 包含了从其他 RDDs 衍生所必需的信息，RDDs 之间维护着这种血缘关系，也称之为依赖。 依赖包括两种，一种是窄依赖，RDDs 之间分区是一一对应的，另一种是宽依赖，下游 RDD 的每个分区与上游 RDD(也称之为父 RDD)的每个分区都有关，是多对多的关系。

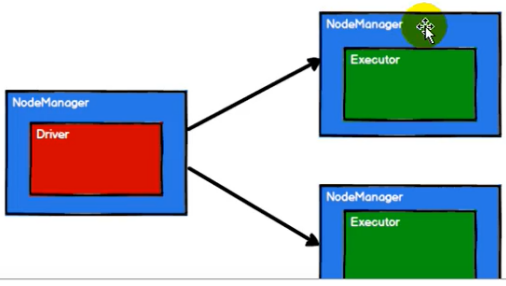
为防止血缘关系中断，需要将血缘关系缓存起来。

In [ ]:
```
1
```

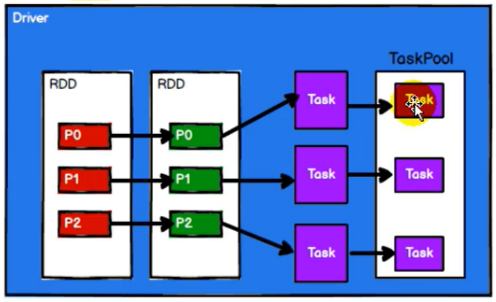
1) 启动 Yarn 集群环境



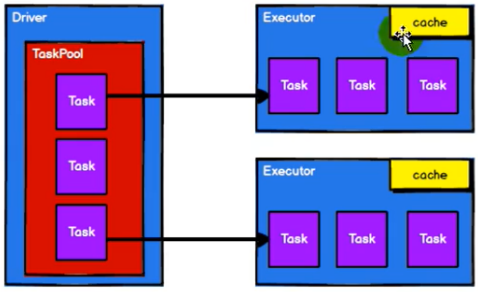
2) Spark 通过申请资源创建调度节点和计算节点



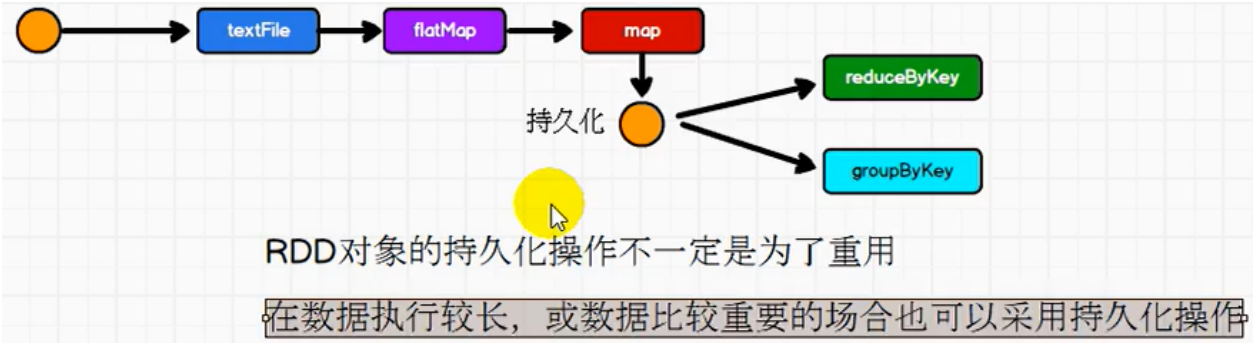
3) Spark 框架根据需求将计算逻辑根据分区划分成不同的任务



4) 调度节点将任务根据计算节点状态发送到对应的计算节点进行计算



In [ ]:
```
1
```



In [ ]:
```
1
```

```
// cache : 将数据临时存储在内存中进行数据重用
// persist : 将数据临时存储在磁盘文件中进行数据重用
//           涉及到磁盘IO，性能较低，但是数据安全
//           如果作业执行完毕，临时保存的数据文件就会丢失
// checkpoint : 将数据长久地保存在磁盘文件中进行数据重用
//           涉及到磁盘IO，性能较低，但是数据安全
//           为了保证数据安全，所以一般情况下，会独立执行作业
//           为了能够提高效率，一般情况下，是需要和cache联合使用
```