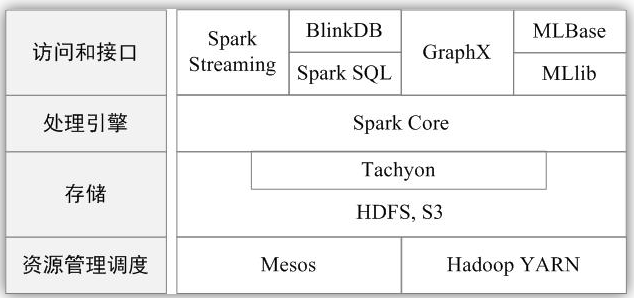
# Spark基础

## [hadoop安装](http://dblab.xmu.edu.cn/blog/install-hadoop-in-centos/)

## [spark安装](http://dblab.xmu.edu.cn/blog/1689-2/)

## 组件



Spark的生态系统主要包含了Spark Core、Spark SQL、Spark Streaming、MLLib和GraphX 等组件，各个组件的具体功能如下：

### Spark Core

Spark Core包含Spark的基本功能，如内存计算、任务调度、部署模式、故障恢复、存储管理等。Spark建立在统一的抽象RDD之上，使其可以以基本一致的方式应对不同的大数据处理场景；通常所说的Apache Spark，就是指Spark Core；

### Spark SQL

Spark SQL允许开发人员直接处理RDD，同时也可查询Hive、HBase等外部数据源。Spark SQL的一个重要特点是其能够统一处理关系表和RDD，使得开发人员可以轻松地使用SQL命令进行查询，并进行更复杂的数据分析；

### Spark Streaming

Spark Streaming支持高吞吐量、可容错处理的实时流数据处理，其核心思路是将流式计算分解成一系列短小的批处理作业。Spark Streaming支持多种数据输入源，如Kafka、Flume和TCP套接字等；

### MLlib（机器学习）

MLlib提供了常用机器学习算法的实现，包括聚类、分类、回归、协同过滤等，降低了机器学习的门槛，开发人员只要具备一定的理论知识就能进行机器学习的工作；

GraphX（图计算）：GraphX是Spark中用于图计算的API，可认为是Pregel在Spark上的重写及优化，Graphx性能良好，拥有丰富的功能和运算符，能在海量数据上自如地运行复杂的图算法。

### GraphX（图计算）

GraphX是Spark中用于图计算的API，可认为是Pregel在Spark上的重写及优化，Graphx性能良好，拥有丰富的功能和运算符，能在海量数据上自如地运行复杂的图算法。

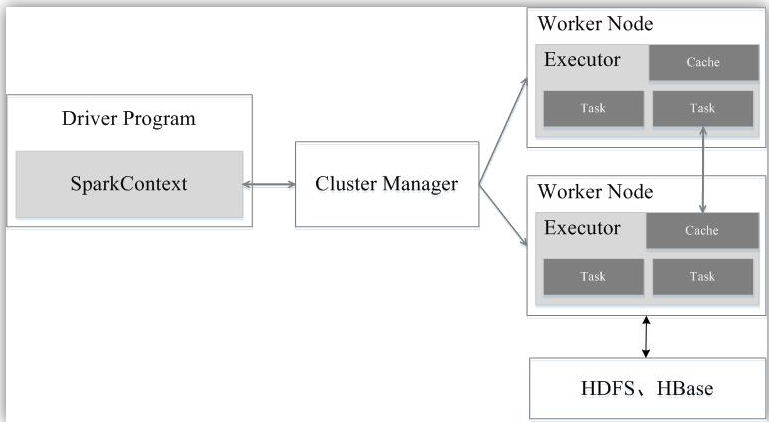
## 运行架构

* RDD：是弹性分布式数据集（Resilient Distributed Dataset）的简称，是分布式内存的一个抽象概念，提供了一种高度受限的共享内存模型；
* DAG：是Directed Acyclic Graph（有向无环图）的简称，反映RDD之间的依赖关系；
* Executor：是运行在工作节点（Worker Node）上的一个进程，负责运行任务，并为应用程序存储数据；
* 应用：用户编写的Spark应用程序；
* 任务：运行在Executor上的工作单元；
* 作业：一个作业包含多个RDD及作用于相应RDD上的各种操作；
* 阶段：是作业的基本调度单位，一个作业会分为多组任务，每组任务被称为“阶段”，或者也被称为“任务集”。

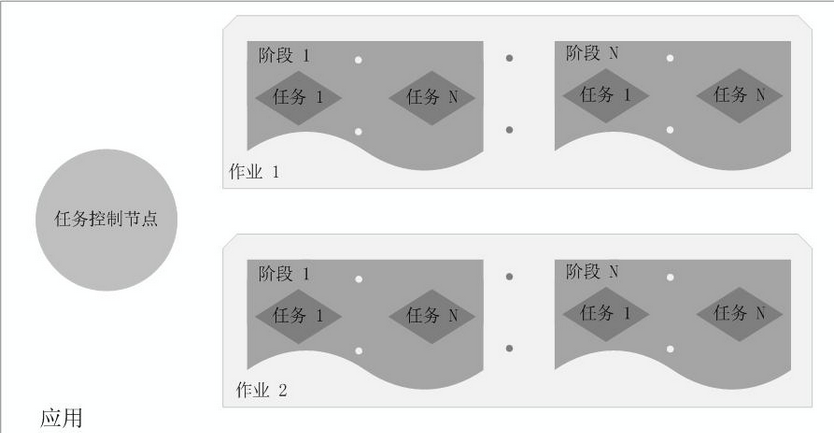
Spark运行架构包括

* 集群资源管理器（Cluster Manager）、
* 运行作业任务的工作节点（Worker Node）、
* 每个应用的任务控制节点（Driver）
* 每个工作节点上负责具体任务的执行进程（Executor）。

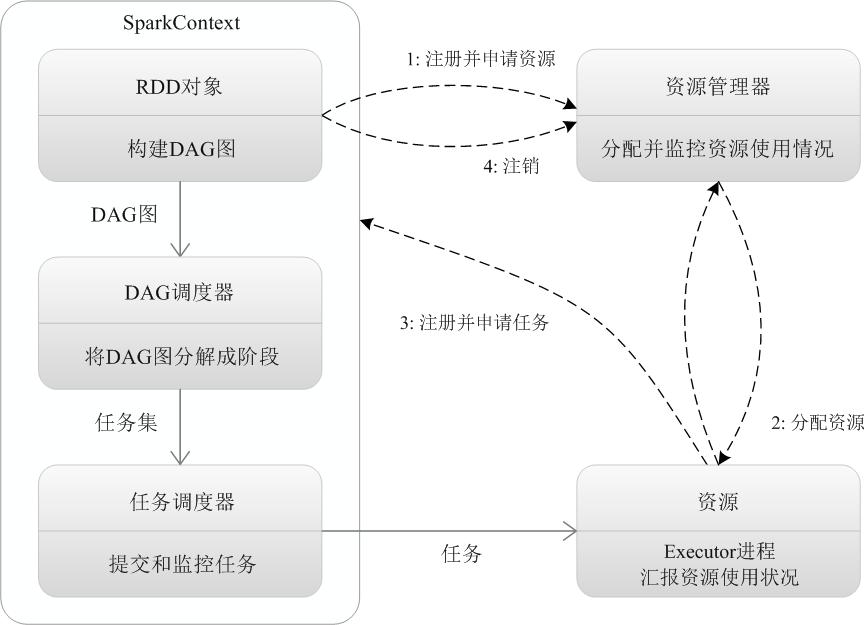
其中，集群资源管理器可以是Spark自带的资源管理器，也可以是YARN或Mesos等资源管理框架。



一个应用（Application）由一个任务控制节点（Driver）和若干个作业（Job）构成，一个作业由多个阶段（Stage）构成，一个阶段由多个任务（Task）组成。当执行一个应用时，任务控制节点会向集群管理器（Cluster Manager）申请资源，启动Executor，并向Executor发送应用程序代码和文件，然后在Executor上执行任务，运行结束后，执行结果会返回给任务控制节点，或者写到HDFS或者其他数据库中。



## Spark运行基本流程



（1）当一个Spark应用被提交时，首先需要为这个应用构建起基本的运行环境，即由任务控制节点（Driver）创建一个SparkContext，由SparkContext负责和资源管理器（Cluster Manager）的通信以及进行资源的申请、任务的分配和监控等。SparkContext会向资源管理器注册并申请运行Executor的资源；  
（2）资源管理器为Executor分配资源，并启动Executor进程，Executor运行情况将随着“心跳”发送到资源管理器上；  
（3）SparkContext根据RDD的依赖关系构建DAG图，DAG图提交给DAG调度器（DAGScheduler）进行解析，将DAG图分解成多个“阶段”（每个阶段都是一个任务集），并且计算出各个阶段之间的依赖关系，然后把一个个“任务集”提交给底层的任务调度器（TaskScheduler）进行处理；Executor向SparkContext申请任务，任务调度器将任务分发给Executor运行，同时，SparkContext将应用程序代码发放给Executor；  
（4）任务在Executor上运行，把执行结果反馈给任务调度器，然后反馈给DAG调度器，运行完毕后写入数据并释放所有资源。

## Spark三种部署方式

包括standalone、Spark on Mesos和Spark on YARN

### standalone模式

与MapReduce1.0框架类似，Spark框架本身也自带了完整的资源调度管理服务，可以独立部署到一个集群中，而不需要依赖其他系统来为其提供资源管理调度服务。在架构的设计上，Spark与MapReduce1.0完全一致，都是由一个Master和若干个Slave构成，并且以槽（slot）作为资源分配单位。不同的是，Spark中的槽不再像MapReduce1.0那样分为Map 槽和Reduce槽，而是只设计了统一的一种槽提供给各种任务来使用。

### Spark on Mesos模式

Mesos是一种资源调度管理框架，可以为运行在它上面的Spark提供服务。Spark on Mesos模式中，Spark程序所需要的各种资源，都由Mesos负责调度。由于Mesos和Spark存在一定的血缘关系，因此，Spark这个框架在进行设计开发的时候，就充分考虑到了对Mesos的充分支持，因此，相对而言，Spark运行在Mesos上，要比运行在YARN上更加灵活、自然。目前，Spark官方推荐采用这种模式，所以，许多公司在实际应用中也采用该模式。

### Spark on YARN模式

Spark可运行于YARN之上，与Hadoop进行统一部署，即“Spark on YARN”，资源管理和调度依赖YARN，分布式存储则依赖HDFS。

