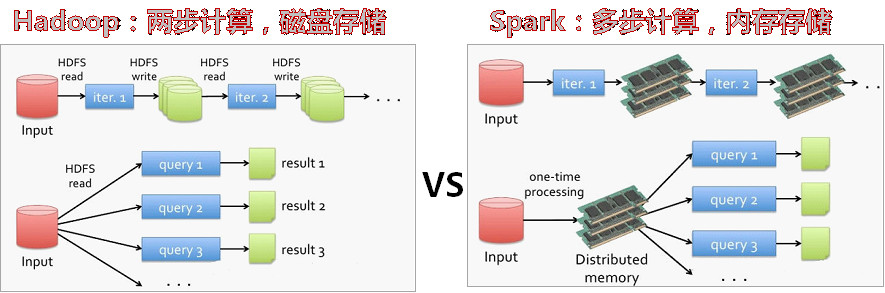
# Spark概述

## 什么是Spark（官网：<http://spark.apache.org>）

Spark是一种快速、通用、可扩展的大数据分析引擎，2009年诞生于加州大学伯克利分校AMPLab，2010年开源，2013年6月成为Apache孵化项目，2014年2月成为Apache顶级项目。目前，Spark生态系统已经发展成为一个包含多个子项目的集合，其中包含SparkSQL、Spark Streaming、GraphX、MLlib等子项目，Spark是基于内存计算的大数据并行计算框架。Spark基于内存计算，提高了在大数据环境下数据处理的实时性，同时保证了高容错性和高可伸缩性，允许用户将Spark部署在大量廉价硬件之上，形成集群。

## 为什么要学Spark

中间结果输出：基于MapReduce的计算引擎通常会将中间结果输出到磁盘上，进行存储和容错。出于任务管道承接的，考虑，当一些查询翻译到MapReduce任务时，往往会产生多个Stage，而这些串联的Stage又依赖于底层文件系统（如HDFS）来存储每一个Stage的输出结果

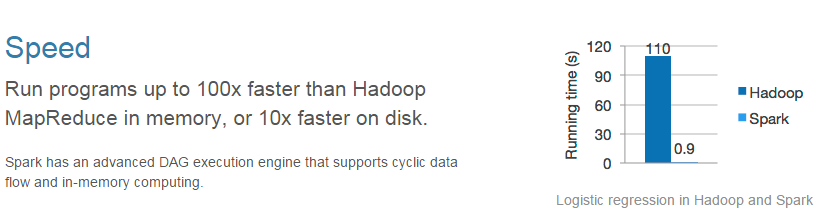


Spark是MapReduce的替代方案，而且兼容HDFS、Hive，可融入Hadoop的生态系统，以弥补MapReduce的不足。

## Spark特点

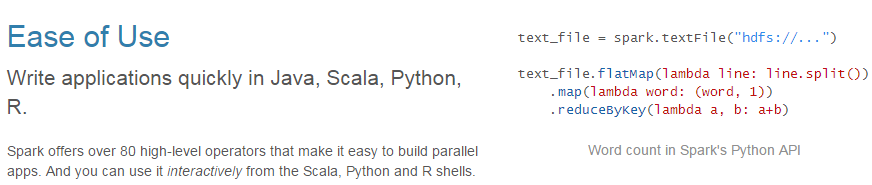
### 快

与Hadoop的MapReduce相比，Spark基于内存的运算要快100倍以上，基于硬盘的运算也要快10倍以上。Spark实现了高效的DAG执行引擎，可以通过基于内存来高效处理数据流。



### 易用

Spark支持Java、Python和Scala的API，还支持超过80种高级算法，使用户可以快速构建不同的应用。而且Spark支持交互式的Python和Scala的shell，可以非常方便地在这些shell中使用Spark集群来验证解决问题的方法。

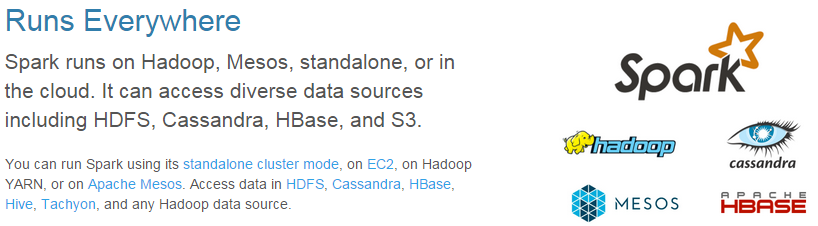


### 通用

Spark提供了统一的解决方案。Spark可以用于批处理、交互式查询（Spark SQL）、实时流处理（Spark Streaming）、机器学习（Spark MLlib）和图计算（GraphX）。这些不同类型的处理都可以在同一个应用中无缝使用。Spark统一的解决方案非常具有吸引力，毕竟任何公司都想用统一的平台去处理遇到的问题，减少开发和维护的人力成本和部署平台的物力成本。

### 兼容性

Spark可以非常方便地与其他的开源产品进行融合。比如，Spark可以使用Hadoop的YARN和Apache Mesos作为它的资源管理和调度器，器，并且可以处理所有Hadoop支持的数据，包括HDFS、HBase和Cassandra等。这对于已经部署Hadoop集群的用户特别重要，因为不需要做任何数据迁移就可以使用Spark的强大处理能力。Spark也可以不依赖于第三方的资源管理和调度器，它实现了Standalone作为其内置的资源管理和调度框架，这样进一步降低了Spark的使用门槛，使得所有人都可以非常容易地部署和使用Spark。此外，Spark还提供了在EC2上部署Standalone的Spark集群的工具。



# Spark集群安装

## 安装

### 机器部署

准备两台以上Linux服务器，安装好JDK

上传spark-安装包到Linux上

解压安装包到指定位置

tar -zxvf spark-2.1.0-bin-hadoop2.6.tgz -C /root/apps

### 配置Spark

进入到Spark安装目录

cd /root/apps/spark-2.4.3-bin-hadoop2.7

进入conf目录并重命名并修改spark-env.sh.template文件

cd conf/

mv spark-env.sh.template spark-env.sh

vi spark-env.sh

在该配置文件中添加如下配置

export JAVA\_HOME=/root/apps/jdk1.8.0\_221

export SPARK\_MASTER\_IP=hdp1

export SPARK\_MASTER\_PORT=7077

#export SPARK\_WORKER\_CORES=4

#export SPARK\_WORKER\_MEMORY=2g

#export SPARK\_WORKER\_INSTANCES=2

export SPARK\_EXECUTOR\_MEMORY=1g

export SPARK\_JAVA\_OPTS=-Dspark.executor.memory=1g

#export HADOOP\_CONF\_DIR=/root/apps/hadoop-2.8.1/etc/hadoop

export HADOOP\_HOME=/root/apps/hadoop-2.8.1

保存退出

重命名并修改slaves.template文件

mv slaves.template slaves

vi slaves

在该文件中添加子节点所在的位置（Worker节点）

Node1

Node2

Node3

node4

保存退出

将配置好的Spark拷贝到其他节点上

scp -r spark-2.4.3-bin-hadoop2.7/ node2:$PWD

Spark集群配置完毕，目前是1个Master，4个Work，在node1上启动Spark集群

/root/apps/ spark-2.4.3-bin-hadoop2.7/sbin/start-all.sh

启动后执行jps命令，主节点上有Master进程，其他子节点上有Work进行，登录Spark管理界面查看集群状态（主节点）：[http://node1:8080/](http://node1.itcast.cn:8080/)

到此为止，Spark集群安装完毕，但是有一个很大的问题，那就是Master节点存在单点故障，要解决此问题，就要借助zookeeper，并且启动至少两个Master节点来实现高可靠，配置方式比较简单：

Spark集群规划：node1，node2是Master；node3，node4，node5是Worker

安装配置zk集群，并启动zk集群

停止spark所有服务，修改配置文件spark-env.sh，在该配置文件中删掉SPARK\_MASTER\_IP并添加如下配置

export SPARK\_DAEMON\_JAVA\_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER -Dspark.deploy.zookeeper.url=zk1,zk2,zk3 -Dspark.deploy.zookeeper.dir=/spark"

1.在node1节点上修改slaves配置文件内容指定worker节点

2.在node1上执行**sbin/start-all.sh**脚本，然后在node2上执行**sbin/start-master.sh**启动第二个Master

配置hdfs 时 需要把 core-site

# 执行Spark程序

## 执行第一个spark程序

/root/apps/spark-2.4.3-bin-hadoop2.7/bin/spark-submit --class org.apache.spark.examples.SparkPi --master spark://hdp1:7077 --executor-memory 1G --total-executor-cores 2 /root/apps/spark-2.4.3-bin-hadoop2.7/examples/jars/spark-examples\_2.11-2.4.3.jar 100

/root/apps/spark-2.4.3-bin-hadoop2.7/bin/spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode cluster --executor-memory 1G --total-executor-cores 2 /root/apps/spark-2.4.3-bin-hadoop2.7/examples/jars/spark-examples\_2.11-2.4.3.jar 100

该算法是利用蒙特·卡罗算法求PI

## 启动Spark Shell

spark-shell是Spark自带的交互式Shell程序，方便用户进行交互式编程，用户可以在该命令行下用scala编写spark程序。

### 启动spark shell

/root/apps/spark-2.4.3-bin-hadoop2.7/bin/spark-shell –master spark://hdp1:7077 --executor-memory 2g --total-executor-cores 2

参数说明：

--master spark://node1:7077 指定Master的地址

--executor-memory 2g 指定每个worker可用内存为2G

--total-executor-cores 2 指定整个集群使用的cup核数为2个

注意：

如果启动spark shell时没有指定master地址，但是也可以正常启动spark shell和执行spark shell中的程序，其实是启动了spark的local模式，该模式仅在本机启动一个进程，没有与集群建立联系。

Spark Shell中已经默认将SparkContext类初始化为对象sc。用户代码如果需要用到，则直接应用sc即可

### 在spark shell中编写WordCount程序

1. 首先启动hdfs
2. 向hdfs上传一个文件到hdfs://node1:9000/words.txt
3. 在spark shell中用scala语言编写spark程序

sc.textFile("hdfs://node1:9000/wordcount/input/words.txt").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).saveAsTextFile("hdfs://node1:9000/words/out")

1. 使用hdfs命令查看结果

hdfs dfs -ls hdfs://node1:9000/out/p\*

说明：

sc是SparkContext对象，该对象时提交spark程序的入口

textFile(hdfs://node1:9000/words.txt)是hdfs中读取数据

flatMap(\_.split(" "))先map在压平

map((\_,1))将单词和1构成元组

reduceByKey(\_+\_)按照key进行reduce，并将value累加

saveAsTextFile("hdfs://node1:9000/out")将结果写入到hdfs中

1. 首先启动hdfs和Spark集群

启动hdfs

/root/apps/hadoop-2.6.5/sbin/start-dfs.sh

启动spark

/root/apps/spark-2.4.3-bin-hadoop2.7/sbin/start-all.sh

1. 使用spark-submit命令提交Spark应用（注意参数的顺序）

/root/apps/spark-2.4.3-bin-hadoop2.7/bin/spark-submit \

--class cn.itcast.spark.WordCount \

--master spark://node1:7077 \

--executor-memory 2G \

--total-executor-cores 4 \

/root/spark-mvn-1.0-SNAPSHOT.jar \

hdfs://node1:9000/words.txt \

hdfs://node1:9000/out

查看程序执行结果

hdfs dfs -cat hdfs://node1:9000/out/part-00000