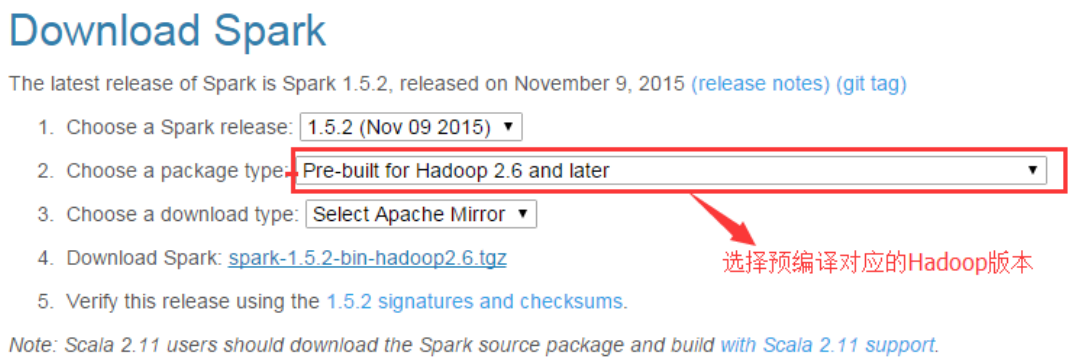
一、 Spark 集群安装

1. 安装

1.1. 机器部署

准备两台以上 Linux 服务器，并安装好 JDK。

1.2. 下载 Spark 安装包



http://www.apache.org/dyn/closer.lua/spark/spark-1.5.2/spark-1.5.2-bin-hadoop2.6.tgz

上传、解压安装包：

1、上传 spark-1.5.2-bin-hadoop2.6.tgz 安装包到 Linux 服务器

2、解压安装包到指定位置：`tar -zxvf spark-1.5.2-bin-hadoop2.6.tgz -C /usr/local`

1.3. 修改 Spark 配置文件

1、进入 Spark 安装目录：`cd /usr/local/spark-1.5.2-bin-hadoop2.6`

2、进入 conf 目录：`cd conf/`

3、重命名并修改 spark-env.sh.template 文件

```
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=/usr/java/jdk1.7.0_45 ## :r! echo /usr/java/jdk...用于自动显示
export SPARK_MASTER_IP=node1.itcast.cn
export SPARK_MASTER_PORT=7077
```

保存退出

4、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（Worker 节点）

node2.itcast.cn

node3.itcast.cn

node4.itcast.cn

保存退出

5、将配置好的 Spark 拷贝到其他节点

scp -r spark-1.5.2-bin-hadoop2.6/ node2.itcast.cn:/usr/local/

scp -r spark-1.5.2-bin-hadoop2.6/ node3.itcast.cn:/usr/local/

scp -r spark-1.5.2-bin-hadoop2.6/ node4.itcast.cn:/usr/local/

6、在 node1.itcast.cn 上启动 Spark 集群

/usr/local/spark-1.5.2-bin-hadoop2.6/sbin/start-all.sh

7、启动后执行 jps 命令，主节点上有 Master 进程，其他子节点上有 Worker 进程

8、登录 Spark 管理界面查看集群状态：http://node1.itcast.cn:8080

Spark 集群安装完毕。但是有一个很大的问题：Master 节点存在单点故障。解决此问题需要借助 zookeeper，并且至少启动两个 Master 节点来实现高可用。

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

（1）安装并启动 zk 集群；

（2）停止 spark 服务，修改配置文件 spark-env.sh，在该配置文件中删除 SPARK_MASTER_IP、SPARK_MASTER_PORT 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" ## Spark 向 zk 写数据的存放目录；

（3）修改 slaves 文件内容，指定 worker 节点；

（4）在 node1 上执行 **sbin/start-all.sh** 脚本，在 node2 上执行 **sbin/start-master.sh** 启动第二个 Master。

补充：搭建 Spark 伪分布式

1、 安装 jdk

2、 解压 spark 压缩包：tar -zxvf spark-1.6.2-bin-hadoop2.6.tgz

3、 bin 目录下启动 spark-shell：./spark-shell

（1） 启动 spark 单机版应用程序来模拟 spark 集群的运行；

（2） spark-shell 是 spark 集群的客户端；若是单机形式，则模拟应用程序在本地执行；

（3） spark-shell 启动时创建 SparkContext 对象实例。

```
[root@mini1 bin]# ./spark-shell
/root/apps/spark-1.6.2-bin-hadoop2.6/conf/spark-env.sh: line 73: unexpected EOF while looking for matching `"'
/root/apps/spark-1.6.2-bin-hadoop2.6/conf/spark-env.sh: line 74: syntax error: unexpected end of file
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.lib.MutableMetricsFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Using Spark's repl log4j profile: org/apache/spark/log4j-defaults-repl.properties
To adjust logging level use sc.setLogLevel("INFO")
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.2
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_131)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context available as sc.
19/06/27 18:07:56 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)
19/06/27 18:07:57 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)
19/06/27 18:08:06 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0
19/06/27 18:08:06 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
19/06/27 18:08:10 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)
19/06/27 18:08:11 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependencies)
19/06/27 18:08:18 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not enabled so recording the schema version 1.2.0
19/06/27 18:08:18 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
SQL context available as sqlContext.

scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@76ffc17c

scala> sc.textFile("/root/apps/testdata/somewords.txt")
res1: org.apache.spark.rdd.RDD[String] = /root/apps/testdata/somewords.txt MapPartitionsRDD[1] at textFile at <console>:28

scala> sc.textFile("/root/apps/testdata/somewords.txt").collect
res2: Array[String] = Array(hello tom hello kitty, hello wangqikang hello chicago, hello world hello tom kitty, hello)

scala> sc.textFile("/root/apps/testdata/somewords.txt").flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).collect
res3: Array[(String, Int)] = Array((chicago,1), (tom,2), (hello,7), (wangqikang,1), (kitty,2), (world,1))

scala> sc.textFile("/root/apps/testdata/somewords.txt").flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).sortBy(_.)
<console>:1: error: identifier expected but ')' found.
       sc.textFile("/root/apps/testdata/somewords.txt").flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).sortBy(_.)
                                                                                                            ^

scala> sc.textFile("/root/apps/testdata/somewords.txt").flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).sortBy(_._2,false).collect
res4: Array[(String, Int)] = Array((hello,7), (tom,2), (kitty,2), (chicago,1), (wangqikang,1), (world,1))

scala> sc.textFile("/root/apps/testdata/somewords.txt").flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).sortBy(_._2,false).saveAsTextFile("/root/apps/testdata/output")
```

注：

（1） Spark 默认使用 HDFS 的接口读写数据（textFile()、saveAsTextFile()）。即使用 InputFormat 读取 key-value 数据，textFile()方法做了处理只保留 value；使用 OutputFormat 写数据。

（2） Spark 中的算子包括两种：Transformation 和 Action。Transformation 延迟加载，触发 Action 时执行。

（3） Scala 原生的集合方法操作单机版数据；Spark 提供的 RDD 上的方法并行计算，操作多台机器上的数据集。

（4） RDD 是分布式集合，是 Spark 中最基本的抽象。

（5） reduceByKey()先在一个分区上聚合，再全局聚合。（类似于 combiner 功能）

（6） 为什么 Spark 的 Transformation 算子延迟加载？

每触发 Transformation 执行一次任务，则需不断地与集群进行交互，所以设计成触发 Action 时提交任务执行。