spark封神之路(2)-spark运行模式

原创

置顶 白眼黑刺猬 于 2021-06-08 13:18:18 发布 318 收藏
分类专栏： Spark系列专栏 文章标签： spark hadoop hdfs flink kafka
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/qq_37933018/article/details/117698697
版权


Spark系列专栏 专栏收录该内容
16 篇文章 7 订阅
订阅专栏

本专栏系列视频教程

2 spark运行模式入门

1. 官网地址 http://spark.apache.org/

2. 文档查看地址 https://spark.apache.org/docs/2.1.1/

3. 下载地址 https://archive.apache.org/dist/spark/

2.1 idea编程开发

创建maven项目，添加依赖

```
<properties>

   <maven.compiler.source>1.8</maven.compiler.source>

   <maven.compiler.target>1.8</maven.compiler.target>

   <spark.version>3.0.1</spark.version>

   <encoding>UTF-8</encoding>

</properties>



<dependencies>

   <!-- 导入spark的依赖，core指的是RDD编程API -->

   <dependency>

       <groupId>org.apache.spark</groupId>

       <artifactId>spark-core_2.12</artifactId>

       <version>${spark.version}</version>

   </dependency>

</dependencies>

<build>

       <pluginManagement>

           <plugins>

               <!-- 编译scala的插件 -->

               <plugin>

                   <groupId>net.alchim31.maven</groupId>

                   <artifactId>scala-maven-plugin</artifactId>

                   <version>3.2.2</version>

               </plugin>

               <!-- 编译java的插件 -->

               <plugin>

                   <groupId>org.apache.maven.plugins</groupId>

                   <artifactId>maven-compiler-plugin</artifactId>

                   <version>3.5.1</version>

               </plugin>

           </plugins>
```

```
    </pluginManagement>
    <plugins>
        <plugin>
            <groupId>net.alchim31.maven</groupId>
            <artifactId>scala-maven-plugin</artifactId>
            <executions>
                <execution>
                    <id>scala-compile-first</id>
                    <phase>process-resources</phase>
                    <goals>
                        <goal>add-source</goal>
                        <goal>compile</goal>
                    </goals>
                </execution>
                <execution>
                    <id>scala-test-compile</id>
                    <phase>process-test-resources</phase>
                    <goals>
                        <goal>testCompile</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>

        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <executions>
                <execution>
                    <phase>compile</phase>
                    <goals>
                        <goal>compile</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>

        <!-- 打jar插件 -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>2.4.3</version>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
```

```
                    </goals>
                    <configuration>
                        <filters>
                            <filter>
                                <artifact>*:*</artifact>
                                <excludes>
                                    <exclude>META-INF/*.SF</exclude>
                                    <exclude>META-INF/*.DSA</exclude>
                                    <exclude>META-INF/*.RSA</exclude>
                                </excludes>
                            </filter>
                        </filters>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

编写spark代码

```
 def main(args: Array[String]): Unit = {

   // 1 构建spark的运行环境 SparkContext

   val conf = new SparkConf()

   conf.setAppName("wc")  // 程序名

    .setMaster("local")   // 运行模式 本地

   val sc = new SparkContext(conf)

   // 编程wordcount 和编写scala程序一样

   // 1 读取数据 2获取行 3 单词 4 单词1 5 聚合 排序

   //在spark中 RDD是编程的基本

   /**

    * 关注点 编程逻辑

    * 思考 : 数据的处理应该是多任务并行执行的

    */

   val res: RDD[(String, Int)] = sc.textFile("d://word.txt")

    .flatMap(_.split("\\s+"))

    .map((_, 1))

    .groupBy(_._1)

    .map(tp => (tp._1, tp._2.size))

    .sortBy(-_._2)

   res.saveAsTextFile("data/wc/")

   //释放环境

   sc.stop()

}

object Demo2 {

 def main(args: Array[String]): Unit = {

   val conf = new SparkConf()

   conf.setAppName("wc").setMaster("local")

   val sc = new SparkContext(conf)

   val rdd1: RDD[String] = sc.textFile("d://word.txt")

   val words: RDD[String] = rdd1.flatMap(_.split("\\s+"))

   // spark相对于scala的基础语法提供了更加丰富的编程实现

   words.map((_, 1)).reduceByKey(_ + _).foreach(println)

   sc.stop()

}

}
```

2.2 local模式

Local 模式就是指的只在一台计算机上来运行 Spark.

通常用于测试的目的来使用 Loc al 模式, 实际的生产环境中不会使用 Local 模式.

2.2.1 上传到linux机器 解压

tar -xxvf

```
drwxr-xr-x. 10 1000 ck          161 Feb 22 04:13 hadoop-3.1.1
drwxr-xr-x.  7 root root        182 Mar  6 16:56 hbase-2.2.5
drwxr-xr-x. 10 root root        184 Mar 11 22:39 hive-3.1.2
drwxr-xr-x.  8   10  143        255 Jul 12  2017 jdk1.8.0_141
-rw-r--r--.  1 root root 185511981 Apr 22 05:05 jdk1.8.0_141.tar.gz
drwxr-xr-x.  7 root root        101 Apr 13 03:42 kafka_2.11-2.0.0
drwxr-xr-x. 15 1000 1000        235 Apr 23 01:53 spark-3.0.0-bin-hadoop3.2
drwxr-xr-x. 11 1000 1000       4096 Mar  4 02:05 zookeeper-3.4.6
```

```
drwxr-xr-x. 2 1000 1000  4096 Jun  6  2020 bin      客户端
drwxr-xr-x. 2 1000 1000   196 Jun  6  2020 conf     配置
drwxr-xr-x. 5 1000 1000    50 Jun  6  2020 data     样例数据
drwxr-xr-x. 4 1000 1000    29 Jun  6  2020 examples 例子
drwxr-xr-x. 2 1000 1000 12288 Jun  6  2020 jars
drwxr-xr-x. 4 1000 1000    38 Jun  6  2020 kubernetes
-rw-r--r--. 1 1000 1000 23312 Jun  6  2020 LICENSE
drwxr-xr-x. 2 1000 1000  4096 Jun  6  2020 licenses
-rw-r--r--. 1 1000 1000 57677 Jun  6  2020 NOTICE
drwxr-xr-x. 7 1000 1000   275 Jun  6  2020 python
drwxr-xr-x. 3 1000 1000    17 Jun  6  2020 R
-rw-r--r--. 1 1000 1000  4488 Jun  6  2020 README.md
-rw-r--r--. 1 1000 1000   183 Jun  6  2020 RELEASE
drwxr-xr-x. 2 1000 1000  4096 Jun  6  2020 sbin     启动角色启停
drwxr-xr-x. 2 1000 1000    42 Jun  6  2020 yarn
```

2.2.2 交互客户端spark-shell

bin/spark-shell

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

Spark context Web UI available at http://doit01:4040

Spark context available as 'sc' (master = local[*], app id = local-1620370084250).

Spark session available as 'spark'.

Welcome to

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.0.0
      /_/
```

scala> val name="doit"

name: String = doit

2.2.3 编程

在这个客户端中,自动创建了spark的运行环境, 所以我们可以直接使用客户端提供的sc对象进行spark编程

这里加载的数据默认是本地路径下的文件

sc.textFile("/word.txt").flatMap(_.split("\\s+")).map((_,1)).groupBy(_._1).map(tp=>(tp._1,tp._2.size)).sortBy(- _._2).saveAsTextFile("/spark/wc/")

提示： 在交互式窗口中可以使用tab键补全代码提示

2.2.4 提交自己的代码

- 编写程序

```
/**

* Author:   Hang.Z

* Date:     21/06/01

* Description:

* 使用spark编写wordcount

*   读取本地的数据

*   将结果保存在本地

*   SparkContext 核心环境对象 ,进行编程

*com._51doit.spark.day01.Demo1

*/

object Demo1 {

 def main(args: Array[String]): Unit = {

   // 1 构建spark的运行环境 SparkContext

   val conf = new SparkConf()

   conf.setAppName("wc")  // 程序名

   // 运行模式 本地

   val sc = new SparkContext(conf)

   // 编程wordcount 和编写scala程序一样

   // 1 读取数据 2获取行 3 单词 4 单词1 5 聚合 排序

   //在spark中 RDD是编程的基本

   /**

    * 关注点 编程逻辑

    * 思考 : 数据的处理应该是多任务并行执行的

    */

   val res: RDD[(String, Int)] = sc.textFile("/word.txt")

    .flatMap(_.split("\\s+"))

    .map((_, 1))

    .groupBy(_._1)

    .map(tp => (tp._1, tp._2.size))

    .sortBy(-_._2)

   res.saveAsTextFile("/spark/wc2/")

   //释放环境

   sc.stop()

}


}
```

打包 上传到linux机器

bin/spark-submit --master local[*] --class com._51doit.spark.day01.Demo1 /wc.jar

• 使用spark-submit来发布应用程序.

```
./bin/spark-submit \

--class <main-class> \

--master <master-url> \

--deploy-mode <deploy-mode> \

--conf <key>=<value> \

... # other options

<application-jar> \
```

[application-arguments]

- – --master 指定 master 的地址，默认为local. 表示在本机运行.
- – --class 你的应用的启动类 (如 org.apache.spark.examples.SparkPi)

• --deploy-mode 是否发布你的驱动到 worker节点(cluster 模式) 或者作为一个本地客户端 (client 模式) (default: client)

• --conf: 任意的 Spark 配置属性， 格式key=value. 如果值包含空格，可以加引号"key=value"

• application-jar: 打包好的应用 jar,包含依赖. 这个 URL 在集群中全局可见。 比如hdfs:// 共享存储系统， 如果是 file:// path， 那么所有的节点的path都包含同样的jar

• application-arguments: 传给main()方法的参数

• --executor-memory 1G 指定每个executor可用内存为1G

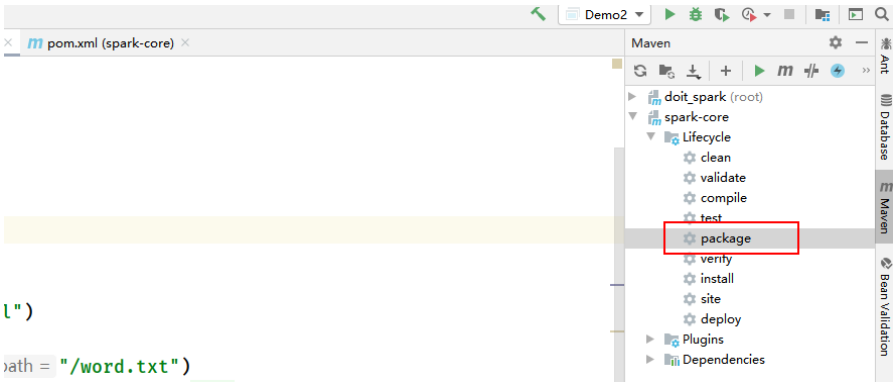
• --total-executor-cores 6 指定所有executor使用的cpu核数为6个

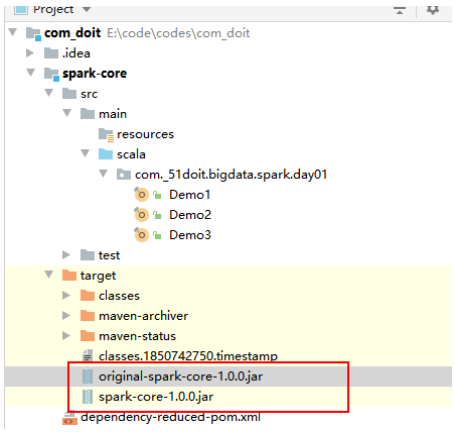
• --executor-cores 表示每个executor使用的 cpu 的核数

• 关于 Master URL 的说明

Master URL	Meaning
local	Run Spark locally with one worker thread (i.e. no parallelism at all).
local[K]	Run Spark locally with K worker threads (ideally, set this to the number of cores on your machine).
local[*]	Run Spark locally with as many worker threads as logical cores on your machine.
spark://HOST:PORT	Connect to the given Spark standalone cluster master. The port must be whichever one your master is configured to use, which is 7077 by default.
mesos://HOST:PORT	Connect to the given Mesos cluster. The port must be whichever one your is configured to use, which is 5050 by default. Or, for a Mesos cluster using ZooKeeper, use mesos://zk://.... To submit with --deploy-mode cluster, the HOST:PORT should be configured to connect to the MesosClusterDispatcher.
yarn	Connect to a YARNcluster in client or cluster mode depending on the value of --deploy-mode. The cluster location will be found based on the HADOOP_CONF_DIR or YARN_CONF_DIR variable.

打包





上传到linux

```
-rw-r--r--.   1 root root 82001704 May  7 03:10 spark-core-1.0.0.jar
drwxr-xr-x.   2 root root        6 Apr 11  2018 srv
```

执行命令

/opt/apps/spark-3.0.0-bin-hadoop3.2/bin/spark-submit --class com._51doit.bigdata.spark.day01.Demo3 --master local[*] /spark-core-1.0.0.jar

21/05/07 03:13:07 INFO Executor: Running task 0.0 in stage 1.0 (TID 1)

21/05/07 03:13:07 INFO ShuffleBlockFetcherIterator: Getting 1 (80.0 B) non-empty blocks including 1 (80.0 B) local and 0 (0.0 B) host-local and 0 (0.0 B) remote blocks

21/05/07 03:13:07 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 10 ms

(spark,1)

(tom,1)

(hello,3)

(jerry,1)

21/05/07 03:13:07 INFO Executor: Finished task 0.0 in stage 1.0 (TID 1). 1267 bytes result sent to driver

21/05/07 03:13:07 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 72 ms on linux01 (executor driver) (1/1)

2.2.5 执行spark样例程序

/opt/apps/spark-3.0.0-bin-hadoop3.2/bin/spark-submit --class org.apache.spark.examples.SparkPi --master local[*] ./spark-examples_2.12-3.0.0.jar 1000

21/05/07 03:19:11 INFO TaskSchedulerImpl: Killing all running tasks in stage 0: Stage finished

21/05/07 03:19:11 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:38, took 9.969789 s

Pi is roughly 3.1416970714169707

21/05/07 03:19:11 INFO SparkUI: Stopped Spark web UI at http://linux01:4040

21/05/07 03:19:11 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!

2.3 standalone模式(集群)自带

spark自带的集群运行模式,采用主从架构工作! 请联想HDFS工作原理,请联想YARN工作原理!.对于大多数情况 Standalone 模式就足够了，如果企业已经有 Yarn 或者 Mesos 环境，也是很方便部署的。

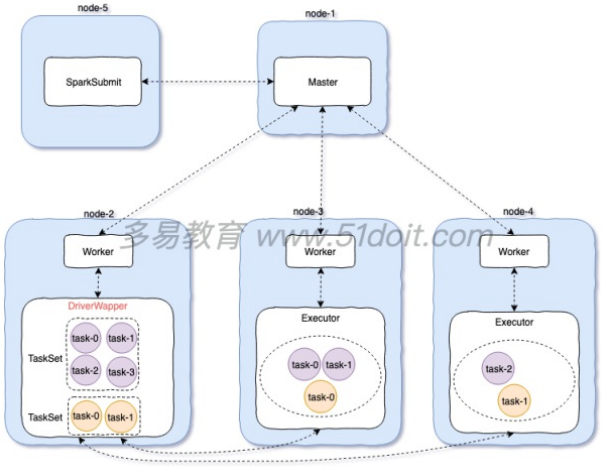
典型的Mater/slave模式，不过也能看出Master是有单点故障的；Spark支持ZooKeeper来实现 HA

- Master节点在Spark中所承载的作用是分配Application到Worker节点，维护Worker节点，Driver，Application的状态。在Spark中，Master本身也提供了基于硬盘的单节点的可用性，也就是可以直接通过重启Master，Master通过读取硬盘里保存的状态，进行单节点的恢复。
- Worker 计算资源的实际贡献者，他要向Master汇报自身拥有多少cpu core和memory, 在master的指示下负责启动executor，executor 是执行真正计算的苦力，由master来决定该进程拥有的core和memory数值，Master 掌管整个cluster的资源，主要是指cpu core和memory，但Master自身并不拥有这些资源，而Driver 是资源的实际占用者，Driver会提交一到多个job，每个job在拆分成多个task之后，会分发到各个executor真正的执行。

集群部署之前

1. 集群的免密
2. 域名映射
3. 学习阶段 防火墙关闭
4. java环境
5. scala环境

spark	doit01机器	doit02机器	doit03机器
角色	master		
角色	worker	worker	worker



Ø 集群中的JAVA环境和SCALA环境

Ø 集群中的域名映射 , 每台机器配置集群中每个机器的域名映射

Ø 集群中的免密配置 , 各个机器免密访问

Ø 集群中的防火墙 , 学习阶段关闭机器中的防火墙

上述环境在前面的学习阶段统统已经搞定!不在过多赘述!

2.3.1 上传,解压

```
drwxr-xr-x. 10 1000 ck          161 Feb 22 04:13 hadoop-3.1.1
drwxr-xr-x.  7 root root        182 Mar  6 16:56 hbase-2.2.5
drwxr-xr-x. 10 root root        184 Mar 11 22:39 hive-3.1.2
drwxr-xr-x.  8   10  143        255 Jul 12  2017 jdk1.8.0_141
-rw-r--r--.  1 root root 185511981 Apr 22 05:05 jdk1.8.0_141.tar.gz
drwxr-xr-x.  7 root root        101 Apr 13 03:42 kafka_2.11-2.0.0
drwxr-xr-x. 15 1000 1000        235 Apr 23 01:53 spark-3.0.0-bin-hadoop3.2
drwxr-xr-x. 11 1000 1000       4096 Mar  4 02:05 zookeeper-3.4.6
```

2.3.2 配置

配置 spark-env.sh文件 ,添加如下信息

export HADOOP_CONF_DIR=/opt/apps/hadoop-3.1.1/etc/hadoop

export SPARK_MASTER_HOST=doit01

export JAVA_HOME=/opt/apps/jdk1.8.0_141

配置slaves文件 添加如下信息 ：配置work角色的节点

A Spark Worker will be started on each of the machines listed below.

linux01

linux02

linux03

2.3.3 分发

将配置好的安装包 ,发送到集群节点上

for i in 2 3 do scp -r spark-3.0.0 doit0$i:$PWD; done

在sbin目录中是集群的启动有关的命令 , 有几个命令

start-master.sh

stop-master.sh

start-slave.sh start-slaves.sh

stop-slave.sh stop-slaves.sh

start-all.sh 一键启动集群

stop-all.sh

为了避免命令和hadoop的命令冲突

[root@doit01 sbin]# mv start-all.sh start-spark.sh [root@doit01 sbin]# mv stop-all.sh stop-spark.sh

2.3.4 启动

启动脚本在SPARK_HOME的sbin目录下 ,我们在学习hadoop的时候 ,有一个脚本是start-all.sh是启动HDFS和YARN两个集群的命令 ,后续的我们会配置spark的系统环境变量 ,所以就会产生冲突 ,我们修改这里的start-all.sh为start-spark.sh

mv start-all.sh start-spark.sh

sbin/start-spark.sh

```
[root@linux01 sbin]# ./start-spark.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/apps/spark-3.0.0-bin-hadoop3.2/logs/spark-root-org.apach
aster-1-linux01.out
linux01: starting org.apache.spark.deploy.worker.Worker, logging to /opt/apps/spark-3.0.0-bin-hadoop3.2/logs/spark-root-
.worker.Worker-1-linux01.out
linux02: starting org.apache.spark.deploy.worker.Worker, logging to /opt/apps/spark-3.0.0-bin-hadoop3.2/logs/spark-root-
.worker.Worker-1-linux02.out
linux03: starting org.apache.spark.deploy.worker.Worker, logging to /opt/apps/spark-3.0.0-bin-hadoop3.2/logs/spark-root-
.worker.Worker-1-linux03.out
j[root@linux01 sbin]# jps
8036 Master
8184 Jps
8124 Worker
```

访问spark提供的WEBUI http://linux01:8080

8088

9870

10002

10000

8020

2181

9000

Spark Master at spark://linux01:7077

URL: spark://linux01:7077
Alive Workers: 3
Cores in use: 12 Total, 0 Used
Memory in use: 19.9 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210507040437-192.168.133.3-40379	192.168.133.3:40379	ALIVE	4 (0 Used)	6.6 GiB (0.0 B Used)	
worker-20210507040438-192.168.133.4-46259	192.168.133.4:46259	ALIVE	4 (0 Used)	6.6 GiB (0.0 B Used)	
worker-20210507040438-192.168.133.5-43960	192.168.133.5:43960	ALIVE	4 (0 Used)	6.6 GiB (0.0 B Used)	

2.3.5 环境变量配置

vi /ect/profile

export SPARK_HOME=/opt/apps/spark-3.0.0-bin-hadoop3.2

export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$HBASE_HOME/bin:$HIVE_HOME/bin:$KAFKA_HOME/bin:$SPARK_HOME/bin::$SP

source /ect/profile

2.3.6 spark-shell

```
第一次启动

[root@linux01 bin]# spark-shell



2021-05-07 04:20:50,923 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

Spark context Web UI available at http://linux01:4040

Spark context available as 'sc' (master = local[*], app id = local-1620375657189).

Spark session available as 'spark'.

Welcome to

    ____              __

   / __/__ ___ _____/ /__

  _\ \/ _ \/ _ `/ __/ '_/

  /___/ .__/\_,_/_/ /_/\_\   version 3.0.0

     /_/


Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_141)

Type in expressions to have them evaluated.

Type :help for more information.


scala>

第二次启动        默认使用集群中所有work的所有的核

[root@linux01 bin]# spark-shell --master spark://linux01:7077

2021-05-07 04:30:36,803 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

Spark context Web UI available at http://linux01:4040

Spark context available as 'sc' (master = spark://linux01:7077, app id = app-20210507043043-0000).

Spark session available as 'spark'.

Welcome to

    ____              __

   / __/__ ___ _____/ /__
```

```
  _\ \/ _ \/ _ `/ __/ '_/
 /___/ .__/\_,_/_/ /_/\_\   version 3.0.0
    /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_141)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

低三次启动　指定程序所有的可用的核数

```
spark-shell --master spark://linux01:7077  --total-executor-cores 6
2021-05-07 04:31:47,494 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://linux01:4040
Spark context available as 'sc' (master = spark://linux01:7077, app id = app-20210507043153-0001).
Spark session available as 'spark'.
Welcome to
    ____              __
   / __/__ ___ _____/ /__
  _\ \/ _ \/ _ `/ __/ '_/
 /___/ .__/\_,_/_/ /_/\_\   version 3.0.0
    /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_141)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

第四次启动　默认情况下一个executor的内存是1G 可以指定executor的内存大小

```
spark-shell --master spark://linux01:7077  --total-executor-cores 6 --executor-memory 512M
2021-05-07 04:33:02,470 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://linux01:4040
Spark context available as 'sc' (master = spark://linux01:7077, app id = app-20210507043309-0002).
Spark session available as 'spark'.
Welcome to
    ____              __
   / __/__ ___ _____/ /__
  _\ \/ _ \/ _ `/ __/ '_/
 /___/ .__/\_,_/_/ /_/\_\   version 3.0.0
    /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_141)
Type in expressions to have them evaluated.
Type :help for more information.
scala>
```

▾ Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210507043309-0002 (kill)	Spark shell	6	512.0 MiB		2021/05/07 04:33:09	root	RUNNING	3 s

▾ Completed Applications (2)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210507043153-0001	Spark shell	6	1024.0 MiB		2021/05/07 04:31:53	root	FINISHED	31 s
app-20210507043043-0000	Spark shell	12	1024.0 MiB		2021/05/07 04:30:43	root	FINISHED	41 s

注意：有在spark的配置文件中配置了hadoop的目录, 所以spark在加载的时候 会读取HDFS的配置文件, 这个时候spark处理的文件系统默认是HDFS分布式文件系统

2.3.7 spark-submit

参数选项

spark-submit

Usage: spark-submit [options] <app jar | python file | R file> [app arguments]

Usage: spark-submit --kill [submission ID] --master [spark://...]

Usage: spark-submit --status [submission ID] --master [spark://...]

Usage: spark-submit run-example [options] example-class [example args]

Options:

--master MASTER_URL spark://host:port, mesos://host:port, yarn,

- k8s://https://host:port, or local (Default: local[*]).

--deploy-mode DEPLOY_MODE Whether to launch the driver program locally ("client") or

- on one of the worker machines inside the cluster ("cluster")
- (Default: client).

--class CLASS_NAME Your application's main class (for Java / Scala apps).

--name NAME A name of your application.

--jars JARS Comma-separated list of jars to include on the driver

- and executor classpaths.

--packages Comma-separated list of maven coordinates of jars to include

- on the driver and executor classpaths. Will search the local
- maven repo, then maven central and any additional remote
- repositories given by --repositories. The format for the
- coordinates should be groupId:artifactId:version.

--exclude-packages Comma-separated list of groupId:artifactId, to exclude while

- resolving the dependencies provided in --packages to avoid
- dependency conflicts.

--repositories Comma-separated list of additional remote repositories to

- search for the maven coordinates given with --packages.

--py-files PY_FILES Comma-separated list of .zip, .egg, or .py files to place

- on the PYTHONPATH for Python apps.

--files FILES Comma-separated list of files to be placed in the working

- directory of each executor. File paths of these files
- in executors can be accessed via SparkFiles.get(fileName).

--conf PROP=VALUE Arbitrary Spark configuration property.

--properties-file FILE Path to a file from which to load extra properties. If not

- specified, this will look for conf/spark-defaults.conf.

--driver-memory MEM Memory for driver (e.g. 1000M, 2G) (Default: 1024M).

--driver-java-options Extra Java options to pass to the driver.

--driver-library-path Extra library path entries to pass to the driver.

--driver-class-path Extra class path entries to pass to the driver. Note that

- jars added with --jars are automatically included in the
- classpath.

--executor-memory MEM Memory per executor (e.g. 1000M, 2G) (Default: 1G).

--proxy-user NAME User to impersonate when submitting the application.

• This argument does not work with --principal / --keytab.

--help, -h Show this help message and exit.

--verbose, -v Print additional debug output.

--version, Print the version of current Spark.

Cluster deploy mode only:

--driver-cores NUM Number of cores used by the driver, only in cluster mode

• (Default: 1).

Spark standalone or Mesos with cluster deploy mode only:

--supervise If given, restarts the driver on failure.

--kill SUBMISSION_ID If given, kills the driver specified.

--status SUBMISSION_ID If given, requests the status of the driver specified.

Spark standalone and Mesos only:

--total-executor-cores NUM Total cores for all executors.

Spark standalone and YARN only:

--executor-cores NUM Number of cores per executor. (Default: 1 in YARN mode,

• or all available cores on the worker in standalone mode)

YARN-only:

--queue QUEUE_NAME The YARN queue to submit to (Default: "default").

--num-executors NUM Number of executors to launch (Default: 2).

• If dynamic allocation is enabled, the initial number of

• executors will be at least NUM.

--archives ARCHIVES Comma separated list of archives to be extracted into the

• working directory of each executor.

--principal PRINCIPAL Principal to be used to login to KDC, while running on

• secure HDFS.

--keytab KEYTAB The full path to the file that contains the keytab for the

• principal specified above. This keytab will be copied to

• the node running the Application Master via the Secure

• Distributed Cache, for renewing the login tickets and the

• delegation tokens periodically.

参数名	参数说明
--master	master 的地址，提交任务到哪里执行，例如 spark://host:port，yarn，local
--deploy-mode	在本地 (client) 启动 driver 或在 cluster 上启动，默认是 client
--class	应用程序的主类，仅针对 java 或 scala 应用
--name	应用程序的名称
--jars	用逗号分隔的本地 jar 包，设置后，这些 jar 将包含在 driver 和 executor 的 classpath 下
--packages	包含在driver 和executor 的 classpath 中的 jar 的 maven 坐标
--exclude-packages	为了避免冲突 而指定不包含的 package
--repositories	远程 repository
--conf PROP=VALUE	指定 spark 配置属性的值， 例如 -conf spark.executor.extraJavaOptions="-XX:MaxPermSize=256m"
--properties-file	加载的配置文件，默认为 conf/spark-defaults.conf
--driver-memory	Driver内存，默认 1G
--driver-java-options	传给 driver 的额外的 Java 选项
--driver-library-path	传给 driver 的额外的库路径
--driver-class-path	传给 driver 的额外的类路径
--driver-cores	Driver 的核数，默认是1。在 yarn 或者 standalone 下使用
--executor-memory	每个 executor 的内存，默认是1G
--total-executor-cores	所有 executor 总共的核数。仅仅在 mesos 或者 standalone 下使用
--num-executors	启动的 executor 数量。默认为2。在 yarn 下使用
--executor-core	每个 executor 的核数。在yarn或者standalone下使用

必备的选项：

--master 需要指定任务运行的模式

--class 指定任务运行的main方法所在的类

spark-submit --master spark://linux01:7077 --total-executor-cores 6 --executor-memory 512M --class com._51doit.bigdata.spark.day01.Demo3 /spark-core-1.0.0.jar

spark-submit --master spark://linux01:7077 --total-executor-cores 6 --executor-memory 512M --class org.apache.spark.examples.SparkPi ./spark-examples_2.12-3.0.0.jar 1000

2.4 历史服务器

在 Spark-shell 没有退出之前, 我们是可以看到正在执行的任务的日志情况:http://linux01:4040. 但是退出 Spark-shell 之后, 执行的所有任务记录全部丢失.所以我们需要配置历史服务器, 来查看以前的shell日志

▾ Running Applications (1)

Application ID		Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210507043722-0003	(kill)	Spark shell	6	512.0 MiB		2021/05/07 04:37:22	root	RUNNING	13 min

▾ Completed Applications (3)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210507043309-0002	Spark shell	6	512.0 MiB		2021/05/07 04:33:09	root	FINISHED	1.2 min
app-20210507043153-0001	Spark shell	6	1024.0 MiB		2021/05/07 04:31:53	root	FINISHED	31 s
app-20210507043043-0000	Spark shell	12	1024.0 MiB		2021/05/07 04:30:43	root	FINISHED	41 s

2.4.1 修改配置文件

mv spark-defaults.conf.template spark-defaults.conf

vi spark-defaults.conf

spark.eventLog.enabled true

spark.eventLog.dir hdfs://linux01:8020/spark_logs

vi spark-env.sh

export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080 -Dspark.history.retainedApplications=30 -Dspark.history.fs.logDirectory=hdfs://linux01:8020/spark_logs"

2.4.2 分发配置文件

scp

2.4.3 创建日志目录

hdfs dfs -mkdir /spark_logs

hdfs dfs -ls /

2.4.4 启动历史服务

sbin/start-history-server.sh

jps

10516 Master

11508 Jps

9318 NameNode

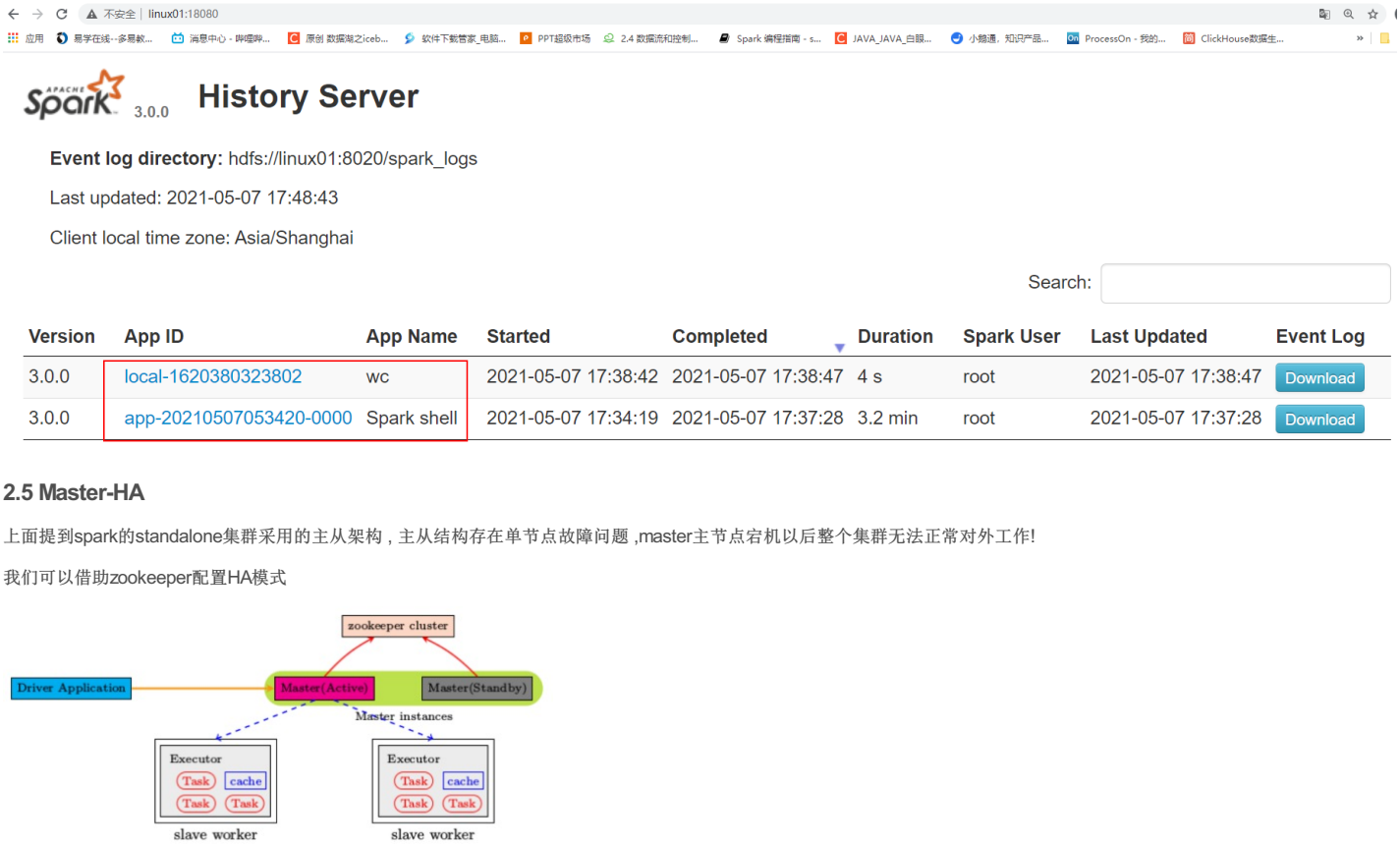
9465 DataNode

11402 HistoryServer

10603 Worker

2.4.5 访问WEB页面

http://linux01:18080



2.5 Master-HA

上面提到spark的standalone集群采用的主从架构，主从结构存在单节点故障问题 ,master主节点宕机以后整个集群无法正常对外工作!

我们可以借助zookeeper配置HA模式

2.5.1 修改配置

在spark-env.sh配置文件中修改如下配置

export SPARK_MASTER_HOST=linux01 注释

export SPARK_DAEMON_JAVA_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER -Dspark.deploy.zookeeper.url=linux01:2181,linux01:2181,linux03:2181 -Dspark.deploy.zookeeper.dir=/spark"

将配置文件同步到集群的每个节点

2.5.2 启动zookeeper

```
#!/bin/bash

for i in 1 2 3

do

ssh doit0${i} "source /etc/profile;/opt/apps/zookeeper-3.4.6/bin/zkServer.sh $1"

done


sleep 2


if [ $1 == start ]

then

for i in {1..3}

do

ssh doit0${i} "source /etc/profile;/opt/apps/zookeeper-3.4.6/bin/zkServer.sh status "

done

fi

sh zk.sh start
```

2.5.3 启动集群

启动集群以后 ,在linux02机器上在启动一个master角色

start-spark.sh

在linux02上执行 sbin/start-master.sh

2.5.4 查看web页面

活跃的主节点

备用主节点

2.5.5 测试

提交程序

spark-submit --master spark://linux01:7077,linux02:7077 --class com._51doit.bigdata.spark.day01.Demo3 /spark-core-1.0.0.jar

停止linux01节点上的master进程

sbin/stop-master.sh

备用节点master成功切换到active状态

提交程序依然正常执行

spark-submit --master spark://linux01:7077,linux02:7077 --class com._51doit.bigdata.spark.day01.Demo3 /spark-core-1.0.0.jar

2.6 yarn模式

独立部署(Standalone）模式由Spark自身提供计算资源，无需其他框架提供资源。这种方式降低了和其他第三方资源框架的耦合性，独立性非常强。但是你也要记住，Spark主要是计算框架，而不是资源调度框架，所以本身提供的资源调度并不是它的强项，所以还是和其他专业的资源调度框架集成会更靠谱一些。所以接下来我们来学习在强大的Yarn环境下 Spark是如何工作的（其实是因为在国内工作中，Yarn使用的非常多)。

Spark 客户端可以直接连接 Yarn，不需要额外构建Spark集群。

有 client 和 cluster 两种模式，主要区别在于：Driver 程序的运行节点不同。

• client：Driver程序运行在客户端，适用于交互、调试，希望立即看到app的输出

• cluster：Driver程序运行在由 RM（ResourceManager）启动的 AM（AplicationMaster）上, 适用于生产环境。

2.6.1 准备一台spark环境机器

```
drwxr-xr-x. 2 1000 1000  4096 Jun  6  2020 bin
drwxr-xr-x. 2 1000 1000   187 May  7 09:58 conf
drwxr-xr-x. 5 1000 1000    50 Jun  6  2020 data
drwxr-xr-x. 4 1000 1000    29 Jun  6  2020 examples
drwxr-xr-x. 2 1000 1000 12288 Jun  6  2020 jars
drwxr-xr-x. 4 1000 1000    38 Jun  6  2020 kubernetes
-rw-r--r--. 1 1000 1000 23312 Jun  6  2020 LICENSE
drwxr-xr-x. 2 1000 1000  4096 Jun  6  2020 licenses
-rw-r--r--. 1 1000 1000 57677 Jun  6  2020 NOTICE
drwxr-xr-x. 7 1000 1000   275 Jun  6  2020 python
drwxr-xr-x. 3 1000 1000    17 Jun  6  2020 R
-rw-r--r--. 1 1000 1000  4488 Jun  6  2020 README.md
-rw-r--r--. 1 1000 1000   183 Jun  6  2020 RELEASE
drwxr-xr-x. 2 1000 1000  4096 Jun  6  2020 sbin
drwxr-xr-x. 2 1000 1000    42 Jun  6  2020 yarn
[root@linux01 spark3-yarn]#
```

2.6.2 修改yarn配置

```
vi spark-env,sh

YARN_CONF_DIR=/opt/apps/hadoop-3.1.1/etc/hadoop/

JAVA_HOME=/opt/apps/jdk1.8.0_141/

HADOOP_CONF_DIR=/opt/apps/hadoop-3.1.1/etc/hadoop/
```

2.6.3 提交应用程序

启动yarn集群 start-yarn.sh

Nodes of the cluster

Cluster
- About
- Nodes
- Node Labels
- Applications
  - NEW
  - NEW_SAVING
  - SUBMITTED
  - ACCEPTED
  - RUNNING

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total
2	0	0	2	0	0 B	12 GB	0 B	0	12

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Sh
3	0	0	0	0	0	0

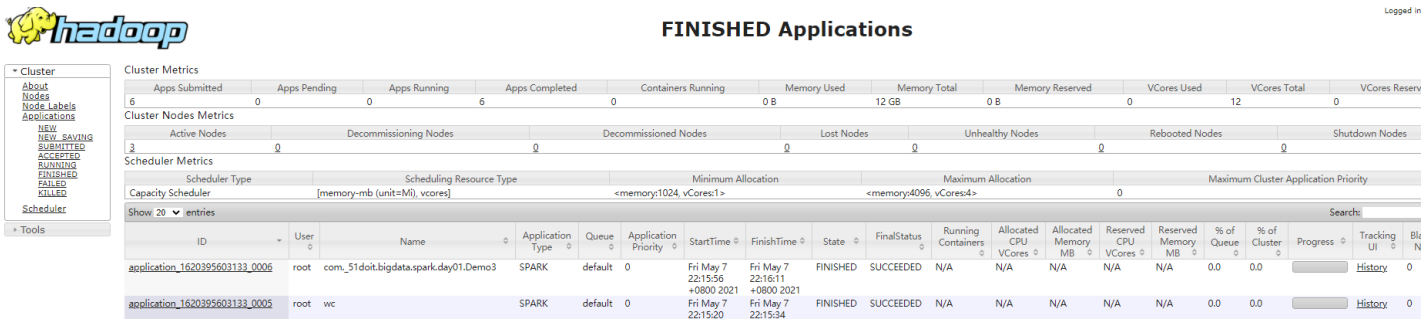
编写自己的spark程序 ;注意在获取SparkContext对象的时候不要设置master

```
object Demo4 {

 def main(args: Array[String]): Unit = {

   val conf = new SparkConf()

   conf.setAppName("wc")

   val sc = new SparkContext(conf)

   val rdd1: RDD[String] = sc.textFile("/word.txt")

   val words: RDD[String] = rdd1.flatMap(_.split("\\s+"))

   // spark相对于scala的基础语法提供了更加丰富的编程实现

   words.map((_, 1)).reduceByKey(_ + _).collect().foreach(println)

   sc.stop()

}

}
```

打包, 上传到linux节点

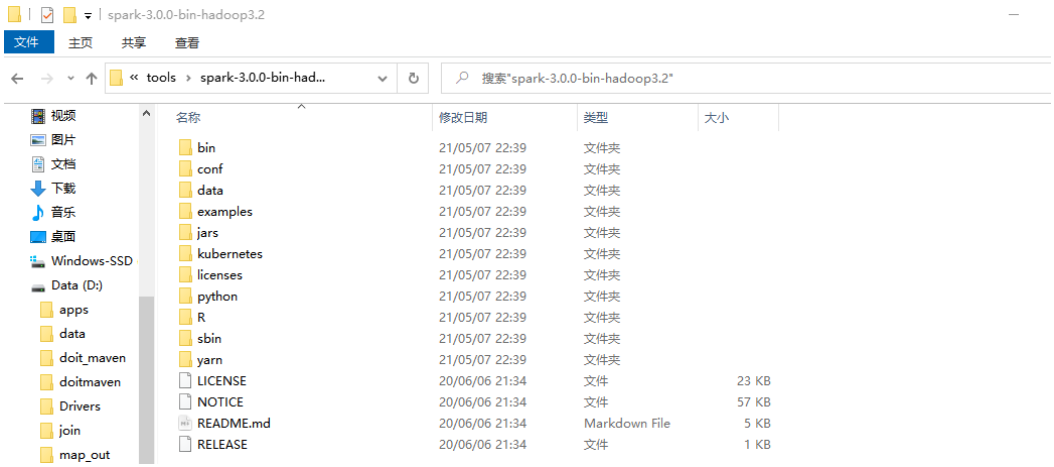
/opt/apps/spark3-yarn/bin/spark-submit --master yarn --deploy-mode cluster --class com._51doit.bigdata.spark.day01.Demo4 /original-spark-core-1.0.0.jar

/opt/apps/spark3-yarn/bin/spark-submit --master yarn --deploy-mode client --class com._51doit.bigdata.spark.day01.Demo4 /original-spark-core-1.0.0.jar



2.7 windows系统环境测试

2.7.1 解压



2.7.2 测试


```
):\tools\spark-3.0.0-bin-hadoop3.2\bin>D:\tools\spark-3.0.0-bin-hadoop3.2\bin\spark-shell.cmd
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://192.168.137.1:4040
Spark context available as 'sc' (master = local[*], app id = local-1620398641393).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.0.0
      /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_65)
Type in expressions to have them evaluated.
Type :help for more information.

scala> sc.textFile("d://word.txt").flatMap(_.split("\\s+")).map((_,1)).reduceByKey(_+_).collect()
21/05/07 22:44:11 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, as a result reporting of ProcessTree metri
cs is stopped
res0: Array[(String, Int)] = Array((d,1), (jim,10), (tom,10), (b,1), ("",3), (hello,5), (f,1), (jerry,5), (tony,5), (cat,6), (e,1
), (a,4), (g,1))
```