小象科技
ChinaHadoop.cn

Spark安装部署

董西成
2017年04月

主要内容

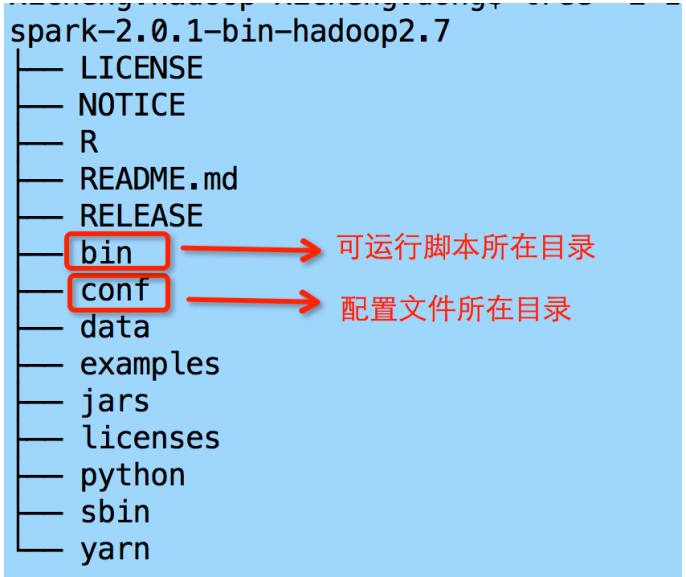
1. 认识Spark安装包
2. Spark安装部署
3. Spark本地与集群模式
4. Spark程序设计初识：Scala语言
5. 总结

主要内容

1. 认识Spark安装包
2. Spark安装部署
3. Spark本地与集群模式
4. Spark程序设计初识：Scala语言
5. 总结

Spark安装包：概述

- 类别
  - ✓ 与Hadoop打包在一起的安装包
    - 比如：spark-2.1.0-bin-hadoop2.7.tgz，spark版本为2.1.0，与hadoop 2.7.0集成
  - ✓ 独立安装包
    - spark-2.1.0-bin-without-hadoop.tgz
- 下载地址
  - ✓ http://spark.apache.org/downloads.html
- Spark安装包



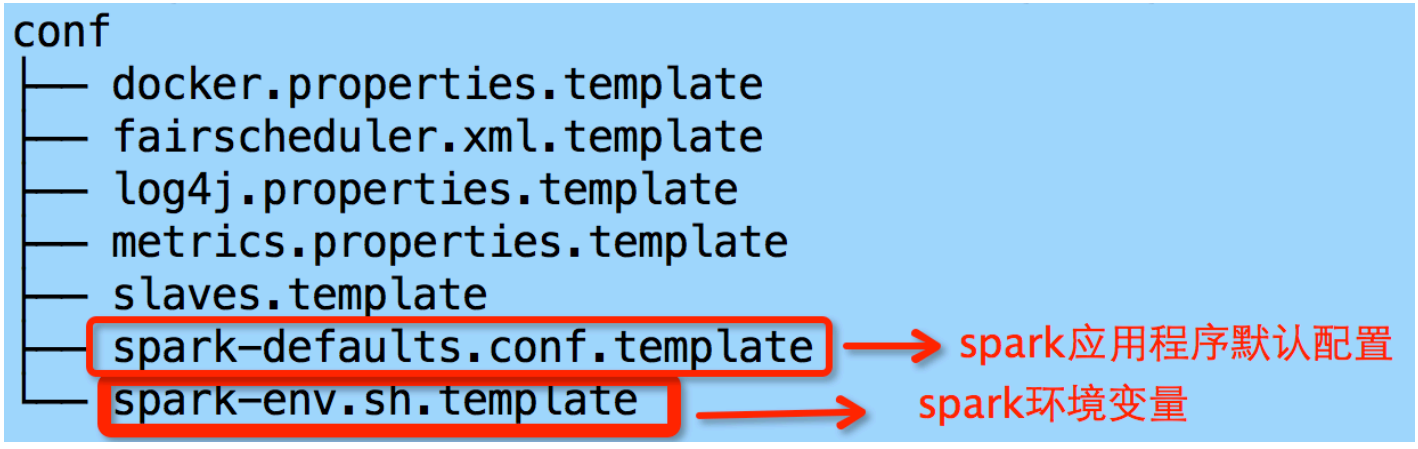
Spark安装包：bin目录

- spark-shell
  - ✓ Spark shell运行模式启动脚本
- spark-submit
  - ✓ 应用程序提交脚本
- run-example
  - ✓ **运行spark提供的示例程序**
- spark-sql
  - ✓ Spark sql命令行启动脚本

```
bin
├── beeline
├── beeline.cmd
├── load-spark-env.cmd
├── load-spark-env.sh
├── pyspark
├── pyspark.cmd
├── pyspark2.cmd
├── run-example
├── run-example.cmd
├── spark-class
├── spark-class.cmd
├── spark-class2.cmd
├── spark-shell
├── spark-shell.cmd
├── spark-shell2.cmd
├── spark-sql
├── spark-submit
├── spark-submit.cmd
├── spark-submit2.cmd
├── sparkR
├── sparkR.cmd
└── sparkR2.cmd
```

Spark安装包：bin目录

- spark-default.conf
  - ✓ 可将spark-defaults.conf.template重命名后产生
  - ✓ 以key/value方式设置spark应用程序的默认参数，比如默认cpu和内数量
- spark-env.sh
  - ✓ 可将spark-env.sh.template重命名后产生
  - ✓ 是一个shell文件，保存了spark的运行环境，比如hadoop配置文件所在路径。



主要内容

1. 认识Spark安装包
2. **Spark安装部署**
3. Spark本地与集群模式
4. Spark程序设计初识：Scala语言
5. 总结

Spark安装部署:基本配置

- 主要任务
  - ✓ 修改conf目录下的spark-defaults.conf和spark-env.sh
  - ✓ 配置并启动spark history server
- spark-defaults.conf配置

```
spark.master=local
```

- spark-env.sh配置
  - ✓ 配置hadoop配置文件所在目录：设置成自己的目录！！！

```
export HADOOP_CONF_DIR=/hadoop-2.7.3/etc/hadoop
```

Spark安装部署：spark historyserver配置与启动

- 找一台节点部署spark history server，比如master
- 在Hadoop配置文件yarn-site.xml增加以下配置
  - ✓ 修改yarn-site.xml（需要重启所有NodeManager生效）

```
<property>
   <name>yarn.log-aggregation-enable</name>
   <value>true</value>
 </property>
<property>
    <name>yarn.log.server.url</name>
    <value>http://master:19888/jobhistory/logs</value>
</property>
```

  - ✓ 分发到所有nodemanager节点上，并重启nodemanager
    - bin/yarn-deamon.sh stop nodemanager
    - bin/yarn-deamon.sh start nodemanager

Spark安装部署：spark historyserver配置与启动

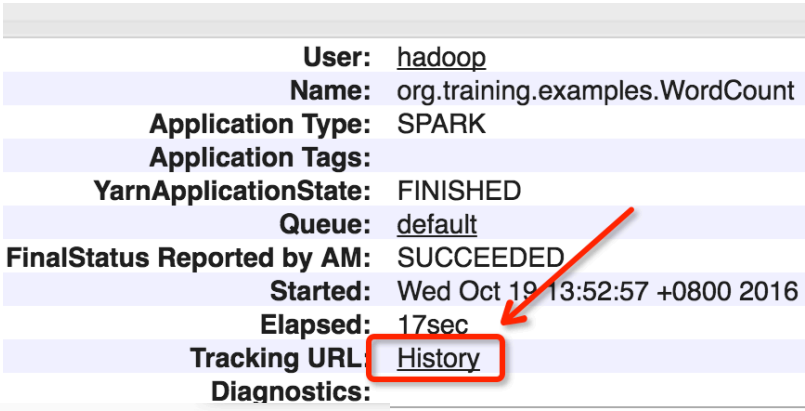
- 修改conf/spark-defaults.conf，增加以下配置
  - ✓ conf/spark-defaults.conf

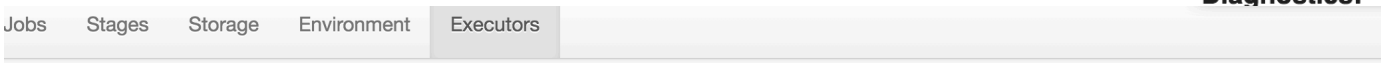
```
spark.yarn.historyServer.address=master:18080
spark.history.ui.port=18080
spark.eventLog.enabled=true
spark.eventLog.dir=hdfs:///tmp/spark/events
spark.history.fs.logDirectory=hdfs:///tmp/spark/events
```

  - ✓ 在HDFS上创建以上两个目录
    - hdfs dfs -mkdir -p /tmp/spark/events
    - hdfs dfs -mkdir -p /tmp/spark/history

Spark安装部署： spark historyserver配置与启动

- 启动Spark history server
  - ✓ sbin/start-history-server.sh
- Spark History server地址
  - ✓ http://master:18080/
- Spark History server使用





MB Total)

	RDD Blocks	Storage Memory	Disk Used	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time	Input	Shuffle Read	Shuffle Write	Logs
	0	0.0 B / 511.1 MB	0.0 B	0	0	2	2	2.4 s	5.1 KB	1124.0 B	2.1 KB	stdout stderr
	0	0.0 B / 511.1 MB	0.0 B	0	0	2	2	2.3 s	2.5 KB	1120.0 B	2.3 KB	stdout stderr
2618	0	0.0 B / 457.9 MB	0.0 B	0	0	0	0	0 ms	0.0 B	0.0 B	0.0 B	stderr stdout

Spark本地模式

在线演示

主要内容

1. 认识Spark安装包
2. Spark安装部署
3. **Spark本地与集群模式**
4. Spark程序设计初识
5. 总结

Spark本地模式

- 进入Spark解压目录
- 进入spark shell模式
  - ✓ bin/spark-shell --master local
- 将以下代码贴如交互式命令行并执行（SparkPi）

```
import scala.math.random
val n = 1000000
 val count = sc.parallelize(1 until n, 100).map { i =>
    val x = random * 2 - 1
    val y = random * 2 - 1
    if (x*x + y*y < 1) 1 else 0
  }.reduce(_ + _)
 println("Pi is roughly " + 4.0 * count / n)
```

- 在浏览器中打开http://localhost:4040

Spark On YARN

- 流程
  - ✓ 安装部署Hadoop集群（分布式模式或伪分布式）
  - ✓ 在spark-env.sh中设置了Hadoop配置文件位置的环境变量
  - ✓ 在命令行中运行前面的SparkPi
  - ✓ 在YARN上观察应用程序运行结果

Spark On YARN

在线演示

主要内容

1. 认识Spark安装包
2. Spark安装部署
3. Spark本地与集群模式
4. **Spark程序设计初识：Scala语言**
5. 总结

构建集成开发环境

- 推荐使用Intellij IDEA
  - ✓ **Eclipse对Scala支持不完善**
- 基本流程
  - ✓ **安装JDK 1.7或更高版本**
  - ✓ **下载Intellij IDEA，打开后，安装scala插件**
  - ✓ **在Intellij中创建scala工程，导入spark-hadoop包**
  - ✓ **编写spark程序**

参考文章：**http://dongxicheng.org/framework-on-yarn/apache-spark-intellij-idea/**

选择开发语言

- Scala
- Java
- Python
- R

Spark Wordcount： Java版

```
private static final Pattern SPACE = Pattern.compile(" ");

public static void main(String[] args) throws Exception {

  if (args.length < 1) {
    System.err.println("Usage: JavaWordCount <file>");
    System.exit(1);
  }

  SparkConf sparkConf = new SparkConf().setAppName("JavaWordCount");
  JavaSparkContext ctx = new JavaSparkContext(sparkConf);
  JavaRDD<String> lines = ctx.textFile(args[0], 1);

  JavaRDD<String> words = lines.flatMap(new FlatMapFunction<String, String>() {
    @Override
    public Iterable<String> call(String s) {
      return Arrays.asList(SPACE.split(s));
    }
  });

  JavaPairRDD<String, Integer> ones = words.mapToPair(new PairFunction<String, String, Integer>() {
    @Override
    public Tuple2<String, Integer> call(String s) {
      return new Tuple2<String, Integer>(s, 1);
    }
  });

  JavaPairRDD<String, Integer> counts = ones.reduceByKey(new Function2<Integer, Integer, Integer>() {
    @Override
    public Integer call(Integer i1, Integer i2) {
      return i1 + i2;
    }
  });

  List<Tuple2<String, Integer>> output = counts.collect();
  for (Tuple2<?,?> tuple : output) {
    System.out.println(tuple._1() + ": " + tuple._2());
  }
```

创建JavaSparkContext

加载数据目录

对每行分词

生成key/value对

统计每个词对一个的频率

打印最终结果

Spark Wordcount： Scala版

```
def main(args: Array[String]) {
  val conf = new SparkConf().setAppName("Wordcount")
  val sc = new SparkContext(conf)
  val input = if (args.length > 0) args(0) else "/tmp/input"
  sc.textFile(input).flatMap(_.split(" "))
    .map(x => (x, 1))
    .reduceByKey(_ + _)
    .saveAsTextFile("/tmp/output")
  sc.stop()
}
```

创建SparkContext

处理输入数据，并产生最终结果，保存到输出目录下

Spark程序设计—Scala

- Java JVM的高层次语言
  - ✓面向对象+函数式编程
- 静态类型
  - ✓性能与Java差不多
  - ✓通常不需要显式写出类型（类型推断机制）
- 与Java结合完好
  - ✓可直接使用任意Java类，可继承自Java类，也可从Java代码中调用Scala代码

Spark程序设计—Scala

- 定义变量：

```
var x: Int = 7
var x = 7 //  类型推断
val y = “hi” //只读的
```

- 函数：

```
def square(x: Int): Int = x*x
def square(x: Int) = {x*x}
def announce(text: String) {
  println(text)
}
```

- Java等价代码

```
int x = 7;
final string y = “hi”;
int square(int x) {
  return x*x;
}
void announce(String text) {
  System.out.println(text);
}
```

Spark程序设计—Scala

- 泛型：

```
var arr =  new Array[Int]
(8)
val lst = List(1, 2, 3)
```

- 索引：

```
arr(5) = 7
println(lst(5))
```

- Java等价代码

```
int [] arr =   new int[8];
List<Integer> lst =
   new ArrayList<Interger>();
lst.add(…)
arr[5] = 7;
system.out.println(lst.get(5));
```

Spark程序设计—Scala

用函数式编程的方式处理集合：

```
var list = List(1, 2, 3)
list.foreach(x => println(x)) //  打印1，  2，  3
list.foreach(println)

list.map(x => x+2) // => List(3, 4, 5)
list.map(_+2)

list.filter(x => x%2 == 1) // => List(1, 3)
list.filter(_ % 2 == 1)

list.reduce((x, y) => x + y) // 6
List.reduce(_ + _)
```

Spark程序设计—Scala闭包

```
(x: Int) => x +2  //闭包标准写法

x => x + 2 //类型推断

_  + 2 //每个元素作用一次

x => {  //代码片段
  var numberToAdd = 2
x + numberToAdd
}
//如果闭包太长，可以函数的方式传入
def addTwo(x：  Int): Int = x + 2
List.map(addTwo)
```

Spark程序设计—Scala集合操作

Method on Seq[T]	Explanation
`map(f: T => U): Seq[U]`	Pass each element through f
`flatMap(f: T => Seq[U]): Seq[U]`	One-to-many map
`filter(f: T => Boolean): Seq[T]`	Keep elements passing f
`exists(f: T => Boolean): Boolean`	True if one element passes
`forall(f: T => Boolean): Boolean`	True if all elements pass
`reduce(f: (T, T) => T): T`	Merge elements using f
`groupBy(f: T => K): Map[K,List[T]]`	Group elements by f(element)
`sortBy(f: T => K): Seq[T]`	Sort elements by f(element)
. . .	

Scala从入门到精通

- 第一节：Scala语言初步
- 第二节：Scala基本类型及操作、程序控制结构
- 第三节：Array、List
- 第四节：Set、Map、Tuple、队列操作实战
- 第五节：函数与闭包
- 第六节：类和对象（一）
- 第七节：类和对象（二）
- 第八节：包和引入
- 第九节：继承与组合
- 第十节：Scala类层次结构、Traits初步
- 第十一节：Trait进阶
- 第十二节：I/O与正则表达式
- 第十三节：高阶函数
- 第十四节：Case Class与模式匹配（一）
- 第十五节：Case Class与模式匹配（二）
- 第十六节：泛型与注解
- 第十七节：类型参数（一）
- 第十八节：隐式转换与隐式参数（一）
- 第十九节：隐式转换与隐式参数（二）
- 第二十节：类型参数（二）
- 第二十一节：类型参数（三）
- 第二十二节：高级类型 （一）
- 第二十三节：高级类型 （二）
- 第二十四节：高级类型 （三）
- 第二十五节：提取器（Extractor）
- 第二十六节：Scala并发编程基础
- 第二十七节：Scala操纵XML
- 第二十八节：Scala与JAVA互操作
- 第二十九节：Scala数据库编程
- 第三十节：Scala脚本编程与结束语

详细资料：

https://yq.aliyun.com/topic/69

Scala上机练习

- 数组操作：

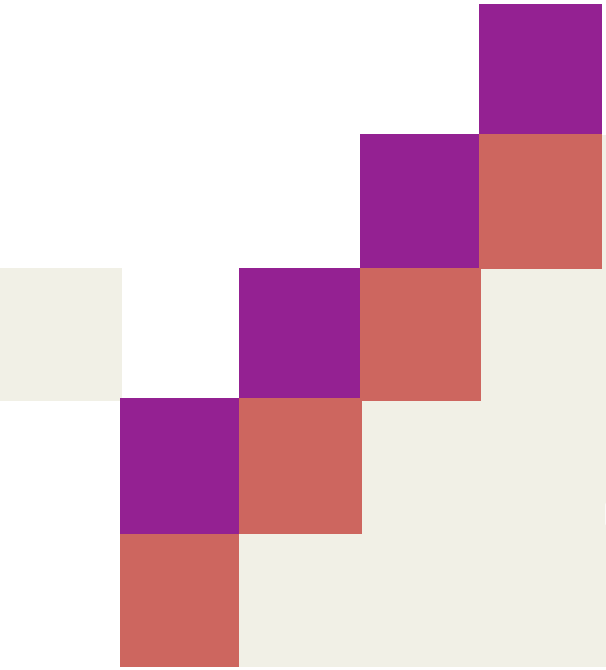
https://yq.aliyun.com/articles/60391?spm=5176.8251999.569296.3.JUn15r

- 集合操作：

https://yq.aliyun.com/articles/60390?spm=5176.8251999.569296.4.JUn15r

- 函数与闭包：

https://yq.aliyun.com/articles/60389?spm=5176.8251999.569296.5.JUn15r



Q&A?

Thank You !