小象科技
ChinaHadoop.cn

简易电影受众分析系统

董西成
2017年04月

主要内容

1. Spark开发环境搭建
2. 电影受众分析系统
3. Spark在线演示

主要内容

1. Spark开发环境搭建
2. 电影受众分析系统
3. Spark在线演示

Spark开发环境搭建

- JDK安装配置
  - ✓ **JDK 1.7及以上版本**
- Scala安装配置
  - ✓ **Scala 2.10**
- Eclipse或Intellij IDEA
  - ✓ 下载最新版

- https://taoistwar.gitbooks.io/spark-developer-guide/content/spark_base/spark_dev_environment.html
- http://blog.tomgou.xyz/shi-yong-intellij-ideapei-zhi-sparkying-yong-kai-fa-huan-jing-ji-yuan-ma-yue-du-huan-jing.html

Spark开发环境搭建

- 创建一个空的Scala Project，项目名称为wordcount，包名为org.training.spark

```
mvn archetype:generate \
-DarchetypeGroupId=org.scala-tools.archetypes \
-DarchetypeArtifactId=scala-archetype-simple \
-DarchetypeVersion=1.1 \
-DremoteRepositories=http://scala-tools.org/repo-releases \
-DarchetypeCatalog=internal \
-DinteractiveMode=false \
-Dversion=1.0-SNAPSHOT \
-DgroupId=org.training.spark  \
-DartifactId=wordcount
```

Spark开发环境搭建

- 在别人工程基础上修改

git clone https://github.com/XichengDong/simplespark

- 进入目录：

cd simplespark

- 编译打包：

mvn package

- 利用阿里云maven中央仓库加速：

http://www.jianshu.com/p/4d5bb95b56c5

主要内容

1 Spark开发环境搭建

2 电影受众分析系统

3 Spark在线演示

电影受众分析系统背景

- 数据集
  - http://grouplens.org/datasets/movielens/
  - MovieLens 1M Dataset
- 相关数据文件
  - users.dat
    - UserID::Gender::Age::Occupation::Zip-code
  - movies.dat
    - MovieID::Title::Genres
  - ratings.dat
    - UserID::MovieID::Rating::Timestamp

电影受众分析系统任务

- 受众分析
  - ✓看过“Lord of the Rings, The (1978)”用户年龄和性别分布
  - ✓年龄段在“18-24”的男性年轻人，最喜欢看哪10部电影
  - ✓得分最高的10部电影；看过电影最多的前10个人；女性看多最多的10部电影；男性看过最多的10部电影

主要内容

1 Spark开发环境搭建

2 电影受众分析系统

3 Spark在线演示

任务1

- 受众分析
  - ✓看过“Lord of the Rings, The (1978)”用户年龄和性别分布
  - ✓年龄段在“18-24”的男性年轻人，最喜欢看哪10部电影
  - ✓得分最高的10部电影；看过电影最多的前10个人；女性看多最多的10部电影；男性看过最多的10部电影

任务1

- 你将学会
  - ✓如何将文本文件（目录）构造成RDD
  - ✓**Transformation**
    - **Map, filter, groupByKey, join**
  - ✓**Action**
    - **Foreach, take**

在线演示(1)

- **Step 1：数据加载**

```
val usersRdd = sc.textFile("users.dat")

val ratingsRdd = sc.textFile("ratings.dat")
```

在线演示(2)

- Step 2: 数据抽取

```
//users: RDD[(userID, (gender, age))]
  val users = usersRdd.map(_.split("::")).map { x =>
    (x(0), (x(1), x(2)))
  }

  //rating: RDD[Array(userID, movieID, ratings, timestamp)]
  val rating = ratingsRdd.map(_.split("::"))

  //usermovie: RDD[(userID, movieID)]
  val usermovie = rating.map{ x =>
    (x(0), x(1))
  }.filter(_._2.equals(MOVIE_ID))
```

在线演示(3)

- **Step 3: Join RDDs**

```
//useRating: RDD[(userID, (movieID, (gender, age))]
  val userRating = usermovie.join(users)

  //movieuser: RDD[(movieID, (movieTile, (gender, age))]
  val userDistribution = userRating.map { x =>
    (x._2._2, 1)
  }.reduceByKey(_ + _)

  userDistribution.foreach(println)
```

任务2

- 受众分析
  - ✓看过“Lord of the Rings, The (1978)”用户年龄和性别分布
  - ✓**年龄段在“18-24”的男性年轻人，最喜欢看哪10部电影**
  - ✓得分最高的10部电影；看过电影最多的前10个人；女性看多最多的10部电影；男性看过最多的10部电影

任务2

- 你将学会:
  - ✓如何将文本文件（目录）构造成RDD
  - ✓Transformation
    - Map, filter, groupByKey, join, sortByKey
  - ✓Action
    - Foreach, take
  - ✓Others
    - broadcast

在线演示(1)

- **Step 1：数据加载**

```
val usersRdd = sc.textFile(“users.dat”)

 val moviesRdd = sc.textFile("movies.dat")

val ratingsRdd = sc.textFile(“ratings.dat”)
```

在线演示(2)

- Step 2: 数据抽取

```
//users: RDD[(userID, age)]
  val users = usersRdd.map(_.split("::")).map { x =>
    (x(0), x(2))
  }.filter(_._2.equals(USER_AGE))

  //Array[String]
  val userlist = users.map(_._1).collect()

  //broadcast
  val userSet = HashSet() ++ userlist
  val broadcastUserSet = sc.broadcast(userSet)
```

在线演示(3)

- **Step 3: Join RDDs**

```
val topKmovies = ratingsRdd.map(_.split("::")).map{ x =>
  (x(0), x(1))
}.filter { x =>
  broadcastUserSet.value.contains(x._1)
}.map{ x=>
  (x._2, 1)
}.reduceByKey(_ + _).map{ x =>
  (x._2, x._1)
}.sortByKey(false).map{ x=>
  (x._2, x._1)
}.take(10)
```

在线演示(4)

- **Step 4: Print result**

```
val movieID2Name = moviesRdd.map(_.split("::")).map { x =>
(x(0), x(1))
}.collect().toMap

topKmovies.map(x => (movieID2Name.getOrElse(x._1, null),
x._2)).foreach(println)
```

任务3

- 受众分析
  - ✓看过“Lord of the Rings, The (1978)”用户年龄和性别分布
  - ✓年龄段在“18-24”的男性年轻人，最喜欢看哪10部电影
  - ✓**得分最高的10部电影；看过电影最多的前10个人；女性看多最多的10部电影；男性看过最多的10部电影**

任务2

- 你将学会
  - ✓如何将文本文件（目录）构造成RDD
  - ✓**Transformation**
    - **Map, filter, groupByKey, join, sortByKey**
  - ✓**Action**
    - **Foreach, take**
  - ✓**Others**
    - **cache**

在线演示(1)

- **Step 1：数据加载**

```
val ratingsRdd = sc.textFile(“ratings.dat”)
```

在线演示(2)

- **Step 2: 数据抽取**

```
val ratings = ratingsRdd.map(_.split("::")).map { x =>
(x(0), x(1), x(2))
}.cache
```

在线演示(3)

- **Step 3: Join RDDs**

```
val topKScoreMostMovie = ratings.map{x =>
  (x._2, (x._3.toInt, 1))
}.reduceByKey { (v1, v2) =>
  (v1._1 + v2._1, v1._2 + v2._2)
}.map { x =>
  (x._2._1.toFloat / x._2._2.toFloat, x._1)
}.sortByKey(false).
    take(10).
    foreach(println)
```

思考题

如何实现：

SELECT store, product, SUM(amount), MIN(amount), MAX(amount), SUM(units)

FROM sales

GROUP BY store, product

已知sales表中store，product，amount和units四列值已经存到如下RDD中：

```
val sales=sc.parallelize(List(
  ("West",  "Apple",  2.0, 10),
  ("West",  "Apple",  3.0, 15),
  ("West",  "Orange", 5.0, 15),
  ("South", "Orange", 3.0, 9),
  ("South", "Orange", 6.0, 18),
  ("East",  "Milk",   5.0, 5)))
```

思考题：答案

```
sales.map{ case (store, prod, amt, units) => ((store, prod), (amt,
amt, amt, units)) }.
  reduceByKey((x, y) =>
   (x._1 + y._1, math.min(x._2, y._2), math.max(x._3, y._3), x._4 +
y._4)).collect
```

下一讲内容预告

- Spark内部实现与源码剖析
  - ✓ **Spark计算引擎运行流程**
  - ✓ **Spark Shuffle的实现**
  - ✓ **Spark源代码阅读方法**
- 问题
  - ✓ **Spark程序是怎么并行化的？**
  - ✓ **Spark引擎内部是否将所有RDD保存到内存？**
  - ✓ **应该如何阅读、编译和运行Spark源代码？**

hadoop123：董西成的微信公众号

专注于Hadoop/spark等大数据相关技术的分享

小象科技
ChinaHadoop.cn

联系我们：

- 新浪微博：ChinaHadoop
- 微信公号：ChinaHadoop

让你的数据产生价值！