

实验三: SPARK RDD 基础编程方法二

胡嘉鑫

102102145

2023 年 11 月 11 日

目录

1 实验目的	1
2 实验平台	1
3 实验步骤	2
3.1 分析用户行为数据	2
3.1.1 二次排序和筛选	2
3.1.2 去重和统计	2
3.1.3 数据连接和格式化	3
3.2 分析校运动会数据	4
3.2.1 计算所有比赛项目的平均成绩	4
3.2.2 统计每个班级的名次总数	4
3.2.3 筛选并统计特定项目的成绩	4
4 出现的问题及其解决方案	4

1 实验目的

- 理解 Spark 工作流程;
- 掌握 Spark RDD 基础编程方法.

2 实验平台

- OS: Linux
- Hadoop v3.3.5

- JDK v1.8
- SPARK v3.4.0

3 实验步骤

3.1 分析用户行为数据

现在有一份用户行为数据，包括用户 ID、时间戳、事件类型和事件内容。请使用大数据计算工具（如 Spark）进行分析，以解决以下问题。

3.1.1 二次排序和筛选

- 从给定的数据集中，使用二次排序的方法，首先按照用户 ID 升序排序，然后按照时间戳降序排序。输出按照排序条件筛选后的前 10 条记录。
- 从给定的数据集中，使用二次排序的方法，首先按照用户 ID 降序排序，然后按照事件类型升序排序。输出按照排序条件筛选后的前 10 条记录。

```

App.scala
15
16 def secondarySort(data : RDD[String]) : Unit = {
17   // 从给定的数据集中，使用二次排序的方法，首先按照用户 ID 升序排
18   // 按照时间戳降序排序。输出按照排序条件筛选后的前 10 条记录。
19   data.map(
20     x => (new SecondarySortKey0(x.split(",")(0).trim, x.split(",")
21       ), sortByKey(false).map(_._2).take(10).foreach(println)
22   )
23   println("=====")
24
25   // 从给定的数据集中，使用二次排序的方法，首先按照用户 ID 降序排
26   // 按照事件类型升序排序。输出按照排序条件筛选后的前 10 条记录。
27   data.map(
28     x => (new SecondarySortKey1(x.split(",")(0).trim, x.split(",")
29       ), sortByKey(false).map(_._2).take(10).foreach(println)
30   )
31 }
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

<exp3_src_spark_rdd_sql/pro1/src/main/scala/net/homework//88732:/bin/bash
24 HuJiaxingM102102145:~/workspace/spark_demo/102102145_exp3_src_spar
25 k_rdd_sql/pro1$ bash run.sh
26 1, 2023-04-15 09:50:00, View, Product D
27 1, 2023-04-15 09:35:00, Click, Product A
28 1, 2023-04-15 09:20:00, Purchase, Product D
29 1, 2023-04-15 08:55:00, Click, Product D
30 1, 2023-04-15 08:55:00, Click, Product D
31 1, 2023-04-15 08:40:00, Add to Cart, Product A
32 1, 2023-04-15 08:40:00, Add to Cart, Product A
33 1, 2023-04-15 08:30:00, Click, Product A
34 1, 2023-04-15 08:30:00, Click, Product A
35 2, 2023-04-15 09:55:00, Click, Product B
36
37
38 11 5, 2023-04-15 09:10:00, Click, Product G
39 10 5, 2023-04-15 09:10:00, Click, Product G
40 9 5, 2023-04-15 10:05:00, Click, Product G
41 8 4, 2023-04-15 09:45:00, Click, Product E
42 7 4, 2023-04-15 08:57:00, View, Product E
43 6 4, 2023-04-15 08:57:00, View, Product E
44 5 3, 2023-04-15 09:05:00, Add to Cart, Product C
45 4 3, 2023-04-15 09:05:00, Add to Cart, Product C
46 3 3, 2023-04-15 08:42:00, Click, Product C
47 2 3, 2023-04-15 08:42:00, Click, Product C
48
49 1 HuJiaxingM102102145:~/workspace/spark_demo/102102145_exp3_src_spar
50 k_rdd_sql/pro1$
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

图 1: 运行结果

3.1.2 去重和统计

- 从给定的数据集中，找出重复的事件记录（完全相同的记录），并将它们去重。输出去重后的数据集。输出的数据格式：用户 ID 事件类型事件内容

- 对去重后的数据集，统计每种事件类型的数量，并输出事件类型和对应的数量。

```
App.scala
15 def distinctAndStatistics(data : RDD[String]) {
16   // 从给定的数据集中，找出重复的事件记录（完全相同的记录），并
17   // 输出去重后的数据集。输出的数据格式：用户ID 事件类型 事件内容
18   val res = data.distinct().map(
19     _.split(",").toList
20   ).map(
21     x => (x(2).trim, (x(0).trim, x(3).trim))
22   )
23   res.foreach(
24     x => println(s"${x._2._1}\t${x._1}\t${x._2._2}")
25   )
26   println("=====")
27   // 对去重后的数据集，统计每种事件类型的数量，并输出事件类型和
28   // 数量
29   res.groupByKey().map(x => (x._1, x._2.size)).foreach(println)
30 }
31 }
```

```
<exp3_src_spark_rdd_sql/pro1/src/main/scala/net/homework//88732:/bin/bas
31
30 [INFO] Total time: 9.162 s
29 [INFO] Finished at: 2023-11-09T19:24:44+08:00
28 [INFO]
27
26 5 Click Product G
25 3 Purchase Product C
24 4 View Product E
23 2 Click Product F
22 5 Click Product G
21 2 Add to Cart Product F
20 2 Purchase Product B
19 3 Add to Cart Product C
18 1 Add to Cart Product A
17 1 View Product D
16 2 Click Product B
15 1 Click Product A
14 1 Click Product A
13 1 Purchase Product D
12 2 Click Product F
11 4 Click Product E
10 3 Click Product C
9 2 View Product B
8 1 Click Product D
7 3 View Product C
6
5 (Add to Cart,3)
4 (View,4)
3 (Purchase,3)
2 (Click,10)
```

图 2: 运行结果

3.1.3 数据连接和格式化

从用户信息数据集中读取用户 ID 和用户名称。将两个数据集根据用户 ID 进行连接，得到一个新的数据集，其中包含用户信息和相关用户行为记录。

- 对连接后的数据集进行进一步处理，将用户信息和活动详情合并成一个格式化的字符串，例如：“用户名称 - 事件类型 - 事件内容”。
- 输出格式化后的报告，包括每个用户的用户名称、事件类型和事件数量。输出格式为：用户名称-事件类型-事件数量

```

App.scala
31
30 val userActivity = userActivityData
29 .map(_.split(",").toList)
28 .map(x => (x(0).trim, (x(2).trim, x(3).trim)))
27
26 userInfo.cache()
25 userActivity.cache()
24
23 // 从用户信息数据集中读取用户 ID 和用户名。将两个数据集根据
22 // 接，得到一个新的数据集，其中包含用户信息和相关用户行为记录。
21 val info = userInfo.join(userActivity)
20 info.cache()
19
18 // 对连接后的数据集进行进一步处理，将用户信息和活动详情合并成
17 // 字符串，例如："用户名 - 事件类型 - 事件内容"。
16 info
15 .map(_.2)
14 .map(x => s"${x._1} - ${x._2._1} - ${x._2._2}")
13 .foreach(println)
12
11 println("=====")
10
9 // 输出格式化后的报告，包括每个用户的用户名、事件类型和事
8 // 输出格式为：用户名--事件类型--事件数量
7 info
6 .map(_.2)
5 .map(x => (s"${x._1}-${x._2._1}", s"${x._2._2}"))
4 .groupByKey()
3 .map(x => (x._1.split("-")(0), x._1.split("-")(1), x._2.size))
2 .foreach(x => println(s"${x._1}--${x._2}--${x._3}"))
1
0 }
App.scala L:157/157 1 100% unix utf-8[scala]

```

```

exp3_src_spark_rdd_sql/pro1/src/main/scala/net/homework//88732/bin/bas
. 31 Carol - Add to Cart - Product C
. 30 Carol - Click - Product C
29 Carol - Add to Cart - Product C
28 Carol - View - Product C
27 Carol - Purchase - Product C
26 Alice - Click - Product A
25 Alice - Add to Cart - Product A
24 Alice - Click - Product D
23 Alice - Click - Product A
22 Alice - Add to Cart - Product A
21 Alice - Click - Product D
20 Alice - Purchase - Product D
19 Alice - Click - Product A
18 Alice - View - Product D
17
16 Eve--Click--3
15 Alice--Add to Cart--2
14 Carol--View--1
13 Bob--View--2
12 Bob--Purchase--2
11 Carol--Click--2
10 Bob--Click--4
9 Alice--Purchase--1
8 Dave--View--2
7 Dave--Click--1
6 Carol--Purchase--1
5 Alice--Click--5
4 Alice--View--1
3 Carol--Add to Cart--2
2 Bob--Add to Cart--1
1 HuJiaxinM102102145:~/workspace/spark_demo/102102145_exp3_src_spar
0 rdd_sql/pro1$

```

图 3: 运行结果

3.2 分析校运动会数据

有一份运动会竞赛结果的数据集，其中包括比赛项目、班级、运动员、成绩和名次。使用 Spark SQL 进行分析。

3.2.1 计算所有比赛项目的平均成绩

```

App.scala
3 // 比赛项目不为空
2 // 且成绩不为空
1 val rddInfo = textData
0 .map(x => {
1   val arr = x.split(",")
2   Row(arr(0).trim, arr(1).trim, arr(2).trim,
3     arr(3).trim.toDouble, arr(4).trim.toLong)
4 })
5
6 val schema = new StructType(Array(
7   new StructField("category", StringType),
8   new StructField("class", StringType),
9   new StructField("player", StringType),
10  new StructField("score", DoubleType),
11  new StructField("rank", LongType)
12 ))
13 //rddInfo.foreach(println)
14
15 val dfInfo = spark.createDataFrame(rddInfo, schema)
16 //dfInfo.show()
17
18 dfInfo.createOrReplaceTempView("info")
19
20 // 计算所有比赛项目的平均成绩
21 println("计算所有比赛项目的平均成绩:")
22 spark
23 .sql("""SELECT ROUND(AVG(score), 2) AS Avg_Score
24       FROM info
25       GROUP BY category""")
26 .show()
27
28 println("=====")
App.scala L:32/113 11 28% unix utf-8[scala]

```

```

[66/170]
[WARNING] org.scala-lang.modules:scala-xml_2.12:1.2.0 requires scala ve
rsion: 2.12.8
[WARNING] Multiple versions of scala libraries detected!
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-compiler-plugin:2.0.2:testCompile (default-testCompile)
@ pro2 ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ pro2 ---
[INFO] Tests are skipped.
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ pro2 ---
[INFO] Building jar: /home/HuJiaxin/workspace/spark_demo/102102145_exp3
_src_spark_rdd_sql/pro2/target/pro2-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 8.364 s
[INFO] Finished at: 2023-11-11T15:37:53+08:00
[INFO]

计算所有比赛项目的平均成绩:
+-----+
|Avg_Score|
+-----+
| 1.83|
| 11.92|
| 6.93|
| 12.74|
| 22.06|
+-----+

```

图 4: 运行结果

3.2.2 统计每个班级的名次总数

- 统计每个班级获得的第一名、第二名和第三名的次数。
- 列出获得第一名次数超过 2 次的班级。

```
App.scala
23 // 统计每个班级获得的第一名、第二名和第三名的次数。
21 println("统计每个班级获得的第一名、第二名和第三名的次数:")
20 spark
19   .sql("""SELECT info.class, COUNT(*) AS First_Num
18         FROM info
17         WHERE rank=1
16         GROUP BY class""")
15   .show()
14 spark
13   .sql("""SELECT info.class, COUNT(*) AS Second_Num
12         FROM info
11         WHERE rank=2
10         GROUP BY class""")
9     .show()
8 spark
7   .sql("""SELECT info.class, COUNT(*) AS Third_Num
6         FROM info
5         WHERE rank=3
4         GROUP BY class""")
3     .show()
2     println("=====")
1 // 列出获得第一名次数超过 2 次的班级。
0 println("列出获得第一名次数超过 2 次的班级:")
1 spark
2   .sql("""SELECT info.class, COUNT(*) AS Num
3         FROM info
4         WHERE rank=1
5         GROUP BY class having count(*)>2""")
6     .show()
7     println("=====")
App.scala L:84/113 41 74% unix utf-8[scala]
```

```
22.06|
+-----+
=====
统计每个班级获得的第一名、第二名和第三名的次数:
+-----+
|class|First_Num|
+-----+
| A班 |      3|
| B班 |      2|
+-----+

+-----+
|class|Second_Num|
+-----+
| A班 |      3|
| B班 |      1|
| C班 |      1|
+-----+

+-----+
|class|Third_Num|
+-----+
| B班 |      1|
| C班 |      4|
+-----+

=====
列出获得第一名次数超过 2 次的班级:
+-----+
|class|Num|
+-----+
| A班 |   3|
+-----+
```

图 5: 运行结果

3.2.3 筛选并统计特定项目的成绩

- 筛选出所有田径项目（如 100 米短跑、200 米短跑）的比赛结果。
- 统计在这些田径项目中获得前三名的个人数量。

```
App.scala
3 // 筛选出所有田径项目（如 100 米短跑、200 米短跑）的比赛结果。
4 println("筛选出所有田径项目（如 100 米短跑、200 米短跑）的比赛结果：")
5 spark
6   .sql("""SELECT
7     FROM info
8     ORDER BY category, rank""")
9   .show()
10
11 // 统计在这些田径项目中获得前三名的个人数量。
12 println("统计在这些田径项目中获得前三名的个人数量：")
13 spark
14   .sql("""SELECT DISTINCT COUNT(*)
15     FROM info
16     WHERE rank <= 3""")
17   .show()
18
19 spark.stop()
20 }
```

category	class	player	score	rank
100米短跑	A班	张三	12.45	1
100米短跑	B班	李四	12.62	2
100米短跑	C班	王五	12.75	3
100米短跑	A班	李华	12.82	4
100米短跑	C班	王明	13.05	5
200米短跑	A班	刘强	21.1	1
200米短跑	A班	张三	21.81	2
200米短跑	C班	王五	22.35	3
200米短跑	C班	王明	22.45	4
200米短跑	B班	李四	22.6	5
跳远	B班	李四	6.95	1
跳远	A班	张三	7.05	2
跳远	B班	李华	6.8	3
跳高	B班	李四	1.9	1
跳高	A班	张三	1.85	2
跳高	C班	王五	1.75	3
铅球	A班	张三	12.34	1
铅球	C班	王明	11.92	2
铅球	C班	王五	11.5	3

统计在这些田径项目中获得前三名的个人数量：

count(1)
15

图 6: 运行结果

4 出现的问题及其解决方案

没有问题.