基于 spark 框架在电商平台中的数据分析和处理

崔正昊 张桂花

四川大学锦城学院 计算机与软件学院 四川 成都 611731

【摘 要】本文通过广大用户对电商平台中的各个商品的点击，下单以及支付数详细信息进行分析和处理。基于 Spark 开源平台，利用 sparkCore，使用 API 实现了针对不同商品和用户之间的浏览记录和下单支付等模式，形成实时推荐优先级列表中最受欢迎的产品。它可以满足用户对商品的需求，增加电子商务平台的销量，大大减少企业和用户的冗余信息过载问题。

【关键词】sparkCore 编程；RDD 算子；Spark 框架；电商平台

1 引言

随着信息技术和互联网时代的蓬勃发展，电商平台发展的重点越来越倾向于大数据的分析和技术。电子商务平台拥有大量的商品和种类，如何让用户准确地选择自己需要的商品，并将相似商品的信息推送给用户，改善用户的购物体验，增加用户的黏性，成为电子商务平台的研究热点。

在这一背景下，频繁项集挖掘技术是目前数据挖掘技术的基础，最初国内外主要采用的关联规则分析、序列项集、相关性分析等数据挖掘技术，它们都是以频繁项集挖掘技术作为核心基础的。[1]

本文实现了一种基于 Spark 框架的数据分析处理系统，根据用户的实时数据来分析并得出热门商品推荐给用户，一方面可以解决用户无法快速准确搜索海量商品的问题，另一方面可以根据不同用户的不同需求推荐个性化商品。基于 Spark 平台的实时推荐系统相较于传统的离线推荐系统，能够得到更快的训练速度以及反馈速度。[2]

2 系统架构

2.1 技术选型

2.1.1 项目架构图

项目架构图如图 1 所示：

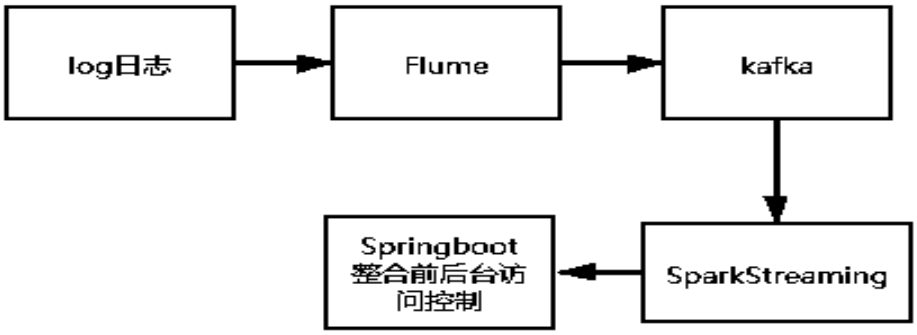

图 1 项目架构图

2.1.2Spark 和 Hadoop

对用户的评分进行实时采集，并且过滤掉无用数据，结合商品的相似度矩阵进行实时分析，得到商品的推荐优先级，并与上一次实时推荐列表进行合并、替换，得到新的实时推荐列表。[3]

通过 Spark 框架对大数据局部频繁项集筛选结果进行分析。经过筛选后的大数据局部频繁项集会在 Spark 框架的储存空间得到保存。[4] 当这些大数据局部频繁项集趋于稳定后，根据本文的算法设计，采用 Spark 技术对这些数据进行存储，对空间中的大数据局部频繁项集进行重新排版和分析。[5]

MapReduce 以 Hadoop 的 MapReduce（distributed computing framework）为基础，在程序执行中遵循线性数据流，先执行 Mapper，后执行 Reducer。[6] 在每次 Mapper-Reducer 周期结束时，将中间结果写入磁盘文件系统。[7] 但它只有 map 和 reduce 两个算子，在大数据盛行的时代，稍微复杂的任务就需要构建多个 job 来执行，当存在 job 依赖的时候，job 之间的数据需要输出到 HDFS 上，所以有 IO 瓶颈。

所以本文设计的是 Spark（基于内存的分布式计算框架），Spark 是一个执行引擎，并非所有的数据都存放在内存，基于内存所以速度很快，而本文则是采用 Spark RDD（弹性分布式数据集），实验中用到的 API 有：flatMap，filter，map，reduceByKey 等。Spark 的经营模式包括 local 经营模式、Standalone 经营模式、YARNClient 经营模式和 YARNCluster 经营模式，不管

是在何种运行模式下，其都要经过 Spark 应用程序的整体运行。[8]

每个 Spark 集群的主节点将资源和任务分配给工作节点，用户定义的应用程序包含在集群上启动不同操作的驱动程序，驱动程序创建 SparkContext 对象以访问计算集群并执行各种 RDD 操作。[9]

2.1.3 Spark Streaming 编程模型

SparkStreaming 是一个分布式处理框架。它有容错性强、吞吐量高等特点。可以使用它里面的 API 函数处理数据。最后结果能存放于数据库、文件系统或者 Dashboard 中。

Streaming 将数据按时间片分段得到批量数据。每一段数据都在 Spark 转换成 RDD，然后把 DStream 人在 Spark 的 Transformation 行动 Streaming 转换成 RDD 在 Spark 的 Transformation 行动，RDD 通过运算转换成中间结果保存在内存中。[10] 在 Spark 端获得数据后，系通过数据的聚合、传输、过滤以及整合后，将最终实验结果展示于控制台。

2.2 数据架构的设计

2.2.1 基于 flume 实时读取数据

Flume 基于流式架构，灵活简单，是实时读取服务器本地磁盘的数据写入到 HDFS 中的首选方案。[11] 主要工作流程为：Source 采集数据并包装成 Event，并将 Event 缓存在 Channel 中，Sink 不断地从 Channel 获取 Event，并解决成数据，最终将数据写入到 HDFS 上。

2.2.2 基于 Kafka 集群的数据传输

我们要建立一个信息系统，实时采集电商平台的用户行为数据流，应用日志的方式去存放，但由于日志的数据流量非常巨大，如何确保数据的可靠性和完整性就成了关键的问题。系统置入应用在日志监控方面能够对于日志文件发生的变化情况进行实时监测，同时还可以根据偏移量来读取最新的日志信息，最后将日志进行缓存。[12]

选择 Kafka 是因为其构建在 Zookeeper 同步服务上，与 Apache Storm 和 Spark 非常好的集成，用于实时流式数据分析。Kafka 的工作流程为：将数据以分区的方式存放于 Brokers 上，例如 Producers 生产了 2 条消息，Topics 有两个分区，那么这两条消息将分别存放于一个 Partition 中，用来保持负载平衡。Topics 将 offset 发送给 Consumers，同时将 offset 存储到 Zookeeper 里。消费者以特定的间隔来请求数据，消费者处理完数据会向代理发送一个该数据已被处理的反馈，Kafka 收到反馈后，更新分区偏移 offset，同时 Zookeeper 里的 offset 也更新。

3 电商网站数据分析和处理

3.1 项目需求

3.1.1 需求描述

准备的数据当中有日期、用户 ID、Session ID（事件 ID）、页面 ID、时间戳、搜索关键词、点击行为、下单行为、支付行为和城市 ID。我们根据每个类别的点击行为、订单行为和支付行为的总数来统计热点类别。

本次项目需求 1 的需求是：处理前数据格式为（品类 ID，点击数，下单数，支付数），处理后变成（品类 ID，综合排名）。点击数量和降序 *0.2 + 下单数量和降序 *0.3 + 支付数量和降序 *0.5 作为综合排名。

同时，处理（品类 ID，点击数，下单数，支付数）数据时，先按照点击数排名，靠前的就排名高；如果点击数相同,再比较下单数；下单数再相同,就比较支付数。

需求 2 为：热门品类中每个品类的 Top10 活跃 Session 统计。

在需求 1 的基础上，增加每个品类用户 session 的点击统计（每个品类有多少用户点击）。

3.1.2 数据准备

数据文件为 user_visit_action.txt；从数据文件中截取一部分内容，表示为电商网站的用户行为数据，中要包含用户的四种行为，搜索、点击、下单、支付。

并设计出数据规则：

（1）数据文件中每行数据采用下划线分隔数据；

（2）每一行数据表示用户的一次行为，这个行为只能是 4 种行为的一种；

（3）如果搜索关键字为 null，表示数据不是搜索数据；

（4）如果点击的品类 ID 和产品 ID 为 -1，表示数据不是点击数据；

（5）针对于下单行为，一次可以下单多个商品，所以品类 ID 和产品 ID 可以是多个，id 之间采用逗号分隔，如果本次不是下单行为，则数据采用 null 表示；

（6）支付行为和下单行为相似。

具体的数据说明如图 2 所示。

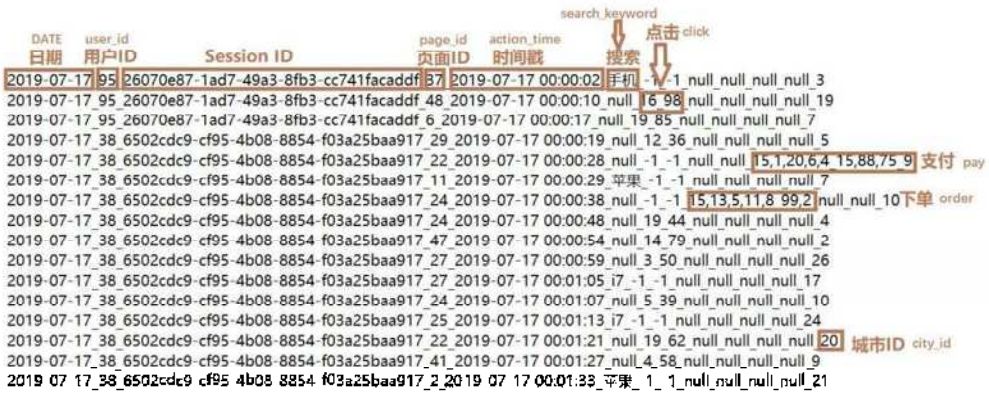

图 2 原始数据说明

3.1.3 数据流分析

（1）获取原始数据；

（2）将原始数据进行结构转化，方便统计。例如：（（省份，广告），1）；

（3）将数据进行分组聚合。例如：（（省份，广告），sum）；

（4）将聚合的结果进行结构转换。例如：（省份，（广告，sum））；

（5）将结构转换后的数据根据省份进行分组。例如：（省份，[（广告 A，sum），（广告 B，sum）]）；

（6）将分组后的数据组内根据数据排序（降序）；

（7）采集数据打印到控制台。

具体数据流分析如图 3 所示。

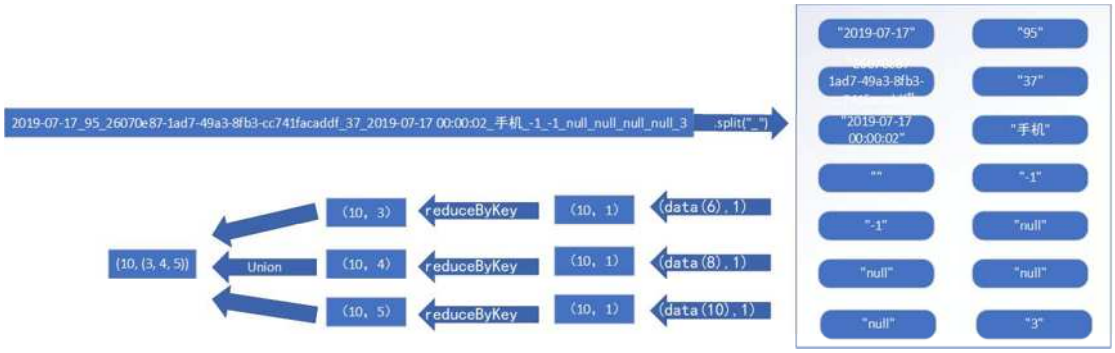

图 3 数据流分析

3.2 项目功能实现

3.2.1 按需求统计出电商平台 TOP10 热门品类

（1）首先读取文件，获取原始数据；

```
// 1.读文件
val actionRDD: RDD[String] = sc.textFile( path = "data/user_visit_action.txt")
```

（2）对点击数据的处理

使用 RDD.filter 过滤掉[点击品类 ID]为“-1”的数据，再处理为（品类,1）的格式。最后用 reduceByKey 将数据聚合为（品类，点击数量）；

```
val clickCountRDD: RDD[(String, Int)] = clickRDD.map(
  action => {
    val datas: Array[String] = action.split( regex = "_")
    (datas(6), 1)
  }
).reduceByKey(_ + _)
```

（3）对下单数据的处理

使用 RDD.filter 过滤掉[下单品类 ID]为“null”的数据，将其处理为（下单品类 1，下单品类 2）的格式，再处理为（品类,1），最后通过 reduceByKey 聚合为（品类，下单数量）；

```
val orderCountRDD: RDD[(String, Int)] = orderRDD.flatMap(
  action => {
    val datas: Array[String] = action.split( regex = "_")
    // (datas(8),1) ?
    val cids: Array[String] = datas(8).split( regex = ",")
    cids.map(id => (id, 1))
  }
).reduceByKey(_ + _)
```

（4）对支付数据的处理

使用 RDD.filter 过滤掉[支付品类 ID]为“null”的数据，接着将处理为（支付品类 1，支付品类 2，…，支付品类 n）的形式，然后处理为（品类,1），最后用 reduceByKey 聚合为（品类，支付数量）；

```
val payCountRDD: RDD[(String, Int)] = payRDD.flatMap(
  action => {
    val datas: Array[String] = action.split( regex = "_")
    val cids: Array[String] = datas(10).split( regex = ",")
    cids.map(id => (id, 1))
  }
).reduceByKey(_ + _)
```

（5）排序

先使用RDD.mapValues用for循环将（品类ID,（点击数量，下单数量，支付数量））的Value分别求和，变成（品类ID,（点击数量和，下单数量和，支付数量和））；

接着使用RDD.sortBy将（品类ID,（点击数量和，下单数量和，支付数量和））变成元祖排序，使用false降序排列；

最后使用RDD.map将排好的处理为（品类ID，点击数量和降序*0.2 + 下单数量和降序*0.3 + 支付数量和降序*0.5）；

```
val resultSortRDD: Array[(String, Double)] = sortRDD.map(x => {
  (x._1, x._2._1 * 0.2 + x._2._2 * 0.3 + x._2._3 * 0.5)
}).sortBy(_._2, ascending = false).take( num = 10)
```

（6）结果为：（品类ID，点击数量，下单数量，支付数量）。

3.2.2 按需求得TOP10热门品类的活跃Session统计

（1）RDD.map取到需求1已经求得的前10品类ID，然后用RDD.contains取所有点击品类当中非空前10的品类ID；

```
val top10AllInfoRDD: RDD[String] = actionRDD.filter(
  action => {
    val datas: Array[String] = action.split( regex = "_")
    val RDD1: Array[String] = resultSortRDD.map(_._1)
    RDD1.contains(datas(6))
  }
)
```

（2）RDD.map直接将（前10点击品类ID）改为（（前10点击品类ID,sessionID),1）；

```
val mapRDD2: RDD[((String, String), Int)] = top10AllInfoRDD.map(
  action => {
    val datas: Array[String] = action.split( regex = "_")
    ((datas(6), datas(2)), 1)
  }
)
```

（3）RDD.reduceByKey将（（前10点击的品类ID,sessionID),1）改为（（前10点击的品类ID,sessionID)，点击数量和）；

```
val reduceRDD2: RDD[((String, String), Int)] = mapRDD2.reduceByKey(_ + _)
```

（4）RDD.map将（（前10点击的品类ID，sessionID)，点击数量和）改为（前10点击的品类,(sessionID，点击数量和））；

```
val countRDD2: RDD[(String, (String, Int))] = reduceRDD2.map({
  case ((category, session), sum) => (category, (session, sum))
})
```

（5）RDD.groupByKey将（前10点击的品类,(sessionID，点击数量和））分组变为（前10点击的品类，[(session1，点击数量和1),(session2，点击数量和2),…；

```
val sortRDD2: RDD[(String, List[(String, Int)])] = groupRDD2.mapValues(
  x=>{
    x.toList.sortBy(_._2)(Ordering.Int.reverse).take(10)
  }
)
```

结语

本文以互联网中的电商平台作为研究背景，以真实的数据作为依据，将 Spark 框架运用到对大数据的处理和分析，摒弃了传统的对软件和硬件的依赖，再加上对代码的优化处理和与 Kafka 集群的集成，合理运用框架内的算子可以精确地计算出对应的商品和用户之间的密切联系以便实施精准推送，项目实验结果也表明算法的实现更简洁、准确，可以更好地服务双方并且节省了不必要的时间。

【参考文献】

[1] 王黎，吕殿基．基于 Spark 框架的大数据局部频繁项集挖掘算法设计 [J]. 微型电脑应用，2021,37(04):130-132+136.

[2] 李雄，文开福，钟小明，杨辉，秦德浩．基于深度学习的人脸识别考勤管理系统开发 [J]. 实验室研究与探索，2019,38(07):115-118+123.

[3] 吴飞贤，段华斌，扈乐华，朱珍珠，宋均．基于 Spark 的商品推荐系统的设计与实现 [J]. 办公自动化，2021,26(03):60-62.

[4] 任卓明，邵凤，刘建国，郭强，汪秉宏．基于度与集聚系数的网络节点重要性度量方法研究 [J]. 物理学报，2013,62(12):522-526.

[5] 宫潘威，时小磊，陶高周，董玉德．CREO 环境下三维模型及信息集成共享实现方法研究 [J]. 机械设计与制造，2019(02):218-221.

[6] 邵梁，何星舟，尚俊娜．基于 Spark 框架的 FP-Growth 大数据频繁项集挖掘算法 [J]. 计算机应用研究，2018,35(10):2932-2935.

[7] 常涛． 改进型 MapReduce 框架的研究与设计 [D]. 北京邮电大学，2011.

[8] 严哲，周斌雄，张祥燊，吴君雄．Spark 计算框架在敏感地理信息检测中的应用研究 [J]. 江西测绘，2021(01):46-49.

[9] 邵梁，何星舟，尚俊娜．基于 Spark 框架的 FP-Growth 大数据频繁项集挖掘算法 [J]. 计算机应用研究，2018,35(10):2932-2935.

[10] 陈静． 基于 Spark 的蜜罐系统的设计与实现 [D]. 西安科技大学，2017.

[11] 贾翼． 基于 Hive 的电商多维分析系统的设计与实现 [D]. 浙江工业大学，2020.

[12] 蒋丛萃，陈巧灵．基于 Spark 平台的电子商务实时推荐系统建设和应用 [J]. 电子商务，2020(11):65-66+94.