计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(5):159−166 [doi: 10.15888/j.cnki.csa.007384]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Hadoop 的高速公路 OD 数据存储模型和计算方法①

何家辉, 赵卓峰, 张　宽, 张　智

[1](北方工业大学 数据工程研究院, 北京 100144)
[2](大规模流数据集成与分析技术北京市重点实验室, 北京 100144)
通讯作者: 何家辉, E-mail: 360963674@qq.com

摘　要: 高速公路 OD 数据是高速公路运营管理和状况分析的一类重要数据, 如何利用海量的收费数据快速生成并有效管理高速公路 OD 数据是当前高速公路智能化建设中的一个重要问题. 针对高速公路 OD 数据的种类多、周期长等问题, 提出一种基于 Hadoop 的高速公路 OD 矩阵存储模型和相应的计算方法. 建立统计高速公路车辆旅行时间、统计高速公路车流量两类 OD 矩阵作为存储模型. 通过基于海量真实的高速公路收费数据的实验和传统的存储高速公路收费数据对比表明本文的存储方法相对于传统的关系数据存储, 拥有更高效的存储效率并节省了存储空间.

关键词: OD 矩阵; 存储模型; MapReduce

引用格式: 何家辉,赵卓峰,张宽,张智.基于 Hadoop 的高速公路 OD 数据存储模型和计算方法.计算机系统应用,2020,29(5):159–166. http://www.c-s-a.org.cn/1003-3254/7384.html

Highway OD Data Storage Model and Calculation Method Based on Hadoop

HE Jia-Hui, ZHAO Zhuo-Feng, ZHANG Kuan, ZHANG Zhi

[1](Institute of Data Engineering, North China University of Technology, Beijing 100144, China)
[2](Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, Beijing 100144, China)

Abstract: Highway OD data is a kind of important data for highway operation management and condition analysis. How to use massive toll data to quickly generate and effectively manage highway OD data is an important problem in the current highway intelligent construction. Aiming at the problems of various types and long periods of highway OD data, a storage model of highway OD matrix based on Hadoop and corresponding calculation method are proposed. Two kinds of OD matrices are established as storage models, i.e. statistics of highway vehicle travel time and statistics of highway traffic flow. The comparison between the experiment based on massive real highway toll data and the traditional storage of highway toll data shows that the storage method proposed has better storage efficiency and saves storage space compared with the traditional relational data storage.

Key words: OD matrix; storage model; MapReduce

1　引言

高速 (Origin-Destination, OD) 数据是指高速路网内从特定的起点到终点之间出行信息的相关数据, 如起点到终点之间的通行流量、旅行时间等数据. 高速 OD 数据中包含了高速公路整体路网的运行状态、高速公路出行特征、车流量的时空分布等多方面重要信息, 对高速公路的规划、管理、维护及养护、衡量区域经济发展状况及地区间的社会经济交流具有重要的参考意义.

传统高速公路 OD 数据的获取方法大多采用定期

① 收稿时间: 2019-09-26; 修改时间: 2019-10-22; 采用时间: 2019-11-05; csa 在线出版时间: 2020-05-07

调查的方式, 通过特定的数据收集方式 (甚至人工) 得到一定时期内的高速 OD 数据, 往往需要耗费大量时间和人力, 而且数据的精度难以保证[1−3]. 此外, 一些研究人员在高速公路断面通行量检测系统基础上通过广义最小二乘法估算的方法来估计时变的 OD 数据[4]. 高速公路收费数据由于需要根据车辆进出收费站信息来收取费用, 其中蕴含了 OD 分析所需要的时空信息, 而且收费数据十分准确. 因此, 相对于前述方法, 其可用来生成更加精细化、更加全面的高速公路 OD 数据.

高速公路收费数据中包含了特定车辆进入某收费站的时间和离开另一收费站的时间, 可以看作是一种收费站点之间天然的 OD 数据. 基于高速公路收费数据可以计算得到收费站点之间在任意时间周期内的出行量、旅行时间等 OD 数据, 也可以针对不同车辆类型的细分得到各类车辆独自的 OD 数据. 然而, 由于高速公路出行的不断增加, 高速公路收费数据规模也越来越大, 一个大省的收费数据一个月可达到数亿条, 而收费站点也接近 1000 个, 意味着由此建立的一个 OD 矩阵有百万数据. 如果按照 1 小时的时间周期建立 OD 数据矩阵, 并且针对出行量、旅行时间等不同指标和不同车辆类型分别统计, 则一天需要生成数百个 OD 数据矩阵. 因此, 无论是在 OD 数据矩阵的计算方面还是 OD 数据矩阵的存储方面都对高速公路信息系统建设带来了极大的挑战. 张晶等提出, 处理海量数据, 必然要面对巨大的计算量, 一般的解决方法是采用分布式计算, 将计算任务分配到多台机器上并行处理, 从而提高运算速度[5].

以 Hadoop 为代表的大数据技术的出现为高速公路 OD 数据的计算和存储提供了新的技术思路, Hadoop 是一个在集群上处理海量数据的开源代码框架, 它具有分割大规模数据、合理分配任务并执行的特性, 在很多大型网站上已经得到了应用, 是目前最广泛应用的开源云计算平台[6,7]. 本文将首先给出基于收费数据的高速公路 OD 数据逻辑表示模型和计算方式定义, 在此基础上进而研究了基于 MapReduce 的统计车辆平均旅行时间和统计车流量的 2 种 OD 矩阵生成算法和基于 HBase 的高速公路 OD 数据矩阵列式存储模型, 其中, 统计车辆平均旅行时间的 OD 矩阵可以掌握高速公路车辆的行驶情况, 统计车流量的 OD 矩阵可以更加清晰的反应出该路段的拥堵情况, 通过实验结果对比得出结论, 给出总结.

2 高速公路 OD 矩阵定义

2.1 定义

定义 1. 收费站 s, 高速公路上所有的收费站所在位置, 定义为 $S=(s_1, s_2, \cdots, s_n)$, $s=(id, type)$, $s \in S$, 其中 id 是收费站的编号, $type$ 为收费站类别, 进站口 $type$ 的值为 0, 出站口 $type$ 的值为 1.

定义 2. 车辆收费数据 L, 车辆收费数据是指高速公路各个收费站点捕获的车辆收费数据, 定义为 $L=\{l_1, l_2, \cdots, l_n\}$, $l=(v_i, s_k^i)$, 其中v_i代表车辆的车牌号码, s_k^i表示车辆 v_i 经过收费站点s_k的时间, s_k表示车辆v_i经过的收费站点.

定义 3. 车辆类别 $VType$, 车辆分为多种类别, 如大货车, 大客车, 小汽车等, 不同类型的车辆使用不同类型的编号.

定义 4. 车辆 OD 矩阵 D, 高速公路上的车辆进站点和出站点为行和列构成的矩阵, 定义为 $D=(d_1, d_2, d_3, \cdots, d_n)$, 这里每一个$d_m$可表示为$d_m=(a_{11}^m, a_{12}^m, a_{13}^m, \cdots, a_{ij}^m, \cdots, a_{nn}^m)$, 其中 m 表示时间段区间 (t_a, t_b), 矩阵每一个元素的值a_{ij}为车辆的出行时间、速度、数量等对车辆的不同度量, 体现为不同形式的值.

基于以上定义, 将高速公路 OD 矩阵划分为静态 OD 矩阵和动态 OD 矩阵, 其中, 静态 OD 矩阵的元素 a_{ij}代表某段时间以收费站点s_i为起点, s_j为终点的车辆交通出行量; 而动态 OD 矩阵是一个矩阵序列, 与时间的相关性远大于静态 OD 矩阵, 动态 OD 矩阵的每一个元素a_{ij}^m表示以收费站点s_i为起点, s_j为终点在时间段 m 内的交通出行量.

2.2 OD 矩阵生成规则

高速公路 OD 数据最明显的内容即为车辆驶入驶出高速的时间和地点, 为了能够明显的反应高速公路车辆行驶情况, 将收费数据生成几类 OD 矩阵, 例如统计高速公路车辆旅行时间的 OD 矩阵, 统计车流量的 OD 矩阵, 统计两站点间车流量和平均旅行时间的 OD 矩阵, 统计不同车型的行驶速度的 OD 矩阵等, 本文中主要研究其中两类, 第一类为统计高速公路所有类型车辆旅行时间的 OD 矩阵; 第二类为统计高速公路所有类型车流量的 OD 矩阵. 其中各参数的含义如下:

(1) 统计量 (statistics): 该 OD 矩阵对应的表统计的指标, 即高速公路所有车型的平均旅行时间和车流量.

(2) 时间区间 (period): 用于表明统计对象的时间属性, 一个时间区间由一个开始时间点 (startInstant) 和一个结束时间点 (endInstant) 构成.

(3) 进站点 (stationIN): 车辆驶入高速的站点位置编号.

(4) 出站点 (stationOUT): 车辆驶出高速的站点位置编号.

(5) 值 (value): 度量统计量的值, 例如 2493 秒 (旅行时间 OD 矩阵) 或 479 辆 (车流量 OD 矩阵).

2.2.1 统计高速公路车辆旅行时间 OD 矩阵

该矩阵是统计一定时间内出站车辆与其进站点间的平均旅行时间. 该 OD 矩阵计算所需的车辆信息要通过数据中的出站时间进行筛选. 在计算平均时间前需要将符合时间的车辆收费数据过滤掉车牌号等无效信息, 保留进出站时间. 平均旅行时间需要把所有进出站点相同的车辆旅行时间作平均计算. 将平均旅行时间放入 OD 矩阵中, 便于更直观、更方便的掌握高速公路整体路网的运行状态, 便于对高速公路进行调控. 统计高速公路车辆旅行时间的 OD 矩阵的逻辑模型如图 1 所示.

Statistics: traveling time for all types of vehicle		
Period: startInstant=2018/4/29 23:00:00 endInstant=2018/4/30 00:00:00		
StationIN	StationOUT	Value
4	106	2561
22	107	2493
3	106	3065
9	107	4852

图 1 旅行时间 OD 矩阵逻辑模型

统计旅行时间 OD 矩阵的计算流程如图 2 所示.

第 1 步. 设置统计数据的时间段.

第 2 步. 读取一条数据 根据统计的时间间隔判断该数据的出站时间是否在该时间间隔内.

第 3 步. 如果是在时间间隔内, 查找该数据对应的车辆的进出站时间和进出站 ID. 否则判断是否有未遍历到的数据, 若仍有未遍历到的数据, 则读取另一条数据, 否则重新设置统计数据的时间段.

第 4 步. 取出站时间满足统计时间间隔的完整数据, 用进出站时间计算该条数据的旅行时间.

第 5 步. 找到该路线所有车辆的旅行时间并求平均值, 将旅行时间平均值放入 OD 矩阵中.

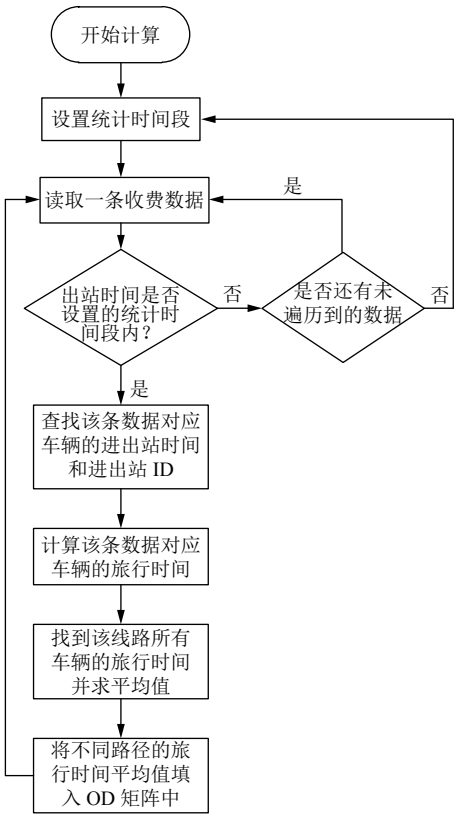

图 2 统计车辆旅行时间 OD 矩阵的计算流程

2.2.2 统计高速公路车流量 OD 矩阵

该矩阵是统计一段时间间隔内两进出点间车辆数目. 该 OD 矩阵计算所需的的车辆信息需要过滤掉收费信息中的员工编号、进站栏位等等无效信息, 再依据数据中的进出站时间筛选. 该 OD 矩阵需要的车流量信息需要通过依据进出站点名称筛选统计两进出站点间流量. 将流量放进 OD 矩阵中能够更加清晰、直观的反应两进出站口间车流量情况, 车辆拥堵状况, 方便及时对高速车流量进行调控, 减少拥堵情况, 降低事故发生机率. 统计高速公路车流量的 OD 矩阵的逻辑模型如图 3 所示.

统计车流量 OD 矩阵的计算流程如图 4 所示.

Statistics: traffic flow for all types of vehicle		
Period: startInstant=2018/4/29 23:00:00 endInstant=2018/4/30 00:00:00		
StationIN	StationOUT	Value
4	106	374
22	107	671
3	106	461
9	107	739

图 3 车流量 OD 矩阵逻辑模型

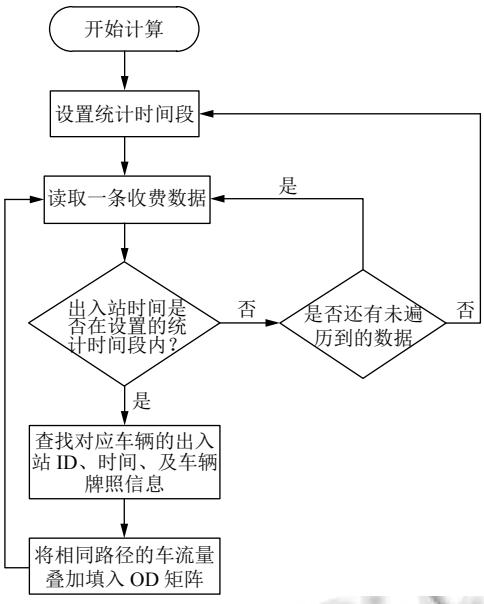

图 4 统计车流量 OD 矩阵的计算流程

第 1 步. 设置统计数据的时间间隔.

第 2 步. 读取一条数据, 根据统计的时间间隔判断该数据的出入站时间是否在该时间间隔内.

第 3 步. 如果是在时间间隔内, 查找该数据对应的车辆的进出站 ID 和时间、车辆牌照信息. 否则判断是否有未遍历到的数据, 若仍有未遍历到的数据, 则读取另一条数据, 否则重新设置统计数据的时间段.

第 4 步. 取进出站时间满足统计时间间隔的完整数据, 将相同路径的车流量叠加, 最终车流量放入流量 OD 矩阵中.

3 基于 HBase 的 OD 数据存储模型

HBase 是 Hadoop 家族中的 NoSQL 数据库, 是一个高可靠性、高性能、列存储、可伸缩、支持实时读写、面向联机事物处理、分布式的大型数据存储系统. 底层的 Hadoop 分布式文件系统 HDFS 具有高容错性且可以被部署在低价的硬件设备之上[8,9].

HBase 表主要由以下几部分构成:

表 (table): HBASE 在表中组织数据, 表名是字符串和字符的组合.

行 (row): 表中数据按水平划分为多行, 每行由唯一的 rowkey 确定.

行健 (rowkey): 是行的唯一标识数据, 没有数据类型, 一般是一个字节数组.

列族 (Column Family, CF): 数据在竖直方向划分成多个列族, 列族要预先定义, 且不容易修改, 每行数据都拥有相同的列族, 但是可能有些行的数据为空, 列族是字符串和字符的组合.

列限定符 (Column Qualifier, CQ, 也叫列标识): 数据在列族中的位置是通过列标识指定的, 每行的列标识可以不同, 列标识没有数据类型, 一般是一个字节数组.

单元 (cell): 单元是行健 (rowkey)、列族 (column family)、列限定符 (column qualifier) 的组合, 这些存储在单元中的数据被称为单元数据.

时间戳 (timestamp): 单元数据是有版本的, 每一个单元数据对应一个版本, 由时间戳来指定.

统计高速公路车辆旅行时间的 OD 矩阵在 HBASE 中存储模型如表 1 所示. 将设置的时间段、进站 ID、出站 ID 作为 RowKey, 将车辆的平均旅行时间作为 value.

表 1 车辆旅行时间 OD 矩阵物理模型

参数	类型
时间段	row, key
进站 ID	row, key
出站 ID	row, key
CF	key
CQ	key
平均旅行时间	value

统计高速公路车流量的 OD 矩阵物理模型如表 2 所示, 将设置的时间段、进站 ID、出站 ID 作为 rowkey, 将车流量作为 value.

表 2 车流量 OD 矩阵物理模型

参数	类型
时间段	row, key
进站 ID	row, key
出站 ID	row, key
CF	key
CQ	key
车流量	value

4 OD 矩阵计算在大数据环境下的实现

4.1 旅行时间 OD 矩阵计算方法在 MapReduce 下的实现

MapReduce 计算框架由 Input, Map, Shuffle, Reduce, Output 5 个部分组成.

Input 阶段: 读入数据并设置统计时间段.

Map 阶段: 读取所有输入数据, 对每行数据进行解析. 以“出站时间+进站点+出站点”作为 key, 计算旅行时间作为 value.

Shuffle 阶段: key 值相同的 value 会存入一组, 传送到 Reduce.

Reduce 阶段: 针对每个 key, 统计 value 的个数 count, 并计算 value 和 sum, 计算 value 均值 sum/count 作为新输出 value.

Output 阶段: 以 key-value 的方式输出计算结果, 最终的 key 为“出站时间+进站点+出站点”, value 为“平均旅行时间”.

该算法实现的流程如图 5 所示, 伪代码如算法 1 所示.

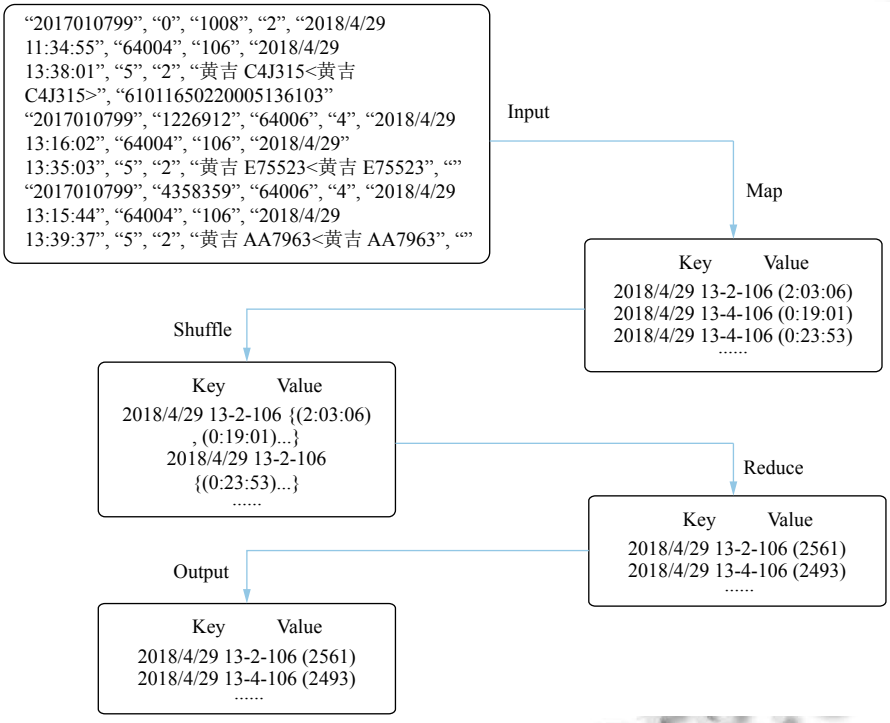

图 5 旅行时间 OD 矩阵实现流程

算法 1. 旅行时间 OD 矩阵计算方法

输入: 高速流量数据
输出: 旅行时间 OD 统计信息

```
1. function MAP(key, value, context)
2.    items = value.split("r");
3.    timeIn = items[4];
4.    timeOut = items[7];
5.    key = items[7].split(":")[0] + items[3] + items[6];
6.    odTime = timeOut – timeIn;
7.    context.write(key, odTime);
8. end function
9. function REDUCE(key, Iterable<ODTime>, context)
10.    count = 0;
11.    totaltime = 0;
12.    avg = 0;
13.    for ODTime TIME: L do
14.       totaltime = totaltime + TIME.getTOTALTIME();
15.       count + +;
16.    end for
17.    avg = totaltime/count;
18.    context.write(key, avg);
19. end function
```

4.2 车流量 OD 矩阵计算方法在 MapReduce 下的实现

车流量 OD 矩阵计算方法的实现与旅行时间类似. 主要区别在 Reduce 阶段, 需要对进出站的时间进行过滤.

Input 阶段: 读入数据并设置统计时间段.

Map 阶段: 读取所有输入数据, 对每行数据进行解析. 以“出站时间+进站点+出站点”作为 key, 旅行时间

作为 value.

Shuffle 阶段: key 值相同的 value 会存入一组, 传送到 Reduce.

Reduce 阶段: 针对每个 key, 统计 value 的个数 count, count 作为新输出 value.

Output 阶段: 以 key-value 的方式输出计算结果, 最终的 key 为“出站时间+进站点+出站点”, value 为“车流量数”.

该计算方法的流程如图 6 所示, 伪代码如算法 2 所示.

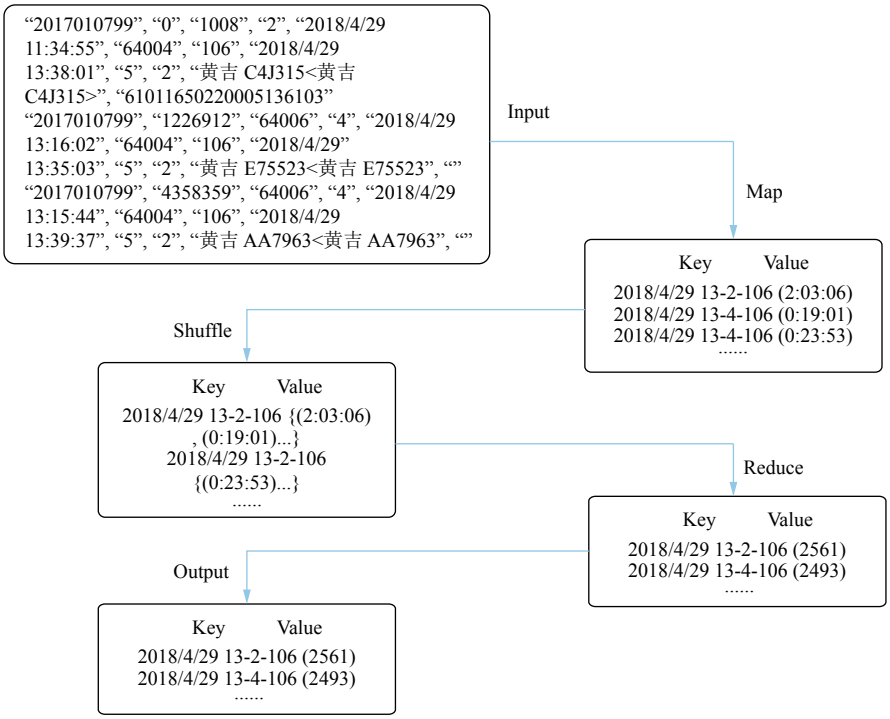

图 6 旅行时间 OD 矩阵实现流程伪代码

算法 2. 车流量 OD 矩阵计算方法

输入: 高速流量数据

输出: 车流量 OD 统计信息

```
1. function MAP(key, value, context)
2.   items = value.split(",");
3.   timeIn = items[4];
4.   timeOut = items[7];
5.   key = items[7].split(",")[0] + items[3]+ items[6];
6.   odTime = timeOut – timeIn;
7.   context.write(key, odTime);
8. end function
9. function REDUCE(key, Iterable<ODTime>, context)
10.   count = 0;
11.   totaltime = 0;
12.   for ODTimo TIME : L do
13.     totaltime = totaltime + TIME .getTOTALTIME();
14.     count + +;
15.   end for
16.   context.write(key, count);
17. end function
```

5 实验评价

5.1 实验环境

本文中生成 OD 矩阵所需的高速公路的收费数据数据量庞大, 但是对实时性要求不高, 所以选用在 Hadoop 集群环境下使用 MapReduce 分布式离线计算框架进行实验. 集群包含 3 个节点, 其中 1 个主节点, 2 个从节点, 每个节点的 CPU 都为 4 核, 运行内存 16 GB, 主机硬盘 2 TB.

5.2 实验数据

实验数据为河南省高速 2018 年的真实收费数据. 该数据记录了 300 余个高速公路出入口, 全年的车辆出入情况, 共计 10 743 万余条记录. 每条数据的记录形式如表 3 所示, 包含车牌号、驶入站点、驶出站点、驶入时间、驶出时间等 12 个属性.

为了方便进行实验和性能评估, 需要对数据进行过滤和清洗. 数据中最核心的属性就是车牌号, 所以按照标准的车辆牌照字符要求: 正规的车辆牌照应该为

7 位有效字符, 第一位为省份汉字简称, 第二位为大写英文字母形式的省内区域代号, 后五位应该是大写英文字母和数字的组合. 根据上述要求, 可能出现的不规范数据问题有如下几项:

(1) 车辆牌照第一个字符不是省份汉字简称.

(2) 省份汉字简称多余 1 个.

(3) 车辆牌照中出现非法字符, 如: 小写字母, 特殊符号等.

(4) 车辆牌照位数不足.

(5) 无牌照信息.

包含上述任何一条的数据将被认为是不符合要求的数据, 将这些数据过滤掉. 本文中提到的实验只需要各种类车辆在时间和空间上的信息, 所以将过滤后的数据进行进一步的压缩, 只保留车牌号、驶入站点、驶出站点、驶入时间、驶出时间和车辆类型这六个属性, 其他属性对于本次实验来说并非必要数据, 所以将其余属性去除以保证在程序运行和计算过程没有额外的不必要消耗, 保证程序运行性能.

表 3 数据属性

属性名称	数据类型	含义
CARDID	VARCHAR	MTC 卡编号
CARDNETWORK	VARCHAR	员工编号
ENTRYLANE	VARCHAR	进站栏位
ENTRYSTATION	VARCHAR	进站站点 ID
ENTRYTIME	VARCHAR	进站时间
ETCCPUID	VARCHAR	ETC 卡编号
EXITLANE	VARCHAR	出站栏位
EXITSTATION	VARCHAR	出站站点 ID
EXITTIME	VARCHAR	出站时间
VEHICLECLASS	VARCHAR	车辆种类
VEHICLELICENSE	VARCHAR	车牌号
VEHICLETYPE	VARCHAR	车辆类型

5.3 计算模型评价

实验对生成旅行时间 OD 矩阵和车流量 OD 矩阵所用时间进行了评价. 在相同的硬件条件下, 控制输入时数据的规模, 统计计算所用时间.

5.3.1 旅行时间 OD 矩阵

实验测试不同的统计天数统计车辆平均旅行时间的 OD 矩阵计算效率如表 4 所示, 实验证明, 数据量越大的情况下, 计算效率越高.

5.3.2 车流量 OD 矩阵

实验测试不同的统计天数统计车流量的 OD 矩阵计算效率如表 5 所示, 实验证明, 当数据量增大时, 计算效率增高.

表 4 不同数据量的旅行时间 OD 矩阵的计算效率

统计天数 (天)	输入数据条数 (千条)	输出数据条数 (千条)	计算时间 (ms)
7	1658	55	65 754
15	4149	124	70 318
30	8743	249	78 345
90	26 065	562	163 660

表 5 不同数据量的车流量 OD 矩阵的计算效率

统计天数 (天)	输入数据条数 (千条)	输出数据条数 (千条)	计算时间 (ms)
7	1658	56	63 004
15	4149	124	76 405
30	8743	249	79 512
90	26 065	734	174 828

5.4 存储模型评价

对于存储模型的设计, 主要考虑设计的模型占用的物理存储空间大小. 传统的存储方式主要是基于 MySQL 的关系型数据存储模式,这种存储方式不适合存储海量的数据, 需要多个表进行存储才便于对数据的处理, 而本文采用基于 HBase 的列式存储模式, 数据量相对较小时, 两种存储方式差别并不大, 而数据量较大时, 本文设计的存储方式节省物理存储空间. 如表 6 所示, 当存储时间长度为 7 天的数据时, 本文设计的存储方式只比传统存储方法节省 3% 的物理存储空间, 当存储时间长度为一年的数据时, 本文设计的存储方式比传统存储方式节省近 10% 的物理存储空间.

表 6 不同数据量的存储模型占用空间 (单位: MB)

方法	7 天	15 天	30 天	90 天	365 天
传统方法	15.93	40.47	83.09	250.76	1209.84
本文	15.3	39.15	78.58	230.81	1095.79

两种存储方式的对比情况如图 7 所示.

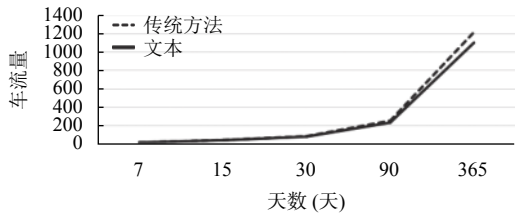

图 7 车流量 OD 矩阵实现流程

6 总结

高速公路经过长时间的发展, 已经形成了比较健全的路网结构, 信息化的建设已经逐步成型, 并覆盖到

高速公路的许多业务领域, 现有的高速公路运营系统对于海量数据的处理效率低下, 本文从大数据分析的角度实现了生成车辆 OD 矩阵的过程, 将生成的矩阵存入 HBase 中, 并对存储的效率和存储空间进行了分析, 以便于后续对生成的 OD 矩阵进行聚类等分析.

参考文献

1 赵明, 王寒凝. 基于车牌照识别技术的 OD 调查系统分析与设计. 公路交通科技: 应用技术版, 2008, (12): 188–190.

2 宋广辉, 刘淑珍, 耿英杰, 等. OD 调查简介. 东北公路, 1996, (2): 77–82.

3 陈满堂. 关于车牌号 OD 调查方法的探讨. 公路, 2004, (9): 86–88. [doi: 10.3969/j.issn.0451-0712.2004.09.020]

4 林勇, 蔡远利, 黄永宣. 高速公路动态 OD 矩阵估计. 长安大学学报 (自然科学版), 2003, 23(6): 83–86.

5 多雪松, 张晶, 高强. 基于 Hadoop 的海量数据管理系统. 微计算机信息, 2010, 26(5-1): 202–204.

6 龙伟, 陈志, 万亚平. 基于 Hadoop 的高速公路联网收费稽查系统. 西部交通科技, 2014, (8): 73–78.

7 Newman A, Li YF, Hunter J. Scalable semantics—The silver lining of cloud computing. Proceedings of the 2008 IEEE 4th International Conference on eScience. Indianapolis, IN, USA. 2008. 111–118.

8 王惟一. 基于存储模型的 HBase 查询优化技术研究[硕士学位论文]. 南京: 南京大学, 2018

9 徐德智, 刘扬, Ahmed S. 基于 Hadoop 的 RDF 数据存储及查询优化. 计算机应用研究, 2017, 34(2): 477–480, 486. [doi: 10.3969/j.issn.1001-3695.2017.02.035]