西安電子科技大學

《云计算技术》大作业

学　　院：计算机科学与技术学院

专　　业：软件工程

方　　向：大数据与云计算

小组编号：第 12 组

小组名称：CCTV6

目录

一、 系统架构

该系统是一个车辆过车信息系统，主要功能是在各个机具实时获取过往车辆信息，并传输至远端，通过大数据分析工具对海量数据进行存储、处理、分析，并最终呈现到 Web 页面上。

在该系统中，我们使用了一系列的技术工具，包括 Apache Kafka、Redis、Hadoop、HBase、Spark、Spring-Boot 等，实现了上述功能。

系统整体架构如下图：

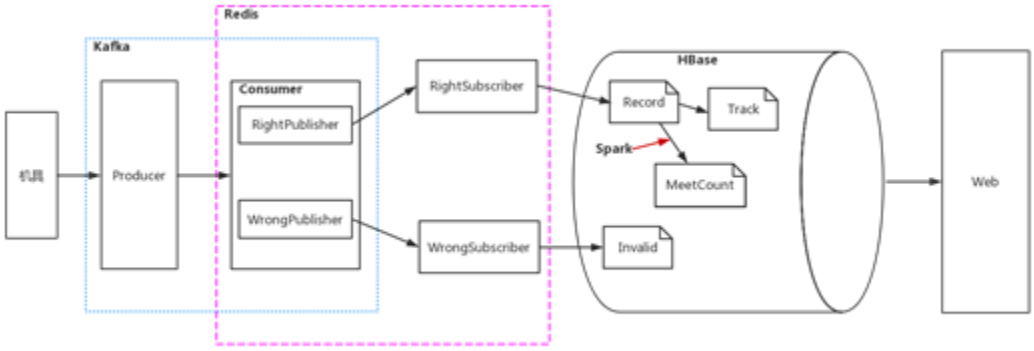

图 1 系统架构图

机具产生数据后，以数据生产者的身份发送至 Apache Kafka 指定的 Topic，在此为 my-topic；然后多个数据消费者订阅该 Topic，读取数据消息，对消息依据规则进行过滤，无效数据和有效数据分别被不同的 Redis 发布者发送到不同的频道里，在此为 right_channel 和 wrong_channel，然后有不同的订阅者订阅不同的频道，对频道中获取的数据进行消费，有效数据和无效数据分别上传至 HBase Record 和 Invalid 两个不同的表中，实现数据正式入库。

数据入库后，因为数据量太大，过车数量统计、车辆相遇次数、车辆轨迹重新统计等功能不适合进行在线处理，采用周期性离线处理的方式。其中，过车数量统计通过 Java 简单读取 Record 表进行数量统计，并写入 VichelCount 表中；车辆相遇次数统计因使用了 Join 操作，使用 Spark 更快地处理；车辆轨迹重现则是通过重现编排行键来完成，因为 HBase 在存储时按照 RowKey 的进行存储，以 eid##timestamp 为行键，可以根据用户需求快速查找到指定的车辆。

同时，为了方便用户使用，Web 提供了分页功能，并对时间进行了格式化展示。

二、 数据流程分析

1. 数据采集过程分析

数据介绍

本实验的源数据模拟了一个地区的所有摄像头在一段时间内拍摄并记录的过车数据，总共约有 56 万条，存储格式为 json。

kafka 数据处理

a) 存在问题

本系统在数据采集端面对的是一个分布式、大批量、实时性的数据集，如果不将数据用某种方法暂时存储，可能会导致：

① 数据库存储请求线程饱和，导致数据丢失。在实验前期，我们没有对 kafka 的 consumer 端发送给 Redis 的数据速率作限制，也会遇到数据丢失的情况。

② 数据无法及时处理，导致数据丢失。这种情况出现在先处理数据后存入数据库的情况，由于计算机性能问题导致来不及处理的数据也会被丢失。

以上均这说明数据的快速获取是对软硬件的极大考验，还需要一些手段来避免这样的问题发生。

b) 数据采集流程

①首先我们需要用 Java 代码实现一个数据发送端来模拟摄像头这个数据发送源。

②创建 kafka 的 topic，同时注册 producer 和 consumer 到该 topic 上。Topic 是将不同的 producer 和 consumer 连接在一起的桥梁，保证数据不会混乱。

③启动模拟数据发送程序和 producer，向 topic 中发送数据。

④启动 consumer 程序，将存储在 topic 的不同 partition（本实验中为 3 个）里的数据取出并传输给 Redis，这个过程可以与 producer 同步进行，也可以异步进行。Partition 存在的原因是可以将数据分布式存储。

⑤在传输过程前会有一个数据过滤程序，将经纬度正常的数据发布给 Redis 的正常信道，有问题的数据则发布给 Redis 另一个专门用于发布错误数据的信道，hbase 部分分别订阅不同的信道，并存储到不同的表中。

流程图如下所示：

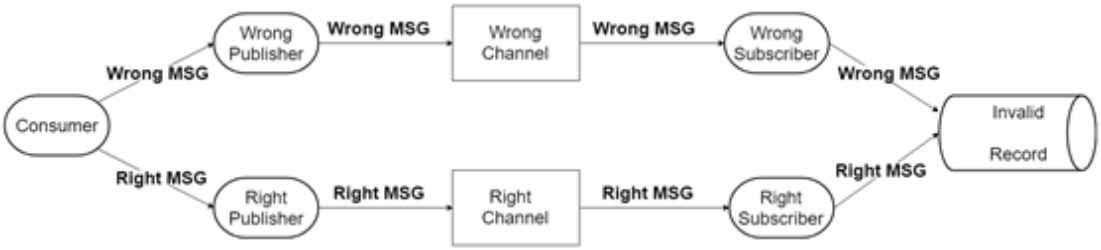

图 2 获取数据流程图

对于数据的过滤，主要由以下步骤完成：

```
tmp = record.split(",");
leng = tmp[tmp.length-2].substring(12).length();
longi = Double.parseDouble(tmp[tmp.length-2].substring(12).substring(0,leng - 2));
leng = tmp[tmp.length-1].substring(11).length();
lati = Double.parseDouble(tmp[tmp.length-1].substring(11).substring(0,leng - 2));
if((longi < 130 && lati < 40))
```

图 3 kafka 数据过滤代码展示

Redis 进行发布订阅

主要方法为：

```
public class MySubThread extends Thread {
        private Jedis myJedis=null;
        private String myChannel="";
        private JedisPubSub myJedisPubSub=null;

        public MySubThread(Jedis jedis, String channel, JedisPubSub jedispubsub) {
                myJedis=jedis;
                myChannel=channel;
                myJedisPubSub=jedispubsub;
        }

        @Override
        public void run() {
                myJedis.subscribe(myJedisPubSub, myChannel);
        }

}
```

图 4 Redis 信道订阅多线程方法

```
ConsumerRecords<String, String> records = consumer.poll(100);
for (ConsumerRecord<String, String> record : records)
{
System.out.printf("offset = %d, key = %s, value = %s%n", record.offset(), r
//                                        MyPubThread rightPubThread=ne
//                                        rightPubThread.start();
jedis.publish(right_channel, record.value());
//pipelineq.sadd(key,record.value());
//record to redis
```

图 5 在 Kafka 的 Consumer 中使用 Redis 发布方法

2. 数据查询和离线处理分析

数据查询

为了实现要求的相关功能，我们在 HBase 中建立了以下几张表

表名	RowKey	列族	列
Record	placeId##time##eid	info	address longitude latitude
Invalid	placeId##time##eid	info	address longitude latitude
MeetCount	meid##oeid	info	count
Track	eid##time	info	placeId address longitude latitude

表 1 HBase 表结构

Record 和 Invalid 表分别用于存储正常数据以及异常数据，MeetCount 表存储的是通过 Spark 进行离线分析后得到的车辆相遇数据，Track 表主要用于辅助实现轨迹重现功能。下面主要对 Track 表的设计思路以及查询方法做一些说明。

Track 表也是经过离线处理得到的，我们将 Record 表中的数据全部读出，重新编排行键，变为以"eid##time"为行键，之后将其插入到 Track 表中。我们知道 HBase 中的数据是按行键字典序进行存储的，这样设计行键可以使同一车辆的数据是连续存储的，并且同一辆车的的数据是按 time 升序存储的。当前端发起一个轨迹重现的请求时，我们根据指定的 eid 和时间范围可以构造出 startRowKey 和 stopRowKey，然后以此为参数进行 scan 操作，这样在查询时，可以直接定位到 startRowKey 所在的 Region 开始扫描，当扫描到的 RowKay 大于等于 stopRowKey 时停止扫描，这样就避免了全表扫描，并

且得到的数据本身就是按 time 升序排列的,所以也无需进行二次排序。综上,建立 Track 表，加快了轨迹重现查表的速度。

上述的方案是基于离线处理的，但是我们也可以把它做成实时的，这时我们只需要在往 Record 表中插数据的同时，以"eid##time"为行键往 Track 表中插一条同样的数据，这样我们就能够实现实时轨迹重现。

离线处理分析

a) 为何选用 Spark 分析数据?

在本次云计算课程使用 Spark 的部分是数据存储与分析，采用 Spark 框架对 HBase 中的数据进行业务分析，并把分析结果存储到 HBase 中。Spark 在该项目中主要的作用是负责统计同一个地点中出现的车辆次数和两辆车的相遇次数统计。由于数据量巨大，仅仅通过简单的编程模型无法更好的完成数据处理任务，而 Spark 这一可伸缩的基于内存计算并且可以直接读写 Hadoop 上任何格式数据的优势，进行批处理时更加高效，并有更低的延迟的计算框架。初次使用，被它的快速感到了震惊同时，又对传统的 MapReduce 和 Spark 进行了对比，发现如下几个特征：

1）基于内存计算，减少低效的磁盘交互；

2）高效的调度算法，基于 DAG；

3)容错机制 Linage，精华部分就是 DAG 和 Lingae

基于以上原因遂使用 Spark 进行数据分析。

b) Spark RDD 在项目中的使用?

在城市车辆智能防控系统这个云计算课程项目中，Spark 的应用中用的最多的便是 Spark RDD 部分。RDD（Resilient Distributed Dataset）叫做弹性分布式数据集，是 Spark 中最基本的数据抽象，它代表一个不可变、可分区、里面的元素可并行计算的集合。RDD 具有数据流模型的特点：自动容错、位置感知性调度和可伸缩性。RDD 允许用户在执行多个查询时显式地将工作集缓存在内存中，后续的查询能够重用工作集，这极大地提升了查询速度。

Spark RDD中算子又极其重要,SparkRDD算子分为两类:Transformation和Action，其中Transformation会延迟加载数据，Transformation会记录元数据信息，当计算任务触发Action时，才会真正开始计算。

城市车辆智能防控系统中使用的算子：

- join()
- filter()
- mapToPair()
- combineByKey()

在本次实验中解决问题最关键的算子是join()个combineByKey(),join()算子的作用和关系数据库中的全链接有异曲同工之妙。而 combineByKey()是一个强大的方法，combineByKey 属于 Key-Value 型算子，做的是聚集操作，这种变换不会触发作业的提交，主要有三个参数，如下：

- combiner function : 一个组合函数，用于将RDD[K,V]中的V转换成一个新的值C1；
- mergeValue function : 合并值函数，将一个C1类型值和一个V类型值合并成一个C2类型，输入参数为(C1,V)，输出为新的C2
- mergeCombiners function : 合并组合器函数，用于将两个C2类型值合并成一个C3类型，输入参数为(C2,C2)，输出为新的C3。

三、 软件功能分析

1. 系统固有功能分析

过车信息

数据写入HBase的Record表后，当用户在Web访问过车信息功能时，通过后端服务读取表中原始数据，并将其展示在前端页面。

过车统计

placeId and EID and Time:yyyy-MM-dd HH:mi 查询

更多

#	EID	PlaceID	Time	Address	Longitude	Latitude
	33040210006258	1	2017-06-01 00:40:06	秀洲公安分局西门段	120.71925	30.77091
	33041100026142	1	2017-06-01 02:31:09	秀洲公安分局西门段	120.71925	30.77091
	33041100026543	1	2017-06-01 03:01:02	秀洲公安分局西门段	120.71925	30.77091
	33041100018856	1	2017-06-01 03:45:27	秀洲公安分局西门段	120.71925	30.77091
	33041100018680	1	2017-06-01 04:14:47	秀洲公安分局西门段	120.71925	30.77091
	33041100004737	1	2017-06-01 05:00:45	秀洲公安分局西门段	120.71925	30.77091
	00002721103586	1	2017-06-01 05:39:53	秀洲公安分局西门段	120.71925	30.77091
	33041100024411	1	2017-06-01 05:51:08	秀洲公安分局西门段	120.71925	30.77091
	33041100020197	1	2017-06-01 06:02:15	秀洲公安分局西门段	120.71925	30.77091
	33040210001356	1	2017-06-01 06:05:22	秀洲公安分局西门段	120.71925	30.77091

图 6 过车信息统计 Web 页

地点过车统计

利用 Spark RDD 处理 Hbase 中的海量数据使用 combineByKey()算子统计过车（参考代码中已给出实现，具体流程如下图）

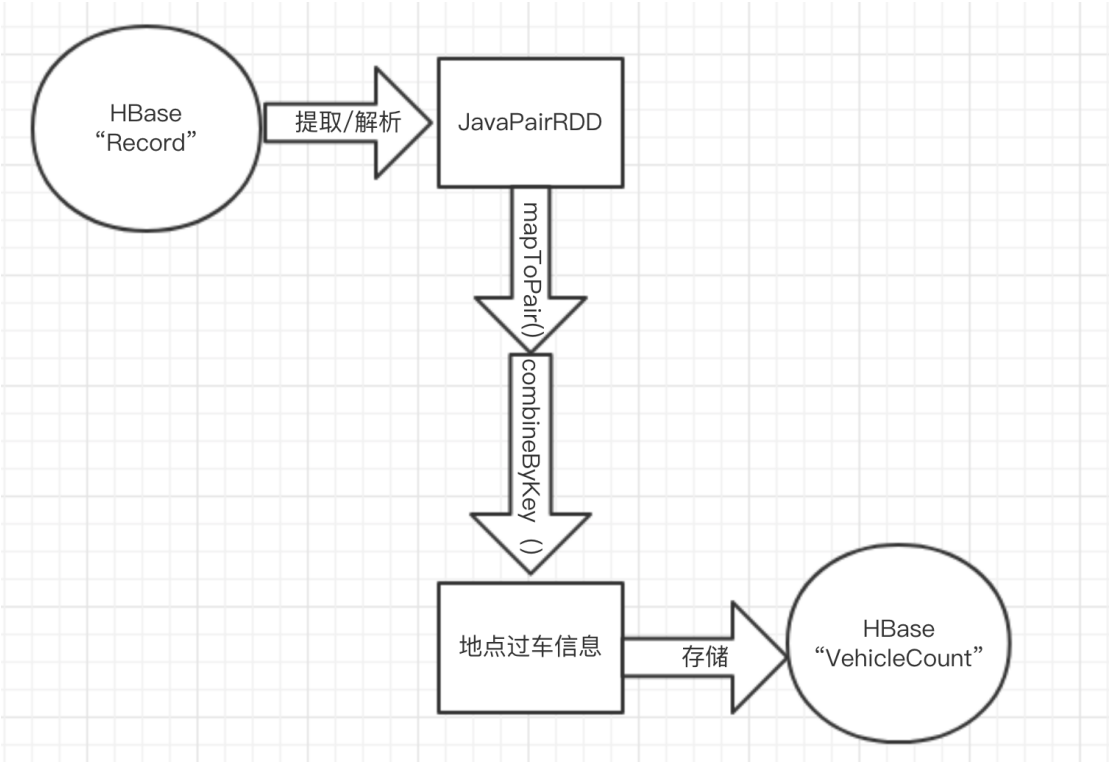

图 7 地点过程统计流程

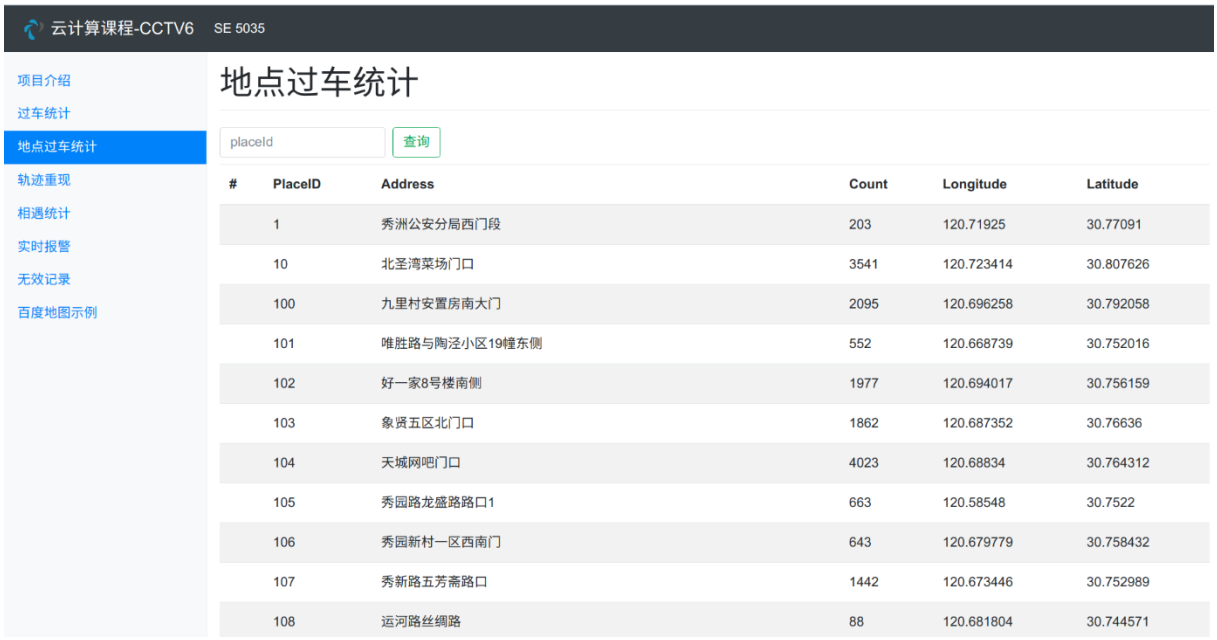

图 8 地点过程统计 Web 页

2. 系统附加功能分析

无效数据过滤

a) 需求：

根据要求，在获取数据的过程中，难免会遇到错误数据。我们所要收集的数据是某一地区内的车辆过车信息，这就需要在获取数据时增加过滤操作，将该地区的经纬度之外的数据过滤掉。

b) 具体步骤：

① 使用程序读取 json 文件，模拟数据源，producer 模块读取数据到 my-topic 话题中，等待 consumer 消费；

② 控制 consumer 将 producer 提供的数据消费掉，此处加入对数据的过滤操作。我们先将 json 按行读取，取出其中的经纬度数值，比较数值是否在合法范围内；

③ 若合法，调用 redis 的接口，将数据传输给“rdb”信道；若不合法，同样是调用 redis 接口，将错误的数据传输给“wrong”信道。

无效记录

#	EID	PlaceID	Time	Address	Longitude	Latitude
	33041100002206	11	1496273822	王江泾医院	300.0	300.0
	33041100013096	146	1496285427	向阳桥	250.0	624.0
	33041100006339	151	1496301901	兴乐路庆元路口1	256.0	287.0
	33041100028592	266	1496301877	新昌苑9幢	350.0	276.0
	00001914238535	90	1496246413	木桥港北区西大门	300.0	300.0

图 9 无效记录 Web 页

Redis 订阅发布

a) 需求：

实现了订阅发布的功能；

b) 具体步骤:

接收到 kafka 传送过来的消息以后，将接收到的消息从不同的信道发送给 hbase 表中。正确的数据发布到“rdb”信道，不合法的数据发布到“wrong”信道。Hbase 的一端通过订阅不同的信道，接收不同的消息。

轨迹重现

a) 需求：

通过用户给定的时间范围和车辆的电子车牌(EID)，重现车辆轨迹，以时间序列

b) 具体步骤：

① 读取 Record 表中的数据，重新编排行键，变为以"eid##time"为行键，将数据插入 Track 表中；

② 根据指定的 eid 和时间范围可以构造出 startRowKey 和 stopRowKey，然后以此为参数进行 scan 操作，由于 HBase 数据存储的特性，得到的就是指定车辆在指定时间范围内按 time 升序排列的结果集；

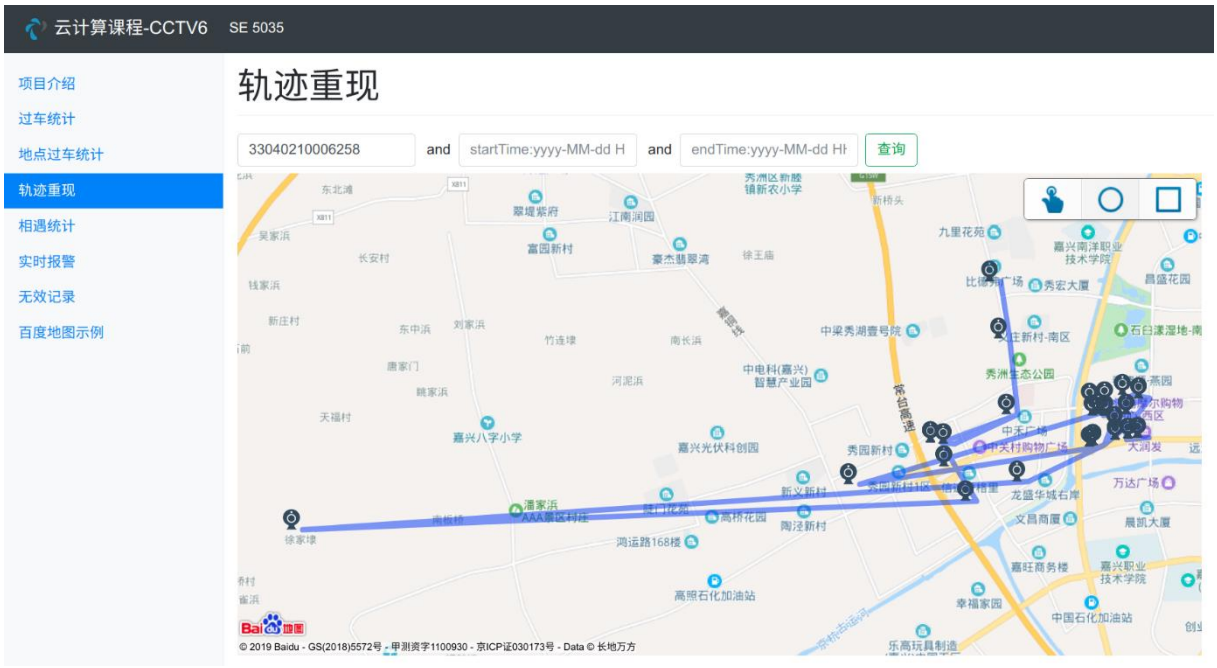

图 10 轨迹重现 Web 页

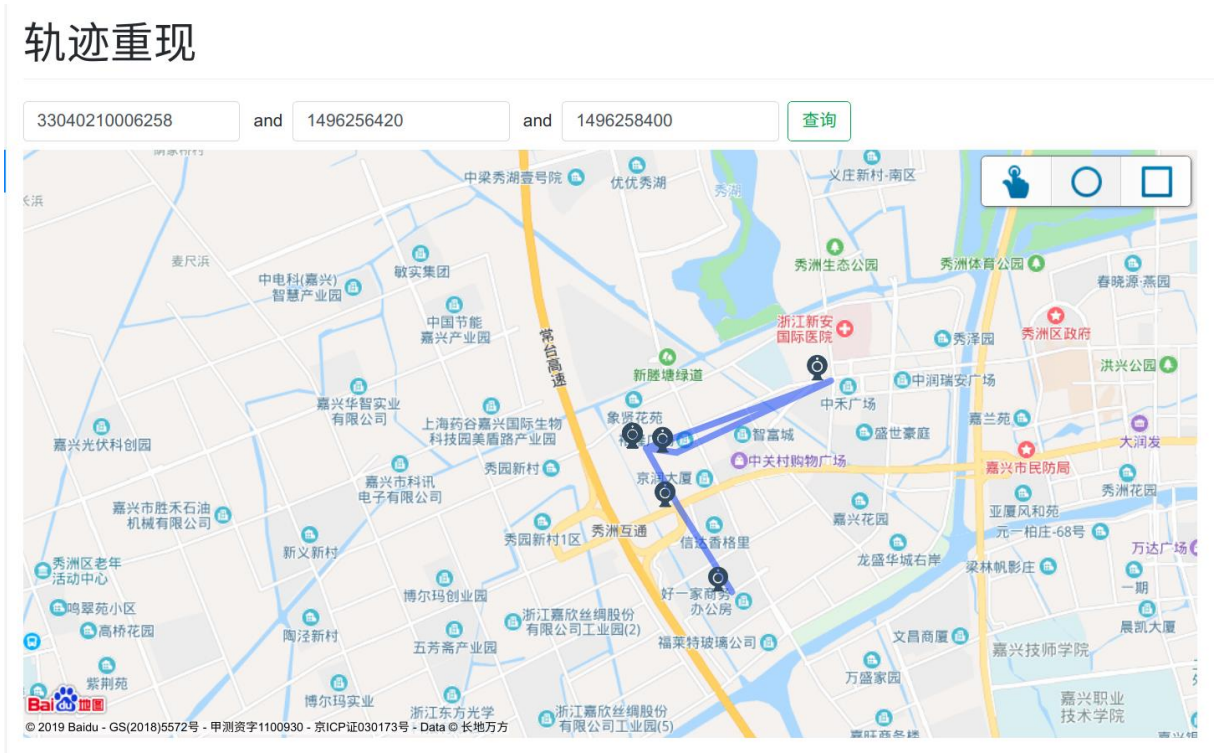

图 11 轨迹重现区间查询 Web 页

相遇次数统计

经过小组分析与讨论遂采取如下方案：

```
(placeId, (eid, time)) join (placeId, (eid, time)) ->
(placeId, (eid1, time1, eid2, time2)) ->
(eid1 != eid2 && |time1 - time2| < 60) filter ->
(placeId, (eid1, eid2)) ->
((eid1, eid2), 1) reduceByKey ->
((eid1, eid2), count)
```

以地点为键，与自身进行 Join 操作，使用 MapReduce 计算与其他车辆相遇的次数，重新编排行键，建立一张新表，即可获取两车相遇次数。

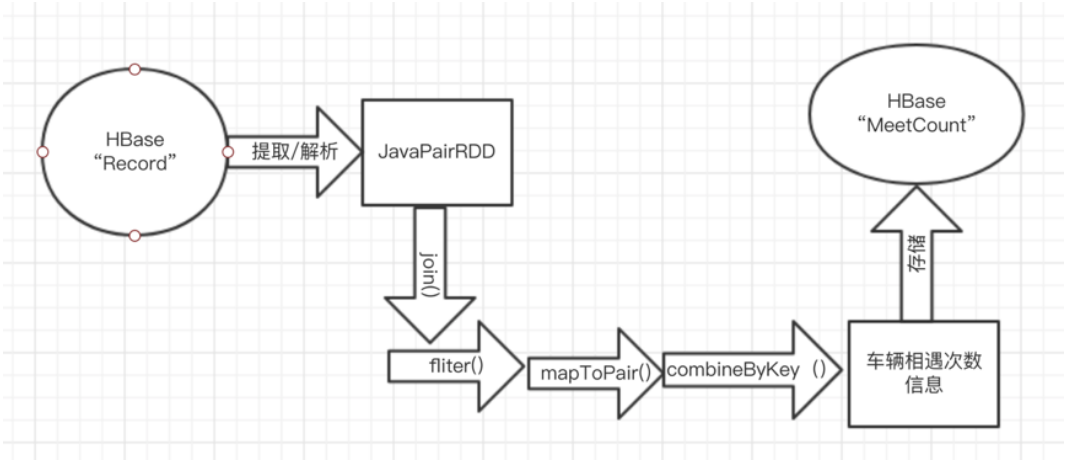

图 12 相遇次数统计流程

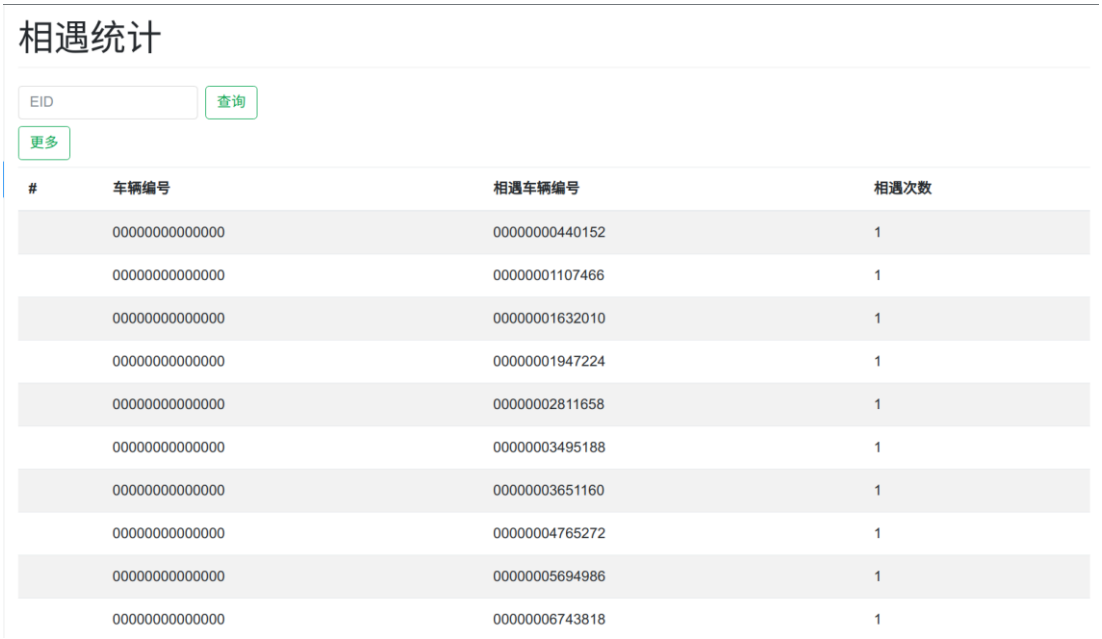

图 13 相遇次数统计 Web 页

四、 实验感受及收获

任晓路：整个课程实验给我带来的最大感受是：现在的任何优秀的软件都是基于众人的智慧，每个人专精于一个方向才能完成像这次这样大的工程。在我和另一位组员在学习和使用 kafka 完成我们的任务的过程中，我学到了规范统一化的代码和管理的重要性。最后感谢各位组员的帮助。

王嘉祥：通过此次实验，感受颇多。其中比较重要的是我经历了这次的多人合作，体会到了团队合作的重要性，从设计、构思到实现，每个成员都不可或缺，也许细节上的因素也会影响整体的效果，所以必须要有统一的标准与信念；另外，我认识到在设计软件时，需要着重考量整体的架构，这方面决定这后续工作是否能够顺利进行。

许鑫：这次的云计算课程收获很大。在经历上，这是第一次超过五个人的小组共同努力的结果，内容上，基本上是第一次接触这类的信息。对于我负责的 redis 的模块，让我对redis的功能有了一定的了解,为何叫它内存数据库,它的存储基本属性是什么，最重要的是 redis 的订阅发布的基本使用方法有了一定的了解，以后也会继续的深入的了解 redis，继续的深入了解分布式存储的内容。

闫杨：通过本次实验，我全面了解了云计算相关软件，对云计算也有了更深的理解。在本次实验中我主要负责 HBase 部分，我深入学习了 HBase 的读写流程以及 HBase 表的结构。在实验过程中，我深深体会到了 HBase 优异的随机读写性能，同时对这种非结构化的数据库有了更深的理解。此次实验也是我参与过的参与人数最多的项目，通过与队友的合作，我体会到了沟通交流与代码规范在团队协作中的重要性。

温哲：本次实验我主要负责的是 hbase 部分，我对于 hbase 的表结构以及 hbase 表的一系列操作进行了学习，对于 hbase 的架构也有了自己的理解与看法。同时为了弄懂对整个项目流程我也了解了 kafka、redis、spark 方面的基础知识和工作流程。本项目人数众多，每个人分工明确，然后再到整体之间的协作，虽然遇到了一系列问题，但最终大家可以一起解决，可见在项目中团队合作的重要性。在本项目中，我也发现规范的代码才能够提高整个项目的完成效率。

安炳旭：此次云计算课程的实验应该是参与前所未有过的的项目，收获颇多，从搭建环境到调试，从遇到问题到解决问题，从单元到整体，从个人到团队，每个部分可能

都遇到了问题，但是每一项都至关重要。我主要负责 Spark 的数据处理部分，在 Spark 官方介绍中写到 Spark 的速度是 MapReduce 的 100 倍，起初我没有概念，但在此次实验之后我再一次为 Spark 的快速而叹服，良好的设计果然给性能会带来极大的提升。一个计算框架是如此，一个大型系统亦如此，好的设计会给之后的开发带来无法想象的便利。

杨杰聪：参与人数如此之多的实验是我之前从未遇到过的，这次实验让我学会了如何在多人间进行沟通、交流、合作。同时，该实验过程中我也了解到了在多人项目中，一个清楚明了的总体规划是非常重要的，其影响了项目的进度。除此，作为负责前端的一员，在此次项目之前，我对前端并没有多少基础，但照着网上的教程、寻求队友的帮助，也能一步一步地完成任务，这增强了我的自信心以及学习能力。这是一个很好的实验设计，给设计者点赞。

郑守建：通过本次实验，我对云计算的技术栈有了全面的认识，体会到了消息队列削峰的作用，学习了 Redis 订阅发布、HBase 提供的 Java API、Spark 的基本使用等内容，并且，在实现 Web 页面过程中，学习了 HBase StartRowKey 等机制，收获很大。同时，通过这次实验，我体会到了团队合作的重要性，该实验由我们小组 8 个人一起协力完成，实验过程中的时间协商、任务分工等都考验着我们的团队协作能力。

五、 课程建议

1. 在上机的内容上，可以有更多的拓展。这一次的加分项的要求并不多，可以在现有的基础上，提出更多的功能要求，或者划分出第三个层次，提出一些难度极高的内容，让更多的人尝试挑战自己，更好的划分出层次。
2. 建议提供一个统一的云上实验环境。
3. Piazza 中的问题是按提问时间排列的，当我们寻找自己遇到的问题时，要一个个点开看，这样太麻烦了，建议同时按内容和时间进行分类。