计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(10):133−139 [doi: 10.15888/j.cnki.csa.006592]

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

基于 Spark Streaming 的实时交通数据处理平台①

谭　亮, 周　静

(四川省交通运输发展战略和规划科学研究院, 成都 611130)
通讯作者: 谭　亮, E-mail: 26101603@qq.com

摘　要: 交通大数据是解决城市交通问题的最基本条件, 是制定宏观城市交通发展战略规划和进行微观道路交通管理与控制的重要保障. 针对于智能交通系统中数据产生快、实时性强、数据量大的特点, 本文基于 Spark Streaming 和 Apache Kafka 的组合构建了一个实时交通数据处理平台, 用于处理通过双基基站采集的数据, 采用时间窗口机制从持续的 Kafka 分布式消息队列中获取数据, 并按照规则将数据分类处理后保存到数据库. 本文对平台的系统架构和内部结构进行了详细的介绍, 并通过实验验证了系统的实时处理能力, 完全可以在大规模高并发的数据流下进行应用.

关键词: 大数据; 流处理系统; 双基基站数据; Spark Streaming; Apache Kafka

引用格式: 谭亮,周静.基于 Spark Streaming 的实时交通数据处理平台.计算机系统应用,2018,27(10):133–139. http://www.c-s-a.org.cn/1003-3254/6592.html

Real-Time Traffic Data Processing System Based on Spark Streaming

TAN Liang, ZHOU Jing

(Sichuan Provincial Institute of Transportation Development Strategy and Planning, Chengdu 611130, China)

Abstract: Traffic big data is the most basic condition for solving urban traffic problems. It is an important guarantee for formulating macro-city traffic development strategy and construction planning. And it is also an important guarantee for carrying out micro-road traffic management and control. In view of the characteristics of intelligent transportation system such as fast data generation, high real-time performance and large amount of data, this paper constructs a real-time traffic data processing platform based on the combination of Spark Streaming and Apache Kafka to process the data collected by dual base stations. Using time window mechanism to get data from Kafka, and process the data to the database according to the rules. In this study, the system architecture and internal structure of the platform are introduced in detail, and the real-time processing capability of the system is verified through experiments, which can be applied under large-scale and high-concurrency data flow.

Key words: big data; stream processing system; double station data; Spark Streaming; Apache Kafka

随着社会经济的发展, 人民生活水平日益提高, 城市交通工具的保有量不断加大, 道路供需矛盾日渐突出, 各种城市交通问题不断涌现. 为改善城市交通现状, 仅仅依靠增加道路基础设施是远不够的, 还需要不断提高交通管理的信息化和智能化水平. 随着智能交通系统的逐步实施, 各类交通检测设备提供了大量的数据, 这些数据为解决城市交通问题打开了新思路, 是解决城市交通问题的最基本条件, 是制定宏观城市交通发展战略规划和进行微观道路交通管理与控制的重要保障[1].

① 收稿时间: 2018-03-09; 修改时间: 2018-04-03; 采用时间: 2018-04-12; csa 在线出版时间: 2018-09-28

因此, 针对智能交通系统中数据产生快、实时性强、数据量大等特点, 研究大数据技术在交通信息化领域的数据处理方法具有非常重要的意义.

1 相关研究现状

通过文献研究发现, 交通大数据在车辆轨迹统计、伴随车辆发现、车流量预测和假套牌车辆甄别等方面的应用日趋增多, 但是在对数据分析之前都需要对数据进行相关的处理. 随着分布式计算技术的发展, 大数据技术在交通数据处理和分析领域得到广泛应用.

文献[2]提出了基于 MapReduce 与 K 近邻搜索算法对海量历史数据进行短时交通流预测, 能在短时间内准确预测下一个时间段内的交通流. 文献[3]中曹波等人利用分布式并行数据处理框架 Spark 对交通摄像头捕捉到的道路交通数据处理生成的车牌识别数据进行处理分析, 基于并行 FPGrowth 算法能快速高效的发现伴随车辆. 文献[4]基于交通摄像头采集的实时监控数据, 利用云基础设施的并行计算能力, 能有效处理大规模的流式车牌识别数据, 即时发现疑似伴随车辆并实时地输出发现结果. 经过对大量文献的研究发现, 目前大数据技术在交通领域的应用主要有两个问题:

(1) 大部分针对离线数据作处理. 目前利用大数据技术对交通数据进行的处理大部分都是针对的离线数据, 采用 MapReduce 编程模型, 将数据处理过程抽象成 Map 和 Reduce 两个操作, 利用“分而治之”思想进行分布式处理[5]. 这种处理方式针对的是静态历史数据, 面对低延时的实时数据流处理, 其计算效率比较低下, 无法满足对流数据进行实时处理的需求.

(2) 针对单一数据源作处理. 目前的文献中利用大数据技术在城市交通领域主要是对单一的数据源进行处理分析, 比如视频监控数据或 RFID 数据, 不能对多数据源的数据进行有效处理, 无法综合有效利用智能交通系统采集的数据, 从而弥补单一数据的限制和缺陷.

针对以上问题, 本文重点研究如何利用大数据技术对智能交通中通过双基基站获取的 RFID 交通数据和视频抓拍数据进行相关的实时计算处理.

2 双基基站技术

双基基站, 是指每个交通数据采集基站包含 RFID 交通数据采集系统和视频抓拍系统 2 套数据采集设备, 即当车辆通过基站时, 能够同时得到 RFID 过车记录和视频抓拍过车记录[6]. RFID 交通数据采集系统主要由贴在车辆挡风玻璃上的 RFID 电子卡、安装在路上的 RFID 读写器、通信传输网络和信息中心组成. RFID 读写器可采集到车辆所携带 RFID 标签中的信息, 主要包括记录序号、基站编号、车道编号、开始读卡时间、结束读卡时间、标签编号、标签类型、车牌号、车牌颜色、车型、车身颜色等. 视频抓拍系统采集到的数据属性, 主要包括记录序号、基站编号、车道编号、过车时间、车牌号、车牌颜色、车身颜色、地点车速、违章类型、图像编号、是否超速等.

通过对以上 2 种设备采集的数据进行研究分析, 这两种数据都具有数据持续到达、数据实时到达和数据规模宏大等特点, 使得系统在单位时间内需要处理的数据量大幅增加, 传统的数据处理架构已不能满足实际需求. 为了解决这个问题, 本文基于 Spark Streaming 和 Apache Kafka 相组合设计了一种的分布式交通数据处理平台.

3 实时交通流数据处理平台

3.1 Kafka 消息系统

Kafka 最初是由 LinkedIn 公司开发的一套高吞吐量的分布式消息订阅和发布系统[7], 其吞吐量可随集群的扩展而线性增加, 在海量数据处理领域被广泛使用. 图 1 为 Kafka 的整体架构, 由生产者 (Producer)、代理 (Broker) 和消费者 (Consumer) 三大部分构成; 其中 Producer 负责收集消息并推送到 Broker, 而 Broker 则负责接收消息, 并将消息本地持久化, Consumer 则是消息的真正使用者, 从 Broker 拉取消息并进行处理[8].

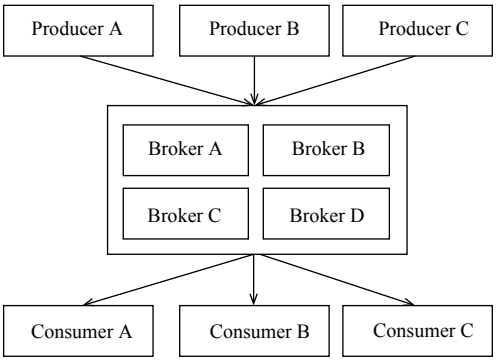

图 1 Kafka 整体结构图

一个主题 Topic 可以认为是一个队列, 每条消息

都必须指定 Topic. 为了使 Kafka 的吞吐率线性提高, 每个 Topic 可分成一个或多个 partition, 每个 partition 在物理存储层面对应一个文件夹, 文件夹下包括这个 partition 的所有消息和索引文件. 一个 Topic 的所有 partitions 被分布在 Kafka 集群中的多个 server 上, 每个 server 上的 Kafka 实例负责该 server 上的 partition 中消息的读写操作. 当消息到来时, 被直接追加到该分区中, 属于顺序写磁盘操作, 因此效率非常高; 当消费者消费消息时, Kafka 会为每个 consumer 保留偏移量信息 offset, 该 offset 由 consumer 控制, 正常情况下 consumer 在消费完一条消息后顺序递增该 offset, 不需要 Kafka 使用锁机制标记哪些消息被哪些消费者消费过, 从而为 Kafka 的高吞吐率提供了有力保障. 通过这种分区机制, 保证了消息的保存/消费的效率, 有效提升了 Kafka 的吞吐率.

为了高可用性 Kafka 引入了 replication 机制, 每个 partition 被备份到多台服务器上, 基于该方案, 就意味着需要对 partition 的多个备份进行调度; 每个 partition 都有一个 server 为主导者, 负责所有的读写操作; 其它 server 作为跟随者, 只是进行消息同步. 如果主导者 server 失效, 其他跟随者 server 来将会接管成为新的主导者, 这样始终只有一个 server 负责读写操作, 使得系统更加简单高效. 但是, 作为主导者的 server 会承载全部的请求压力, 为了更好的负载均衡, Kafka 会尽量将所有的 partition 均匀分配到整个集群上, 从而确保整体的稳定高效.

Kafka 以时间复杂度为 O(1) 的方式提供消息持久化能力, 具有高吞吐率、高可靠性和易扩展的特点, 可同时支持离线数据处理和实时数据处理. 基于此, kafka 完全适用于分布式交通大数据处理系统.

3.2 Spark Streaming

流式计算作为大数据计算领域的一种主要模型, 当前主流的流计算框架有 Twitter 公司开发 Storm、Yahoo 公司开发 S4、微软的 Timestream 以及 UC Berkeley AMPLab 开发的 Spark Streaming 等[9]. 其中, Spark Streaming 是基于 DStream 模型并构建在 Spark 计算引擎上的分布式流式计算框架, 可以实现高吞吐量的、具备容错机制的实时流数据的处理.

DStream (Discretized Stream) 是 Spark Streaming 中一个关键的程序抽象, 表示一个持续不断输入的数据流, 可以基于 Kafka 等输入数据流创建[10]. 在内部, 一个 DStream 是通过一组时间上连续的弹性分布式数据集 RDD 序列组成; 每个 RDD 包含了一定时间间隔内的数据流, 是一个不可变的分布式可重算的数据集, 如图 2 所示. 由于 Spark Streaming 是基于 Spark 处理引擎的, 其计算流程实际上就是将输入数据分成一段一段的 DStream, 每一段 DStream 都转换成针对 Spark 的 RDD, 然后通过 Spark 引擎将 RDD 经过操作变成中间结果保存在内存中, 最后再根据业务的需求对中间结果进行叠加或存储到外部设备.

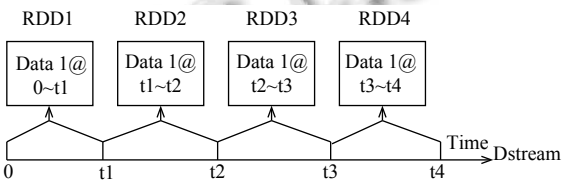

图 2 Spark Streaming 处理过程图

Spark Streaming 支持从多种数据源获取数据, 比如 TCP Socket、Flume、Kafka、Twitter、ZeroMQ 等. 针对于 Kafka 消息系统, 在系统开发过程中直接调用 Kafka 的简单消费者 API CreateDirectStream 读取数据, 并创建和 Kafka 分区一样的 RDD 分区个数进行一一对应, 从而提高系统性能.

选择 Spark Streaming 一方面是因为数据流入系统就可以进行实时处理并得出结果, 框架具有很好的扩展性、容错性和吞吐量, 适合数据连续产生和需要实时处理的应用场景; 另一方面是因为它基于 Spark 批处理引擎构建, 能和批处理、即时查询放在同一个软件栈中, 实现数据的无缝共享, 从而降低开发和维护成本, 也为系统后期提供离线数据处理和挖掘服务提供基础.

3.3 平台系统架构

本文利用 Kafka 和 Spark Streaming 框架设计的分布式交通流数据处理平台的系统架构如图 3 所示, 包括数据接收部分、Kafka 消息订阅系统、数据处理程序、以及数据存储部分.

数据接收部分, 用于接收从前端数据采集器发送的数据, 而且要保证与客户端数据传输的高效性和稳定性. Netty 作为一个高性能、异步事件驱动的网络应用程序框架[11], 利用多线程或 IO 多路复用技术可以同时并发处理成千上万个客户端的接入请求, 在大数据分布式计算领域被广泛应用, 本文设计的数据处理平

台采用 Netty 作为通信服务器. 在本系统中前端数据采集器作为客户端, Netty 服务器作为服务端收集从客户端发送过来的数据请求消息. Netty 数据接收服务器, 被视作消息生产者, 负责将接收到的数据通过创建的生产者接口写入 Kafka 消息系统, 不同类型的交通数据发送到 Kafka 的不同 Topic 中.

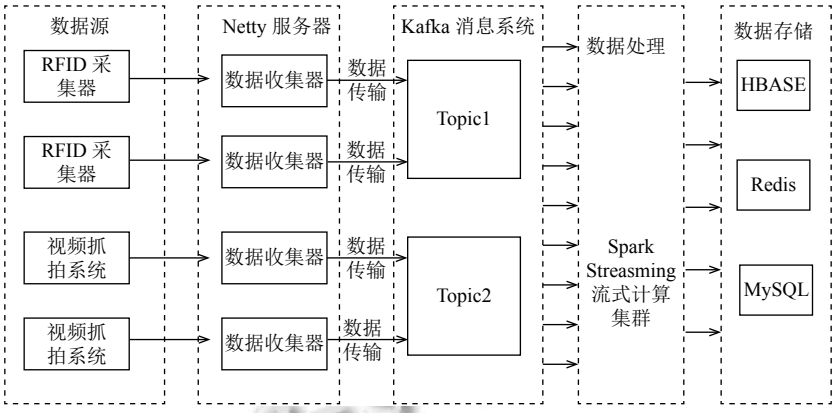

图 3 系统整体架构图

Kafka 消息系统, 将来自 RFID 采集设备和视频抓拍设备的数据汇集成数据流. Kafka 消息系统将不同类型的数据分散在不同的 Topic 中; 并根据不同的业务处理需求, 将数据转发到 Spark Streaming 集群中不同的处理模块中进行处理.

数据处理程序, 运行在 Spark Streaming 集群上, 是系统的核心模块; 以 Kafka 消息系统作为数据输入流进行实时处理. Spark Streaming 从 Kafka 消息队列中按照时间窗口不断提取数据, 然后进行业务处理, 其中包括直接业务处理程序和间接业务处理程序两种. 直接业务处理程序在本文中主要指过车流量统计; 间接业务处理程序首先进行数据拼接处理, 然后在按照业务规则进行处理, 本文主要包括轨迹数据统计和异常数据提取. 随着业务的不断扩展, 可以根据需要开发其他的业务处理程序方便快捷的部署在集群上.

数据存储部分, 对处理的结果数据进行存储. 当数据经过数据处理程序完成数据拼接、统计和异常处理后, 需要针对不同的处理结果采用不同的存储方式, 主要有关系型数据库、内存数据库和分布式数据库等. 车辆轨迹处理结果存储在 HBase 中, 将 HBase 划分为多个不同的域, 以车牌号和时间字符串的逆序字符串为键进行存储; 对各个采集点的过车流量统计结果存储在内存数据库 Redis 中; 将异常数据存储到关系数据库 MySQL 中.

3.4 处理平台内部结构

图 4 表示交通数据处理平台的内部结构图. Spark Streaming 集群由多个工作节点组成, 每个工作节点上运行多个 Spark Executor, 在 Spark Executor 上运行相关的业务处理程序. 同时, 在每个工作节点上安装分布式数据库系统 HBase、内存数据库 Redis 和关系型数据库 MySQL; HBase 用来存储车辆轨迹数据; MySQL 用来存储异常数据记录; Redis 用存储采集点的过车流量数据. 为了保证系统的可扩展性, 本文选择的三种存储方式 HBase、Redis 和 MySQL 都可以采用集群方案进行部署.

来自视频抓拍设备的数据使用话题 Topic1 写入 Kafka 消息系统, RFID 数据采集器的数据使用 Topic2 写入 Kafka 消息系统. 为了保证系统的吞吐量, 需要对 Kafka 主题进行分区, 通过分区从而提高系统的并发读写能力. 在系统的内部结构中, 每个 Spark Executor 运行一个直接业务处理程序或者间接业务处理程序. 同时, Topic1 的每个消息分区有一个数据消费者 (间接业务处理程序); Topic2 的每个消息分区有两个数据消费者: 一个是直接业务处理程序, 指的是基于 Spark Streaming 流式集群的过程流量统计处理程序; 另一个是间接业务处理程序, 首先与 Topic1 的视频数据按照规则进行拼接程处理, 然后按照业务规则进行实时数据处理, 包括车辆轨迹和异常数据的处理. 一个或多个业务处理程序组成一个消息消费者小组, 同时保证每个分区的数据至少会被每个消费者小组中的一个消费者接收.

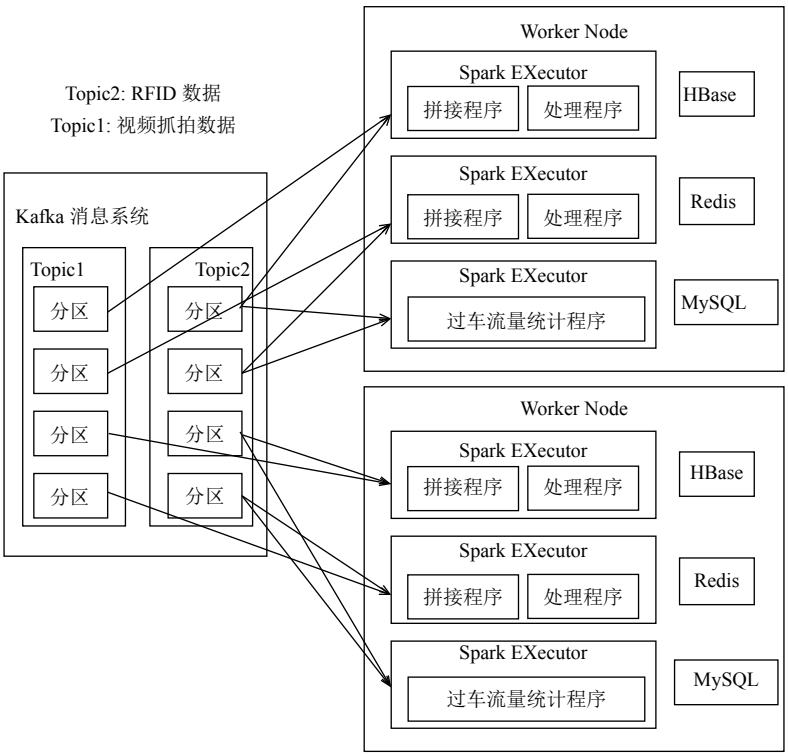

图 4 系统内部结构图

图 4 描述的系统内部结构包含两个工作节点组成的 Spark Streaming 集群, 每个工作节点运行三个 Spark Executor, 一个 Spark Executor 负责一个业务处理程序; 其中, 有两个 Spark Executor 负责间接业务处理程序, 一个 Spark Executor 负责直接业务处理程序. 图中每个消息话题分了四个分区, 间接业务处理程序消费一个 Topic1 的消息分区和一个 Topic2 的消息分区的数据; 直接业务处理程序负责消费 Topic2 的两个消息分区的数据. 为保证系统性能, Topic 的分区个数最好设置为消费者的倍数, 同时最好保证每个消费者负责处理的消息分区个数是相等的.

数据拼接程序, 属于间接业务处理程序中的首要模块. 数据拼接程序根据 Kafka 消息系统从 Kafka 中按照时间窗口不断提取数据, 按照设定的时间间隔从持续的 Kafka 分布式消息队列中获取 RFID 过车数据和抓拍数据, 每次累计获取设定时间段以内的数据进行拼接处理, 然后将处理结果供其他处理程序使用, 具体拼接处理如图 5 所示. 首先将 RFID 数据、抓拍数据分别封装为相应的 RDD; 然后对两类数据 RDD 分别进行转换, 得到键值对形式的 RDD, 以方便比对连接操作的进行, 其中键为车牌号、时间、基站编号三个字段组成的字符串; 最后, 将两种数据的 RDD 根据键值进行比对并连接.

间接业务处理程序除了数据拼接程序之外, 还运行着多个实时数据处理程序, 包括车辆轨迹处理程序、异常数据处理程序. 每个实时处理程序根据数据拼接程序处理的结果再按照业务规则进行处理, 并把数据处理的结果存储起来.

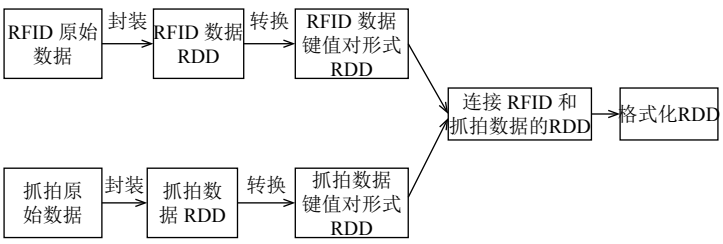

图 5 数据拼接过程

(1) 车辆轨迹数据统计程序

车辆轨迹处理程序根据拼接处理的格式化 RDD, 利用 Spark Streaming 提供的比对函数对车牌号和时间字符串进行逆序处理, 使用车牌颜色和通行时间字段对数据 行过滤, 得到车辆轨迹数据; 以车牌号和时间字符串的逆序字符串为键, 将所述车辆轨迹数据存储在 HBase 中, 将 HBase 划分为多个不同的域, 以车牌号和时间字符串的逆序字符串为键进行存储.

(2) 异常数据提取程序

异常数据提取程序根据拼接处理的格式化 RDD, 利用异常数据的判定规则分别对 RFID 过车数据和抓拍数据进行过滤, 提取出异常数据, 并对异常数据进行统计. 数据异常类型分为数据缺失、车牌颜色字段缺失、抓拍图片链接确实、号牌不一致、颜色不一致. 根据数据异常类型, 首先判断 RFID 数据是否缺失, 如果存在 RFID 数据, 则判断 RFID 数据中颜色字段是否存在、抓拍数据中抓拍图片链接是否存在, 如果字段完整, 则判断 RFID 数据和抓拍数据中号牌、颜色是否一致, 最后, 将提取出的异常数据存储到 MySQL 数据库中, 并标识异常类型. 采用关系型数据库 MySQL 进行存储.

(3) 过车流量统计程序

过程流量统计程序将 RFID 过车数据转换为以采集点字段和精确至小时的时间字符串为键的键值对形式; 根据 Spark Streaming 分布式大数据处理的原理, 对具有相同键的数据记录进行计数, 然后对每个采集点的统计结果以设定的时间间隔进行求和, 得到各个采集点在相应时间段内的过车流量记录; 使用内存数据库对各个采集点的过车流量进行存储.

4 系统实验测试

为了保证平台的数据处理能力, 同时考虑系统的可扩展性和负载均衡, 本文搭建了如图所示的测试环境. 测试环境使用 vSphere 搭建 6 台虚拟机集群, 每个虚拟机的配置为 8 核 CPU, 内存 16 GB, 磁盘 80 G, 其具体部署情况如表 1 所示. 在实验过程中, 采用 LoadRunner 工具模拟 RFID 采集器和视频抓拍设备采集数据; Netty 作为数据通信接收器, 用于接收从 LoadRunner 发送过来的数据并写入给 Kafka 消息订阅系统; Zookeeper 负责分布式应用程序的协调服务.

表 1 测试环境部署结构表

服务器	运行程序
服务器 1	Spark master、Zookeeper、HMaster、Kafka、Redis master、MySQL master、
服务器 2	Spark worker1、Zookeeper 、HRegionServer、Kafka、Redis slave、MySQL slave
服务器 3	Spark worker2、Zookeeper 、HRegionServer、Kafka、Redis slave、MySQL slave
服务器 4	Spark worker3、Zookeeper 、HRegionServer、Kafka、Redis slave、MySQL slave
服务器 5	Spark worker4、Zookeeper 、HRegionServer、Kafka、Redis slave、MySQL slave
服务器 6	Netty

衡量系统的实时处理能力通常是测试系统处理一条数据所需要的时间. 系统的处理每条数据的时间越短, 说明系统的实时处理能力越强. 在测试过程中, 将 LoadRunner 工具的并发量分别设置为 2000、4000、6000、8000、10 000, 分别表示模拟 2000、4000、6000、8000、10 000 个采集设备采集数据, 采集频率都为 1 条/秒. 启动系统进行计算处理, 观察并记录一条 RFID 数据和视频抓拍设备的数据从传输到处理完成所需要的时间.

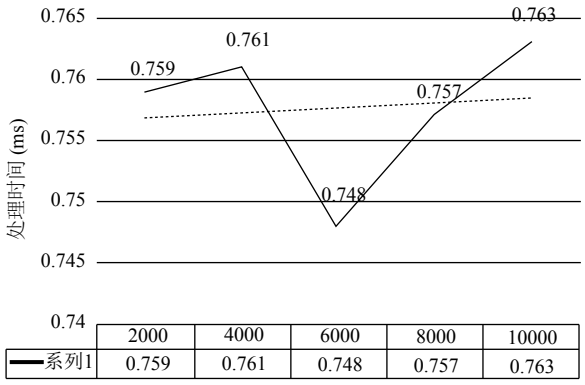

图 6 实验结果

图 6 中横坐标表示数据采集设备的并发数量; 纵坐标表示计算处理时间. 从图中可以看出当并发量在 2000 到 10 000 的时候, 其处理时间在 0.748 ms 和 0.763 ms 之间波动, 而且变化的幅度较小, 其处理时间都接近于 0.7 ms. 根据实验结果, 一条数据从传输到处理完成所需的时间是毫秒级的, 说明本系统具备很好的实时处理能力, 完全可以在大规模高并发的数据流场景中应用.

Spark Streaming 与 Kafka 组合使用, 当数据量较

小, 很多参数的默认配置便能够满足应用情况, 但是当数据量大的时候, 就需要从多方面入手对系统性能进行调优, 一方面是在开发过程中尽量选择高性能的算子, 比如使用 mapPartitions 替代普通 map、使用 foreachPartitions 替代 foreach 等; 另一方面是根据实际情况通过实验对各项参数进行调整, 比如对 Spark Streaming 的批处理时间 batchDuration、Kafka 拉取量 maxRatePerPartition 和分区数量 partition 等参数进行实验调优.

5 结论

本文提出的基于 Spark Streaming 和 Kafka 相组合的交通数据处理平台充分利用了当前最先进的实时大数据流处理技术, 并针对交通数据流的特点对系统架构和内部结构进行了研究和设计. 经过研究分析, 本文设计的实时交通数据平台具有如下优势:

(1) 具有高效的数据处理能力. 系统采用当前先进的流式处理框架 Spark Streaming 和高可靠、高吞吐量的 Kafka 消息系统相组合作为系统核心框架. 由于 Spark Streaming 的数据处理过程是基于内存实现的, 具备很强的实时处理能力, 其吞吐率比传统架构的处理方式提高上百倍.

本文采用 Spark Streaming 流处理技术, 按照设定的时间间隔从持续的 Kafka 分布式消息队列中获取 RFID 过车数据和抓拍数据, 每次累计获取设定时间段以内的数据, 再按照规则进行拼接和处理. 利用时间窗口机制, 可以有效解决数据等待问题, 提高平台的数据处理能力.

(2) 具有很强的可扩展性. 系统架构中的 Kafka 消息系统、Spark Streaming 集群、数据存储 (HBase、Redis 和 MySQL) 以及 Netty 数据通信服务器都可以采用分布式集群架构, 具有很强的扩展能力. 所以, 在实际应用的时候, 只需要根据目前的处理需求设计系统部署方案即可, 不必过多考虑将来的业务扩展需求, 从而减少项目初期部署的设计负担和建设成本.

(3) 具有很好的业务功能扩展能力. 在当前的设计中, 本文只提供了车辆轨迹数据统计、过车流量统计和异常数据提取这三个实时数据处理功能, 将来可以在系统中根据实际业务需求方便地增加新的处理程序. 新增加的业务处理程序将会与原有的处理程序并行进行, 不会影响现有的业务处理功能.

(4) 具有双基基站数据处理能力. 本文设计的系统可以同时利用和处理从 RFID 采集器和视频抓拍设备采集的数据流, 将两种数据流分类处理后进行存储, 有利于后期的查询和分析.

参考文献

1 陆化普, 孙智源, 屈闻聪. 大数据及其在城市智能交通系统中的应用综述. 交通运输系统工程与信息, 2015, 15(5): 45–52. [doi: 10.3969/j.issn.1009-6744.2015.05.007]

2 Chen C, Liu Z, Lin WH, *et al*. Distributed modeling in a mapreduce framework for data-driven traffic flow forecasting. IEEE Transactions on Intelligent Transportation Systems, 2013, 14(1): 22–33. [doi: 10.1109/TITS.2012.2205144]

3 曹波, 韩燕波, 王桂玲. 基于车牌识别大数据的伴随车辆组发现方法. 计算机应用, 2015, 35(11): 3203–3207. [doi: 10.11772/j.issn.1001-9081.2015.11.3203]

4 朱美玲, 王雄斌, 张守利, 等. 基于大规模流式车牌识别数据的即时伴随车辆发现. 中国科学技术大学学报, 2016, (1): 47–55.

5 朱继召, 贾岩涛, 徐君, 等. SparkCRF: 一种基于 Spark 的并行 CRFs 算法实现. 计算机研究与发展, 2016, 53(8): 1819–1828.

6 于影, 王静, 宁丹, 等. 基于双基站交通数据采集技术的假套牌车辆识别方法. 交通信息与安全, 2017, 35(4): 76–83. [doi: 10.3963/j.issn.1674-4861.2017.04.010]

7 Ranjan R. Streaming big data processing in datacenter clouds. IEEE Cloud Computing, 2014, 1(1): 78–83. [doi: 10.1109/MCC.2014.22]

8 韩德志, 陈旭光, 雷雨馨, 等. 基于 Spark Streaming 的实时数据分析系统及其应用. 计算机应用, 2017, 37(5): 1263–1269.

9 崔星灿, 禹晓辉, 刘洋, 等. 分布式流处理技术综述. 计算机研究与发展, 2015, 52(2): 318–332.

10 易佳, 薛晨, 王树鹏. 分布式流数据加载和查询技术优化. 计算机科学, 2017, 44(5): 172–177.

11 魏晓辉, 李聪, 李洪亮, 等. 支持大规模流数据处理的在线 MapReduce 数据传输机制. 吉林大学学报 (理学版), 2015, 53(2): 273–279.