计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150458
53(6):1332-1341,2016

基于云架构的交通感知数据集成处理平台

赵卓峰　丁维龙　韩燕波
(北方工业大学计算机学院　北京　100144)
(大规模流数据集成与分析技术北京市重点实验室(北方工业大学)　北京　100144)
(edzhao@ncut.edu.cn)

An Intergrated Processing Platform for Traffic Sensor Data Based on Cloud Architecture

Zhao Zhuofeng, Ding Weilong, and Han Yanbo
(*School of Computer Science and Technology, North China University of Technology, Beijing* 100144)
(*Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data (North China University of Technology), Beijing* 100144)

Abstract With the continuous expansion of the scope of traffic sensor networks, traffic sensor data becomes widely available and is continuously being produced. Traffic sensor data gathered by large amounts of sensors shows the massive, continuous, streaming and spatio-temporal characteristics compared with traditional traffic data. How to provide intergrated support for multi-source, massive and continuous traffic sensor data processing is becoming one key issue of the implementation of diversified traffic applications. However, due to the absence of support for spatio-temporal traffic sensor data, it is difficult to develop corresponding applications and optimize the data transfer among different nodes in currenent distributed computing platforms. In this paper, we propose a traffic domain-specific processing model based on spatio-temporal data object. The spatio-temporal data object is treated as the first-class object in the distributed processing model. According to the model, we implement an intergrated processing platform for traffic sensor data based on the share-nothing architecture of cloud computing, which is designed to combine spatio-temporal data partition, pipelined parallel processing and stream computing to support traffic sensor data processing in a scalable architecture with real-time guarantee. Applications of the platform in real project and experiments based on real traffice sensor data show that our platform excels in performance and extensibility compared with traditional traffic sensor data processing system.

Key words cloud architecture; traffic sensor data; spatio-temporal data object; real-time MapReduce; stream computing

摘　要　海量、多源、不间断的交通感知数据环境下,如何提供集成化的交通感知数据处理支持是多样化交通应用实施中的难点.现有的通用计算框架及平台由于缺少对具有时空相关等特征的交通感知数据和应用间交通感知数据共享的支持,使得交通感知数据处理应用的开发存在较高的复杂性并且易于造

收稿日期:2015-06-09;**修回日期**:2015-09-11
基金项目:国家自然科学基金重点项目(61033006);北京市自然科学基金项目(4131001,4162021);北京市属高等学校创新团队建设项目(IDHT20130502);北方工业大学校科研基金项目
This work was supported by the Key Program of the National Natural Science Foundation of China (61033006), the Natural Science Foundation of Beijing (4131001, 4162021), the Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges under Beijing Municipality (IDHT20130502), and the Research Funding of North China University of Technology.

成大量重复的数据跨节点传输而影响应用性能.针对此问题,通过分析交通感知数据及其处理需求特征,提出一种基于可跨应用共享的时空数据对象的交通感知数据处理模型,通过引入时空数据对象这一新的概念抽象并提供易并行划分的时空数据对象组织及共享支持,实现分布计算中对时空型交通感知数据的优化管理.在此基础上,设计并实现了交通感知数据集成处理平台.通过实际应用和基于真实交通数据的实验测试表明:该平台相对于传统的交通感知数据处理方法及系统在性能及扩展性等方面均具有一定的优势.

关键词 云架构;交通感知数据;时空数据对象;实时 MapReduce;流计算

中图法分类号 TP333

随着交通相关的传感、监测技术的发展,各类智能交通系统在实践应用中不断产生并积累了大量反映实际交通状态的交通感知数据,这些通过各种交通传感设备(如全球定位系统传感器、车牌识别传感器、交通流量传感器、路况传感器、车况传感器等)实时采集并集中汇聚的交通感知数据逐渐成为智能交通系统中一类新的、关键的数据内容,基于此类数据提供更加精确、全面、智能的交通管理及信息服务成为当前智能交通系统中的研发热点.然而,如何面向海量、多源、不间断的交通感知数据提供可扩展、实时、连续的处理支持以满足多样化、动态变化的交通应用需求成为当前智能交通系统建设的一个核心问题.

近年来,研究者们从时空数据管理、大规模数据的分布式处理、数据流管理、流式计算等不同角度开展了大量与上述需求相关的研究工作,通过这些工作我们可以看到,当前在与本课题相关的研究领域表现出 3 个方面的发展趋势:

1) 体系架构方面.以 MapReduce[1] 为代表的大数据处理工作采用了具有水平扩展优势的无共享集群云架构,该种架构已经成为了大数据处理方向事实上典型架构之一.

2) 计算模型方面.近年来则针对数据流及大数据实时处理情景体现出了由批处理计算到流式计算的发展趋势[2].

3) 数据查询处理优化方面.从时空特征角度建立各类数据组织管理模型及索引机制成为优化具有时空属性数据的研究重点.

然而,现有工作在应对交通感知数据处理需求时尚存在 3 个问题:

1) 当前主流的大数据处理技术,无论是批处理计算还是流式计算,其计算模型抽象层次都过高,缺乏对特定数据模式(特别是具有时空特征的交通感知数据模式)的直接支持,使得在编写海量交通感知数据处理应用时对程序员有较高的能力要求;

2) 传统时空数据管理工作虽然从不同角度探索了时空数据的存储、索引等方面的优化技术,但大多数工作局限于单机环境可存储管理的数据规模,在应对需要分布式环境的大规模数据时表现出一些不足,特别是缺乏对具有良好伸缩性和可靠性保障的云架构的利用;

3) 在现有云架构下典型的数据处理模式下,一个作业内的数据往往局限于其自身使用,针对交通领域一种交通感知数据可支撑数十种交通应用的情况,现有的处理模式由于难以提供数据在应用间的可控共享与运行时支撑,从而会在分布式环境中造成大量不必要的数据复制与网络传递,影响数据处理的性能.

本文针对上述问题并结合时空型交通感知数据及其处理类型特征,提出并设计了一个基于云架构的海量交通感知数据集成处理平台.该平台在现有基于无共享云架构的分布式计算模型基础上引入时空数据对象作为一级实体,并通过提供分布式时空数据对象模型及索引的支持来提供对时空型交通感知数据模式的直接支持,借此实现跨应用的交通感知数据共享,为承载多样化的交通感知数据处理应用提供集成化的支持.

1 需求分析与相关工作

1.1 交通感知数据集成处理需求

智能交通系统中涉及的交通感知数据主要包括实时感知数据流和历史交通感知数据 2 类.此外,基于交通感知数据的交通业务应用需求也是多样化的,它们往往需要集成来自不同系统的多元数据(如车辆 GPS 数据、交通流量数据、车牌识别数据、路网数据、车辆备案数据、嫌疑车/黄标车/公交车数据等以及基于初始交通感知数据计算得出的交通流量数据、道路旅行时间数据等 2 次交通感知数据)来支撑不同部门、不同单位的业务需求,例如实时采集的道路车辆车牌识别数据可以支撑套/假牌、区间超速、黄标车等违章车辆自动甄别业务,交通流参数、路段

旅行时间等实时路况计算业务,黑名单车辆布控、伴随车辆分析等刑/技侦业务.由此可以看到交通感知数据及处理需求具有典型的"多源性和多样性".为此,迫切需要提供有效的数据集成与共享方式来支持在多样化的交通应用中使用这些海量的交通感知数据.此外,智能交通系统中的交通感知数据主要围绕车辆、道路、监测点及交通设施等核心交通对象产生,具有典型的对象相关特性,如某车辆一天的行驶轨迹数据、某路段各时段的车流量数据等.同时这些交通感知数据又都涉及时间和空间2个维度的属性,往往可以看作是交通对象在特定道路空间(即交通路网)的时间序列数据,具有一定的时空关联关系和分布特征,大多交通应用也都是围绕着这些具有时空属性的交通感知数据展开的.我们把交通感知数据的上述特性称为"对象相关性和时空相关性".因此,需要充分利用这些特性来优化交通感知数据在分布式环境中的并行划分和组织管理.

另一方面,根据对上述交通感知数据处理需求的分析,我们可以将交通感知数据计算任务按照4类计算特征划分,如表1所示.这4类计算任务从流计算和批计算的不同角度对交通感知数据处理提出了集成化的支撑需求.

Table 1 Description of Types of Traffic Sensor Data Computing

表1 交通感知数据计算类型描述

Types of Traffic Sensor Data Computing	Characteristic	Typical Computing Tasks
Offline computing on history data	Aggregation on batch data	Accompanying vehicles mining
Online computing on real-time data	Single tuple's queries from basic data on stream	Blacklist car detection, fake license plate detection
	Single tuple's comparison from windowed data on stream	Cloned license plate detection, over-speed vehicle discovery
	Multiple tuples' aggregation from windowed data on stream	Real-time traffic flow counting, real-time travel time computing

1.2 相关工作

近年来,与上述需求相关的研究工作涉及多个研究方向,包括时空数据管理、分布式大数据处理、流数据处理及流式计算等.其中,时空数据管理是数据库领域针对位置或形状随时间而变化的各类时空对象进行管理的一个研究方向,其主要内容涉及时空对象数据的建模、索引与查询,可以支持对交通系统中车辆、人员等移动对象的管理[3-4].但传统时空数据模型在空间数据模型基础上引入时态信息来对时空对象建模,但这类模型只适合对小规模、低频率时空数据进行建模,而且其上的查询处理主要基于SQL语言,缺少对具有海量交通数据复杂处理的支持,特别是现有的时空数据管理方法大多局限于单机环境,在应对大规模、不间断的交通数据方面也缺乏分布式环境下的有效解决方案[5-6].

以MapReduce[1]为代表的一系列分布式计算技术为大数据处理带来了新的思路,其强调在大量低端通用服务器的无共享集群云架构下建立面向海量数据处理的并行计算模型和可伸缩环境,遵循了BASE(basic availability, soft state and eventual consistency)设计原则[7]来获得大规模数据处理系统的可伸缩性和可用性,并在以搜索为主的大规模互联网数据处理等应用中得到了良好的验证.然而,MapReduce及其相关扩展工作都属于对持久化数据的批处理方式,在每次处理时都需要初始化运行环境,同步地在Map和Reduce阶段载入、处理大规模数据,并在节点间传输大量数据以及进行Map/Reduce任务的同步.按照这种方式处理不间断到达的交通数据很难满足其流式计算中的实时性需求[8].

在流数据处理方面,数据流管理系统(data stream management system, DSMS)被设计用来满足持续到达的数据序列的处理需求[9].由于数据流的持续性和无限性,传统的数据流系统受数据采集速度、传输带宽、内存容量和计算能力等因素的限制不可能处理数据流的所有记录(即只能支持有限历史数据),因而一般采用滑动窗口模型(sliding window model)或界标模型(landmark model)来划定处理边界,或者通过抽样(sampling)或概要数据方式形成一个子集代表数据全集,侧重于针对相对小规模的数据进行处理[10].此外,数据流系统支持处理类型大多为相对简单的查询(如查找特定数据记录或序列模式等),不支持复杂分析与计算,处理后的数据流也会被丢弃,难以再次利用[11].因此,在面对多样化的海量交通感知数据处理需求时它们不得不面对处理能力和扩展性方面的问题.

从2010年开始,流式计算逐渐成为大数据处理的一个研究热点,也出现了Storm[12],Spark Streaming[13]等知名的流式计算系统,该方向可以看作是数据流系统在大数据新背景下的一个新发展[2].在该计算模型中,数据以流和批等不同形式到达,多个处理单元中的计算程序对数据进行不间断的实时计算和传递.一个典型的流式计算模型可以看作一系列算子(点)和数据流(边)组成的数据流图(表示为有向非循环图),然而这种高度抽象的计算模型在面向交通数据处理需求时由于不能提供对交通感知数据模式的直接支持,在交通数据处理应用研发中需要进行大量重复的复杂开发工作.

2 基于时空数据对象的交通感知数据处理模型

2.1 模型描述

按照1.1节所述的处理要求,面对高速到达的实时感知数据及大规模历史感知数据形式存在的交通感知数据,如何利用其时空相关及对象相关等特征进行有效表示和组织以支持在多样化的感知数据处理应用中共享使用是海量交通感知数据集成处理的关键所在.

为了在已有状态的计算任务为核心的流式计算模型上加强对交通感知数据模式的直接支持,我们提出一种基于时空数据对象的感知数据处理模型,其核心特点是引入适于并行化的时空数据对象并允许在同一数据对象上执行不同计算任务来解决引言提到的交通感知数据集成处理所面临的3个问题.该模型将时空数据对象作为交通感知数据集成处理中的一级实体,时空数据对象是一种易于并行划分及动态维护的内存数据对象,通过对其管理待处理的数据并记录处理的状态,也可实现跨计算任务的共享.

下面我们给出如图1所示的基于时空数据对象的交通感知数据处理模型中的核心概念.

1) 交通感知数据记录(data record, DR).DR记录相关交通对象在不同时间、空间属性下的状态,可表示为〈key,value〉对形式的数据单元,其中时间、空间及对象标识常被用作交通感知数据的key.

2) 实时感知数据流(data stream, DS).DS表示实时获取到的感知数据记录序列,它往往以流的形式进入系统.

3) 历史感知数据(history data, HD).HD表示长时间积累的感知数据记录集,它来自对实时感知数据进行持久化存储的数据集.

4) 时空数据对象(data object, DO).DO表示可供感知数据计算任务使用的感知数据记录集合,该集合中感知数据记录的key都满足一定的约束,比如给定的时间或空间范围.

时空数据对象可以方便地表示对参与计算的交通感知数据的并行化划分,按照同一种划分规则得到的时空数据对象可被归入同一组管理.同时,不同的时空数据对象可以在不同计算任务(包括跨应用的计算任务)间共享.

5) 计算任务(computing task, CT).CT表示感知数据处理中的基本单元,每个计算任务可以指定需要计算的时空数据对象组,计算结果可以形成新的时空数据对象.

根据同组时空数据对象的划分情况(即根据key划分成的时空数据对象数目),每个计算任务在执行时会产生多个实例,每个实例处理相应的时空数据对象.

与典型的流式计算模型一样,计算任务可以放置

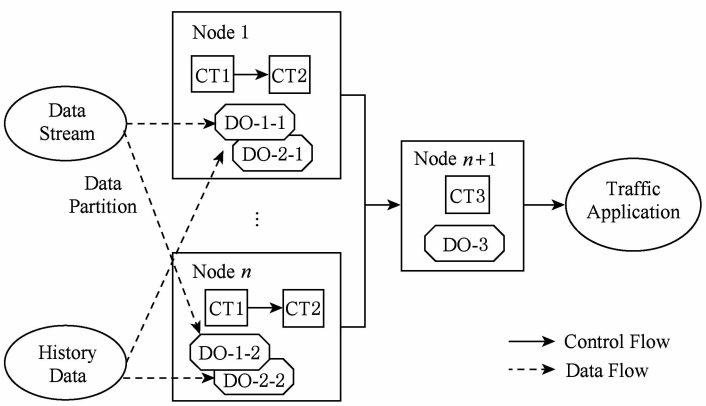

Fig. 1 Traffic sensor data processing model based on temporal data objects.

图1 基于时空数据对象的交通感知数据处理模型

到不同的节点上,按照处理逻辑可以构成一个计算任务间的有向无环图.

2.2 多维时空数据对象组织模式

根据 2.1 节所述模型,适于并行化组织的时空数据对象成为交通感知数据处理模型中除计算任务外的另一核心要素.为了支持时空数据对象管理,需要提供一种适于分布式环境下并行处理的时空数据对象组织方式.

通过对交通感知数据及其处理需求特征归纳,交通感知数据主要从时间、空间、对象 3 个维度进行划分,相关交通业务基本围绕这 3 个维度交叉进行数据处理,如围绕车辆对象的车辆监管业务、围绕路段/路径空间属性的实时路况业务以及围绕时间属性的不同周期交通数据统计业务.因此,我们首先从这 3 个维度对时空数据对象进行划分,具体可以根据不同类型交通感知数据处理需求并通过定义这 3 个维度上的 Hash 函数来完成不同节点下的数据对象划分及分布方案.同时,为了提供细粒度的数据共享支持,还可以对每个维度下划分得到的数据对象从其他维度进一步分解,为此我们在节点内采用 B 树结构来对其进行组织.这样,可以形成一种基于 Hash B 树的分层次索引结构以组织每个维度划分得到的时空数据对象.图 2 给出了首先从时间维度进行划分的时空数据对象组织结构.数据对象中包含的具体交通感知数据记录可以在树的叶节点按照时间顺序以链表形式存储;同时,还可以先从空间角度或者车辆对象角度进行数据划分,进一步再采用 B 树结构对划分后的数据按照其他维度进行组织.

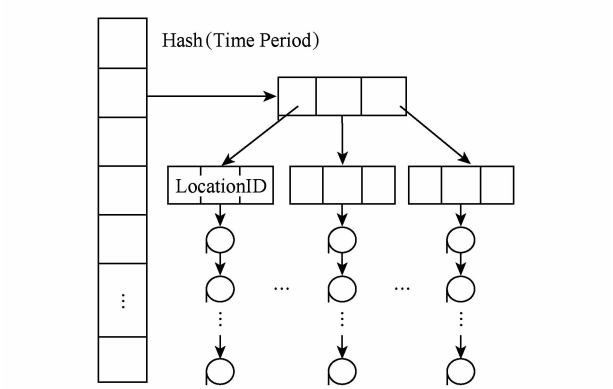

Fig. 2 Structure of spatio-temporal data objects divided from the time dimension in the beginning.

图 2 先从时间维度进行划分的时空数据对象组织结构

根据上述时空数据对象组织结构,可以看出对 Hash 表的任意划分能形成对 Hash B 树的划分,因此该结构具有较好的可划分性,适于分布环境下的并行划分.同时,由于用作 Hash 键的时间、空间和交通对象值都可预测并具有唯一的 Hash 值,因此可以通过分配足够的 Hash 表项使得该结构下的插入和查找操作的复杂度仅为 $O(1)$.

2.3 时空数据对象核心操作

由于时空数据对象包含参与计算的感知数据及计算过程中涉及的中间状态数据,它们主要通过对原始感知数据(包括实时感知数据和历史感知数据)进行划分及执行特定的操作得到或产生,而这些数据对象可以被不同的计算任务共享使用.根据 1.1 节对交通感知数据处理需求的归纳,针对时空数据对象的创建、划分、组织维度重组、键值变换、数据更新等需求,我们设计了表 2 所列的 5 类时空数据对象核心操作.这些不同类别的操作可以用于支持计算过程中涉及的对共享数据对象的处理,以满足不同计算任务对时空数据对象的处理需求.此外,用户也可以指定计算过程中新产生的时空数据对象是否需要共享.

Table 2 Core Operations Available for Spatio-temporal Data Objects

表 2 时空数据对象核心操作

Operation Type	Meaning
Operation for object creation: $load(f:F\rightarrow O)$	Load the spatio-temporal data object from the original data
Operation for object partition: $partition(p:Partitioner(O.k))$	Partition the data from the temporal, spatial and object dimensions
Operation for object regrouping: $regroup(f:O_{d_1}\rightarrow O_{d_2})$	Regroup the data from one dimension to another
Operation for object transformation: $transform(f:(k_1,v_1)\rightarrow(k_2,v_2))$	Transform the objects from one kind of key-value pair to another kind of key-value pair
Operation for object edition: $edit(f:Editor(O))$	Modify the data object, such as appending, deleting, or updating the object attributes

上述操作均是以时空数据对象为中心的操作,它们针对交通感知数据的处理语义明确.这些操作的设计思路参考了 1.2 节所述的 Storm 和 Spark 等相关工作.其中,*load* 操作借鉴了 Storm IBlot 接口中的 *prepare* 操作,在核心计算业务前准备数据,专注于感知数据从持久化存储中读取并加载至内存;*transform* 操作和 *edit* 操作,则参考了 Spark 中的 *transform* 操作和 *action* 操作,前者专注于存在键变化的对象,后者专注于存在值变化的对象;而 *partition* 操作和 *regroup* 操作则提供对交通感知数据在时间、空间、对象 3 个维度不同组织方式的支持.

3 交通感知数据集成处理平台及应用

3.1 平台实现

根据第 2 节的处理模型,我们实现了云架构下的海量交通感知数据集成处理平台.平台由 1 个控制节点和多个处理节点集群组成,其中,控制节点负责时空数据对象和计算任务的调度、监控和容错处理,具体包括时空数据对象和计算任务元数据管理、状态信息收集及生命周期控制以及集群节点协调控制.任务节点负责接收外部实时感知数据和历史感知数据并创建时空数据对象、接收动态部署计算任务并执行任务、向控制节点定期报告时空数据对象和计算任务信息.

在具体实现中,针对 1.1 节归纳的 4 类交通感知数据处理类型,可以看到这些典型的计算任务在时空数据被接入后往往需要经历多类业务计算,也即需要多个计算任务的处理.因此,如何组织时空数据对象并分配至计算任务以及如何调度各计算任务是系统实现必须考虑的重要问题.

1) 在数据组织方面.我们采用基于外存的多维倒排索引来组织海量离线的时空数据,即在指定的时间阈值下(如 1 d 或 1 周),分别构建数据在时间、空间和对象上的倒排索引,同时根据配置的副本度,在系统的计算节点上均匀布局.针对实时在线的时空数据,则采用 2.2 节的 Hash B 树结构,在内存中以轻量级有限空间的方式组织维护.

2) 在数据分发和任务调度方面.我们分别针对表 1 所述的计算类型,将计算任务分类并按照所设定的计算阈值(计算频率、窗口大小等)划分任务关于获取数据的需求,并对相同需求的计算分配相同的数据对象,对离线计算任务采用尽可能靠近数据位置的分配方式.

系统通过扩展 Hadoop MapReduce 实现,改进的主要内容包括:调整 JobTracker 中任务调度模式为时空数据对象相关的调度,并增加了时空数据对象的调度功能和状态监控功能;在 TaksTracker 中增加了对时空数据对象内存数据结构的支持;去除了 TaksTracker 中 Map 和 Reduce 任务执行过程中对 HDFS 文件系统的读写,中间结果改为以时空数据对象形式在内存中管理.关于系统实现的更多详细细节可参考文献[14-15].

3.2 应用实例

上述平台已被应用到基于车牌识别数据的城市车辆管控系统项目中,本节通过其中的城市道路旅行时间计算和伴随车辆分析 2 个交通应用来展示海量交通感知数据集成处理平台的应用方法和效果.

1) 旅行时间计算

路段旅行时间作为城市交通出行信息的关键指标,可以直接用来评判城市道路的运行状况和拥堵水平,有效的旅行时间监测与分析也可以为城市路网规划、城市道路交通管理与控制、公众出行路线选择提供合理依据.旅行时间计算问题可以看作是:给定旅行时间计算时间周期和一定时间范围内的车牌识别数据集,对受测道路路网中的所有路段求其在给定时间范围不同时间区间上的路段旅行时间.对于不同时间区间上的路段旅行时间,可以通过计算其在不同时间区间上的所有单车旅行时间,并进一步取中位数方法求得最终结果.

针对旅行时间计算处理逻辑中涉及的车牌识别数据加载、单车旅行时间计算、单车旅行时间中位数查找 3 个子任务,可采用 2 次 MapReduce 迭代处理.

第 1 次 MapReduce 处理中的 Map 函数完成车牌识别数据的读入及划分和如 2.2 节所述的数据结构组织,具体可通过 *load* 操作来进行装载并得到 Hash B 树结构的数据对象,进一步利用 *partition* 操作进行划分.具体地,首先从时间维度,为支持时间区间划分采用时间区间作为 key,相同时间区间的车牌识别数据在 Hash 表的同一项中用 B 树组织;其次,监测点作为空间划分基础被用来组织最终的车牌识别数据,每个监测点的车牌识别数据在 B 树的叶节点用链表按照时间顺序进行组织,最终以形如〈key:时间区间+监测点,value:车牌号+时间〉的键值对组织数据;Reduce 函数利用 *transform* 操作形成指定时间和路段下的单车旅行时间数据对象,即将上述 Hash B 树中叶节点的监测点识别数据变换为特定车辆在不同路段的旅行时间数据,其可根据路段信息对中间结果按照车牌号进行重组得到形为〈key:时间区间+路段(监测点 1、监测点 2),value:车牌号及时间点 1 和时间点 2〉的键值对.

第 2 次 MapReduce 处理中的 Map 函数利用 *edit* 操作计算单车路段旅行时间而不做任何数据变换,Reduce 函数同样只进行所有单车旅行时间中位数查找,最后形成最终结果数据对象,即得到形如〈key:时间区间+路段,value:旅行时间值〉的键值对.〈201310020830+LD0014[JCD06,JCD07],360〉是一个路段旅行时间计算最终结果示例,该示例表示

在 2013-10-02T8:30—8:45 的时间区间,0014 号路段(即监测点 JCD06 到监测点 JCD07 的路段)的旅行时间为 360 s.

2) 伴随车辆分析

伴随车辆分析主要是在海量车辆监控数据基础上分析车辆移动对象轨迹间的相似关系,可以协助公安民警办案、犯罪嫌疑车辆查询,也可以为城市道路规划提供参考,具有重要的实际意义. 伴随车辆分析问题可以简单地理解为:给定点伴随时间阈值、轨迹相似度阈值和轨迹长度阈值,利用已有车辆监控数据集,找出在给定的时间范围内所有具有伴随关系的车辆相似轨迹集合的查询分析.

伴随车辆分析过程可分解为轨迹分析与筛选、点伴随判定、轨迹相似性计算 3 次 MapReduce 迭代处理.

① 车辆轨迹分析与筛选

该步骤读取原始车牌识别数据,并将数据按时间属性进行划分,然后按照车辆进行数据对象组织,可以通过对旅行时间计算中时空数据对象施加 *regroup* 操作获得. 在此基础上,通过一次 MapReduce 运算可统计得出所有车辆在给定时间段内的有效轨迹数据. 其中,Map 函数将调用 *transform* 操作得到形如〈key:车牌号,value:时间和监测点〉的数据对象;Reduce 函数通过 *transform* 操作将相同 key 的 Map 函数输出进一步进行合并形成单车轨迹数据对象,并对单车轨迹长度小于给定轨迹长度阈值的轨迹数据进行删除.

② 点伴随判定

点伴随判定主要读取第 1 次 MapReduce 过滤后的数据对象,并通过第 2 次 MapReduce 进行计算返回具有点伴随关系的车辆对,即在同一监测点邻近时间范围内出现的 2 个车辆. 在此次 MapReduce 中,首先在 Map 函数中直接读取一次 MapReduce 得到的数据对象并通过 *transform* 操作将其转换为形如〈key: 监测点 ID,value: 监测时间和车牌号〉的数据对象,然后传递给 Reduce 函数;经过同一个监测点的识别数据会发给同一个 Reduce 函数处理,Reduce 函数对接收的车辆识别数据,按监测时间先后排序,随后开始点伴随计算. 点伴随计算从车俩轨迹链表头结点开始,取第 1 个监测数据和之后的数据比较,判断 2 个时间差是否小于点伴随时间阈值,如果满足则直接将 2 个车牌号输出到结果,其中 key 为车牌号 1 和车牌号 2 组合,value 为固定值 1.

③ 轨迹相似性计算

通过第 3 次 MapReduce 完成轨迹相似性计算,即对第 2 次 MapReduce 得到的点伴随结果根据车辆轨迹进行统计,然后按照轨迹相似度计算公式判定伴随车辆. 在此次处理中,Map 函数直接读取第 2 次 MapReduce 得到的结果并直接输出. 其中,输出结果的 key 为具有点伴随关系的 2 个车牌,value 值为固定值 1;Reduce 函数接收 Map 函数输出的键值对,利用 *edit* 操作计算 key 相同的数目,然后根据车辆轨迹计算轨迹相似度,并返回满足轨迹相似度阈值的车牌对.

4 实验与分析

4.1 实验设置

实验环境采用的是在 5 台服务机上搭建的集群环境,并在其上部署基于 Hadoop 扩展了中间结果缓存和时空数据对象处理机制后的平台实现. 其中,Master 节点配置为 4 核 CPU、4 GB 内存,Master 节点同时也被当作计算节点;另外 4 台 Slave 节点配置为 2 核 CPU、4 GB 内存,作为计算节点. 此外,每台服务器的有效容量为 80 GB,集群总存储容量为 400 GB. 实验中采用的数据为北京市 1 000 多个带识别功能的道路摄像头采集到的真实车牌识别数据.

为了从性能对比、关键参数影响和扩展性 3 方面对基于本文系统实现的旅行时间计算和伴随车辆分析功能进行验证分析,我们设计了 2 组实验:

实验 1. 选取北京市 2012-11-01—2012-11-20 期间 20 d 的真实车牌识别历史数据(约 1 亿条)作为原始计算数据集,分别测试 5 min,15 min,1 h 这 3 个时间周期下路段旅行时间计算的性能、关键参数影响和扩展性情况. 同时,还选取直接在 Hadoop 平台上实现而并未对车牌识别数据进行特别处理的旅行时间计算方法(LMR 方法)[16] 作为比较对象,与本文基于本文平台实现的旅行时间计算方法(CMR 方法)进行性能比较.

实验 2. 选取北京市 2012-11-13 全天采集到的真实车牌识别数据(这组车牌识别数据中涉及的车牌数量(即车辆数)约 230 万,道路监测点为 1 794 个),数据记录为 970 万余条,大小 0.94 GB,对本文平台下实现的伴随车辆查询方法(MPST 方法)进行性能及关键参数影响测试. 同时,还利用同组实验数据测试了文献[17-18]中提出的单机环境下的伴随车辆查询方法(即 TMN-Tree 方法和 ACR 方法)以进行对比.

4.2 实验结果分析

1) 性能对比分析

① 旅行时间计算性能

从图 3 可以看出,随着参与计算的车牌识别数据集数据量的增加,2 种计算方法的计算时间均呈线性增加;但 CMR 方法在计算效率上比 LMR 方法有较高的提升,并且 CMR 方法受时间周期差异的影响比 LMR 方法小很多,5 min,15 min,1 h 这 3 个不同时间周期下计算时间的差异均在 100 s 以内.

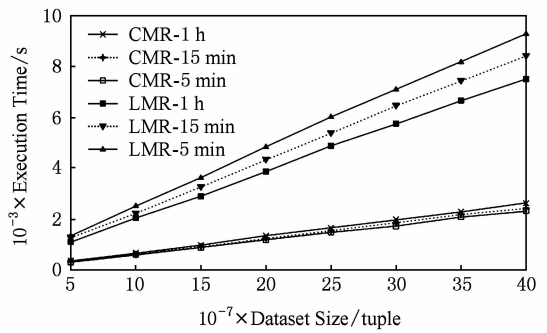

Fig. 3 The impact of vehicle license plate recognition data on the performance of travel time computing.

图 3 车牌识别数据量对旅行时间计算性能的影响

此外,从图 3 还可以看到,LMR 方法在计算时间周期越短(即时间段划分粒度越细)时计算时间越长,5 min 周期下的计算时间最长;而 CMR 方法恰好相反,在计算时间周期变短时计算时间反而会略微减少,5 min 周期下的计算时间最短.究其原因,主要因为当计算时间周期较小时,需要计算旅行时间的时间区间会大幅增加,使得 Hadoop 运行态中的 Map 任务和 Reduce 任务大增并带来较大的任务执行调度代价.传统 LMR 方法由于未根据车牌识别数据特征进行划分优化,执行中需要大量的 Map 任务和 Reduce 任务间的同步等待,从而使得小时间周期下的计算时间变长;而 CMR 方法由于通过先时间后空间的划分模式使得旅行时间计算过程可以避免 Map 任务和 Reduce 任务间不必要的数据依赖,这样单个 Map 任务和 Reduce 任务一次处理的数据量(受时间区间大小影响)成为影响计算时间的主要因素,因此使得短时间周期下的计算时间反而变短.

② 伴随车辆分析性能

我们分别针对 1 h,3 h,8 h,24 h 不同时间范围内的车牌识别数据进行了伴随车辆查询测试,其中采用的 1 h 识别数据量约为 41 万条,3 h 的识别数据量约为 155 万条,8 h 的识别数据量约为 370 万条,全天 24 h 的识别数据量约为 970 万条.在具体实验中,分别使用上面提到的 3 种不同的伴随车辆查询方法对相同时间范围的数据进行查询,其中点伴随时间阈值取值为 1 min,轨迹长度阈值取值为 10,轨迹相似度阈值取值为 75%.实验结果如图 4 所示:

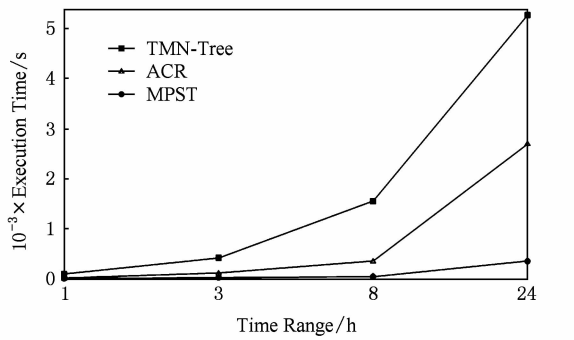

Fig. 4 Comparison of accompanying cars query time under different time ranges.

图4 不同时间范围的数据规模下的伴随车辆查询时间对比

由图 4 可见,在 1 h 的数据规模下本文 MPST 方法性能略高于 TMN-Tree 方法和 ACR 方法,这说明在数据规模较小的情况下性能提升还相对不明显;但在 3 h,8 h,24 h 数据规模时,MPST 方法通过利用并行化的方式可以在分布式环境下获得较好的查询性能提升,大幅提高查询效率,查询处理性能相对于传统单机的查询算法最高可提高 10 倍以上.

2) 关键参数影响分析

① 旅行时间计算

我们针对旅行时间计算中不同受测路段规模下的计算性能进行了实验.从图 5 可以看出,随着受测路网中路段数目的增加,本文 CMR 计算方法的计算时间基本平滑,而 LMR 方法则在路段数增大时表现出计算时间线性增长的趋势.这表明 CMR 计算

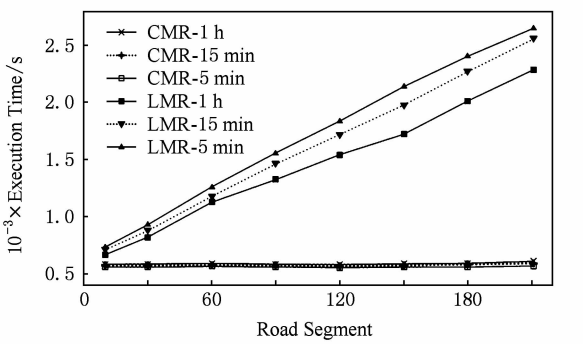

Fig. 5 The impact of the number of road on travel time computing.

图 5 路段数对旅行时间计算的影响

方法的计算性能基本不受路段数目的影响,当我们增加受测路网规模(即增加路段数)时,并不影响旅行时间计算的计算性能.主要原因在于,路段数在旅行时间计算中的主要影响是会增大计算中间结果的规模,CMR 方法由于采用时空数据对象结构优化了中间结果的处理,因此其受路段数变化的影响较小.

② 伴随车辆分析

我们还针对本文方法中用于点伴随判定的时间间隔阈值和轨迹相似度阈值的不同对查询性能和查询结果集大小的影响进行了测试,实验在 24 h 的车牌识别数据(千万记录量级)上进行.

由图 6 可以看到,增加用于点伴随判定的时间间隔阈值会极大影响查询性能,相应的查询时间在同样的轨迹相似度阈值下会接近线性增长.而轨迹相似度阈值的提高反而会降低相应的查询时间,这主要是因为相似度要求高反而会减少查询处理过程中中间结果数据的规模,从而降低查询时间.

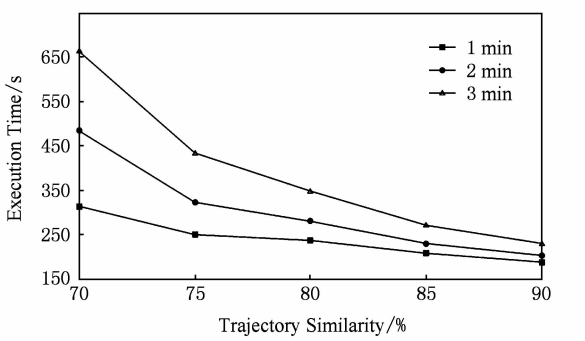

Fig. 6 Accompanying cars query performance changes under different similarity thresholds.

图 6 不同相似度阈值下的伴随车辆查询性能变化

图 7 则给出了不同阈值下查询结果集大小的实验结果.由图 7 可以看出,降低相似度阈值和放大用于点伴随判定的时间间隔阈值都带来结果集的增大;但当相似度阈值降低到 80% 及以下时,结果集数据量开始明显增大,并且不同时间间隔阈值下结果集数据量的差距将变得愈发突出.

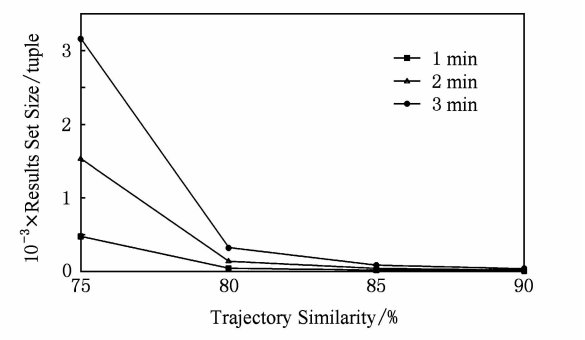

Fig. 7 Results set size change of accompanying cars query under different similarity thresholds.

图 7 不同相似度阈值下伴随车辆查询结果集大小变化

3) 扩展性分析

在扩展性方面,由于系统采用了无共享集群的云架构模式并在 Hadoop 基础上实现,对旅行时间计算和伴随车辆分析 2 个不同应用具有相同的扩展性效果.这里我们仅给出旅行时间计算的扩展性实验结果,如图 8 所示.从图 8 可以看出,随着计算节点数的增加,计算时间会逐步降低,并且可以看出细粒度时间周期的计算时间更少.这表明本文平台的实现并未影响原 Hadoop 架构的扩展性.

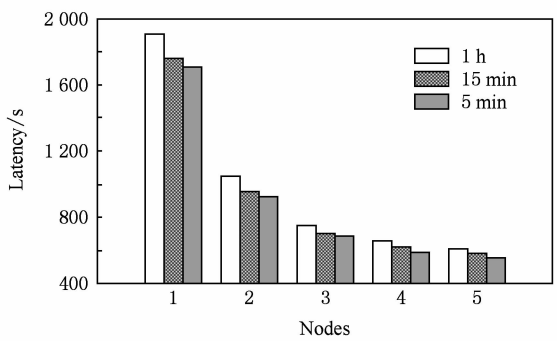

Fig. 8 The impact of computing nodes on travel time calculations.

图 8 计算节点数对旅行时间计算的影响

5 结束语

本文以交通这一特定领域为牵引,针对该领域下海量交通感知数据的时空、对象相关及其处理需求的多样化特征,提出一种基于可共享、易并行化的时空数据对象的感知数据处理模型,并在此基础上设计并实现了交通感知数据集成处理平台.该平台可以为基于交通感知数据的集成处理应用提供从数据组织、获取及计算方面统一的集成化支持,并通过数据的共享实现多样化交通感知数据处理应用的整体优化.实际的应用效果和相关实验验证表明,该平台相对于传统的海量交通感知数据处理系统在性能及扩展性等方面都具有一定的优势.下一步的工作包括处理模型的理论分析、时空数据对象的动态管理以及更加全面的实验测试分析.

参 考 文 献

[1] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113

[2] Sun Dawei, Zhang Guangyan, Zheng Weimin. Big data stream computing: Technologies and instances [J]. Journal of Software, 2014, 25(4): 839-862 (in Chinese)
(孙大为, 张广艳, 郑纬民. 大数据流式计算:关键技术及系统实例[J]. 软件学报, 2014, 25(4): 839-862)

[3] Meng Xiaofeng, Ding Zhiming. Mobile Data Management: Concepts and Techniques [M]. Beijing: Tsinghua University Press, 2009 (in Chinese)
(孟小峰, 丁治明. 移动数据管理:概念与技术[M]. 北京: 清华大学出版社, 2009)

[4] Zhou Aoying, Yang Bin, Jin Cheqing, et al. Location-based services: Architecture and progress [J]. Chinese Journal of Computers, 2011, 34(7): 1155-1171 (in Chinese)
(周傲英, 杨彬, 金澈清, 等. 基于位置的服务:架构与进展[J]. 计算机学报, 2011, 34(7): 1155-1171)

[5] Sergio I, Eduardo M. Location-dependent query processing: Where we are and where we are heading [J]. ACM Computing Surveys, 2010, 42(3): Article 12

[6] Ding Zhiming, Gao Xu. A database cluster system framework for managing massive sensor sampling data in the Internet of things [J]. Chinese Journal of Computers, 2012, 32(6): 1175-1191 (in Chinese)
(丁治明, 高需. 面向物联网海量传感器采样数据管理的数据库集群系统框架[J]. 计算机学报, 2012, 32(6): 1175-1191)

[7] Pritchett D. BASE: An acid alternative [J]. Queue, 2008, 6(3): 48-55

[8] Meng Xiaofeng, Ci Xiang. Big data management: Concepts, techniques and challenges [J]. Journal of Computer Research and Development, 2013, 50(1): 146-169 (in Chinese)
(孟小峰, 慈祥. 大数据管理:概念、技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-169)

[9] Abadi D J, Ahmad Y, Balazinska M, et al. The design of the Borealis stream processing engine [C] //Proc of the 2nd Biennial Conf on Innovative Data Systems Research. New York: ACM, 2005: 277-289

[10] Motwani R, Widom J, Arasu A, et al. Query processing, resource management, and approximation in a data stream management system [C] //Proc of the 1st Biennial Conf on Innovative Data Systems Research. New York: ACM, 2003: 176-187

[11] Golab L, Tamer M. Issues in data stream management [J]. SIGMOD Record, 2003, 32(2): 5-14

[12] Toshniwal A, Taneja S, Shukla A, et al. Storm @ Twitter [C] //Proc of the 33rd ACM Int Conf on Management of Data (SIGMOD 2014). New York: ACM, 2014: 147-156

[13] Zaharia M, Das T, Li H, et al. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters [C] //Proc of the 4th Conf on Hot Topics in Cloud Computing. Berkeley, CA: USENIX Association, 2012: 181-184

[14] Qi Kaiyuan, Zhao Zhuofeng, Fang Jun, et al. Real-time processing for high speed data stream over large scale data [J]. Chinese Journal of Computers, 2012, 35(3): 477-490 (in Chinese)
(亓开元, 赵卓峰, 房俊, 等. 针对高速数据流的大规模数据实时处理方法[J]. 计算机学报, 2012, 35(3): 477-490)

[15] Qi Kaiyuan, Han Yanbo, Zhao Zhuofeng, et al. MapReduce intermediate result cache for concurrent data stream processing [J]. Journal of Computer Research and Development, 2013, 50(1): 111-121 (in Chinese)
(亓开元, 韩燕波, 赵卓峰, 等. 支持高并发数据流处理的MapReduce中间结果缓存[J]. 计算机研究与发展, 2013, 50(1): 111-121)

[16] Zhang Shuai, Zhao Zhuofeng, Ding Weilong. Urban road trip time measured calculation based on MapReduce [J]. Computer & Digital Engineering, 2014, 42(9): 1542-1546 (in Chinese)
(张帅, 赵卓峰, 丁维龙. 基于MapReduce的城市道路旅行时间实测计算[J]. 计算机与数字工程, 2014, 42(9): 1542-1546)

[17] Chang J, Song M, Um J. TMN-tree: New trajectory index structure for moving objects in spatial networks [C] //Proc of the 10th Int Conf on Computer and Information Technology. Piscataway, NJ: IEEE, 2010: 1633-1638

[18] Zhao Xinyong, An Shi. Research on accompanying cars recognition in practical application [J]. Journal of Transportation Systems Engineering and Information Technology, 2012, 12(3): 36-40 (in Chinese)
(赵新勇, 安实. 伴随车检测技术应用研究[J]. 交通运输系统工程与信息, 2012, 12(3): 36-40)

Zhao Zhuofeng, born in 1977. PhD and associate professor. Senior member of China Computer Federation. His current research interests include streaming computing, Internet of things technology and cloud computing.

Ding Weilong, born in 1983. PhD and assistant professor. Member of China Computer Federation. His main research interests include real-time data processing, distributed system and cloud computing.

Han Yanbo, born in 1962. PhD, professor and PhD supervisor. Senior member of China Computer Federation. His current research interests include cloud computing, big data science and service computing.