doi:10.3969/j.issn.1672－0504.2019.04.001

基于Spark/HBase的交通流数据存储及索引模型探讨

李 欣

(河南财经政法大学中原经济区"三化"协调发展河南省协同创新中心,河南 郑州 450046;
河南财经政法大学资源与环境学院,河南 郑州 450046)

摘要:为了对海量增长的交通流数据进行处理和管理,需要基于大数据框架设计更加高效的数据存储及索引模型,以满足智能交通应用的需求。该文设计了基于Spark/HBase的系统架构以及基于混合时空编码行键和动态扩展属性列族的交通流数据存储及索引模型,并在此模型基础上,通过语义解析、时空行键索引查询、并行属性条件过滤实现交通流大数据高效语义查询。对比实验证明,该文设计的交通流大数据并行处理框架在清洗、索引和存储数据时运算高效,构建的混合时空编码行键索引时空权重均衡,能够实现更加高效的交通流大数据访存管理,可为智能交通应用提供技术基础。

关键词:Spark;HBase;交通流;时空编码行键;语义查询

中图分类号:P208;TP311.13 **文献标识码**:A **文章编号**:1672－0504(2019)04－0001－08

0 引言

随着空间信息服务的不断进步,智能交通信息服务呈现出大众化、广域化、实时化的特点,面对海量增长的交通流时空数据[1],应用系统必须快速完成数据处理和存储任务;而并行处理和内存计算技术[2]可以充分发挥计算机集群的数据处理能力,突破传统空间信息系统的技术瓶颈,为高性能实时交通信息服务提供技术途径。传统的交通信息系统在存储数据时,主要是将交通流数据作为空间数据的一种应用特例,对空间信息服务系统[3]中的数据模型进行有针对性的优化改造,进而实现数据的存储和管理;而在数据管理过程中,运算框架、数据模型和数据索引会对处理和查询效率产生较大影响。

在运算框架方面,常利用并行计算提高数据管理的效率。已有研究成果中,利用MPI框架的有基于碟形并行神经网络算法的交通诱导系统[4]、基于空间域划分方法的微观交通仿真系统[5]等,虽然MPI框架结构简单、移植性强,但并行进程的容错性差、通信开销大,使其用于交通信息服务系统时的开发难度大,数据并发处理效率较低;利用MapReduce框架的有利用本地阶段化流水线和中间结果缓存技术对实时处理能力进行改进的城市车辆数据实时采集与处理系统[6]、利用读写开销估算和缓冲区信息对并发读写性能进行优化的车辆监管系统[7]等,MapReduce框架的主要问题在于框架的磁盘读写机制造成了频繁的IO访问,同时其计算任务复杂,需要反复分配运算资源,必须使用更高效的新框架处理实时海量增长的交通流数据,以满足智能交通应用的需求。本文拟利用Spark框架的内存计算和迭代计算能力,实现交通流大数据的高效清洗、索引和入库。

在数据模型方面,传统的移动目标数据模型(如基于动态属性的移动目标位置模型[8]、基于时空几何体的对象模型[9,10]、基于事件的时空数据模型[11]等)都可用于表达移动对象或轨迹的时空信息,但其主要针对的是传统的较小规模的交通数据管理任务。在此基础上,一些学者基于分布式的理念,对交通信息系统数据模型进行了改进,如基于分布式时空数据对象及其索引的交通感知数据处理平台[12]、基于时空划分旅行时间的并行计算模型[13]、基于窗口阈值和分布策略优化的交通流数据DRTS实时存储方法[14]、基于数据切分重组和热点缓存机制的交通数据优化管理系统[15]等。以上研究成果虽然通过并行化的方法进行了优化,但仍然基于传统的关系型数据库(如Oracle、MySQL等)构建,存在读写延时高、数据结构固定、字段扩展能力差的问题,其吞吐和扩展能力在很大程度上依赖于底层的数据库管理系统(DBMS),将其用于管理交通流数据仍然受到较大限制。本文拟利用HBase分布式数据库的行

收稿日期:2018－10－08; 修回日期:2018－12－24
基金项目:国家自然科学基金项目(41771445、41871159);河南财经政法大学博士科研基金项目(800257)
作者简介:李欣(1981－),男,博士,讲师,主要从事地理信息系统理论与实践应用研究。E－mail:lixin992319@163.com

键索引能力和属性列族动态扩展能力，建立适用于智能交通应用的数据模型，为实现交通流大数据的高效存储和查询奠定基础。

在数据索引方面，最为经典的是格网索引[16,17]、R-tree 索引[18]、四叉树索引[19]，但其主要应用于传统的空间数据结构，在实时海量增长的交通流大数据环境下，难以维持原有的索引性能；在其基础上的改进索引有结合 R-tree、B*-tree 和 Hash 表改进生成的轨迹数据 HBSTR-tree 索引[20]、结合 R-tree 和格网索引的针对不均匀分布移动对象的区域索引[21]、利用 GeoHash 编码实现的邻近目标索引[22,23]、利用交通流数据和 GeoHash 编码映射实现的移动车辆索引[24,25]等，但其中有的基于传统空间索引方法进行分布式改进，或索引建立过程较为复杂，或时间和空间权重不均衡。因此，针对智能交通应用需求，仍需结合交通流数据模型对索引和查询方法进行优化。本文拟设计一种基于 HBase 分布式数据库的数据表行键索引，为高效的交通流大数据并行查询提供时空权重均衡的索引方法，为智能交通管理应用提供技术基础和数据支撑。

1　系统架构

交通流数据为多源传感器采集的异构数据，为了实现此类大数据的检测统计、远程管理、预测发布等智能应用，必须要建立一套多源数据预处理系统，实现实时海量增长的交通流数据的高效存取、清洗、索引和查询，为智能交通应用提供高效数据支撑。本文在 Spark 框架基础上，设计了一种交通流数据预处理系统框架（图 1）。

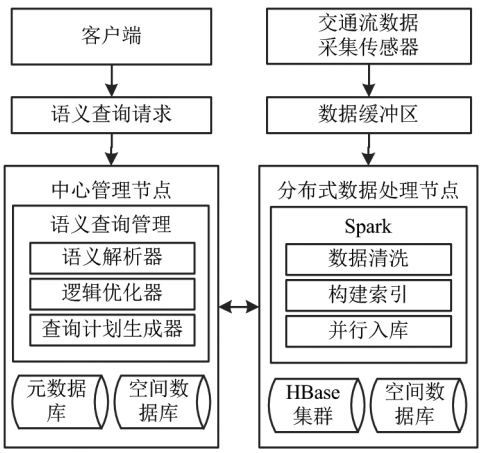

图 1　交通流数据预处理系统框架

Fig. 1　Framework of traffic flow data preprocessing system

该系统框架分为两个层次：1）分布式数据处理节点。该节点主要负责对多源传感器采集的异构交通流数据进行缓冲预处理，利用 Spark 框架将交通流数据转换为弹性分布式数据集（RDD），通过 Spark 并行计算对交通流数据进行清洗，构建 HBase 的混合时空编码行键用于数据索引；按照数据记录所处空间位置及区域数据量进行切片划分，通过制定的负载均衡分配规则将交通流数据分配至对应分布式节点；最后由 HBase 集群数据库对清洗后的海量交通流数据进行分布式存储，同时将数据的组织和分布信息更新并上传至中心管理节点的元数据库。2）中心管理节点。客户端通过人机交互界面对所需交通流数据进行语义查询，由中心管理节点的语义解析器将查询条件分解为基本查询单元，并根据基本查询单元的逻辑关系构成查询条件树；提取查询时间区间和空间范围，利用混合时空编码行键查询算法从 HBase 集群中提取命中的数据记录，形成备查中间数据集，由逻辑优化器生成逻辑运行表，并根据一定优化规则对查询逻辑进行优化；最后通过查询计划生成器对元数据库中分布式节点的数据信息进行遍历，将一系列的查询计划分解转化为对交通流备查数据集的实际查询操作。

该系统架构可解决两项关键问题：一是存储，数据在采集完成后，可以利用 Spark 的内存计算和迭代计算机制执行较为复杂的清洗筛选算法[26]，同时并行构建本文设计的数据索引，完成交通流大数据高效处理和入库；二是访问，提取分析数据时，需充分利用兼顾时空权重的数据索引，实现目标数据的快速定位和提取。

2　基于 HBase 的交通流数据模型及并行存储方法

本文设计了基于 HBase 混合时空编码行键的交通流数据模型，利用混合时空编码行键对每条数据记录进行索引，然后利用 Spark 框架的并行计算能力对交通流数据进行清洗、索引和入库，为后续的查询、分析、预测应用建立数据基础。

2.1　交通流数据模型

根据 HBase 表的逻辑结构，基于 HBase 混合时空编码行键的交通流数据模型由行键 RowKey、时间戳 TimeStamp 和列族 ColumnFamily 组成（图 2）：RowKey 为一种混合时空编码行键，用于对交通流数据进行唯一索引；TimeStamp 用于标记数据的记录时间以区分数据版本；ColumnFamily 为按需动态扩展的属性列族，包含了交通流数据的空间数据和属性数据。

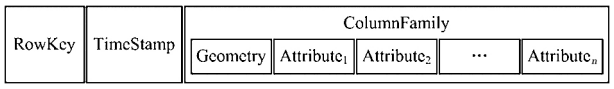

图 2　基于 HBase 的交通流数据模型
Fig. 2　Traffic flow data model based on HBase

2.2　混合时空编码行键

本文结合 HBase 进行数据查询时的行键匹配规则，针对交通流数据的时空特征，设计了 HBase 混合时空编码行键（Mixed Spatial-Time RowKey）（图 3）。其中的 GeoHash 编码[22]是一种将经纬度信息通过二分法分割转换的二进制编码，可将某点的经纬度坐标按照 GeoHash 编码规定的取值区间迭代二分并交叉组合，形成二进制编码串，然后执行 Base32 编码[27]，即可得该点坐标的 GeoHash 编码；其特点在于编码对应的空间位置唯一，两组编码相同，前缀位数越多表示空间位置越近，编码位数越多表示空间范围越精确（12 位编码表示约 3.46 cm^2 的范围），用其表示交通流空间信息可以满足实际应用需求。

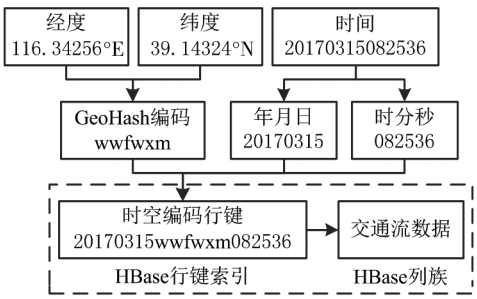

图 3　混合时空编码行键结构
Fig. 3　Mixed spatial-time RowKey structure

HBase 在进行行键匹配索引时遵循从左到右的次序，故按照智能交通应用中的查询规则，将行键中的时间信息分为“年月日”和“时分秒”两部分，并将表示空间信息的 GeoHash 编码插入其中。交通流应用中对数据的大部分查询操作是针对一天之内的数据，故将“年月日”作为行键编码的前缀，查询数据时会优先将符合查询日期的数据集扫描出来，然后按照 GeoHash 编码进行位置匹配，最后按照“时分秒”返回查询结果数据集；此查询过程效率较高，几乎可以不使用行键过滤器即可得到查询结果。

在进行空间位置匹配时，GeoHash 编码不但可以表示实际地理目标空间范围，而且其截取长度可以体现空间范围的大小（图 4）。使用 GeoHash 编码进行空间匹配时，所有与该编码前缀一致的数据记录都落在该矩形范围内，与常用的格网索引具有一定的相似性；但与格网索引相比，它还具有独特的优点：1）GeoHash 编码截取长度决定了查询范围的大小，在进行空间查询时不但可以灵活改变，而且根据 HBase 的行键前缀匹配规则，在检索指定范围的交通流数据时效率较高；2）GeoHash 编码和时间信息组合为混合时空编码行键，可更加高效地表达目标在时空维度上的特征。对于时间范围超过一天的查询操作，可将查询条件以天为单位分解为多个单日查询，以保证 GeoHash 编码的空间索引作用。

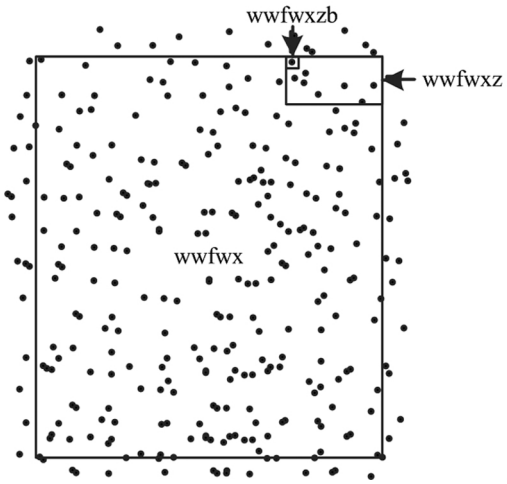

图 4　不同位数 GeoHash 编码表示范围示意
Fig. 4　The corresponding regions of the GeoHash code with different digits

2.3　动态扩展属性列族

对于多源交通流数据，不同的采集传感器获取的数据结构各不相同，如 GPS 传感器获取的数据结构包括定位时间、经纬度、高程、速度和车辆 ID 等，而摄像头传感器数据结构包括摄像头 ID、车辆颜色、车牌号、车速、检测时间、检测地点、检测部门和车辆照片等，因此，可以利用 HBase 的列族动态扩展能力管理多源异构的交通流数据。在使用 HBase 列族存储各类信息时（图 5），首先将其分为空间信息和属性信息两类。对于空间信息，可将定位坐标、检测位置等空间数据转换为 wkb 二进制流来表达几何体，并存储至列族 Geometry 中；使用 wkb 有利于实现异构平台之间的互操作，为系统的可扩展性提供便利。而对于其他非几何属性，首先提取其属性名作为列族的列限定符，利用 HBase 的列族动态扩展功能按需增加，然后提取非几何属性值作为对应列值；此方式充分利用了 HBase 的扩展性能，也有利于保持多源数据结构的完整性。

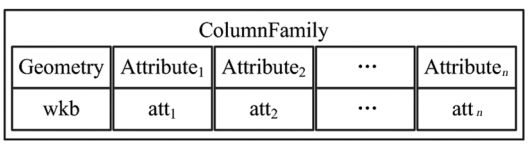

图 5　基于 HBase 的交通流列族结构
Fig. 5　The column family of traffic flow data based on HBase

2.4　预分区并行数据入库

以上的交通流数据模型能够满足数据索引和存储的需求，但在大数据环境下还需要考虑存储过程中的负载均衡问题。HBase 的 Region 分区的默认切分策略为：导入数据时，一直向某 Region 分区写入数据，直到该 Region 分区大小达到指定阈值时才进行分裂；这种默认策略会导致 HBase 集群负载失衡，从而浪费计算和存储资源。因此，本文首先对历史交通流数据进行统计，按照统计量将研究区域划分为大小不一的格网，交通流密集的热点区域格网较小，交通流较少的边缘区域格网较大，每个格网内的交通流数据在单位时间内的增量相对均衡，然后为每个格网预先创建一个 Region 分区，以实现负载均衡。由于本文设计的混合时空编码行键与空间位置密切相关，在进行数据入库时，很容易根据行键计算该数据记录所处的格网位置，并将其存储在对应的 Region 中。

在设置 Region 分裂策略时，首先根据需求设置 Region 大小，单个 Region 最大值为 10 GB，以避免系统初始运行时的频繁分裂；同时将分裂规则设置为 KeyPrefixRegionSplitPolicy，可以在分裂时指定 RowKey 前缀位数，将相同前缀位数的数据记录分配至同一 Region 中，结合本文设计的混合时空编码行键，相同日期（年月日）和空间范围（GeoHash 编码前缀）内的交通流数据会被分配至同一 Region。

利用以上基于交通流统计量的格网划分方法和基于行键前缀位数的 Region 分裂策略，可以在一定程度上实现交通流数据入库时的负载均衡。

交通流数据预分区并行入库流程（图 6）为：1）对历史交通流数据进行统计，按照统计量将研究区域划分为大小不一的格网，每个格网内的数据在单位时间内的增量相对均衡，然后为每个格网预先创建 Region 分区；2）从交通流数据缓冲区提取数据，并基于 Spark 框架将其转换为 Spark RDD；3）在 Map 运算中使用孤立点检测算法[26]，对 RDD 中的交通流数据记录进行并行数据清洗，实验证明该算法对冗余、丢失、错误类型“脏数据”的识别率在 90％以上，数据清洗效果较好；4）对于清洗后的交通流数据集，提取每条数据记录的空间和属性数据，根据空间位置信息计算 GeoHash 编码，并结合时间信息构建混合时空编码行键；5）提取交通流空间和属性数据的名称和值，作为 HBase 列族的列限定符和列值；6）将混合时空编码行键和交通流列族组合成交通流数据记录，并根据规则匹配入库至对应预分区 Region。

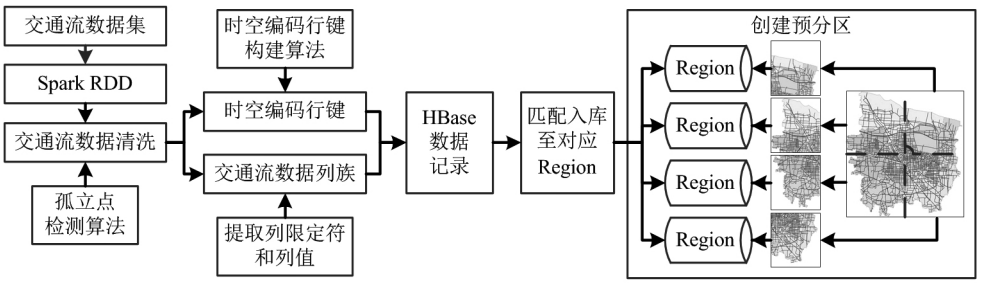

图 6　交通流数据预分区并行入库流程

Fig. 6　Traffic flow data importing process based on pre-partition and parallel technology

3　基于混合时空编码行键的交通流语义查询

3.1　语义查询框架

对于分布式的交通流大数据智能应用，在进行数据查询时，其查询语句可能较为模糊，查询条件包含时间、空间及其他属性，因此，需要通过语义解析对查询条件进行任务分解，提取出查询时间区间、空间范围和属性信息后，充分利用混合时空编码行键实现备查数据集的提取，并利用 Spark 完成并行属性过滤，最终将结果返回客户端。

交通流数据语义查询流程（图 7）为：1）由客户端根据应用需求输入语义查询条件；2）根据用户输入的模糊查询条件进行语义分析，由语法规则将查询条件分解为基本查询单元，并根据基本查询单元的逻辑关系构成查询条件树，同时提取查询时间区间和空间范围，以便从 HBase 集群中进行时空条件查询；3）利用混合时空编码行键查询算法从 HBase 集群中提取命中的数据记录，形成备查中间数据集，使用 Spark 框架将其转换为 Spark RDD，以备其他属性查询条件进行过滤；4）使用逻辑优化器根据一定优化规则对语义解析器生成的查询条件树进行优化，对逻辑上冗余或失效的查询语句进行删减合并，生成逻辑运行表，并进行验证和保存；5）读取查询条件树及优化后的逻辑运行表，将一系列的查询计划分解转化为对交通流备查数据集的实际查询操作，最终得到语义查询结果。

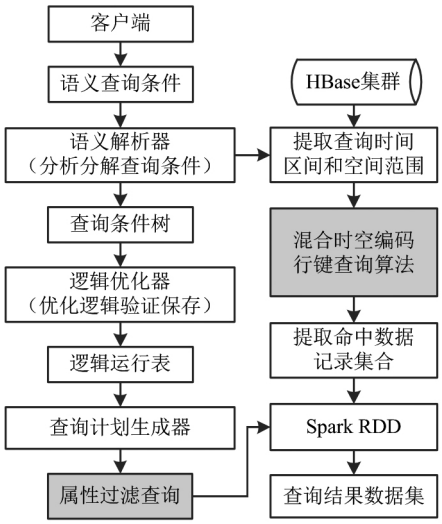

图 7　交通流数据语义查询框架

Fig. 7　Framework of traffic flow data semantic query

3.2　混合时空编码行键查询算法

基于混合时空编码行键的交通流数据查询算法输入项：时空查询时间区间[$Time1$, $Time2$]，查询区域空间范围及其外接矩形左上角点坐标 $P1(B1, L1)$，右下角点坐标 $P2(B2, L2)$；算法输出项：按照时空匹配查询到的交通流数据集。

算法步骤为：1）输入交通流时空查询的时间区间和空间范围条件；2）判断时间区间[$Time1$, $Time2$]是否在一天之内，若是，则执行步骤 3），否则将时间区间按照日期进行分割，生成查询时期列表，对其中每一天执行步骤 3）、步骤 4）；3）根据查询所需精度，确定 GeoHash 编码前缀位数，分别计算查询区域外接矩形角点 $P1$ 和 $P2$ 所在的 GeoHash 格网，按照外界矩形空间范围，生成所有位于该范围内的 GeoHash 格网编码列表；4）将查询所需时间（年月日）与 GeoHash 格网编码列表组合为时空查询 RowKey 前缀列表，扫描 HBase 集群得到的交通流数据集合作为备查数据集；5）利用 Spark 将备查数据集转换为 RDD，根据查询逻辑优化器和查询计划生成器构建的非时空属性过滤条件，对 RDD 进行并行属性条件过滤，同时根据查询区域形状进行二次空间位置过滤，最终得到满足语义查询条件的数据集合。

3.3　并行属性条件过滤

基于 Spark 的并行属性条件过滤流程为：对于经过行键时空匹配获得的备查交通流 RDD，调用 map 算子将过滤任务分配给多个分布式节点并行执行，分布式节点通过判断非时空属性是否匹配进行过滤，同时按照空间包含关系判断数据记录所处位置是否在查询区域内部，过滤后的 Spark RDD 中包含的交通流数据记录即为最终查询结果。

4　实验与分析

4.1　实验数据与运行环境

本文使用某交通管理平台采集的多源交通流数据进行实验，数据采集传感器包括视频、地磁、微波和 GPS，交通流数据记录日均增量约2 000万条，实验提取了 2017 年 3 月 13－27 日共计 14 天的数据，数据采集范围为郑州市主城区。

实验计算机集群由 5 台服务器组成，一台为中心管理节点，负责分布式节点元数据管理和交通流数据语义查询，另 4 台为分布式数据处理节点，负责对采集的交通流数据进行清洗、索引和并行入库。服务器硬件配置：CPU 为 Intel XeonE5－2640 2.6 GHz，8 核，内存 16 GB；主要软件及版本见表 1。

表 1　主要软件及版本

Table 1　The version of main softwares

软件	版本
Linux	CentOS-7-x86_64-LiveKDE-1511
Java	jdk-7u79-linux-x64
Hadoop	hadoop-2.6.1
Spark	spark-1.6.0-bin-hadoop2.6
HBase	hbase-1.2.0
MPI	MPICH 2-1.0.7
Oracle	Oracle 11g

4.2　并行计算框架对比实验

交通流数据具有实时、海量、多源等特点，必须利用高效的计算框架对其进行清洗、统计和存储，本文选取了 3 种方法对交通流数据进行并行清洗和入库操作。方法 1：传统的并行计算框架 MPI＋Oracle 数据库。MPI[4]是一种基于主从进程消息传递的并行程序函数库，此方法由中心节点的主进程分配计算任务，由分布式节点的从进程执行清洗算法，并为数据创建基于历史统计量的不规则格网索引，最后将处理后的交通流数据导入 Oracle。方法 2：大数据计算框架 MapReduce＋HBase。此方法在分布式节点中配置了 48 个 Map 运算，用于执行清洗算法，并为数据构建 HBase 混合时空编码行键索引，利用基于统计量的 Region 预分区策略实现数据存储负载均衡，同时在中心节点的 Reduce 运算中完成交通流元数据更新。方法 3：Spark＋HBase。此方法与方法 2 类似，主要区别在于此方法将交通流数据集转换为 Spark RDD，运用内存计算模式执行清洗算法，并构建行键索引。

为了对比以上 3 种方法的执行效率，数据清洗算法均选择孤立点检测算法，实验数据使用了 50 万、200 万、500 万、1 000万、1 500万和2 000万条共 6

组数据，以体现不同数据量条件下 3 种方法运行效率的变化趋势（图 8）。

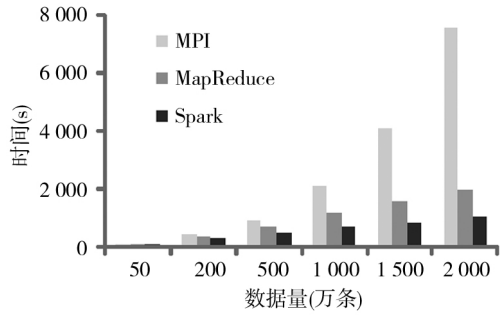

图 8　并行计算框架效率对比
Fig. 8　Efficiency comparison of parallel computing frameworks

通过图 8 可知，在数据量较小时，三者的数据处理耗时接近，但随着数据量的增长，数据处理时间差别急剧增加。原因在于：MPI 方法中，虽然其框架结构简单，自身所耗费的计算资源较少，但随着数据量持续增加，数据清洗算法中的迭代过程耗时更长，数据处理总体效率大大下降；MapReduce 方法中，其框架封装了并行处理、负载均衡、容错等技术细节，开发难度较低，比 MPI 方法耗时少，但该方法的磁盘读写和资源分配机制效率较低，交通流数据处理总体效率仍有提升空间；Spark 方法的优势在于内存计算和迭代计算，可以按时间区间将数据转换为 RDD，由清洗算法进行处理，而存储数据时，则需要计算每一条交通流数据记录的混合时空编码行键，以实现高效的时空语义查询，会增加少量运算时间，但总体上，数据量越大，Spark 方法的优势越明显。综上，本文设计的基于 Spark/HBase 的交通流数据存储模型更适合于清洗处理和相关数据挖掘算法。

4.3　区域查询对比实验

在数据处理时构建的数据索引会在查询和提取交通流数据时起到关键作用。本文设计的混合时空编码行键索引中的 GeoHash 编码具有与格网索引类似的作用，因此，本研究设计了区域查询对比实验，以分析二者的工作效率。

由于混合时空编码行键前缀为“年月日”，为了在对比索引效率时排除时间因素的影响，实验中使用了一天内数据量不同的 5 组数据集（数据记录分别为 200 万、500 万、1 000万、1 500万、2 000万条）进行查询；同时，两种索引均在 HBase 数据库中构建，以排除不同数据库的性能差异。1）格网索引。按照研究区范围构建基于历史统计量的不规则格网索引，交通流密集的热点区域格网较小，交通流较少的边缘区域格网较大，每个格网内的交通流数据在单位时间内的增量相对均衡。对交通流数据清洗处理后，根据每条数据记录的空间位置，计算其所在的格网单元，并将格网单元标识与时间数据组合，作为 HBase 中交通流数据的行键索引。在进行区域查询时，首先计算与查询范围相交的格网单元集合，将其所包含的数据集作为备查数据集，然后对备查数据集中的数据进行空间坐标匹配过滤，提取查询结果数据集。2）GeoHash 行键索引。根据前文设计的基于混合时空编码行键的查询算法，首先根据查询所需精度，确定 GeoHash 编码前缀位数，生成覆盖查询区域的 GeoHash 格网编码列表，然后将其中包含的数据集作为备查数据集，利用 Spark 将备查数据转为 RDD 后进行并行位置过滤，得到最终查询结果。

从实验结果（图 9）可知，随着数据量的增加，两种方法的查询耗时随之增加，而格网索引的查询耗时以及随数据量增加而产生的耗时增量均多于 GeoHash 行键索引。分析其原因在于：虽然本文使用基于历史统计量的不规则格网索引为创建 HBase 预分区的依据，可在一定程度上实现格网内交通流数据量的相对均衡，但热点区域和边缘区域格网大小不一，进行查询时，与查询范围相交的格网单元集合面积与真实查询范围面积之差也较大，即真实查询范围外的冗余无效范围较大，从格网单元集合中提取的冗余备查数据记录也相对较多，查询耗时增加；而 GeoHash 行键索引在构建完毕之后，可根据查询需要灵活指定索引前缀位数，从而得到冗余较小的覆盖查询区域的 GeoHash 格网，同时由于利用了 HBase 的行键索引特性和 Spark 的并行内存运算，使其区域查询效率相对较高。

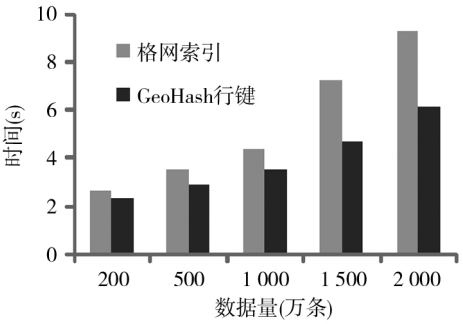

图 9　区域查询算法效率对比
Fig. 9　Efficiency comparison of regional query algorithms

4.4　时空索引对比实验

在智能交通应用中，高效查询不但要体现在空间维度上，还要体现在时间维度上。HBase 混合时空编码行键的前缀顺序就体现了索引在时间和空间上的权重，并且对索引效率起着决定性作用。因此，本文设计了 3 种行键结构，并对其时空查询性能进

行对比分析。1）Geo-Time 行键，结构为“GeoHash 编码＋年月日时分秒”。行键中 GeoHash 编码在前，根据 HBase 行键索引的左侧匹配原则，在查询数据时，先搜索符合空间位置要求的数据记录，然后按照时间进行过滤。2）Time-Geo 行键，结构为“年月日时分秒＋GeoHash 编码”。行键中时间信息编码在前，在查询数据时，先根据时间进行匹配，然后按照空间进行过滤。3）Mixed Spatial-Time 行键，结构为“年月日＋GeoHash 编码＋时分秒”。大部分查询操作是针对一天之内的数据，通过年月日可快速搜索符合日期的数据集，然后按照 GeoHash 编码进行空间位置匹配，即可找到符合准确时分秒查询条件的数据。

（1）按照时间点查询不同范围的交通流数据。如在 2017 年 3 月 15 日 10：30－10：31，查询 2 km^2、5 km^2、10 km^2、15 km^2、20 km^2 范围内的所有交通流数据，结果如图 10 所示，3 种行键性能排序为 Time-Geo＞Mixed Spatial-Time＞Geo-Time。HBase 的行键索引特性决定了位于行键左侧的编码起到主要索引作用，因此，在时间查询条件确定的情况下，Time-Geo 行键会优先完成时间条件匹配，极大地减少了数据扫描范围，然后通过行键过滤器查询 GeoHash 空间范围内的数据记录，其效率最高。而 Geo-Time 行键结构则相反，其查询时间会随着查询范围的增大而增长，效率最低。

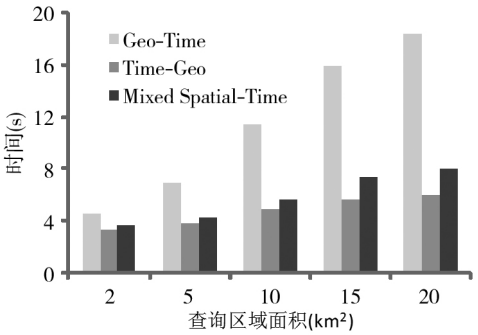

图 10　不同空间范围时空索引效率对比
Fig. 10　Efficiency comparison of spatio-temporal indexes in different space range

（2）按照固定范围查询时间区间内的交通流数据。如在郑州市 CBD 范围内，查询 2 h、4 h、6 h、8 h、10 h 时段内的所有交通流数据，结果如图 11 所示，3 种行键性能排序为 Geo-Time＞Mixed Spatial-Time＞Time-Geo。在空间范围查询条件确定的情况下，Geo-Time 行键的优势得以体现，通过空间范围条件优先匹配可以快速减少扫描数据范围，此时其效率最高。而 Time-Geo 行键的查询时间会随时间区间增大而增长，其效率最低。

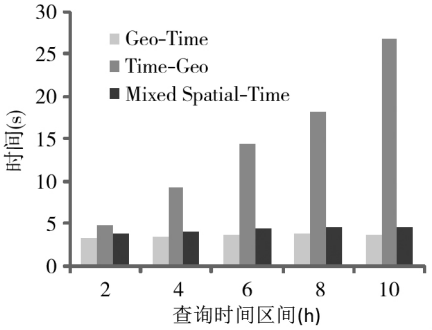

图 11　不同时间区间时空索引效率对比
Fig. 11　Efficiency comparison of spatio-temporal indexes in different time interval

由以上实验结果可知，Geo-Time 行键和 Time-Geo 行键都只能在一种信息维度上具备较好的查询性能，在另一维度上会随着空间范围或时间区间的增加而导致效率急剧下降。本文设计的 Mixed Spatial-Time 行键的时间编码和 GeoHash 编码在时空维度上都可以起到索引效果，无论在哪种查询条件下，都具备较好的查询效率，且结果更为稳定；尤其是在时空区域查询上的性能与数据量级关系不大，此特点更适用于涉及海量数据的智能交通应用。

5　结语

为了满足智能交通信息服务的需求，必须实现对实时海量增长的交通流大数据的准确高效处理与管理。本文设计了一套基于 Spark 和 HBase 的分布式交通流数据处理框架。基于 HBase 的交通流数据模型利用混合时空编码行键实现了数据时空信息的有效索引，利用 HBase 列族的动态扩展能力保持了多源异构交通流数据信息的完整性，并利用 Spark RDD 和 HBase Region 预分区完成了高效并行数据入库；基于混合时空编码行键的交通流语义查询方法，通过语义解析将时间、空间和属性查询条件分解重构，并充分利用混合时空编码行键的索引功能，实现备查交通流数据集的快速提取，减少进行非时空属性查询时的数据量，最终利用 Spark RDD 进行查询条件过滤，实现高效语义查询。

并行计算框架对比实验证明，在 Spark 框架下利用 HBase 数据结构的特点实现并行交通流数据清洗、索引和入库，充分利用了框架高效的内存计算和迭代计算能力，比传统方法性能提升显著，更能满足交通流大数据的应用需求。区域查询和时空索引对比实验证明，本文设计的基于 HBase 的混合时空编码行键，可以有效为多源交通流数据建立时空索引，而且在时间维度和空间维度的权重较为均衡，在各种查询条件下都具备较好的查询效率。

参考文献：

[1] 李德仁，马军，邵振峰. 论时空大数据及其应用[J]. 卫星应用，2015(9)：7-11.

[2] 罗乐，刘轶，钱德沛. 内存计算技术研究综述[J]. 软件学报，2016，27(8)：2147－2167.

[3] 李德仁. 展望大数据时代的地球空间信息学[J]. 测绘学报，2016，45(4)：379－384.

[4] 邓清清. 交通诱导系统的流量预测和路径优化并行算法研究[D]. 大连：大连理工大学，2008.

[5] 郑伟. 微观交通仿真中的路径规划和车辆更新的并行计算[D]. 西安：长安大学，2014.

[6] 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.

[7] 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.

[8] WOLFSON O，SISTLA A P，CHAMBERLAIN S，et al. Updating and querying databases that track mobile units[J]. Distributed and Parallel Databases，1999，7(3)：257－387.

[9] ERWIG M，GÜTING R H，SCHNEIDER M，et el. Spatio-temporal data types：An approach to modeling and querying moving objects in databases[J]. GeoInformatica，1999，3(3)：265－291.

[10] 王宏勇，郭建星. 空间运动对象与时空数据类型研究[J]. 地理与地理信息科学，2005，21(5)：1－5.

[11] PEUQUET D J，NIUDUAN. An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data[J]. International Journal of Geographical Information Systems，1995，9(1)：7－24.

[12] 赵卓峰，丁维龙，韩燕波. 基于云架构的交通感知数据集成处理平台[J]. 计算机研究与发展，2016，53(6)：1332－1341.

[13] 赵卓峰，丁维龙，张帅. 海量车牌识别数据集上基于时空划分的旅行时间计算方法[J]. 电子学报，2016，44(5)：1227－1233.

[14] 孙艳华，王丽娜. 一种面向流数据的分布式实时存储方法[J]. 电脑知识与技术，2015，11(19)：5－6.

[15] 钟运琴，朱效民，程振林，等. 一种基于分布式存储计算架构的高效能海量空间数据管理方法[A]. 全国高性能计算学术年会[C]. 2011.

[16] 肖伟器，冯玉才，缪勇武. 空间对象数据库的网格索引机制[J]. 计算机学报，1994，17(10)：736－742.

[17] 陆锋，周成虎. 一种基于 Hilbert 排列码的 GIS 空间索引方法[J]. 计算机辅助设计与图形学学报，2001，13(5)：424－429.

[18] GUTTMAN A. R-trees：A dynamic index structure for spatial searching[A]. Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data[C]. 1984. 18－21.

[19] FINKEL R A，BENTLEY J L. Quad trees a data structure for retrieval on composite keys[J]. Acta Informatica，1974，4(1)：1－9.

[20] GONG J，KE S N，ZHU Q，et al. An efficient trajectory data index integrating R-tree，Hash and B*-tree[J]. Acta Geodaetica et Cartographica Sinica，2015，44(5)：570－577.

[21] 郭超，李坤，王永炎，等. 面向实时定位系统的位置区域索引[J]. 计算机研究与发展，2011，48(10)：1908－1917.

[22] 金安，程承旗，宋树华，等. 基于 Geohash 的面数据区域查询[J]. 地理与地理信息科学，2013，29(5)：31－35.

[23] 方金云，刘羽，姚晓，等. 基于 Spark 的空间数据实时访存技术的研究[J]. 地理信息世界，2015，22(6)：24－31.

[24] SHEN D D，FANG J，HAN Y B. A nearby vehicle search algorithm based on HBase spatial index[A]. Web Information System & Application Conference[C]. 2015. 71－74.

[25] 李冬，房俊. 基于 HBase 的交通数据区域查询方法[J]. 计算机与数字工程，2017，45(2)：230－234.

[26] 王晓原，张敬磊，吴芳. 交通流数据清洗规则研究[J]. 计算机工程，2011，37(20)：191－193.

[27] JOSEFSSON S. The Base16，Base32，and Base64 Data Encodings[M]. RFC Editor，2003.

Discussion on the Data Storage and Index Model of Traffic Flow Based on Spark/HBase

LI Xin

(*Collaborative Innovation Center of Three-Aspect Coordination of Central Plain Economic Region，Henan University of Economics and Law，Zhengzhou 450046；College of Resource and Environment，Henan University of Economics and Law，Zhengzhou 450046，China*)

Abstract：In order to process and manage traffic flow data with massive growth，it is necessary to design an efficient data storage and index model based on big data framework to meet the needs of intelligent transportation applications. This paper designs a system framework based on Spark/HBase，and a traffic flow data storage and index model based on mixed spatial-time RowKey and dynamically extended attribute column family. On the basis of this model，efficient semantic query of traffic flow big data is realized through semantic analysis，spatial-time RowKey index query and parallel attribute condition filtering. The comparison experiment proves that the traffic flow big data parallel processing framework designed in this paper is efficient in cleaning，indexing and storing data. The spatial-time weight of mixed spatial-time RowKey index is balanced. This method can achieve more efficient traffic flow data access and storage management，and can establish the technical foundation for the intelligent transportation application.

Key words：Spark；HBase；traffic flow；spatial-time RowKey；semantic query