文本复制检测报告单(去除本人已发表文献)

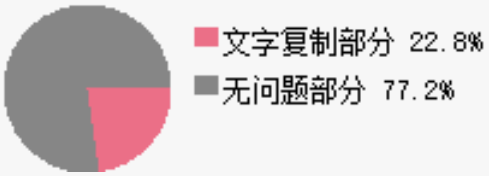
№:ADBD2019R_20190305131041201904121151347051967015 38　　检测时间:2019-04-12 11:51:34

检测文献:　谢冲
作者:　谢冲
检测范围:　中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
学术论文联合比对库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
CNKI大成编客-原创作品库
个人比对库
时间范围:　1900-01-01至2019-04-12

检测结果

去除本人已发表文献复制比：22.8%

重复字数：	[9230]	总段落数：	[7]
总字数：	[40483]	疑似段落数：	[6]
疑似段落最大重合字数：	[3851]	前部重合字数：	[2763]
疑似段落最小重合字数：	[111]	后部重合字数：	[6467]



指　标：☑疑似剽窃观点　☑疑似剽窃文字表述　☐疑似自我剽窃　☐疑似整体剽窃　☐过度引用

表　格：0　　公　式：0　　疑似文字的图片：0　　脚注与尾注：0

3.6%（111）　中英文摘要等（总3076字）
52.8%（3851）　第1章绪论（总7294字）
42.7%（3133）　第2章技术理论基础（总7343字）
13.3%（1165）　第3章时空矢量对象的索引设计及存储（总8789字）
5.1%（452）　第4章时空矢量对象的查询算法设计（总8801字）
0%（0）　第5章实验性能评价及结果分析（总2787字）
21.6%（518）　第6章总结与展望（总2393字）

（注释：无问题部分　文字复制部分）

疑似剽窃观点　(1)

第6章总结与展望

1. 目前，我认为还可以在以下几个方面继续开展相关研究，以推动大数据时代矢量数据管理基础理论和关键技术的快速发展。

1. 中英文摘要等　　总字数：3076

相似文献列表

去除本人已发表文献复制比：3.6%(111)　　文字复制比：3.6%(111)　　疑似剽窃观点：(0)

1	姚晓闯_B1311686_矢量大数据管理关键技术研究	3.6%（110）
	姚晓闯 -《学术论文联合比对库》- 2017-06-01	是否引证：否
2	矢量大数据管理关键技术研究	1.6%（50）
	姚晓闯(导师：郧文聚;朱德海) -《中国农业大学博士论文》- 2017-05-01	是否引证：否

原文内容

摘要

随着全球卫星导航定位系统、传感网、移动互联网、物联网等技术的快速发展，海量时空位置信息如车辆轨迹、个人轨迹、社交媒体数据，checkin数据等数据爆发式增长。这些数据具有流式动态、时空多维、规模巨大、蕴含价值、稀疏等特征。在这样的时空大数据背景下，基于现有的GIS商业化空间数据管理平台对矢量大数据的管理能力也几近极限，无法满足全国行业应用的需求，基于关系型数据库的存储管理模式也面临着严峻挑战：1）对于异源异构的非结构化时空数据，其难以提供自适应的存储组织方案；2）面对流式生成的时空数据，其难以提供强大的高并发读写能力支持；3）对于持续快速增长的数据量，其难以提供高可扩展性支持。针对上述问题，本文在分布式数据库HBase的基础上，利用NoSQL数据库的优势，提出一套完整的存储组织、时空索引、查询检索的方案。该系统在时空立方体的基础上，设计实现了时空串接编码索引。且根据分布式数据库HBase的特性，对于矢量对象数据集，分别构建了元数据，编码表，数据表。其中元数据主要记录数据集的元信息，如包含数据的编码方案，矢量类型，空间参考等信息；编码表用于时空范围查询，k-NN查询；数据表是为了时空矢量对象的属性查询。在本文设计的时空的范围查询中，本文能够根据时空编码，同时满足点对象和线面对象的范围查询。在k-NN查询中，本文提出一种顾及数据分布的k-NN查询算法，即能够根据数据的分布控制网格的大小再进行扩充搜索。经实验验证，该系统能够在大数据场景下存储海量的时空矢量数据对象，且同时支持低延迟的时空范围查询、k-NN查询。

关键词：大数据；分布式数据库；时空索引；k-NN查询

Abstract

As data collection methods mature and diversify, data sources such as personal smart devices, floating car GPS, internet of things, and social media are becoming increasingly abundant, and the amount of data have been accumulating in an explosive manner. These data have streaming dynamics, multi-dimensional, large scale, sparse and other characteristics. In the background of such spatio-temporal big data, based on the existing GIS commercial spatial data management platform, the ability to manage vector big data is also near the limit, unable to meet the needs of national industry applications, and the storage management model based on relational database is also faced severe challenges: 1) It is difficult to provide adaptive storage organization scheme for heterogeneous spatiotemporal data; 2) It is difficult to provide strong high concurrent literacy support in the face of streaming generated spatiotemporal data. 3) It is difficult to provide high scalability support for the continuous and rapidly growing amount of data. In view of the above problems, based on the distributed database HBase, this paper proposes a complete storage organization, spatiotemporal index, query and retrieval scheme based on the advantages of NoSQL database. Based on the spatio-temporal cube, the system is designed to realize spatio-temporal indexing of space-time serial coding. According to the characteristics of the distributed database HBase, metadata, coding tables, and data tables are respectively constructed for vector dataset. The metadata mainly records the meta information of the data set, such as the coding scheme, the vector type, the spatial reference and so on; the coding table is used for the spatio-temporal range query, the k-NN query; and the data table is the attribute query for the spatio-temporal vector object. In the k-NN query, this paper proposes a k-NN query algorithm that takes into account the data distribution. The algorithms can control the size of the grid according to the distribution of the data and then expand the search. The experimental results show that the system can store a large number of spatio-temporal vector data objects in a big data scenario, and at the same time support low-latency spatio-temporal range query and k-NN query.

Keywords：Big data; Distributed database; Spatial-temporal index; k-NN Query

目录

指　标

疑似剽窃文字表述

1. 基于现有的GIS商业化空间数据管理平台对矢量大数据的管理能力也几近极限，无法满足全国行业应用的需求，
2. the existing GIS commercial spatial data management platform,

2. 第1章绪论

总字数：7294

相似文献列表

去除本人已发表文献复制比：52.8%(3851)　　文字复制比：52.8%(3851)　　疑似剽窃观点：(0)

1	姚晓闯_B1311686_矢量大数据管理关键技术研究 姚晓闯 -《学术论文联合比对库》- 2017-06-01	21.7%（1582） 是否引证：否
2	矢量大数据管理关键技术研究 姚晓闯(导师：郧文聚;朱德海) -《中国农业大学博士论文》- 2017-05-01	20.6%（1501） 是否引证：否
3	基于Hadoop平台的时空数据索引和查询技术研究 何立志(导师：李龙海) -《西安电子科技大学硕士论文》- 2018-06-01	17.5%（1280） 是否引证：否
4	基于空间填充曲线高维空间查询算法研究 徐红波(导师：郝忠孝) -《哈尔滨理工大学博士论文》- 2010-03-01	7.3%（533） 是否引证：否
5	Z曲线网格划分的最近邻查询 刘润涛;陈琳琳;田广悦; -《计算机工程与应用》- 2012-12-03 1	2.9%（208） 是否引证：否
6	王广钰 王广钰 -《学术论文联合比对库》- 2017-04-10	1.9%（138） 是否引证：否
7	基于Hadoop的时空大数据的分布式检索方法 王广钰(导师：李英玉) -《中国科学院大学(中国科学院国家空间科学中心)硕士论文》- 2017-05-01	1.9%（138） 是否引证：否
8	数据网格QoS保障与资源优化关键技术研究 曲明成(导师：杨孝宗) -《哈尔滨工业大学博士论文》- 2011-04-01	1.2%（89） 是否引证：否
9	大数据时代的交管综合应用云平台 徐文斌; -《第八届中国智能交通年会论文集》- 2013-09-26	1.2%（84） 是否引证：否

10	基于iOS的汽车租赁客户端的设计与实现	0.9%（64）
	李文慧(导师：朴勇) -《大连理工大学硕士论文》- 2018-03-01	是否引证：否
11	基于知识图谱的Web of Scholars系统设计与实现	0.8%（61）
	康文杰(导师：夏锋) -《大连理工大学硕士论文》- 2018-04-01	是否引证：否
12	HBase中半结构化时空数据存储与查询处理	0.8%（59）
	封孝生;张翀;陈晓莹;唐九阳;葛斌; -《国防科技大学学报》- 2016-06-28	是否引证：否
13	改进K-means算法在文本聚类中的应用	0.7%（53）
	周本金(导师：陶以政) -《中国工程物理研究院硕士论文》- 2018-05-01	是否引证：否
14	基于Hadoop的IPTV故障预判算法的研究及系统实现	0.7%（50）
	王堂辉(导师：周亮) -《南京邮电大学硕士论文》- 2018-11-14	是否引证：否
15	基于HBase的空间数据分布式存储和并行查询算法研究	0.6%（47）
	丁琛(导师：鲍培明) -《南京师范大学硕士论文》- 2014-04-30	是否引证：否
16	基于云计算的BIM关键技术应用研究	0.6%（46）
	毕振波(导师：王慧琴) -《西安建筑科技大学博士论文》- 2015-06-01	是否引证：否
17	探索计算机技术发展新方向中的云计算	0.6%（41）
	李大伟; -《科技资讯》- 2018-11-03	是否引证：否

原文内容

第1章绪论

1.1 研究背景与意义

随着全球卫星导航定位系统、传感网、移动互联网、物联网等技术的快速发展，海量时空位置信息如车辆轨迹、个人轨迹、社交媒体数据，checkin数据等数据爆发式增长[1-2]。这些数据具有流式动态、时空多维、规模巨大、蕴含价值、稀疏等特征。在这样的时空大数据背景下，基于现有的GIS商业化空间数据管理平台对矢量大数据的管理能力也几近极限，无法满足全国行业应用的需求[3]，基于关系型数据库的存储管理模式也面临着严峻挑战：1）对于异源异构的非结构化时空数据，其难以提供自适应的存储组织方案；2）面对流式生成的时空数据，其难以提供强大的高并发读写能力支持；3）对于持续快速增长的数据量，其难以提供高可扩展性支持。随着云计算、NoSQL[4-5]数据库等新一代IT技术的发展与成熟，相关理论与方法已经开始逐步渗透到GIS领域，其分布式存储模式、高并发读写能力、以及高可扩展性/高可用性特征促使数据库管理模式逐渐向分布式数据库转变[6]。目前很多研究学者基于NoSQL数据库开展了广泛的时空数据存储管理研究。而以HBase[7]为代表的NoSQL数据库仅支持Key-Value模式的简单查询，对多维复杂查询能力支持不足。目前许多工作都围绕如何提高多维或空间查询性能展开研究。其中部分研究仅针对矢量数据，但不支持其时间维度的查询，无法进行时空查询；部分研究仅支持点数据对象的存储及查询，不支持更为复杂的线、面矢量数据；部分研究方案虽能够同时针对点，线面矢量数据进行时空查询，但仍受限于HBase平台的架构，无法取得性能上的突破，范围查询效率，k-NN查询效率仍有进一步提高和优化的空间。

针对上述问题，本文拟在分布式数据库HBase的基础上，利用NoSQL数据库的优势，提出一套完整的存储组织、时空索引、查询检索的方案，使该系统能够在大数据场景下存储海量的时空矢量数据对象，且同时支持低延迟的时空查询、k-NN查询。

1.2 国内外研究现状

1.1.1 时空数据库的研究现状

空间数据库是管理时空数据的有效手段，也是数据查询、分析应用的基础。目前，海量时空大数据的存储和管理主要采用三种方式：一是基于分布式扩展的关系型数据库进行存储；二是基于分布式文件系统进行存储；三是基于非关系型NoSQL数据库进行存储。其中第一种方式，在海量数据的情况下，可以通过提高水平扩展性存储更多的数据，但由于关系型数据库其本身的ACID局限性，不适用于高并发读写的场景，且无法适应非结构化的存储。在这里主要介绍第二、三类方式。

1.2.1.1分布式文件系统

分布式文件系统（Distributed File System）是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连；该系统对普通计算机具有较好的支持，在集群方面也具有良好的可扩展性和容错性；同时对大规模用户并发情况下的数据密集型服务也能提供良好的支持。其中，Apache软件基金会的Hadoop Distributed File System（HDFS）[8]文件系统是具有这种特点的分布式文件系统的典型代表。在基于HDFS分布式文件系统的矢量数据存储方面，国内外做了大量的研究。Tan.H基于Hadoop[9]对空间数据分析进行了研究[10]，提出了一个适合HDFS存储的GPS数据模型，实现了R-tree索引的并行构建，并基于此研发了多种时空数据查询功能。Ablimit Aji研制了Hadoop-GIS[11]，基于HDFS和MapReduce建立了一套高性能空间数据仓库系统，通过空间分割实现多种空间数据查询，其实验表明该系统明显优于并行空间数据关系系统。Ahmed Eldawy提出了一套基于开源Hadoop框架——SpatialHadoop[12-15]，通过HDFS存储空间数据集，除此之外该系统涵盖了基于Pig的空间高级编程语言、空间数据索引、空间查询以及可视化等一整套空间大数据解决方案，解决了矢量大数据空间分析的基本难题，在业界非常具有代表性和借鉴价值。ST-Hadoop[16]是基于Spatial Hadoop的支持时空数据的 Map Reduce框架，其内部建有二级时空索引，第一级为时间索引而第二级为空间索引，空间索引直接继承自Spatial Hadoop的Quad-Tree[17]和 kd-Tree[18]等。其中，ST-Hadoop 将数据按照不同尺寸的时间分片（Time-Slice）进行多次划分，并冗余存储到HDFS，运用了一种以存储资源换查询效率的思想策略。Hadoop云计算平台提供的分布式文件系统HDFS不仅支持基于磁盘存取的MapReduce并行处理计算，同时对基于内存的Spark并行计算[19]也有无缝的支持。因此，基于HDFS和Spark组合的矢量数据存储于管理平台也成为了可能。例如，Jia Yu研发的GeoSpark[20]，采用三层架构，即Apache Spark层、空间弹性分布数据层和空间查询操作层，其中数据存储采用HDFS作为存储系统。

1.2.1.2 非关系型NoSQL数据库

NoSQL，即Not Only SQL，泛指非关系型数据库。目前，主流的NoSQL数据库包括基于文档的存储方式的

MongoDB[21]；基于列存储的BigTable[22]、HBase；基于键值对进行存储的Redis[23]等。由于非关系型NoSQL数据库具有分布式、可扩展性、不需要预先定义表结构等诸多优点，近年来深受研究学者和商业公司的青睐，在矢量大数据存储和管理领域也有很多研究工作。吴飞[24]基于MongoDB的LBS数据管理系统，充分考虑系统集群和数据分片与系统负载均衡之间的关系，有效解决了矢量数据存储查询的技术瓶颈；雷德龙[25]等提出了基于MongDB存储和处理矢量空间数据的方法，采用Geojson格式以文件形式来存储矢量数据，设计并实现了基于数据存储层、核心业务层和表现层组成的矢量数据云存储与处理系统，有效提高了数据的并行访问和处理性能。张景云[26]提出了一种以内存数据库Redis的轻量级矢量地理组织方法，能在高并发情况下有效提高矢量地理数据服务性能。范永健[27]等基于HBase设计了矢量数据的表结构，其中Key为空间对象ID，S个列族分别用于存储空间对象的属性信息、坐标信息和拓扑信息。Nishimura面向LBS服务，设计了md-hbase[28]来存储管理空间数据，其通过空间划分构建了基于四叉树和kd-tree的空间索引，考虑到了数据的空间分布，并在此基础上实现了空间上的范围查询和最邻近查询。locationTech推出的Geomesa[29]系统是一种能够快速存储、索引以及提取时空大数据的分布式方案，其将空间时间通过三维的Z曲线交叉编码作为时空索引，并能够同时应用于HBase等分布式数据库。Geowave[30]同样也是一个能存储，获取以及分析海量多维时空数据的分布式键值存储方案，其使用Compact Hilbert[31]曲线将n维数据用一维表示，并用作时空索引。G-HBase[32]行键设置为Geohash[33]，并在各个Region上建立建立R树，用于提高查询效率和k-NN查询效率，且支持点、线、面对象。但其不支持时间属性，无法完成时空查询。Zhangy C[34]对于时空点数据建立了两层索引，第一层以Z曲线作为行键对空间进行过滤。在空间上相近的数据Z order编码相似，所以再以行键排序的HBase的Region Server上存储时也更加靠近。且在各个Region Server上，其对时空三维进行归一化后，整体建立了一颗基于内存的八叉树，用于进一步过滤从而提高查询效率。Ma Y[35]将时间进行粗粒度划分并建立B+[36]树，且在各个时间段下单独构建R树进行空间划分。基于非关系型NoSQL数据库存储空间大数据不仅方便部署，同时也能够很好地对多源空间数据进行高效集成，有利于数据密集型的并行计算。李雪梅[37]设计了一种基于分布式数据库HBase的GIS数据管理系统。针对矢量空间数据的存储、索引与检索，提出了一种新的Rowkey 设计，既考虑经纬度又考虑空间数据类型和属性，使得在按空间位置检索矢量地理信息时，能通过HBase的Rowkey迅速定位需要返回的数据。丁琛[38]提出了面向HBase的k-NN查询算法，基于查询热点并行化构建了索引表；使用索引表快速查找到k近邻对象,降低了查询时间,提高了查询效率。

1.1.2 时空索引技术

时空数据是一种描述空间状态随时间变化的数据类型，有助于帮助人们了解过去、把握现在、预测未来，当前采集到的科学数据基本是带有时间属性的，时间维度的增加极大地丰富了空间对象的数据信息。时空数据查询是时空数据分析处理的重要手段，而时空数据索引技术是时空数据查询的基础。

1.2.2.1 基于R树索引

目前，大多数主流时空索引是通过基于空间R树索引的变形扩展获得的。 R树是Guttman在1984年提出的空间数据的索引结构[39]，现在广泛用于各种商业应用和原型系统。R树是一种多级平衡树，它是B树在多维空间上的扩展，其中的每个节点代表一个区域，是包含所有子节点区域的的最小包围矩形（MBR），当中的各个叶子节点则指向各个空间对象。其查询过程是从根节点遍历R树，将查询区域与节点对应的的MBR进行检验比较。如果查询区域与遍历节点MBR有包含和交叉关系，则继续遍历该节点的子节点，当所有与查询区域相交的节点都遍历完成后，查询过程结束。如图1-1所示，QR所代表的圆为查询区域，按照上述查询过程，将根节点Root与QR进行比较，因为N1最小包围盒包含QR，则N1首先被访问; 继续访问N1的子节点；又N3，N4与QR相交，则继续访问N3，N4的子节点，从而最终找到N4中的n5。在R树中，点查询其原理与范围查询类似。为了查询n7，N1，N2同时被访问，且最终在N6中找到n7，尽管n7不是N1的子节点，但N1同样被访问了。其原因是R树节点的最小包围矩形之间有重叠导致，可能会有多个搜索路径。而R*树是R树的变体[40]，其通过减少节点MBR之间的重叠关系从而降低多余的节点访问次数。

图1-1 R树示意图

3DR-Tree是由Y. Theodoridis和M.Vazirgiannis等人在1996年提出的时空索引结构[41]。核心思想是将时间视为对象的另一个空间维度，并使用R-Tree执行三维空间索引。也就是说，时间段[T1，T2]中空间位置（X，Y）处的运动物体可以由从（X，Y，T1）到（X，Y，T2）的三维空间线段表示。传统空间R树的二维最小外包矩形增加了维度T（时间）并成为三维最小外包立方体。 3DR-Tree可以被认为是R-Tree的多维扩展，两者都是树结构一致的。3DR-Tree的基本操作，包括节点插入和删除，节点分裂和节点查询，类似于R-Tree的基本操作。 3DR-Tree使用n + 1维空间。表示n维时间3DR-Tree的数据也有其局限性，适用于对象空间位置或范围随时间变化较小的情况，因为如果对象的地理位置或范围发生很大变化，则构建3DR-Tree 3D最小外包立方体的MBR过大，导致MBR在整个索引中大量时间和空间重叠的概率很大，这极大地影响了3DR-Tree的检索效率。

HR-Tree 的全称是History R-Tree，也属于R-Tree家族中的一员，是R-Tree的一种时空变体，该索引结构由M Nascimento和J Silva在1998年提出[42]。HR-Tree的核心思想是：采用二级索引机制，首先将时间维度数据单独分割出来并为之构建第一级索引，然后再针对各个时间片下的空间数据用R-Tree作为第二级索引。HR-Tree的实例结构图如图1-2所示，其首先将所有移动对象的时间戳信息提取出来，并按照时间顺序对其进行排序，生成有序的时间戳列表，如图中的List(T0，T1，T2)然后对各个时间戳下的移动对象根据其空间位置信息构建对应的R-Tree，如图中的R1、R2和R3。为了节省存储空间，HR-Tree采用了“子树重叠”策略，策略的中心思想是：对于时间戳相邻的两棵R-Tree，若在该时间间隔内存在地理空间位置没有发生变化的对象，即两个树之间存在相同的子树，则直接对相同的子树部分进行沿用，只保留其一个版本。以图中 R1和R2 为例说明，在时间间隔[T0，T1]内，只有对象c1进行了位移，故R2 直接沿用了R1没有发生变化的子树B1和C1，仅新生成了A2 分支。并且在 A2分支中仍直接沿用了与R1相同的a1 和 b1，只新建了c2。

图1-2 HR树示意图

Hilbert R树提高了结点存储利用率，优化了R树结构[43]。Hilbert R树的思想是利用Hilbert曲线对d维空间中的数据进行线性排序，进而对结点进行排序，从而得到面积、周长最小化的结点。QR*[44]树是一种基于四叉树和R*树的空间索引结构，它结合了四叉树和R*树的优点，可以将查询范围限定在索引空间的某一部分，然后再用类似R*树的查询算法进行查询，在数据量非常巨大的应用领域中提高了查询效率。

R树及其变种树在建树过程中存在最小外包矩形之间重叠的现象，在高维空间中该重叠现象尤为严重，空间查询操作将访

问大量多余的结点。基于R树的空间查询算法的执行时间指数依赖于空间维数在高维空间中查询操作几乎是线性扫描整个数据空间。这种现象被称为“维数灾难”。

1.2.2.2 基于空间填充曲线的索引

空间填充曲线最初是由意大利科学家皮亚诺于1890 年构造的，由希尔伯特于1891年正式提出，之后空间填充曲线得到了深入的研究和广泛的应用。空间填充曲线是一类可以“填充”N维数据空间的曲线[46-50]。按照特定的填充规则，空间填充曲线可以通过有限次数的近似运算将多维数据空间划分为许多非常小的网格。空间填充曲线像线一样穿过每个网格并按照线性顺序对这些网格编号，它对每个网格只访问 1 次，且互不交叉。空间填充曲线将空间数据点一一映射到一维空间。常用的空间填充曲线包括Z-order和Hilbert两类。m 阶空间填充曲线将二维空间划分成大小相等、互不重叠的22m个网格，将 d 维空间划分成 2dm个网格，m 阶 Z 曲线相当于对坐标轴分割了 m 次，点所在网格的 x 坐标或 y 坐标的分量为m个。当维数d一定时，曲线的阶数越高，空间

分割越细，每个网格越小。以二维空间为例，3 阶 Z曲线将数据空间划分成 22 * 3=64 个网格。图1-3是3阶 Z 曲线以及Hilbert曲线的示意图，编码值反映的是网格的位序，由网格的编码值便可得知其在空间中的位置，而通过计算数据点的编码值即可知道点具体在哪个网格，编码值的计算通过对网格坐标进行位交叉操作得到。

图1-3 3阶空间填充曲线

1.3 主要研究内容

以上内容总结了在大数据的环境下，国内外研究机构或学者针对矢量大数据相关方面开展的研究工作，其中部分研究仅针对矢量数据，但不支持其时间维度的查询，无法进行时空查询；部分研究仅支持点数据对象的存储及查询，但不支持更为复杂的线、面矢量数据；部分研究方案虽能够同时针对点，线面矢量数据进行时空查询，但仍受限于HBase平台的架构，无法取得性能上的突破，范围查询效率，k-NN查询效率仍有进一步提高和优化的空间。针对上述问题，本文拟在分布式数据库HBase的基础上，利用NoSQL数据库的优势，提出一套完整的存储组织、时空索引、查询检索的方案，使该系统能够在大数据场景下存储海量的时空矢量数据对象，且同时支持低延迟的时空查询、k-NN查询。具体的研究内容如下：

1）时空矢量对象的索引设计及存储

时空索引设计是时空查询算法的核心，本课题设计的时空索引主要是时空串接编码。在数据集存储过程中，分别设计元数据表，编码表，以及数据表，从而支持数据的范围查询以及k-NN查询。

2）点、线面矢量对象的时空范围查询算法设计

针对点矢量对象，设计时空范围查询的算法；针对线面矢量对象，则引入层

级的概念，从时空串接编码的角度设计相关时空范围查询算法。

3）时空k-NN查询算法设计及优化策略

针对点线面矢量对象，设计时空k-NN查询算法。并通过考虑空间分布，自

适应设定搜索网格的边长，从而提高k-NN查询的效率。

1.4 论文组织结构

本论文一共分为6个章节，各章节的内容安排如下所述：

第一章为绪论，首先介绍本课题的研究背景以及说明本研究的意义所在；然后简要阐述本课题研究的相关技术在国内外的研究和发展现状；接着重点列举本课题所完成的主要工作内容；最后介绍本文的撰写逻辑与章节安排。

第二章则重点介绍Hadoop平台的概述，以及HBase数据库的系统架构，逻辑存储及物理存储。

第三章对时空矢量对象的索引设计及存储进行详细介绍，对于索引设计，分别介绍串接编码的不同串接方式，对于存储方案则从元数据表、编码表、数据表进行介绍，并在章末介绍序列化工具及数据的批量导入流程。

第四章主要详细介绍本课题中，对于点、线面的时空范围查询，k-NN查询算法设计及其优化算法。

第五章主要介绍时空范围查询以及k-NN查询的实验。在时空范围查询实验中，主要进行时空串接编码的性能对比，以及各自应用场景，以及范围查询中最大递归次数与平均查询时间的性能对比。在k-NN查询算法中，则验证不同参数k下算法的性能以及考虑数据分布后k-NN算法的优化。

第六章对本论文所做的技术工作与研究成果进行总结，并以此为基础提出进一步优化的方案，制定下一步的研究工作计划。

指　标

疑似剽窃文字表述

1. 1.2 国内外研究现状
   1.1.1 时空数据库的研究现状
   空间数据库是管理时空数据的有效手段，也是数据查询、分析应用的基础。目前，海量时空大数据的存储和管理主要采用三种方式：一是基于分布式扩展的关系型数据库进行存储；二是基于分布式文件系统进行存储；三是基于非关系型NoSQL数据库进行存储。
2. 非关系型NoSQL数据库
   NoSQL，即Not Only SQL，泛指非关系型数据库。
3. 由于非关系型NoSQL数据库具有分布式、可扩展性、不需要预先定义表结构等诸多优点，近年来深受研究学者和商业公司的青睐，在矢量大数据存储和管理领域也有很多研究工作。
4. 基于非关系型NoSQL数据库存储空间大数据不仅方便部署，同时也能够很好地对多源空间数据进行高效集成，有利于数据密集型的并行计算。
5. 时空索引技术

时空数据是一种描述空间状态随时间变化的数据类型，有助于帮助人们了解过去、把握现在、预测未来，当前采集到的科学数据基本是带有时间属性的，时间维度的增加极大地丰富了空间对象的数据信息。时空数据查询是时空数据分析处理的重要手段，而时空数据索引技术是时空数据查询的基础。

6. 3DR-Tree可以被认为是R-Tree的多维扩展，两者都是树结构一致的。3DR-Tree的基本操作，包括节点插入和删除，节点分裂和节点查询，类似于R-Tr
7. HR-Tree的核心思想是：采用二级索引机制，首先将时间维度数据单独分割出来并为之构建第一级索引，然后再针对各个时间片下的空间数据用R-Tree作为第二级索引。
8. 为了节省存储空间，HR-Tree采用了“子树重叠”策略，策略的中心思想是：对于时间戳相邻的两棵R-Tree，若在该时间间隔内存在地理空间位置没有发生变化的对象，即两个树之间存在相同的子树，则直接对相同的子树部分进行沿用，只保留其一个版本。
9. Hilbert R树的思想是利用Hilbert曲线对d维空间中的数据进行线性排序，进而对结点进行排序，从而得到面积、周长最小化的结点。
10. R树及其变种树在建树过程中存在最小外包矩形之间重叠的现象，在高维空间中该重叠现象尤为严重，空间查询操作将访问大量多余的结点。基于R树的空间查询算法的执行时间指数依赖于空间维数在高维空间中查询操作几乎是线性扫描整个数据空间。这种现象被称为“维数灾难”。

1.2.2.2

11. 空间填充曲线最初是由意大利科学家皮亚诺于1890 年构造的，由希尔伯特于1891年正式提出，之后空间填充曲线得到了深入的研究和广泛的应用。
12. 按照特定的填充规则，空间填充曲线可以通过有限次数的近似运算将多维数据空间划分为许多非常小的网格。空间填充曲线像线一样穿过每个网格并按照线性顺序对这些网格编号，它对每个网格只访问 1 次，且互不交叉。空间填充曲线将空间数据点一一映射到一维空间。
13. 当维数d一定时，曲线的阶数越高，空间

分割越细，每个网格越小。以二维空间为例，3 阶 Z曲线将数据空间划分成

14. 反映的是网格的位序，由网格的编码值便可得知其在空间中的位置，而通过计算数据点的编码值即可知道点具体在哪个网格，编码值的计算通过对网格坐标进行位交叉操作得到。
15. 研究内容

以上内容总结了在大数据的环境下，国内外研究机构或学者针对矢量大数据相关方面开展的研究工作，其中部分研究

16. 提高k-NN查询的效率。

1.4 论文组织结构

本论文一共分为6个章节，各章节的内容安排如下所述：

第一章为绪论，首先介绍本课题的研究背景以及说明本研究的意义所在；然后简要阐述本课题研究的相关技术在国内外的研究和发展现状；接着重点列举本课题所完成的主要工作内容；最后介绍本文的撰写逻辑与章节安排。

第二章则重点

17. 算法的优化。

第六章对本论文所做的技术工作与研究成果进行总结，并以此为基础提出进一步优化的方案，制定下一步的研究工作计划。

3. 第2章技术理论基础

总字数：7343

相似文献列表

去除本人已发表文献复制比：42.7%(3133)　　文字复制比：42.7%(3133)　　疑似剽窃观点：(0)

1	Java之美[从菜鸟到高手演练]之Hadoop原理及架构 - 智慧演绎，无处不在 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	20.0%（1472） 是否引证：否
2	分布式计算框架Hadoop - John's blog - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2013	19.8%（1457） 是否引证：否
3	分布式计算框架Hadoop - yarsen的专栏 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	19.8%（1457） 是否引证：否
4	被动雷达系统控制平台的设计和实现初稿2.0(1) -《学术论文联合比对库》- 2017-04-24	14.6%（1072） 是否引证：否
5	3911173__电子政务云平台的设计与实现 -《学术论文联合比对库》- 2017-03-28	14.5%（1062） 是否引证：否
6	基于Hadoop的电子政务平台设计与实现 龙琦(导师：刘晓玲;王全贵) -《湖南大学硕士论文》- 2017-04-10	14.4%（1059） 是否引证：否
7	面向教学应用的虚拟现实模型构建及场景管理关键技术研究 -《学术论文联合比对库》- 2013-03-22	12.6%（928） 是否引证：否
8	bs_随_201021111111 随 -《学术论文联合比对库》- 2013-03-28	12.6%（928） 是否引证：否
9	201021110001-l1l_7 l -《学术论文联合比对库》- 2013-04-12	12.6%（923） 是否引证：否

10	基于Hadoop的空间矢量数据的分布式存储与查询研究 陈俊欣(导师：张凤荔) -《电子科技大学硕士论文》- 2016-03-18	9.5%（698） 是否引证：否
11	201321060644陈俊欣 -《学术论文联合比对库》- 2016-03-21	9.3%（680） 是否引证：否
12	一种ABE的设计与实现 -《学术论文联合比对库》- 2015-04-08	8.5%（626） 是否引证：否
13	李军 李军 -《学术论文联合比对库》- 2015-09-13	7.3%（537） 是否引证：否
14	媒体稿件管理平台的设计与实现 汉超(导师：孔令波) -《北京交通大学硕士论文》- 2017-06-01	6.5%（480） 是否引证：否
15	921_J201163342_冯学龙 冯学龙 -《学术论文联合比对库》- 2015-05-15	6.3%（465） 是否引证：否
16	企业电子发票信息系统的设计与实现 朱口天 -《学术论文联合比对库》- 2017-09-26	5.8%（427） 是否引证：否
17	论文9.30——董再旺 董再旺 -《学术论文联合比对库》- 2014-09-30	5.7%（420） 是否引证：否
18	921_J201163342_冯学龙 冯学龙 -《学术论文联合比对库》- 2015-05-26	5.5%（404） 是否引证：否
19	Hadoop分布式文件系统架构和源码分析报告-百度文库 -《互联网文档资源（http://wenku.baidu.c）》- 2012	5.5%（402） 是否引证：否
20	HBase多条件复杂查询的实现方法研究 廖一陈(导师：沈波) -《北京交通大学硕士论文》- 2017-03-01	5.4%（398） 是否引证：否
21	98_陈俊欣_基于Hadoop的空间矢量数据的分布式存储与查询研究 陈俊欣 -《学术论文联合比对库》- 2016-03-10	5.4%（396） 是否引证：否
22	基于Hadoop的数据统计系统的设计与实现 叶溟 -《学术论文联合比对库》- 2014-06-11	4.2%（307） 是否引证：否
23	27-叶溟 叶溟 -《学术论文联合比对库》- 2014-06-06	4.2%（307） 是否引证：否
24	Hadoop平台下基于移动充电器模式的电动汽车充电研究 张祥民 -《学术论文联合比对库》- 2017-03-09	3.5%（259） 是否引证：否
25	姚晓闯_B1311686_矢量大数据管理关键技术研究 姚晓闯 -《学术论文联合比对库》- 2017-06-01	2.4%（178） 是否引证：否
26	赵龙_S20153080844_基于Hadoop的全国耕地连片度计算方法研究 赵龙 -《学术论文联合比对库》- 2017-11-27	1.8%（130） 是否引证：否
27	基于大数据技术的人口数据分析平台设计与实现 杨麟(导师：章韵) -《南京邮电大学硕士论文》- 2018-11-14	1.7%（126） 是否引证：否
28	数据通信网分布式测量系统的设计与实现 尚立(导师：高会生;许俊现) -《华北电力大学硕士论文》- 2018-06-01	1.3%（99） 是否引证：否
29	基于MapReduce的结构化查询机制的设计与实现 范波(导师：段翰聪) -《电子科技大学硕士论文》- 2011-03-25	1.2%（91） 是否引证：否
30	使用HBase Coprocessor协处理器 - - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	1.1%（82） 是否引证：否
31	交通安防大数据的实时快速检索关键技术研究 雷力(导师：刘鹏) -《浙江大学硕士论文》- 2017-01-14	0.5%（40） 是否引证：否

原文内容

第2章技术理论基础

2.1 Hadoop概述

Hadoop是一个由Apache基金会所开发的分布式系统基础架构。用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力进行高速运算和存储。 Hadoop的框架最核心的设计就是：HDFS和MapReduce。HDFS为海量的数据提供了存储，而MapReduce则为海量的数据提供了计算。

1.1.1 分布式文件系统HDFS

HDFS [51]架构原理采用master/slave架构。一个HDFS集群包含一个单独的NameNode和多个DataNode。NameNode作为master服务，它负责管理文件系统的命名空间和客户端对文件的访问。NameNode会保存文件系统的具体信息，包括文件信息、文件被分割成具体block块的信息、以及每一个block块归属的DataNode的信息。对于整个集群来说，HDFS通过NameNode对用户提供了一个单一的命名空间。DataNode作为slave服务，在集群中可以存在多个。通常每一个DataNode都对应于一个物理节点。DataNode负责管理节点上它们拥有的存储，它将存储划分为多个block块，管理block块信息，同时周

期性的将其所有的block块信息发送给NameNode。图2-1为HDFS系统架构图，主要有三个角色，Client、NameNode、DataNode。

图2-1HDFS架构图

文件写入时：Client向NameNode发起文件写入的请求。NameNode根据文件大小和文件块配置情况，返回给Client它所管理部分DataNode的信息。Client将文件划分为多个block块，并根据DataNode的地址信息，按顺序写入到每一个DataNode块中。当文件读取：Client向NameNode发起文件读取的请求。NameNode返回文件存储的block块信息、及其block块所在DataNode的信息。Client读取文件信息。HDFS 数据备份HDFS被设计成一个可以在大集群中、跨机器、可靠的存储海量数据的框架。它将所有文件存储成block块组成的序列，除了最后一个block块，所有的block块大小都是一样的。文件的所有block块都会因为容错而被复制。每个文件的block块大小和容错复制份数都是可配置的。容错复制份数可以在文件创建时配置，后期也可以修改。HDFS中的文件默认规则是write one（一次写、多次读）的，并且严格要求在任何时候只有一个writer。NameNode负责管理block块的复制，它周期性地接收集群中所有DataNode的心跳数据包和Blockreport。心跳包表示DataNode正常工作，Blockreport描述了该DataNode上所有的block组成的列表。

备份数据的存放是HDFS可靠性和性能的关键。HDFS采用一种称为rack-aware的策略来决定备份数据的存放。通过一个称为Rack Awareness的过程，NameNode决定每个DataNode所属rack id。缺省情况下，一个block块会有三个备份，一个在NameNode指定的DataNode上，一个在指定DataNode非同一rack的DataNode上，一个在指定DataNode同一rack的DataNode上。这种策略综合考虑了同一rack失效、以及不同rack之间数据复制性能问题。副本的选择：为了降低整体的带宽消耗和读取延时，HDFS会尽量读取最近的副本。如果在同一个rack上有一个副本，那么就读该副本。如果一个HDFS集群跨越多个数据中心，那么将首先尝试读本地数据中心的副本。安全模式：系统启动后先进入安全模式，此时系统中的内容不允许修改和删除，直到安全模式结束。安全模式主要是为了启动检查各个DataNode上数据块的安全性。

1.1.2 并行计算框架MapReduce

MapReduce[52]编程模式是由Google于2004年研发，主要用于超大集群环境下TB级海量数据的并行处理和计算。MapReduce模型由Map“映射”阶段和Reduce“规约”两个阶段组成。它是以一组键值对Key-Value的集合作为输入，而另一对键值对为输出，因此这种编程模式特别适合于非结构化和结构化的海量数据的搜索、挖掘，分析和机器学习等。Map Reduce 框架这种“分而治之”的思想运用有许多方面，当数据量足够大的时候，许多操作就会分派节点来进行，整个集群中则是主节点 Job Tracker 管理下的各个子节点 Task Tracker 一起来完成，然后所得到每个子节点的中间结果，合并得到最终的结果。在整个并行计算的任务处理过程中，会抽象出两个函数：Map和Reduce函数。Map 会把任务分解成多个小任务，Reduce 则会整合分解后的多小任务来得到最终处理的结果.。Map Reduce 框架一直都是围绕一个键值对来进行的，无论是数据分割和数据合并，都是从一个输入到一个输出的键值对，它最关键的两个函数就是Map和 Reduce，Map 完成的任务就是得到键值对的中间值，这是需要函数输出的一系列相同的值，而Reduce 函数则是接收 Map 函数的输出，主要做一个键值对的合并关键操作，这样就能得到大量数据的一个分布式多线程操作，最后的键值对会保存下来作为记录输出，当然它也是一个键值对。从不同的需求来看，Map 和 Reduce 函数都是需要根据不同的需要而重新编写的，这样就可以利用这个并行的计算框架来解决许多分布式的关键问题，比如分布式存储与数据处理，工作调度、负载平衡，数据传输等等。因此，利用Map Reduce 处理海量的空间矢量数据是一个比较好的选择，利用Map 函数归一化然后将结果交给Reduce函数处理。采用Map Reduce 的计算模式将传统的空间查询操作并行化，理论上可以有效提高空间数据查询的效率。

2.2 数据库HBase介绍

创建于2007年2月的HBase[53-56]项目，是Google BigTable的开源实现，在Hadoop生态里是很重要的一部分，是一种持久化的、多维有序映射、稀疏的、分布式的非关系型数据库，其主要用于存储海量的结构化数据，具有可伸缩、随机实时读写、搞高可靠性的特点。在普通硬件环境的基础上，能支撑十亿量级的行和百万量级列的大表，适合应用在数据存储量大（PB级），写入性能要求高，数据结构多源，业务简单的场合。

1.1.1 HBase系统架构

HBase集群采用Master/Slave的架构，由Zookeeper服务器、主服务器Master、Region服务器（RegionServer）以及客户端（Client）等将部分组成，如图2-2所示。

图2-2 HBase系统架构图

Zookeeper为整个集群提供协调服务，维护集群Master的失效重启且保证其永远只有一个；并实时对各RegionServer节点的状态作监控，同时将其具体的状态信息向Master报备；Zookeeper还保存着-ROOT-表和.META.表的地址（每次查询发起时寻找对应Region的入口），以及每个HBase表的元数据信息，如模式设计等。

Master是HBase集群的管理节点，一方面负责在RegionServer宕机时将原来的Region重新分配至其他节点，以调整负载均衡；同时管理着用户的CRUD（建表、读取、更新、删除）操作。

RegionServer作为集群的核心部件，上面分布有Master分配的若干Region和一个HLog日志文件，由节点单独维护；同时处理来自Client的读写请求，在HDFS上进行相应的IO操作；此外，负责切分因为数据行增多或数据量增加而超过阈值的Region。

Client为用户的数据读写请求提供入口，通过远程调用协议（RPC）与Master进行通信，获取Region的节点信息，然后在RegionServer上做读写类的操作。

当用户发起数据读取请求时，Client首先访问Zookeeper获得-ROOT-的位置，接着访问-ROOT-表获取.META.的位置信息，然后访问.META.表得知用户请求的表所对应的Region位置信息，最后Client直接访问对应的Region获取用户数据。这样一个三层的查询结构，尽管需要多次网络操作，但 Client包含了cache缓存，可以提高查询效率；且当得知数据所在的RegionServer后直接访问相应节点的方式，跳过主节点，使得Master的负载较低，也不会成为集群查询的瓶颈。如此的架构设计，使得HBase有着相当出色的负载均衡和容错的性能。

1.1.2 HBase表的逻辑存储与物理存储

HBase数据与存储模型在数据模型上，HBase采取与Google BigTable相同的数据模型，以表的形式存储数据，表具体的逻辑视图如表2-1所示。HBase底层存储采用Key-Value形式存储，表拥有一个大的映射关系，其中行键（RowKey）、列族

（Column Family）、列限定符（Column Qualifier）和时间戳（TimeStamp）共同组成了Key，而Value由这4维坐标唯一确定。行键是每行数据记录的唯一标识，类似于关系型数据库的主键，按照字典序（byte order）排列；列族是列的集合，列都以列族为前缀（Column Family:Qualifier），使用表前需要事先定义好列族，但二者均支持动态扩展，以及独立检索。对数据的每一次修改操作均有一个64位整型的时间戳与之对应，即每一行数据对象可有多个时间版本，在写入RegionServer节点时系统自动赋值，并按照时间顺序降序组织，每次数据访问时优先取最新的版本。所有的单元格（Cell）数据均以字节码的形式存储，没有类型之分。在表2-1中，对于类似如下一些空白的单元值，实际的物理存储上没有为其开辟空间，如果请求了空白值，会返回NULL，因此，HBase表也可以设计得非常稀疏。

表21 HBase逻辑表结构

Rowkey Timestamp Column Family

Qualifier1 Qualifier2

Key1 T3 Value11 Value21

T2Value22

T1 Value12 Value23

Key2 T4 Value13

在存储模型上，HBase的存储、权限控制和检索都基于列族。每张表由若干个Region组成（一个Region包含了一定行键范围内的数据），每个Region当数据行数达到设定阈值后，一个Region会如细胞分裂般拆分为2个，而这些Region都分散地存储在不同的RegionServer节点上，但一个Region内部的数据只会存储在同一节点。Region是HBase中分布式负载均衡的原子单位，但不是存储的原子单元，一个Region会根据该表列族的个数产生对应的Store，每个Store负责管理一个列族，每个Store都有1个MemStore，以及若干StoreFile，这取决于改该列族上的数据量，甚至可以没有StoreFile文件。如图2-3所示。

图 23 HBase存储模型

MemStore位于RegionServer服务器的主内存中，当写数据时，会首先写入到MemStore中，并在MemStore中进行排序，当数据累计达到某一阈值后，创建新的MemStore，并将原来的MemStore添加到刷新队列里，等待MemStore中的内容Flush到硬盘里，形成一个StoreFile文件。同时，为记录此前的数据变更已持久化到硬盘，还需在Zookeeper里作备份。这样设计的原因是：HBase的数据实际是存储在HDFS上的，而为了检索优化的考虑，必须满足HBase的行键按照排序存储的需求，但HDFS上的数据又不可随意修改，换言之，即必须排好序再作持久化，这就产生了排序与写数据的矛盾。为克服这个困难，HBase将数据的写入过程做了一个中转，先把新数据缓存到MemStore，利用内存的高速读写能力做好排序工作，然后再持久化到HDFS。

StoreFile是只读文件，一旦创建无法修改，所以HBase的更新操作没有删除改写，而是一个将新数据以一个个StoreFile文件不断append至数据库中的过程。当StoreFile文件数量超过一定阈值后，会自发地触发一次合并（compact）操作，对相同RowKey的数据项作合并，生成一个更大的StoreFile。当然StoreFile也不会无限扩大，如果StoreFile的文件大小达到阈值时，又会平均地分裂（split）该StoreFile文件为两个新的StoreFile。因为排序工作已经在内存中完成了，StoreFile中的数据都是有序的，而且StoreFile包含了内存索引，所以合并效率特别高。HBase即以这样的（compact-split）机制，控制着StoreFile数量与大小之间的平衡。

表 22 HBase数据模型与物理存储文件映射表

数据模型物理存储模型

行键 Region

列族 Store

Cell单元值 StoreFile/HFile

--- MemStore

--- HLog

除了StoreFile和MemStore，数据还以HLog的格式存储在HDFS上，以防在系统出错或宕机时，如果某个节点掉线了，其MemStore的内存数据就不会丢失。HLog的原理与MemStore相似，让每个RegionServer都有一个日志对象，当数据写入时，同时在MemStore和HLog中刷写一份相同的数据，HLog定期作数据持久化，然后覆盖已存在的数据。当RegionServer节点掉线了，Master主机会向Zookeeper发消息，Master会先处理HLog，将不同Region的HLog数据进行划分，放到对应的Region目录下，然后重新分配失效的Region。分配到Region的节点在加载Region的过程中，如果有HLog，就会读取其数据写入MemStore中，然后刷新到StoreFile，完成数据恢复。HBase的数据模型与实际物理存储模型之间的对应关系大致如表2-2所示。

1.1.3 HBase协处理器

HBase与传统的关系型数据库相比，其写入性能高了一个数量级，但其查询性能要低一个数量级。通常访问HBase数据的方式是，使用Scan方法做全表扫描或者Get方法直接获取，根据需要可以使用Filter过滤掉冗余的部分数据，最后在获取到的数据上进行业务运算。但如果查询结果非常庞大，对如此大体量的数据做传输势必造成极大的网络延迟，而且在客户端还要进行复杂的运算，对服务器的内存和CPU也是巨大的挑战，这样的过程十分浪费时间。除此之外，0.92版本之前的HBase还存在一个问题，即很难进行简单的排序、求和、计数等操作。因此为了应对这样的困境，协处理器机制应运而生。

HBase的协处理器（Coprocessor）发端于Jeff Dean的一篇关于Google BigTable的Coprocessor特性的演讲，其主要思想即是把复杂的处理环节，分发到各个RegionServer节点上，使得大部分计算可以在数据库端处理完成，从而提高了处理的效率；加之服务器只返回处理后的结果给客户端，客户端只需进行较小的部分计算，这既减小网络传输的损耗和耗时，也缓解了客户端的压力，在性能方面也是很大的提升。

相比于Google BigTable的协处理器，同样将计算过程转移到存储节点，HBase Coprocessor最大的区别在于它是一套在Master和RegionServer进程内运行的框架，可在运行期间动态加载，然后执行相应的功能。HBase Coprocessor的运行原理非常类似于MapReduce的分析处理组件，但极大简化了MapReduce的处理模型。MapReduce的加载在初始化阶段耗时长，且每次运行又要重新加载，难以满足线上请求的即时性；而Coprocessor偏轻量级，能动态加载，加载速度快。

HBase Coprocessor有2种不同的类型，分别为Observer（观察者）和Endpoint（终端）。Observer，是在某一特定事件（如Get或Put数据）发生前或发生后触发，执行相应的回调函数。其位于客户端和HBase数据库之间，在二者发生请求调用和响应返回的过程中，修改数据访问，适用于统计行数或构建辅助索引等操作，类似于设计模式中的Observer模式和关系型数据库中的触发器。以RegionObserver为例，调用过程如图2-4所示：

图24 Observer调用原理图

Endpoint把用户自定义的功能代码加载到服务器端，当执行时可以通过客户端的RPC调用触发。通过扩展了Hbase的RPC协议，将数据库端代码的执行结果返回给客户端作进一步处理，非常适合利用Endpoint来实现求和或求均值等聚合计算，类似于关系型数据库的存储过程。以Endpoint统计行数为例，调用过程如图2-5所示。

图25 Endpoint调用原理图

HBase的Coprocessor机制把任意计算逻辑推到托管数据的HBase节点（RegionServer）上，且执行代码是跨所有RegionServer并行运行的。这个特性把HBase集群从水平扩展存储系统，转变为高效的、分布式的数据存储和数据处理系统。基于Coprocessor的功能扩展开发是HBase开源社区的一大热点，本框架即是借助于Coprocessor将大量的聚合任务放在了服务端，在一定程度上以提高查询可视化的效率。

指　标

疑似剽窃文字表述

---

1. 第2章技术理论基础
   2.1 Hadoop概述
   Hadoop是一个由Apache基金会所开发的分布式系统基础架构。用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力进行高速运算和存储。 Hadoop的框架最核心的设计就是：HDFS和MapReduce。HDFS为海量的数据提供了存储，而MapReduce则为海量的数据提供了计算。
2. 一个HDFS集群包含一个单独的NameNode和多个DataNode。NameNode作为master服务，它负责管理文件系统的命名空间和客户端对文件的访问。NameNode会保存文件系统的具体信息，包括文件信息、文件被分割成具体block块的信息、以及每一个block块归属的DataNode的信息。对于整个集群来说，HDFS通过NameNode对用户提供了一个单一的命名空间。DataNode作为slave服务，在集群中可以存在多个。通常每一个DataNode都对应于一个物理节点。DataNode负责管理节点上它们拥有的存储，它将存储划分为多个block块，管理block块信息，同时周期性的将其所有的block块信息发送给NameNode。图2-1为HDFS系统架构图，主要有三个角色，Client、NameNode、DataNode。
   图2-1HDFS架构图
   文件写入时：Client向NameNode发起文件写入的请求。NameNode根据文件大小和文件块配置情况，返回给Client它所管理部分DataNode的信息。Client将文件划分为多个block块，并根据DataNode的地址信息，按顺序写入到每一个DataNode块中。当文件读取：Client向NameNode发起文件读取的请求。NameNode返回文件存储的block块信息、及其block块所在DataNode的信息。Client读取文件信息。HDFS 数据备份HDFS被设计成一个可以在大集群中、跨机器、可靠的存储海量数据的框架。它将所有文件存储成block块组成的序列，除了最后一个block块，所有的block块大小都是一样的。文件的所有block块都会因为容错而被复制。每个文件的block块大小和容错复制份数都是可配置的。容错复制份数可以在文件创建时配置，后期也可以修改。
3. NameNode负责管理block块的复制，它周期性地接收集群中所有DataNode的心跳数据包和Blockreport。心跳包表示DataNode正常工作，Blockreport描述了该DataNode上所有的block组成的列表。
   备份数据的存放是HDFS可靠性和性能的关键。HDFS采用一种称为rack-aware的策略来决定备份数据的存放。通过一个称为Rack Awareness的过程，NameNode决定每个DataNode所属rack id。缺省情况下，一个block块会有三个备份，一个在NameNode指定的DataNode上，一个在指定DataNode非同一rack的DataNode上，一个在指定DataNode同一rack的DataNode上。这种策略综合考虑了同一rack失效、以及不同rack之间数据复制性能问题。副本的选择：为了降低整体的带宽消耗和读取延时，HDFS会尽量读取最近的副本。如果在同一个rack上有一个副本，那么就读该副本。如果一个HDFS集群跨越多个数据中心，那么将首先尝试读本地数据中心的副本。安全模式：系统启动后先进入安全模式，此时系统中的内容不允许修改和删除，直到安全模式结束。安全模式主要是为了启动检查各个DataNode上数据块的安全性。
4. MapReduce模型由Map“映射”阶段和Reduce“规约”两个阶段组成。它是以一组键值对Key-Value的集合作为输入，而另一对键值对为输出，因此这种编程模式特别适合于非结构化和结构化的海量数据的搜索、挖掘，分析和机器学习等。Map Reduce 框架这种“分而治之”的思想运用有许多方面，当数据量足够大的时候，许多操作就会分派节点来进行，整个集群中则是主节点 Job Tracker 管理下的各个子节点 Task Tracker 一起来完成，然后所得到每个子节点的中间结果，合并得到最终的结果。在整个并行计算的任务处理过程中，会抽象出两个函数：Map和Reduce函数。Map 会把任务分解成多个小任务，Reduce 则会整合分解后的多小任务来得到最终处理的结果.。Map Reduce 框架一直都是围绕一个键值对来进行的，无论是数据分割和数据合并，都是从一个输入到一个输出的键值对，它最关键的两个函数就是Map和Reduce，Map 完成的任务就是得到键值对的中间值，这是需要函数输出的一系列相同的值，而Reduce 函数则是接收 Map 函数的输出，主要做一个键值对的合并关键操作，这样就能得到大量数据的一个分布式多线程操作，最后的键值对会保存下来作为记录输出，当然它也是一个键值对。从不同的需求来看，Map 和 Reduce 函数都是需要根据不同的需要而重新编写的，这样就可以利用这个并行的计算框架来解决许多分布式的关键问题，比如分布式存储与数据处理，工作调度、负载平衡，数据传输等等。因此，利用Map Reduce 处理海量的空间矢量数据是一个比较好的选择，利用Map 函数归一化然后将结果交给Reduce函数处理。采用Map Reduce 的计算模式将传统的空间查询操作并行化，理论上可以

有效提高空间数据查询的效率。

5. 2 HBase表的逻辑存储与物理存储
   HBase数据与存储模型在数据模型上，HBas
6. 持久化到HDFS。
   StoreFile是只读文件，一旦创建无法修改，所以HBase的更新
7. StoreFile中的数据都是有序的，而且StoreFile包含了内存索引，所以合并
8. 除了StoreFile和MemStore，数据还以HLog的格式存储在HDFS上，
9. Master会先处理HLog，将不同Region的HLog数据进行划分，放到对应的Region目录下，然后重新分配失效的Region。分配到Region的节点在加载Region的过程中，如果有HLog，就会读取其数据写入MemStore中，然后刷新到StoreFile，完成数据恢复。
10. 通常访问HBase数据的方式是，使用Scan方法做全表扫描或者Get方法直接获取，根据需要可以使用Filter过滤掉冗余的部分数据，最后在获取到的数据上进行业务运算。
11. 类似于MapReduce的分析处理组件，但极大简化了MapReduce的处理模型。

4. 第3章时空矢量对象的索引设计及存储

总字数：8789

相似文献列表

去除本人已发表文献复制比：13.3%(1165)　　文字复制比：13.3%(1165)　　疑似剽窃观点：(0)

1	201091303526472 -《学术论文联合比对库》- 2013-05-28	9.5%（833） 是否引证：否
2	基于Redis的矢量数据组织研究 张景云(导师：江南) -《南京师范大学硕士论文》- 2013-04-18	9.4%（824） 是否引证：是
3	Java实现数据序列化工具Avro的例子 - 专注于大数据技术研究和应用 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	2.3%（201） 是否引证：否
4	AVRO - u013061459的博客 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	1.9%（170） 是否引证：否
5	Avro序列化操作（1）：环境搭建和Schema处理 - hua245942641的专栏 - 博客频道 - CSDN.NET -《网络（http://blog.csdn.net）》- 2017	1.9%（167） 是否引证：否
6	徐迎晓_13222010150_陈钦彦_基于iOS的数据收集客户端的设计与实现 徐迎晓 -《学术论文联合比对库》- 2016-03-26	1.7%（151） 是否引证：否
7	Google Protocol Buffer持久化框架分析_rotosix -《网络（http://blog.sina.com）》- 2013	1.7%（149） 是否引证：否
8	Hadoop 生态系统 -《网络（http://www.aiweibang）》- 2016	1.6%（140） 是否引证：否
9	021_GS132147C_姚飞 姚飞 -《学术论文联合比对库》- 2016-05-04	1.2%（106） 是否引证：否
10	SA15225048_罗鹏_1 罗鹏 -《学术论文联合比对库》- 2017-09-19	1.2%（103） 是否引证：否
11	智能电视个性化推荐系统的设计与实现 罗鹏 -《学术论文联合比对库》- 2017-09-19	1.2%（103） 是否引证：否
12	Spark SQL下Parquet内幕深度解密 - 囧芝麻的博客 - CSDN博客 -《网络（http://blog.csdn.net）》- 2017	1.2%（103） 是否引证：否
13	1401220045_李长亮_幼教校园管理系统的设计与实现_荆琦_552e0567efbcedc80b87d7d631eeb0e9_20171011231145 李长亮 -《学术论文联合比对库》- 2017-10-13	0.9%（80） 是否引证：否
14	幼教校园管理系统的设计与实现 -《学术论文联合比对库》- 2017-10-11	0.9%（80） 是否引证：否
15	陈晓佳_143520085211005_面向个性化电台的实时大数据分析系统设计与实现 陈晓佳 -《学术论文联合比对库》- 2016-04-18	0.9%（76） 是否引证：否
16	MapReduce容错机制的改进研究 吴慧城(导师：李肯立) -《湖南大学硕士论文》- 2014-05-15	0.5%（40） 是否引证：否
17	基于云计算技术的化合物相似性分析系统 李杰辉(导师：张亮) -《复旦大学硕士论文》- 2012-04-25	0.4%（35） 是否引证：否
18	张琦 张琦 -《学术论文联合比对库》- 2016-01-09	0.4%（32） 是否引证：否
19	云计算环境下资源调度策略的研究与实现 李振双(导师：张洪欣) -《北京邮电大学硕士论文》- 2014-01-06	0.3%（30） 是否引证：否

第3章时空矢量对象的索引设计及存储

3.1 矢量对象

本文遵循国际标准化组织制定的标准简单要素规范，进行相应矢量数据模型和结构的设计，简单要素规范以制定一个标准的基于的方案为目的，并使其能够支持简单地理要素集的存储、提取、查询、更新。简单要素类和要素类的区别是要素类可以是点、线、面，也可以是复合模式，维护拓扑数据，可以生成拓扑类，而简单要素类是不含拓扑数据的，容量比要素类要小得多。简单要素类型有点、线、面和几何体集合。

1）点实体

点数据是一个零维的几何对象，记录几何信息点坐标（和属性代码多点是复合点类，是多个点的集合，如果这些复合点中任意两个点都是不相等的，那么这个复合点则可被称为是简单的。

2）线实体

记录两个或一系列采样点的坐标，并加属性代码。线实体又可分为曲线类、线串类、环类，其中的曲线类是一维的几何对象，通常是由顺序连接的两个或者两个以上的点构成的。简单要素规范中只定义了曲线的一个子线串类。且规定该线串中任意两个顶点之间必须线性差值，若曲线没有两次穿过同一个点，就可认为该曲线为简单的。若首尾相连，那么该曲线是闭合的；如果闭合同时又简单，则形成环、如果不闭合，则拓扑封闭，且其边界是其起始点和终点。若曲线各个顶点线性插值，则形成线串。每个连续的顶点对构成一个线段。多个线的集合还可构成复合曲线，当构成复合曲线的所有线简单时，这个复合曲线为简单的。若由多个线串构成复合曲线，则构成了复合线串。

3）面实体

面是二维的几何对象，通过记录面实体的边界来表现。简单标准规范定义

的简单面是具有一个外边界和零个或者多个内部边界。多边形类（是唯一可被实例化的简单平面。关于多边形可以定义为：多边形是一个拓扑封闭的面，其边界由一组线性环构成，且边界上任意两个环不能相交，其只能以相切的方式存在，且多边形内部是连接的点集。

4）复合多边形

复合多边形的每个子面都是多边形，任何两个复合多边形的子对象内部是不能交叉的，而且任意两个复合多边形其子对象的边界也是不能相交的，只允许在有限个点上相接。复合多边形不能有分割线和毛刺，且是内部闭合的。

说当前已有很多开源框架对于简单要素进行了实现，例如Java Topology Suite(JTS)包。JTS封装了包含点、线、面等简单要素类型，且实现了简单要素之间的拓扑计算，例如判断相交、包含等，通过利用JTS包可以直接对点线面对象进行封装。

3.2 时空索引设计

时空索引设计是时空查询算法的核心，而本课题设计的的时空索引是建立在时空立方体的基础上，则先引入时空立方体的概念，再详细介绍时空索引的设计。

3.1.1 时空立方体

将空间和时间抽象起来，通过联合表达为时空一体是时空立方体模型的核心思想。如图3-1所示，为经度、纬度和时间形成的三维时空立方体示意图。时空单元是时空立方进行时空维度剖分形成的构成时空立方的最小粒度，其具有以下两个特点：

1）时空单元由时间范围和空间范围来标识。

2）所有时空单元覆盖的时间范围粒度和空间范围的粒度是一样的，即具有相同的时间分辨率和空间分辨率。需要指出的是，时空单元的分辨率是因应用而异的，如可以将时间分辨率设为1小时，空间分辨率设为1平方公里。

考虑到时间轴没有上界，本课题将时间轴划分为多个区间，即时间Bin，形成多个时空立方体，其中的每一个时空立方体都在指定的粗粒度时间范围，空间范围内。而每个时空单元则具有不同的细粒度时间范围，空间范围。

时空单元具有固定的分辨率，使得时空范围与时空单元之间的换算必须是简单高效的。为此，我们可以通过时空单元来粗略表达时空对象的时空范围，从而能够借助时空单元来快速定位时空对象。具体来说，首先从时空范围快速计算出候选的时空单元（集），然后从时空单元（集）中筛选出符合要求的时空

对象。

图 31 时空立方体示意图

3.1.2 时空编码

时空编码是指对时空立方体的时空单元进行编码，通过降维方法将3维的时空单元表示为1维的二进制串，从而能够快速检索出满足时空条件的时空对象。本论文主要采取时空串接编码，即先分别编码时间和空间，并将两种编码值串接在一起。编码首要考虑的问题是时空邻近性保持，即时空邻近的时空单元的1维编码值也应尽可能的接近。时空串接编码分别编码时间和空间，其中的核心问题是如何编码时间值和空间值，前者是在某一时间顺序和粒度下对时间信息进行编码，而后者则可看作是空间填充曲线的编码问题。

（1）时间编码

时间编码的主要目的是将字符串（UTC时间或北京时间）或者时间戳表达的时间信息在指定的时间顺序和粒度下进行的编码表达，结果为满足应用需要的1维编码值，主要考虑三方面因素：1）粒度，2）起点，3）顺序。

时间信息具有显著的粒度性，常用的粒度有年（Y）、月（M）、周（W）、日（D）、时（W）、分（M）和秒（S），在不同的应用场景下关注的时间粒度是不同的。例如，一些应用如交通场景要求时间信息精确到秒，而另一些应用如气候变化等时间信息准确到天就可以满足需求。合适的粒度是必要的，这样既可以避免细粒度时间带来的冗长编码，也可以尽可能减少粗粒度时间导致的信息损失。

考虑到编码的高效性，编码起点是必须的，否则可能需要对应用不感兴趣的一大段时间（如公元前）进行毫无用处的编码。默认情况下，本项目将格林威治时间1970年1月1日00时00分00秒，作为时间编码的起点，但也可以为数据集指定其时间编码起点。显然，编码起点也是有粒度的，假设编码粒度为小时，那么默认的编码起点则为1970年1月1日00时。

时间顺序是指在时间信息中，时间粒度成员的先后顺序，通常顺序可能从年、月、日到时、分、秒，但应用也可以根据其时间查询的特征，指定不同的时间顺序。在不同的时间顺序下，同一时间的编码是不同。假设时间粒度选为日，1970年2月1日在YMD顺序下的编码为32，而在DMY顺序下的编码为2。显然，后者对于可不同月而同1天的数据分析较为有利，其原因是编码值是临近的。

（2）空间编码

空间填充曲线是一种对高维空间进行降维表达的常用方法，该方法将空间区域划分成大小相同的网格，再根据一定的方法对这些网格进行编码，并为每一个网格指定唯一的编码值。空间填充曲线主要考虑的是在一定程度上保持空间网格在编码序列中的空间临近性，即空间上相邻的网格在编码序列中的编码值相似。常用的空间填充曲线包括Z-order和Hilbert两类。本项目在默认情况系采用Z-order编码系统，如图3-2所示为3阶的Z曲线。

图3-2 Z-order曲线

（3）串接方式

时空编码表达是对时空单元的编码进行形式化的表达，作为数据集的一项元数据，即每一个数据集应说明其时空单元的编码结构和方式。本课题拟采用Json来表达时空编码。为了实现一套统一的串接编码接口，则定义如下Json文件。由于为了不同查询需求，一个数据对象集可能有很多个编码表，其中id为其编码表的标识；Structure则表示为串接编码的策略，如”S_T”，”T_S”，”T_S_T”分别对应ST，TS，TST串接编码；timeCoarseBin则对应粗粒度时间；order则为输入时间的形式；在encoding中，精度precision为空间编码所需的bit位。

```
{
"id":1,
"encoding":{
"precision":40,
"order":"yyyyMMdd"
},
"structure":"S_T",
"timeCoarseBin" : "D"
}
```

3.2 存储系统方案设计

第2章介绍了HBase的基本架构，且3.1介绍了本文的时空索引设计算法，本小节将在此基础上进一步介将时空点、线面对象存储至HBase数据库的存储方案。本文的存储系统方案设计主要由元数据，编码表，数据表三部分构成。其中元数据主要记录数据集的元信息，如包含数据的编码方案，矢量类型，空间参考等信息；编码表用于时空范围查询，k-NN查询，其主要是为了完成时空矢量对象与时空立方体的映射关系，且编码表中只存储矢量对象的空间位置信息，不支持属性查询；数据表是为了时空矢量对象的属性查询。且对于点、线面数据，元数据表和数据表没有大的区别，而由于其几何属性的差别，编码表的方式有所不同。

图3-3 时空矢量对象数据库结构

3.2.1 元数据表

在存储时空矢量对象时，在创建数据集的同时需要写入对应的元数据信息， 一个时空要素对象数据集的元数据在元数据表中则存储为一条记录，其行键为数据集的名称。元数据表中必要的字段存储在列族E中，其他由用户定义的字段放置在列族F下，通常为列族E中字段的字典信息，包括字段类型、取值范围等。元数据表的结构如表3-1所示。

表3-1 时空要素对象子库元数据表结构

RowKey E

DatasetName CS DT ST SR BB TN DE

F

F1 F2 F3 …

表3-2 时空要素对象子库元数据表结构说明

列族字段名称字段说明取值类型备注

E CS 编码方案字符串例如：{"id":1,"encoding":{"precision":40,"order":"yyyyMMdd"},"structure":"S_T","timeCoarseBin" : "M"}

DT 矢量对象类型整型例如：点-1，线-2，面-3

ST 起算时间点字符串例如：“1970-01-01 00:00:00”

SR 空间参考整型例如：4326

BB 数据集最小包围盒字符串例如：(X1, Y1, X2, Y2)

TN 数据集中记录总数整型例如：8000

DE 数据集的描述信息字符串例如：xxx市xxx区2019年1:500比例尺地形图更新数据集

F F1 自定义自定义

F2

F3

…

3.2.2 编码表结构

时空对象存在于地理空间中，时间维度上表现为状态，因此与时空单元具有天然的对应关系。为了实现快速时空检索

，两者必须进行关联和映射。在时空立方中，1个时空单元中可能存在多个时空对象，而1个时空对象可能跨越多个时空单元，因此，时空对象与时空单元是一种n:m的映射关系。而对于点对象，则为1对1的关系，对于线、面对象则有可能是多对多的关系。且由于点与线面不同的映射关系，其编码表中行键也有所区分。

3.2.2.1 时空点对象

时空点对象的存储模式是将时空单元中的点对象按列分别存储。该模式的编码表中的一条记录代表某一时空单元中存储的时空点对象，其行键为时空单元的时空编码STC，编码方案可由元数据表确定。编码表包含一个列族F和若干子列ObjectID。一个时空单元中可能包含多个时空点对象，按列存储模式将各点对象的唯一标识作为列族F的子列，并且在每个列中存储点对象的几何信息。编码表的结构如表1-3所示。其中ObjectID1，ObjectID2，ObjectID3为对象的唯一标识。Value为时空矢量对象空间属性，时间属性的序列化数据。

表3-3 点对象编码表结构

RowKey F

STC ObjectID1 ObjectID2 ObjectID3 …

Value1 Value2 Value3 …

3.2.2.2 时空线面对象

由于线面对象的几何属性与点对象不同，线面对象在空间上可能跨越多个时空单元。若使用最细粒度的空间单元，可以更细致的逼近线面对象的几何形状，但所需的空间单元数量会非常多。在编码表存储时，空间编码是行键时空编码的重要组成部分，若一个线面对象由多个空间单元组成，则需在这些空间单元所在的行键中都进行冗余存储，如此一来一份线面对象则备份多次，浪费了编码表的存储空间。而如果找到一个合适的层级，找到合适粒度的空间单元，使线面对象能够能被少数空间单元包含，则可以减少冗余存储，提高存储空间效率。其具体方法如下。

找到线面对象的最小包围矩形，并求出其左下角，右上角的空间坐标在预设最大层级的Z曲线编码，其编码分别为Z1，Z2。

利用二进制运算，求出Z1,Z2编码的最大公共前缀，则可以求出Z1，Z2所在的公共父节点，即最小包围矩形所在的一个四叉树节点的Z曲线Range。此时会出现以下三种情况。

1）最小包围矩形MBR包含于一个四叉单元内部，且其与四叉树节点单元中心相交，如图中的多边形1，其Z1，Z2都为0，二进制编码为0000，则其公共前缀bit位为4，对应的层级L应为公共前缀bit位的一半，则为2。

2）如图中的多边形2 ，Z1=8，二进制编码为1000， Z2=10，二进制编码为1010，则其公共前缀为2，对应的层级为1，在其对应的象限Range中，其最小包围矩形不与其中心相交，则可对该象限Range再划分一次，最终可以得到两个与之相交的象限Range包含最小包围矩形，则对应层级为2。

3）如图中的多边形3，其最小包围矩形中，Z1=6，二进制编码为0110，Z2=13，二进制编码为1101，其公共前缀为0，对应层级则为0，但由于最小包围矩形不与其中心相交，则对该象限Range，第0层的编码0划分至第一层的0，1，2，3四个象限。其与1，3象限相交，且同意不与中心相交，则可继续往下划分，最终得到2层级的4个象限Range。

图3-4 多边形划分示意图

以上是线面对象的冗余备份机制，而在存储过程中，线面对象其编码表的行键为时空单元的时空编码STC。对于点对象，其所有点数据所在的空间单元，都是由确定的精度precision，即最大层级划分。由于线面对象的冗余备份机制，各个线面对象所对应的空间单元所在的层级是不同的，需在时空编码中加入相应的层级信息，其行键是层级L与时空编码的组合。时空线、面对象编码表的结构类似于点对象按列存储的编码表结构，在线面对象的编码表中，同样只有一个列族F和若干子列ObjectID，每一个子列存储线、面对象的的时空信息的序列化数据。时空线、面对象编码表结构如表3-4所示。其中ObjectID为线面对象的唯一标识id。Value存储序列化之后的几何信息。一个线、面对象可能跨越不超过四个时空单元，所以同一个线面对象的序列化数据可能存在于至多4个行键中。

表3-4 线、面对象编码表结构

RowKey F

ObjectID1 ObjectID2 ObjectID3 …

L_STC1 Value1

L_STC2 Value1 Value2

L_STC3 Value1 Value2 Value3

L_STC4 Value1 Value2 Value3

…

3.2.3数据表结构

时空点对象的数据表只包含一个列族F，且该列族下只有一个子列V，其中存储完整的点对象信息。数据表的行键为点对象的唯一标识符，通过数据表的行键能够访问该点对象的几何与属性信息。数据表的结构如表3-5所示。为了便于其他的属性查询需要，除了对象ID也可以使用其他属性作为行键。

表3-5 时空点对象数据表结构

RowKey F

ObjectID1 V

Value

3.3 数据序列化和批量导入

3.3.1 序列化

HBase中存储的数据均为字节流，因此存储矢量对象时，需要将矢量对象的所有信息转化为字节流来存储，这个过程称之为“序列化”。反过来，将字节流转化为矢量对象称之为“反序列化”。Avro是Hadoop中的一个子项目，也是Apache中一个独立的项目，是一个数据序列化的系统，可以将数据结构或对象转化成便于存储或传输的格式。它由Doug Cutting在2009年牵头开发，目的是用来支持数据密集型的应用，适合于远程或本地大规模数据的存储和交换。Avro是一个基于二进制数据传输高性

能的中间件，是一种与编程语言无关的序列化系统，它提供了丰富的数据结构类型、快速可压缩的二进制数据格式、存储持久性数据的文件容器、远程过程调用（RPC）以及简单的动态语言结合功能。当前除了Apache Avro系统外，Thrift, Protocol Buffers等系统也提供类似功能，但与其他系统相比，Avro序列化系统具有以下优势：

（1）动态类型：Avro不需要生成代码。数据总是依赖于模式（schema），该模式允许在不生成代码、静态数据类型等情况下对数据进行完全处理。Avro数据的读写操作非常频繁，而这些操作都需要使用模式，这样就减少了写入每个数据的开销，使得序列化过程快速而又轻巧。这种数据及其模式的自我描述方便于动态脚本语言的使用，有助于构建通用数据处理系统和语言。

（2）无标记数据：由于在读取数据时存在模式，因此需要用数据编码的类型信息要少得多，从而导致较小的序列化大小，这有助于数据的压缩。

（3）不用手动分配字段ID：当模式发生更新时，旧模式和新模式都始终存在，因此负责读数据的应用程序可以继续处理而无需做任何改动，这个特性使得它特别适合用于流式数据等消息系统上。

基于以上优势，本文采用Avro进行数据的序列化，Avro在读写文件时用到的模式schema通过JSON语言来描述，数据最终被序列化成二进制数组字符串。本课题在Avro的模式定义时，直接将数据序列化为JTS的简单要素类型对象，这样做的优点有以下几点：

（1）JTS在对基本要素进行定义时，需要对基本属性进行设置，包括简单要素的空间属性。此外，JTS还允许用户自定义点的其他属性，例如时间等，如此则可以同时保存空间时间信息，因此利用JTS结合Avro进行序列化可以方便地对任意属性的矢量对象进行定义和封装。

（2）将简单要素以JTS的几何对象来保存和序列化，反序列化之后直接得到简单要素对象，而不是字符串，因而去除了解析字符串得到简单要素属性的过程，减少了得到数据后对数据进行解读的时间，提高整体查询效率。

（3）JTS不仅对简单要素类型进行了定义，还包含多种拓扑关系的计算，例如判断相交、包含、毗邻关系等，因此在对要素进行精过滤时，可以直接通过JTS的拓扑计算功能，快速筛选出符合查询条件的要素。

（4）由于一个时空矢量对象具有经度、纬度、时间等多种属性值，直接将具有属性信息的字符串进行序列化后，得到的二进制数组长度较大，在数据量较大时会消耗大量内存，而利用JTS封装成对象后，所有的属性被组织成一个类对象，在经过Avro序列化后会大量降低数组大小，起到压缩的作用。

3.3.2 批量导入

在向HBase 中导入数据的情景下，通常会选择使用标准的客户端 API 对 HBase 进行直接的插入操作，或者在MapReduce作业中使用 TableOutputFormat 作为输出。但该方式在大量数据写入时效率比较低下，并对 HBase 节点稳定性造成影响（ RegionServer 无响应）。实际上，在面对需要导入大量数据的情况下，借助HBase的Bulk Load特性可以更加便捷、快速地向HBase数据库中导入数据。HBase的数据实际上是以特定格式存储在 HDFS 上的，因而 Bulk Load 就是先将数据按照HBase的内部数据格式生成持久化的 HFile 文件，然后复制到合适的位置并通知 RegionServer ，即完成巨量数据的入库。在生成 HFile 时无需占用 Region 资源，降低了 HBase 节点的写入压力，在大量数据写入时能极大地提高写入效率。采用bulk load的方式加载数据将节约大量的CPU和网络资源。使用 Bulk Load 特性将数据导入 HBase 通常需要分为三个阶段：

（1）从数据源中提取数据

通常需要导入的外部数据都是存储在其它的关系型数据库或一些文本文件中，我们需要将数据提取出来并放置于 HDFS 中。借助 Sqoop 这一工具可以解决大多数关系型数据库向 HDFS 迁移数据的问题。

（2）通过 MapReduce 任务生成 HFile

在进行数据导入时，需要对数据进行预处理，如过滤无效数据、数据格式转换等。通常按照不同的导入要求，需要编写不同的 Mapper；Reducer 由 HBase 负责处理。为了按照 HBase 内部存储格式生成数据，一个重要的类是 HFileOutputFormat2。为了更有效地导入数据， 每一个输出的 HFile 要恰好适应一个 Region。为了确保这一点，需要使用 TotalOrderPartitioner 类将 map 的输出切分为 key 互不相交的部分。HFileOutputFormat2 类中的 configureIncrementalLoad() 方法会依据当前表中的 Region 边界自动设置 TotalOrderPartitioner。

（3）完成数据导入

一旦数据准备好，就可以使用 completebulkload 工具将生成的 HFile 导入HBase 集群中。completebulkload 是一个命令行工具，对生成的 HFile 文件迭代进行处理，对每一个 HFile， 确定所属的 region， 然后联系对应的 RegionServer， 将数据移动至相应的存储路径。

如果在准备数据过程中，或者在使用 completebulkload 导入数据过程中， region 的边界发生了改变（split），completebulkload 工具会按照新的边界自动切分数据文件。这个过程可能会对性能造成影响。

第4章时空矢量对象的查询算法设计

第3章已介绍时空矢量对象的索引设计及存储，本章将在其基础上介绍时空矢量对象的范围查询，邻近查询引擎的实现方法及其相关研究。范围查询，k-NN查询是空间数据分析的重要手段。在串接时空编码下，空间范围查询算法是时空范围查询算法，k-NN算法的基础。所以本章介绍时空范围查询算法，k-NN算法之前将介绍空间填充曲线下的空间范围查询转换算法。

4.1 空间范围查询转换算法

将查询空间范围转化为一系列一维编码Range是空间查询方案的重要算法。以Z曲线为例，本节将分别介绍矩形，以及不规则多边形转化为一维编码Range的算法设计。

图4-1 空间查询转化Range示意图

对于矩形范围查询，其基本思想可以由图4-1表示。

（1）如图4-1（a），蓝色的矩形为查询范围，红色分界线为全局范围的第

一次划分。且在第一次划分中，蓝色矩形分别与红色分界线的四个节点Range相交，且被划分成四份，且记查询矩形于左下节点Range相交的部分为Q。

（2）对第一次划分生成的四个节点Range分别进行判断。以Q为例，在

第二次划分中（黄线所示），四个节点的Range中，右上节点的Range被Q完全覆盖，则右上节点无需再往下一层级划分。而其他三个节点Range仍与Q相交，则需进一步往下一层级划分，直至如图所示的最大划分次3。

根据此原理，蓝色查询矩形可以被划分为多个大小不同的Range段，如图一（b）中的3-3、5-5、7-7、9-9、11-11、12-15、18-18、24-24、26-26、33-33、35-39、48-48、50-50。若最大划分次数过大，结果集生成的查询Range数将会过多，则会严重影响范围查询的效率。而为了减少Range的个数，本文采取的主要思路是对SpatialRangeSet，Range进行排序，然后进一步合并。如图一（c）示意，黄色填充区域中的值虽然不在蓝色查询矩形中，但通过加入这些结果到Range中，可以将离散的Range合并起来，形成更长的Range。经过合并后，其中可行的一种合并结果可以为3-15、18-18、24-26、33-39、48-50，最终的Range个数为5。起到了减少查询次数的效果，而通过合并生成的Range进行HBase查询，得到同时包含真实值，假真值的结果。所以为了剔除这些假真值，需要通过近一步的精查询操作进行过滤。

表4-1 自顶向下划分空间算法伪代码

Algorithm 1： 自顶向下划分空间范围递归算法

Input：nodeQbbox (minLat,minLon,maxLat,maxLon) curRecursivemaxRecursiveSpatialRangeSet(∅)Output：SpatialRangeSet if curRecursive == maxRecursive:node.selected = Truereturnend ifif curRecursive < maxRecursive:for each child in node.children do:if Qbbox contains child:child.selected = Trueelse if Qbbox intersects child：redo Algorithm1(child, Qbbox, curRecursive+1，maxRecursive)end forend if if node.children are all selected:node.selected = true;else for each child in node.children do:if child is selected:SpatialRangeSet.add(child)end ifreturn SpatialRangeSet

该算法自顶向下划分的伪代码如表4-1所示，首先通过Z曲线编码得到，Qbbox的左下角，及右上角编码的Z曲线编码值，并对Z1，Z2的二进制编码计算其最长公共前缀，得到node节点，node节点所在的空间范围能够完整包含空间查询范围Qbbox。以node，空间范围Qbbox，最大递归次数maxRecursive作为输入变量，执行算法1，其步骤如下所示。

（1）若当前递归次数已经到达最大递归次数maxRecursive，则设定node为

选中状态，并停止递归。

（2）若当前递归次数小于最大递归次数maxRecursive，则遍历其四个孩子节点，若孩子节点所在空间范围包含于查询范围Qbbox，则置孩子节点为选中状态，否则若孩子节点与查询范围Qbbox相交，则以孩子child作为参数，当前递归次数+1，输入继续执行该算法。

（3）若该节点的所有孩子都为选中状态，则设置该节点node为选中状态，并将节点node加入结果集。否则单独加入其被选中的孩子节点。

（4）最终返回SpatialRangeSet结果集。

对于多边形查询，本文采用的思路是，仍以多边形的最小包围矩形所在的node节点作为算法输入，但从自顶向下进行划分判断时，不以node及其孩子节点与最小包围矩形判断是否覆盖或与其相交，而此处多边形的真实形状进行判断。

如图4-2，图（a）中最大递归次数参数maxRecursive为5，图（6）中递归参数为6，由图可知，最大递归次数越大，近似的更加精确，所生成的Range更多。

图4-2 多边形范围查询示意图

4.2 时空范围查询算法

时空范围查询可以定义为在给定的时间范和空间区域（通常矩形，也可为不规则多边形）的条件下，从数据库中取出所有满足查询条件的时空矢量对象。对于点矢量对象，本章主要介绍其分别在时空串接编码，时空一体编码下的时空范围查询算法。而对于线面对象，由于每个线面都需要引入空间上层级的概念，此处只使用时空串接编码。

4.2.1 点对象范围查询

将查询空间范围，矩形或者多边形范围按照4.1介绍的算法转换为一系列对应的SpatialRanges后，则可以按照时空编码的串接编码设计模式将查询的时间范围和转化后的SpatialRanges组合成时空编码，其若是TS，TST串接，则拼接时将T放在前，若是ST，则将S放在前。以TS串接编码为例，时空范围查询主要包括以下几个步骤，范围查询算法流程图如图4-3所示：

图4-3 范围查询示意图

（1）首先利用getSpatialRanges(Qbbox) 函数按照刚4.1介绍的算法将空间范围Qbbox转换为对应的一系列的编码段，用SpatialRanges表示。其中每一个Range段的结构为{ rangeStart，rangeEnd }，其中rangeStart，rangeEnd表示该段Range的

起始、终止空间填充曲线编码值。

（2）connectHTable(tablename)表示连接列数据库中名为tablename的表；用于从HBase中查询结果。

（3）将查询时间范围Qtime按照时间粒度timeCoarseBin离散化，得到一个递增的粗粒度时间Bin序列。

（4）实例为T-S编码方案，则遍历每个时间结点Bin，在每个时间结点下再拼接SpatialRanges，通过该方式得到多个时空查询片段;若为S-T编码，则在串接过程中空间编码在前；对于T-S-T编码，需在T-S的基础上将细粒度时间信息放入编码末尾，使细粒度时间直接存储于时间编码当中，并将串接后的时空编码Range转化为HBase Scan接口的查询条件，从HBase数据库中得到初步的粗查结果。

（5）对于粗查结果，再根据查询空间范围Qbbox以及查询时间范围Qtime进行精过滤，最终得到精确结果。

其伪代码如表4-3所示。算法中使用到的部分函数和变量解释如下：

Qbbox：设置空间范围，由左下角和右上角的坐标决定；Qtime：设置时间范围，由起始时间和终止时间来决定；timeCoarseBin：表示时空单元中时间维的划分粒度。

表4-3 点对象时空范围查询算法描述

Algorithm 2： 点对象时空范围算法

Input：Qtime(Tstart,Tend) Qbbox (minLat,minLon,maxLat,maxLon) Output： ResultSet SpatialRanges = getSpatialRanges(Qbbox); table=connectHTable(tableName); QtimeBins = getTimeBins(Qtime,timeCoarseBin); for each timeBin in QtimeBins do:for each range in SpatialRanges do:strartRowkey = Tstart + range.rangeStart;endRowkey = Tend + range.rangeEnd;rowkeyRange = (startrowkey , endrowkey);Scan = scan(rowkeyRange)ResultScanner=table.get Scanner(Scan);for each Result in ResultScanner do:if Result in Qrange and Result in Qtime:Resultset.add(Result);endifend forend forend forreturn ResultSet

4.1.2 线面对象范围查询

不同于点对象，每个线面对象时空编码所在的层级可能不唯一，其编码方式需要引入层级的概念。层级概念与空间编码相关度紧密，则此处不宜使用时空一体编码，跟适合时空串接编码方式。对于时空串接编码，线面对象的时空范围查询主要包括以下几个步骤。

（1）对于空间范围Qbbox，可以同样根据其左下角，右上角的Z曲线值，

通过获取其Z曲线值的公共前缀，可以得到能够包含该范围的合适的空间单元，该空间单元所对应层级L，并将该层级作为起始层级，根据之前所述可得，可以得到1,个，2个或至多4个象限Range，称之为QudrantRanges。每一个QudrantRange结构为{ rangeStart，rangeEnd }，其内部长度为4的倍数，其中rangeStart，rangeEnd表示该段QudrantRange的起始、终止编码值。

（2）connectHTable(tablename)表示连接列数据库中名为tablename的表；用于从HBase中查询结果。

（3）将查询时间Qtime按照时间粒度timeCoarseBin离散化，得到一个时间结点序列。

（4）从层级L开始，逐层递增，直至遍历到规定的最大层级，不断往下搜索各个层级的线面对象，且在搜索每一层的过程中，将上一层编码的QudrantRangse对应至该层级的编码，称为NewQudrantRanges，遍历每个时间Bin，在每个时间Bin下拼接NewQudrantRanges，得到多个时空查询片段，并将其转化为HBase scan接口的查询条件，最终从HBase数据库中得到粗查结果，并使用协处理器行键过滤过滤掉不属于时空范围的时空数据。

时空范围查询的伪代码如表4-4所示。算法中使用到的部分函数和变量解释如下：Qbbox：设置空间范围，由左下角和右上角的坐标决定；Qtime：设置时间范围，由起始时间和终止时间来决定；timeCoarseBin：表示时空单元中时间维的划分粒度。

表4-4 时空范围查询算法描述

Algorithm 4： 线面对象时空范围算法

Input：Qtime(Tstart,Tend) Qbbox (minLat,minLon,maxLat,maxLon) maxLevelOutput： ResultSet QudrantRanges，startLevel = getSpatialRanges(Qbbox); table=connectHTable(tableName); QtimeBins = getTimeBins(Qtime,timeCoarseBin);l = startLevel;for l < maxLevel do :for each timeBin in QtimeBins do:for each range in SpatialRanges do:strartRowkey = Tstart + range.rangeStart;endRowkey = Tend + range.rangeEnd;rowkeyRange = (startrowkey , endrowkey);Scan = scan(rowkeyRange)ResultScanner=table.get Scanner(Scan);for each Result in ResultScanner do:if Result in Qrange and Result in Qtime:Resultset.add(Result);endifend forend forl = l + 1end forreturn ResultSet

4.1.3 协处理器精过滤算法

在2.1.4节中，我们了解到HBase的协处理器机制可以允许用户自定义功能加载到数据库服务器端，从而将复杂的计算操作在服务器端并行完成，减少了原始数据返回客户端后的串行处理时间。因此一个优化策略是利用Endpoint协处理器来执行精过滤的操作，即在服务器端进行精过滤，并将精查结果返回给客户端进一步处理。

利用HBase的Endpoint协处理器，用户可以将粗查结果的筛选代码部署到 HBase服务器端，HBase将利用底层的多个数据节点并发执行精过滤的操作。即在每个Region范围内执行精过滤的代码，将每个Region的符合查询条件的结果在各个Region Server进行统计并记录，最后将各Region Server的结果进行汇总并返回给客户端。这样整体查询的执行效率就会提高很多。

在HBase服务端利用协处理器进行过滤时，而由于粗查结果中取到的结果都是字节流，需对数据进行反序列化得到数据的经纬度，从而根据查询时间范围，空间范围进行精确过滤，而大量的数据反序列化是一个非常耗时的步骤。为了尽可能减少反序列化的时间，本文先通过行键解码得到粗略的时空立方体，而存储于该行键中的所有列的矢量对象数据的时空范围应都包含于该时空立方体范围内部，所以可先通过时空立方体先进行一步过滤，会有如下三种情况。1）若时空立方体与时空查询范围完全相离，则直接舍弃该数据。2）若时空立方体完全包含于时空查询范围，则可以无需反序列化，直接将结果加入结果集。3）而对于与其相交的时空立方体则需通过对数据反序列化，得到真实的经纬度进行最终过滤。

通过协处理器的行键过滤方法，其一可以使粗查结果直接在HBase服务端进行过滤，其二可以大大减少反序列化的次数

，整体提高查询效率。

4.3 时空k-NN查询算法

4.3.1 基本策略

时空最邻近查询可以描述为给定坐标和参数k，返回查询时间范围内的k个对象。其设计思路如下：首先根据k值估算出一个以给定坐标为中心的网格的空间范围，然后在执行时空范围查询。如果返回结果超过k，则计算与给定坐标距离最近的k个对象，得到查询结果；如果返回结果不足k，则进一步扩大其邻域进行迭代查询，直到查询到的数据量达到为止。为了更好地描述算法，本文首先定义三种距离。如图三对于空间点p(x1，x2)，q(y1,y2)，d1距离为两点之间的欧式距离，d2距离定义为点到网格cell的最近距离。

图4-4 距离示意图

表4-4 k-NN查询算法描述

Algorithm 2： k-NN查询算法

Input：PkQtime(Tstart,Tend) //查询时间范围PriorityQ//优先队列VisitedCells //访问标识ResultSet//表示查询的时空数据结果列表。Output： ResultSet //查询结果集cell = coorToCell(P); //获取查询点P所在的cellltable = connectHTable(tableName); //连接数据库PriorityQ.push(cell) //将P所在cell插入队列尾部while PriorityQ is not empty：curCell = Priority.popFront()VisitedCell.add(curCell)neighbors = getNeighbors(curCell)UpdatePriorityQ(VisitedCells, neighbors, PriorityQ)SortByd2(PriorityQ)rowkeyRange = cellToRange(curCell);Scan = scan(rowkeyRange)ResultScanner=table.get Scanner(Scan);for each Result in ResultScanner do:Resultset.add(Result); //将计算结果添加至结果集end forif type is Point do:SortByd1 (ResultSet)else:SortByd3 (ResultSet)end ififResultset.size >= k :overlappedCells = getKthCircleCellsif VisitedCells.contain(overlappedCells):return Resultset(0,K)else:for each cell in overlappedCells and not in VisitedCells do:PriorityQ.push(cell)end forSortByd2(PriorityQ)end ifend if end whilereturn ResultSet

近邻查询的伪代码如表4-4所示。其中输入P为查询坐标点，k为邻近个数，Qtime为查询时间范围，PriorityQ用于存储待访问的cell，VisitedCells用于标记已经访问过的cell；输出ResultSet用于存储查询结果。近邻查询主要包括以下4个步骤，其流程图如图四所示。

（1）首先将查询点P所在的cell压入优先队列PriorityQ.并将cell添加至Visitedcells

（2）若队列不为空，且搜索范围未超过最大查询范围，则弹出队首cell，并将其周围八邻域cell，从未访问过的cell即不存在于Visitedcells的cell，对于按照d2距离排序后压入队尾。

（3）将cell所在的空间范围转化为空间查询条件，请求HBase数据库并返回结果,将符合时间查询范围Qtime的结果，存入Resultset，按照d1顺序排序。

图4-5 k-NN查询示意图

（4）若Resultset.size < k 则返回步骤2）；否则找到距离点P第k远的点，

计算两点之间的距离D，以点P为中心，距离D为半径做圆Circle，并Circle相交的cell之前都已经访问过，则返回Resultset，否则将仍为访问过的cell压入队尾，并对队列按照d2距离排序，并返回步骤2）。步骤4）原理如图五所示，其展示了两种k-NN查询的案例，其中k=3。在图4-5（a）中，Circle包含了{p1, p2, p3},且只在一个阴影cell内（定义阴影cell为已经访问过的cell），则可确定{p1, p2, p3}为最终结果。在图4-5（b）中，Circle包含了其他的未访问过的cell，且p6，p8距离查询点P较p3更近。所以需要将未访问过的cell加入队列，继续搜索。

4.3.2 自适应网格优化算法

当k-NN查询的中心坐标处于数据密集的区域，若初始网格的边长较小，在k-NN查询不断往外扩搜索的同时，可以比较高效率的访问到其空间相近的矢量对象。而若中心坐标处于数据稀疏区域，若同样设定较小的初始网格，在k-NN查询扩散搜索的同时，将会有大量网格返回的是空值，严重影响到k-NN查询算法的效率。而本文提出一种考虑数据分布的自适应网格优化算法，使其能够根据数据分布设定不同的网格大小，在整体上提高k-NN查询的算法的稳定性。其主要思想如下：

1）设定初始网格，一般为编码的最大划分所对应的网格大小。并找到中心

坐标对应的网格，以及其八邻域的网格。

2）访问HBase数据库，利用协处理器，快速得到中心网格及其周围八邻域

的数据总计数count，且只返回计数，不返回任何具体数据。

3）通过count与用户参数k比较，若count远远小于k，则可认为对于该k

来说，中心坐标所处的区域属于数据稀疏区域，需要进一步进行扩大网格边长进行搜索，将网格大小扩大一倍，并重新执行步骤1）；否则则认为数据不处于数据

稀疏区域。可以直接使用当前的网格大小。

6. 第5章实验性能评价及结果分析

总字数：2787

相似文献列表

原文内容

第5章实验性能评价及结果分析

5.1 实验环境

本章节主要针对时空编码方式，时空范围查询以及，k-NN查询的算法进行验证。实验采用的集群节点配置如表5-1，集群测试的基本环境共有3个节点（其中一个主节点Master，两个Slave节点），每个节点的cpu配置是2路4 Core i7-7700 CPU@3.60GHz ，Hadoop版本为2.7.5，HBase版本为1.4.9 ， Zookeeper版本为3.4.5。

表 51 集群节点配置情况

属性配置信息

CPU 2路 4core Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz

Memory 64G

Network BandWidth 1Gbps

OS CentOS7 64bit

JVM Version JVM 1.8.0

Hadoop Version Hadoop 2.7.5

HBase Version HBase 1.4.9

Zookeeper Version Zookeeper 3.4.5

5.2 查询实验

5.2.1 时空范围查询实验

本次实验采用纽约曼哈顿区的出租车数据，该数据集记录了2015年1月以及2016年1-3月共4个月的出租车信息，共计数据4700万条记录。时空编码中，T为每条记录的上车的点的时间信息，时间粒度为天。S为上车点的经纬度信息，空间编码中精度precision为40位，即经纬度各自20个bit位数。

根据4.2.1介绍的时空范围查询算法，最大迭代次数maxRecursive是其重要的参数，maxRecursive越小，则用格网近似表达的多边形越粗略，会产生大量假真值，但生成的Range数较少，查询HBase的Scan操作较少；maxRecursive越大，则用格网近似表达的多边形越精细，假真值变少，但生成的Range数变多，查询HBase的Scan操作也会越多。本实验主要根据随机生成的不规则多边形，对最大递归次数maxRecursive进行控制变量，并比较实验结果，权衡Scan操作次数与假真值的数量，尽可能找到查询效率最佳的递归次数。

如图5-1所示，其为TS编码下，利用自顶向下划分算法得到的范围查询效率图。其中查询时间范围为2016年1月1日-2016年3月30日，查询空间范围为(-73.90,40.75)到（-73.85，40.80）。其中横坐标为最大递归次数，纵坐标为各个递归次数下，多次查询时间取平均的结果。由图可见，在TS编码下，自顶向下划分算法的效率随递归次数的增长大致呈U字形走向，当最大递归次数为6-8次时，查询效率最大。当最大递归次数小于6时，由于粗查询得到的假真值过多，导致精过滤压力过大，降低查询效率；当最大递归次数大于8时，由于划分过细，得到的Range数量过多，导致查询次数增大，使得查询效率降低。综上所述，对于TS编码而言，当最大递归次数为6-8次时，能较好地平衡查询次数和减少假真值，查询效率最高。

图5-1 TS编码自顶向下划分效率对比图

类似，如图5-2所示，为同样的查询时间范围下，查询空间范围下，ST编码自顶向下划分算法的效率图。自顶向下划分算法的查询时间随递归次数的增长大致呈下降趋势，且当最大递归次数大于9后趋于平衡。结合理论可以看出，当最大递归次数小于9时递归前9时，查询效率陡降的原因是由于ST编码的假真值极大，减少假真值对效率的提升远大于增加查询次数对效率的降低，而随着递归次数不断增大，假真值的减少量逐渐减少，而划分生成的Range数量不断增加，两者对效率的影响趋于平衡，使得查询效率趋于稳定。

图5-2 ST编码自顶向下划分效率对比图

5.2.2 编码对比实验

图5-3不同索引方案查询效率对比图

在第3章中设计了多种编码串接方案，在不同查询情景下，各种索引方案有着不同的表现。图5-3展示了其效率对比图，其中，横轴每一个查询情景均进行了多次实验并取平均时间，且各实验均在有协处理器的条件下进行。由图所示，TS编码的查询效率在大空间范围下的查询效率远远优于ST编码，且在大时间范围下的查询效率和ST编码持平，出现这种结果的原因是，HBase的行键是根据字典排序的，因此行键中排在前面的具有更高的索引效果，例如TS编码中排在前面的T具有更优的索引效果，因此在时间点或小时间范围下的查询时，能立即定位到指定位置。同理，ST编码中，S具有更优的索引效果，因此在面向大空间范围的查询时，其查询效率较低。

5.2.2 时空k-NN查询实验

本实验同样采用纽约曼哈顿区的出租车数据，该数据集记录了2015年1月以及2016年1-3月共4个月的出租车信息，共计数据4700万条记录。且采用TS时空编码，其中T为每条记录的上车的点的时间信息，时间粒度为天。S为上车点的经纬度信息，空间编码中精度precision为40位，即经纬度各自20个bit位数。

k-NN网格搜索默认使用空间编码的精度40，网格大对应的长度为宽38m，长19m。如图5-2所示，该图为4个月数据上车点的热力度，其中序号为1，2，3点处于数据密集区域，序号4，5，6点处于数据稀疏区域。

图5-4 查询点数据分布图

表5-2 查询点详细坐标.

数据分布情况序号经度纬度

数据密集 1 -73.985000 40.744500

2 -73.950459 40.784039

3 -74.001578 40.753176

数据稀疏 4 -73.976279 40.676324
5 -73.957290 40.679517
6 -73.955011 40.703358

各点详细坐标如表5-2所示，本文k-NN实验以序号1-6坐标点为查询点，查询时间范围都固定为2015年1月1日至2015年1月11日。实验结果如表5-2所示，纵向比较来看，在k一定的情况下，数据分布越稀疏，其k-NN查询所消耗的时间越多。横向比较来看，随着k的不断增加，k-NN查询所消耗的时间同样不断增加，且在数据密集区域k-NN查询时间更为稳定，没有明显的线性增长，而在数据稀疏区域k-NN查询时间也会呈明显的线性增长。

数据分布序号查询时间（ms）
k=100 k=200 k=500 k=1000 k=5000
数据密集 1 501 524 502 675 1204
2 610 795 1086 1683 2021
3 1049 1185 1379 1732 3412
数据稀疏 4 1863 2985 4004 4507 15238
5 1485 2233 4788 11374 50393
6 5884 6158 7248 11289 34084

7. 第6章总结与展望

总字数：2393

相似文献列表

去除本人已发表文献复制比：21.6%(518)　文字复制比：21.6%(518)　疑似剽窃观点：(0)

1	矢量大数据管理关键技术研究 姚晓闯(导师：郧文聚;朱德海) -《中国农业大学博士论文》- 2017-05-01	9.2%（220） 是否引证：否
2	201321060644陈俊欣 -《学术论文联合比对库》- 2016-03-21	9.0%（216） 是否引证：否
3	基于Hadoop的空间矢量数据的分布式存储与查询研究 陈俊欣(导师：张凤荔) -《电子科技大学硕士论文》- 2016-03-18	8.7%（207） 是否引证：否
4	姚晓闯_B1311686_矢量大数据管理关键技术研究 姚晓闯 -《学术论文联合比对库》- 2017-06-01	7.9%（189） 是否引证：否
5	面向异构资源集成的虚拟实验平台研究 陈天赐(导师：王建新) -《中南大学硕士论文》- 2011-05-01	2.5%（61） 是否引证：否
6	基于网格的数据流聚类算法研究 张丽(导师：姜保庆) -《河南大学硕士论文》- 2011-05-01	1.8%（43） 是否引证：否
7	SiC陶瓷/UHMWPE复合装甲弹道性能研究 张友敏(导师：胡德安) -《湖南大学硕士论文》- 2018-04-20	1.4%（34） 是否引证：否

原文内容

第6章总结与展望

6.1 论文总结

本文在分布式数据库HBase的基础上，利用NoSQL数据库的优势，提出一套完整的存储组织、时空索引、查询检索的方案，使该系统能够在大数据场景下存储海量的时空矢量数据对象，且同时支持低延迟的时空查询、k-NN查询。

本文的主要工作总结如下：

1）首先本文在时空立方体的基础上，设计了时空串接编码，其支持以空间编码在前粗粒度时间编码在后的ST编码，粗粒度时间在前空间编码在后的TS编码，以及粗粒度时间在前，细粒度时间在两边，空间编码夹中间的TST编码。

2）其次根据分布式数据库HBase的特性，对于时空矢量对象数据集，本文分别构建了编码表，数据表，元数据表。其中元数据主要记录数据集的元信息，如包含数据的编码方案，矢量类型，空间参考等信息；编码表用于时空范围查询，k-NN查询，其主要是为了完成时空矢量对象与时空立方体的映射关系，且编码表中只存储矢量对象的空间位置信息，不支持属性查询；数据表是为了时空矢量对象的属性查询。

3）对于时空矢量点对象的范围查询，本文分别采用不同串接方式的时空串接编码，尝试找出各个编码方案的适用场景。对于时空线面对象，再引入层级的概念后，同样能够支持线面对象的范围查询。

4）对于时空矢量对象的k-NN查询，本文首先获取查询点所在的网格，并不断从八邻域扩散，直至找到符合其时间范围内的k个结果。提出一种顾及数据分布的可自适应控制网格大小的k-NN查询算法。其原理是通过一个预设的网格大小，向HBase数据库服务器发出请求，得到周围八邻域的数据总数，而不获取数据。并通过总数与k-NN查询的k比较，决定是否进一步扩大网格大小。

本文主要有以下两点创新：

1）通过协处理器行键过滤的服务端过滤算法：使用空间填充曲线作为空间索引技术后，其查询过程会将查询条件转化为多个一维的Range段，且合并Range后，会带来假真值，需做精过滤处理。而在客户端做假真值过滤需要提高网络传输量，影响查询效率。而通过引入HBase的协处理器机制，可以在通过Scan操作得到粗查询结果之后，通过服务端协处理器，先对行键解码得到粗略时空立方体，行键所对应的数据应都在该时空立方体内部，所以可以先通过时空立方体进行一步过滤，否则需

对数据进行反序列化得到准确经纬度再进行过滤，而反序列化相比数据库扫描时间是一个非常耗时的步骤。若时空立方体与时空查询范围完全相离，则直接舍弃该数据；若时空立方体完全包含于，则直接将结果加入结果集；而对于与其相交的时空立方体则需通过对数据反序列化，得到精确的经纬度进行最终过滤。

2）提出一种顾及数据分布的可自适应控制网格大小的k-NN查询算法在k-NN查询，不断通过八邻域向外搜索扩充的过程中，当查询点在数据密集区域时，较小的网格大小可能可以很快搜索到查询结果，但对于数据稀疏的区域，若同样使用较小网格则可能需要耗费非常长的时间才能搜索到近邻查询结果。提出一种顾及数据分布的可自适应控制网格大小的k-NN查询算法。其原理是通过一个预设的网格大小，向HBase服务器发出请求，得到周围八邻域的数据总数，而不获取数据。并通过总数与k-NN查询的k比较，决定是否进一步扩大网格大小。

6.2 展望

本文针对矢量大数据管理方面进行了较为全面的研巧工作，初步解决了大规模矢量数据的存储、索引、处理从及可视化等实际应用问题，并取得了一定的研巧成果，但由于作者的研究时间和精力有限，本文研究工作仍然存在着不足和需要改进的内容。目前，我认为还可以在以下几个方面继续开展相关研究，以推动大数据时代矢量数据管理基础理论和关键技术的快速发展。

1）对于线面对象，本文采取冗余备份的思想，但由于线对象一般是狭长的形状，用至多4份的空间格网去覆盖线对象，仍会造成很多的空间浪费，需要进一步研究线对象的存储机制。提高查询效率。

2）本文只完成了两类查询，即范围查询即k-NN查询。更多空间关系的查询包括空间的连接查询，以及复杂的拓扑关系判断，还需要进一步探索。

3）整个HBase在海量空间数据库的管理方面并没有传统关系数据库那么成熟，包括对于数据分片的管理，对于操作的实时监控，以及用户的操作的权限管理，数据的复制与备份等等。探究HBase的空间数据处理引擎，不断丰富和提升其管理功能，为用户提供准确、快速的空间数据支持也是今后研究的一个方向。

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] 李国杰, 程学旗. 大数据研究:未来科技及经济社会发展的重大战略领域——大数据的研究现状与科学思考[J]. 中国科学院院刊, 2012, 27(6):647-657.

[3] 李清泉, 李德仁. 大数据GIS[D]. , 2014.

[4] Han J, Haihong E, Le G, et al. Survey on NoSQL database[C]//2011 6th international conference on pervasive computing and applications. IEEE, 2011: 363-366.

[5] Cattell R. Scalable SQL and NoSQL data stores[J]. Acm Sigmod Record, 2011, 39(4): 12-27.

[6] Moniruzzaman A B M, Hossain S A. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison[J]. International Journal of Database Theory & Application, 2013, 6.

[7] George L. HBase: the definitive guide[J]. Andre, 2011, 12(1): 1-4.

[8] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]//MSST. 2010, 10: 1-10.

[9] White T. Hadoop: The definitive guide[M]. " O'Reilly Media, Inc.", 2012.

[10] Tan H, Luo W, Ni L M. Clost: a hadoop-based storage system for big spatio-temporal data analytics[C]//Proceedings of the 21st ACM international conference on Information and knowledge management. ACM, 2012: 2139-2143.

[11] Aji A, Wang F, Vo H, et al. Hadoop gis: a high performance spatial data warehousing system over mapreduce[J]. Proceedings of the VLDB Endowment, 2013, 6(11): 1009-1020.

[12] Eldawy A, Mokbel M F. The ecosystem of SpatialHadoop[J]. SIGSPATIAL Special, 2015, 6(3): 3-10.

[13] Eldawy A, Mokbel M F. Spatialhadoop: A mapreduce framework for spatial data[C]//2015 IEEE 31st international conference on Data Engineering. IEEE, 2015: 1352-1363.

[14] Eldawy A, Alarabi L, Mokbel M F. Spatial partitioning techniques in SpatialHadoop[J]. Proceedings of the VLDB Endowment, 2015, 8(12): 1602-1605.

[15] Eldawy A, Mokbel M F. The era of big spatial data: A survey[J]. Foundations and Trends® in Databases, 2016, 6(3-4): 163-273.

[16] Alarabi L, Mokbel M F, Musleh M. St-hadoop: A mapreduce framework for spatio-temporal data[J]. GeoInformatica, 2018, 22(4): 785-813.

[17] Samet H. The quadtree and related hierarchical data structures[J]. ACM Computing Surveys (CSUR), 1984, 16(2): 187-260.

[18] Miller F P, Vandome A F, Mcbrewster J. kd-tree[M]. Alpha Press, 2009

[19] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets[J]. HotCloud, 2010, 10(10-10): 95.

[20] Yu J, Wu J, Sarwat M. Geospark: A cluster computing framework for processing large-scale spatial data[C]//Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2015: 70.

[21] Banker K. MongoDB in action[M]. Manning Publications Co., 2011.

[22] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data[J]. ACM Transactions on Computer Systems (TOCS), 2008, 26(2): 4.

[23] Carlson J L. Redis in action[M]. Manning Publications Co., 2013.

[24] 吴飞. 基于 MongoDB 的 LBS 数据管理系统关键技术研究[J]. 测绘通报, 2014, 7: 121-124.

[25] 雷德龙, 郭殿升, 陈崇成, 等. 基于 MongoDB 的矢量空间数据云存储与处理系统[J]. 地球信息科学学报, 2014, 16(4): 507-516.
[26] 张景云. 基于Redis的矢量数据组织研究[D]. 南京师范大学, 2013.
[27] 范建永, 龙明, 熊伟. 基于HBase的矢量空间数据分布式存储研究[D]. , 2012.
[28] Nishimura S, Das S, Agrawal D, et al. MD-HBase: A scalable multi-dimensional data infrastructure for location aware services[C]//2011 IEEE 12th International Conference on Mobile Data Management. IEEE, 2011, 1: 7-16.
[29] Hughes J N, Annex A, Eichelberger C N, et al. Geomesa: a distributed architecture for spatio-temporal fusion[C]//Geospatial Informatics, Fusion, and Motion Video Analytics V. International Society for Optics and Photonics, 2015, 9473: 94730F.
[30] Whitby M A, Fecher R, Bennight C. Geowave: Utilizing distributed key-value stores for multidimensional data[C]//International Symposium on Spatial and Temporal Databases. Springer, Cham, 2017: 105-122.
[31] Hamilton C H, Rau-Chaplin A. Compact Hilbert indices: Space-filling curves for domains with unequal side lengths[J]. Information Processing Letters, 2008, 105(5): 155-163.
[32] Miller F P, Vandome A F, Mcbrewster J. Geohash[M]. Alphascript Publishing, 2010.
[33] Van Le H, Takasu A. G-hbase: A high performance geographical database based on hbase[J]. IEICE TRANSACTIONS on Information and Systems, 2018, 101(4): 1053-1065.
[34] Zhangy C, Zhuy L, Longy J, et al. A hybrid index model for efficient spatio-temporal search in HBase[J]. arXiv preprint arXiv:1805.07599, 2018.
[35] Ma Y, Rao J, Hu W, et al. An efficient index for massive IOT data in cloud environment[C]//Proceedings of the 21st ACM international conference on Information and knowledge management. ACM, 2012: 2129-2133.
[36] Abel D J. A B+-tree structure for large quadtrees[J]. Computer Vision, Graphics, and Image Processing, 1984, 27(1): 19-31.
[37] 李雪梅, 邢俊峰, 刘大伟, 等. 基于HBase 的海量 GIS 数据分布式处理实践[J]. 大数据, 2016, 2(3): 2016032.
[38] 丁琛. 基于 HBase 的空间数据分布式存储和并行查询算法研究[D]. 南京: 南京师范大学, 2014.
[39] Guttman A. R-trees: A dynamic index structure for spatial searching[M]. ACM, 1984.
[40] Beckmann N, Kriegel H P, Schneider R, et al. The R*-tree: an efficient and robust access method for points and rectangles[C]//Acm Sigmod Record. Acm, 1990, 19(2): 322-331.
[41] Theodoridis Y, Vazirgiannis M, Sellis T. Spatio-temporal indexing for large multimedia applications[C]//Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems. IEEE, 1996: 441-448.
[42] Graham R L, Hell P. On the history of the minimum spanning tree problem[J]. Annals of the History of Computing, 1985, 7(1): 43-57.
[43] Kamel I, Faloutsos C. Hilbert R-tree: An improved R-tree using fractals[R]. 1993.
[44] Fu Y C, Hu Z Y, Guo W, et al. QR-tree: a hybrid spatial index structure[C]//Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693). IEEE, 2003, 1: 459-463.
[45] Mokbel M F, Aref W G, Kamel I. Analysis of multi-dimensional space-filling curves[J]. GeoInformatica, 2003, 7(3): 179-209.
[46] Moon B, Jagadish H V, Faloutsos C, et al. Analysis of the clustering properties of the hilbert space-filling curve[J]. IEEE Transactions on knowledge and data engineering, 2001, 13(1): 124-141.
[47] 陆锋, 周成虎. 一种基于Hilbert排列编码的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.
[48] Mokbel M F, Aref W G. Irregularity in multi-dimensional space-filling curves with applications in multimedia databases[C]//Proceedings of the tenth international conference on Information and knowledge management. ACM, 2001: 512-519.
[49] Z曲线网格划分的最近邻查询[J]. 计算机工程与应用, 2013, 49(22):123-126.
[50] 郝忠孝. 时空数据库新理论[M].
[51] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]//MSST. 2010, 10: 1-10.
[52] J.Dean and S.Ghemawat, “Mapreduce: simplified data processing on large clusters,” Communications of the ACM, vol.51, no.1,pp.107–113, 2008.
[53] 张延辉,孟鑫,李立松.HBase企业应用开发实战[M]. 北京:机械工业出版社,2014.
[54] Nick Dimiduk, Amandeep Khurana. HBase实战[M]. 北京:人民邮电出版社,2013.
[55] Lars George. HBase权威指南[M]. 北京:人民邮电出版社,2013.
[56] HBase[EB/OL].http://en.wikipedia.org/wiki/NoSQL.

致谢

在论文完成之际，谨向我给予帮助的老师、同学、朋友表示衷心的感谢。

首先，我要感谢范老师，虽然范老师长期在外出差，但范老师仍然给我提供了无微不至的关怀，对我的学习生活悉心指导。同时我要感谢关老师，在我论文的开题与写作当中给予的帮助与支持。关老师渊博的知识、勤奋的精神以及对学生高度负责任的态度人令我难忘，激励着我上进。

其次，衷心感谢导师吴教授。回顾在硕士三年的学习中，吴教授不仅提供了优越的学习环境，而且其严谨的治学态度和对我的严格要求促进我进步。两年来，我在学习和科研上所取得的每一点成绩、每一点进步无不浸透着导师的心血。在此谨向尊敬的导师致以最诚挚的谢意。

还要感谢在我论文写作过程中给予指导意见的关洪礼同学以及谌诞楠同学。在我论文开题与实验设计、论文写作当中

，2位同门给了我莫大的帮助与意见，正是有了同门的督促与指导、帮助与扶持，才有了本毕业论文的完成。感谢我的室友王振林、杨杰和方忠浩三年的陪伴，在生活上给予了我莫大的帮助，你们的存在，才有温馨的311宿舍。

最后，感谢我的父母亲人，多年来对我的关爱与资助，你们的期待与关心是我不断前进的动力。我要感谢自己，感谢自己本科时期的努力，才有了能够在测绘遥感信息工程国家重点实验室读研究生的资格，拥有了继续深造的机会；感谢自己研究生三年的勤奋努力，自强不息，找到一份满意的工作。

指　标
疑似剽窃观点
1. 目前，我认为还可以在以下几个方面继续开展相关研究，以推动大数据时代矢量数据管理基础理论和关键技术的快速发展。
疑似剽窃文字表述
1. 6.2 展望 本文针对矢量大数据管理方面进行了较为全面的研巧工作，初步解决了大规模矢量数据的存储、索引、处理从及可视化等实际应用问题，并取得了一定的研巧成果，但由于作者的研究时间和精力有限，本文研究工作仍然存在着不足和需要改进的内容。 2. 本文只完成了两类查询，即范围查询即k-NN查询。更多空间关系的查 询包括空间的连接查询，以及复杂的拓扑关系判断，还需要进一步探索。 3）整个HBase在海量空间数据库的管理方面并没有传统关系数据库那么成 熟，包括对于数据分片的管理，对于操作的实时监控，以及用户的操作的权限管理，数据的复制与备份等等。探究HBase的空间数据处理引擎，不断丰富和提升其管理功能，为用户提供准确、快速的空间数据支持也是今后研究的一个方向。 3. 论文完成之际，谨向我给予帮助的老师、同学、朋友表示衷心的感谢。 首先，我要感谢范老师， 4. 年来，我在学习和科研上所取得的每一点成绩、每一点进步无不浸透着导师的心血。在此谨向尊敬的导师致以最诚挚的谢意。 还要感谢

说明：1.总文字复制比：被检测论文总重合字数在总字数中所占的比例

2.去除引用文献复制比：去除系统识别为引用的文献后，计算出来的重合字数在总字数中所占的比例

3.去除本人已发表文献复制比：去除作者本人已发表文献后，计算出来的重合字数在总字数中所占的比例

4.单篇最大文字复制比：被检测文献与所有相似文献比对后，重合字数占总字数的比例最大的那一篇文献的文字复制比

5.指标是由系统根据《学术论文不端行为的界定标准》自动生成的

6.红色文字表示文字复制部分;绿色文字表示引用部分;棕灰色文字表示作者本人已发表文献部分

7.本报告单仅对您所选择比对资源范围内检测结果负责

amlc@cnki.net

http://check.cnki.net/

http://e.weibo.com/u/3194559873/