硕士学位论文

基于 Spatial-Spark 海量网络空间数据分析与应用

Analysis and Application about Internet Spatial Data Based on Spatial-Spark

作　者: 高　峰
导　师: 高井祥　教授
孙久运　副教授

中国矿业大学
二〇一七年六月

学位论文使用授权声明

本人完全了解中国矿业大学有关保留、使用学位论文的规定, 同意本人所撰写的学位论文的使用授权按照学校的管理规定处理:

作为申请学位的条件之一, 学位论文著作权拥有者须授权所在学校拥有学位论文的部分使用权, 即: ① 学校档案馆和图书馆有权保留学位论文的纸质版和电子版, 可以使用影印、缩印或扫描等复制手段保存和汇编学位论文; ② 为教学和科研目的, 学校档案馆和图书馆可以将公开的学位论文作为资料在档案馆、图书馆等场所或在校园网上供校内师生阅读、浏览. 另外, 根据有关法规, 同意中国国家图书馆保存研究生学位论文.

(保密的学位论文在解密后适用本授权书).

作者签名: 导师签名:

年 月 日 年 月 日

中图分类号: P2 学校代码: 10290

UDC: 910.1 密 级: 公 开

中国矿业大学

硕士学位论文

基于 Spatial-Spark 海量网络空间数据分析与应用

Analysis and Application about Internet Spatial Data Based on Spatial-Spark

作 者 高峰 导 师 高井祥, 孙久运

申请学位 工学硕士 培养单位 环境与测绘学院

学科专业 大地测量学与测量工程 研究方向 3S 技术集成及其应用

答辩委员会主席 吴侃 评 阅 人

二〇一七年六月

致谢

矿大七年的求学经历即将结束，纵有千万般不舍，也要走向社会，用自己的在矿大学到的知识回报这个社会。作为矿大新百年的学子，矿大「好学力行、求是创新、艰苦奋斗、自强不息」的校训将成为我人生中的座右铭，在接下来的人生中，矿大精神将永远激励我前进，在这再次感谢矿大对我培养。

首先感谢我的导师高井祥老师和孙久运老师，您们是我研究生生涯中的指明灯，指引我走向学术生涯道路。两位老师孜孜不倦的精神使我受益终身，你们为我提供了优越的科研环境，耐心和用心地指导我们开展学术活动，严谨的科研思维和扎实的工作作风是我的楷模。当我申请去中科院电子所苏州研究院实习，你们坚定的支持是我最大的动力，再次感谢你们。

还要感谢中科院电子所苏州研究院胡岩峰院长，张昱老师、廉海明老师，在苏州实习期间，是你们的指导使我扩展了科研的视野。将我储备的专业知识在实践中加以应用，也让我认识到自己专业领域的不足。在苏州也认识了一些有趣的人，这些为我的人生增加了许多美好的回忆，谢谢你们能提供给我实习的机会。

感谢实验室张明杰、杨文环师兄和许世娇、王慧师姐，是你们在学习和生活上给了我巨大的帮助，你们的帮助使我快速地适应了研究生生活。感谢实验室同门张新耐、赵冰峰、孙梦娇和李光雨，非常有幸和你们一起学习生活，一起成长，非常感谢你们在我外出实习期间帮我处理学校一些事务，和你们在一起的日子是我人生中一段宝贵的财富。

感谢论文中所有参考文献的作者，是你们杰出的工作，让我产生一点点灵感，你们是巨人的肩膀。还要感谢 LaTeX 中文社区的肖立顺师兄，你无偿提供的矿大毕业论文 LaTeX 模板，使我能够使用优美的 LaTeX 完成论文排版工作。

也要感谢远在家乡的家人，是你们坚定不移的支持，是我读研的全部动力，这份理解和付出无法报答，希望你们身体健康，永远幸福。

最后，感谢各位专家和老师，在百忙之中评阅本论文并提出宝贵的意见和建议，我会聆听你们的教导继续努力。

摘要

数字城市是智慧城市重要的组成部分，也同时面临着海量空间数据获取、管理、分析和挖掘等挑战。移动互联网的发展使得网络空间数据呈现爆炸式增长，其中蕴含的信息对智慧城市建设有着重要的参考建议，然而这些数据存在着异质性、不规则性和海量性等特点，使得空间数据查询、空间数据挖掘和空间知识提取愈发难以处理。

传统的空间分析工具面对上述需求往往捉襟见肘，本文对当前流行的并行计算框架 Spark 进行空间扩展，构建 Spatial-Spark 并行空间计算框架。以此为基础，对海量新浪微博 POI 进行同位模式挖掘，对全国新浪微博用户空间位置进行人口网络图分析，本文所作的工作和结论如下：

(1) 对 Spark RDD((Resilient Distributed Datasets) 进行空间维度上扩展，对点、线和面构建了相应的 Spatial RDD，支持海量空间数据读写、空间坐标转换和分区空间数据索引。提供空间拓扑查询、空间 K 邻居查询和空间连接查询三个常用的空间查询模块，通过搭建 Hadoop/Spark 计算集群，验证了 Spatial-Spark 在处理海量空间数据方面的优势。

(2) 使用新浪微博 API 获取全国范围内微博 POI 数据，对其进行同位模式挖掘。首先分析同位模式挖掘算法的关键，使用 Spatial-Spark 对全连接算法进行并行化设计。对上海、武汉和重庆三市二阶模式进行比较，不同城市呈现不同模式；选择距离阈值 $d = 500$m 和空间参与度阈值 0.6，对北京市微博 POI 类别进行同位模式挖掘，结果显示阶数越高越呈现商业聚集模式，其中最高六阶模式为 (KTV，中餐厅，咖啡厅，甜品店，美容美发店，酒吧)。

(3) 根据全国新浪微博用户在 2016 年春节期间的空间位置数据，使用 Spatial-Spark 构建全国城市之间人口流动网络图。首先计算每个城市人口流入量、流出量和流入流出比，发现全国城市在春节期间人口流动呈现多样性；然后采用 PageRank 算法计算城市在人口流动网络图中的权重，发现城市权重与城市 GDP 发展的存在相关性，并根据权重将中心城市划分四个层次；最后对社群挖掘算法进行并行化改进，对人口流动网络图进行社群挖掘，发现城市联系紧密性与省份有关，地理位置对其影响很大，但也存在突破地理空间位置限制的城市。

本文包含图36幅，表21张，参考文献82篇。

关键词: Spatial-Spark, 新浪微博, 同位模式, 网络分析

Abstract

Digit City plays a vital role in smart city, and facing explosing amount data fetch, management, analysis and mining challenges. Mobing internet developments cause the cyber spatial data increaing unbelievablely which contains much informations for building the smart cities. Howerver, the heterogeneous, irregular and countless data leads the spatial data's analysis, mining and knowledge discovering becoming more intractable.

Traditional spatial analysis method cannot meet the above requires, so this thesis builds the parallel spatial computing framwork, called Spatail-Spark, based on popular open-source parallel computing framwork. Using this framework, it mines the spatial co-location pattern in Weibo's POIs and population graph according to the Weibo users's spatial locations. The achievements and conclusions of this thesis are listed as follows:

(1) It expands the Spark RDD in spatial geometries called Spatial RDD, including point, line and polygon. And those RDDs support the spaital data reading and writing, spatial coordinates converting and spatial index building in separate partition. It also provides three spatial query modules: sptaial topology query, spatial K near neighbour query and spatial join query. At last, it designs experiments to figure out the efficiency of Spatial-Spark by various comparsions.

(2) After fetching out Weibo's POIs through Weibo's API, it conducts the spatial co-location pattern mining algorithm in those data. It analysis the vital key of spatial co-location pattern and redesigns the parallel algorithm for efficient performance using Spatial-Spark. It comes out that the there are huge differences among the different cities, taking the Shanghai, Wuhan and Chongqing. It also provides that the patterns shows more commercial characteristics when the pattern's order is bigger on the condition that the spaital distance threshold is 500 meters and the participating index threshold is 0.6. Moreover, the six order pattern is (KTV, Chinese Restaurant, Coffe Bar, Dessert Stall, Barber Shop, Bar Pub).

(3) It constructs the cities floating population graph using Spatial-Spark from the national Weibo's users location in 2016 Chinese New Year holiday. At first, it calculates the popluation amount of flow-in and flow-out, and flow ratio, which displays the diversity of floating population among the national cities. Then it figures out the national cities' weights in floating population network under PageRank algorithm. It finds out the relationship between city's weight and city's development and divides the natianal cities into four levels. Lastly, it applys the parallel community mining algorithm

in this network and uncovers the phenomenon that the cities's popluaiton relationships are almost limited by the provinces and there are also some exceptional cases.

This thesis contains 36 figures, 21 tables and 82 references。

Keywords: Spatial-Spark, Weibo, Co-Location Pattern, Graph Analysis

目录

Contents

图清单

表清单

变量注释表

min_s	最小支持度
min_c	最小置信度
R	空间邻近关系
$row\_instance$	行实例
$table\_instance$	表实例
$PR(c, f_i)$	空间参与率
$PI(c)$	空间参与度
min_prev	最小参与度
δ_i	流入量
ω_i	流出量
ψ_i	流入流出比
Q	模块值
ΔQ	模块增益值

1 绪论

1 Introduction

1.1 研究背景和意义 (Background and Research Significance)

智慧城市是由数字城市、物联网和云计算三大类支撑技术组成，其中数字城市是智慧城市的核心，主要包含天地空一体化空间信息快速获取、海量空间数据管理、空间信息可视化技术、空间数据分析挖掘和网络服务技术[1]。互联网尤其移动互联网的快速发展，使得数字城市面临巨大的挑战：空间数据获取手段不再局限于传统人工测量，基于用户地理位置服务 (Location Based Service, LBS) 功能使得网络中包含了用户的空间数据；每天产生的海量数据对数据存储、索引和查询提出高性能要求；海量空间数据使得空间知识愈发贫乏，急需海量空间数据挖掘相关研究。

相关调查表明，中国大陆是全球移动互联网用户最多的地区，移动互联网设备上各种 APP 记录了每个用户生活的方方面面，比如打车软件记录城市居民出行数据，O2O 软件反映了城市线下商圈服务区域情况。其中以新浪微博为代表的社交网络应用最为值得研究，超过一大半移动互联网用户都拥有新浪微博账号。作为高频使用应用程序，新浪微博已经成为数字城市空间数据不可忽视的重要数据来源。在社交网络中，用户的行为往往是自发进行的，这些数据蕴含了城市居民生活的真实信息，快速精确的对这些数据进行空间书分析和挖掘对智慧城市建设有着重大意义，主要包含以下三个方面：

(1) 对个人而言，对网络空间数据挖掘，可以发现不同的知识，为生活带来便利，如交通路线规划、旅游景点推荐等，构建城市智慧生活。

(2) 对商业公司而言，通过空间数据挖掘发现目标客户使用习惯或群体性活动规律，对商业推广计划提出指导性意见，如广告投放、精准营销等，是城市智慧商业重要组成部分。

(3) 对政府决策部门而言，城市居民的社交网络中的行为是社会现实现象的反映，能够发现社会的经济、文化、交通等众多方面活动规律。为智慧城市发展的各项决策提供建议。

智慧城市提出了海量空间数据高效处理的需求，目前针对海量数据普遍采用分布式并行计算，其核心思想是将海量数据划分为若干个子集，将每个子集分配给不同的线程、进程或者是机器进行并行处理，最后将每个子集计算出来的结果进行合并，得到全局计算结果。Hadoop MapReduce[2] 是业界广泛使用的并行分布式计算框架，通过对数据处理过程的高度抽象，使普通开发人员编写分布式并行计算成为可能。但是现实业务需求的更迭，这些计算框架面对迭代计算

需求显得捉襟见肘，Spark 作为新一代的开源通用并行计算框架[3]，大大扩展了 MapReduce 表现能力，而且通过内存计算独特的运行机制，避免了 IO 操作，使得 Spark 更好适应数据挖掘和机器学习等一些迭代算法[4]。但 Spark 在处理空间数据的时候，其抽象程度仍然较低，不支持空间数据类型表达和空间数据运算，因此高效海量空间数据并行计算框架的需求迫在眉睫，空间分析和空间数据挖掘算法并行化设计也值得进一步研究。

1.2 国内外研究现状 (Research Progress at Home and Abroad)

1.2.1 并行计算框架研究现状

自 2004 年 Google 公司公布分布式并行计算框架思想后[5]，开源分布式并行计算框架 Hadoop 被业界广泛使用，其效仿 GFS(Google File System) 设计思想，提出了 HDFS(Hadoop Distributed File System) 分布式文件系统，同时也提供 MapReduce 编程模型，该计算模型简单，适合大规模数据分析。以 Hadoop 为核心，发展出完整的大数据处理分析技术生态圈，如非关系型数据库 HBase[6]，机器学习库 Mahout[7]，资源管理平台 YARN[8] 等等。整个技术生态圈在开源社区中不断完善，在生产实际中广泛使用。

随着业务场景的不断改变，Hadoop MapReduce 的缺陷越来越被用户所诟病，UC Berkeley AMP 实验室提出 Spark 并行计算框架，并以弹性分布式数据集 (RDD) 为核心提出一套完整的技术栈。与 MapReduce 相比该计算框架的特点在于内存计算，避免了中间数据的 IO 操作，RDD 还提供一系列丰富的操作算子[3]，使之表现能力大大提高。Spark 支持多种编程语言，除了开发语言 Scala，还支持传统的 Java 和脚本语言 Python，也支持统计语言 R。与 Hadoop MapReduce 只能编写 Jar 包提交程序不同，Spark 还提供类似 Shell 的交互式编程方式，通过即时反馈快速搭建算法原型，提高开发效率。

Spark 一经推出，受到了学术界和工业界广泛关注，学术界对 Spark 源码进行改进，使之更加稳定，增加新的功能，截止目前 Spark 版本已经发展到 2.0 版本，在 GitHub 上已超过一千名开发者参与。工业界也在广泛使用 Spark 来处理业务逻辑，有些商业公司也还推出商业并行计算云平台[9,10]，将并行计算资源向整个社会开放。用户根据需求购买相应的虚拟机资源，该资源能够快速完成并行计算运行环境配置，省去了用户搭建计算平台等复杂的工作。

1.2.2 并行计算空间数据分析研究现状

近年来，伴随着并行计算框架的提出，国内外学者做了大量并行计算环境下的空间分析的研究。

马磊在 HDFS 上分级建立 R 树空间索引[11]，加快海量空间数据查询速度；

方金云等人根据空间查询特性和 Spark 分布式内存计算模型[12]，设计 HBase 分布式存储 Spark 分布式内存计算框架的空间区域查询算法；温馨实现一种基于 Shark/Spark 的分布式空间数据分析框架[13]，通过自定义函数和空间函数下推两种实现空间查询方式；李璐明，蒋新华等人使用 Spark 对海量空间数据进行密度聚类[14]，并根据 Spark 并行计算特点提出了并行密度聚类算法 PClusterdp；靳凤营，张丰等对矢量空间数据，使用 Spark 进行空间叠加分析，性能与传统 Oracle+ArcSDE 两种方法进行发现 Spark 在处理海量中有较大的优势[15]。

Abouzeid A. 将分布式技术和关系型数据结合进行一些尝试[16]，使用 MapReduce 管理多个数据库，转换 HadoopDB 中的 SQL 语句，将操作推入到数据库层处理；Witayangkurn A. 和 Horanont T. 等人将 Hadoop-GIS 与 Hive 相结合，实现 MapReduce 处理边界对象[17]；美国环境系统研究所（ESRI）开发 Esri Geometry API for Java 和 Spatial Framework for Hadoop，将 ArcGIS 向 Hadoop 拓展，通过在 Hive 中自定义空间查询函数进行空间数据查询[18]；Nathan K. 在 Hadoop 平台上使用了 MapReduce 并行计算框架对比分析处理，并对比 Hadoop 云计算平台和传统的单节点价格的 PostGIS 空间数据库进行分析处理；Ahmed E. 和 Mohamed F.M. 提出 SpatialHadoop 框架，通过改进 Hadoop 源码，增加了空间数据处理的支持，如空间数据类型、空间操作、空间查询和空间过滤[19]；Jia Y.，JinXuan W. 和 Mohamed S. 等人尝试了在 Spark 上处理空间数据，对比了 Spark 与 SpatialHadoop 空间计算性能[20]。

综合文献资料，可以了解到国内外学者对大数据计算框架在空间数据处理中的应用逐渐增多。从原先的 MapReduce 计算框架向 Spark 计算框架进行迁移；数据存储由传统的关系型数据库向 key-value 非关系型数据库转变；数据格式从常用的 Shapefile 文件向 WKT，JSON 等文本型数据格式转变。但是这些试探性研究尚未能够将空间数据类型和空间数据分析纳入到 Spark 计算体系中，并空间数据挖掘算法向并行化重新设计研究较少。

1.2.3 微博空间数据研究现状

国内外学者很早就注意到社交网络中包含了巨大的研究价值，Andrienko G. 和 Andrienko N. 等人通过研究 Twitter(类似新浪微博) 中的推文分析美国西雅图人们每天生活习惯模式[21]；Linna L. 和 Michael F. 通过结合 Twitter 和 Flickr(图片社交网站) 分析社交网络在空间、时间和经济方面的发展状况[22]；Hollenstein L. 和 Purves R. 通过带有地理标签的 Flickr 图片分析人们在现实世界中活动规律[23]；Xingjian L. 和 Jianghao W. 通过带有地理位置的新浪微博，分析国内和海外用户分布特征[24]；刘大均，胡静等人根据新浪微博旅游版块分析中国旅游空间分布格局和影响因素[25]；徐艳等人通过手动筛选新浪微博用户关系数据分析重庆市

城市网络空间特征分布[26]。

从上述研究现状中可以看出，目前对社交网络中空间信息探索研究越来越得到重视，通过线上数据发现线下真实状况具有较强的现实意义，然而目前基于新浪微博研究存在样本量较小，选择片面性和数据存在人工干预等不足，如何通过类「全样本」新浪微博空间数据分析，从宏观角度发现有价值的信息对现实有着重要的指导意义。

1.3 研究内容和技术路线 (Contents of Research and Technology Route)

1.3.1 研究内容

本文将从数字城市对海量空间数据高效处理需求出发，研究 Spark 并行计算框架的核心内容，对 Spark RDD 进行空间扩展，使之能够支持空间数据类型和空间分析。

以此计算框架为基础，将串行的空间数据挖掘算法并行化改进，以海量新浪微博 POI 和用户位置数据进行空间数据分析和挖掘。主要研究内容如下：

(1) 分析现有的开源分布式计算框架 MapReduce 和 Spark 工作机制，并比较了各自的优缺点；

(2) 分析新浪微博空间数据获取接口，编写新浪微博空间数据的应用程序；

(3) 对 Spark RDD 进行空间扩展，开发 Spatial-Spark 并行计算框架，使该框架支持点、线和面基本空间几何对象和空间索引建立，在此基础上提供常用的并行化空间分析运算接口；

(4) 以 Spatial-Spark 为基础，对新浪微博 POI 数据进行并行化空间模式挖掘；

(5) 使用 Spatial-Spark 对新浪微博用户位置信息构建人口流动网络图，分析城市人口流动量、城市网络权重和设计并行化社群挖掘算法。

1.3.2 技术路线

从 Spark RDD 空间扩展出发，对空间数据的点、线和面分别形成 PointRDD、LineRDD 和 PolygonRDD，并分区建立空间索引，在此基础上提供并行化空间拓扑查询、空间 KNN 查询和空间连接查询等空间分析操作，构建并行空间计算框架 Spatial-Spark。

对全国的新浪微博 POI 数据和新浪微博用户位置数据，使用 Spatial-Spark 对海量的空间数据进行数据分析，主要包括统计分析、同位模式挖掘、权重分析和社群挖掘。本文的技术路线见图1–1。

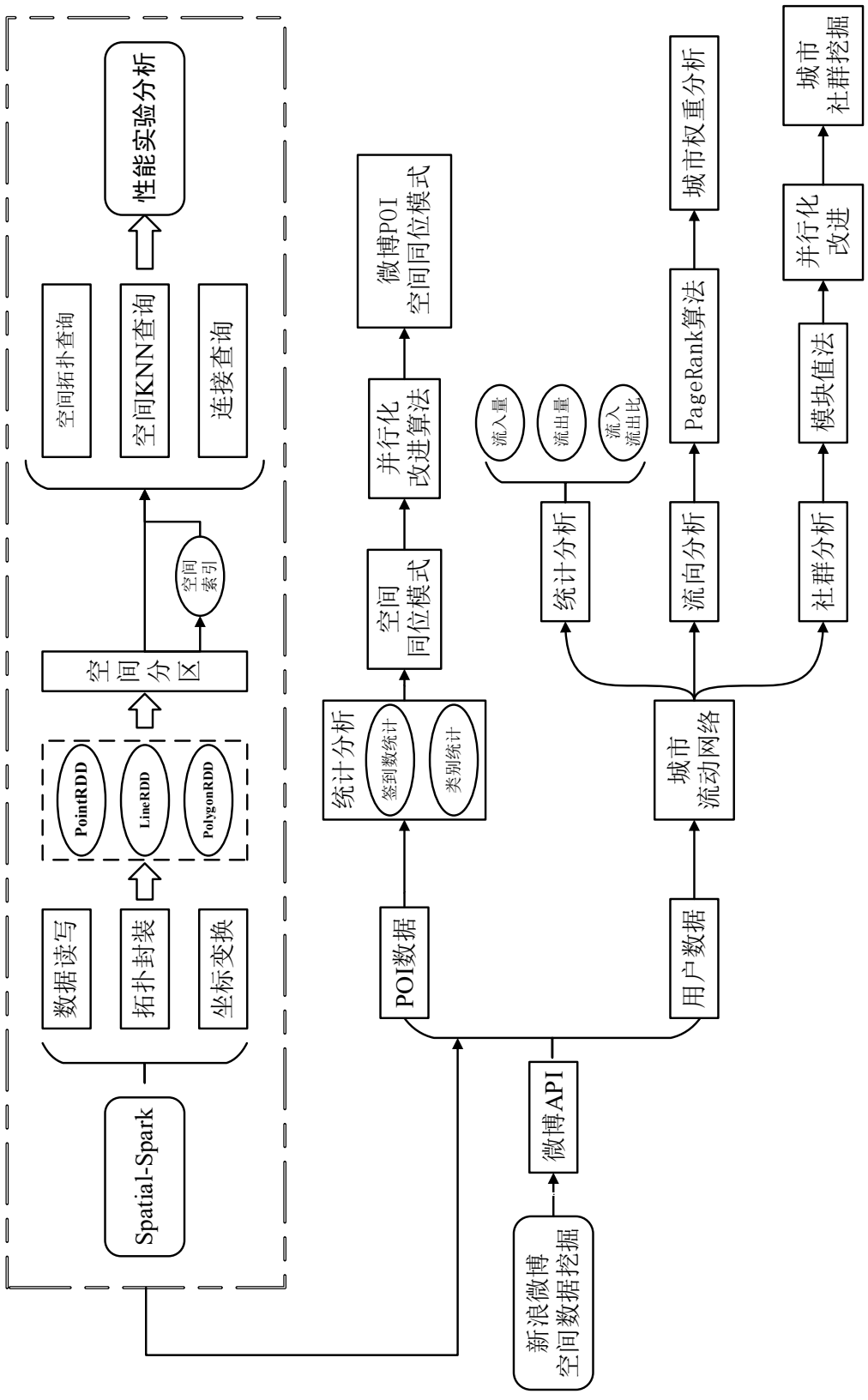

图 1–1 技术路线

Figure 1–1 Technology route

1.4 论文结构安排 (Contents and Structure of This Thesis)

第一章：绪论。概要地说明了论文的研究背景及意义和国内外相关主题研究现状，阐述了本文的工作，以及论文的主要内容和技术路线。

第二章：相关技术。重点介绍了 Spark 并行计算框架，空间数据挖掘相关算法和新浪微博 API 接口。

第三章：Spatial-Spark 计算框架。设计了 Spatial-Spark 计算框架，对空间数据的支持，空间索引的建立和空间运算，最后用实验比较 Spatial-Spark 在空间分析方面的优势。

第四章：新浪微博 POI 数据分析。主要包括类别统计分析和 POI 类别空间同位模式挖掘。

第五章：新浪微博用户流动网络分析。主要包括人口流动量统计、城市权重分析和网络社群挖掘。

第六章：总结与展望。对论文的研究内容和成果进行总结，指出了研究内容的局限性并对后期工作提出建议。

2 相关技术

2 Related Technology

2.1 Hadoop 技术 (Hadoop Technology)

自从 Google 公司公布分布式计算三架马车 MapReduce、GFS 和 BigTable 之后，Apache 基金会开放了 Hadoop 大数据计算平台，并发展成为 Apache 顶级项目，包含了 Google 公司三架马车的开源实现，分别为 Hadoop MapReduce、HDFS 和 HBase。Hadoop 一经推出备受开源社区广泛关注，其拥有较高的容错性、可靠性和适用性等优点，最重要的是 Hadoop 可部署在廉价的普通 PC 机器上，大大降低开发成本。Hadoop MapReduce 是 Hadoop 的计算模型，该模型将所有数据处理流程分解为 Map 阶段和 Reduce 阶段[5]，这个计算模型的优势在于使用简单，它隐藏了并行化、容错、位置优化和负载均衡的细节，使得没有并行和分布式经验的开发人员能够处理业务逻辑，这也是 Hadoop 成为大数据计算框架被广泛关注和研究的原因。

2.1.1 HDFS 机制

HDFS 是 Hadoop 分布式计算框架中数据存储基础，同其他分布式文件系统 PVFS(Parallel Virtual File System)、Lustre 和 GFS 一样，HDFS 将系统元数据存储在特定的节点上，称之为 NameNode；应用程序所需数据存储在其余的节点上，称之为 DataNode。为了保证容错性，需要指定特定节点作为 SecondaryNameNode，它是 NameNode 的备用节点，负责拉取主控服务器的日志。当 NameNode 发生失效，替代原来的 NameNode 的进行工作，所有节点使用 TCP 协议进行连接和通信[2]。

HDFS 采用主从结构，集群一般由一个 NameNode 和多个 DataNode 组成。不同于 PVFS 和 Lustre 的 RAID(Redundant Array of Independent Disks) 文件保护机制，HDFS 将数据文件切分成若干数据块 (Block)，默认为 64M，将这些文件分散到不同 DataNode 节点并且每个数据块拷贝若干份，通过这些措施来提高数据的可靠性，图2–1为 HDFS 体系结构。

HDFS 对数据的操作主要表现为数据的读写和数据块的备份：

数据读写操作：当客户端 (Client) 向 NameNode 发起读取数据请求，NameNode 通过元数据判断该数据存在，如果存在，HDFS 将返回该数据所在的详细位置给客户端。客户端将使用 TCP 协议与数据对应的 DataNode 进行通信读取数据；数据写入的过程与之稍微不同，客户端同时向 NameNode 和 DataNode 发送请求，当 NameNode 接收到来自 DataNode 的消息立即返回确认消息，DataNode 开始进行数据写入。

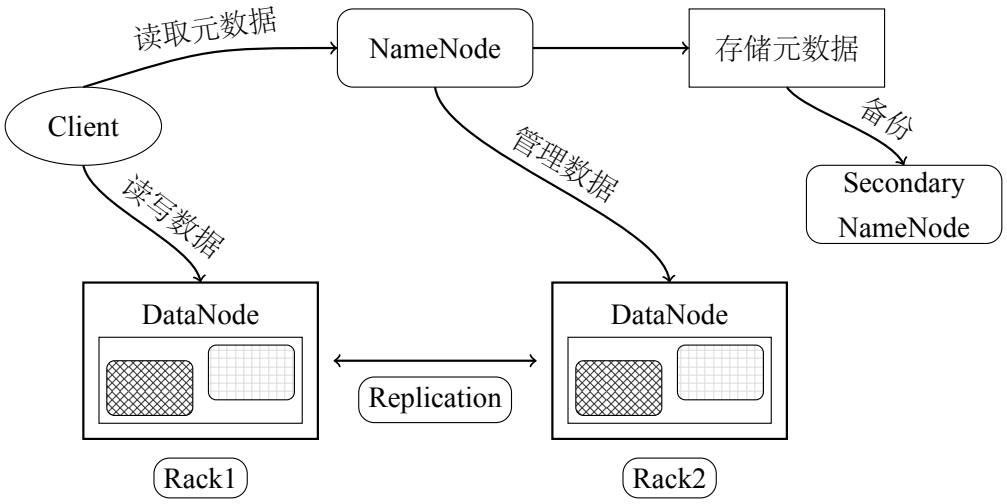

图 2–1 HDFS 体系结构

Figure 2–1 HDFS architecture

数据块备份操作：对于较大的集群，由于 DataNode 众多，不可能同时管理 Block。通常将将这些 Block 分布在不同的机架上，机架内部通信带宽往往比机架间大得多。因此通过 NameNode 接受来自 DataNode 的心跳和 Block 的报告，将 Block 备份到不同的节点和机架上[27]。

2.1.2 MapReduce 编程模型

Hadoop MapReduce 是 Google MapReduce 开源实现，其核心思想则是源于函数式编程语言。它将分布式业务逻辑从复杂的细节中抽象出来，开发人员只需要关注 Map 函数和 Reduce 函数便可以通过集群系统来执行程序，实现分布式计算，而无需关注消息传递和计算任务分配等复杂的问题。

Map 阶段关注的是任务的分解，通过 Map 算子将每个输入输出组成 Key/-Value 键值对作为输入和输出形式。Reduce 算子是对 Map 算子的输出进行汇总，以 Key 按照特定的分区算法，并行执行 Reduce 函数。

以 Word Count 任务为例，MapReduce 运行机制见图2–2，按照执顺序包括：输入分片 (Input Split)、Map 阶段、Shuffle 阶段和 Reduce 阶段[28]。

(1) 输入分片：在进行 Map 计算之前，MapReduce 会根据输入数据计算输入分片，每个输入对应着一个 Map 任务，输入分片和 HDFS 数据文件的 Block 相关，即每个输入数据文件是 HDFS 默认大小的整数倍，不满整数向上取整。如果存在较多的小文件，将会导致 Map 执行阶负载不均衡。

(2)Map 阶段：开发人员重写 Map 函数，以满足特定的业务逻辑需求，而且 Map 函数一般在数据所在节点执行，从而降低了通信消耗时间。

(3)Shuffle 阶段：Map 阶段到 Reduce 阶段之间过程就是 Shuffle 过程，也是 MapReduce 性能提升的重点地方。MapReduce 计算的是海量数据，内存中不可能

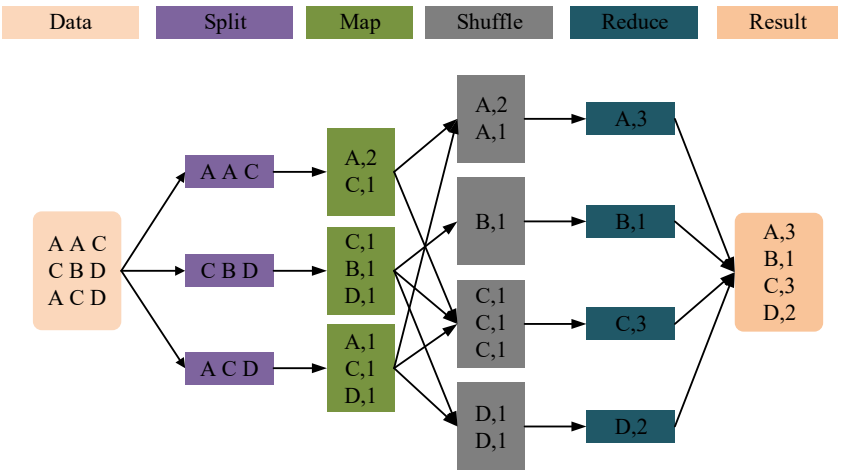

图 2–2 Word-Count 处理流程

Figure 2–2 Word-Count procession

存储所有数据，Map 阶段在输出时会将部分数据写入磁盘，等 Map 输出所有数据后，Map 会采用类似归并排序 (Merge Sort) 算法对输出结果合并。这个过程中会有一个 Partitioner 操作。每个 Partitioner 操作对应一个 Reduce 作业，Partitioner 就是 Reduce 阶段输入分片，开发人员可以编写 Partitioner 函数，提高 Reduce 阶段执行效率。

(4)Reduce 阶段：对来自 Shuffle 阶段的输入数据进行汇总处理，开发人员根据业务需求，自行编写 Reduce 函数，并将结果数据存储在 HDFS 上。

2.1.3 MapReduce 不足

在 Hadoop1.0 版本中，MapReduce 框架有唯一的 Master、JobTracker 和每个集群节点一个 Slaver、TaskTracker 共同组成。其缺点显而易见，JobTracker 是 MapReduce 的集中处理节点，存在单点故障的风险；JobTracker 完成了太多的任务，造成了过多的资源消耗等。在 Hadoop2.0 版本中，推出了 YARN(Yet Another Resource Negotiator) 统一的资源管理平台，该平台能够很好地将不同的任务隔离开来，增加集群的健壮性。

虽然 MapReduce 通过高度抽象，能够很好地描述大部分业务逻辑，但还是过于底层。简单的过滤 (Filter)，分组 (Group by) 等操作都需要编写冗长的 Map 和 Reduce 接口函数，无形中增加了开发人员的负担和出现故障的概率[29]。而且整个流程中的中间过程数据都要缓冲到 HDFS 中，磁盘 IO 时间开销都是系统性能的瓶颈，尤其是针对数据挖掘、机器学习等需要反复迭代的算法，MapReduce 编程模型往往是捉襟见肘。

2.2 Spark 计算框架 (Spark Framework)

Spark 作为炙手可热开源并行计算框架，吸收和借鉴了 MapReduce 思想，并提出了新型的内存计算模型，将整个计算过程中数据保存在集群内存中，从而大大提高了 Spark 表现力。Spark 同样隐藏了并行化的细节，使用者只需要关心业务需求。

2.2.1 Spark 框架体系

BDAS(the Berkeley Data Analysis Stack) 是 AMP 实验室打造的一个开源的大数据处理一体化的技术生态系统，见图2–3, 整个生态主要分为三大部分，核心部分为 Spark Runtime 支持的 Spark Core，其包含了 Spark 最基本、最核心的功能和基本分布式并行计算算子，这些算子提供了 Java，Scala，Python 和 R 编程语言接口。在上层部分是为处理特定业务场景而设计的生态组件，是 Spark 的高层组件，分别为：结构化查询工具 Spark SQL、分布式机器学习库 MLlib、并行图计算框架 GraphX 和流计算框架 Spark Streaming。底层为集群管理器，如 YARN，Mesos 等等，保证了 Spark 运行框架运行的健壮性。

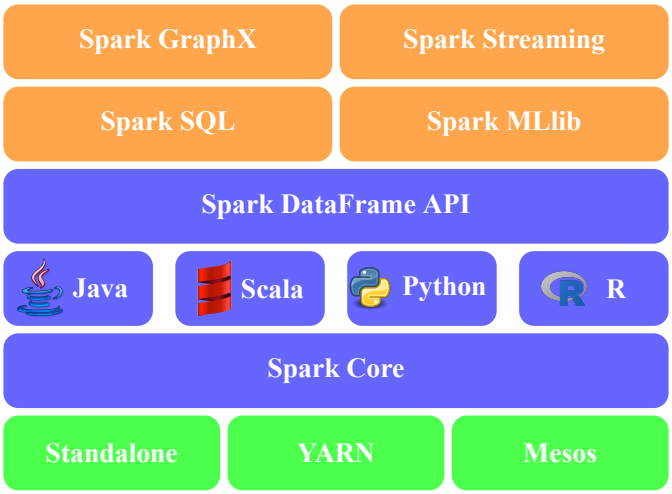

图 2–3 Spark 体系架构

Figure 2–3 Spark architecture

(1)Spark SQL

Spark SQL 是 Spark 1.0.0 版本中新加入的组件，是 Spark 生态系统中最活跃的组件之一，为结构化数据提供了方便的存储和查询操作[30]。Spark SQL 提供了方便的调用接口，用户可以通过使用 Scala、Java、Python 等开发基于 Spark SQL API 的数据处理程序，也支持传统的 SQL 语句与 Spark 进行交互。

(2)Spark MLlib

Spark MLlib(Spark Machine Learning library) 常用的机器学习算法的实现[31]。

Spark MLlib 的在机器学习方面优点主要有以下三点:①Spark 是内存计算的计算框架，这一点非常适合机器学习算法迭代计算的特点，避免 Hadoop MapReduce 这类 IO 频繁的计算框架；②Spark MLlib 使用和部署非常方便，支持 Scala 和 Python 交互式开发环境，方便机器学习算法使用人员快速验证算法原型系统；③Spark MLlib 作为 Spark 生态系统的子成员，可以去 Spark 生态系统进行无缝结合。

(3)Spark GraphX

是 Spark 分布式图计算框架，它是常用图算法在 Spark 上的并行实现，并且提供了图计算中用于图和图计算中的 API。是 Graph Lab 和 Pregel 在 Spark 上的重写及优化, 是 Spark 生态圈中非常重要的组件[32]。

Spark GraphX 的核心抽象概念是弹性分布式属性图 (Resilient Distributed Property Graph)，是一种点和边都带有属性的有向多重图。它扩展了 Spark RDD 的抽象，实现了统一表示。弹性分布式属性图有 Table 和 Graph 两种视图，对应的这两种视图只需要一份物理存储，两种视图都有各自的操作符，基于 Spark 的 RDD 可以很轻松的进行操作。通过分布式计算提高了效率。

(4)Spark Streaming

Spark Streaming 是建立在 Spark 上流应用计算框架[33]，它将流式计算分解成一系列短小的批处理作业，也就是将输入数据按一定大小划分为一段段的数据 (Discretized Stream)，每一段数据都 RDD，然后将 Spark Streaming 中对 DStream 的 Transformation 操作变为针对 RDD 的 Transformation 操作，将 RDD 经过操作变成中间结果保存在内存中。

2.2.2 RDD 介绍

弹性分布式数据集 RDD，是 Spark 的核心抽象，是对分布式内存的抽象表达，它表示已被分区、只读的、并提供了一组丰富的操作方式来操作这些数据集合。这些数据集可以全部或者部分缓存在内存中，在多次计算间重复使用，省去了大量的磁盘 IO 操作。

现有的并行计算框架的数据结构在处理两种应用场景显得不够高效：①迭代式计算，主要为分布在图计算领域和机器学习问题中；② 交互式计算工具，操作能够立即返回结果。RDD 通过高度受限的共享内存方式解决上述问题，基于稳定物理内存中的数据集合执行批量操作 (如 Map，Join 和 Group by) 操作。与其他分布式共享内存系统选择检查点 (Check Points) 和回滚 (Roll Back) 机制不同，RDD 选择血统 (Lineage) 机制来保证可靠的容错性，每一个 RDD 记录如何从 RDD 中衍生出来的，一旦某个 RDD 发生数据丢失，根据这些记录进行重新构建，而不需要进行高昂代价的检查点操作来进行数据恢复。虽然 RDD 不是通

用的共享内存的抽象，却包含了高可靠性、强可伸缩性和良好的表述的能力，因此成为了 Spark 一切操作的核心。

RDD 从本质上来讲，就是只读的集合，不过集合是分布在不同的分区上的，每个分区就是一个 Dataset。RDD 在进行转换的过程中产生了依赖，如果 RDD 的每个分区被一个子 RDD 继承，那么称之为窄依赖 (Narrow Dependency)；若 RDD 的分区被多个子 RDD 继承，那么称之为宽依赖 (Wide Dependency)。不同的操作产生不同的依赖。图2–4分别展示了 map 产生的窄依赖和 join 产生的宽依赖[34]。

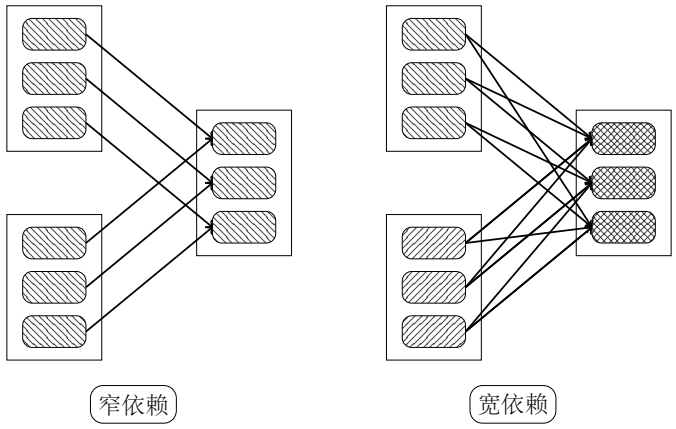

图 2–4 窄依赖和宽依赖

Figure 2–4 Narrow dependency and wide dependency

RDD 提供了丰富的编程接口来操作数据集合，主要分为两种：Transformation 和 Action。

Transformations 返回值还是一个 RDD，如 Map，Filter，Union 等操作。它可以理解为任务分配的过程，其采用的是 Lazy 策略，如果是只提交 Transformation 是不会提交任务执行，只有在 Action 操作的时候才回被触发。

Action 操作是将 RDD 持久化起来，每调用 Action 操作，都会触发一次 Spark 的作业提交，将规划的任务 (Job) 提交给计算引擎，由计算引擎将其分为多个 Task，再分发相应的计算节点，开始真正处理过程。从目前的 Spark 的 API 来看，Action 操作主要分为两种：①Action 操作将标量或者集合返回给 Spark 的客户端程序。②Action 操作将 RDD 直接保存到外部文件系统或者数据库中。

2.2.3 Spark 优势

Spark 一栈式解决方案有很多优势，具体如下:

(1) 快速处理。由于 Spark 采用内存计算，避免了 Hadoop 的磁盘操作，节省了 I/O 操作，在迭代式计算上 Spark 有着明显的优势。

(2) 通用性强。Spark 提供了一个强大的技术堆栈，是一个可以处理流式处理、机器学习、即时查询、图分析等多种无缝大数据处理链接计算平台。

(3) 可以与 Hadoop 集群集成。Spark 可以单独运行，也可以运行在 Mesos、Yarn 等集群资源管理系统上，也可以读取已有的任何 HDFS 数据。

2.3 空间数据挖掘 (Spatial Data Mining)

空间数据挖掘是多门学科交叉研究领域，包含了多种研究技术，常用的方法有但不仅限于地理信息系统、计算几何、模式识别、机器学习、深度学习等。因此空间数据挖掘技术丰富多彩，下面简单介绍几种常用的方法。

(1) 统计分析方法

统计方法是空间数据分析最基础的方法，以数学统计知识为基础。关注点是空间对象的非空间属性，简单的统计方法有最大值，最小值，平均值，方差等等，复杂的方法有方差分析，P 值估计等等，通常来讲这些统计结果都用图表进行表达。

(2) 聚类分析方法

聚类分析方法是非监督数据挖掘中最主要的方法，通过对空间对象相似性判断，将整个空间对象划分为若干簇[35]。在聚类算法中，距离函数非常重要，常见的距离函数选择有欧几里得距离、曼哈顿距离和更通用的 L_p 距离，除了数值型属性之外，空间聚类还需要将空间位置纳入考虑范畴，因此不同属性的应当赋予不同的权重。

(3) 空间分析方法

空间分析是空间数据挖掘的特色，通过各种空间分析，对空间信息进行加工，从而发现更多知识。常见的方法有空间缓冲区分析、空间核密度分析和空间趋势面分析等等，空间分析常常结合地学第一定律进行统计分析，拓展了统计分析方法关注的内容，发现新的知识[36]。

(4) 空间关联规则挖掘方法

关联规则挖掘是根据销售事务数据库交易项商品同时出现的规律[37]，Apriori 算法是关联规则挖掘最著名的算法。空间关联规则是关联规则在空间方面的拓展，将空间关系纳入到考虑范畴。以空间关联为基础，发展出空间同位模式挖掘，比如生物在生态环境中的共生现象。

2.4 新浪微博数据接口 (Weibo API)

目前新浪微博数据获取方式主要有两种:[38]: ①通过网络爬虫程序获取数据; ②通过新浪微博提供了 API 接口获取相应主题的数据。

网络爬虫是能够自动获取并下载网页的计算机应用程序[39]，根据一定规则不停的向网络服务器发送请求，获取特定的信息。由于互联网是通过 URL 进行连接，可以不间断对整个网络内容进行获取，需要制定相应的爬虫策略，一般有

与深度优先和广度优先两种调度策略。互联网服务器则将网页中的文本内容、图片、音频视频等内容发送给应用程序，应用程序则将这些数据按照特定的格式保存下来。

由于个人用户频繁使用爬虫从网站中获取数据将会对网站服务器带来较大的负载，因此新浪微博之类的网站会从爬虫程序的 IP 地址或者验证码方式限制请求次数。而且通过爬虫获取的数据噪声较大，需要进行大量的清洗工作，因此本文将使用新浪微博 API 接口获取海量空间数据。

2.4.1 微博 API 使用

作为丰富数据资源的平台，新浪微博给其他应用程序开放了访问这些数据的 API，应用程序通过这些 API 获取微博内容、用户个人信息、签到和空间位置等相关信息。这些 API 其实就是一系列 HTTP Get 请求，API 接口的参数作为请求的参数发送给新浪微博服务器，新浪微博返回相应的结果。新浪微博将这些 API 封装到不同编程语言的 SDK 中，包括了 Python、Java，C#、PHP 等。

微博 API 的使用需要对用户的身份进行鉴别，目前新浪微博对开放平台用户采用 OAuth2.0 的方式进行授权[40]。OAuth2.0 授权验证具有简单、安全等特点，是社交网络平台对外开放权限的主要方式，使用流程见图2–5。

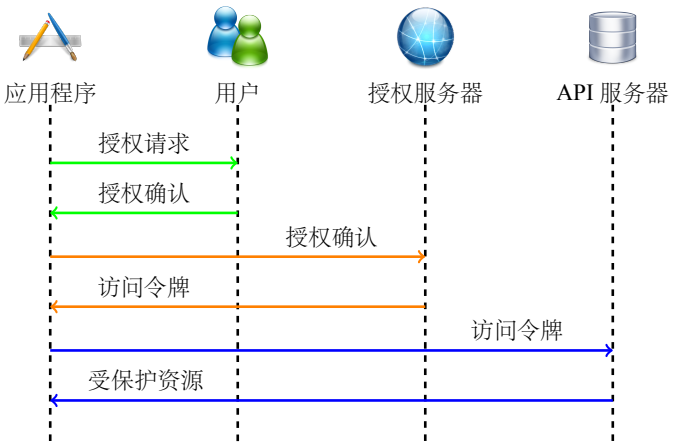

图 2–5 新浪微博 API 使用流程

Figure 2–5 The usage of weibo's api

第三方应用程序想要获得 API 接口的使用权，首先要在新浪微博开发平台进行申请应用程序的 AppKey 和 AppSecret，这个 AppKey 和 AppSecret 是该应用程序的标识码。用户在使用该应用程序的时候，首先要在授权页面登录自己新浪微博账号，给予授权。然后应用程序将该用户的授权发送给新浪微博的授权服务器，返回访问令牌 (Access Token)。那么该应用可以以访问令牌访问这些 API 接口，获得相应的数据，在整个获取数据过程中用户的隐私得到保护。

2.4.2 空间数据相关 API

新浪微博 API 接口主要有粉丝服务接口、微博接口、收藏接口、公共服务接口、位置服务接口、地理信息系统接口和地图引擎接口等，部分空间数据相关 API 接口说明见下表2–1。

表 2–1 微博接口描述 (部分)

Table 2–1 Description of Weibo api(partly)

接口 API 地址	说明
location/geo/address_to_geo	根据地址返回地理信息坐标
location/pois/search/by_area	根据区域按坐标点获取 POI 点的信息
location/pois/show_batch	批量获取 POI 点的信息
location/citycode	城市代码对应表
place/nearby/pois	获取附近的 POI 点
place/pois/show	获取地点的详细信息

新浪微博 API 接口使用非常简单，传入相应的接口指定的参数，即可返回查询结果。以 location/geo/address_to_geo 接口为例，参数见表2–2。

表 2–2 接口参数说明

Table 2–2 Description of api's parameters

参数	类型	说明
access_token	String	采用 OAuth 授权方式必填参数
address	String	需要获取的坐标的实际参数

当应用程序通过该接口将符合要求的参数传入发送请求后，服务器将结果以 JSON 的格式将结果返回至客户端。

2.5 本章小结 (Chapter Summary)

本章首先介绍了 Hadoop 大数据计算框架，分别介绍了 Hadoop 分布式存储机制 HDFS 和分布式计算模型 MapReduce，并分析了其中的优缺点；然后介绍了目前流行的并行计算框架 Spark，着重介绍了 Spark 的核心数据结构 RDD 和上层应用接口；接着介绍了常用的空间数据挖掘算法，最后介绍了新浪微博 API 使用方法和与空间相关数据获取接口。

3 Spatial-Spark 计算框架

3 Spatial-Spark Computation Framework

3.1 空间数据分析处理 (Procession of Spatial Data)

3.1.1 空间数据格式

空间数据量大且空间数据格式也是各式各样。开放地理空间信息联盟（Open GIS Consortium, OGC）定义了一些空间矢量数据格式[41]，方便能够进行空间数据分析处理和交换，下面介绍几种常用的空间矢量数据格式。

(1)WKT 数据格式

WKT（Well-Know-Text）数据格式以文本形式描述，用来表示点，线和面空间对象，见表3–1。

表 3–1 WKT 矢量空间数据

Table 3–1 Vector spatial data in WKT

几何类型	WKT 表示方式
ST_Point	POINT(10.05 10.28)
ST_LineString	LINESTRING (10.05 10.28 , 20.95 20.89)
ST_Polygon	POLYGON((10 10, 10 20, 20 20, 20 15, 10 10))

(2)GeoJSON 数据格式

GeoJSON 数据是通过 JSON 数据表达简单的数据格式，如点、线、多边形和这些几何类型的集合以及他们非空间属性信息，表达方式如下:

```
{
  ”type”:”feature”,
   ”geometry”:{
     ”type”:”LineString”,
       ”coordinate     ”:[[[-100.50,57.14],[-89.45,62.17],[-37.21,79.73]]],
       ” fields ”:{
            ”prop1”:”value”,”prop2”:”string”
            }
   }
}
```

由于 JSON 格式在序列化和网络传输中的优势，非常适合分布式的数据存储，空间数据和属性数据都存储在普通的文本中。

(3)GML 数据格式

GML(Geography Markup Language) 是以 XML 格式的空间数据表达形式，并使用 XML Schema 文件定义技术，目前版本为 2.1.1。XML Schema 具有类型继承、命名空间等特性，通过 XLink 来表现地理空间实体的关系。

```
<PhotoCollection xmlns="http://www.myhotos.org"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.myphotos.org" >
    <items>
      <item>
        <name>Lynn</name>
          <description>A shot of the falls</description>
          <where>North Vancouver</where>
          <position>
            <gml:Point srsDimension="2"
              srsName="http://www.opengis.net/def/crs/EPSG/0/4326" >
                <gml:pos>49.40 -123.26</gml:pos>
            </gml:Point>
          </position>
      </item>
    </items>
</PhotoCollection>
```

3.1.2 空间数据转换

空间数据来源各异，每种数据来源有各自的坐标参考系统，如 GPS 接收机获取的地理空间数据是以 WGS84 坐标系统为基准；摄影测量往往采用该区域最适宜的坐标参考系统为基准。不同的大地坐标系统和平面投影系统往往使得空间数据处理和分析正确性得不到保证，因此很有必要将多源空间数据纳入到同一个空间参考系统下。

Proj.4 是开源 GIS 中著名的地图投影库[42]，在 GRASS GIS, MapServer, PostGIS 等众多 GIS 软件中都直接或者间接使用 Proj.4，2008 年 OSGeo 将 Proj.4 纳入到 MetaCRS 的一部分，Proj.4 的主要功能是提供了大地坐标系和投影坐标之间的正反算，不同参考基准的坐标变换。Proj.4 使用 C 语言编写，但不同的语言也有各自相应的 Proj.4 库，如 Java 语言的 Proj4j，C# 语言的 Proj4.Net，JavaScript 语言的 Proj4js 等等。

以 Java 语言的 Proj4j 库为例，Proj4j 主要模块的如下:

(1)Datum 基准

定义了空间椭球基准，主要参数包括椭球的名称，长半轴和短半轴，以及质

心偏离中心的位置，Proj4j 内置了若干个常用的椭球基准如 WGS84,IRE65 等。

(2)CoordinateReferenceSystem 参考系统

不同的坐标参照系统需要指定不同的参数，如高斯投影的坐标参考系统需要指定中央经度和投影带宽度，而兰伯特投影需要指定投影的第一纬度、第二纬度，中央纬度和中央经度。

(3)Projection 投影

Proj4j 提供了 96 种投影方式，Projection 是这些投影类的基类，每个投影类提供了 project 函数和 projectInverse 函数，分别代表了投影正算和投影反算。

(4)CoordinationTransform 坐标换算

坐标转换提供了不同坐标参考系统之间坐标换算类，如将高斯投影坐标换算兰伯特投影坐标，通过两个坐标参考系统对象创建坐标换算类，生成 CoordinationTransform 对象，调用 transform 函数完成坐标换算。

3.1.3 空间数据分析

空间数据分析的正确性是 GIS 的一个重要参照指标，尤其在涉及空间数据数据几何拓扑关系时需要严格正确。JTS Topology Suite 是加拿大 Vivid Solutions 公司提供的一套开源的空间几何对象拓扑操作工具包，主要包含的功能和特色如下：

(1) 实现了 OGC 关于简单要素的 SQL 查询规范定义的空间数据模型。

(2) 完整的、一致的和基本的二维空间算法的实现，包含了空间分析和空间运算[43]。

(3) 提供了常见的空间数据格式读写接口。

JTS Topology Suite 在 GIS 中最重要的应用是计算两个几何对象之间的空间拓扑关系，每个派生自 Geometry 类的几何对象都能够进行相互空间分析和空间运算，详细见表格3–2。

表 3–2 空间分析和空间运算

Table 3–2 Spatial analysises and spatial operations

类别	函数	功能	类别	函数	功能
空间分析	Contain	包含关系	空间分析	Overlap	重叠关系
	Equal	相等关系		Touch	相接关系
	Intersect	相交关系	空间运算	Buffer	缓冲区运算
	Within	内部关系		Intersection	交集运算
	Disjoint	相离关系		Difference	差集运算
	Cross	内部相交关系		Union	并集运算

3.2 RDD 空间扩展 (Spatial Extension of RDD)

空间数据爆炸式增长对空间数据处理提出了新的挑战，主要有以下两点：①系统可扩展性 (System Scalability)：数据的存储、读取和写入能够有效地处理 GB、TB 级别甚至 PB 级数据；②交互式高性能 (Interactive Performance)：在处理空间数据查询后，能够高效回应查询请求[20]。

3.2.1 空间数据类型

Spark 采用抽象数据类型 RDD，通过分布式平行数据结构处理海量数据，使用独特的内存计算使得在处理数据方面有着较高的性能优势。但是 RDD 没有完善的空间数据支持和空间操作，因此开发人员需要在 Spark 提供 API 上层重新定义空间数据读写和操作函数，但这些没有统一的标准。

为了在分析海量空间数据时能够专注于分析算法，本文设计出一套海量空间数据分析框架：Spatial-Spark。Spatial-Spark 扩展了 Spark RDD，使之能够支持常见的空间数据类型、空间索引和空间操作，整个 Spatial-Spark 体系结构见图3–1，底层为数据存储层，核心部分为中间 Spatial-RDD 空间几何对象层，最上层为空间分析层，提供了高度定制的 API 接口。

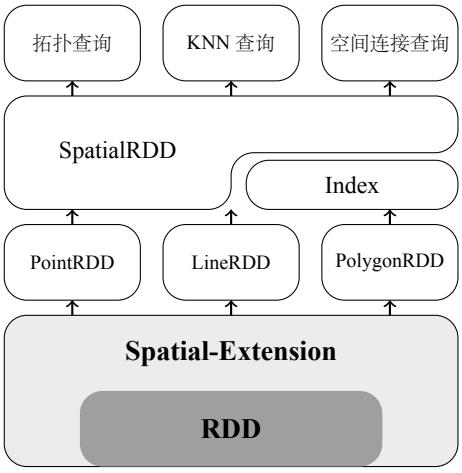

图 3–1 Spatial-Spark 体系架构

Figure 3–1 Spatial-Spark architecture

空间数据类型是 Spatial-Spark 的核心，通过 RDD 将点、线和面分别封装成 PointRDD、LineRDD 和 PolygonRDD。这些 Spatial-RDD 表示在集群中并行分区存储空间几何对象的集合，主要有两种方式生成 Spatial-RDD：①从 HDFS 中读取数据，主要流程见图3–2，文本空间数据以行记录存储在 HDFS 中，先经过文件格式装换，将 WKT，GeoJSON 等数据转换成空间对象，再空间数据坐标参考系统转换统一，最后生成 Spatial-RDD，其中转换和投影步骤以接口形式提供，用户可以重写接口，完成定制化需求；②Spatial-RDD 相互之间空间运算，如 PointRDD

可以通过缓冲区操作生成 PolygonRDD 对象，PolygonRDD 对象通过提取重生成 PointRDD。

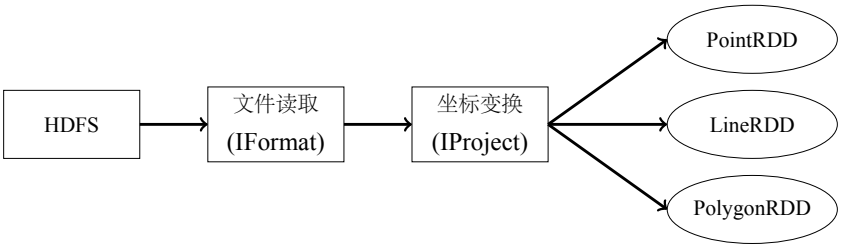

图 3–2 HDFS 生成 Spatial-RDD 流程

Figure 3–2 Generating Spatial RDD from HDFS

3.2.2 空间数据索引

RDD 分区

RDD 内部数据集合在逻辑上 (以及物理上) 被划分为多个小集合，这样每个小集合就被成为分区。以图3–3为例，RDD1 有五个分区 (Partition)，分布在四个 DataNode 上面，而 RDD2 有三个分区，分布在三个 DataNode 上面。在源码级别，RDD 类存储一个 Partition 列表，每个 Partition 对象都包含一个 index 成员，通过 RDD 编号加 index 就能从唯一的分区的 Block 编号[34]，持久化的 RDD 就能通过这个 Block 变化从 HDFS 中获取对应的分区数据。

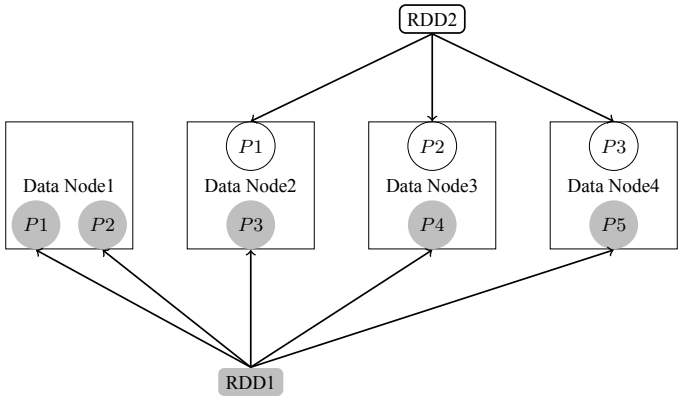

图 3–3 RDD 分区

Figure 3–3 RDD partition illustration

分区的个数决定了并行计算的粒度，多个分区能够并行计算，充分利用分布式计算资源。通常来讲，分区的数量为计算集群的 CPU 核心数量的 3 – 4 倍。创建分区的方法主要有两种：①在 SparkContext 对象读取数据的时候指定分区的数量；②调用 RDD 的分区器函数，进行重新分区。Spark 在读取数据时默认使用哈希分区器，该分区器实现简单，运算速度快，但缺点是不关心键值的分布情

况，其散列到不同分区的概率会因数据而异。往往会带来分区负载的不平衡性。

考虑到空间数据空间分布特点，Spatial-Spark 提供了空间分区，重写了 Partitioner 函数，能够尽可能将空间上分布相邻的空间对象处于同一 RDD 分区。具体算法步骤如下：

(1) 根据空间要素集合的边界 (boundary) 和分区数量 n，将整个 boundary 划分为 $\sqrt{n} \times \sqrt{n}$ 个网格，每个网格拥有一个 id 值。

(2) 将RDD<Geometry> 进行 Map 或者 MapPartition 操作，使之转换为 PairRDD<Integer,Geometry> 描述的 Key-Value 对象，其中 Key 为网格中与其相交或者包含的网格的 id 值，如果有多个网格与之相交，取其中一个。

(3) 对PairRDD<Interger,Geometry> 按照 key 进行重分区计算，使每个分区能够拥有分布较为均匀的几何对象，而且尽量保证相邻的空间的对象在同一个分区。

分区索引

好的空间索引能够极大地方便海量空间数据查询，而空间数据中最常用的空间数据索引方式就是 R 树。R 树索引是有美国加州大学 Guttman A. 教授提出的一种空间数据库的动态索引算法[44]，该数据结构的核心思想是通过最小外包矩形 (Minimum Bounding Rectangle, MBR) 表达一个或者一组空间几何对象。与 B 树一样，R 树是一棵高度平衡树，将所有的空间几何对象都存放在叶节点，插入和删除节点不需要完全重构 R 树，只需要在局部进行相关拓扑调整即可，一棵典型的 R 树如图3–4所示。

(1)R 树插入

作为 B 树的一个变种，R 树的插入与 B 树相类似，首先根据待插入空间对象的最小外包矩形待确定插入的叶节点，然后插入对象，如果该叶节点包含的空间对象数目超过规定的最大数目，则将该节点进行分裂[45]，并将其中的一个节点向上传递，递归执行，如果直至根节点，完成树高度的提升。节点分裂的好还的标准是两个新节点的最小外包矩形的面积之和最小。

(2)R 树查询

R 树查询只能检索出与给定窗口相交的空间实体对象。如果两个最小外包矩形是分离的，那么他们所代表的空间实体也肯定是分离的，但如果两个最小矩形有相交，不能保证所包含的空间几何实体能够窗口相交。因此，R 树空间检索检索策略是：

过滤：从 R 树中筛选出候选节点，排除那些的不能满足相交条件的几何对象，但在候选几何对象中也可能包含一些不满足条件的对象。由于判断最小外包矩形是否相交的计算成本低，可大幅度提高计算效率。

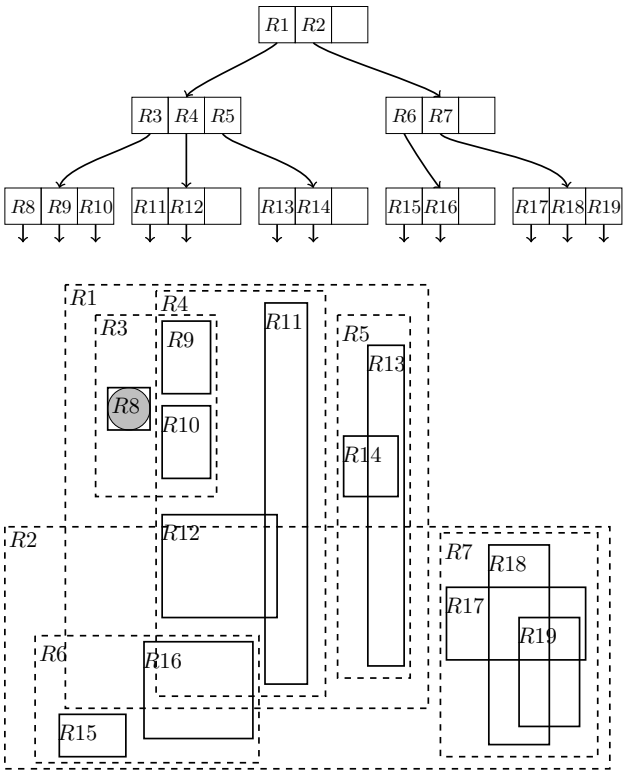

图 3–4 R 树示意图

Figure 3–4 R tree illustration

精选：依次判断每个候选几何对象，判断与查询窗口是否相交。可以借助 JTS Topology Suite 工具判断几何对象之间的拓扑关系。

在 Spatial-Spark 框架中，为了加快空间分析的速度，可选择在 RDD 的每个分区建立 R 树空间索引。通过 RDD 的 MapPartition 函数将同一分区的空间对象汇集起来，每个分区建立一棵 R 树，将同一分区的几何对象的最小外包矩形插入树中。通过 Spark 缓存机制，将索引内容缓存到 Spark 内存中，方便后续查询分析等工作。

3.2.3 空间数据查询

空间查询是 GIS 中最重要组成部分，在大规模空间数据中查询也提出了空间数据查询实时性要求。Spatial-Spark 在 Spatial-RDD 上层提供了空间查询层，包括空间拓扑查询 (Spatial Topology Query)，空间 k 邻居查询 (Spatial k Nearest Neighbor Query) 和空间链接查询 (Spatial Join Query)。使用者向查询层发起空间查询请求，Spatial-Spark 将查询的结果返回。如果空间数据在 Spatial-RDD 层事先建立的了索引，那么用户可以借助索引，提高查询速度。

空间拓扑查询

空间拓扑关系是空间分析的特色，空间拓扑查询过程是给定一个空间几何对象和空间拓扑关系条件，从特定的空间数据集筛选出符合条件的空间数据。

平面空间实体对象引入空间实体外部、内部和边界，构成了空间实体的基本组件。假设空间实体 A 的边界 ∂A，内部为 A°，补为 A^{-}，空间实体 B 的边界为 ∂B，内部为 B°，补为 B^{-}，两两的交集就构成了空间关系描述的 9 元组矩阵[46]。

$$R(A,B)=\begin{bmatrix}\partial A\cap\partial B & \partial A\cap B^{\circ} & \partial A\cap B^{-}\\ A^{\circ}\cap\partial B & A^{\circ}\cap B^{\circ} & A^{\circ}\cap B^{-}\\ A^{-}\cap\partial B & A^{-}\cap B^{\circ} & A^{-}\cap B^{-}\end{bmatrix}$$

矩阵中每一个元素的取值都有空集和非空集两种，9 个元素共存在 512 中可能，当然其中绝大部分的空间拓扑关系时不存在的。以点、线和面为准给出所有可能空间对象之间可能拓扑关系，见表3–3。

表 3–3 空间几何拓扑关系

Table 3–3 Geometry topology rules

几何对象	点	线	面
点	Equal Disjoint	Within Disjoint	Within Touch Disjoint
线	Contain Disjoint	Equals Within Overlap Contain Disjoint Intersect	Within Disjoint Touch Intersect
面	Contain Disjoint	Contain Disjoint Touch Intersect	Equal Within Disjoint Touch Intersect Overlap Contain

RDD 中的 filter 算子可以筛选 RDD 中符合条件的元素，Spatial-Spark 实现了 Function 接口的通用空间拓扑查询类。该类接受待查询的空间元素 Geometry 和待判断的空间拓扑关系条件 Condition。在类中重写 call 函数即可, 通用实现如

下。

```
@Override
public Boolean call(Geometry goe){
    return geo.condition(this.query)
}
```

当在 Spatial-RDD 中如果已经在每个分区构建好 R 树索引，可以通过索引查询相交 (Intersect)、包含 (Contain)、相等 (Equal)、重叠 (Overlap) 和被包含 (Within) 的拓扑关系。RDD 中的 FlatMapPartition 算子可以对每个分区进行操作，并将结果展平 (Flat) 返回，用户定义 Spatial-Spark 预先实现了 FlatMapFunction 接口的类，该类构造函数只包含带查询对象的最小外包矩形 (Envelope)，通过 R 树高效的查询函数返回最小外包矩形与待查询的外包矩形的空间几何对象。索引查询是初步空间查询，在返回的结果中，在对数据进行精确空间拓扑查询, 通用实现如下。

```
@Override
public Iterator<Geometry> call(Iterator<STRTree> t){
    STRtree tree = t.next()
    return tree.query(this.query.getEnvelopeInternal())
}
```

空间 k 邻居查询

空间 K 个近邻居查询在现实生活中有着广泛应用，尤其在移动互联网时代，位置推荐、商场选址和公共交通等方面有着广泛应用[47]。Spatial-Spark 在 K 邻居查询，算法借助优先级队列数据结构，算法主要分为两步：①选择：针对每个分区，接受一个查询点和查询邻居数量 K，在该分区中，构建一个容量为 K 的优先级队列，依次将计算查询点与分区内几何对象的空间欧氏距离，并添加至优先级队列中，如果队列已满，将集合中距离最大元素删除再添加；②合并：针对每个分区的优先级队列，再次筛选出前 K 个最小距离的几何要素，KNN 通用查询类如下。

```
@Override
public Iterator<Geometry> call(Iterator<Geometry> inputs){
        while(inputs.hasNext()){
            if(pq.size()< this.k){
                pq.offer(inputs.next());
            }else{
                Geometry geo = inputs.next();
                double distance = geo.distance(this.query);
```

```
                    double longestDistance = pq.peek().
distance(this.query);                    if (longestDistance > distance){
                        pq.poll();pq.offer(geo);
                    }
                }
            }
}
```

空间连接查询

空间连接查询是常用且复杂的空间数据查询操作，类似于数据库的两张表 Join 查询，从两个空间对象集合中查询出符合特定空间关系的空间几何对象对[48]。例如两个空间数据集 A 和 B，空间连接是从 A 中的空间对象到 B 中的空间对象使用空间拓扑关系 t，返回分别来自 A 和 B 的满足条件的几何对象对，空间对象的复杂性和海量性将会导致空间连接运算需要大量的计算，时间复杂急剧上升。Spark 的并行计算特点为空间连接运算提供了新的模式，因此 Spatial-Spark 采用了并行化方式实现了空间连接查询算法。算法的主要步骤分为两步。

(1) 过滤步骤

借鉴 Spatial-Spark 空间分区算法，首先将其中任意空间数据集的边界按其最小的几何对象外包矩形分割成规则的网格，每个网格赋予编号。将空间数据集的每一个元素与判断与之相交的最小外包矩形编号，通过 RDD 相应的 MapToPair 操作转换成 PairRDD，其中键 key 为网格的编号，值 value 为空间几何对象，再使用转换函数 cogroup，按照 key 将连个要素集合并起来。

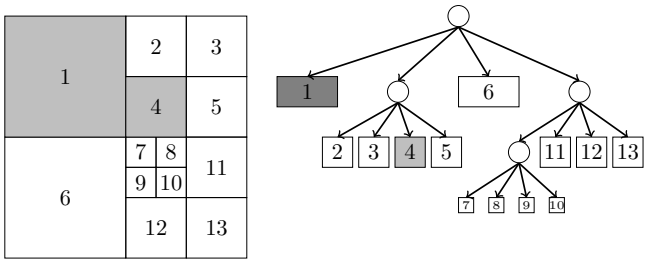

图 3–5 四叉树示意图

Figure 3–5 Quadtree illustration

在空间元素最小外包矩形与格网相交时，可以预先将整个格网数据进行预处理。四叉树也是一种树状索引数据结构，地理空间对象局部范围信息可以用四叉树进行存储，其采取从整体到局部划分的方式建立索引。建立过程如下：对空间范围平面平均分成四个部分，如果改部分的空间对象属性一致，则不再划分，否则继续划分四个小部分，如此重复递归执行。每划分一次，四叉树深度增加 1，

对于深度为 n 的四叉树索引，包含的空间对象至多为 $(2^n \times 2^n)$ 个，四叉树示意图见3–5。

对于空间连接查询中预先处理的格网建立的四叉树索引，所有的叶节点的深度相同，平均每次查询的时间复杂度为 $\log_4 n$，而遍历整个格网的时间复杂度为 n。

(2) 求精步骤

在过滤阶段中，同一网格内空间对象，再根据空间拓扑要求精确筛选，返回结果中每个元素表示一对满足条件的空间拓扑要求的几何对象对，整个空间连接过程见算法3-1。

算法 3-1 空间连接查询

输入:

空间 RDD1:spatial_rdd1;

空间 RDD2:spatial_rdd2;

输出:

空间连接对象对, spatial_join;

```
1: // 与网格相交生成 key-value;
2: spatial_pair_rdd1 = spatial_rdd1.flatToPair();
   spatial_pair_rdd2 = spatial_rdd2.flatToPair();
3: // 按 key 进行 cogroup 操作;
4: spatial_pair_groups = spatial_pair_rdd1.cogroup(spatial_pair_rdd2);
5: // 每个 value 进行详细拓扑判断
6: spatial_pair_values = spatial_pair_groups.mapValue();
7: // 去重操作
8: spatial_pair_join = spatial_pair_values.reduceByKey();
9: // 去掉 key
10: spatial_join = spatial_pair_join.mapToPair();
11: return spatial_join;
```

3.3 Spatial-Spark 实验分析 (Experiments and Analysises of Spatial-Spark)

3.3.1 实验平台及配置

为了验证 Spatial-Spark 计算框架在空间数据分析中的优势，搭建了 Hadoop/Spark 计算集群[49,50]，整个集群有 10 台服务器组成，每台服务器安装了 VMware 虚拟机，每台虚拟机使用的操作系统为 Centos 6.4 Linux 操作系统，每台虚拟机的配置见表3–4：

Hadoop/Spark 集群部署繁琐，涉及到各个方面的技术，其中包括 Linux 系统的配置，Hadoop 的配置与调试，Java 和 Scala 语言库的安装，Spark 计算框架配置

表 3–4 集群配置

Table 3–4 Cluster configurations

项目	配置
内存	4G
CPU 核心	双核四线程
硬盘容量	30G
操作系统	CentOS 64 位
Hadoop 版本	2.6
Spark 版本	1.4
JDK	1.7
以太网络	1000Mbps

和部署。为了能够使集群能够相互通信，对 10 台服务器 IP 地址划分和 Hostname 修改，因为集群之间需要进行数据交换处理，所以要配置使节点之间能够 SSH 免密码通信。各节点 IP 配置见表3–5：

表 3–5 节点 IP

Table 3–5 Nodes' IP addresses

节点	IP 地址	节点	IP 地址
Master	192.168.5.100	Node5	192.168.5.105
Node1	192.168.5.101	Node6	192.168.5.106
Node2	192.168.5.102	Node7	192.168.5.107
Node3	192.168.5.103	Node8	192.168.5.108
Node4	192.168.5.104	Node9	192.168.5.109

其中 Master 节点为主节点，守护 Hadoop 的 Namenode 进程、Yarn 的 ResourceManager 进程和 Spark 的 Master 进程。而 Node1 ~Node9 节点为从节点，守护 Hadoop 的 Datanode 进程和 Spark 的 Worker 进程，其中 Node1 节点另外守护 Hadoop SecondaryNameNode 进程，当 Master 节点的 Namenode 进程出现故障，Node1 节点将被担当起 NameNode 节点的作用。

Spark 计算模式主要分为三种：Local 模式、Standalone 模式和 Yarn-Cluster 模式。Local 模式是在本地运行，Spark 的 Local 模式部署简单，适合单机上调试编写的 Spark 应用程序；Standalone 模式是 Spark 自身实现资源调度框架[51]，当不需要其他计算框架的时候如 MapReduce、Storm 等，只使用 Spark 进行大数据计算时，可以采用 Standalone 模式，其中 Spark Shell 交互式运行环境就是运行在该模式之上；Yarn-Cluster 模式借助 Yarn 统一管理整个计算资源，用户在 Yarn 集群中的服务和 Spark 应用的资源完全隔离。

为了方便调试，本实验采用 Standalone 模式，当整个集群配置完毕后，启动

Spark-Shell，采用 Scala 交互式语言编写 WordCount 程序，返回正确结果表明整个 hadoop-spark 集群配置成功。

3.3.2 实验对比分析

(1)MapReduce 与 Spatial-Spark 空间过滤筛选对比

在相同的集群中，分别编写 MapReduce 应用程序和 Spatial-Spark 应用程序，对相同规模的空间数据进行空间数据分析。选用的数据为全国所有道路线状空间对象 (包括高速公路、国道、省道、县道)，首先对原有的 ShapeFile 数据转换成 OGC 标准的 WKT 文件，按行存储为一个空间对象，在集群中通过 Hadoop 相关命令，存放到 HDFS 中，集群中 HDFS 的 Block 的大小为 128M。空间过滤算法定义为选择其中一条道路，求解其外包矩形中包含的所有其他道路。

实验分为五组，实验数据量分别为 200M、500M、1G、2G 和 3G，结果见图3–6，当数据量不大的时候，Spatial-Spark 相对于 MapReduce 优势不够明显，当数据量达到一定程度后，Spatial-Spark 内存计算的优势突显出来，在相同的条件下，MapReduce 消耗的时间是 Spatial-Spark 的两个数量级。

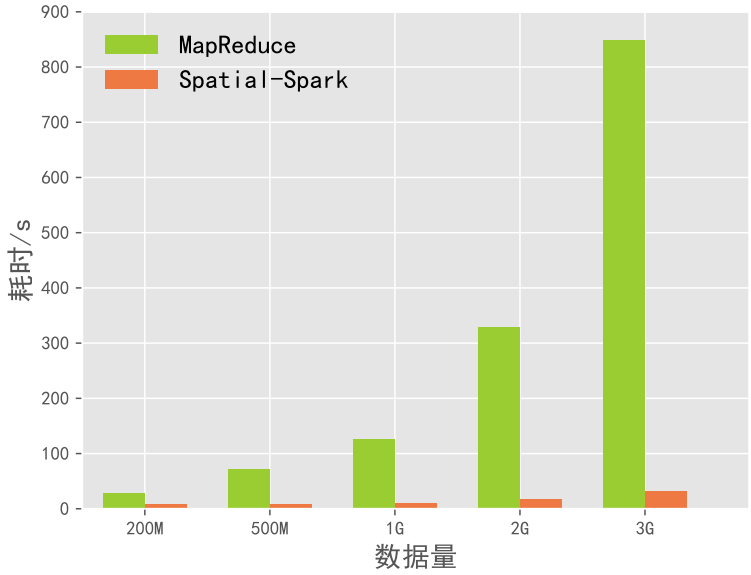

图 3–6 MapReduce 和 Spatial-Spark 查询对比

Figure 3–6 MapReduce and Spatial-Spark topology query comparsion

(2)Spatial-Spark 集群扩展性能分析

Spatial-Spark 计算框架的优越性在于其横向扩展性 (Scale Out)，通过增加廉价的计算机，使得计算性能呈现显著性增加。在空间大数据分析中也不例外，本实验通过分析动态调整工作节点的数目，比较在相同的数据规模下各个不同工作节点数目下，消耗的时间对比。

空间连接查询是空间运算中计算量较大的运算，选用的数据为全国县界面对象，共 2917 个面对象，与全国道路中进行连接操作，获取每个县与之相交的道路。

实验共分为四组，使用的 Work Node 的数量分别为 2、4、6 和 8，实验数据量 150M 县界对象，1G 的全国道路。为实验结果见图3–7，第一组实验引发 java.lang.OutOfMemoryError 异常，集群内存不足。其余实验组时间消耗大致相同。

内存大小也是影响 Spatial-Spark 运算速度中重要因素，Spark 在提交任务时候，可以通过 executor-memory 参数指定每个工作节点提供给本次计算的内存，Standalone 工作模式将会统一管理这些内存和资源调配。通过调整内存参数，分析空间链接操作耗时。

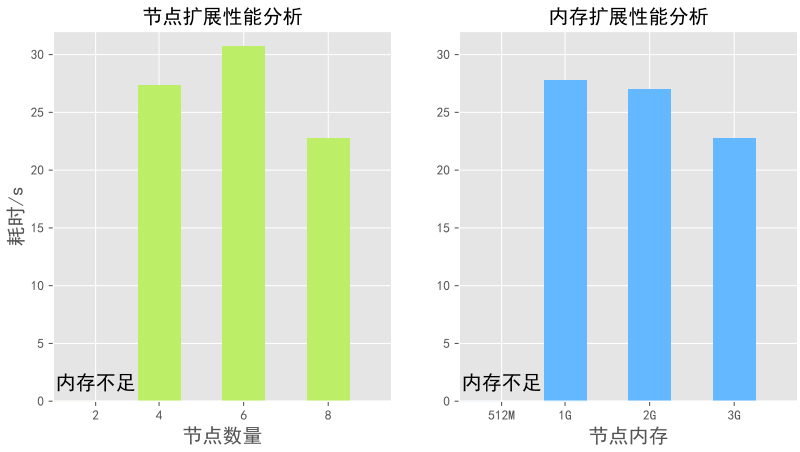

图 3–7 Spatial-Spark 扩展性实验

Figure 3–7 Spatial-Spark scale-out results

实验共四组，每一组节点的内存分别为 512M、1G、2G 和 3G，实验数据量 150M 县界对象，1G 的全国道路。实验结果见图3–7，第一组实验引发 java.lang.OutOfMemoryError 异常，集群内存不足。其余实验随着集群内存的增大，时间消耗也呈下降趋势。

(3)Spatial-Spark 空间索引性能分析

Spatial-Spark 不仅仅是 RDD 的空间拓展，而且在分布式空间计算中引入了空间索引，并将索引通过 Spark 存储策略缓存在内存中，并根据空间数据的分区，为每个分区建立了索引。由于 R 树索引存储的为空间对象的最小外包矩形，因此在相关空间分析时，需要在初步筛选后进行精确空间判断分析。

KNN 空间查询是常用的空间数据查询分析，选择全国兴趣点 (POI) 共八百多万条数据，共 2.3G，按行存储数据，包含了空间和非空间数据。

实验总共分为四组，POI 数目分别为 20W、300W、800W 和 1600W，结果见

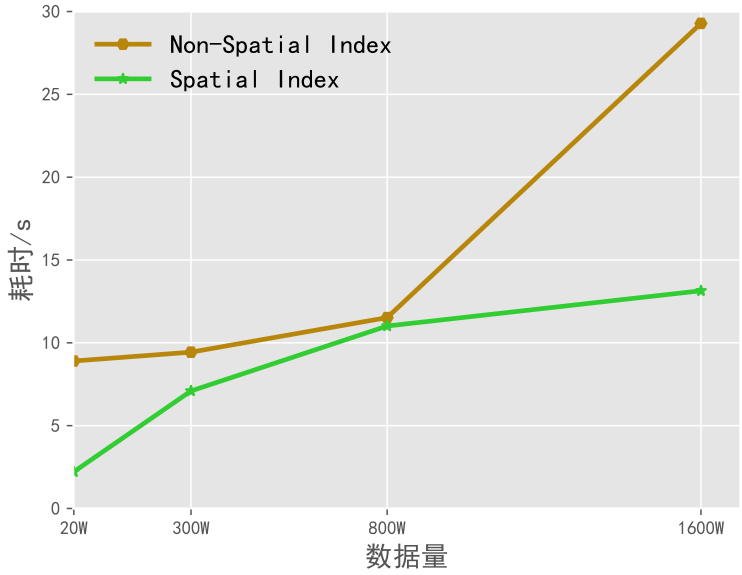

图 3–8 Spatial-Spark 索引和非索引比较

Figure 3–8 Spatial-Spark index Vs. non-index query comparsion

图3–8，对已经建立空间索引的 KNN 查询消耗时间比非空间索引少，随着数据量增加，空间索引优势越发明显。

3.3.3 Spatial-Spark 优化方案

Spark 程序都具有「内存计算」的天性，所以集群中的所有资源：CPU、网络带宽或者内存都是成为 Spark 程序性能的瓶颈。

(1) 数据序列化

序列化的作用是能够将数据在集群网络传输，因此序列化的对于提高分布式程序的性能起到重要的作用，一个不好的序列化方式将会极大地降低计算速度。因此优化对象的为提高 Spark 应用程序的第一选择[52]。Spark 提供了两种序列类库: ① Java 序列化：在默认情况下，Spark 采用 Java 的 ObjectOutputStream 序列化一个对象，只要类实现了 java.io.Serializable 接口。Java 序列化十分灵活方便，但是速度较慢；②Kryo 序列化：Spark 也能使用 Kryo 序列化对象，Kryo 不仅速度快，而且生成的结果更为紧凑。但 Kryo 序列化使用比较繁琐，需要提前注册要序列化的类。

在 Spatial-Spark 中，实验表明使用 Java 序列化对象某一 RDD 消耗为 434M 内存，当改用 Kryo 序列化后占该 RDD 消耗 53M 内存，优化效果明显。

(2) 内存优化

Spark 内存计算给大数据分析带来了便利，但针对特别大的分析数据，内存无法完整加载，RDD 持久化 API 提供了多种序列化存储级别, 见表3–6，不同的

序列化选择，使得在处理大数据时在效率和内存之间选择不同的权衡。

表 3–6 存储级别策略

Table 3–6 Storage level stragies

存储级别	说明
MEMORY_ONLY	全部序列化到内存
DISK_ONLY	全部序列化到磁盘
MEMORY_AND_DISK	序列化到内存和磁盘
MEMORY_ONLY_SER	序列化到内存字节数组

用多大内存来缓存数据是内存回收是非常重要的参数，在默认情况下，Spark 采用运行内存的 60% 空间来进行 RDD 缓存，所以在程序运行期间只有 40% 的内存可以用来创建对象。当程序运行过程中 JVM 频繁进行垃圾回收，会大大减低程序运行速度，为了提高效率，可以手动修改缓存大小比例。

3.4 本章小结 (Chapter Summary)

本章着重介绍了 Spatial-Spark 大数据空间分析框架，首先分析了矢量空间数据格式种类，空间数据转换和开源空间数据分析包；接着对 Spark 核心数据结构 RDD 空间扩展为 Spatial-RDD，并着重对空间数据进行空间分区索引，以此为基础，构建了常见空间分析应用 API。以实验为基准，分析了 Spatial-Spark 的性能方面的特点。

4 POI 空间数据分析

4 Spatial Data Analysis of Weibo POIs

新浪微博签到是新浪微博用户在使用移动终端发送微博时，使用移动终端的定位功能，用户可以选择附近热门的位置或者自行添加位置的功能[53]。这些位置信息将会被存储到新浪微博数据服务器中，在微博网页端，可以查看该签到位置的详细信息，这些签到数据称为感兴趣点 POI(Point Of Interest)[54]，见图4–1。

图 4–1 微博网页 POI

Figure 4–1 POI in web

4.1 POI 数据获取 (POI Data Fetch)

新浪微博提供了 place/nearby/pois 接口获取附近的 POI 数据，该接口参数见表4–1。

表 4–1 POI 接口参数 (部分)

Table 4–1 POI api parameters(partly)

接口	参数	类型	说明
POI	lat	Float	纬度
	long	Float	经度
	range	Int	查询半径，最大 10km

为了获取全国范围内 POI 数据，需要对全国区域内进行「地毯」式查询，而上述新浪微博 API 接口的坐标参考系统是不一致的，查询点经纬度属于椭球坐标系统 (大地坐标系统)，而查询半径属于平面坐标系统 (投影坐标系统)，因此需要针对查询坐标进行一步预处理。

(1) 投影: 地图投影多种多样，按照某种特定的要求地图投影可分为等角投影、等距投影和等积投影[55]，我国常用的投影是高斯投影，但高斯投影存在离中央经线越远，变性越大等缺点，而且需要划分较多的投影带，处理较为繁琐。

Lambert 投影属于等角圆锥投影，通过投影圆锥面割于椭球体面的两纬线，减少南北方向的变形。Lambert 投影主要用于南北方向跨度较大的地区成图，我国绝大多数省 (区) 位于中纬地区，因此采用 Lambert 正轴等角割圆锥投影，参数为中央经度为 105°E，第一标准纬度 25°N，第二标准纬度为 47°N

(2) 栅格化: 为了实现「地毯」式搜索，需要将投影后的全国边界面要素进行栅格化处理，由于新浪微博 API 在查询半径上限为 10km，为了减少查询次数，设计方案如图4–2，栅格化的大小为 $10/\sqrt{2}\times 2 = 10\sqrt{2} = 14.1\text{km}$

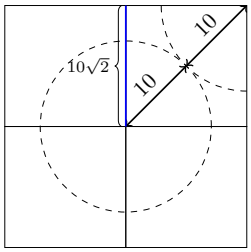

图 4–2 查询示意图

Figure 4–2 Query illustration

(3) 投影坐标反算：栅格化后，每个点坐标为投影平面坐标系，为了与新浪微博 API 接口匹配，需要将查询点的平面坐标通过 Lambert 投影反算，转换成经度和纬度 (椭球坐标)。

本文使用新浪微博 C# SDK 开发了新浪微博数据获取应用程序，启动程序后，登录新浪微博账号，程序向新浪微博授权服务器获取授权后，获得调用新浪微博 API 的权限。为了防止频繁访问服务器而导致返回不全数据或者空数据，需要在每次完成请求后线程暂停数毫秒，同时为了能够多进程并发执行，将查询点拆分为若干个文件，程序界面见图4–3，每个进程使用一份查询点文件。

这些 POI 包含了位置、类别、签到数目等数据, 见表4–2。

表 4–2 新浪微博 POI 数据 (部分)

Table 4–2 Properties of Weibo POI(partly)

属性	说明	属性	说明
PID	POI 编号	Name	POI 名称
Longitude	POI 经度	Latitude	POI 纬度
Category	POI 类别	CheckNumber	POI 签到数量

新浪微博返回的数据为 JSON 格式，表4–2中的每个属性值都是以 key−value

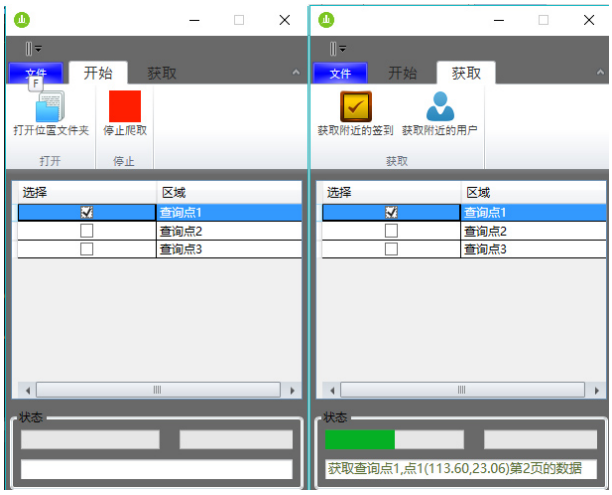

图 4–3 程序界面

Figure 4–3 Program interface

形式保存在 JSON 文本中，但是由于网络或者新浪服务器等问题，有些 JSON 记录数据不全，噪声较多，因此这些数据不适宜保存在传统的关系型数据库中，因为数据库表的结构需要在使用前定义，因此将获取的 900 多万条微博 POI，共 2.3G 数据按行存储到 HDFS 中。

4.2 统计分析 (Statistical Analysis)

Spatial-Spark 不仅仅支持空间数据，对于非空间数据处理仍然保留了 Spark 计算框架的接口，Spark RDD 提供了丰富的操作如合并 (Union)，去重 (Distinct)，过滤 (Filter) 等操作，方便开发者使用。

4.2.1 热门签到地点

每个 POI 数据都包含了签到数量，在 RDD 中使用 sortBy 和 take 操作，通过简单的排序，可以获取签到数量前 10 位的 POI, 详细内容见表4–3。

表 4–3 热门签到地点

Table 4–3 Hottest POIs in top 10

POI 名称	签到数量	POI 名称	签到数量
星光公益站	399622	丽江古城	160032
浦东机场	184263	成都双流国际机场	158121
厦门高崎国际机场	181911	望京	148109
中关村	166640	首都机场 T3 航站楼	140470
深圳宝安国际机场	162857	广州白云机场	136998

4.2.2 热门签到类别

每个 POI 数据也包含了该 POI 所属的类别，使用 RDD 的 groupBy 和 reduceByKey 两个算子，统计出每个 POI 类别的数量，以新浪微博 Logo 制作出 POI 类别词云，见图4–4，词条频数越大，词条字体越大。从图中可以看出生活娱乐类别是新浪微博 POI 类别中数量最多的，从侧面说明了新浪微博 POI 对人活动信息的反映，而这些信息对商家的广告投放、微博旅游推荐[56]有重要的参考意义。

图 4–4 POI 类别词云

Figure 4–4 Word cloud of pois' category

4.3 同位模式 (Co-Location Pattern)

频繁模式是频繁出现在数据集中的模式[57]，一般认为频繁模式中的项与项之间存在一定的相关性，同时它们之间也存在推导规则，根据一定算法可以发现事物之间关系并进行预测[58]。目前空间关联分析可以划分为单主题和多主题的两种模式，单主题空间模式发现是传统关联分析在空间维度上拓展，但是存在人工干预等缺点；多主题模式单纯地寻找在空间聚集一起的事物，也就是空间空位模式 (co-location pattern)。

4.3.1 关联规则算法

关联规则是无监督机器学习领域中研究最活跃、最广泛的一种算法[59]，通过对数据库记录进行多次扫描，就可以获得非常简单的表述形式。而且关联规则也非常容易解释，数学模型描述如下：

假设事务项 (Transaction) 为 T，每个事务项 T 都是由集合 $I = I_1, I_2, \ldots, I_m$ 组合而成，其中集合 I 中项 (Item) 各不相同。P 为项 I_i 的一个组合，如果 P 属于 T，则称事务 T 包含 P，那么关联规则的表达形式是 $P \to Q$，其中 $P \in T, Q \in T$ 并且 $P \cap Q = \emptyset$。关联 $P \to Q$ 在数据集 D 中成立，需要满足支持度 s 和置信度 c

的约束[60]。置信度是事务集 D 中包含组合 P 又包含组合 Q 的百分比；支持度是组合 P 出现的事务中，包含组合 Q 事务的百分比[61]。

在这里定义强规则是指既要不小于支持度阈值 (min_s) 又不小于置信度阈值 min_c 的规则，如果 $s(P \rightarrow Q) \geq min\_s$，则称 $P \cap Q$ 是频繁项集；如果 $c(P \rightarrow Q) \geq min\_c$，则称规则 $P \rightarrow Q$ 成立。

关联规则挖掘算法中最著名要属 Agrawal R. 和 Srikant R. 在提出的布尔规则关联频繁集合算法 (Apriori 算法)[62]，该算法的核心思想为两条：①频繁项的子集必须也是频繁的；②非频繁项的超集必定非频繁。因此 Apriori 算法主要分为两个步骤：

(1) 连接步

L_{k-1} 自相连产生 k 项集的生成 C_k，再通过 C_k 筛选出 L_k。根据 Apriori 算法第二个条件，在生成 L_k 集合过程中，可以避免一些非频繁项连接。

(2) 剪枝步

如果所有的频繁 k 项集都包含在 C_k 中，但 C_k 中的成员却不一定全是频繁的。通过 Apriori 算法的第一个条件，如果某 k 项集中存在非频繁项，那么该 $k+1$ 项必定非频繁，可将其删除做剪枝处理。

4.3.2 单主题空间关联规则

Koperski K. 将传统的关联规则挖掘拓展至空间关联规则挖掘，单主题空间关联规则研究得到了广泛的关注和研究。单主题的空间关联规则挖掘以某一类空间实体展开[63]，发现周边其他类别的空间实体与其关系。

实体之间的关系可分为非空间谓词和空间谓词，非空间谓词表示空间对象的非空间的性质的谓词，如价格、种类、颜色等；而空间谓词则表达空间对象由于空间对象位置而形成的相互之间的联系，一般来讲分为空间拓扑关系、空间距离关系和空间方位关系。典型的的空间关联规则如式(4-1) 所示，其中 P_i 和 Q_j 中至少有一个为空间谓词。

$$P_1 \wedge P_2 \wedge \ldots \wedge P_m \rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n(s\%, c\%) \tag{4-1}$$

当空间对象分布不均匀的时候，单主题空间关联规则会出现关注对象不突出，不利于反映真实的空间关联性。改进的算法有基于 k 邻近的空间关联模式挖掘[64]，该算法优化了空间关系选择方案，选择空间 k 个近邻对象，然后建立空间关系事务项，使用经典的关联规则算法进行数据挖掘。该算法呢通过选择 k 个近邻空间对象，避免的空间对象分布不均匀的问题，但是该算法需要指定 k 值，k 值过小，空间关系表现非常匮乏，而 k 值过大，将会导致空间事务项阶数激增，关联关系噪声较大[65]。

4.3.3 多主题同位模式

传统的数据通常是相互独立的，而空间上分布的对象则是相关的，也就是空间并置 (co-located)，即两个对象位置越近，就越有可能有相似的性质或者有较强的相关性。Shekhar S. 等人提出了基于空间相关的同位模式分析[66]，即主题的空间关联规则。该模型将事务概念泛化，以一定距离领域范围包含的空间对象作为空间关联规则的事务，提出空间实体分布的同位模式，并且很好地考虑到空间实体之间的相关性[67]。

一般来讲，距离越近的空间对象越具有相关性，因此同位模式选择特定的距离作为判断空间对象关联依据，并且将其以事务表项保存。以图4–5为例，空间范围内有个 A, B, C 三类事物，包含了 $a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, b_5, c_1, c_2$和$c_3$ 空间实体对象，空间邻近关系 R 定义为两个空间实体对象之间的距离不大于距离 d，见式(4-2)，如果满足空间邻近关系 R 则将空间对象连接起来。

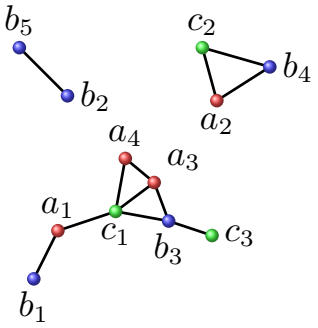

图 4–5 空间关系示意图

Figure 4–5 Spatial relation illustration

$$R(a_1, b_1) \Leftrightarrow (distance(a_1, b_1) \leq d) \tag{4-2}$$

空间特征集代表了空间不同种类对象的集合，记为 $F = \{f_1, f_2, \ldots, f_n\}$；每一个具体空间位置上的对象称之为空间实例；空间实例集 $I = \{I_1, I_2, \ldots, I_m\}$，如果有 $\{R(I_j, I_k) | 1 \leq j \leq m, 1 \leq k \leq m\}$，称 I 为一个簇；空间 co-location 模式是一组空间特征的集合 C，其中 $C \in F$，一个 co-location 模式 C 的长度称此 co-location 的阶 (Order)；如果簇 I 的包含了 co-location 模式 C 中的所有特征，并且 I 中没有任何一个子集可以包含 C 中所有特征，那么称 I 就是 co-location 模式 C 的一个行实例 $row\_instance(C)$，co-location 的所有行实例表示为一个表实例 $table\_instance(C)$。

在图4–5中，空间特征集为 $F = (A, B, C)$，而$(c_2, b_4, a_2), (a_3, c_1, b_3)$，$(a_1, b_1)$，$(a_1, c_1)$，$(c_1, a_4)$和$(b_3, c_3)$ 表示为一个簇。(A, B, C) 可以表示为一个 co-location 模式，(a_2, b_4, c_2)和(a_3, c_1, b_3) 表示了该 co-location 的行实例，所有的行实例将组成表实例。

参与率和参与度是衡量 co-location 模式挖掘中重要参数，反映了一个空间模式的频繁程度。

(1) 参与率

设 f_i 为某个空间特征，f_i 在 k 阶 co-location 模式 C 中的参与率 (participation ratio) 表示为 $PR(c, f_i)$，它是 f_i 的实例在空间 co-location 模式 C 的所有不重复实例中不重复出现的个数与总实例的比例，定义见式(4-3)。

$$PR(c, f_i) = \frac{|\pi_{f_i}(table\_instance(c))|}{|table\_instance(\{f_i\})|} \tag{4-3}$$

例如图4–5中，同位模式 (A, B, C) 所有的实例为 (a_2, b_4, c_2)和(a_3, c_1, b_3)，空间特征 A 的实例对象有 2 个 (a_2, a_3)，而空间特征 A 的所有实例有 4 个 (a_1, a_2, a_3, a_4)，所以 $PR(\{A, B, C\}, A) = 2/4 = 0.5$；同理 $PR(\{A, B, C\}, B) = 2/5 = 0.4$，$PR(\{A, B, C\}, C) = 2/3 = 0.67$。

(2) 参与度

co-location 模式 $c = \{f_1, f_2, \ldots, f_k\}$ 的参与度 (Participation Index) 表示为 $PI(c)$，它是 co-location 模式 C 的所有空间特征的 PR 值中的最小值，定义见式(4-4)

$$PI(c) = min_{i=1}^{k}\{PR(c, f_i)\} \tag{4-4}$$

图4–5中，$PI(\{A, B, C\}) = min(0.5, 0.4, 0.67) = 0.4$。

min_prev 是需要给定的最小参与度阈值，只有当 $PI(c) \geq min\_prev$ 时，称 co-location 模式 C 是频繁的。co-location 算法主要是基于最小参与度，由于最小参与度概念与 Apriori 算法相同的性质 (向下闭合)，很大程度上能降低算法复杂度的开销，此类算法研究主要有：

①Join-Based 算法：一种基于完全连接的算法。该算法严格使用 Apriori 算法，k 阶模式表之间通过连接 (Join) 操作生成 $k+1$ 阶候选者，对 $k+1$ 表实例按照最小参与度筛选，该算法能够产生完整的 co-location 模式。

②Partition-Join 算法：一种基于部分连接的算法。该算法采用分治的思想，首先把空间所有按照距离阈值划分为不相交的实体块，块与块之间实体不满足空间关联性，这样只需要进行块内部空间对象连接，大大降低了连接的计算量[68]。算法的关键之处如何很好划分出空间实体块，降低复杂度。

③CPI-Tree 算法：一种基于前缀树结构的无连接改进算法[69]。该算法以前缀树表达空间实体对象之间的邻近关系，通过树结构可以快速地处理表实例的增长，拥有较高的性能。但是如果数据量较大，存储和遍历树的代价将会增加，是一种折中的解决方案。

4.3.4 Spatial-Spark 同位模式

全连接算法是一个依据先验原理，基于 Join-Based 操作产生候选模式和表实例的方法。主要流程分为以下两步:

(1) 候选行实例生成

连接两个 $k-1$ 个空间对象相同的频繁 k 阶模式实例表 P_{k1} 和 P_{k2}，生成 $k+1$ 阶的候选行实例，在生成的过程中，检查行实例的所有对象是否满足空间距离阈值 d。

(2) 表实例的生成

根据所有候选行实例，按照每个行实例的模式进行汇总形成候选表实例，对每个表实例计算所有特征的参与度，按照参与度过滤不满足阈值的表实例。

算法过程中，首先需要进行二阶模式行实例和表实例生成，算法时间复杂度将达到 $O(n^2)$，当数据量急剧增加时候，算法的时间消耗是不可接受的，需要进行并行化处理。借鉴 Spatial-Spark 计算框架中的空间连接运算，根据空间同位模式的距离阈值 d，将整个平面空间位置划分为均匀网格。以空间对象落入网格的横纵坐标编号作为空间实体的 Key，将所有空间实体生成 Key-Value 形式，见图4–6中 I 所示，每个空间实体对象拥有唯一的坐标。

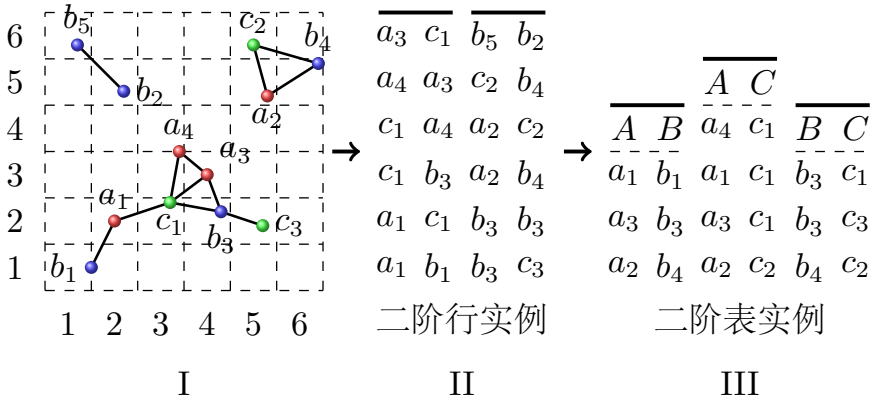

图 4–6 二阶模式生成

Figure 4–6 Generation of 2 order location pattern

紧接着对所有实体对使用 cogroup 算子进行自相连，生成所有满足距离阈值的空间实体对，为了完整得获取所有空间邻近关系，对坐横纵坐标差值绝对值不大于 1 的空间实体可判定为在可能存在邻近关系，图4–7中 I, II 和 III 分别表示了三种相等情况。再使用 filter 算子进行过滤判断严格的邻近关系，经过 mapToPair 操作生成所有的行实例，最后使用 reduce 操作按空间参与度过滤生成表实例。

图4–6展示了完整的二阶生成模式流程，II 为列出所有行实例，III 则按照空间实体的特征进行汇总，按照参与度进行剪枝操作。生成二阶模式后，根据同位

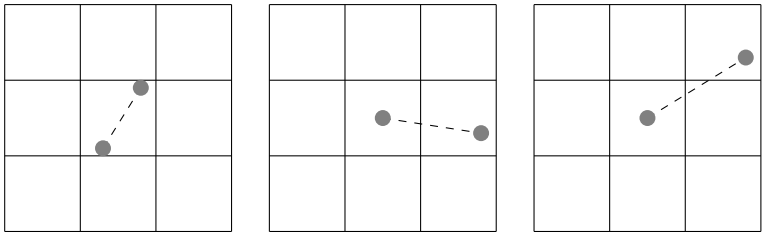

图 4–7 坐标相等判断

Figure 4–7 Judgement of coordinate equality

模式的连接和剪枝步骤，生成更高阶数同位模式，见算法4-2。

算法 4-2 co-location 算法

输入:

点集:points;

距离阈值:d;

参与度阈值:threshold;

输出:

空间同位模式：colocation patterns;

1: P_2 = Spatial-Spark.join(points, d, threshold)
2: **while** P_k is not empty and k < N **do**
3: C_{k+1} = gen_candinate_cocolation(P_k,k) //k+1 阶候选集合
4: C_{k+1} = pruning(C_{k+1}, d) // 剪枝处理
5: T_{k+1} = gen_table_ins()// 生成表实例
6: P_{k+1} = Select_Colocation_Pattern(T_{k+1}, threshold) // 筛选表实例
7: $k = k + 1$ // 下一轮迭代
8: **end while**
9: **return** $P_2,\ldots,P_k$

为了将不同的表实例进行连接操作，需要重写 Join 操作中 key 相等判断操作，定制了 RowPattern 类，重写 equal 方法，当且仅当只有一个空间实例不等的时候，该行实例才相等。

RDD 提供了 Join 操作，该操作对每个 key 下元素进行笛卡尔乘积，返回的结果再展平。当对 k 阶行实例进行 join 操作后，对新加入的新添加的 POI 对象进行距离阈值 d 判断，获得有效的 $k+1$ 阶行实例。以行实例的 POI 的类别 Pattern 为 key 进行 reduce 操作生成表实例，按照参与率阈值进行剪枝操作，生成 $k+1$ 阶表实例。设 k 阶模式 $C1$ 和 k 阶模式 $C2$ 连接得到 $k+1$ 阶候选模式 $c3$，$C1$ 的行实例个数为 $K1$，$C2$ 的行实例的个数为 $K2$，$C3$ 的行实例个数为 $K3$，在算法过程中对 $C1$ 和 $C2$ 表进行连接操作，之后再 $C3$ 表中进行查找，整个算法时间复杂度 $O((k-2)K1 \times K2 \times K3)$，实现过程如下。

```
while(colocations.count!=0){
JavaPairRDD<RowPattern,Iterator<POI>>
        rawRowInstances=colocations.join(colocations);
  JavaPairRDD<RowPattern,Iterator<POI>>
      validRowInstances=rawRowInstances.filter(
        //距离阈值判断
        distanceThresholdCheck();
      )
  JavaPairRDD<Pattern,Iterator<Iterator<POI>>>
      validTableInstance = validRowInstances.reduceBykey(
        //按pattern合并行实例
        combineRowInstances();
      ). filter (
        //按照参与度进行剪枝
        participateratioCheck();
      );
  colocations=validTableInstance.mapToPair(
      //展开下一阶行实例
      flatRowInstances();
  )
}
```

4.4 微博 POI 同位模式 (Co-Location Patterns in Weibo POIs)

微博用户签到地点是非常有趣的数据，反映了新浪微博用户在线下的活动情况，对现实生活的商圈发现、地点推荐等重要的参考依据。POI 空间模式挖掘中，为了过滤掉噪声信息，需要经过两步预处理：①过滤签到数量较少的 POI，选择签到数量大于 10 的 POI 点；②过滤 POI 所属类别不清楚，如「一般地点」、「楼宇」和「大门」等。

4.4.1 二阶同位模式识别

以不同的城市为研究区域，探索不同城市微博 POI 空间模式分布特征。选择空间距离阈值 $d = 500\text{m}$，上海、武汉和重庆三市微博 POI 类别二阶特征模式。由于新浪微博 POI 类别分类并非严格区分，比如「餐饮美食」与「中餐厅」存在着包含关系，因此我们选择某一类别进行重点关注，比如选择「高等院校」在新浪微博 POI 中的与其他类别的同位关系及其空间参与度，见表4–4。

从表4–4可以看出上海、武汉和重庆三市高等院校与周边同位模式 POI 类别差别不大，都是与高校配套的相关服务类型，但是每个城市高等院校的周边类别差异较大。由于上述三城市高等院校数量依次减少，因此相应的空间模式的空间

表 4–4 高等院校二阶同位模式

Table 4–4 Advance academics' 2-order patterns

上海市	武汉市	重庆市
(高等院校, 校园生活):0.556	(高等院校, 校园生活):0.531	(高等院校, 校园生活):0.306
(高等院校, 日本料理):0.472	(高等院校, 图书馆):0.528	(高等院校,ATM):0.238
(高等院校, 西餐厅):0.454	(高等院校, 糕饼店):0.384	(高等院校, 科研机构):0.224
(高等院校, 甜品店):0.451	(高等院校, 快餐厅):0.368	(高等院校, 图书馆):0.224
(高等院校, 培训机构):0.446	(高等院校, 四川菜):0.361	(高等院校, 超市):0.217
(高等院校, 医院):0.443	(高等院校, 火锅店):0.358	(高等院校, 特色餐厅):0.215
(高等院校, 糕饼店):0.441	(高等院校, 连锁酒店):0.354	(高等院校, 电子卖场):0.210
(高等院校, 餐饮美食):0.432	(高等院校, 医院):0.328	(高等院校,KTV):0.205

参与度呈现下降趋势。

以 (高等院校, 培训机构) 二阶模式为例，研究在不同空间距离阈值条件下在上述三市中空间参与度分布情况，见图4–8，随着距离阈值增大，同位模式的空间参与度逐渐上升，而不不同城市之间差异也非常明显。

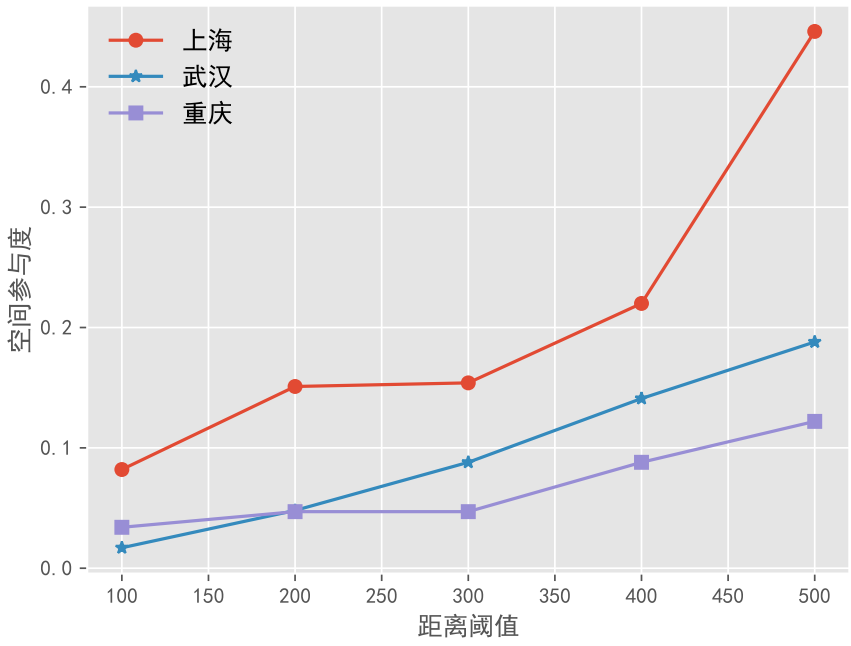

图 4–8 (高等院校, 培训机构) 模式不同距离阈值在不同城市空间参与度

Figure 4–8 (Academic institutions & Training institutions) Pattern's Paticipation indexes in different distances and cities

4.4.2 高阶同位模式识别

北京市是全国新浪微博 POI 数量最多的城市，对该城市的新浪微博 POI 进行空间同位模式更具有代表性。在空间同位模式挖掘算法中，连接距离阈值 d 和参与度阈值 threshold 是两个非常重要的，不同的参数将生成不同的空间同位模式, 因此不同参数对北京市的 POI 进行空间同位模式挖掘结果见表4–5。

表 4–5 不同距离和参与率空间同位模式

Table 4–5 Co-location patterns in defferent distances and PI's thresholds

距离阈值	参与率阈值	2 阶	3 阶	4 阶	5 阶	6 阶
300	0.5	9	4	1	-	-
	0.6	8	2	-	-	-
	0.7	5	1	-	-	-
400	0.5	23	22	13	3	-
	0.6	17	9	2	-	-
	0.7	11	3	-	-	-
500	0.5	40	40	26	11	2
	0.6	29	29	19	7	1
	0.7	22	17	7	1	-

通常来讲，距离阈值越大，参与率阈值越小，空间模式的生成阶越大，每一阶的模式数目越多。本文选择 $d = 500\text{m}$，参与度阈值为 0.6，该参数既可以保留足够的细节信息，又能较好地反应空间分布的整体特征[70]，新浪微博微博 POI 空间同位模式见表4–6。

通过对北京市新浪微博 POI 数据同位模式挖掘可以发现，同位模式阶数越高，POI 模式越呈现商业模式，根据同位模式的「闭合性」，五阶模式中的 (电影院,KTV, 咖啡厅, 美容美发店)，六阶模式中的 (KTV，中餐厅，咖啡厅，甜品店，美容美发店，酒吧) 在所有低阶模式中全部出现，加上新浪微博 POI 是用户手动添加，这些地点在实体空间聚集性也反应了用户的集聚行为，因而这些对商业广告投放、商业选址和商业促销等活动有很好的建议。

4.5 本章小结 (Chapter Summary)

本章以新浪微博 POI 为研究重点，首先分析了如何快速地使用新浪微博 API 获取这些数据；然后使用统计分析，得到了 POI 在签到数量和按类别统计信息，最后重点分析了如何使用 Spatial-Spark 对新浪微博 POI 数据同位模式挖掘并行化设计，研究了上海、武汉和重庆二阶同位模式的特征，以北京为研究对象通过设定距离阈值和参与度阈值，获取了该城市新浪微博 POI 空间同位模式，并得

表 4–6 POI 空间同位模式

Table 4–6 Co-location patterns of POIs

阶数	实例
二阶	(中餐厅, 校园生活)，(校园生活, 医院)，(甜品店, 中餐厅)，(咖啡厅, 校园生活)，(咖啡厅, 酒吧)，(清真餐馆, 酒吧)，(电影院, 甜品店)，(旅馆招待所, 咖啡厅)...
三阶	(中餐厅, 校园生活, 医院)，(电影院,KTV, 美容美发店)，(甜品店, 中餐厅, 美容美发店)，(美容美发店, 酒吧, 甜品店) (中餐厅, 校园生活, 咖啡厅)...
四阶	(电影院,KTV, 咖啡厅, 美容美发店)，(KTV, 中餐厅, 甜品店, 美容美发店)，(甜品店, 中餐厅, 咖啡厅, 美容美发店)，(甜品店, 咖啡厅, 美容美发店, 酒吧)...
五阶	(电影院,KTV, 咖啡厅, 甜品店, 美容美发店)，(KTV, 中餐厅, 咖啡厅, 美容美发店, 酒吧)，(甜品店, 中餐厅, 咖啡厅, 美容美发店, 酒吧)...
六阶	(KTV, 中餐厅, 咖啡厅, 甜品店, 美容美发店, 酒吧)

到一些结果，这些对商业决策有重要的参考意义。

5 人口空间流动网络分析

5 Analysis of Population Spatial Floating Network

人口流动是直接塑造城市间关系的重要动力，也是城市发展活力指标[71]。城市之间的人口流动研究始终是人文地理学、城市地理学和经济地理学所关注的热点。现代社会发展，交通网络的日趋发达将城市发展与人口流动紧密地联系在一起，人口活动不再局限于一个封闭的城市。从全球、国家或区域层面来看，人口流动已经跨越了地理的限制。传统的人口流动的统计方法费时费力，社交网络应用使用 LBS 功能为研究这列问题提供了新思路，通过新浪微博提供的 API 接口获取特定时间段内全国范围内发送微博的用户，根据用户的账户位置信息和发送微博的位置信息，构建基于社交地理的人口流动网络，如图5–1所示。

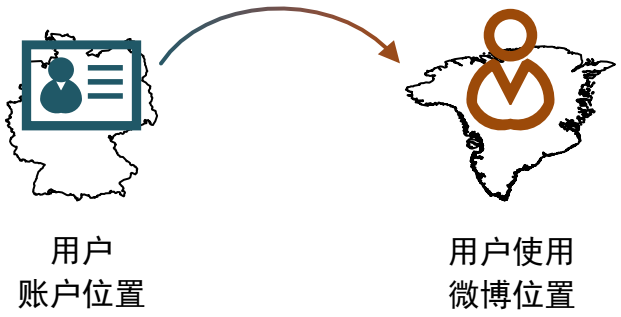

图 5–1 流动示意图

Figure 5–1 Floating illustration

5.1 数据获取 (User Data Fetch)

新浪微博提供了 place/nearby/users 接口来获取附近发微博的用户，接口参数见表5–1。

这些新浪微博用户数据包含了用户的名称、账号位置和用户当时所在空间位置，部分属性见格式见表5–2。

春节前后是全国人口流动最为频繁的时间段，也是最能反映全国人口流动情况。因此选择 2016 年春节前后 1 月 10 日和 2 月 10 日为起讫时间，获取全国微博用户位置数据，共 4.3G，约 1500 万条用户记录。返回的数据为 JSON 格式，以键值的形式保存表5–2中的属性，由于网络等其他问题，当请求量大的时候 JSON 数据中噪声较多，为了方便处理这些数据因此将数据按行存放到 HDFS 中。

人口在不同城市之间流动构成了人口流动网络图，城市相当于网络图中节

表 5–1 NearbyUsers 接口参数

Table 5–1 Nearby users api parameters

参数	类型	说明
lat	Float	纬度
long	Float	经度
starttime	Int64	开始时间
endtime	Int64	结束时间
range	Int	查询半径，最大 10km

表 5–2 用户地理空间位置 (部分)

Table 5–2 Users' location information(partly)

属性	说明	属性	说明
ID	用户新浪微博 ID	Name	用户新浪微博名称
Province	用户账户省份	City	用户账户城市
Lon	用户位置经度	Lat	用户位置纬度

点，不同城市之间人口流动相当于图中顶点之间的有向连接边，而人口流动数量则相当于边的权重。

GraphX 作为 Spark 上层图处理模块，提供了并行化图分析算法。为了方便分析城市人口流动分析，使用 Spatial-Spark 空间查询将全国数据抽象为一个 Graph 对象，其中边 E_{ij} 为的权重为城市 j 中城市 i 用户的数量，算法流程图见5–2。

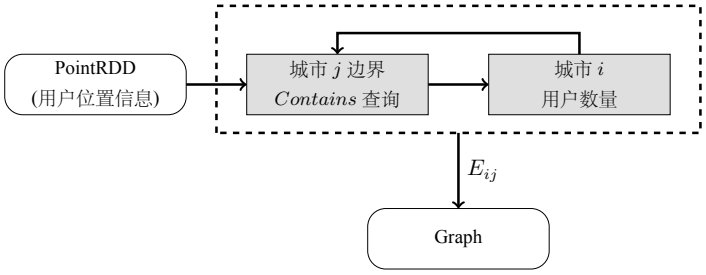

图 5–2 Spatial-Spark 生成人口流动网络图

Figure 5–2 Generation floating population graph using Spatial-Spark

5.2 人口流动统计 (Statistic of Floating Population)

5.2.1 流动指数

为了定量分析人口流动情况，定义如下指数:

流入量: 账户为其他城市在 i 城市发送微博的微博用户量，用 δ_i 表示;

流出量: 账户为 i 城市在其他城市发送微博的微博用户量，用 ω_i 表示;

流入流出比： 流入量/流出量，$\psi_i = \delta_i / \omega_i$。

5.2.2 并行化设计

GraphX 丰富的图操作运算中[72]，其中 aggreateMessages 是最核心最强大的接口，该方法将边和其关联两个顶点作为一个 Triplet 进行整体并行处理，如图5–3所示，节点 A 和 B 以及它们之间的有向边组成一个 Triplet。首先进行类似 map 操作，将每个 Triplet 生成相应的消息，发送关联的顶点，可以选择发送至目的节点，也可以选择发送至源节点；然后对发送到同一个节点的消息进行 reduce 操作，进行合并操作。

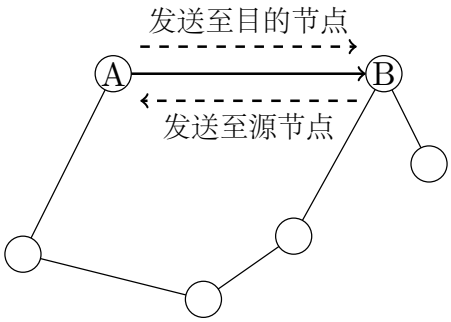

图 5–3 Triplet 操作示意图

Figure 5–3 Triplet operations illustration

在计算城市人口流入量的时候，aggregateMessages 接口中 map 操作选择将消息发送给有向边的目的节点；在计算城市人口流出量的时候，map 操作选择将消息发送给有向边的源节点，在 reduce 阶段所有消息汇总。将上述两步生成的结果进行 join 和 mapValue 操作，计算每个城市的人口流动流出比，算法流程见图5–4。

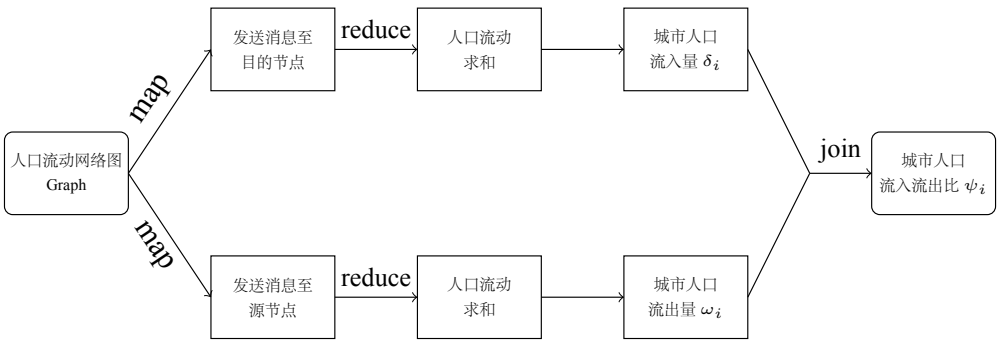

图 5–4 人口流动统计流程图

Figure 5–4 Flow chart of population statistic

5.2.3 统计分析

以全国地级市和直辖市边界为空间查询范围，使用 aggerateMessages 函数分别计算出各自的流入量 δ_i、流出量 ω_i 和流入流出比 ψ_i，结果栅格化渲染结果见图5–5。

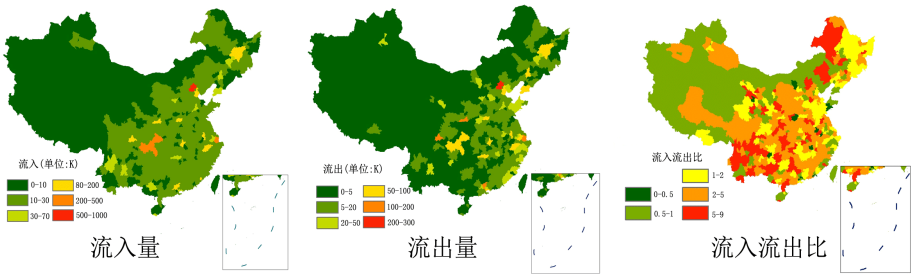

图 5–5 流出、流出和流入流出比

Figure 5–5 Flow-in,flow-out and flow-ratio

统计各个城市的人口流动数目，发现成都、西安、北京、广州、郑州等地为全国人口流动集中城市[73]。并且人口流动量前 20% 的城市占全国总用户的 70.1%，符合人口统计学和社会科学的「80/20 法则」，即 80% 的事物或现象集中在前 20% 的研究对象中。将前 20% 城市人口流动统计见图5–6，可以看出人口流动流动量符合幂律分布 (Power law) 分布。

从图5–5看出城市流入流出比呈现较多的复杂性：ψ_i 远大于 1 表明该城市在春节期间人口呈现流入趋势；ψ_i 远小于 1 表明该城市在春节期间人口呈现流出趋势；而接近于 1 表明该城市人口在春节期间流动量持平，主要城市的 ψ_i 见表5–3。

表 5–3 城市流入流出比

Table 5–3 Cities' flow ratio

流入流出比 ψ_i	说明	城市
大于 1	流入	辽源、鄂州、西双版纳、毕节、永州、七台河、娄底、姿阳、白色、黄冈、松原、三亚、娄底、大理等
接近 1	持平	上海、北京、成都、武汉、哈尔滨、嘉兴、杭州、长沙，广州、西安、太原、郑州、沈阳、乌鲁木齐等
小于 1	流出	东莞、马鞍山、芜湖、青岛、苏州、深圳、包头、宁波、无锡、唐山，石家庄、呼和浩特等

流入流出比远大于 1 的城市主要有：①春节返乡人群所在的城市，主要分布在人口劳动力输出的中西部城市，四川、贵州、湖南、湖北等省份；剩余部分城市位于东北地区。②南方著名旅游城市，海南，云南等省份。随着旅游等第三

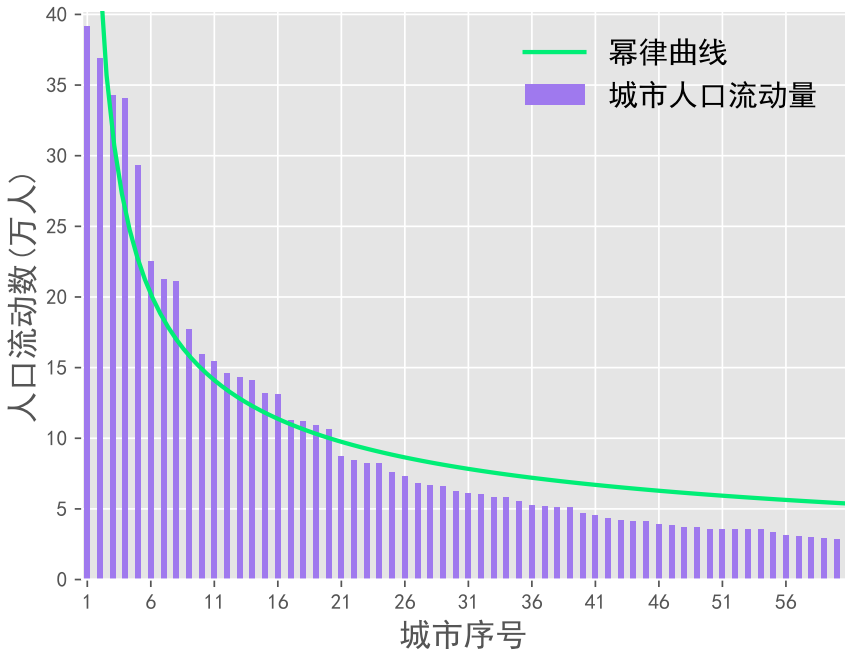

图 5–6 人口流动统计图

Figure 5–6 Population flow statistic of cities

产业发展，越来越多的人选择春节假期去度假，而不是回乡团聚，这些使得旅游城市在春节期间迎来较大的人流量。

流入流出比接近 1 的城市春节期间人口流入量与流出量大致相同，这些城市主要是某个区域的连接中心，一般为省会城市，如上海，北京，成都，武汉等等。

流入流出比小于 1 的城市主要有：①沿海以劳动密集型产业发展起来的移民城市，集中在长三角和珠三角地区，据公安部最新数据表明，深圳市常住人口超过 1300 万，但本地人口只有 250 万，外来人口超过 80%，位列全国第一，苏州以 50% 外来人口排名第二。②重工业城市，其中以钢铁为代表重工业发展起来的城市如包头 (包钢)，唐山 (首钢)，马鞍山 (马钢) 等，人口的流出可能与目前这些钢铁产业发达的城市产能过剩相关[74]。

对人口流入流出比较大的城市，对这些城市交通、旅游等部门提供了决策依据，如航班的规划、旅游服务部门提前做好准备工作等等；也可以通过从中人口的流入量，窥探城市经济走向和市场的活力。

5.3 人口流向分析 (Analysis of Floating Population Route)

全国城市之间流动网络见图5–7，从图中可以看出人口流向存在着不均衡性，明显的东部、中部和西部阶梯之分，东部地区城市之间流动量大，流动密度强，

中部地区次之而西部地区最小。

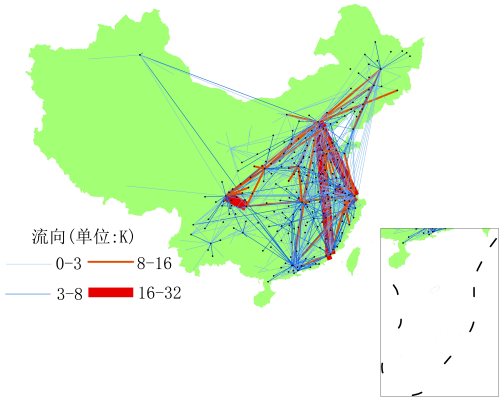

图 5–7 城市人口流向

Figure 5–7 Cities' population flow

5.3.1 PageRank 算法

PageRank 算法是 Google 搜索引擎的核心技术[75]，该算法的核心思想是将整个互联网抽象成网络图，每个网页抽象成一个节点，而节点之间的 URL 连接代表了网络图的边。被许多重要页面引用的页面，往往还是重要的页面。算法开始假设所有节点拥有相同的权重，根据节点的之间的链接权重设计出转移矩阵，通过权重转移矩阵相乘，重新分配权重，不停迭代循环，直至节点权重收敛。

以人口流动为例，将城市抽象成有向网络图中的各个节点，人口的流动抽象成节点之间的连通路径，构建出每个城市之间的流动矩阵 M:

$$M = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \dots & \dots & \ddots & \dots \\ \alpha_{1n} & \alpha_{2n} & \dots & \alpha_{nn} \end{bmatrix}$$

式中 α_{ij} 为 i 城市人口流动至 j 城市的概率:

$$\alpha_{ij} = \begin{cases} \frac{\beta_{ji}}{\sum_{k=0}^{k=n,k\neq i} \beta_{ki}} & i \neq j \\ 0 & i = j \end{cases} \tag{5-1}$$

式(5-1)中 β_{ki} 为在 k 城市微博用户中，账户位置为 i 城市的用户数量。假设初始城市权重是相同的，即:

$$V_0 = \begin{bmatrix} 1/n & 1/n & \dots & 1/n \end{bmatrix}^T \tag{5-2}$$

通过 n 次迭代 $V_n = M \times V_{n-1}$，直至 V_i 收敛，计算出每个城市在流动网络中的权重。

5.3.2 城市流动网络权重

GraphX 模块中 Graph 类实现了 PageRank 算法，分为两种实现方式：一种是给定迭代次数的静态方法；另一种是给定迭代收敛阈值的动态方法。本文选择动态方法，给定收敛阈值 0.0001，计算得到权重前 20 位的城市见表5–4。

表 5–4 城市 Pagerank 排名

Table 5–4 Cities' Pagerank rank

序号	城市	PageRank 值	序号	城市	PageRank 值
1	北京	0.058	11	厦门	0.014
2	成都	0.033	12	哈尔滨	0.013
3	上海	0.031	13	南京	0.012
4	西安	0.028	14	长沙	0.012
5	广州	0.026	15	沈阳	0.012
6	武汉	0.024	16	天津	0.011
7	杭州	0.017	17	长春	0.010
8	郑州	0.016	18	大连	0.009
9	深圳	0.015	19	济南	0.009
10	重庆	0.014	20	太原	0.008

以国家统计局 2015 年全国各直辖市、地级市和自治州的 GDP 数据，与城市人口流动网络权重进行相关性分析。发现相关系数达 0.8，表明呈现正相关关系，绘制散点图如图5–8左所示。为了更好的进行回归分析，将权重的平方根作为应变量，城市 GDP 数值作为响应变量，进行线性回归分析，结果如图5–8右所示，回归方程为:$\mathrm{GDP} = 1.037 \times 10^5 rank^{0.5} - 2669$，回归系数 P-value < 0.001，通过假设检验。

从图5–8中可以看出，并不是所有的点都很好的拟合回归曲线，存在「异常」点。杠杆值可以描述点对整体回归的影响，高杠杆值可以说明改点属于异常点。据此绘制上述回归分析的残差平方与杠杆值散点图，见图5–9，图中右下方和左上方的点属于异常点。

城市特定时间段内在人口流动网络中的权重与实体城市权重存在明显的关系，出现了以下特征：

(1) 城市权重与城市经济发展相对一致性。城市的权重与城市 GDP 呈现一定的正相关关系，即城市的权重越高，城市的 GDP 数值会高。图5–9中也表明了在相关关系中异常点，主要分为两种：①北京、成都、广州和武汉等城市权重超出该城市 GDP 数值，其中北京最为明显；②上海、西安、天津、苏州和深圳市等城市的 GDP 发展远远大于该城市权重。

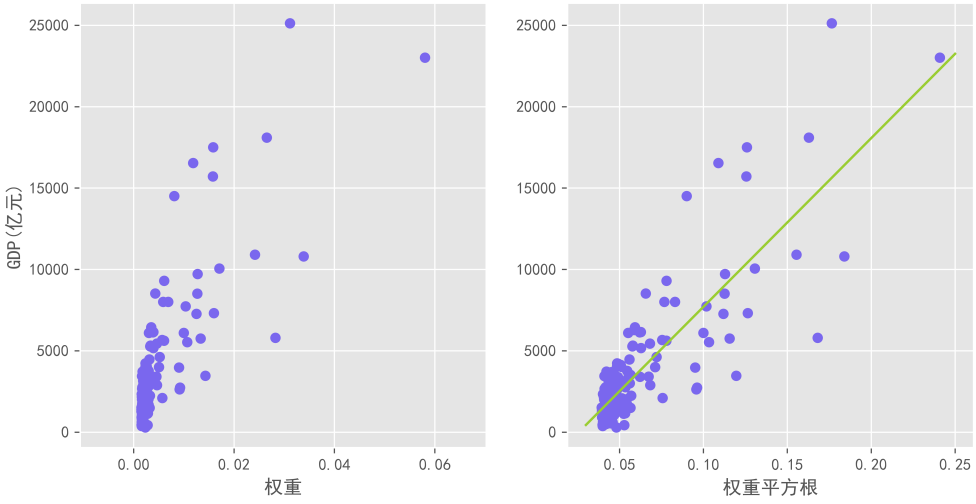

图 5–8 权重和 GDP 相关性分析

Figure 5–8 Ranks & GDP correlation analysis

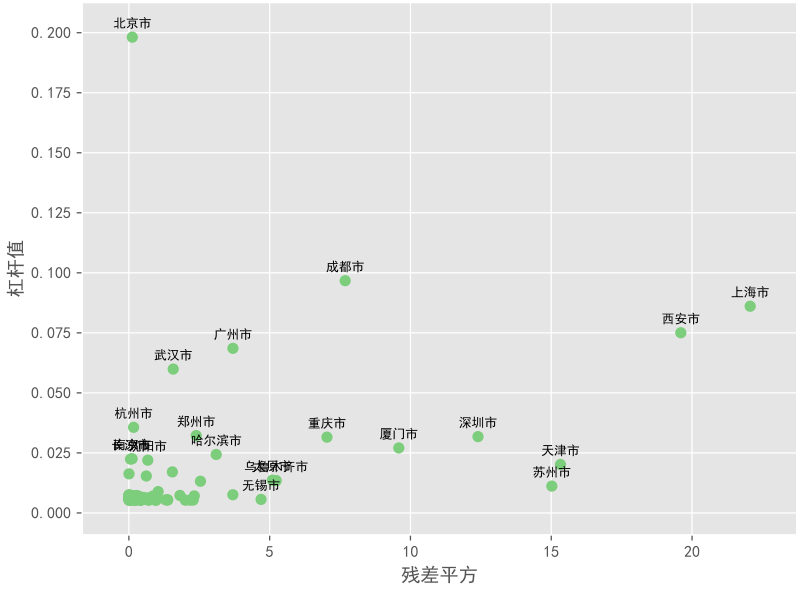

图 5–9 残差平方与杠杆值散点图

Figure 5–9 Residual square & leverage scatter plot

(2) 城市权重在流动网络层级分布。通过对各个城市 PageRank 值分层，北京作为第一层级，是全国人口流动网络的中心；成都、上海、西安、广州和武汉为第二个层级，是全国人口流动网络节点的副中心；杭州、郑州、深圳、重庆、厦门等为第三个层级，是区域人口流动网络的中心；哈尔滨、南京、长沙、沈阳和天津等为第四个层级，是地区人口流动网络的中心。

(3) 东部城市与中西部城市权重的差异明显。城市权重也符合「80/20 法则」，排名前 20 城市权重和 0.81。按照东部、中部、西部三大地区划分，东部地区城市权重和 0.51，中部地区城市权重和 0.32，西部地区城市权重和 0.17。虽然中部、西部地区也存在着权重较高的城市，但东部区域在城市权重平均水平上明显高于中部和西部，因此东部城市在人口流动网络中扮演重要的角色。

5.4 人口流动网络社群挖掘 (Community Mining of Floating Population Network)

现实世界中诸多系统都是以网络的形式存在，如人际关系网、机构协作网络、万维网等，这些统称为复杂网络 (Complex Network)，可以通过简单的观测可以发现小规模网络的的特征、内部特征和划分，但是对于大规模和超大规模的网络，直观的显示很难通过肉眼发现网络的特征和行为，因此需要使用复杂网络分析手段进行处理[76]。

网络社群挖掘 (Community Mining，CM) 是复杂网络聚类中研究最多的方法，社群挖掘方法对分析复杂网络中的拓扑结构，理解复杂网络中隐藏的规律和预测复杂网络的行为不仅有重要的理论意义，而且有广泛的应用前景。城市之间人口流动也是属于一种复杂网络，对该网络进行社群挖掘能够有效的发现城市在人口流动的联系，反映全国城市人口流动聚集效应。

5.4.1 社群挖掘算法

社群挖掘作为网络研究的经典问题，一直受到学者的广泛关注，目前社群挖掘算法主要有：GN(Girvan Newman) 算法，定义边的介数，依次删去介数最大的边，每次删除更新边的介数，如此循环，该算法在求解介数的时间复杂度较高[77]；LP(Label Propagation) 算法使用邻居节点信息决定当前节点的社群，也可以应用到多社群挖掘，但是结果存在震荡问题，性能不稳定[78]。目前应用最多的是基于模块 (Modularity) 法，通过对比社群挖掘结果与随机图的差异判断社群划分的优劣，研究表明基于模块值算法策略性能最好[79]。

基于模块值算法通常先假设网络满足某种统计分布，进而通过极大似然估计方法得到网络社群。核心思想为模块值 Q 最大化，见式(5-3)，Q 是评价网络

社群划分的优劣的质量指标。

$$Q = \frac{1}{2m}\sum_{vw}\left(W_{vw} - P_{vw}\delta(C_v, C_w)\right) \tag{5-3}$$

其中 W_{vw} 为网络中节点 v 和节点 w 之间的权重；m 为权重之和；P_{vw} 为零模型中节点之间的期望权重；C_v 表示节点 v 所属的社群的类别，如果节点 v 和节点 w 所属同一个社群，则 $\delta(C_v, C_w)$ 为 1，否则为 0。

模块法开始将每个节点视为 N 个独立社群，选择使 Q 值增大最多或减少最小的连个顶点进行社群合并。例如 A, B 连个社团合并，a_i 为社团 A 中的成员，b_j 为社团 B 中的成员，合并后的 Q 的改变量用 ΔQ 表示公式(5-4)，ΔQ 为一个矩阵，其元素 e_{ij} 为社团 i 和社团 j 合并后 Q 的变化值。

$$\Delta Q_{AB} = Q_{AB} - Q_A - Q_B = \frac{1}{2m}\sum_{a_i \in A, b_j \in B}\left\{\left(A_{ij} - \frac{k_i k_j}{2m}\right) \times \delta\left[r(i), r(j)\right]\right\} \tag{5-4}$$

合并之后，对应的 ΔQ 矩阵元素 e_{ij} 需要更新，如果选择让 A, B 社团合并为社团 D，则新的社团 D 与其他社团之间 (如社团 C) 。在计算新一轮的合并 Q 时，易知 $\Delta Q_{DC} = \Delta Q_{AC} + \Delta Q_{BC}$，上一次的迭代计算可以被下一轮计算所采用，只需将 Q 矩阵中的 i, j 社团相关的行列相加，最多进行 $n-1$ 次计算来构建完整的社群划分[80]。

5.4.2 社群挖掘算法并行化实现

传统基于模块值社群挖掘算法是采用串行实现：逐个选择节点，选择合并社群，选择模块值最大的合并社群。而且需要两两考虑到每个节点是否属于同一社群，算法时间复杂度将达到 $O(n^2)$，当数据量急剧增加的时候，算法的时间消耗是可不接受的。因此在并行计算框架下进行基于模块值进行社群挖掘，需要对公式(5-3)简化为公式(5-5)。其中 $\sum_{in}$ 是社群 c 内部相互连接权重和，$\sum_{tot}$ 该社群 c 外部连接权重和。通过对比可以发现去掉了节点是否属于同一社群的判断，避免了节点两两判断，简化了计算[81]。

$$Q = \sum_{c}\left\{\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2\right\} \tag{5-5}$$

判断某个节点 i 划分到其邻居节点计算模块值增益函数见公式(5-6)，其中 $k_{i,in}$ 为节点 i 与社区 C 连接权重总和，k_i 为节点 i 与其余节点连接权重。

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{5-6}$$

算法主要分为两个步骤，第一步计算每个节点按照公式(5-6)确定顶点 i 模块值增益最大进行合并社群，第二步骤更新网络图，步骤见算法5-3。

算法5-3每次迭代过程中依次选择图中的节点，对节点的所有的邻居节点尝试进行合并，对模块值增益最大且大于零的节点进行合并。但是这种计算方式

算法 5-3 社群挖掘算法

输入:

网络图:graph;

最多迭代次数: maxIterations;

模块值变化阈值: modularityThreshold;

输出:

模块值最大社群图: graph;

初始化 $C_i, \{i = 1, 2, \ldots, N\}$
iterate=0
while changeRate<modularityThreshold || iteration < maxIterations **do**
  changeRate=0
  for all *i* in graph.vertexs **do**
    $Q_i = [\ \ ]$
    for all *j* in *i* neighbors **do**
      Q_i += $\Delta Q(i, j)$ // 计算模块值增益
    end for
    Q_{max}=max(Q_i)
    if $Q_{max} > 0$ **then**
      graph=updategraph() // 如果最大模块值增益大于零，合并顶点
      changeRate= Q_{max}
    end if
  end for
  iterate += 1
end while
return graph

不利于并行计算框架，而且 Spark RDD 是不可变的数据结构，每次合并社群就要生成新的 Graph 对象，内存消耗非常大。因此选择每次迭代更新多个节点，通过 Graph 类的 aggregateMessages 函数同时对图的所有边及其相关的节点进行同时操作，因此需要定义 map 阶段的消息，按照公式(5-6)的变量，定义如下消息结构。

```
public class CombineMsg implements Serializable{
  public Integer degree;//该顶点的权重
  public Long commumity;//该顶点所属社区编号
  public Long commumityTotalDegree;//该社群权重总和
  public Long neighborDegree;//目标社群的权重
  public Long neighborCommumnity; //目标社群的编号
  public Long neighborCommumityTotalDegree;//目标节点的社群总权重
  public Long edgeWeight;//节点与与目标社群之间权重
}
```

对于发送到同一节点的消息集合，使用公式(5-6)，选择模块值增益值最大且大于 0 社群连边进行合并。合并完毕后生成的消息图包含了本次社群合并连接顶点，并与原图进行 joinVertice 操作，完成图的更新。

由于所有边是并行操作，会出现「社群归属延迟」问题，如图5–10比如顶点 1 所在社群 $c1$ 与顶点 2 所在社群 $c2$ 进行合并，而社群 $c2$ 与顶点 3 所在社群 $c3$ 进行合并，那么将会导致在 joinVertice 操作时候，这些顶点成为孤立的顶点。从拓扑结构来看，社群 $c1, c2, c3$ 应该属于同一个社群，因此对依据模块值增益生成的消息图计算连通分量，直接使用 Graph 对象调用 connectedComponents 函数即可，最后与原图进行 joinVertice 操作，完成图的更新，流程示意图见图5–11。

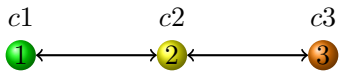

图 5–10 社群归属延迟

Figure 5–10 Delay of commumity combine illustration

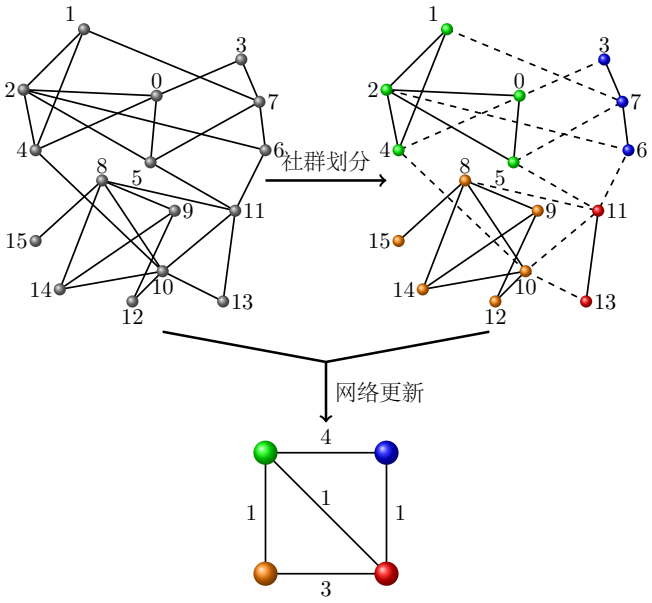

图 5–11 社群合并示意图

Figure 5–11 Community combine illustration

5.4.3 城市社群挖掘

由于人口流动往往存在较强的区域集聚性，在人口流动网络中体现出就是存在较强的社群结构，使用并行化社群挖掘算法，反应全国城市人口流动的集聚效应[82]。对全国地级市和直辖市人口流动网络进行社群挖掘，当整个人口流动网络划分为 9 个社群时，网络模块值最大，划分效果最好，城市社群具体划分结果见图5–12和表5–5。

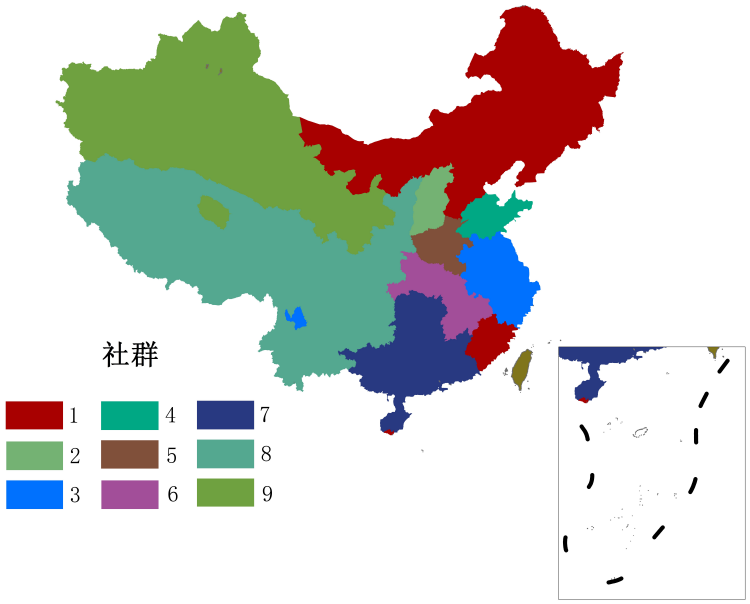

图 5–12 全国城市社群划分

Figure 5–12 National communities illustration

表 5–5 全国城市社群划分详细

Table 5–5 National cities communities details

社群	组成城市	社群	组成城市
1	北京、天津、河北城市、东三省城市、福建城市、三亚等	6	上海、江苏城市、安徽城市、浙江城市、丽江
2	山东城市	7	广东城市、广西城市、湖南城市等
3	河南城市	8	重庆、四川城市、贵州城市等西南城市
4	山西城市	9	新疆城市、甘肃城市等西北城市
5	湖北城市、江西城市		

对全国社群划分而言，也可以对其中的社群内部进一步使用社群挖掘算法，Spatial-Saprk 强大的运算能力可以很容易的做到这一点，比如上述全国社群 1 进一步使用社群挖掘算法，结果见图5–13，每个社群中城市划见表5–6。

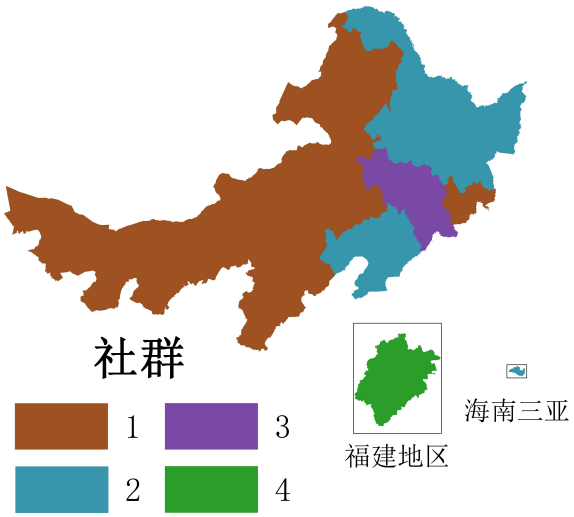

图 5–13 子社群划分

Figure 5–13 Sub-community illustration

表 5–6 子社群划分

Table 5–6 Sub-community details

子社群	组成城市	子社群	组成城市
1	北京，天津，河北，内蒙古，吉林延边	2	黑龙江，辽宁，海南三亚
3	吉林城市	4	福建

分析每个社群的组成城市和城市之间地理空间分布，体现了以下特征：

(1) 城市之间人口流动以省份为划分的特征较为明显，省份内部城市人口流动紧密。以山东、山西、河南等省份为代表，其各自内部城市在人口流动网络中单独划分为一个社群。

(2) 城市社群组成受地理空间位置影响较大，有明显的区域特征。在全国城市社群划分中，存在以北京为中心的北方城市社群，上海为中心的东部城市社群，广东为中心的南部城市社群等集聚现象。

(3) 城市社群组成也存在不受地理位置影响的特殊情况，如福建省的城市与北方城市组成同一社群，从侧面表明其人口流动突破了地理空间的限制；丽江市与东部城市组成同一社群、三亚市与北方城市组成同一社群，从一定程度表明这些旅游城市在春节假期的游客来源的组成。

(4) 当对全国社群进行子社群进行社群挖掘，可以发现以省份为集聚现象越发明显，但是也可以发现少数城市与其他省份人口联系更为密切，而这些往往是更值关注的内容，比如吉林的延边与北京、天津、河北等城市组成的社群更为紧密，而海南三亚与黑龙江和辽宁组成的社群更为紧密。

通过对人口流动网络社群挖掘，可以快速地从海量数据中发现信息，这些信息既可以验证现实中已有的观念，如省份内部人口流动紧密，空间邻近的省份人口交流频繁；也可以发现未知的现象，如福建省人口流动与北京、天津等省份城市更为密切，海南三亚与北方城市尤其是黑龙江省份城市人口联系紧密。这些现象可以对省份和区域的决策者关于城市规划、旅游航班调整、城市交通日常通勤等等方面提供很好的建议。

5.5 本章小结 (Chapter Summary)

本章以全国新浪微博用户位置入手，构建了全国城市 2016 年全国城市人口流动网络，首先通过构建城市人口流动指标，计算了城市人口流入、流出和流入流出比；借鉴 PageRank 计算每个城市在人口流动网络中的权重，并与该城市 GDP 进行相关性分析，分析城市权重分布情况。最后对人口流动网络中进行社群挖掘，将全国城市划分为 9 个社群，并分析了划分的特征。

6 结论和展望

6 Conclusion and Perspective

6.1 结论 (Conclusion)

为了高效处理海量空间数据，本文以 Spark RDD 为基础进行空间扩展，构建了 Spatial-Spark 并行空间计算框架。以此框架为基础，对海量新浪微博空间数据进行有效的分析，主要包括新浪微博 POI 数据和新浪微博用户位置数据分析和挖掘，并对相关算法进行了并行化改进。

论文主要的工作及其成果总结如下：

(1) 对 Hadoop MapReduce 和 Spark 并行计算框架进行分析，比较各自的优缺点。重点研究了 HDFS 分布式存储策略和 RDD 的计算特点，分析 Spark 并行计算框架的优点，主要包括内存计算带来的性能上优势和丰富的表现算子，为空间数据分析和挖掘提供了可能。

(2) 以 Spark 为基础，对其进行空间方面的拓展，构建了并行空间计算框架 Spatial-Spark，主要包括空间数据分布式读写和空间数据变换，对基本的空间数据类型点、线和面分别拓展成为 PointRDD、LineRDD 和 PolygonRDD，并根据空间数据分布特征，提供了空间分区功能；以 R 树为工具，对每个分区建立的空间索引，以加快空间数据分析速度。在此基础上，提供了空间数据分析常用的模块，主要包含空间拓扑查询、KNN 空间邻近查询和空间连接查询。最后搭建了分布式 Hadoop/Spark 计算集群，验证了 Spatial-Spark 在空间数据处理方面的优势。

(3) 分析了新浪微博提供的 API 接口，使用 C# SDK 编写了获取新浪微博 POI 和全国新浪微博用户位置数据的应用程序。

(4) 研究了空间关联规则算法，重点分析了空间同位规则挖掘挖掘算，并结合 Spatial-Spark 提出了并行化空间连接算法优化方案，对全国城市微博 POI 类别进行空位模式挖掘。首先针对上海、武汉和重庆三市进行二阶模式挖掘，分析 (高等院校、培训机构) 模式在不同距离阈值下三市的空间参与度。选择空间距离阈值 $d = 500\text{m}$ 和空间参与度阈值 0.6，对北京市进行同位模式挖掘，发现同位模式阶数越高，空间模式的数量越少，直至六阶，(KTV，中餐厅，咖啡厅，甜品店，美容美发店，酒吧) 为六阶空间模式的实例。由于新浪微博 POI 人为因素影响较大，因此这些 POI 类别集聚的现象更加表达了现实线下人们活动现象，这对商家商业推广提供了很好的建议。

(5) 新浪微博用户使用微博位置和用户账号位置形成人口流动，使用 Spatial-Spark 空间查询构建全国城市在 2016 年春节期间人口流动网络图。首先根据

GraphX 并行计算功能，统计了各个城市人口流入量、流出量和流出流出比，发现了全国城市人口流动情况的多样性；然后使用 PageRank 算法，计算出每全国城市在人口流动网络中的权重并与城市 GDP 进行关联分析，发现城市权重与城市 GDP 发展相关联，并确定了各个等级区域人口流动网络的中心；最后对整个人口网络进行进行社群挖掘，通过改进后模块值法，发现了全国城市人口流动社群，发现城市联系紧密性与省份有关，地理位置对其影响很大，但也存在突破地理空间位置限制的城市。

6.2 展望 (Perspective)

本文以新浪微博为研究内容，使用 Spark 进行了空间数据分析和挖掘进行了尝试，但是仍需在以下几个方面进行深入研究：

(1) 空间连接分区索引使用：空间索引能有效地加快空间查询速度，但在 Spatial-Spark 上层空间连接模块中尚未有效地使用已经建立的空间索引的进行加速算法，由于 R 树构造的特殊性，所以不能通过最小外包矩形进行 Join 操作。

(2) 同位模式算法改进：在新浪微博 POI 类别同位模式挖掘中，使用全连接算法，虽然通过并行化改进在二阶模式生成中减低了时间复杂度，但是在生成更高阶的过程中，RDD 提供的 join 操作包含了较多的不满足条件的组合模式，因此如果改进 RDD 在 join 操作的选择是值得研究的方向。

(3) 微博数据获取方式改进：由于新浪微博 API 的限制性，不能获取关于用户的所有微博信息，因此在只能获取用户的位置同时不能获取用户的历史微博数据，这些导致通过用户微博位置的注册地判断人口流动存在有偏性。如果采用网络爬虫的方式，突破新浪微博 API 的限制，获取更多的用户的信息，那么将会发现更加有价值的信息。

参考文献

[1] 李德仁. 智慧地球时代测绘地理信息学的新使命 [J]. 中国测绘. 2013, 37(1):32–33.

[2] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]. IEEE Symposium on Mass Storage Systems & Technologies. [S.l.], 2010: 1-10.

[3] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets[C]. Usenix Conference on Hot Topics in Cloud Computing. [S.l.], 2010: 10-10.

[4] 李文栋. 基于 Spark 的大数据挖掘技术的研究与实现 [D]. 济南: 山东大学, 2015.

[5] Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters.[C]. Conference on Symposium on Opearting Systems Design & Implementation. [S.l.], 2004: 137-150.

[6] Lars G. Hbase[M]. [S.l.]: [s.n.], 2011.

[7] Lyubimov D, Palumbo A. Apache Mahout: Beyond MapReduce[M]. [S.l.]: CreateSpace Independent Publishing Platform, 2016.

[8] Vavilapalli V K, Murthy A C, Douglas C, et al. Apache hadoop yarn: yet another resource negotiator[C]. Symposium on Cloud Computing. [S.l.], 2013: 5.

[9] Aliyun. Aliyun, an alibaba unit, is building china's first "cloud hospital"[J]. Flare. 2015.

[10] Varia J. Best Practices in Architecting Cloud Applications in the AWS Cloud[M]. [S.l.]: [s.n.], 2011: 457-490.

[11] 马磊. 一种基于 HDFS 的分布式多级 R 树空间索引研究 [D]. 北京: 中国测绘科学研究院, 2016.

[12] 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.

[13] 温馨, 罗侃, 陈荣国. 基于 shark/spark 的分布式空间数据分析框架 [J]. 地球信息科学学报. 2015, 17(4):401–407.

[14] 李璐明, 蒋新华, 廖律超. 基于弹性分布数据集的海量空间数据密度聚类 [J]. 湖南大学学报 (自科版). 2015, 42(8):116–124.

[15] 靳凤营, 张丰, 杜震洪, 等. 基于 spark 的土地利用矢量数据空间叠加分析方法 [J]. 浙江大学学报 (理学版). 2016, 43(1):40–44.

[16] Abouzeid A, Bajda-Pawlikowski K, Abadi D, et al. Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads[J]. Proceedings of the Vldb Endowment. 2009, 2(1):922–933.

[17] Witayangkurn A, Horanont T, Shibasaki R. Performance comparisons of spatial data processing techniques for a large scale mobile phone dataset[C]. Com.Geo. [S.l.], 2012: 1-6.

[18] ESRI. Gis tools for hadoop: Big spatial data analytics for the hadoop framework[M]. [S.l.]: [s.n.], 2014.

[19] Eldawy A, Mokbel M F. Spatialhadoop: A mapreduce framework for spatial data[C]. IEEE International Conference on Data Engineering. [S.l.], 2016: 1352-1363.

[20] Yu J, Wu J, Sarwat M. Geospark: a cluster computing framework for processing large-scale spatial data[C]. The Sigspatial International Conference. [S.l.], 2015: 1-4.

[21] Andrienko G, Andrienko N, Bosch H, et al. Thematic patterns in georeferenced tweets through

space-time visual analytics[J]. Computing in Science & Engineering. 2013, 15(3):72–82.

[22] Li L, Goodchild M F, Xu B. Spatial, temporal, and socioeconomic patterns in the use of twitter and flickr[J]. Cartography and Geographic Information Science. 2013, 40(2):61–77.

[23] Hollenstein L, Purves R. Exploring place through user-generated content: Using flickr to describe city cores[J]. Journal of Spatial Information Science. 2010, 1(1):21–48.

[24] Liu X, Wang J. The geography of weibo[J]. Environment & Planning A. 2015, 47(6):1231–1234.

[25] 刘大均, 胡静, 程绍文, 等. 中国旅游微博空间分布格局及影响因素–以新浪旅游微博为例 [J]. 地理科学. 2015, 35(6):717–724.

[26] 徐艳, 黎明. 基于新浪微博视角下的城市网络空间特征分析——以重庆市主城区为例 [J]. 西南师范大学学报 (自然科学版). 2014, 39(6):43–49.

[27] Oriani A, Garcia I C. From backup to hot standby: High availability for hdfs[C]. IEEE Symposium on Reliable Distributed Systems. [S.l.], 2012: 131-140.

[28] Singh A J, Sarjolta V. Mapreduce wordcount: Execution and effects of altering parameters[M]. [S.l.]: [s.n.].

[29] Grolinger K, Hayes M, Higashino W A, et al. Challenges for mapreduce in big data[C]. Proc. of the IEEE World Congress on Services. [S.l.], 2014: 182-189.

[30] Guller M. Spark SQL[M]. [S.l.]: Apress, 2015.

[31] Mllib W I. Mllib: Scalable machine learning on spark[M]. [S.l.]: [s.n.].

[32] Xin R S, Gonzalez J E, Franklin M J, et al. Graphx: a resilient distributed graph system on spark[C]. International Workshop on Graph Data Management Experiences and Systems. [S.l.], 2013: 2.

[33] Guller M. Spark Streaming[M]. [S.l.]: Apress, 2015.

[34] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C]. Usenix Conference on Networked Systems Design and Implementation. [S.l.], 2012: 2-2.

[35] 杨春成. 空间数据挖掘中聚类分析算法的研究 [D]. 郑州: 解放军信息工程大学, 2004: 61-62.

[36] 李小文, 曹春香, 常超一. 地理学第一定律与时空邻近度的提出 [J]. 自然杂志. 2007, 29 (2):69–71.

[37] 曾玲, 熊才权, 胡恬. 关联规则在空间数据挖掘中的研究 [J]. 计算机与数字工程. 2005, 33(6):71–73.

[38] 廉捷, 周欣, 曹伟, 等. 新浪微博数据挖掘方案 [J]. 清华大学学报自然科学版. 2011, (10): 1300–1305.

[39] Nemeslaki A, Pocsarovszky K. Web crawler research methodology[J]. General Information. 2011.

[40] Recordon D, Hardt D, Hammerlahav E. The oauth 2.0 authorization protocol[J]. International Journal of Pharmaceutics. 2011, 271(1–2):95–113.

[41] Bychowski C T, NavTech, Williams J, et al. Open gis consortium, inc.[M]. [S.l.]: [s.n.], 2003.

[42] Urbanek S. proj4: A simple interface to the proj.4 cartographic projections library[M]. [S.l.]: [s.n.], 2012.

[43] Johansson M, Harrie L. Using java topology suite for real-time data generalisation and integration[J]. Proceedings of the Workshop of the International Society for Photogrammetry & Remote Sensing. 2002.

[44] Guttman A. R-trees: a dynamic index structure for spatial searching[C]. ACM SIGMOD International Conference on Management of Data. [S.l.], 1984: 47-57.

[45] Huang P W, Lin P L, Lin H Y. Optimizing storage utilization in r-tree dynamic index structure for spatial databases □[J]. Journal of Systems & Software. 2001, 55(3):291–299.

[46] 谢俊平, 杨敏华. 拓扑关系和方向关系的统一表达模型 [J]. 测绘科学. 2012, 37(2): 150–152+158.

[47] 董亭亭. 大数据下空间数据索引和 kNN 查询技术的研究 [D]. 大连: 大连理工大学, 2013.

[48] Jacox E H, Samet H. Spatial join techniques[J]. Acm Transactions on Database Systems. 2007, 32(1):7.

[49] Ghosh A. Install apache hadoop on ubuntu on single cloud server instance[M]. [S.l.]: [s.n.], 2017.

[50] Ghosh A. Install apache spark on ubuntu single cloud server with hadoop[M]. [S.l.]: [s.n.], 2017.

[51] Fernández A M, Torres J F, Troncoso A, et al. Automated Spark Clusters Deployment for Big Data with Standalone Applications Integration[M]. [S.l.]: [s.n.], 2016: 1-10.

[52] Zhao Y, Hu F, Chen H. An adaptive tuning strategy on spark based on in-memory computation characteristics[C]. International Conference on Advanced Communication Technology. [S.l.], 2016: 484-488.

[53] 韩华瑞, 代侦勇. 湖北省微博签到活动空间差异分析——以新浪微博为例 [J]. 测绘与空间地理信息. 2016, (10):159–162.

[54] Ye M, Yin P, Lee W C, et al. Exploiting geographical influence for collaborative point-of-interest recommendation[C]. Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July. [S.l.], 2011: 325-334.

[55] Yang Q H, Snyder J P, Tobler W R. Map projection transformation[J]. Taylor & Francis Ltd London. 1999.

[56] 朱晨曦, 晏王波. 基于微博签到的地理空间信息研究 [J]. 地理空间信息. 2016, 14(5): 28–30.

[57] Agrawal R, Srikant R. Mining sequential patterns[C]. Eleventh International Conference on Data Engineering. [S.l.], 1995: 3-14.

[58] Agrawal R, Gehrke J E, Gunopulos D, et al. Automatic subspace clustering of high dimensional data for data mining applications[M]. [S.l.]: [s.n.], 1999: 94–105.

[59] 王佐成, 汪林林, 薛丽霞, 等. 空间关联规则的双向挖掘 [J]. 计算机科学. 2006, 33(7):

199–203.
[60] 陈江平. 空间关联规则挖掘算法研究 [J]. 计算机工程. 2004, 30(23):53–55.
[61] 蔡伟杰, 张晓辉, 朱建秋, 等. 关联规则挖掘综述 [J]. 计算机工程. 2001, 27(5):31–33.
[62] Koperski K, Han J. Discovery of spatial association rules in geographic information databases[C]. International Symposium on Advances in Spatial Databases. [S.l.], 1995: 47-66.
[63] Ester M, Kriegel H P, Sander J. Spatial data mining: A database approach[C]. International Symposium on Advances in Spatial Databases. [S.l.], 1999: 47-66.
[64] 万幼, 边馥苓. k-邻近空间关系下的同位关联模式挖掘 [C]. 地理信息系统全国博士生学术论坛. 南京, 2008.
[65] Bian F. A novel spatial co-location pattern mining algorithm based on k-nearest feature relationship[J]. Geomatics & Information Science of Wuhan University. 2009, 34(3):331–334.
[66] Jin S Y, Shekhar S. A joinless approach for mining spatial colocation patterns[J]. IEEE Transactions on Knowledge & Data Engineering. 2006, 18(10):1323–1337.
[67] Huang Y, Shekhar S, Xiong H. Discovering colocation patterns from spatial data sets: A general approach[J]. IEEE Transactions on Knowledge & Data Engineering. 2004, 16(12): 1472–1485.
[68] Yoo J S, Shekhar S, Smith J, et al. A partial join approach for mining co-location patterns[C]. ACM International Workshop on Geographic Information Systems, Acm-Gis 2004, November 12-13, 2004, Washington, Dc, Usa, Proceedings. [S.l.], 2004: 241-249.
[69] Wang L, Bao Y, Lu Z. Efficient discovery of spatial colocation patterns using the icpi-tree[J]. Open Information Systems Journal. 2009, 3(2):69–80.
[70] 禹文豪, 艾廷华, 周启. 设施 poi 的局部空间同位模式挖掘及范围界定 [J]. 地理与地理信息科学. 2015, 31(4):6–11.
[71] 甄峰, 王波, 陈映雪. 基于网络社会空间的中国城市网络特征——以新浪微博为例 [J]. 地理学报. 2012, 67(8):1031–1043.
[72] Gonzalez J E, Xin R S, Dave A, et al. Graphx: graph processing in a distributed dataflow framework[C]. Usenix Conference on Operating Systems Design and Implementation. [S.l.], 2014: 599-613.
[73] 曹盼盼, 阎春宁. 人类通信模式的幂律分布和 zipf 定律 [C]. 全国复杂网络学术会议论文. 青岛, 2009: 51-56.
[74] 冯梅, 陈鹏. 中国钢铁产业产能过剩程度的量化分析与预警 [J]. 中国软科学. 2013, (5): 110–116.
[75] Boldi P, Santini M, Vigna S. Pagerank[J]. Acm Transactions on Information Systems. 2009, 27(4):1–23.
[76] Girvan M, Newman M E. Community structure in social and biological networks[J]. Proceedings of the National Academy of Sciences of the United States of America. 2002, 99(12): 7821.
[77] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Pro-

ceedings of the National Academy of Sciences of the United States of America. 2001, 99(12): 7821.

[78] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks.[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics. 2007, 76(2):036106.

[79] Newman M E. Modularity and community structure in networks[J]. Proceedings of the National Academy of Sciences. 2006, 103(23):8577.

[80] Newman M E J. Communities, modules and large-scale structure in networks[J]. Nature Physics. 2012, 8(8):25–31.

[81] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics Theory & Experiment. 2008, 2008(10):155–168.

[82] 杨小柳. 从地域城市到移民城市: 全国性城市社会的构建 [J]. 民族研究. 2015, (5):41–51.

作者简历

一、基本情况

姓名: 高峰　性别: 男　民族: 汉　出生年月: 1992-02　籍贯: 江苏扬州

2010-09 — 2014-07　中国矿业大学环境与测绘学院工学学士;

2014-09 — 2017-06　中国矿业大学环境与测绘学院工学硕士研究生.

二、学术成果

1. 高峰，土地确权内业处理软件，软件著作权，登记号：2015SR091014

三、获奖情况

1. 2014-2015 年度　一等奖学金
2. 2015-2016 年度　二等奖学金
3. 2016-2017 年度　三等奖学金

四、研究项目

1. 国家高分重点项目：中科院电子所苏州研究院，2016，参与;
2. 国家自然基金项目：数字摄影测量目标精密定位于精度估计研究，编号：41171343，参与;
3. 山东省梁山县农村土地确权登记颁证项目，2014，主要参与;

学位论文原创性声明

本人郑重声明: 所呈交的学位论文《基于 Spatial-Spark 海量网络空间数据分析与应用》, 是本人在导师指导下, 在中国矿业大学攻读学位期间进行的研究工作所取得的成果. 据我所知, 除文中已经标明引用的内容外, 本论文不包含任何其他个人或集体已经发表或撰写过的研究成果. 对本文的研究做出贡献的个人和集体, 均已在文中以明确方式标明. 本人完全意识到本声明的法律结果由本人承担.

学位论文作者签名:

年　月　日

学位论文数据集

关键词 *	密级 *	中图分类号 *	UDC	论文资助
Spatial-Spark, 新浪微博, 同位模式, 网络分析	公开	P2	910.1	无

学位授予单位名称 *	学位授予单位代码 *	学位类别 *	学位级别 *
中国矿业大学	10290	工学	硕士

论文题名 *	并列题名 *	论文语种 *
基于 Spatial-Spark 海量网络空间数据分析与应用	Analysis and Data Mining about Spatial Data of Sina Weibo Based on Spatial-Spark	中文

作者姓名 *	高峰	学号 *	TS14160005
培养单位名称 *	培养单位代码 *	培养单位地址	邮编
环境与测绘学院	10290	江苏省徐州市	221116
学科专业 *	研究方向 *	学制 *	学位授予年 *
大地测量学与测量工程	3S 技术集成及其应用	三年	2017 年

论文提交日期 *	2017 年 6 月
导师姓名 *	高井祥	职称 *	教授

评阅人	答辩委员会主席 *	答辩委员会成员 *
	吴侃	王永波, 杨化超, 陈国良, 刘志平

电子版论文提交格式 文本 (✓) 图像 (　) 视频 (　) 音频 (　) 多媒体 (　) 其他 (　)
推荐格式: application msword; application pdf

电子版论文出版 (发布) 者	电子版论文出版 (发布) 地	权限声明

论文总页数 *	66

注: 共 33 项, 其中带 * 为必填数据, 共 22 项.