武汉大学研究生课程论文

课程名称：	地理信息科学与技术
教师姓名：	罗 学 年
学生姓名：	叶 小 川
学生学号：	2022282140108
学　　院：	测 绘 学 院
专　　业：	资 源 与 环 境

二零二二年十二月

时空大数据管理与分析

测绘学院 资源与环境 叶小川 2022282140108

摘要：时空数据同时具有时间和空间两个数据特征，是最为典型的大数据。在日常生活如导航、智慧城市等有着重要意义，研究者根据时空大数据的特性探索了多种空间索引机制，并提出基于 Hadoop 和 Spark 等分布式系统的时空大数据管理系统，本文探讨了时空大数据系统的技术进展及其可行性。

关键词：时空数据，大数据，Hadoop，空间索引，Spark;

1 引言

时空数据同时拥有时间和空间两个特征，是一种结构复杂的多维数据，作为最典型的大数据之一，其量级也非常庞大。太空望远镜每周能收集约 150GB 的时空数据， 而美国宇航局（NASA）的航天器每天能收集约 4TB 的数据。时空数据作为最重要的大数据之一，其在日常生活中的各个领域正发挥着积极作用，例如打车行业、交通监控和智慧城市等。时空大数据的价值在于分析其时间、空间、对象之间的关联关系并 对其背后隐藏的规律进行发掘与利用。因此，如何对海量庞杂的时空数据进行高效的 存储、组织、管理和查询是目前亟需解决的问题，同时也是现在热点的研究对象。

时空数据库（Spatiao-Temporal Database）是可以同时支持空间维度与时间维度两种特性的数据存储和管理系统，是 20 世纪 90 年代诞生的科研领域。发展至今，国内外在该领域都取得了一定的研究成果。

国外欧美国家在时空数据库领域起步较早，在学术研究方面位于世界前列。时空数据库领域里著名的研究学者 Ralf H.Güting 和 Markus Schneider 著有《Moving Objects Databases》[1]一书，该书籍系统性地介绍了关于移动对象的数据模型、索引结构和 查询方法等内容，在当前该领域中具有相当高的参考价值。国外学者们也提出了各种 时空数据模型，为时空数据库的发展贡献了自己的一份力量。例如，Langran 提出了时间快照模型以及在此基础之上改进后的基态修正模型[2]；Yeh 等人通过时间函数将时间属性与空间属性集成起来，提出了 BTS 时空模型[3,4]。国内虽然在时空数据库领域起步较晚，但最近几年已有不少高校平台、公司企业 和研究机构在该领域奋起直追，如浙江大学、北京大学、滴滴公司和中科院等，并且也取得了一定成果。浙江大学的高云君[5]对时空数据库查询处理中的关键技术进行了研究，包括并行最近邻查询、并行 Skyline 查询和历史连续 kNN 查询等，丰富了国内在时空查询技术方面的理论。北京测绘的王天明和李莹等[6]对智慧城市云平台下的时空数据库进行了相关设计与研究。

2 时空大数据索引与查询技术

时空索引是时空数据库中最为核心的一个模块，相比于传统地理空间数据库中的空间索引，时空索引加入了时间维信息。

2.1 R-tree 索引及其变种

R 树[7]最早是由 Guttman 于 1984 年提出，是空间索引领域中非常重要的一种索引结构。R 树可以看成是 B 树[8]在多维空间上的自然扩展，具有高度的平衡性。从逻辑层面上看，R 树的每一个节点对应着数据空间内的一个区域，名为最小包围矩形（Minimal Bounding Rectangle），简称 MBR。若区域对应的节点为 R 树的非叶节点，则节点的每一个子项内将会存储一个子节点及其对应的 MBR；若区域对应的节点为 R 树的叶节点，则节点的每一

个子项内将会存储一个实际数据对象的引用及其 MBR，但不会存储实际数据对象本身。通过 MBR 结构，树可以简单有效的判断在某个节点的管理范围内是否可能存在待查询的数据对象，从而在查询过程中尽可能早的排除掉那些不包含待查询数据对象的节点。

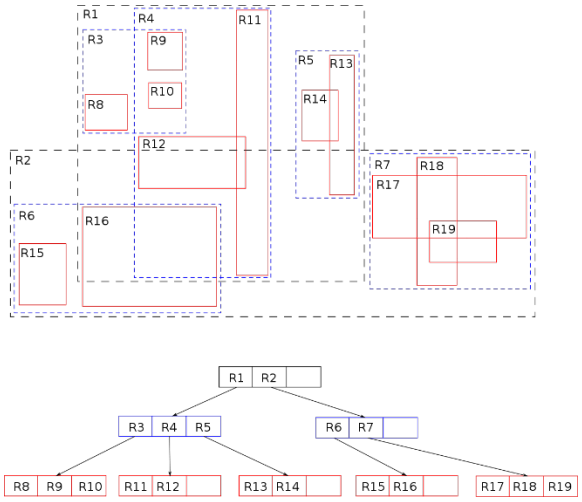

图 1 R 树

R 树中最重要的操作是插入新数据：首先从根节点开始，迭代调用一个选择叶节点的算法 ChooseLeaf 来决定数据插入的位置，直至某个叶节点 T。若 T 中还有足够的空间则直接插入，否则需要调用分裂节点的算法 SplitNode 将 T 一分为二，然后向上插入到父节点中，并在父节点也溢出的情况下继续采用该策略向上传导，直至根节点。若根节点也溢出，则同样将其分裂并创建一个新的根节点指向之前旧的根节点分裂后的两个节点。显然在插入新数据的整个过程中，选择叶节点以及分裂满溢节点这两个算法的优劣程度，将会对整棵 R 树的索引结构和构建效率起到至关重要的影响。其中原始 R 树采取的做法是：在选择叶节点的过程中以最小 MBR 增长为目标进行选取，在分裂满溢节点的过程中以最小 MBR 面积总和为目标进行切分。

R 树面临的一个主要问题是当不同节点间的 MBR 存在大量重叠时，在查询过程中就可能需要对多条路径都进行检索才能得到最终结果，因此效率会大大降低。为了避免该问题，Sellis 等人提出了 R+树[9]。它采用了对象分割技术，要求跨越多个子空间的数据对象必须拆分为两个或多个不相交的 MBR，以实现非叶节点之间数据的零重叠。R+树虽然解决了 R 树中多路径搜索的问题，但也使得某一特定对象的信息可能会存储在多个节点中，需要耗费更多的存储空间，同时 在构建过程中也需要一系列额外的复杂更新操作来维持其结构特性。

R 树的另一个重要变种是由 Beckmann 等人提出的 R*树[10]，他们认为原始 R 树存在的 MBR 重叠现象本身并不意味着平均查询性能的下降，而如何合理的插入数据才是影响 R 树查询性能的关键。为此他们提出了一系列可能影响检索性能的参数，将它们进行组合并进行了大量的实验，在每次选择叶节点和分裂节点 时都会进行多次尝试，然后根据事先定义的效益值计算方法来决定最终方案。实验显示，R*树与原始 R 树相比有着更高的空间利用率，以及较少的节点分裂次数，使得数据对象的 MBR 更接近于正方形，从而极大的提升了性能，但同时也大大增加了 CPU 的计算代价和时间成本。

2.2 网格索引

网格索引[11]是地理信息系统（GIS）中常用的空间索引之一。其思想是使用一定大小的网格将整个空间区域进行划分，则每个区域可看成是一个类似桶的数据存储结构，负责维护进入该区域内的实际数据点，每个数据点都将唯一落在一个网格中。当需要插入或删除一个数据点时，只需要根据其空间坐标就可以计算得出其所归属的网格范围，然后将其简单加入

或删除即可。一般来说，影响网格索引性能的关键在于网格大小的选择。网格越大，对应的网格索引表的记录数就越少，但是平均每个网格内的数据点数也就会越多，同时不同数据点落在同一个网格中的几率也会越高；反之，网格越小，网格索引表中的记录数就越多，但平均每个网格内的数据点数也会相对变少，同时不同数据点落在同一个网格中的几率也会降低。

2.3 四叉树索引

四叉树[12]由 Finkel 和 Bentley 于 1974 年提出，是地理信息系统（GIS）中常用的空间索引之一。其基本思想是：将某个已知的空间范围划分为四个子空间，每个子空间又可以根据需要决定是否继续划分，如此递归执行直到满足某个预先 设定的终止条件。四叉树的理念与构建步骤都并不复杂，并且在空间数据对象的 分布比较均匀的情况下，数据的插入和查询效率都较高。但相反的若空间数据对象的分布较为倾斜，则对应于数据密集区域的四叉树的层数会持续增加，最终导致整个四叉树结构的严重失衡，查询效率也会因此而大幅下降。一棵典型四叉树的结构如图所示。

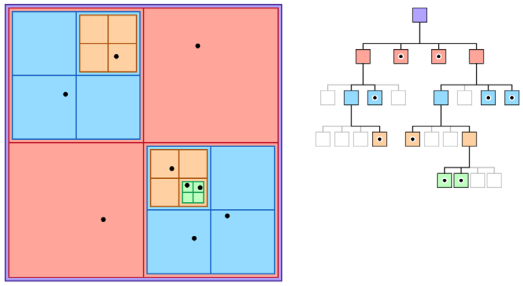

图 2 四叉树

2.4 kd 树索引

kd 树[13]是在 k 维欧几里德空间里组织数据的一种结构，可看成是 k 维的二叉检索树。它的每个节点都表示 k 维空间中的一个点，并且和一个矩形区域相对应，树的根节点则和整个空间范围相对应。kd 树要求用平行于坐标轴的横纵分界线将空间划分为若干子区域，使得每个子区域中的数据点个数不超过某个给定值。分界线仅仅起到划分界限的作用，它的选取没有硬性的限制，一般来说都选用通过某个数据点的横向线或者纵向线。位于分界线上的数据点，规定对于左右分界线来说属于右部，对于上下分界线来说属于上部。

在插入数据时，若 kd 树为空，则插入数据点成为新的根节点；否则继续在其左子树或右子树上查找，直到某个叶节点。此时该数据点将对应的成为该叶节点的左子节点或右子节点。k-d 树删除节点的操作在思想与传统的二叉检索树类似，但实际步骤却会复杂很多。由于在构建过程中会对整个空间区域按照不同的维度进行交替式的划分，因此在对节点进行删除时还必须考虑对应子树的结构维护问题，从而保持整棵 kd 树的空间布局。

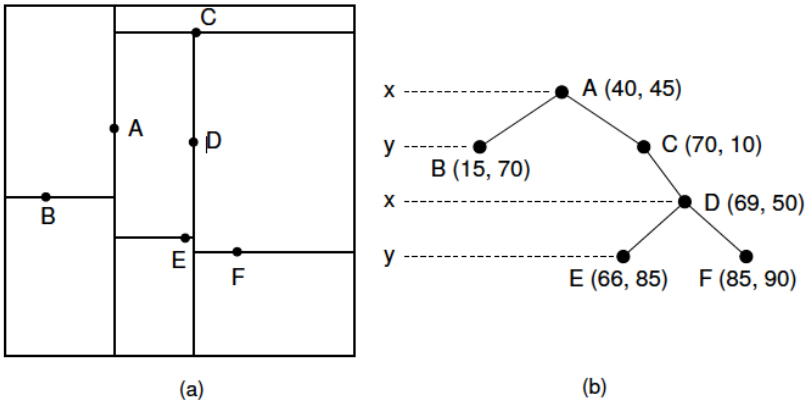

图 3 kd 树

一棵典型 kd 树的结构如图 3 所示。图中左侧表示的是该 kd 树对应的空间划分效果，右侧表示的是最终构建得到的 kd 树的索引结构。以根节点为例，划分依据为对应数据点的 x 轴坐标值，则对于根节点的左子树中的所有数据点。

3 分布式时空大数据索引与查询系统实现

以 Hadoop 、Spark 为代表的的分布式系统的出现，为处理海量数据提供了新的解决思路。分布式系统的存储能力与计算能力会以线性增长方式进行水平扩展[14]，由一台扩展到上千台，并且它的生态圈具有非常多的组件，从而用来对大数据进行计算、存储和分析，主要介绍 Hadoop 和 Spark 分布式系统。

3.1 基于 Hadoop 的时空大数据管理与应用

（1）相关技术

自从 2003 年 GFS[15]和 2004 年 MapReduce[16]论文的发表到 2008 年 Hadoop 成为了 Apache 的顶级开源项目，它经历了迅速的成长，多种的应用场景已经证明了它的成功、多样性与生命力。

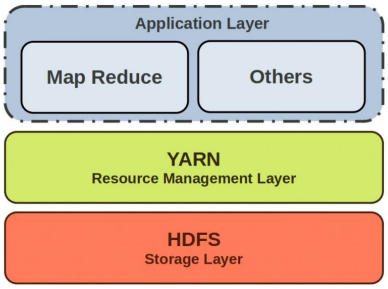

图 4 Hadoop 生态

Hadoop 项目由众多模块组成，其内部生态圈如图 4 所示，各个模块的具体介绍如下：

HDFS 是 Hadoop 中数据存储管理的基础，负责管理 Hadoop 集群中每个存储节点上的文件。HDFS 具有高容错性，能够自动保存多个副本，出现故障后系统会自动检测并通过副本及时处理。HDFS 也能够实现高吞吐量的流式数据访问

YARN 是任务调度和集群资源管理框架。

MapReduce 是一个并行计算框架，主要由 Map 和 Reduce 两大模块组成，数据根据键值对的方式进行处理，Map 在接收到传入的一堆错杂无序的 Key-Value 数据之后，经过用户自定义解析处理完成后仍然以 Key-Value 的形式输出到 Reduce，Reduce 最终将处理完成之后的最终结果写到 HDFS 中的某个目录中。MapReduce 的出现，把一个复杂任务成功地拆分成为了多个简单的小任务，这些小任务分别在多个计算节点上进行计算，大大提高了效率。

（2）应用实例

Eldawy 等[17]基于 Hadoop 提出了 SpatialHadoop，一个处理空间数据的高效 MapReduce 框架，通过两层空间索引建立数据库，在 MapReduce 框架下可以进行高效的 kNN、Range 查询、和 SpatialJoin 等操作。

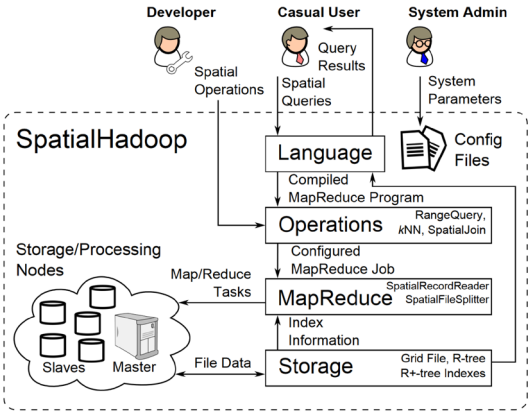

图 5 SpatialHadoop 架构

SpatialHadoop 的结构如图 5 所示，其中通过建立 Master-Slaves 计算集群，对数据进行存储，不同用户可以通过 operation 层进行 mapreduce 计算操作。该系统被证明可以有效分布式处理海量的空间数据，避免单个计算节点带来的计算瓶颈。

3.2 基于 Spark 的时空大数据管理与应用

（1）相关技术

Spark 是 Apache 中进行大数据分析处理的引擎[18]，是基于内存计算的并行计算架构，所以它的数据处理速率更快，具有高容错能力和高可伸缩性，在性能和通用性上都有显著优势。

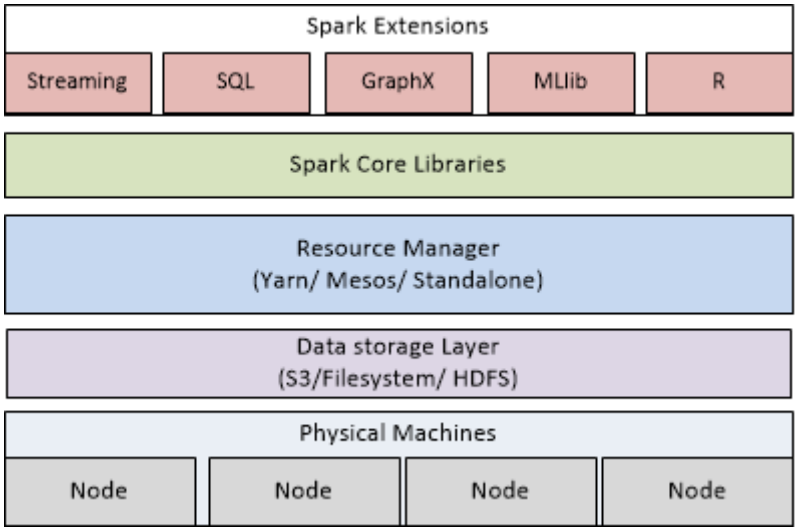

图 6 Spark 生态

Spark Core 是 Spark 的核心，用来进行批处理，Spark 在项目中常常被用来做迭代计算和交互式分析。

弹性分布式数据集 RDD（Resilient Distributed Dataset）是 Spark 中最核心的一个概念，每个 RDD 可以被分割成为多个区，并且每个区都是一个数据集片段，集群中的不同节点上可能会存储着同一个 RDD 的不同分区。

（2）应用实例

苏敏章[19]提出了基于 Spark 的时空数据查询与分析系统 ST-Spark，实现了范围查询、kNN 查询、预测分析、时空聚类分析等功能。具有快速查询、有效分析的优点。

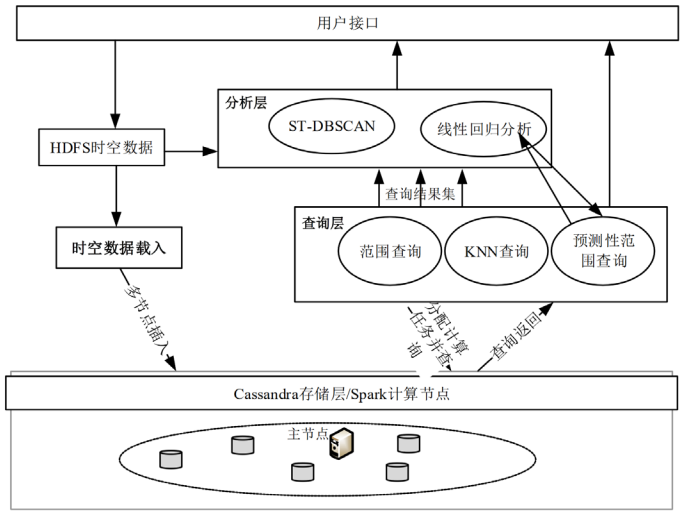

图 7 ST-Spark 架构

ST-Spark 架构如图所示，其中以分布式 NoSQL 数据库 Cassandra 为存储引擎，设计了契合 Spark 计算模型的时空数据存储和网格索引模型。基于设计的存储与网格索引模型，提出了 Spark 上的时空范围查询与 K 邻近查询方法。通过 Cassandra 服务端过滤优化，解决了查询时空索引时粗过滤效率低下的问题。提出“网格外扩法”， 确定至少包含 K 个时空对象的网格集合，提高了 K 邻近查询效率。对时空轨迹数据进行多项式拟合、线性回归预

测分析。为提高预测正确性，设计了基于滑动窗的多项式拟合时空轨迹预测分析方法。

4 结论

随着地理信息技术的发展，时空数据有着数据结构复杂和数量级庞大的特点，针对如何对时空数据进行索引、存储和计算，研究者提出了分布式的系统，很好解决了时空大数据单台服务器的运算瓶颈，对时空大数据扩大范围、进行更高要求的计算操作来服务生产生活需要打下基础。

参考文献

[1] GÜTING R H, SCHNEIDER M. Moving Objects Databases[M]. Elsevier, 2005.
[2] LANGRAN G. Time in Geographic Information Systems[M/OL]. London: CRC Press, 2020. https://doi.org/10.1201/9781003062592.
[3] YEH T S, DE CAMBRAY B. Modeling highly variable spatio-temporal data[C]//Australasian Database Conference. Citeseer, 1995: 0.
[4] YEH T S, DE CAMBRAY B. Time as a geometric dimension for modeling the evolution of entities: a 3D approach[C]//Int. Conf. on Integrating GIS and Environmental Modeling. 1993.
[5] 高云君. 时空数据库查询处理关键技术研究[D/OL]. 浙江大学, 2008[2022-12-01].
[6] 王天明, 李莹, 梁建平. 智慧城市时空数据库设计与研究[J/OL]. 北京测绘, 2017(06): 72-76. https://doi.org/10.19580/j.cnki.1007-3000.2017.06.017.
[7] GUTTMAN A. R-trees: A dynamic index structure for spatial searching[C]//Proceedings of the 1984 ACM SIGMOD international conference on Management of data. 1984: 47-57.
[8] BAYER R, MCCREIGHT E. Organization and maintenance of large ordered indices[C]//Proceedings of the 1970 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control. 1970: 107-141.
[9] SELLIS T, ROUSSOPOULOS N, FALOUTSOS C. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects.[R]. 1987.
[10] BECKMANN N, KRIEGEL H P, SCHNEIDER R, 等. The R*-tree: An efficient and robust access method for points and rectangles[C]//Proceedings of the 1990 ACM SIGMOD international conference on Management of data. 1990: 322-331.
[11] NIEVERGELT J, HINTERBERGER H, SEVCIK K C. The grid file: An adaptable, symmetric multikey file structure[J]. ACM Transactions on Database Systems (TODS), 1984, 9(1): 38-71.
[12] FINKEL R A, BENTLEY J L. Quad trees a data structure for retrieval on composite keys[J]. Acta informatica, 1974, 4(1): 1-9.
[13] BENTLEY J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM, 1975, 18(9): 509-517.
[14] WU J. Distributed system design[M]. CRC press, 2017.
[15] GHEMAWAT S, GOBIOFF H, LEUNG S T. The Google file system[C]//Proceedings of the nineteenth ACM symposium on Operating systems principles. 2003: 29-43.
[16] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.
[17] ELDAWY A, MOKBEL M F. A demonstration of spatialhadoop: An efficient mapreduce framework for spatial data[J]. Proceedings of the VLDB Endowment, 2013, 6(12): 1230-1233.
[18] KARAU H, WARREN R. High performance Spark: best practices for scaling and optimizing Apache Spark[M]. O'Reilly Media, Inc., 2017.
[19] 苏敏章. 基于 Spark 的时空数据查询与分析关键技术研究[D/OL]. 西安电子科技大学, 2018[2022-11-19].