

毕业设计外文资料翻译

学 院： 信息科学与工程学院

专 业： 软件工程

姓 名： 郭迅

学 号： 181401304

外文出处：Ilba,M.(2021).Parallel algorithm for improving the performance of spatial queries in SQL: The use cases of SQLite and PostgreSQL databases. Computers & Geosciences, 155, 104840. doi:10.1016/j.cageo.2021.104840

附 件： 1.外文资料翻译译文；2.外文原文。

|  |
| --- |
| 指导教师评语：  签名：  年 月 日 |

附件1：外文资料翻译译文

# SQL中提高空间查询性能的并行算法：SQLite/SpatiaLite和PostgreSQL/PostGIS数据库的用例

**摘要**——本文提出了一种执行空间查询并行处理的开源算法，在此期间，使用空间索引完成对要进行空间关系测试的对象的初始选择。然后通过使用 OFFSET 和 LIMIT 子句将这些数据进一步细分为更小的子组，其中分配利用复杂计算的空间关系测试，从而创建并行运行的多个进程。该算法使用来自 SQLite/SpatiaLite和PostgreSQL/PostGIS 数据库的数据进行了测试。在处理涉及六个线程的空间关系查询时，与在 SQLite/SpatiaLite 数据库上的单线程处理相比，该算法在性能上产生了3.6倍的最大加速提升，在PostgreSQL/PostGIS数据库上产生了5.1倍的最大加速提升。在单层分析（例如，面积计算、缓冲区生成）中，观察到查询处理的加速时间提高了5倍。

关键字: SpatiaLite, PostGIS, 算法，空间数据处理，并行和高性能计算

# 一、简介

多年来，空间对象数据库一直用于存储、管理和分析空间数据。使用这些比使用传统文件资源（例如，在 shapefile 或其他文件中收集的数据）更有效。 除了数据集更新操作之外，使用此类数据库执行空间分析也更有效。当代空间数据库不仅仅是数据库：由于查询语言（例如 SQL）的存在，它们也构成了一个成熟的空间分析应用程序。

为了显着提高使用此类数据库执行空间分析的效率，引入了基于相互对象关系的空间对象索引系统。尽管使用了对象索引系统，但大型空间数据集的分析对于单台计算机来说是一项艰巨的任务，因此已经创建了特殊的并行处理平台来处理大型数据集（例如，MapReduce、Hadoop等框架）。数据的处理和分析基于将计算分配给许多单独的计算机（即计算机集群），从而使基于数万亿对象的分析能够在可接受的时间内完成。分布式平台还可以分析空间数据。例如，分布式计算系统用于管理和分析大型气候空间数据集和执行地震信号分析。然而，使用这种系统在经济上并不总是可行的，较小的数据库可以在一台计算机内成功处理。

本文提出的研究的主要目的是测试多核处理与 SQLite/SpatiaLite 和 PostgreSQL/PostGIS 数据库中的空间分析相关的单个 SQL 查询的可能性。 因此，使用 Python 语言编写了一种算法，该算法在单个数据库中调用并行 SQL 查询。 通过区分分析中涉及的不同类型的对象、它们的数量和使用的线程数来评估最终解决方案的性能。 此外，还评估了存储数据库的数据存储介质的速度是否对并行处理速度有影响。 SQLite/SpatiaLite 和 PostgreSQL/PostGIS 数据库的以下特性促使作者使用它来评估所提出算法的性能：

* 易于使用，可以使用标准 SQL 查询和函数；
* 不需要分布式计算机网络；
* 在多台计算机之间划分数据不是问题（例如，在分布式系统中，很难在多台计算机之间划分数据以检查位于整个区域中的空间对象的拓扑关系）；
* 与环境集成的可能性，用于在开放式 QGIS 应用程序中可视化空间查询结果；

在本文的其余部分安排如下。第2节简要总结了与所提出算法相关的先前研究，第3节讨论了用于开发算法的方法。第4节描述了为评估我们提出的方法的性能而进行的实验，然后第5节讨论了这些实验的结果。最后，第6节总结了我们的介绍并提出了未来工作的途径。

# 二、相关工作

在各种应用中使用多线程并涉及空间数据库的研究已针对各种目的进行，从数据存储本身（即提高数据导入/导出的效率）到数据分析。 此外，一些已发表的研究涉及基于图形处理单元 (GPU) 解决方案的多线程处理的使用。 然而，大多数解决方案都是基于使用 MapReduce 框架的分布式处理，例如 Hadoop 和类似系统。

许多已发表的研究中都可以找到关于数据库中多线程空间连接的研究，这些研究描述了实现并行算法可以获得的好处.例如，Papadopoulos和Manolopoulos (2003)描述了在无共享架构中批量加载索引结构和空间数据的并行算法的实现，展示了为此目的使用多核处理器的好处。另一方面，Kim等人。 (2013)提出了R树遍历算法，该算法可以根据位置搜索对象，并注意到中央处理单元 (CPU) 线程数量的增加与查询执行性能的提高相关；但是， 24个CPU内核上运行的查询仅比在两个 CPU 内核上运行的性能提高了 3倍。Dai（2009）的博士论文也提出了优化空间索引的方法，讨论了基于R-tree索引和线性空间索引的并行查询问题。

由于GPU上的并行处理能力要大得多，因此人们对其在数据库处理中使用的兴趣也有所增加。谢哈布等人。(2017)描述了如何使用Microsoft SQL Server的示例将SQL 查询拆分为关系数据库中的较小部分，其中使用 GPU 可以将性能提高 39 倍。张等人。 (2014)提出了一种解决方案，旨在加速以线性形式对有关出租车上车地点和街道的数据聚合进行分析。另一方面，是 Nvidia CUDA 计算平台的一个例子，Simion 等人。 (2012) 表明查询处理速度在个别情况下可以从62倍提高到318倍。GPU的并行计算，例如 PostgreSQL数据库扩展，也可以支持三维对象数据库（Real and Silva，2018）。虽然使用基于GPU的解决方案在并行处理中产生了最大的好处，但使用所有可用的 CPU 线程也为数据库系统带来了巨大的性能优势。

雷等人。(2013)介绍了使用Amazon EC2实例在PostgreSQL/PostGIS中进行并行空间数据分析。他们观察到使用64核（在16个4核EC2节点上）的最佳情况下速度提高了63.4倍，但在最坏情况下，他们观察到速度仅提高了7.3倍。他们对所有节点使用空间数据的空间去簇。一个名为Niharika的基础设施创建了平衡的数据空间分区，对于所有节点/核心上的最佳负载非常重要。

SQLite是一个关系数据库管理系统，也与并行组件一起使用。Limkar和Jha（2019年）描述了空间数据（即物联网生成、加密）并行计算的框架。他们建议使用并行构建索引R-tree、R+-tree和R\*-tree方法进行索引。阿洛马里等人。(2019)介绍了SQLiteXTS并行数据库加密系统，描述了使用SQLite数据库系统在移动设备上进行的并行数据加密。Kai(2016)讨论了基于共享内存的锁管理器，它有效地允许编写并发线程，并描述了多核的使用如何使每分钟的事务处理率从47%增加到110%（取决于CPU频率）。克雷默等人。(2017)提出了对名为CuDB的SQLite的升级，它具有混合CPU/GPU处理，在对未索引表的子字符串搜索中，提供的速度比使用SQLite获得的速度提高了400倍。

根据上面示例中提供的结果，SQLite和PostgreSQL适用于多线程计算。

# 三、方法

在本节中，介绍了如何处理标准SQL查询的多线程处理。第3.1节描述了如何根据空间索引系统、SQL LIMIT和OFFSET子句拆分数据。下一节3.2介绍了如何针对两个空间数据库系统（SQLite/SpatiaLite和PostgreSQL/PostGIS）执行基于空间索引的查询。本节还描述了所使用的Python算法的主要部分。

3.1 拆分输入数据

在制定所提出的算法时，作者面临的主要问题是将查询输入数据分成子组，以便由单独的处理器内核进行处理。一般来说，这个过程并不简单，必须有效地进行。数据拆分的一个例子是分组，它可能是有效的，但原则上仅限于点数据。使用平铺进行数据拆分是并行操作大型栅格数据集的流行方法。

并行处理分布的另一种数据拆分方法用于基于范围的分布式处理，例如，使用Hadoop平台上的MapReduce实现。但是，这种方法需要在硬盘驱动器上保留部分数据库或将其存储在易失性存储器中。在不适合并行处理的简单空间数据库系统中，这种约束是数据库更新的重大障碍。基于空间编码的方法是在Hadoop平台上划分大空间数据的另一种方法，与随机抽样相比，可以提高查询性能。

作者选择的策略是基于空间索引对预选分析的记录数进行均等划分。

公开可用的数据库系统包含不同类型的此类索引系统。例如，IBMDB2系统采用了两种对象索引方法：一种使用空间网格索引用于平面数据，而Geodetic Voronoi索引用于球形投影中的大地数据。PostgreSQL/Post GIS和Oracle Spatial系统使用基于边界框的分层R-Tree空间索引系统。SQLite/Spatia Lite系统使用Virtual Spatial Index，它是强大的R\*-tree系统的开发。还创建了专有索引系统，例如基于并享受在SSD驱动器上保存数据库的好处的索引系统（例如，称为e FIND的高效框架系统和闪存优化并用于处理无线通信环境中的高效空间查询）,分析BIM数据,或在Oracle Spatial中管理LiDAR数据。每个索引系统在创建时间、对原始数据的引用数量和操作速度方面都有所不同。

基于对象XY位置的空间索引是最常用于加速查询的方法。R\*-tree索引代表了R-tree索引的改进，由N.Beckmann、H.P.Kriegel、R.Schneider和B.Seeger开发。基于R\*-tree的空间索引由空间对象的分层排列的边界框（最小和最大XY）构成（图1）。通过由边界框构成的树，可以快速找到具有可能关系（例如，相交）的对象对。R\*-tree之所以吸引人，是因为它高效，它同时支持空间数据和点，并且其创建时间略高于其他R-tree变体。

通过对数据集的不同部分调用相同的SQL查询来执行并行处理。在所提出的算法中，这些部分使用SQLLIMIT和OFFSET子句进行拆分，从而可以选择数据集的特定部分，而无需进行磁盘写入操作。此外，要分析的记录初步受到SQLite/SpatiaLite和PostgreSQL/PostGIS数据库中实施的空间索引的限制。

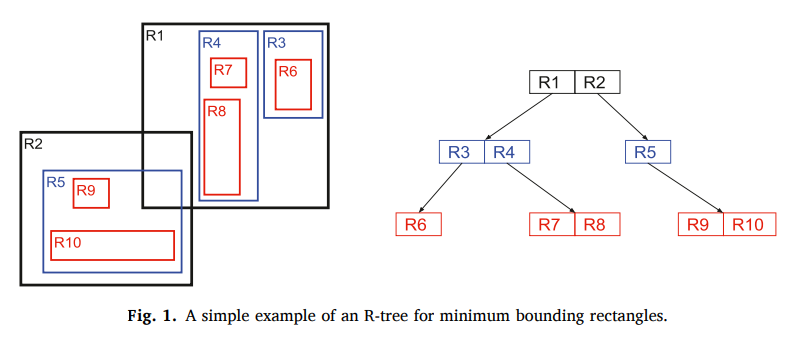
由于SQL查询的并行执行，所提出的方法以变量的形式在单独的数据集中返回SQL查询的结果，这些变量可以组合成一个完整的数据集并保存到磁盘中的单独文件或参与正在执行的分析。

3.2 多核 SQL 空间查询算法

作者提出的算法是用Python编程语言编写的。要在具有SQLite/SpatiaLite数据库的计算机上运行算法，需要安装名为mod\_spatialite(SpatiaLite,2017b)的sqlite3库扩展(SpatiaLite4.4.0)。在具有PostgreSQL(9.6)PostGIS(2.3.0)数据库的计算机上，需要安装psycopg2库扩展。

算法的第一部分确定有多少对象将由分析的单个线程处理，这个数字可以通过计算所有对象并将结果除以用于分析的预期线程数来确定。

对于基于单层空间对象的分析（例如，缓冲、面积计算、长度等），使用COUNT(\*)函数计算层的行数就足够了。在基于两层拓扑的分析中（例如，intersects()、touches()、within()、crosses()等），行数是通过仅基于空间的初步选择来估计的。指数。在SQLite/SpatiaLite中使用空间索引的示例如下：



**SELECT** \*

**FROM** firstlayer

**INNER** **JOIN** secondlayer **ON** secondlayer.ROWID **IN** (

**SELECT** ROWID

**FROM** SpatialIndex

**WHERE** f\_table\_name = sencodlayer **AND**

search\_frame = firslayer.Geometry)

在 PostgreSQL/PostGIS 中使用空间索引的示例如下：

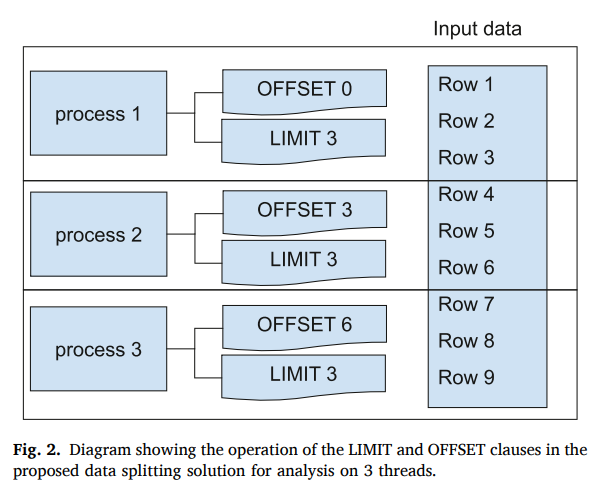
**SELECT** \*

**FROM** firstlayer **JOIN** secondlayer **ON** firstlayer.geom && second.geom

选择创建了一组新的对象对，可以通过空间函数进行逻辑测试，并且对象对可以通过定义分配给单个进程的对象数量限制在所有进程中平均分配为如下：

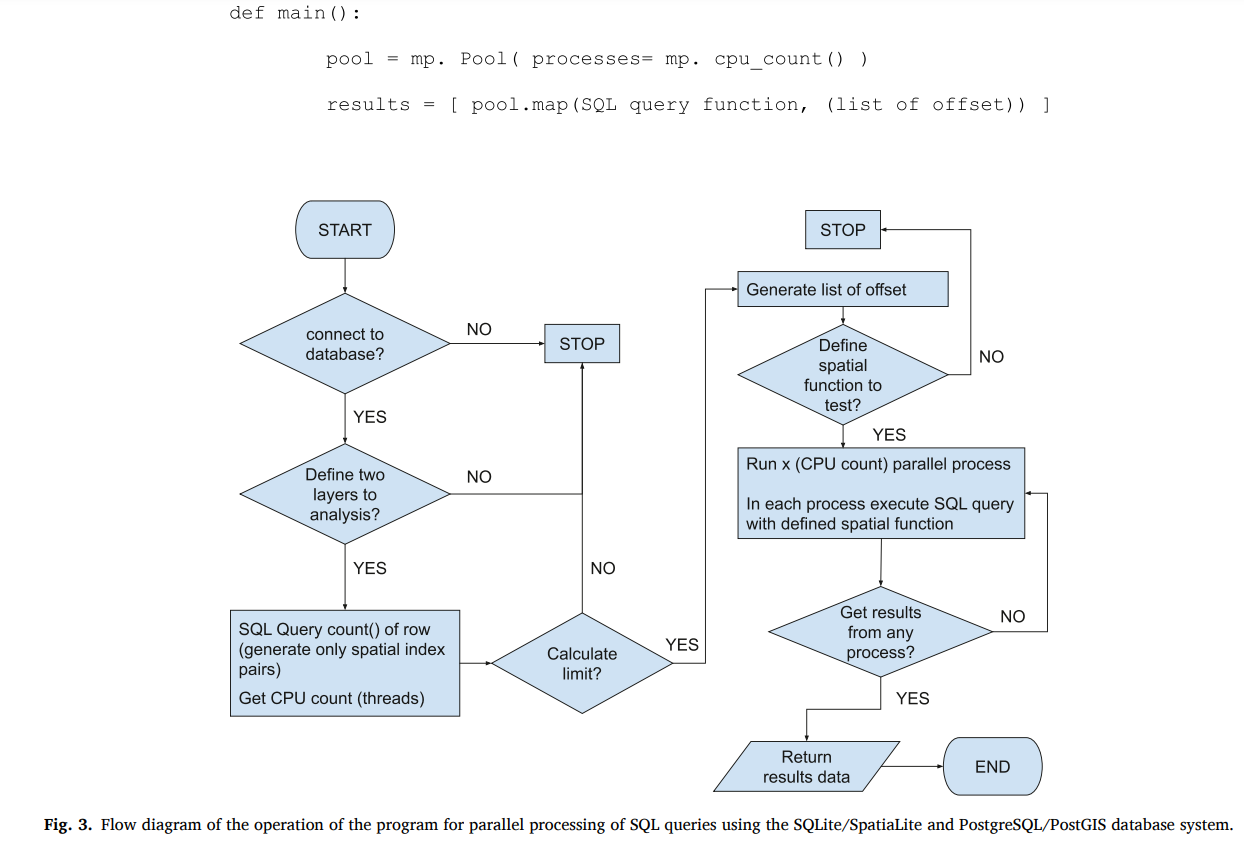
由于除法导致的非整数值的出现以及集合中所有对象对的使用所需的确定性，值1被添加到计算的数字中。

分析的各个线程的OFFSET值是根据LIMIT值计算的；第一个进程的第一个值为0，对于第二个进程，第一个值增加LIMIT值，对于后续的n个进程，第一个值增加(n-1)\*LIMIT值。图2显示了由九个元素组成的数据集中三个并行过程的OFFSET和LIMIT子句操作的说明性示例。



对准备分析的部分数据集执行SQL查询的并行处理是由Python多处理包（Python文档，2019年）执行的。multiprocessing包允许远程和本地并发并使用子进程而不是线程。多处理模块允许在给定计算机上充分利用多核处理器（以及多个处理器）。使用该包的代码采用以下形式：

结果以对象或值列表的形式存储在results变量中（取决于SQL查询类型）。图3显示了所开发算法的流程图。



整个程序代码仅限于主要函数：row\_count()–根据其拓扑计算要分析的所有对象对的数量；gen\_list\_offset()-将分析拆分为线程的行偏移数；和main\_query()-主SQL查询的函数。程序代码可在GitHub平台(https://github.com/MateuszIlba/SpatiaLite-parallel-processing,https://github.com/MateuszIlba/PostGIS-parallel-processing)的开放许可下获得，并且完整的代码在附录A和C中给出。

# 四、实验评估

4.1 实验数据

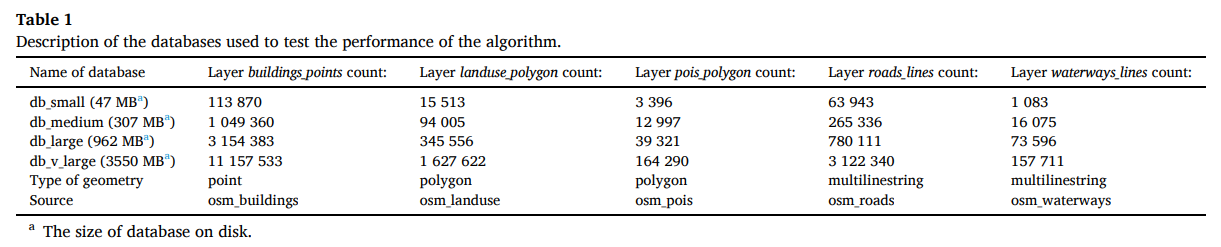
为了测试该算法，数据库是根据波兰公开可用的OpenStreetMap数据创建的（Ilba，2020年）。处理性能使用四个不同大小的数据库进行测试，其结构如表1所示。测试是在Windows系统上在配备i5-8400处理器（6核，2.8GHz，3.8GHzboost）的计算机上进行的具有16GB的RAM。数据库存储在GoodRAMIRDMPro256GBSSD驱动器上，数据以\*.shp文件的形式从Geofabrik(2018)获得。

数据库包含来自该地区的数据：

* db\_small - 克拉科夫市的区域；
* db\_medium - 整个波兰省；
* db\_large - Malopolskie、Podkarpackie 和 Dolnoslaskie省份；
* db\_v\_large - 覆盖整个波兰国家。

4.2 性能测试方法

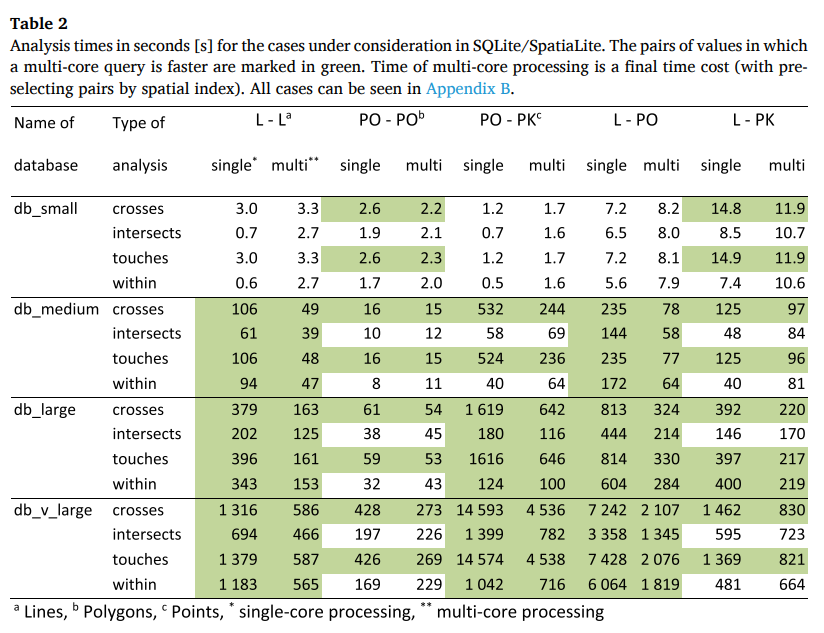
这些测试是根据在 Jackpine 基准测试中执行的测试选择的，其中比较了空间对象数据库。 选择了基于对象拓扑关系的空间查询——不同层组合之间的 intersects()、touches()、within() 和 crosses()：lines -lines (layer1 = 'roads\_lines', layer2 = 'waterways\_lines')， 多边形 - 多边形 (layer1 = 'landuse\_polygon', layer2 = 'pois\_polygon'), 多边形 - 点 (layer1 = 'landuse\_polygon', layer2 = 'buildings\_points'), 线 - 多边形 (layer1 = 'roads\_lines',layer2 = 'landuse\_polygon') , 和线 – 点 (layer1 = 'roads\_lines', layer2 = 'buildings\_points')。



查询执行时间是从查询开始到得到满足特定空间关系的对象数量（即Count）的形式来测量的。测试没有考虑在数据库中记录分析结果所花费的时间，因为在SQLite数据库中并行记录是不可能的。相反，分析结果必须记录在单个线程中。加速由以下公式计算：

因此，与传统SQL查询相比，低于1的值表示速度下降。

4.3 性能结果



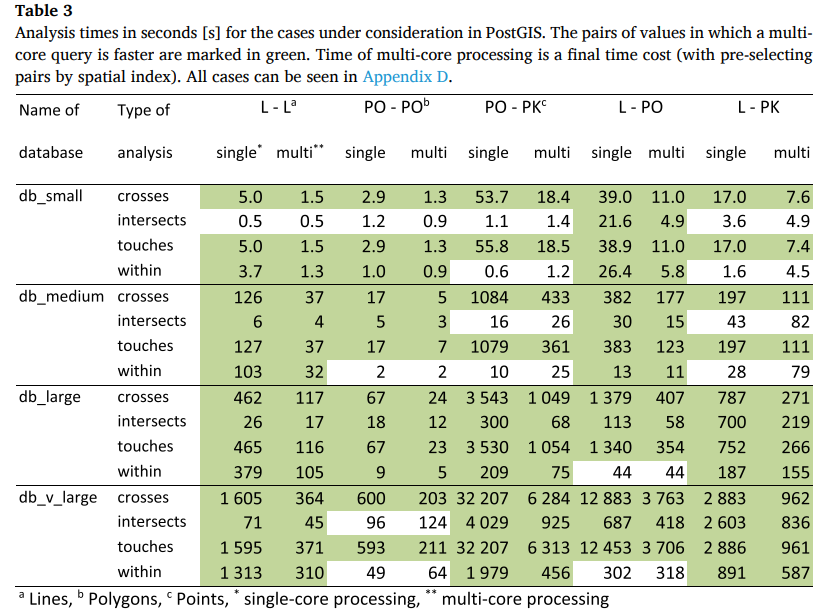


表2和表3显示了基于单核和多核查询处理的所有分析时间，时间值适用于整个查询。在多线程处理的情况下，还包括基于空间索引的查询（以确定限制和偏移量）。

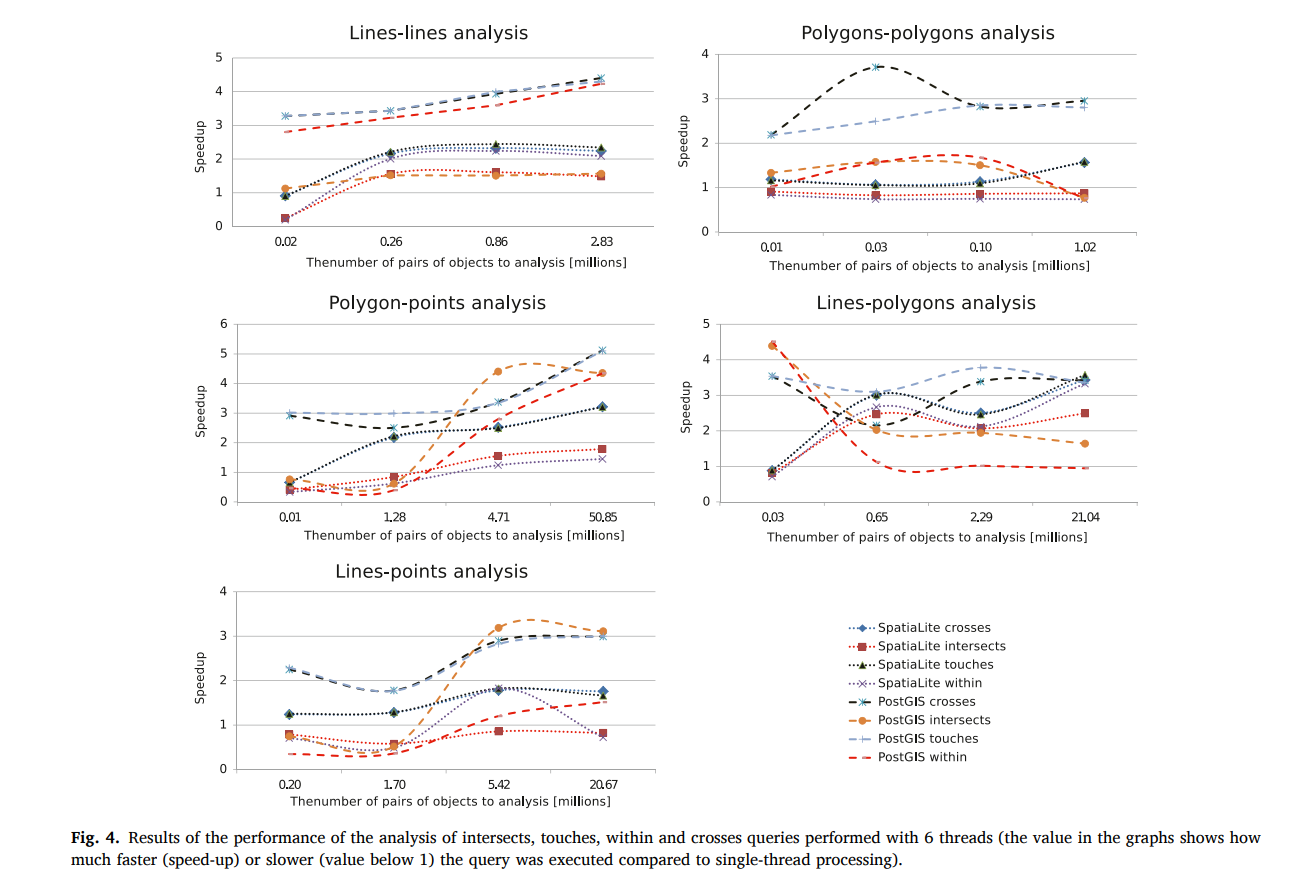
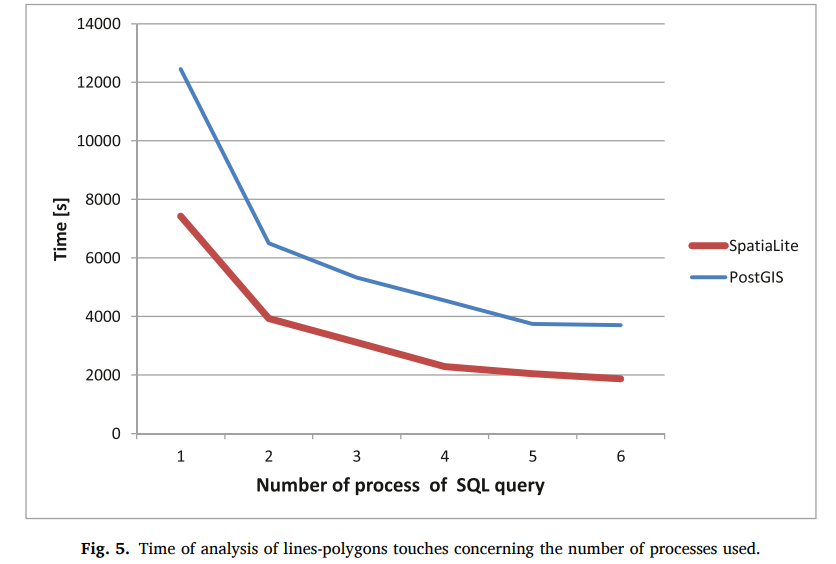
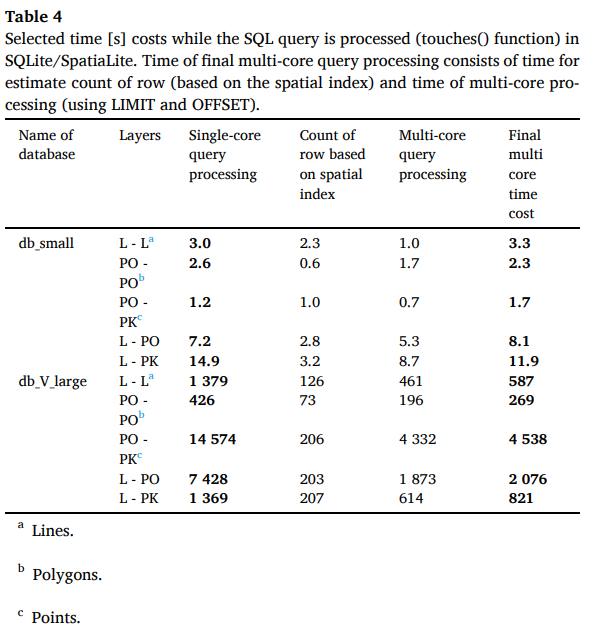


图4以数值的形式显示了性能测试结果，表明与传统的单线程处理相比，使用所有处理器内核（六个线程）时性能增加或减少了多少。用于图表的值是通过将单线程处理时间除以多线程处理时间获得的。

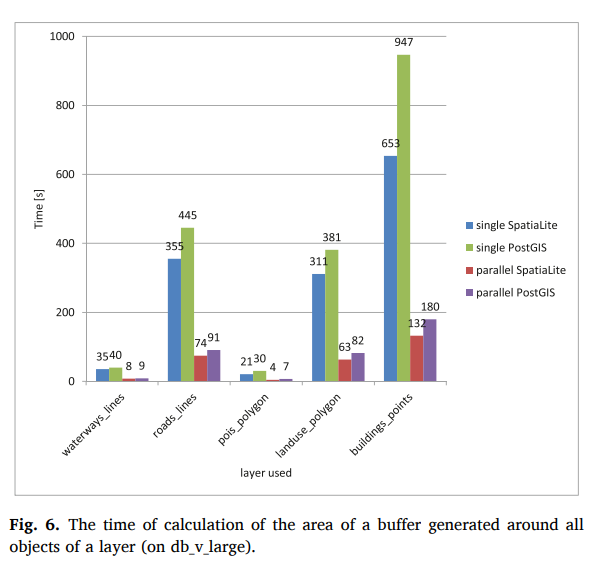


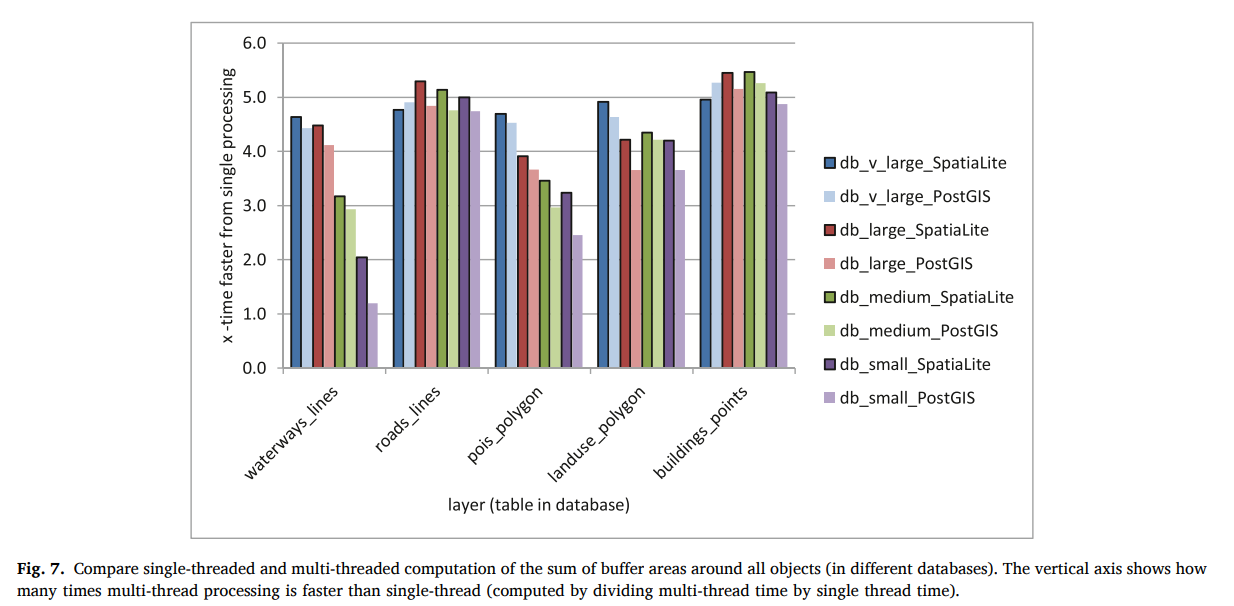
研究人员还检查了线程数对提高分析速度的影响。为此，选择了线多边形接触分析，并更改了该分析期间使用的线程数（图5）。

表4显示了在SQLite/SpatiaLite数据库上运行的算法的不同部分所选择的时间成本（以秒为单位）。附录B和D显示了所有测试的所有结果。该算法的第一部分是一个函数，它估计要受到LIMIT和OFFSET子句的数据，它基于应用COUNT()函数来获得空间连接中的所有对（仅基于空间索引）。该算法的第二部分实现了所有核并行的选定SQL查询的数量（使用LIMIT和OFFSET子句拆分指定拓扑测试的数据）。



还进行了一项测试，该测试包括仅基于一层的分析、层周围缓冲的生成以及其表面的计算。对于最大数据库的五层中的每一层，6线程分析的性能提高了单线程分析的五倍（图6）。还分析了数据库大小的影响，如图7所示。





研究人员检查的最后一个元素是数据存储设备（即保存SQLite/SpatiaLite数据库的位置）对多线程处理性能的影响。最大的数据库保存到HDD（512GB，5400rpm，高达100MB/s读/写）、SSD（256GB、SATA3、MLC，高达500MB/s读/写）和一个RAM磁盘（4GB，最高10000MB/s读/写）。在db\_v\_large上测试了查询处理的速度，并使用了两层的相交查询，即landuse\_polygon和building\_points。处理速度的差异可以忽略不计，如下所示：HDD–769[s]、SSD–775[s]和RAM磁盘–769[s]。

# 五、讨论

多线程处理性能在SQLite/SpatiaLite上处理带有多边形对象对的线的touches()查询时，分析时间最多加快3.6倍，在处理crosses()和touches()在PostgreSQL/PostGIS上查询具有点对象对的多边形。Tarmur和Ozturan¨(2019)介绍了使用ApacheSpark在区域（空间连接）中分类点的工作。使用空间函数和云计算，与MicrosoftSQLServer相比，他们的速度提高了1.6到4.5倍。需要空间拓扑的任务的性能提升接近于研究人员在并行SQLite/SpatiaLite和PostgreSQL/PostGIS空间连接（例如，点和多边形空间交叉点）中的结果。

还测试了线程数（范围从1到6）对分析加速的影响（图5）。随着用于分析的线程数增加，性能明显下降。此外，在STARK、Geospark和Spatial-Spark系统中对大量线程（即8、16和24）进行的测试显示性能略有提升。对这种非线性性能提高的解释被假设为单个内核之间计算任务的不均匀分布。为分析空间关系而创建的对象对组通常不是同质的；更复杂的几何对象集群被分配给单个计算线程，导致它们承受更大的负载。研究人员通过观察各个线程完成任务的不均衡来证实了这一假设，其中一些线程在整个分析时间过去一半后完成了任务。此外，在分析结束时，一个线程总是加载最多的复杂对象。这个问题的解决方案可能在于采用不同的方式来拆分要分析的对象。

研究人员还对基于处理单层的分析进行了测试，以计算该层所有对象周围生成的缓冲区的面积（图6和图7）。作为基准测试的结果，所有六个处理器线程的使用将性能提高了最多单个处理的5.5倍。郭等人。(2020)提出了在传统GIS平台（例如ArcGIS、MapGIS）中实现并行缓冲区功能，与单线程相比，六个线程的最大增加约为5.5倍。研究人员的结果是一样的，但是，除了生成缓冲区之外，他的分析还计算了它的面积。

由于数据库读取操作次数的增加，当使用慢速HDD与非常快的RAM磁盘时，分析时间是否会发生显着变化。但是，用于分析的SQLite/SpatiaLite数据库的磁盘类型不影响分析时间；也许使用更大的数据库可以显示访问速度和保存数据库的存储设备的吞吐量之间的差异，因为存储设备之间的差异在其他应用程序中很明显。

实现所提出算法的限制是可用于将结果存储在Python变量中的RAM内存量。如果SQL查询的结果表大于可用RAM，则会观察到速度变慢，因为系统开始将内存交换到磁盘。

PostgreSQL/PostGIS数据库可用于云计算，可以使算法适应分布式环境。所提出的算法可以允许将数据库拆分为更少的数据分区，具体取决于在云中工作的计算机数量。

# 六、 结论

本文介绍了一种多线程使用SQLite/SpatiaLite和PostgreSQL/PostGIS数据库进行空间关系分析的方法。使用Python编程语言显着提高了空间分析的速度。基准测试的结果显示，与传统的单线程处理相关联的SQLite/SpatiaLite和PostgreSQL/PostGIS的加速时间分别增加了3.6倍和5.1倍，并且在小型数据库的情况下（即到数千个对象），由于在SQLite/SpatiaLite上将数据进程拆分为单独的线程所需的时间，性能下降。测试还表明，关于空间关系的SQL查询处理速度的提高并不是使用的计算线程数的正线性函数。也就是说，使用的核心越多，生产率的提高就越小。

对比较性能的图表（图4、表2、表3）的分析表明，随着分析中使用的对象数量的增加，性能会系统地提高。与获得的单线程处理相比，db\_v\_large数据库可以最大程度地提高并行处理的速度。在使用线性层和线和点进行分析的情况下，性能增益不再随着使用db\_large数据库中的对象而增加，并且分析对象数量的增加不会导致多核处理性能。在多边形与多边形分析、多边形与点分析和线与多边形图层分析的情况下，性能的提高伴随着参与分析的空间对象数量的增加。

对于小型SQLite/SpatiaLite数据库，多线程处理的引入导致几乎所有测试中的处理性能下降，除了touches()和crosses()成对分析：多边形与多边形和线与点对象，其中一个小的观察到性能有所提高。这是由于将要分析的对象拆分为单独的线程（准备合并对象对和计算对数）所需的时间。在db\_large和db\_v\_large（大型数据库）的情况下，与查询处理时间相比，此时间的比例很小。

大部分时间用于记录的初步计算（表4，附录B）以将数据划分为单独的处理线程（即，用于LIMIT和OFFSET命令的数据）。但是，这种初步计算允许将查询分解为单独的处理器线程，而无需拆分数据库。

在小型PostgreSQL/PostGIS数据库上引入多线程处理导致SQL查询的平均速度提高了2.3倍，使用线和多边形以及空间函数intersects()的速度提高了4.4倍。与SQLite/SpatiaLite相比，小型数据库中对象对的count()的初步计算要快得多。与SQLite/SpatiaLite数据库系统相比，PostgreSQL/PostGIS上的查询处理消耗了更多的时间。例如，使用标准单核查询对db\_v\_large进行带点crosses()的多边形分析消耗32207[秒]（在SQLite/SpatiaLite上为14593[秒]），使用par消耗6284[秒]等位基因（6核）查询（在SQLite/SpatiaLite 4536[秒]上）。但是，PostgreSQL/PostGIS（单处理与并行处理）的性能提升通常比SQLite/SpatiaLite大。

作者在进一步改进 SQL 空间查询中的多核处理的工作中，将改进将要分析的数据拆分为单个线程的方法。 改进为空间关系分析准备的数据分布将有可能提高性能，因为它将允许在整个分析过程中更好地使用 CPU 内核。

**附件2：外文原文**