高性能并行及分布式地理空间大数据处理技术和平台及其在智慧城市中的应用

张健挺，纽约城市大学城市学院计算机系，2015-11-24

引言

在过去的十年里，以智能手机为代表的具有定位功能的移动设备产生了大量的空间及轨迹数据。这些数据在和城市基础信息空间匹配后可以在个体，群体，城市及区域甚至全球尺度上揭示人类移动模式。空间数据处理技术对于城市研究，交通规划和基于空间的服务等方面具有不可替代的作用。

经过近半个世纪的发展，以GIS和空间数据库为代表的地理空间数据处理技术在处理中小尺度数据集上已经相对成熟。这些技术绝大部分基于单计算节点（单机）和单CPU，涉及的算法往往高度串行化。已有的实现很多是基于二三十年前的计算机体系结构和计算模型（cost model），无法充分利用当前大内存，异构，多尺度高度并行化的硬件体系。一种经常出现的情况是，尽管从市场购买的计算节点配备大内存，多CPU并具有矢量处理器(Vector Processing Unit, VPU)和可以作为通用计算的图形处理器(Graphics Processing Unit, GPU)，许多商用和开源软件只能低效地利用一个CPU核，从而造成极大的计算资源浪费。

从另一个方面，基于市场驱动的商业数据（主要是关系数据）处理技术在过去的十年间得到极大的发展。MapReduce及其Hadoop开源实现极大的简化了并行化和分布化传统关系数据处理系统的技术瓶颈。新一代的大数据处理系统如Impala和Spark可以充分高效地利用基于内存的数据处理技术从而大大提高系统的性能。但是这些系统尚不能处理非关系数据，包括地理空间数据。尽管少数先进大数据处理系统（如Impala）开始使用VPU（基于SSE SIMD Intrinsics）以处理少数特定运算（如字符匹配）并在性能上取得巨大成功（基于常用基准数据集如TPC-H)，这些高性能数据处理技术尚未被引入空间数据处理。

张健挺团队在国际上较早从事基于通用计算图形处理器（GPGPU）的高性能大规模地理空间数据处理技术。该研究获得美国国家基金会（NSF）为期四年的研究资助（2013-2017，45万美元）。由于工业界在GIS和空间数据库研发方面处于领先地位，在经历90年代地理空间技术大发展后，NSF已多年未资助常用算法和系统层次的矢量地理空间数据处理研究。由于成功地把握计算机软硬件发展趋势，地理空间大数据的需求和我们的先期工作，我们的研究项目书在经历激烈竞争后（当年CISE/IIS中尺度核心项目标书成功率在5%-10%之间），成为已知的首个NSF资助的高性能地理空间大数据研究项目。

在过去的两年里我们已将基于单计算节点的高效并行空间数据结构和算法扩展到分布式系统并在Amazon EC2 g2.2xlarge 云计算环境下取得满意的实验结果。初步结果表明，在单机环境下，相对于基于传统开源软件的实现，在充分利用大内存，GPU加速和并行数据结构和算法改进后，我们的技术可以提升3-4个数量级（从小时级到秒级）。在分布式环境下，由于我们高效的单机处理技术，我们的系统显著领先以明尼苏达大学SpatialHaoop为代表的最新相似系

统。这些研究结果经同行评议后已在多个专业期刊，国际会议及研讨会上发表（参见论文列表[1]，大部分文章可以以PDF格式下载全文或其技术报告版本）。2014年底，我们应ACM SIGSPATIAL Special 编辑部的邀请，摘要介绍了我们在2014年中以前的工作并以”Large-Scale Spatial Data Processing on GPUs and GPU-Accelerated Clusters”为标题发表于该期刊（6(3):27-34[2]）。在过去的一年里，我们在分布式地理空间大数据处理方面又取得一些很好的进展。下面我们在总结已有工作基础之后，以智慧城市大数据应用为背景，在算法，系统和应用层次分别探讨一些研究方向。

已有工作总结

第一阶段2009-2011：单计算节点栅格空间数据编码和索引

传统的高性能计算技术主要包括基于共享地址空间的矢量计算和基于零共享的分布式计算。经过几十年的发展，这些原为超级计算机设计的并行和分布式技术已逐渐在商用计算机上以不同的形式得以实现并发展。从另一个方面，新一代的超级计算机也更多的采用商用计算机作为通用计算节点以降低软硬件成本并被更广泛地使用。

以Nvidia CUDA为代表的通用计算图形处理器（GPGPU）技术出现于2007年并迅速在计算机视觉，医学图像处理和数值模拟计算方面得以应用。这主要是由于这些领域的算法具有较高的并行度并能相对容易地移植到新的硬件体系结构下。我们的初期工作（2009-2010）主要是借鉴这些已有工作并将其应用于栅格地理空间数据管理。在“Indexing large-scale raster geospatial data using massively parallel GPGPU computing”一文中[3]（ACMGIS 2010 会议短文），我们为GPU设计了一种基于MinMax 四叉树的并行数据结构并提出了基于CUDA的并行建树的算法。尽管该技术较传统基于CPU的技术有显著提升并极大的鼓舞了我们的信心，我们后来意识到该实现在GPU内存寻址效率方面有较大缺陷。经过改进后，该技术在效率和代码易读性上都得到极大的提升，并以“High-Performance Quadtree Constructions on Large-Scale Geospatial Rasters Using GPGPU Parallel Primitives”为标题发表于IJGIS（27（11）：2207-2226[4]）。

一个相关但不同的工作是在GPU上对栅格数据进行基于四叉树的编码以同时支持压缩和查询。在“Parallel quadtree coding of large-scale geospatial data on GPGPUs”一文中（ACMGIS 2011 会议短文[5]），我们先将栅格数据在位平面（bitplane）上分解然后对每一个位平面的二进制位图（bitmap）建树。与传统四叉树技术不同，该技术并不存储子节点的位置从而大大节省存储开销。在从上到下逐层树解码时，该技术利用GPU大量线程去并行解码计算。在到达最后一层后，GPU线程被应用于合并树结构和底层非零象限

---

[1] http://www-cs.ccny.cuny.edu/~jzhang/Publications.htm
[2] http://dx.doi.org/10.1145/2766196.2766201
[3] http://dx.doi.org/10.1145/1869790.1869859
[4] http://www.tandfonline.com/doi/abs/10.1080/13658816.2013.828840
[5] http://dx.doi.org/10.1145/2093973.2094047

（quadrant）。该设计简洁明了并可应用于多核CPU。在”Quadtree-Based Lightweight Data Compression for Large-Scale Geospatial Rasters on Multi-Core CPUs”一文中（IEEE Big Data Conference 2015 会议短文[6]），使用NASA SRTM 30米DEM数据，我们评估了该技术在多核CPU上的性能。我们已改进该设计的GPU实现并将在期刊文章以详细介绍该技术并对比它在GPU和多核CPU上的性能。

第二阶段2011-2013：单计算节点矢量空间数据索引和查询

在2010年底，我在CCNY土木系研究交通的同事要求我帮助处理近两年全纽约的出租车纪录数据。这些包括起始和终止位置的空间数据引起我极大的研究兴趣。在尝试应用已有商用和开源软件去将这些空间位置匹配不同街道（polyline）后，我们发现处理全年约1.7亿（170 million）个纪录需要数十个小时，远远大于交互式查询能够承受的范围。这个实际需要启发我们去设计和实现基于GPU加速的矢量数据空间索引和查询功能。基于已有积累的GPU经验，我们很快设计和实现了在GU上基于网格的空间索引(spatial indexing)和基于二叉树搜索的空间过滤（spatial filtering）技术。尽管空间索引和空间过滤都是基于并行原语（parallel primitives），最后一步空间提炼（spatial refinement)的实现是基于原生CUDA编程以尽可能提高效率。该技术的初步结果发表于2012 ACM SIGKDD 在北京的UrbComp研讨会[7]和2012 ACM CIKM 在夏威夷DOLAP研讨会[8]。经过显著扩展后，这项研究发表于Information System (Elsevier)期刊（vol 44:134-154[9]）. 需要指出的是，尽管我们使用的2009全年出租车数据在当时尚未公开，2013及以后的数据现在已经可以公开下载，这为我们更全面的检验并改进该技术提供了方便。

四叉树和R树在传统矢量空间索引技术研究中具有重要的地位。根据为栅格数据开发的基于GPU的MinMax四叉树经验，我们设计并实现了基于GPU的点数据四叉树。在设定每个象限允许的最大点数目之后，该技术自上而下逐层分解点数据集直到每个叶节点代表的象限都符合要求。在每一层上，该技术使用包括sort，scan，reduce，gather/scatter在内的并发元语去寻找并组织叶节点来建树。由于并发元语言可以在包括GPU在内的多种并行硬件平台上高效运行，该设计在高效性和跨平台性之间有很好的折衷。同样的思路被后来应用于基于GPU的R树技术。在集成点数据四叉树和基于GPU的点在多边形判断之后，我们的技术被应用于纽约出租车数据以计算每个起始/终止点落入的多边形（包括行政区划，人口统计区划和高精度地籍）。该结果以”Speeding up Large-Scale Point-in-Polygon Test Based Spatial Join on GPUs”为标题发表于2012年ACM BigSpatial研讨会论文集[10]。关于R树的基于GPU的建树和查询研究以“Parallel spatial query processing on GPUs using R-trees”为标题发表于次年（2013）的

---

[6] http://www-cs.ccny.cuny.edu/~jzhang/bqtree_coding.htm
[7] http://dx.doi.org/10.1145/2390226.2390229
[8] http://dx.doi.org/10.1145/2390045.2390060
[9] http://dx.doi.org/10.1016/j.is.2014.01.005
[10] http://dx.doi.org/10.1145/2447481.2447485

ACM BigSpatial 研讨会论文集[11]。由于R树是针对非点数据（使用MBR近似线和多边形数据），在GPU上的空间查询要远较点数据四叉树复杂。我们设计并实现了基于深度遍历和基于广度遍历的GPU空间查询算法及其综合（hybrid）。实验结果表明基于广度遍历的设计一般具有较高的效率。这是由于在GPU上广度遍历具有更好的内存寻址效率和线程间的负载平衡。在和基于网格的空间索引对比后，我们发现简单的基于网格的空间索引在内存空间充足的情况下在GPU上事实上具有更高的效率。这也是我们没有进一步改进基于四叉树和R树索引技术的原因。我们希望在继续提高这三种基于GPU的空间索引技术性能的基础上对它们做一个全面的分析对比并在期刊上发表。

与基于点到路网（polyline）最近距离和点在多边形内判断的空间查询不同，我们另一项研究是基于点到多边形边界的最近距离的应用研究。使用纽约出租车点数据和地籍数据，由于每个土地块（land parcel）都有一个土地使用类型，在将每一个出租车纪录的起始和终止位置映射到最近的土地块之后，我们可以获得该纪录起止位置的土地使用类型并使用它们的组合去近似该用车纪录的目的，如（办公用地，家庭用地）的组合何以在一定程度上代表与上下班有关的用车。与点到线空间查询的结果相似，在使用GPU加速后，关于用车目的的计算时间可以从数十小时降低到数十秒。在经过简单的归类后，我们使用2009纽约全年出租车纪录数据计算了基于14种土地利用类型的用车目的矩阵（14*14）并简单分析了一些有趣的观察结果。该研究以“High-Performance Spatial Query Processing on Big Taxi Trip Data using GPGPUs”为标题发表于IEEE Big Data Congress 2014会议论文集[12]（应用类长文）。通过与城市研究人员的更紧密的合作并将该研究应用于多个城市（如北京，上海和新加坡）做综合对比分析，我们相信该应用研究适合在交叉领域期刊发表。

在2012-2013年度我们完成的另一项研究是基于GPU的复杂多边形内部索引。该技术在一定程度上是对点数据四叉树的扩展并应用于更有效地索引复杂多边形。复杂多边形具有多结点（vertex），多环（ring），不规则和复杂拓扑结构。使用单个MBR去近似多边形进而建立R树往往不具有理想的删减效果（pruning）从而降低查询效率。我们的技术将复杂多边形逐层分解成不同大小的象限并以象限为单位建立四叉树。与点数据四叉树不同，由于复杂多边形及其MBR有可能在空间上重合，因此代表分解后象限的四叉树叶结点可能在空间上重合。这就意味着每一个四叉树的叶节点对应着一个多边形ID表而不仅仅只是一个ID。我们的设计是将所有的多边形ID表组合成一个向量（一维数组）同时在四叉树节点中增加一个指向这个数组位移的域以获取与该节点相连的多边形ID表。由于新增加了一个节点的域和一个数组，新的平行建树和查询算法显著复杂。该研究经以“Data Parallel Quadtree Indexing and Spatial Query Processing of Complex Polygon Data on GPUs”为标题发表于2014年在杭州举办的VLDB ADMS（Accelerating Data Management Systems Using Modern Processor and Storage Architectures）国际研讨会[13]。值得一提的是该研讨会的程序委员会包括多名国际知名高性能数据管理领域的专家。我们非常高兴我们针对地理空间数据的研

---

11 http://dx.doi.org/10.1145/2534921.2534949
12 http://dx.doi.org/10.1109/BigData.Congress.2014.20
13 http://www.adms-conf.org/2014/adms14_zhang.pdf

究能够得到这些主要针对商业数据研究的同行的理解和认可。在已有研究的基础上，我们已设计并实现了基于扫描线（ScanLine）的复杂多边形内部索引算法并在概念上具有更高的效率。我们希望将这项研究成果尽快发表。

第三阶段2014-：分布式矢量空间大数据查询平台的研发

从2014年春天开始，我们将研究重点从单机计算转移到分布式地理空间数据的研究上来。一个重要的原因是基于Hadoop的分布式地理空间数据引起了广泛的研究和应用兴趣。University of Minnesota (UMN) Mohamed Mokbel 研究组（SpatialHadoop）和Emory University Fusheng Wang 研究组(HadoopGIS,后转入Stony Brook University)在这方面取得了很好的先期研究结果。在仔细分析他们的源码后我们认为尽管Hadoop有较大的用户群，该系统在体系设计方面存在较大缺陷。主要问题之一是为了简化并行化的复杂度并支持容错，该系统大量使用不必要的硬盘读写。同时 SpatialHadoop使用JTS，HadoopGIS使用GEOS开源软件包作为底层几何计算引擎（以判断空间关系）。JTS及其C++ 版本的GEOS距离其主要设计开发时期已近二十年。尽管长时间的维护在一定程度上保证了他们的可靠性，为二十年前计算机体系结构设计的算法和实现已很难在当前体系结构下获得最优化。我们在两个方面进行了分布式地理空间数据处理研究的尝试，一是将我们的单机并行空间索引（indexing），空间过滤（filtering）和空间提炼（refinement）算法设计与最新大数据处理系统（如Spark和Impala）集成，另一个是从底层开发轻量级分布式执行引擎以更高效内在（native）地支持空间数据处理。

我们首先选择了Cloudera Impala (https://github.com/cloudera/impala)开源软件作为第一个研究方向的尝试。这主要是由于Impala的后端基于C++并开始支持基于SSE4的SIMD矢量操作（主要应用于字符串处理）和基于开源编译器LLVM的即时（Just In Time - JIT）编译优化。这些及其它先进的特性对于我们研发一个高性能空间大数据处理引擎非常有吸引力。在经过超过6个月全力理解Impala设计及实现后，我们终于成功地实现了预期目标并开发了ISP原型系统。我们的实现以Impala行组（row batch)为单位，抽取其中的空间数据并实时建立空间索引以支持空间查询（包括多核CPU和GPU）。尽管Impala在一个安装实例（instance)内并不支持多核CPU，在克服了许多技术困难后我们设计并实现了基于OpenMP和Intel TBB的多线程并发实现。在多核CPU上，为了与我们基于数据并行的空间提炼(refinement)实现的对比，我们也开发了基于GEOS的多线程实现。基于多核CPU和GEOS的设计与实现以“Large-scale spatial join query processing in Cloud”为标题发表于2015 IEEE ICDE CloudDM 在汉城的研讨会[14]。基于GPU的设计与实现以“Scalable and efficient spatial data management on multi-core CPU and GPU clusters: A preliminary implementation based on Impala”为标题发表于2015 IEEE ICDE HardBD 在汉城的研讨会[15].ISP设计与实现的源码可以从我们的网页上下载 http://geoteci.engr.ccny.cuny.edu/isp/.

---

[14] http://dx.doi.org/10.1109/ICDEW.2015.7129541
[15] http://dx.doi.org/10.1109/ICDEW.2015.7129567

由于Spark提供的RDD矢量操作元语与我们熟知的并行元语具有高度的一致性，我们在较短的时间内实现了一个基于Spark的地理空间大数据的处理原型系统并被命名为SpatialSpark（已开源：http://simin.me/projects/spatialspark/）。与作为一个终端应用系统的Impala不同，Spark被设计成一个并行数据处理的开发环境。这使得基于Spark的开发较基于Impala的扩展容易得多。尽管运行于Java虚拟机的Spark目前尚无法本地化地使用SIMD并行计算（包括多核CPU和GPU）以充分利用硬件资源，开发基于Spark的软件需要对传统算法作出较大的重新设计以适应Scala函数程序语言编程的要求，这使得重写的程序能够更有效地表达问题的并行性并得以高效执行。我们的实验结果表明，SpatialSpark较基于Impala和GEOS的实现具有较高的运行效率。一个意外的发现是，在空间提炼时，使用基于Java的JTS往往比基于C++的GEOS快数倍。尽管一般认为基于C++的实现比基于Java的实现具有更高的运行效率，我们的初步解释是，GEOS频繁申请和释放大量小内存块导致其在当前硬件体系结构下运行效率低下。尽管通过即时申请和释放内存块以降低内存消耗在二三十年前的小内存（MB级）计算机上有其特殊意义，这种设计已远远不适用于当前的商用计算机（一般具有GB级以上内存）。大量的内存申请和释放操作开销可能远远大于实际计算开销。相反，Java使用另外的线程进行隐式批量内存申请和释放（包括garbage collection），可以带来更高的效率。这些初步研究结果发表于IEEE 2015 ICDE CloudDM 在汉城的研讨会(标题见上)。

随着对Impala和Spark系统理解和应用的加深，我们逐渐意识到它们对于空间大数据处理应用的局限性。必须指出的是，与数据库管理系统演化相似，已有的的大数据管理系统主要是针对基于关系模型的商用数据，而类似于Postgres数据库的能够兼容地理空间数据模型的大数据系统尚未出现。同时已有大数据系统往往具有复杂的代码库，理解和扩展这些系统以支持包括地理空间数据在内的新数据类型都比较困难。这启发我们从底层开发轻量级（lightweight)分布式执行引擎以高效地支持复杂数据类型。与Hadoop，Impala和Spark类似，我们的设计建立在HDFS分布式文件系统之上以兼容已有的数据集并降低数据文件的管理复杂度。在考察了主要数据通讯模块（Boost ASIO, ZMQ, Apache Thrift等）之后，我们决定采用Apache Thrift （在这方面与Impala相似）并尽可能多地支持异步操作（包括空间计算，网络通讯和硬盘读取）以提高系统效率。尽管我们的初步实现尚有很大的改进空间，在Amazon EC2 上的实验表明系统开销较ISP明显降低，尤其是在GPU加速的集群上。该研究以“Lightweight Distributed Execution Engine for Large-Scale Spatial Join Query Processing”为标题发表于IEEE Big Data Congress 2015会议论文集[16]（研究类长文）。

在独立开发了SpatialSpark, ISP和LDE之后，我们感觉有必要系统地比较我们的工作和HadoopGIS和SpatialHadoop,不仅仅是从运行时间上，更重要的是在体系设计及基础平台层面。由于我们已有的工作表明基于数据并行的设计和SIMD加速的实现的ISP和LDE显著地优于SpatialSpark，我们将重点放在比较SpatialSpark和SpatialHadoop和HadoopGIS.由于这三个系统都运行于JVM，都

[16] http://dx.doi.org/10.1109/BigDataCongress.2015.30

使用JTS作为空间提炼引擎，并都不支持SIMD硬件加速，它们在一定层度上更具有可比性。我们在仔细分析了这三个系统在数据预处理，空间分割和局部空间查询三个步骤的工作流程和异同点之后，使用公开的空间数据（包括点，线和面）和常用的空间查询操作（包括点到线距离和点在多边形内判断）对它们进行了较为系统的性能分析。分析结果表明，作为一个较早开发的系统，HadoopGIS采用了Hadoop Stream（流）模型将空间查询功能集成到Hadoop中。尽管使用流模型可以较为容易地集成已有的GIS模块并将串行的设计和实现在粗粒度上快速并行化和分布化，额外的MapReduce工作单元（job）和大量的磁盘读写大大降低其应有的效率。相反，SpatialHadoop从底层集成了空间索引和空间查询功能并设计了高效的HDFS分布式文件结构从而大幅度降低了非必要系统开销因而具有较高的效率。由于SpatialSpark采用基于内存的处理，基于Scala编程语言和Spark框架结构更有效地表达和执行空间操作（目前以查询为主）中的并行性，因而当集群系统内存充足的情况下，它往往具有更高的效率。我们的实验也表明，SpatialHadoop对系统内存需求较小并具有较好的鲁棒性（robustness)而SpatialSpark常具有较之数倍的高效。更详细的实验结果和讨论可以在我们发表于ICPP HPC4BD 2015北京研讨会论文集找到，标题是“Spatial Join Query Processing in Cloud: Analyzing Design Choices and Performance Comparisons”[17]。

其它基于GPU的高效空间数据处理技术的研发

Voronoi Diagram 在栅格数据和点数据之间起着非常重要的桥梁作用。内插大量点数据（如点云Point Cloud)到栅格数据（如地形高程模型DEM）在实际中有广泛应用。受到新加坡国立大学（NUS） Tiow-Seng Tan研究组“Parallel Banding Algorithm to Compute Exact Distance Transform with the GPU”一文的启发，我们认为 发表于ACMGIS 2011年的基于3D图形渲染的点数据内插技术TerraNNI可以在支持CUDA的通用计算GPU上得以更高效的实现。在显著改进 Tiow-Seng Tan研究组公布的源码的基础上，我们获得了较TerraNNI高数倍到数十倍的加速比。该研究以“Constructing Natural Neighbor Interpolation Based Grid DEM Using CUDA”为标题发表于2012 COM.GEO 会议论文集[18]。由于时间和精力的限制，尽管没有机会进一步发展该技术，我们认为该技术在点数据组织管理，在GPU上规则化不规则（变长）数据读写等方面具有很好的借鉴价值，值得进一步探讨，尤其是基于DEM的应用（如可视度visibility分析，流域和水网提取等方面）。

在我2013年夏天访问美国橡树岭国家实验室（ORNL）时，一项工作是有效地利用该实验室当时排名世界第一（现第二）的Titan超级计算机去加速生成基于全美国县域的高程直方图。一个直观的解决方式是在使用点在多边形内判断去匹配200亿（20 billion)个30米见方的栅格和近3000个复杂多边形（县域），计算强度极大。另一个直观的解决方式是按照DEM的精度（30米）栅格化县域多边形然后进行简单的直方图统计。由于高精度栅格化复杂多边形存在内存限制和并行困难，这个方法也不可行。我们的设计是将DEM分割成相对较小的

---

[17] http://www-cs.ccny.cuny.edu/~jzhang/papers/sjc_compare_tr.pdf
[18] http://dx.doi.org/10.1145/2345316.2345349

块并对每个小栅格块建立直方图。接着重复使用（长）方形在多边形内的判断GPU实现去并行地判断每个小栅格块是否在对应的多边形内。如果小栅格块全部在多边形内，我们可以通过简单合并它们的直方图以更新基于多边形的直方图。如果小栅格块与多边形相交，我们需要使用点在多边形内的判断以进一步将该栅格归并入相应的多边形直方图。该设计的每一步都可以在GPU上并行并大大降低内存开销（相对于方法二）和计算开销（相对于方法一）。在成功地将该设计在单机上实现后，我们使用MPI将其扩展到分布式计算环境下。使用8个Titan计算节点（每个计算节点配备一个Nvidia GTX Titan GPU），端到端运行时间已少于10秒。我们相信该技术可以使用更多的计算节点以处理更精细的高程数据（如1米）和高精度多边形数据（如乡镇）。该研究以“High-Performance Zonal Histogramming on Large-Scale Geospatial Rasters Using GPUs and GPU-Accelerated Clusters”为标题发表于IEEE IPDPS ASHES 2014 研讨会论文集[19]。

2015年夏天我实验室购入4个Nvidia K1 SoC (System on Chips)开发板，每个开发板（SoC）配备四核 ARM A15 CPU和192核 Nvidia Kepler GPU。由于每个开发板售价低于200美元，在添加了路由器和闪存硬盘后，我们以总额接近1000美元的价格配置了一个超小型（tiny）GPU加速的集群计算机。在我们完成LDE的设计和开发后，一个想法是将SpatialSpark和LDE都移植到这个超小型集群上并与常规集群计算机做性能比较并探讨能量消耗和性能之间的关系。除了K1 SoC只能支持32位操作系统从而不能充分利用闪存硬盘做缓存外，尽管基于ARM和基于Intel的硬件在体系结构和指令集方面存在较大差异，移植SpatialSpark和LDE都比较顺利。我们的初步实验结果表明低功耗CPU和GPU并不能显著降低能量消耗。该研究以“Tiny GPU Cluster for Big Spatial Data:A Preliminary Performance Evaluation”为标题发表于 ICDCS HPBDC 2015 研讨会[20]。

进一步工作打算

经过近六年的探讨，我们已初步在高效数据结构和并行算法层面对高性能地理空间大数据处理技术的研究形成一个较为完整的框架体系。基于这些并行设计的单计算节点和分布式处理系统显著领先同类系统。同时在学习消化领先大数据处理系统的基础上，我们从底层开发了一个轻量级分布式执行引擎（LDE）以更好地支持地理空间大数据处理技术的研发。我们下一步的工作主要集中在一下几个方面：

1）在轻量级分布式执行引擎LDE的基础上设计更高效的数据结构和算法，不断提升基于SIMD的多核CPU和GPU的空间操作的效率。对于常用空间查询算子（如点到polyline的距离，点在多边形内判断），采用公开数据作为测试数据源并定期在网上公布测试结果。这可以为该领域设置性能测试基准并吸引学术界的对比研究和工业界的关注。LDE也可以作为一个框架和平台为其它空间处理操作提供并行化和分布式化的支持。需要指出的是，由于LDE主要为研究而开发

---

[19] http://dx.doi.org/10.1109/IPDPSW.2014.113
[20] http://dx.doi.org/10.1109/ICDCSW.2015.33

设计，它的强项并不在易用性和鲁棒性。因此实验结果在和基于商业或开源系统对比时，必须将这些因素考虑进去。

2）扩展SpatialSpark以开发基于Spark的实用大数据处理系统。随着内存技术的发展，我们认为基于Spark的系统将逐渐取代传统基于Hadoop（主要是指MapReduce Runtime)的系统，尽管在近几年内可能会出现一些混合系统以逐渐适应大数据处理应用的演化。尽管Scala和Spark在可以预期的将来尚不会支持基于SIMD的硬件并行操作，我们的许多基于数据并行的设计可以较容易地扩展到Spark上。作为一个可能的研究方向，尽管扩展Spark以本地化支持非关系操作在方法和实践上都存在较大的困难（如同从关系数据库发展到对象－关系数据句库经历了漫长的时间），如何将空间操作转化为关系操作以充分利用Spark的日益优化高效的关系操作引擎可能更具有现实可行性。初步想法是将空间数据在合适的粒度上或归并（主要是针对点数据）或分解（主要是针对线和面数据）成基本单元，然后以这些基本单元的SFC（Space Filling Curve）编码（如Morton Code)作为关键字以使用关系查询算子来做非空间匹配。非空间匹配的结果可以经过排序等操作恢复空间关系以做进一步的局部空间操作。

3）高性能空间大数据处理系统在智慧城市中的应用。一个方向是在我们的基于出租车Origin-Destination (OD)数据和土地利用类型数据的用车目的分析的基础上，将大数据处理技术应用于更多的数据类型（如轨迹数据和城市功能区）以更好地分析出行目的并支持交通规划和城市研究。我们2012年UrbComp北京研讨会的另外一篇论文（题为“Smarter Outlier Detection and Deeper Understanding of Large-Scale Taxi Trip Records: A Case Study of NYC”[21]）和2013年“Big Data Techniques and Technologies in Geoinformatics”一书章节（题为“Efficient Frequent Sequence Mining on Taxi Trip Records Using Road Network Shortcuts”[22]）使用基于最短路径的合成轨迹发现一些有趣的现象值得进一步探讨。这些计算都是基于传统串行（尽管高效）算法。利用已有的工作基础将这些算法并行化和分布式化适合作为初期工作去展示高性能空间大数据处理系统在智能城市中的应用。中国大城市（如北京，上海，杭州和深圳）可能提供大量的轨迹数据，这也为开发进一步的智慧城市应用提供了可能。

4）可视化技术与地理空间大数据处理的系统集成。ACM KDD 和IEEE ICDM 两个主要数据挖掘会议系列近两三年里都推出了与可视化相关的研讨会（分别是IDEA和DAVA).地理空间大数据的管理和分析在技术上更具有挑战性，需要高效可视化的支持，从而对高性能处理提出了更高的要求。我们最近与IBM T.J. Watson 研究中心 SystemG 项目组合作在ACMGIS UrbanGIS 2015 研讨会上发表工作题为“Prototyping A Web-based High-Performance Visual Analytics Platform for Origin-Destination Data: A Case study of NYC Taxi Trip Records”的文章[23]报告了我们在这一方面的进展。初步的想法是将地理空间大数据处理技术模块化和网络化（Web Services） 以更方便地支持系统集成。

---

21 http://dx.doi.org/10.1145/2346496.2346521
22 http://www-cs.ccny.cuny.edu/~jzhang/freseq.html
23 http://www-cs.ccny.cuny.edu/~jzhang/papers/whp_od_tr.pdf