DOI:10.13203/j.whugis20140774 文章编号:1671-8860(2017)02-0163-07

基于 NoSQL 数据库的空间大数据分布式存储策略

李绍俊[1] 杨海军[2] 黄耀欢[1] 周 芹[3]

1 中国科学院地理科学与资源研究所,北京,100101
2 环境保护部卫星环境应用中心,北京,100094
3 北京超图软件股份有限公司,北京,100015

摘 要:基于关系型数据库的空间数据存储与处理是地理信息系统(geographic information system,GIS)领域的主流模式,但伴随着物联网、移动互联网、云计算及空间数据采集技术的发展,空间数据已从海量特征转变为大数据特征,对空间数据的存储和管理在数据量和处理模式上提出了新的挑战。首先分析了基于传统的集中式存储与管理模式在处理和应用大数据方面的局限性,包括存储对象的适应性、存储能力的可扩展性及高并发处理能力要求;然后在分析当前几大主流 NoSQL 数据库特点的基础上,指出了空间大数据基于 NoSQL 数据库的单一存储模式在数据操作方式、查询方式和数据高效管理方面存在的局限性;最后结合 GIS 领域空间大数据存储对数据库存储能力的可扩展性及数据处理和访问的高并发要求,提出基于内存数据库和 NoSQL 数据库的空间大数据分布式存储与综合处理策略,并开发了原型系统对提出的存储策略进行可行性和有效性进行了验证。

关键词:空间数据库;大数据;NoSQL 数据库;分布式存储

中图法分类号:P208 **文献标志码**:A

新技术的发展给空间数据存储与管理又提出了新的挑战。物联网、移动互联网和云计算技术及应用的蓬勃发展,使得空间数据在数据量和应用模式上发生了转变;此外,传感器技术的发展,使采集数据的空间分辨率和时间分辨率显著提高,导致所获取的数据规模成指数级快速上升,面对动辄以 TB(trillionbyte),甚至 PB(petabyte)计的数据,也给空间数据存储和处理带来巨大的压力[1-3]。

传统的基于关系型数据库的空间数据存储与管理已经无法满足大数据存储和处理的实际应用要求,随着互联网领域云技术、非关系型数据库技术的迅速发展,各种分布式 GIS 技术的研究成为研究热点。本文主要着眼于空间大数据的存储与管理,首先分析了传统集中存储模式在大数据存储方面的局限性;然后,针对互联网领域大数据的成功解决方案,研究当前主流 NoSQL 数据库的特点,分析其在处理空间数据时的优势与不足;最后,针对这些需求和存在的问题,本文提出空间大数据分布式存储与处理策略,并在原型系统中进行试验验证。

1 空间数据存储技术现状

基于成熟的关系型数据库设计空间数据引擎,集中存储和管理空间数据是当前应用的主流模式。

1.1 空间数据集中存储模式及其局限性

从空间数据引擎和关系数据库与应用程序结合的紧密程度来看,可以将空间数据引擎的体系结构分为内置模式、三层结构模式和两层结构模式[4,5],这些集中存储模式很好地解决了海量空间数据存储和管理的问题。

目前,各种地理空间信息获取手段多样、数据规模巨大、更新频率快、数据应用现势性强。传统的空间数据集中,单一的存储方式不能满足大数

收稿日期:2015-04-17
项目资助:国家自然科学青年基金(51309210)。
第一作者:李绍俊,博士生,主要从事地理信息系统软件研究。lishaojun@supermap.com
通讯作者:杨海军,博士,高级工程师,主要从事生态环境遥感应用研究。yanghj@lreis.ac.cn

据高并发甚至高时效的应用要求，空间数据从海量特征转变为大数据特征对存储技术的要求主要表现如下。

1）存储对象的变化。关系型数据库不擅长处理大量位置相关的视频、音频、图片等数据。

2）存储能力的扩展。在容量上，传统关系型数据库难以维护动辄千万级别的二维表，且存储能力的横向扩展也非常困难。在性能上，由于需要维护数据的完整性、一致性，数据存储的性能受到很大影响。

3）并发访问能力。关系型数据库区别传统文件的最大优势在于其对数据的多用户并发访问能力，但在云服务、互联网等应用领域，用户的并发量要求是关系型数据库无法满足的。

1.2 基于NoSQL的空间数据单一存储模式优势与不足

在互联网领域，基于非关系型数据库的NoSQL数据库技术已经得到成功应用。从数据存储的角度来说，NoSQL数据库非常适合空间大数据的存储，但互联网领域的应用模式与GIS领域存在一定差异，由此产生的局限性主要包括如下方面。

1）数据操作方式的局限性。在空间数据库中对数据的修改是常用操作，但NoSQL数据库一般不建议对数据进行修改，不慎重的修改操作甚至会因为引起相关存储数据的大量迁移而导致性能急剧下降。

2）数据查询方式的局限性。基于空间数据的各种专题图展示能力以数据库的查询能力为重要基础，需要按图层的属性信息提取数据，因此基于数据库的按字段查询、排序、统计等功能是重要基础。NoSQL数据库提供的查询能力非常有限，MongoDB也不能满足GIS常用查询的需要。

3）单一空间索引算法或技术的局限。类似关系型数据库，空间数据的索引技术是空间数据存储技术的重要研究内容[6-9]，但有一定局限性[10-14]。在大数据应用环境下，应该考虑把空间索引技术从算法层面提升到方法策略层面，才能解决空间数据的高效检索问题。

本文综合NoSQL数据库和关系型数据库各自的优势，提出一种混合的空间大数据分布式存储策略，既满足大数据存储的需要，又能满足传统GIS应用的需求。

2 空间大数据分布式存储策略

本文提出空间大数据分布式存储策略是针对传统的矢量、栅格及各种移动设备产生的实时/近实时的流式大数据存储与管理，设计基于内存数据库、关系型数据库、NoSQL数据库的三层物理存储体系及数据调度系统。

2.1 空间大数据存储与管理系统

面向空间大数据的分布式存储与管理系统具备存储和管理矢量、栅格和流式大数据的能力，整个系统能够对外提供多种能力，既包括实时/近实时的流式数据的存储和管理，又能对分布式数据库中的数据进行快速提取从而支持传统应用，如专题图展示、空间关联分析等。基于该系统的应用框架结构见图1。

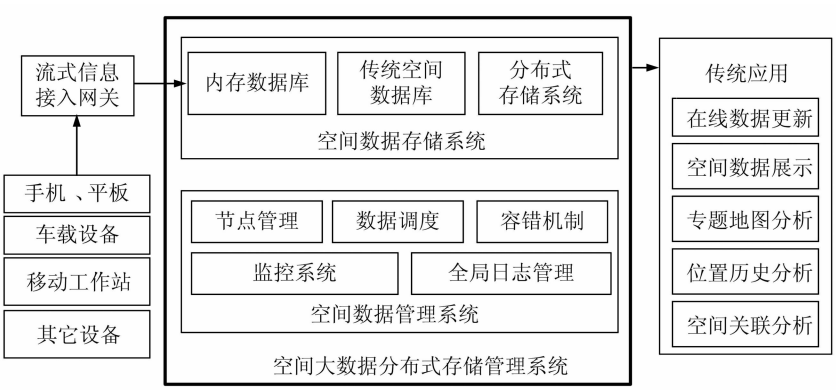

图1 空间大数据分布式存储管理系统应用框架图

Fig. 1 Application Diagram of the System of Spatial Big Data Storage and Management

空间数据存储子系统。综合内存数据库、传统空间数据库和NoSQL数据库，三者结合对外提供流式数据处理能力、传统空间数据库查询和分析能力及分布式系统大数据处理能力。

1）内存数据库，高效的数据操作层。I/O处理速度比传统数据库要快很多，一般都在10倍

以上。

2) 传统空间数据库,提供数据集中存储的能力,与此同时使得整个空间大数据分布式存储管理系统对外具备传统 GIS 的能力,保证整个系统对外提供的能力不退化。

3) 分布式存储系统,基于 NoSQL 数据库的分布式存储系统作为大数据仓库,为空间大数据提供低成本、高可扩展、高可用性、高通量 I/O 的持久化存储和数据提取能力。

2.2 关键技术

1) 空间数据在 NoSQL 数据库中的存储实现。本文主要研究 MongoDB 的数据库存储特点。MongoDB 是一种基于文档的 NoSQL 数据库,与空间数据库中的各类对象的映射关系见表 1。

表 1　各数据库中空间数据存储组织方式

Tab. 1　Object Types Mapping in Database

	SQLite:Mem	PostgreSQL	MongoDB
空间位置信息存储格式	Text	bin	BSON
数据源对应的数据库对象	database	database	database
图层对应的数据库对象	table	table	collection
空间对象对应的数据库对象	row	row	document

2) NoSQL 数据库中空间大数据的组织结构设计。本文主要讨论 MongogDB 的 Replica Set 集群和 Sharding 集群。在实际应用中,应用场景会极大地影响数据存储策略。本文的实验场景中,通过多次优化验证,采用的是根据地理范围进行外部分片,再搭建 Sharding 集群,可称之为滞后 Sharding 集群。

3) NoSQL 数据库中空间大数据的快速提取技术。除空间数据的组织结构会影响数据的提取效率,空间索引策略也是提高数据访问效率的关键点。本文设计的空间索引策略充分与集群方案结合,是基于元数据的多级图幅索引。

4) 数据合理调度问题。考虑空间大数据高效分析和处理的需要,其存储需要设计内外存协同的存储模式。将高频率访问热点数据直接内存于数据库中,实现高通量 I/O;其余海量的低频率"冷"数据以归档形式存储在关系型数据库或者 NoSQL 数据库中。

5) 空间数据访问接口设计。对关系型数据提供的 API(application programming interface)基于开源的 OGDC(open geospatial database connectivity)[15],类似于空间数据库领域的 ODBC,提供最基础的空间数据库访问接口,包含了多种关系型数据库的驱动程序,如 Oracle、DB2、MySQL、DM、Kingbase 等;对分布式存储系统,本文基于 OGDC 接口进行完善,为数据的并行存取提供通道。

3　空间大数据分布式存储实践

本文分别选取 SQLite、PostgreSQL 和 MongoDB 作为内存数据库、关系型数据库和 NoSQL 数据库的存储实践实现原型系统,重点解决空间大数据在 MongoDB 中的底层存储和上层逻辑组织结构等关键技术,实现大数据仓库中的空间数据快速存储与提取,对本文的存储策略进行验证。

实验选取空间大数据入库和空间叠加统计分析两个场景对本文提出的存储策略进行对比和验证。

实验数据采用全国大范围土地覆盖测试数据,共 10 101 幅 shape 文件,数据量 5.4TB,约 11 亿个空间面对象。

实验主要硬件为三台 32 核、内存 128GB 的服务器挂载磁盘存储;操作系统为 64 位 RedHat。

3.1 原型系统存储体系结构

空间大数据存储原型系统分三层结构,包括数据库层、中间件层和数据访问接口层,见图 2。

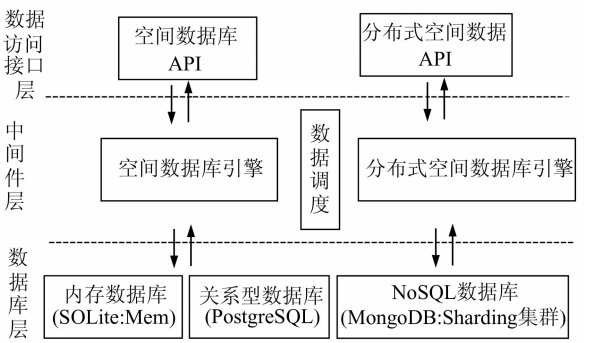

图 2　分布式存储系统体系结构图

Fig. 2　Distributed Storage Architecture of Spatial Big Data

1) 数据库层。SQLite 是一个跨平台的开源小型桌面数据库,提供全内存模式;PostgreSQL 是一个传统类型的开源关系型数据库;MongoDB 是目前为止与关系型数据库最接近的 NoSQL 数据库,实验搭建 MongoDB 的 Sharding 集群。

2) 中间件层。属于核心部分,包括传统的空间数据库引擎、分布式空间数据库引擎和数据调度子模块。传统的空间数据库引擎主要负责空间

数据在关系型数据库中的存储和管理，由于SQLite是关系型数据库，SQLite:Mem和PostgreSQL的存储处理方式基本类似；分布式空间数据库引擎负责空间大数据在MongoDB中的组织。各数据库中数据的组织方式见表1。

关系型数据库中，图层与表对应。矢量图层的每个对象对应表中的一行，这一行数据就包括了位置信息和属性信息，对于MongoDB来说，每个对象包装成一个带有空间信息和属性信息的BSON对象。

数据调度主要是基于数据操作接口，在不同的数据库之间进行数据的输入和提取。

3）数据访问接口层。主要为上层应用提供操作存储于不同数据库的空间数据的接口。

3.2 空间大数据入库实验

实验目标：将5.4TB shape数据全部导入MongoDB，并建立空间索引。

实验目的：对本文空间大数据分布式存储方案、空间索引策略和内存数据库的应用进行验证。

1）数据分布式存储方案

数据入库主要是对数据集合（记为collection）写操作，由于MongoDB采用的是数据库级锁，过多并发对同一数据库（记为database）中同一集合或不同集合的插入操作都会导致锁争用而产生性能瓶颈；通过Sharding集群的实例节点写入数据，由于节点间通信及维护开销，单个shape导入的性能比单节点写入性能慢2～3倍。为提高入库效率，本实验放弃对单个collection的Sharding集群存储，将研究区域的数据进行划分，存储到两个存储节点的6个database中，每个database包含多个collection，每个collection包含200万～300万个对象（见图3）。图3中，A1、A2；B1、B2；C1、C2等为研究区域的数据分片，DB_A1、DB_A2、DB_B1、DB_B2、DB_C1、DB_C2等为对应的存储节点数据库，Shard Node1、Shard Node2为存储节点。集群搭建采用滞后方式，即两个存储节点先作为单独实例启动，数据入库完成后再搭建Sharding集群。实际环境中也建议在数据入库完成后再建立各Sharding集群结点的备份结点，搭建数据副本-分片集群（Replica Set-Sharding）。

本实验环境下，对单个数据库级别的database开启10进程并发持续写入能够获得较好的效果。

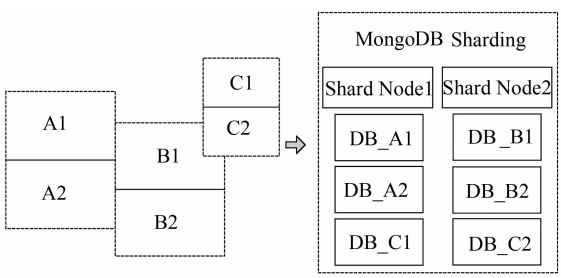

图3 基于MongoDB的分布式空间数据存储方案

Fig. 3 Spatial Database Storage in MongDB Sharding

2）基于元数据的多级空间索引策略

通过元数据表记录不同存储粒度的地理范围：存储结点级别（Shard Node）、数据库级别（database）和图层级别（layer）。图层的数据粒度为300万～400万对象，用图幅索引进行划分，即每个对应一张collection。在存储结构上，layer对应的database.collection的Tile_ID（图幅标识）列记录空间对象所在的图幅号，索引表的结构如图4(b)所示。其中各变量满足以下条件：

① $\sum_{i=0}^{k} n_i = N$，N为对象总个数；

② $t_i \in T$，$T = \bigcup \text{Obj\_ID}_{\text{Tile}}$，$\text{Obj_ID}_{\text{Tile}}$为索引表对象Obj_ID

N:1

Obj_ID	空间对象	Tile_ID
1	x_{1i}, y_{1i}	t_1
2	x_{2i}, y_{2i}	t_2
3	x_{3i}, y_{3i}	t_3
...	x_{mi}, y_{mi}	t_i
n	x_{ni}, y_{ni}	t_k

(a) 空间数据图层

Obj_ID	图幅范围	Obj_Count
1	MBR_1	n_1
2	MBR_2	n_2
3	MBR_3	n_3
...	MBR_i	n_i
t_k	MBR_k	n_k

(b) 空间数据索引图层

图4 图幅索引的数据表

Fig. 4 Index Collection of the Tile Index

3) 内存数据库的应用

入库过程中对图层建立图幅索引需要多次对 collection 进行 ID 查询,统计地理范围极值等操作,而 MongoDB 并不直接提供对列求极值的操作算子,需要遍历数据。本实验利用内存数据库,实现了单个 shape(要素)入库处理时间不随目标层数据量增加而增长的目标。步骤如下。

(1) 单个要素(shape)导入内存数据库得到 MemLayer;

(2) 对 MemLayer 切分图幅 $T_1, T_2, \cdots T_n$;

(3) 将 n 图幅数据追加至目标图层。对第 T_i 图幅数据,获取数据范围、对象数,获取目标数据集的新图幅标识(Tile_ID),写入目标图层。

数据写入完毕,实际上图层的图幅索引也建立完毕。本文统计 3 个图层约 60 个 shape,每个 shape 约 500MB,11 万个对象,单个 shape 处理的平均时间见表 3。实验中整个入库过程 shape 处理速度基本平稳。

表 2　单个要素平均处理时间

Tab. 2　Average Time Consumption for One Shape File

	耗时/s	备注
导入内存数据库	20.6	-
内存图层切分图幅	2.3	12 列×3 行
追加到目标图层	46.7	-
总计	69.6	-

4) 实验结果

基于以上方案,对 5.4TB 的矢量数据采用 2 个存储节点、60 进程并发入库并建立空间索引的总时间耗费为 3 h 42 min,有效地保障了海量数据高并发环境下的入库效率。

3.3　叠加分析统计实验

实验目标:从 MongDB 中提取安徽省相关数据,并对安徽省各县市土地覆盖类型进行分类统计。参与叠加分析的安徽省行政区划图层为 L_AnHui,包含 105 个面对象。

实验目的:对本文的空间索引策略的有效性进行验证;对本文的内存数据库、关系型数据库和分布式数据库的协同应用策略进行验证。

实验步骤分为数据提取到关系型数据库 PostgreSQL 和基于关系型数据库的叠加分析统计。

1) 数据提取过程

充分利用基于元数据的空间索引,提取图幅粒度的相关数据输入 PostgreSQL 数据库。

(1) 根据 L_AnHui 得到矩形查询范围 rcBounds;

(2) 查询元数据表找到与查询范围(rcBounds)相关的 layer:$L_1, L_2, \cdots, L_n$。数据入库后,存储结点、数据库和图层级别的元数据总计 354 条,因此对元数据的范围查询时间在 ms 级别;

(3) 对图层 L_i 查询 rcBounds 相关的图幅 $T_{i1}, T_{i2} \cdots T_{im}$;

(4) 取图层 L_i 的 $T_{i1}, T_{i2}, \cdots, T_{im}$ 图幅的数据,采用图幅追加的方式写入 PostgreSQL。

各步骤时间统计见表 3。与安徽省相关的图层共计 8 个,这 8 个图层中再次过滤得到3 861个图幅。最终分 8 个图层启用 8 个进程写入 PostgreSQL,总对象数 12 574 739。

2)叠加分析统计

在 PostgreSQL 中对提取的土地覆盖数据与 L_AnHui 进行叠加裁剪,并按土地覆盖类型字段 CType 对各县市进行分类统计求和。假设 L_AnHui 中的对象为 $A_1, A_2, \cdots, A_n$,土地覆盖数据各 layer 为 $L_1, \cdots, L_m$,对于 A_i 和 L_j 计算步骤如下。

表 3　数据提取时间记录

Tab. 3　Data Extraction Bases on Spatial Index

	耗时/s	备注
元数据查询	0.6	-
图层内图幅查询	2.3	8 个图层
图幅数据追加	978	3 861 个图幅
总计	981	12 574 739 对象

(1) 通过 L_j 的图幅索引,提取被 A_i 完全包含的图幅 TID_in 和相交的图幅 TID_intersect。通过图幅 TID_in 取对象面积以及 CType 值,得到结果 R_in1;

(2) 对于 TID_intersect 提取各图幅中空间对象的四至,与 A_i 叠加:被 A_i 包含的取出对象面积及 CType 值,写入结果 R_in2;相交的对象写入内存数据库图层 Mem layer 等待计算;

(3) 在内存数据库中对 Mem layer 和 A_i 进行叠加裁剪(线段打散→求交→拓扑构面),并计算裁剪后的对象面积,得到结果 R_clip;

(4) R_in1、R_in2 和 R_clip 结果合并。

该过程通过逐级过滤,尽量过滤掉不需要进行叠加裁剪的对象,并将裁剪过程迁移至内存数据库,减少计算过程中对数据库的频繁访问,提高计算效率。

表 4 对比了是否采用内存数据库的测试结果,实验中实际参与叠加计算的对象总数为 182 947 个。

3) 实验结果

本实验中,基于元数据的空间索引能够快速过滤定位到相关图幅,从MongoDB中提取安徽省相关数据并输入PostgreSQL的耗时主要用在数据的I/O;叠加分析和统计过程中采用内存数据源减少了与数据库的交互,有效提高了空间分析的效率。

表4 各图层叠加统计耗时

Tab. 4 Time Consumption on Overlay Analysis

图层	图幅数	对象数	耗时A/s	耗时B/s(内存数据库)
L1	73	253 454	192	126
L2	385	1 264 189	643	397
L3	656	2 159 563	1072	658
L4	656	2 111 632	1123	744
L5	658	2 176 057	1070	723
L6	762	2 475 838	1119	691
L7	581	1 879 148	649	419
L8	90	254 858	215	141

4 结 语

面向大数据的空间数据分布式存储策略结合了传统的基于关系型数据库的GIS基本能力和NoSQL数据库提供的大数据分布式存储能力,既满足了大数据存储的需要,又保证了整个系统对外提供的GIS功能不退化。本文基于SQLite、PostgreSQL和MongoDB数据库,开发了原型系统,并选取了两个实验场景对该存储策略的有效性进行了验证。空间大数据的存储和管理是一个相对复杂的问题,根据应用的实际情况,大数据的空间索引策略、数据分解方法、数据调度策略等都值得进一步深入研究。

参 考 文 献

[1] Mooney P, Corcoran P, Winstanley A C. Geospatial Data Issues in the Provision of Location-based Services[C]. Proceedings of the 7th International Symposium on LBS & Telecartography, Guangzhou, China, 2010

[2] Gong Jianya. Concepts and Development of Spatial Database Management System[J]. *Science of Surveying and Mapping*, 2001, 26(3):4-9(龚健雅. 空间数据库管理系统的概念与发展趋势[J]. 测绘科学, 2001, 26(3):4-9)

[3] Liu Jingnan, Fang Yuan. Research Progress in Location Big Data Analysis and Processing[J]. *Geomatics and Information Science of Wuhan University*, 2014, 39(4):380-385(刘经南, 方媛. 位置大数据的分析处理研究进展[J]. 武汉大学学报·信息科学版, 2014, 39(4):380-385)

[4] Zhou Qin, Li Shaojun. Study on Spatial Data Cache Technology Based on Oracle Spatial[J]. *Geo-Information Science*, 2007, 9(3):39-44(周芹, 李绍俊. 基于Oracle Spatial的空间数据库缓存技术研究[J]. 地球信息科学, 2007, 9(3):39-44)

[5] Zhou Qin, Li Shaojun, Li Yunjin, et al. The Key Technique and Development of Spatial Database Engine[C]. The Fourth Member Representative Assembly of China Geographic Information System Association, Beijing, China, 2007(周芹, 李绍俊, 李云锦, 等. 空间数据库引擎的关键技术及发展[C]. 中国地理信息系统协会第四次会员代表大会, 北京, 2007)

[6] Zhong Y, Han J, Zhang T, et al. A Distributed Geospatial Data Storage and Processing Framework for Large-scale WebGIS[C]. The 20th International Conference on Geoinformatics, Hong Kong. China, 2012

[7] Han D, Stroulia E. HGrid: A Data Model for Large Geospatial Data Sets in HBase[C]. Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, CA. USA, 2013

[8] Wei L Y, Hsu Y T, Peng W C, et al. Indexing Spatial Data in Cloud Data Managements[J]. *Pervasive and Mobile Computing*, 2014, 15: 48-61

[9] Chen Chongcheng, Lin Jianfeng, Wu Xiaozhu, et al. Massive Geo-spatial Data Cloud Storage and Services Based on NoSQL Database Technique[J]. *Journal of Geo-Information Science*, 2013, 15(2): 166-174(陈崇成, 林剑峰, 吴小竹, 等. 基于NoSQL的海量空间数据云存储与服务方法[J]. 地球信息科学学报, 2013, 15(2):166-174)

[10] Chang F, Dean J, Ghemawat S, et al. Bigtable: A Distributed Storage System for Structured Data[J]. *ACM Transactions on Computer Systems*, 2008, 26(2):1-26

[11] Ghemawat S, Gobioff H, Leung S T. The Google File System[C]. 19th ACM Symposium on Operating Systems Principles, New York, USA, 2003

[12] Burrows M. The Chubby Lock Service for Loosely-coupled Distributed Systems[C]. Proceedings of the 7th Symposium on Operating Systems Design and Implementation, Berkeley, USA, 2006

[13] Chen Jirong, Le Jiajin. Reviewing the Big Data Solution Based on Hadoop Ecosystem[J]. *Computer Engineering & Science*, 2013, 35(10):25-35(陈吉荣, 乐嘉锦. 基于Hadoop生态系统的大数据解决方案综述[J]. 计算机工程与科学, 2013, 35(10):25-35)

[14] Hecht R, Jablonski S. NoSQL Evaluation: A Use Case Oriented Survey[C]. 2011 International Con-

ference on Cloud and Service Computing, Hong Kong, China, 2011

[15] Li Shaojun, Zhong Ershun, Zhou Qin, et al. Study on Opening Geospatial Database Connectivity[J]. *Journal of Geo-Information Science*, 2013, 10(2): 193-199(李绍俊,钟耳顺,周芹,等. 开放式空间数据库访问接口的开发应用[J]. 地球信息科学学报, 2013, 10(2):193-199)

Geo-spatial Big Data Storage Based on NoSQL Database

LI Shaojun[1]　*YANG Haijun*[2]　*HUANG Yaohuan*[1]　*ZHOU Qin*[3]

1　Institute of Geographic Sciences and Natural Resources Research, CAS, Beijing 100101, China

2　Satellite Environment Center, Ministry of Environmental Protection, Beijing 100094, China

3　Beijing Supermap GIS Technologies Company, Beijing 100015, China

Abstract: Geospatial data in databases have shifted to conform to the characteristics of big-data in tandem with the development of the Internet, mobile Internet, cloud computing, and especially, spatial data acquisition technologies. Faced with tackling spatial big data, traditional spatial database management techniques based on Relational Database Management Systems have encountered problems including the unstructured characteristics of the spatial object, the high scalability of storage capacity, and the high concurrency in big data application environment. This paper focuses on the mainstream of NoSQL databases that successfully deal with unstructured big data and are widely used in Internet applications, but lack of spatial characteristics. The data operational and query modes cannot meet the requirments of GIS applications. To resolve this problem, this paper proposes a strategy that takes a NoSQL database as a warehouse for spatial big data and a traditional spatial database as the application server. The storage system architecture and the key technology and solutions are discussed. A prototype system was developed based on MongoDB, PostgreSQL and SQLite to verify the feasibility and effectiveness of the strategy.

Key words: spatial database; big data; NoSQL database; distributed storage

First author: LI Shaojun, PhD, specializes in geographic information software. E-mail:lishaojun@supermap.com

Corresponding author: YANG Haijun, PhD, senior engineer. E-mail: yanghj@lreis.ac.cn

Foundation support: The National Natural Science Foundation of China, No. 51309210.